UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

Automated Analysis of Cryptographic Constructions

PH.D THESIS

Miguel Ambrona Castellanos
MSc in Mathematical Engineering

2018

DEPARTAMENTAMENTO DE LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERIA DE SOFTWARE

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

Automated Analysis of Cryptographic Constructions

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF:
Doctor of Philosophy in Software and Systems

Author: **Miguel Ambrona Castellanos**
MSc in Mathematical Engineering

Advisor:
Prof. Gilles Barthe

September 2018

Thesis Committee:

Chair	
Manuel Hermenegildo Salinas	*Full Professor* Universidad Politécnica de Madrid (UPM) Spain
Secretary	
Dario Fiore	*Assistant Research Professor* IMDEA Software Institute Spain
Members	
Manuel B.M. Barbosa	*Assistant Professor* Universidade do Porto Portugal
Ángel L. Pérez del Pozo	*Visiting Professor* Universidad Rey Juan Carlos (URJC) Spain
Mehdi Tibouchi	*Distinguished Researcher* NTT Secure Platform Laboratories Japan
Substitute Members	
María Isabel González Vasco	*Associate Professor* Universidad Rey Juan Carlos (URJC) Spain
Claudio Soriente	*Senior Researcher* NEC Laboratories Europe GmbH Germany

Abstract of the Dissertation

Computer-aided cryptography is an emerging area of research concerned with the application of formal methods and techniques from automated reasoning in cryptography. It allows cryptographers to outsource tedious or error-prone tasks to computers (including mundane parts of the analysis and validation of cryptographic proofs, verifying complex calculations or exploring design spaces of cryptographic constructions). Computer-aided cryptography has been successfully applied to the analysis of several primitives in classic public-key systems and symmetric schemes. However, it is still not clear how automated methods can be applied to more recent and advanced constructions. In this thesis we develop new techniques and tools to broaden the scope of computer-aided cryptography, with special emphasis on pairing-based cryptography.

Our first contribution consists of a novel method (and an implementation) for automatically checking the security of cryptographic schemes (such as structure-preserving signatures, message authentication codes or assumptions defined over bilinear groups) in the generic group model, under strong standard definitions of security. Our work improves on previous works, which consider weaker security models. Our new approach allows us to reduce the security of schemes to the absence of solutions to a system of constraints. We then develop dedicated constraints-solving algorithms for testing that the systems are unsatisfiable.

Our second contribution focuses on attribute-based encryption (ABE). More precisely, we present several new results about predicate encodings, a cryptographic primitive that can be used to build ABE in a modular way. We propose a purely algebraic formulation of the notion of privacy for predicate encodings, that leads to several new applications, such as logical combinations of predicate encodings and optimization techniques that resulted in improved ABE constructions (with extra features and better performance).

Our third contribution builds on the first two. We provide, for the first time, computer-aided techniques and algorithms (plus an implementation) for automatically proving the security of attribute-based encryption constructions in the generic group model. Our method allows us to deal with indistinguishability-based security definitions (as is required for ABE), which is a very important achievement. Previous works on automated analysis mainly focused on computational security experiments instead.

As a last contribution we also study indifferentiability of symmetric constructions (such as different variants of Feistel networks or Even-Mansour ciphers). Unlike in our previous contributions, where our analysis was oriented towards proof search, here we focus on formalizing and synthesizing attacks. We develop methods for fully automated attack search. First, we formally define the notion of universal indifferentiability distinguishers and provide methods for proving the universality of candidate distinguishers. Then, we develop (and implement) heuristics that take the description of a cryptographic component and try to find a universal distinguisher for it.

All our contributions share a common methodology: we leverage techniques from formal methods, expressing cryptographic constructions and security definitions in a symbolic language. We then provide computational soundness theorems for such symbolic models that guarantee that the conclusions derived symbolically can be lifted to the actual non-symbolic model. We demonstrate the effectiveness of our approaches by developing several tools that implement our techniques, which are then evaluated on a wide range of examples from the literature. The results presented in this thesis expand the scope of computer-aided cryptography, capturing stronger security notions and new settings.

Resumen de la Tesis Doctoral

La criptografía asistida por ordenador es un área de investigación cada vez más popular que estudia la aplicación de métodos formales y técnicas de análisis automatizado a la criptografía. Permite que los criptógrafos deleguen en un ordenador algunos cálculos tediosos y propensos a errores (por ejemplo, partes rutinarias del análisis y validación de pruebas criptográficas, verificar cálculos complejos o explorar diferentes variantes de construcciones criptográficas). La criptografía asistida por ordenador ha sido utilizada con éxito para el análisis de varias primitivas, tanto en sistemas de clave-pública clásicos, como en sistemas de clave simétrica. Sin embargo, todavía no está muy claro cómo se podrían aplicar dichos métodos automatizados a las construcciones más recientes y avanzadas. En esta tesis se desarrollan nuevas técnicas y herramientas que amplían el alcance de la criptografía asistida por ordenador, con especial énfasis en la criptografía basada en *pairings*.

La primera contribución se trata de un nuevo método (con su correspondiente implementación) para comprobar la seguridad de esquemas criptográficos (como *structure-preserving signatures*, *message-authentication codes* o *asunciones* definidas sobre *bilinear pairings*) de forma completamente automática y garantizando niveles de estandarizados de seguridad. Primero, nuestro nuevo método estudia cómo reducir la seguridad de dichos esquemas a la ausencia de soluciones a sistemas de restricciones. Después, desarrollamos técnicas para resolver dichos sistemas y demostrar que no admiten ninguna solución.

La segunda contribución se centra en *attribute-based encryption* (ABE). Más concretamente, presentamos nuevos resultados sobre *predicate encodings*, una primitiva criptográfica que se utiliza para construir ABE de manera modular. Proponemos una caracterización puramente algebraica de la noción de privacidad de los *encodings*, que da lugar a diversas nuevas aplicaciones: combinadores lógicos de *predicate encodings* y técnicas de optimización que han resultado en construcciones de ABE mejoradas (con nuevas propiedades y mejor eficiencia). Además, nuestra caracterización hace posible nuestros siguientes resultados.

La tercera contribución construye sobre las dos primeras. Proponemos, por primera vez en criptografía asistida por ordenador, técnicas y algoritmos (además de una implementación) para demostrar automáticamente la seguridad de construcciones ABE en el *generic group model*. Nuesto método permite estudiar experimentos de seguridad basados en *indistinguibilidad* (como require ABE), lo que supone un logro muy importante. Los trabajos anteriores sobre análisis automático se centraban especialmente en experimentos de seguridad *computacional*.

Como última contribución también estudiamos la noción de *indiferenciabilidad* en construcciones de clave simétrica (como por ejemplo, diferentes variantes de *Feistel networks* o *cifrados Even-Mansour*). Al contrario que en nuestras primeras contribuciones, donde nuestro análisis estaba orientado a la búsqueda de demostraciones, aquí nos centramos en formalizar y sintetizar ataques. Desarrollamos métodos para automatizar la búsqueda de ataques. Primero, definimos formalmente el concepto de distinguidor universal y proporcionamos métodos para probar la universalidad de distinguidores candidatos. Después, desarrollamos e implementamos heurísticas que toman como entrada la descripción de una componente criptográfica y tratan de encontrar un ataque universal.

Todas nuestras contribuciones comparten una metodología común: utilizamos técnicas de la teoría de métodos formales, expresando las construcciones criptográficas y sus definiciones de seguridad en un lenguaje simbólico. A continuación, proporcionamos teoremas que garantizan su *validez computacional* de forma que las conclusiones obtenidas simbólicamente se puedan extrapolar al modelo real sin símbolos. Además, demostramos la efectividad de nuestros métodos a través de varias herramientas que implementan nuestras técnicas y que han sido evaluadas en un amplio conjunto de ejemplos de la literatura. Los resultados presentados en esta tesis amplían el alcance de la criptografía asistida por ordenador, permitiendo el análisis de nuevas primitivas y garantizando mejores nociones de seguridad.

A mi hermano, mi madre y mi padre

Acknowledgements

Me gustaría mostrar mi agradecimiento a todas las personas que me han ayudado de alguna forma a haber logrado terminar mi tesis doctoral. A todas las que me han apoyado, enseñado, inspirado, animado, divertido, distraído, y ofrecido su tiempo e interés, aunque no mencione a todo el mundo, no me olvido de vosotros.

Haber llegado hasta aquí, en parte, se lo debo a la persona que me dio la idea de hacer un doctorado. Mariemi, muchas gracias por haberme animado a hacerlo y por todo tu interés, apoyo y confianza en mí. Gracias, Gilles, por haberme dado esta oportunidad de aprender tanto en un entorno tan estimulante, por haberme enseñado a trabajar de forma independiente y por tu confianza. Benedikt, thank you for all your help, especially during the beginning of my PhD, without you, it would have been impossible. Dario, muchas gracias también a ti, por toda tu ayuda durante estos años. 正幸さん, thank you for giving me the opportunity of working in such an amazing environment, NTTは凄い！ Thanks a lot for all your time and for having taught me so much. And thank you too, 美也子さん, for so many nice discussions and all your time and help, 本当にありがとうございます，とても嬉しいです。 I also want to thank all my friends from NTT Laboratories for being so nice with me. Especially Avic and Hridam, thanks for so many good moments together. Thanks, Mehdiさん, for your help and for showing us the best of Tokyo gastronomy. And of course, thank you so much, 佐穂さん, for all your help, 最高の秘書です！ Thank you, Aishwarya, for all your energy and good mood, my first conference trip would not have been the same without you. Obviously, undoubtedly, I totally appreciate remembering every unexpected moment past, Itsaka, seriously, ¡son extremadamente divertidos! estuvo bien, ocurrente, único, ¡*taré*! Thank you to all my co-authors, especially Romain, for our fruitful discussions and the nice moments together. Muchas gracias a todos mis compañeros de IMDEA, por todos los buenos momentos que hemos pasado juntos y por todo vuestro apoyo en momentos difíciles. A Ignacio y Antonio, por vuestros útiles comentarios sobre cómo mejorar la introducción de este documento. Y en especial a áquellos con los que más tiempo he compartido: Dr. Nappa, Горан, Damir, Miriam, Sergio, Артем, Наталя, Platon, Pedro, Álex.

Luca, ti devo ringraziare per tante cose: per essere stato il migliore compagno di ufficio che si può avere (tranne che per i coppini quando ero concentrato), per insegnarmi la tua lingua, per avere avuto tanta pazienza nel rispondere alle mie domande e per avermi aiutato con tutti i miei problemi. Chana, ha sido un placer conocerte y determinante para el final de mi doctorado, je te dois une fière chandelle! Muchas gracias por todo tu tiempo, ilusión, por toda tu energía y por contagiarme de ella, *tu es ouf!* Muchas gracias a mis amigos de ajedrez, por tantos buenos momentos de distracción y diversión y en especial a Ricardo, gracias por tu buen humor y tu ilusión. A mis amigos de la Casa de la Moneda, por todo vuestro interés y singular humor, especialmente a Luis, muchas gracias por todo lo que me has enseñado y por todo tu interés y a Andrés, por seguir interesándote y compartiendo frikadas. A mis mejores amigos de la universidad, Santi, Álex, María, Carlos, gracias por vuestro interés y por haber compartido tantos buenos momentos conmigo en los últimos años. Gracias a mis mejores amigos de siempre, Antonio y Marta, me gusta seguir sabiendo de vosotros y que hayáis estado siempre ahí.

Irene, tú eres sin duda la persona que más me ha ayudado a sacar este proyecto adelante. Gracias por todo tu cariño, tu tiempo, por haber compartido todo conmigo, por haberme animado en los momentos de desmotivación y por nunca dejar que me rindiera. Por preocuparte a veces tú más que yo por mi propia tesis y por que lo siguiera intentando. Haber terminado esta tesis te lo debo a ti, muchas gracias de corazón.

Elena, muchas gracias por toda tu ayuda en estos últimos meses de mi doctorado, no sé qué habría hecho sin ti... Muchas gracias por todo el tiempo que hemos compartido, por preocuparte tanto, por ser tan buena conmigo, porque eres la persona más optimista que conozco (o que existe) y la más importante para mí. Gracias por contagiarme de tus ganas de vivir. Me siento muy agradecido y muy afortunado por haberte conocido.

Me gustaría terminar dándome las gracias a mí, por no haberme rendido nunca. Y a toda mi familia, por haberse interesado siempre y haberme apoyado en todo momento. Y por último, a las tres personas a las que dedico este trabajo: mi hermano, mi madre y mi padre. Por su apoyo *incondicional*, una de las cosas más valiosas que existen.

Contents

List of Figures

List of Tables

List of Publications

This thesis comprises four papers. The first three have been published in top peer-reviewed academic conferences, while the fourth one is work in progress that will be submitted soon to a top conference (such as EuroCrypt or Crypto):

- *Automated Unbounded Analysis of Cryptographic Constructions in the Generic Group Model*
  Miguel Ambrona, Gilles Barthe, Benedikt Schmidt.
  In Proceedings of EuroCrypt 2016. [18]

- *Generic Transformations of Predicate Encodings: Constructions and Applications*
  Miguel Ambrona, Gilles Barthe, Benedikt Schmidt.
  In Proceedings of Crypto 2017. [19]

- *Attribute-Based Encryption in the Generic Group Model: Automated Proofs and New Constructions*
  Miguel Ambrona, Gilles Barthe, Romain Gay, Hoeteck Wee.
  In Proceedings of ACM CCS 2017. [17]

- *Automated synthesis of indifferentiability attacks*
  Alejandro Aguirre, Miguel Ambrona, Gilles Barthe, Serdar Erbatur.

I contributed in the elaboration of all of them as the main author. Every work comes with an implementation of the methods and techniques described in it. I am the main developer of all these tools and they are publicly available and open-source.

The source code corresponding to every work can be found respectively in:

- https://github.com/generic-group-analyzer/gga-unbounded
- https://github.com/miguel-ambrona/abe-relic
- https://github.com/miguel-ambrona/ggm-symbolic-solver
- https://github.com/miguel-ambrona/indiff

Other conference articles co-authored during the term of my PhD grant, but not included in this dissertation:

- *Lower Bounds on Structure-Preserving Signatures for Bilateral Messages*
  Masayuki Abe, Miguel Ambrona, Miyako Ohkubo, Mehdi Tibouchi.
  In Proceedings of SCN 2018. [1]

- *Signature Schemes from SS-NIZK Arguments for any Language without Language Extension*
  Masayuki Abe, Miguel Ambrona, Miyako Ohkubo.
  Under submission (EuroCrypt 2019).

1 Introduction

Rerum omnium magister usus.
(Experience is the teacher of all things.)
Julius Caesar, BC 40

Computer assistance is an emerging practice in the design and verification of cryptographic primitives and protocols. Automated analysis contributes favorably to the development of cryptography and it is feasible in an increasing number of subareas of this discipline.

In this thesis, we advance its feasibility in the framework of selected emerging cryptographic primitives such as Attribute-Based Encryption, Structure-Preserving Signatures and others.

1.1 Cryptography

Cryptography has been a very powerful tool for humans throughout history. One of the first documented uses of cryptography can be attributed to Julius Caesar [183], who used to apply a very simple transformation to the written commands he was transmitting by horseback messengers during the Gallic Wars. This technique prevented the enemy from making sense of the crucial information that was being transmitted in case the messenger was captured. However, the legitimate receiver of the directive knew how to interpret the apparent *nonsense*. The mentioned technique, known as the *Caesar Cipher*, is a very simple example of an *encryption scheme*.

Many other examples of encryption schemes have been proposed and utilized from ancient times: *Substitution Ciphers* (Caesar Cipher belongs to this class), that were vulnerable to frequency analysis; the *Vigenère Cipher* (formerly proposed by Giovan Battista Bellaso [56] in 1553), which was known as *the indecipherable cipher* because it remained unbroken for three centuries until a general method for deciphering it was discovered by Friedrich Kasiski in 1863; the *Enigma Machine*, used in the Second World War and whose cryptoanalysis gave rise to the birth of computers; and many others.

During centuries, cryptography has been all about encryption, as the main concern was to achieve *confidentiality*. Note that the etymology of "cryptography" evidences that fact (from Ancient Greek, *κρυπτός*, *kryptos* or *hidden*; and *γράφειν*, *graphein* or *to write*). Nowadays, cryptography has other purposes [79], trying to provide more functionalities such as *accountability*, *auditability*, *authenticity/trustworthiness*, *availability*, *integrity*, *non-repudiation* or *privacy*. All these services play an essential role in our lives. In fact, technology would not have been developed in the same way without the existence of cryptography. Over the last few decades, the human civilization has experienced an exponential progress, due to the massive demographic development and the birth and adoption of computer systems in our daily life. The Internet, one of the greatest advances in history, has changed the way we communicate, interact and *share knowledge*, accelerating research and the development of science in general. Nonetheless, many of the opportunities that these advances provide come with serious security concerns and would not be possible without cryptography: online-shopping or home-banking would be unfeasible through an insecure channel (like the Internet) without compromising our personal information and credentials; and many confidential communications between different parts of the world would not be possible, slowing down many advances and discoveries. Furthermore, cryptography is the meeting point of many disciplines (abstract algebra, probability theory, complexity theory, geometry, etc) and brings new challenges to all of them, favoring their development.

Nowadays, cryptography is treated in a much more rigorous way, through the so-called *provable-security* paradigm. This paradigm enforces some security requirements on the primitives, that are minutely studied, and the confidence about a specific construction is established through a formal proof of the fact that it meets such requirements. Moreover, modern cryptography follows the, very important, *Kerckhoffs's principle* [136], proposed in 1883, which establishes that the description of a cryptographic scheme should be publicly available and everything about the system should be public knowledge except the *secret key*: the only missing piece of information for a potential adversary, without which it is unfeasible to perform a successful attack against the primitive. This principle was reformulated by Shannon in 1949 [180]: "*one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them*".

In the past, the mentioned maxims were not considered and cryptography used to rely on *security by obscurity* (primitives were relying on the secrecy of their design, disregarding potential vulnerabilities or security flaws) or simply on the absence of a practical attack known by the designers of the system. This lack of formalism caused that every encryption scheme proposed throughout history was eventually broken, and this fact probably inspired the following words by the American writer (and cryptography lover) [166]

> *...it may be roundly asserted that human ingenuity cannot concoct a cipher which human ingenuity cannot resolve...*
>
> Edgar Allan Poe, 1841

Note that Poe's words are applicable to the modern rigorous approach to cryptography, which is not a complete guarantee of security:

- Modern cryptography relies on the hardness of certain well-studied problems (factoring large integers, computing the discrete logarithm over a finite group, lattice-based problems, etc) that are currently intractable. If these problems are assumed to be hard to solve, the security of cryptosystems can be proven under such *hardness assumptions*. However, the progress of computer science could produce new algorithms for addressing some of these problems in an efficient way. Moreover, the development of *quantum computers* [181] would threaten the hardness of many of these problems and cryptography in general.

- Even when a cryptographic system is proven to be theoretically sound, it might be susceptible to attacks when put into practice. Implementations are error-prone, engineers can make mistakes and unexpected vulnerabilities might appear. Furthermore, even when these implementations are tested and validated they could still be vulnerable to other exploits: *side-channel attacks* [95, 139, 140]. These attacks have been successfully used for breaking actual cryptographic implementations of theoretically sound algorithms by exploiting secret-dependent variations of non-functional properties such as timing, power consumption, electromagnetic leaks, etc.

For the sake of security and to prevent such unexpected attacks, cryptography needs to be constantly revisited and updated. Computer assistance plays an important role in assuring the security of cryptographic primitives and verifying the increasingly complex systems that our society demands.

1.2 Computer Assistance

Over the years, the design and analysis of cryptographic systems has been performed manually. Analyzing these systems to verify their validity and soundness is far from being a trivial task and pen-and-paper calculations may be error-prone, as witnessed by unfortunate failures [105, 128, 184].

Computer assistance has been successfully applied to many areas such as engineering and production (architectural and industrial design), languages (automated translation), medicine (detection, diagnosis, surgeries), biology (DNA sequencing, biotechnology), etc. In some of these disciplines, computer assistance plays an essential role, while in others it helps scientists to accomplish more complex achievements and facilitates several tedious tasks. If it has had so much success in so many areas, *why not applying computer assistance to the field of cryptography?*

Computer-aided cryptography [38] is an emerging approach that advocates using automated tools based on formal methods for analyzing the security and implementation of cryptographic schemes. The high level of assurance provided by computer-aided cryptography is particularly important for cryptographic schemes that are already deployed in real-world systems, such as RSA-OAEP [55, 108] or TLS [96], but also for schemes that are required in many applications and hold the promise of widespread deployment. One such example is provided by *Attribute-Based Encryption* (ABE) [115, 174], a novel form of public-key encryption (Public-Key Encryption). ABE supports fine-grained access control on encrypted data and has many applications, including electronic medical records [16], messaging systems [149], online social networks [31] information-centric networking [129], etc. These applications make ABE an ideal domain for computer-aided cryptography.

There exist two main paradigms in the framework of computer assistance:

Interactive Assistance. This paradigm is based on human-machine interaction for the development of formal proofs. It allows to formalize results for which the user has a clear idea of how the desired proof looks like, or an intuition about why the result to be proven is correct. The human-machine collaboration prevents the user from making mistakes or taking wrong proof steps. There exist many general purpose *proof assistants* (such as Coq or Isabelle) that allow to perform these tasks, usually with significant effort from the user. Some cryptographic primitives have been formalized in these systems, e.g., Barthe, Cederquist and Tarento [39] use Coq to build machine-checked proofs of security in the Generic Group Model (GGM). For another remarkable work on facilitating the analysis of cryptography under these proof assistants we refer to [165].

The tools developed as a result of this work can be used interactively, following this paradigm. Users can interact with the tools by introducing commands and visualizing the missing goals to be proven. However, our work mainly focuses on automated analysis.

Automated Analysis. Automated reasoning is an emerging practice in cryptography. It helps scientists in the design and analysis of cryptographic systems. *Full automation* is the ideal scenario, where the user just needs to describe the cryptographic primitive, which is then analyzed in a completely automated way. However, full automation comes at a price: the scope of the analysis needs to be limited to certain class of primitives, e.g., the class of Structure-Preserving Signatures (SPS) over bilinear groups. Research in this area tries to broaden the number of cryptographic schemes that can be handled by automation.

One of the main advantages of automated analysis is trustability. Automated methods and tools to prove the soundness and reliability of cryptographic systems can be thoroughly verified once and for all. Once the theoretical basis of these methods and its implementation is verified, they can be trusted and applied to several different instances. However, gaining the same level of confidence without automated methods would require a deep analysis for every individual instance.

Automated analysis is very versatile. New techniques for automated reasoning can be built on earlier existing systems that are sound and reliable. For example, new advances in computer-aided cryptography can benefit from existing tools like SMT solvers [37, 94] or First-Order Theorem Provers [170, 178]. On the other hand, improvements on the analysis of cryptographic constructions can be of independent interest in other areas of automated reasoning.

Automated analysis allows to approach the design of new cryptographic primitives from a synthesis-based perspective. Having exploration tools that generate several different cryptographic schemes (of certain class) varying iterative parts of their structure, combined with automated analysis tools can lead to the discovery of new cryptographic primitives with favorable properties. This approach has been applied in practice, an example is the work by Barthe et al. [44], where the authors explore synthesis in the framework of Structure-Preserving Signatures and they find a new re-randomizable SPS scheme that improves on existing schemes on size and efficiency. Building catalogs of cryptographic constructions with quantitative measures of their security and efficiency is an important direction of research.

Having tools for automatically deriving full or partial proofs is extremely useful for the design of new cryptographic primitives. Users can focus on the ingenious parts of the demonstration, which require *human thinking*, delegating other parts to the computer. This practice not only prevents users form making mistakes along the most tedious details of the argument, but also allows them to never lose the intuition behind the design, which can potentially lead to more complex or interesting achievements. There is a remarkable work by Halevi [123] where he discusses this idea of (informally) dividing proofs in two categories.

> *Most (or all) cryptographic proofs have a creative part (e.g., describing the simulator or the reduction) and a mundane part (e.g., checking that the reduction actually goes through). It often happens that the mundane parts are much harder to write and verify, and it is with these parts that we can hope to have automated help.*
>
> Shai Halevi, 2005

Furthermore, automated reasoning can be used to discover subtle flaws or attacks that are not easy to visualize by intuitive reasoning.

Another advantage of automated analysis is that it brings cryptography closer to non-experts. Cryptographic proofs are often technical and usually require a strong background on mathematics and logic. Automated analysis is a helper tool that broadens the community of researchers who can work in the field and investigate new cryptographic primitives and techniques, potentially leading to more efficient constructions and better quality discoveries.

1.3 Research Questions

Computers are devices that can be *instructed* to perform sequences of operations at a breakneck speed. Such a promising quality makes them an extremely important component of many essential systems. It is understandable that we try to take advantage of this important feature and involve computers in every task of our lives. Many of these tasks are particularly suitable for the application of computers: business accounting, managing inventories, processing images and an innumerable amount of other examples. Note that all the mentioned examples have something in common: they have a repetitive/predictable nature. But, can computers be beneficial for other more uncertain goals? If computers are nothing more than *automata* which are instructed to perform specific tasks:

> *How can we program a computer to perform a task that we do not even know how to complete?*

There are examples of tasks that computers can perform without being explicitly programmed for (such as image recognition, email spam and malware filtering, videos surveillance, etc), many of them are the focus of the mentioned machine learning techniques. However, analyzing cryptographic primitives is extremely challenging for automation. We cannot hope to *"solve cryptography"* having a tool that analyzes every single cryptographic construction. One reason is that cryptography is an evolving discipline: new notions, constructions, techniques are constantly created and revisited. Furthermore, even though automation is an emerging practice, the number of primitives that nowadays can be analyzed automatically is limited to certain classes of schemes:

> *Is there a likeness that characterizes the cryptographic constructions that are amenable for automation?*

In this thesis we investigate this idea, trying to answer to the previous question. Roughly, we can conclude that automation is specially adequate to prove security of primitives that can be expressed in a symbolic model. The first step for automating the analysis of cryptographic primitives is building a model in which the primitive can be expressed, usually, a model based on *symbols* with an underlying algebraic structure or equational theory. Computers can be programmed to decide notions about schemes expressed in these symbolic models. We note that the *symbolic model* notion is frequently associated to *Dolev-Yao* model, due to Needham and Schroeder [161] and Dolev and Yao [99]. In such a model, primitives are represented by function symbols that are treated as black-boxes. Extra equations may be added to the model to capture algebraic properties of the primitives, but the model is said to *assume perfect cryptography* in the sense that only the equalities that are explicitly given to the model in form of equations hold. Our notion of *symbolic model* is slightly more ambitious in the sense that we do not stop there. We try to give an answer to the following question:

> *Can the conclusions derived about the symbolic representation of a cryptographic primitive be extrapolated to actual security statements about the construction?*

Usually, cryptographic security experiments are probabilistic, e.g., every probabilistic polynomial-time (p.p.t., for short) algorithm has a probability of breaking the cryptographic construction that is upper-bounded by a negligible function; while statements in symbolic models are deterministic, e.g., certain system of constraints has or has not a solution. In our work we try to relate both notions, achieving more realistic and meaningful results. We can say our work is close to the line of work on automation of cryptographic proofs in both the *computational* [88] and *Dolev-Yao* models, we refer to [59] for an overview of these models.

Relating both models is commonly done *ad hoc*, i.e., a theoretical result relating the *symbolic experiments* with the *probabilistic experiments* (the so-called Master Theorem) is proven for every individual class of primitives. That is one of the reasons why, when considering such more realistic symbolic models, automated analysis in cryptography is so segregated, every class of schemes needs a model whose theoretical basis requires individual attention, which leads to the question:

> *What characterizes the classes of primitives that admit a symbolic model with its corresponding Master Theorem?*

Describing the cryptographic primitives to be analyzed is another crucial point in automated analysis. Automated tools usually take as input a file where the cryptographic primitive and the security notion to be proven are described. It is desirable that the language for describing primitives is as rich as possible, capturing a wide variety of schemes. However, as discussed above, automated methods are limited. The expressivity of the language must be adequate to the limitation of the model. If the language were too expressive, many cryptographic schemes that could be captured by it, would not be handled by the automation, which is undesirable. In that case we can say the language is not *accurate*. If the automation can handle any instance expressed in the language, we say the approach is complete.

> *How should the language for describing primitives be in order to achieve a good trade-off between expressivity and accuracy?*

Developing automatic tools that are *complete* is immensely challenging. Having complete tools is an intriguing line of research which has many advantages, e.g., one can prove qualitative/quantitative properties about classes of schemes by exhaustive search.

In this thesis we try to broaden the scope of automated analysis in cryptography, extending existing *Master Theorems*. This enlarges the number of primitives that can be handled by symbolic models and improves the security guarantees that can be proven about them.

1.4 Thesis Contributions

The main contribution of this thesis is on advancing the study of automated methods for analyzing cryptographic constructions. In this section we provide a high-level overview of our contributions. We refer to the corresponding chapters for more details.

- **Automated methods for stronger security guarantees in the GGM.** The gold standard in provable-security is to demonstrate security in the standard model. However, proofs in the standard model sometimes rely on atypical hardness assumptions. In such situations, it is essential to prove that the hardness assumptions used in the security proofs meet some minimal requirements, for instance the absence of algebraic attacks. The accepted method for validating new DDH-like assumptions is to show absence of generic attacks, i.e., attacks that solely exploit the underlying algebraic structure, using the Generic Group Model (GGM) [156, 160, 172, 182] or its bilinear and multilinear variants [43, 63].

The Generic Group Model provides an algebraic setting for describing a wide class of DDH-like assumptions, and is supported by Master Theorems that give a purely algebraic condition that ensures the security of an assumption in the GGM (or its variants). The Generic Group Analyzer (gga) [43] is a tool that assists proofs of security assumptions in the GGM, by automatically checking the conditions of one of such Master Theorems.

New Master Theorem. The first main contribution of this thesis is to extend the Master Theorem used by the gga to a general setting where adversaries can make arbitrarily many queries to oracles with group inputs, and where the winning conditions can be described using a rich language. As for simpler Master Theorems, ours (Theorem 2) yields a sufficient condition for the security of cryptographic constructions. However, this simpler condition cannot be expressed in finite-dimensional linear algebra: informally, each adversarial query to an oracle taking group elements as inputs increases the dimension of the system to be analyzed, and therefore allowing arbitrarily many queries leads to a system that is not finite-dimensional. As a consequence, the algebraic approach of the Generic Group Analyzer cannot be used to automatically evaluate sufficient conditions given by our Master Theorem.

New tool. The second main contribution of this work is an automated method for proving the validity of these conditions, using a combination of methods from constraint solving, computer algebra, and symbolic cryptography. Building on these two contributions, we implement an analyzer that can analyze many cryptographic constructions, including Digital Signatures and message authentication codes.

We refer to Chapter 3 for more details on these contributions.

- **Improving expressivity and efficiency Predicate Encryption.**

  Predicate Encryption (PE) [66, 138] is a form of public-key encryption (PKE) that supports fine-grained access control for encrypted data. In PE, everyone can create ciphertexts while keys can only be created by the master key owner.

Ciphertexts are associated with descriptive values x in addition to a plaintext, secret keys are associated with descriptive values y, and a secret key decrypts the ciphertext if, and only if, $\mathsf{P}(x, y) = 1$ for some boolean predicate P, which is in stark contrast to traditional public-key encryption, where access is all or nothing. Attribute-Based Encryption (ABE) is a special case of PE, where x is associated to a set of attributes and y is associated to an access policy. The simplest example of ABE is Identity-Based Encryption (IBE) [64, 81, 179] where x and y are identities and P corresponds to equality. The security requirement for ABE enforces resilience to collusion attacks, namely any group of users holding secret keys for different values learns nothing about the plaintext if none of them is individually authorized to decrypt the ciphertext. This should hold even if the adversary *adaptively* decides which secret keys to ask for, as is inevitable in real-world scenarios.

Predicate encodings [76, 190] are symmetric primitives that can be used for building Predicate Encryption and Attribute-Based Encryption schemes.

We pursue the study of predicate encodings and establish several general results and new constructions that broaden their scope and improve their efficiency. Our results lead to expressivity and performance improvements on state-of-the-art Predicate Encryption constructions.

Predicate encodings. We show that the information-theoretic definition of α-privacy used in [76, 190] is equivalent to an algebraic statement (furthermore independent of α) about the existence of solutions for a linear system of equations. Leveraging this result, we prove a representation theorem for predicate encodings: every triple of encoding functions implicitly defines a unique predicate for which it is a valid predicate encoding. Conversely, every predicate P that admits a predicate encoding is logically equivalent to the implicit predicate induced by its encoding functions. Moreover, our algebraic definition of privacy simplifies all our subsequent results.

First, we define a generic optimization of predicate encodings that often leads to efficiency improvements and reduces the number of required group elements in keys and ciphertexts. We prove the soundness of the transformations and validate their benefits experimentally on examples from [76, 190]; we successfully apply our simplifications to reduce the size of keys and ciphertexts by up to 50% and to reduce the number of group operations needed in some of the existing encodings.

Second, we define generic methods for combining predicate encodings. We provide encoding transformations for the *disjunction*, *conjunction* and *negation* of predicates, and for the *dual* predicate (see Definition 28).

Tag-based encodings. We show that our results on predicate encodings generalize to tag-based encodings. In particular, we give a purely algebraic characterization of the hiding property of tag-based encodings. Moreover, we demonstrate that the hiding property can be strengthened without any loss of generality, by requiring equality rather than statistical closeness of distributions.

Comparison of encodings. We compare the expressivity of the three core primitives (*predicate encodings*, *pair encodings* and *tag-based encodings*) corresponding to the three different modular frameworks. We provide an embedding that produces an information-theoretical pair encoding from every predicate encoding. Then, we use this encoding to compare our constructions (of boolean combinations of predicate encodings) with similar constructions for pair encodings that were introduced by [24].

In addition, we provide a transformation from tag-based encodings into predicate encodings.

New constructions. We develop several new constructions of predicate encodings and predicate encryption:

- **Combining predicates.** We show how to combine our results to build *Dual-Policy Attribute-Based Encryption (DP-ABE)* [28, 30] in the frameworks of predicate encodings and tag-based encodings (Section 4.6.1.1). Additionally, we consider the idea of combining arbitrary encodings with a *broadcast encryption* encoding to achieve direct revocation of keys. The former encoding takes care of revocation, while the latter encodes the desired access structure.
- **Improved predicate encodings.** We provide new instances of predicate encodings that improve on best known ones proposed in [76] and have additional properties. (Section 4.6.2.1.)
- **Extra features.** Finally, we show how to construct a weakly attribute-hiding predicate encoding for boolean formulas and how to enhance any predicate encoding with support for delegation. (Section 4.6.3.)

Implementation and evaluation. We implement a general library for predicate encryption with support for the predicate encoding and pair encoding frameworks. Our library uses the Relic-Toolkit [21] for pairings with a 256-bits Barreto-Naehrig elliptic curve [36]. We use our library for validating our constructions; experimental results are presented in the relevant sections. Our scalability experiments show that predicate encodings can be used for real applications.

- **Automated methods for analysis of Attribute-Based Encryption.**

  We propose, implement, and evaluate fully automated methods for proving security of ABE in the Generic Bilinear Group Model [63, 68]. Concretely, we introduce the class of Rational-Fraction Induced ABE (RFI-ABE), which includes many constructions from the literature, and prove for every ABE in this class that their security in the GGM is equivalent to security in a symbolic model, where the experiments are purely algebraic. Then, we introduce a notion of symbolic security for RFI-ABE, and prove that every symbolically secure RFI-ABE is secure in the symbolic model. Leveraging the fact that symbolic security suffices to conclude security of a Rational-Fraction Induced ABE in the GGM, we develop a constraint-solving method for proving symbolic security. Informally, the constraint-solving method can automatically (dis)prove the existence of solutions for systems of (in)equations between rational fractions. We implement the constraint-solving method and use it to evaluate several schemes, including schemes from the literature, various new schemes of independent interest, and some subtly insecure schemes. Our tool finds automated proofs for most constructions, and attacks for the insecure schemes.

  En route, we prove a Master Theorem relating security in the GGM to security in a symbolic model. The main technical difference with prior work is that our Master Theorem handles rational fractions instead of polynomials [32, 33, 63].

  Automated proofs. Our main theorem establishes that every RFI-ABE which satisfies symbolic security is also secure in the GGM, and justifies using automated methods for proving symbolic security. Informally, our notion of symbolic security asserts the (non-)existence of a solution to a system of equations between rational fractions; one specificity is that these equations may include so-called big operators, i.e., expressions of the form $\sum_{i=1}^{n} e_i$ or $\prod_{i=1}^{n} e_i$, where n can take arbitrary values. Because neither symbolic computation nor algorithmic verification tools can deal with big operators (the former do not support big operators and the latter operate on a bounded state space), we develop constraint-solving methods that can successfully analyze the systems of equations representing cryptographic constructions.

  In contrast to prior works, the main novelty of our tool is to consider systems of equations between rational fractions, rather than polynomial expressions. We stress that our tool achieves soundness but does not constitute a decision procedure; this means that our tool never makes mistakes but can sometimes fail to produce an output.

- **Formal model for the analysis of indifferentiability.**

  The framework of indifferentiability was introduced by Maurer et al. in 2004 [157]. It extends the classical notion of indistinguishability and simplifies the analysis of cryptographic constructions. In particular, the indifferentiability of certain (real) cryptographic component $\mathcal{C}$ from another (ideal) component $\mathcal{R}$ guarantees that the security of a cryptosystem depending on $\mathcal{R}$ is not affected if $\mathcal{R}$ is replaced by $\mathcal{C}$.

  In this work, we propose, implement and evaluate automated methods for analyzing cryptographic components in the framework of indifferentiability, with special emphasis on formalizing and automatically finding universal distinguishers.

  In particular, we introduce the notion of indifferentiability under universal algebraic attacks. A distinguisher is algebraic if performs operations from a restricted class, in the spirit of the Generic Group Model [156, 160, 182] and the Algebraic Group Model [107]. Roughly, we consider distinguishers that are restricted to perform operations that are used as building blocks of the cryptographic component, thereby, if a primitive is built based on $\oplus$ of n-bit strings and *permutations* $P : \{0,1\}^n \to \{0,1\}^n$, distinguishers are allowed to use these two operations, but they are not allowed to compute other operations such as the bit-wise conjunction of two bit-strings, for example. We review existing attacks from the literature and show that they fall into the class of universal algebraic attacks.

  In order to capture algebraic distinguishers, we define a *symbolic model* of indifferentiability and prove a Master Theorem which relates indifferentiability under universal algebraic attacks to symbolic indifferentiability (for a similar notion of universal algebraic attacks). The main benefit of the symbolic model is that winning conditions are now expressed in purely algebraic terms.

  We develop algorithms (decision procedures) for testing the universality of a distinguisher. These algorithms leverage techniques from unification theory such as *deductibility* or *static equivalence*.

  Tool. We implement our methods and evaluate their effectiveness on actual case studies: Feistel networks, Even-Mansour ciphers, confusion-diffusion networks and others. Formalizing and corroborating known indifferentiability attacks.

  Automated synthesis of attacks. As an independent contribution, our framework can be used to automatically find indifferentiability attacks on various cryptographic primitives. We propose two different heuristic methods in this direction and explore these approaches on primitives from the literature. In the case of 5-*rounds Feistel networks* we present a new attack, with a different structure to the one proposed by Coron et al. in [85] found with our tool.

We believe our results complement other works in the framework of indifferentiability. Our tool allows to formalize existing results (and potentially new ones) gaining confidence about their validity and extending the scope of computer-assistance to this delicate and important topic in cryptography.

1.5 Thesis Organization

In Chapter 2 we introduce some relevant cryptographic notions and notations and we present the Generic Group Model (GGM), the main cryptographic model for security considered in this thesis.

Chapters 3 and 5 are devoted to describing techniques for proving the security of cryptographic constructions in the GGM. Chapter 3 is mainly focused on computational experiments over bilinear groups, covering primitives such as Structure-Preserving Signatures (SPS) or Message Authentication Codes (MAC). On the other hand, Chapter 5 is focused on decisional experiments, with particular emphasis on Attribute-Based Encryption.

Chapter 4 presents results on predicate encodings, a building block for ABE. More concretely, a new algebraic characterization of *privacy* of an encoding is presented, which results crucial for the methods from Chapter 5. Due to this, it is advisable to read Chapter 4 before reading Chapter 5. Furthermore, this characterization allows to prove *generic transformations* and *combinations* of predicate encodings, improving the expressivity and efficiency of state-of-the-art ABE constructions (we refer to Chapter 4 for more details about these results).

Chapter 6 focuses on the notion of indifferentiability, with special focus on automatically finding attacks to the security of cryptographic components under this security notion, as well as IND-CPA and IND-CCA attacks.

Finally, in Chapter 7 we present our conclusions and possible future research directions.

2 Preliminaries

Begin at the beginning and go on till you come to the end, then stop.

Lewis Carroll, 1865

In this section, we introduce some common notation used throughout this thesis and some useful definitions. We also provide some background about the Generic Group Model. More specific notation is used in the rest of this document, which is described at the beginning of the chapter where it is used.

We note that the definitions related to the standard model (Section 2.2) are not extremely important for the rest of this dissertation, but are useful to illustrate the differences with the Generic Group Model.

Notation. We define $[n]$ as the range $\{1, \ldots, n\}$ for an arbitrary $n \in \mathbb{N}$. For finite sets S, we use $x \xleftarrow{\$} S$ to denote that x is uniformly sampled from S. We denote by $\varnothing$ the empty set or the empty list, depending on the context. For any set or list S, we write $|S|$ to denote its cardinality. For set $S = \{1, 2, 3\}$, we abuse notation and write $(a_i, b_i)_{i \in S}$ to denote the tuple or vector $(a_1, a_2, a_3, b_1, b_2, b_3)$, we extend this notation to arbitrary *ordered* finite sets.

A positive function $\mathsf{negl} : \mathbb{N} \to [0, 1] \subseteq \mathbb{R}$ is called *negligible* if for every polynomial $p(x) \in \mathbb{R}[X]$ there exists a constant λ_0 such that for every $\lambda \geqslant \lambda_0$ it holds $\mathsf{negl}(\lambda) < 1/p(\lambda)$. A function $\delta : \mathbb{N} \to [0, 1]$ is said to be *overwhelming* if the function $1 - \delta$ is negligible.

By $y \leftarrow A(x)$ we denote a process of computation where algorithm A takes x as input and outputs y. We denote by $A^{\mathcal{O}(s,\cdot)}$, the fact that algorithm A has oracle access to algorithm $\mathcal{O}$, where $\mathcal{O}$ takes two inputs and has value s hard-wired. Algorithm A can call $\mathcal{O}$ varying the second argument, getting the answer by $\mathcal{O}$ where the first argument is always s (note that s is possibly unknown to

A). We naturally extend the above notation to multi-input algorithms, using "$\cdot$" for the accessible inputs.

We use λ to denote the *security parameter*. Very roughly, it is desired that attacking *secure* cryptographic primitives requires an exponential time in λ. For algebraic groups $\mathbb{G}$ of order $n \in \mathbb{N}$, we usually assume that n is a λ-bit integer, i.e., $2^{\lambda-1} < n < 2^{\lambda}$.

2.1 Definitions

2.1.1 Bilinear Groups

Bilinear groups or *pairing groups* are algebraic groups with an structure that, roughly, allows to efficiently compute multiplications in the exponent. Bilinear groups have been used on countless occasions for developing simple, efficient and rich cryptographic constructions. A remarkable example is the celebrated Boneh-Franklin Identity-Based Encryption [64], where the use of pairings allowed the authors to solved an open problem proposed by Shamir in 1984 [179], but there are many other primitives that cryptographers only know how to construct by using bilinear groups.

Definition 1 (Bilinear group)**.** *Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{t}$ be cyclic groups of order $n \in \mathbb{N}$ and let generators g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively. A* bilinear group *is a tuple $(n, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_\mathrm{t}, e)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathrm{t}$ is a map satisfying*

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$$

for every $a, b \in \mathbb{N}$. It is required that the pairing be non-degenerate*, i.e., $e(g_1, g_2)$ generates $\mathbb{G}_\mathrm{t}$.*

We frequently use implicit representation of group elements: for $a \in \mathbb{Z}_p$ we write $[\![a]\!]_s$ to denote the implicit representation of g_s^a, element of $\mathbb{G}_s$ for $s \in \{1, 2, \mathrm{t}\}$, following [101].

Groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are known as the *source groups*, while $\mathbb{G}_\mathrm{t}$ is called the *target group*. In some cases, there exist additional isomorphisms between the source groups and, according to this criteria, Galbraith et al. [109] classify pairings in three different settings:

- Type I: there exist an efficiently computable isomorphism in both directions, $\Psi_1 : \mathbb{G}_1 \to \mathbb{G}_2$ and $\Psi_2 : \mathbb{G}_2 \to \mathbb{G}_1$. This setting is known as the *symmetric pairings*, while the other two are known as *asymmetric*. In the symmetric case, it is common to write $\mathbb{G}_1 = \mathbb{G}_2$.

- Type II: there exist an efficiently computable isomorphism in one direction, i.e., $\Psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- Type III: there are no (known) efficiently computable isomorphism between the source groups.

It is common to assume that there exists a probabilistic polynomial-time algorithm, GGen, that on input the security parameter 1^λ, outputs the description of a pairing group $\mathcal{G} = (n, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_t, e)$ of order n, where $n \in \mathbb{N}$ is a λ-bit integer. Throughout this thesis, we will assume n to be prime unless otherwise stated.

2.1.2 Digital Signature Scheme

A Digital Signature is a cryptographic scheme designed to enforce the *authenticity* of digital messages. It allows a verifier to gain confidence about the origin of the message. To have the guarantee that it was created by a known sender (*authentication*) and that the messages was not modified or altered while being transmitted (*integrity*). It also prevents the sender from denying to have produced the message (*non-repudiation*).

Digital Signatures have a tremendous importance in cryptography, they are a versatile building block used in many cryptographic protocols and constructions and in many countries, they have legal significance.

A very important class of Digital Signature schemes is the class of *Structure-Preserving Signature* schemes (SPS) [2], where the public key, messages and signatures consist exclusively of group elements, what makes SPS schemes a very useful building block for protocol design over bilinear groups. For example, Structure-Preserving Signature schemes are typically combined with *non-interactive proofs*, e.g., [54, 117, 119, 120, 135, 150, 151] to produce efficient *Non-Interactive Zero-Knowledge* proof systems (NIZK).

Definition 2 (Digital Signature Scheme)**.** *A digital signature scheme is a tuple of polynomial-time algorithms* $\{\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}\}$ *where:*

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$ *is a probabilistic algorithm that, given a security parameter λ, generates a verification key* pk *and a signing key* sk*, and outputs* $(\mathsf{pk}, \mathsf{sk})$*. The security parameter defines the message space $\mathcal{M}$.*
- $\mathsf{Sign}(\mathsf{sk}, m) \rightarrow \sigma$ *is a probabilistic algorithm that, computes a signature σ for input message $m \in \mathcal{M}$ by using the signing key* sk*.*
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \rightarrow \{0, 1\}$ *is a deterministic algorithm that, given a message and a signature, outputs 1 for acceptance or 0 for rejection according to the input.*

Correctness

The signer, who owns the secret key sk can run the Sign algorithm to produce a signature for certain message m. We require that the signature scheme be correct, i.e., the verification algorithm Verify must accept a genuinely generated signature. More precisely:

Definition 3 (Correctness of Digital Signatures)**.** *For the scheme to be correct, it must hold that for all $m \in \mathcal{M}$, the probability*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk},m)\end{array} \; : \; 1 = \mathsf{Verify}(\mathsf{pk},m,\sigma)\right]$$

is lower-bounded by an overwhelming function in λ.

Security

We also require the signature scheme be *secure*. Very roughly, even in the presence of valid signatures for different messages, it must be hard to produce a signature that passes the verification on a different new message. More formally:

Definition 4 (Unforgeability against Chosen-Message Attacks: $\mathsf{EUF\text{-}CMA}$)**.** *A signature scheme, $\mathsf{SIG} = \{\mathsf{Setup},\mathsf{Sign},\mathsf{Verify}\}$, is* existentially unforgeable against adaptive chosen-message attacks *if the following advantage function is negligible in λ for any p.p.t. adversary $\mathcal{A}$:*

$$\mathrm{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)\\ (\widehat{\sigma},\widehat{m}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{sk}}}(\mathsf{pk})\end{array} \; : \; \begin{array}{l}\widehat{m} \notin \mathcal{Q} \;\wedge\\ 1 = \mathsf{Verify}(\mathsf{pk},\widehat{m},\widehat{\sigma})\end{array}\right]$$

where $\mathcal{O}_{\mathsf{sk}}$ is an oracle that, given m, executes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},m)$, records m to $\mathcal{Q}$, and returns σ.

A weaker security notion is *unforgeability against Random-Message Attacks* ($\mathsf{EUF\text{-}RMA}$), where the adversary is allowed to get several genuinely-signed messages, but not of its choice. Specifically, the oracle the formal definition of this notion would differ from the one above only in the oracle $\mathcal{O}_{\mathsf{sk}}$ that would not receive any input and on every call, it would internally sample some random message $r \in \mathcal{M}$ uniformly at random, compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},r)$, add r to $\mathcal{Q}$ and return signature σ.

2.1.3 Encryption Scheme

An Encryption Scheme is a cryptographic scheme designed to enforce the *privacy*. It is a mechanism that allows to hide a message in such a way that the authorized receivers can access it, while unauthorized parties cannot extract any information from the encoded message.

Definition 5 (Public-Key Encryption Scheme, PKE)**.** *A public-key encryption scheme is a tuple of polynomial-time algorithms* $\{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *where:*

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *is a probabilistic algorithm that, given a security parameter* λ*, generates a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*, called the* public key *and the* private key *respectively. The security parameter defines the message space* $\mathcal{M}$*.*

- $\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{ct}$ *is a probabilistic algorithm that takes a public key* pk *and encrypts message* $m \in \mathcal{M}$*, producing a ciphertext* ct*.*

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to m$ *is a deterministic algorithm that takes as input a private key* sk *and a ciphertext* ct *and outputs a message* $m \in \mathcal{M}$ *or a special symbol* $\perp$*.*

Correctness

It is required that decryption under secret key sk of a ciphertext produced for public key pk, be successful, if $(\mathsf{pk}, \mathsf{sk})$ is a pair of keys generated by KeyGen. More formally,

Definition 6 (Correctness of Public-Key Encryption)**.** *For the scheme to be correct, it must hold that for all* $m \in \mathcal{M}$*, the probability*

$$\Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct} \leftarrow \mathsf{Sign}(\mathsf{pk}, m)\end{array} : \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) = m\right]$$

is lower-bounded by an overwhelming function in λ*.*

Security

Defining security for a public-key encryption scheme is far from trivial. One would desire that the guarantees of security that the encryption provides are such that, given a ciphertext, it is impractical to recover the original message. However such a definition does not seem enough for having good guarantees of privacy, especially in certain scenarios. Think of a situation where the number of possible plaintexts is small, for example, a situation in which a trader performs *buy/sell* transactions via encrypted messages, so that her decisions remain hidden except for the party that sells/buys the financial goods. An eavesdropper may not be able to recover the original messages that are being transmitted, however it may be able to distinguish between *buy* and *sell* orders, which is definitely undesirable.

In 1982, Goldwasser and Micali proposed the notion of *semantic security* [114], which was a breakthrough for the formal analysis and understanding of cryptography. In fact, these authors received in 2012 the ACM Turing Award for such definition and other revolutionary contributions to the field of cryptography. Roughly, an encryption scheme is said to be semantically secure if only negligible information about the plaintext can be efficiently extracted from a given ciphertext. However, this semantic security is hard to be applied in cryptographic proves. The same authors demonstrated that semantic security is equivalent to a different notion of security, known as *indistinguishability against chosen-plaintext attacks*. This second definition is easier to adopt and deal with and facilitates the security reductions in practice. In Section 2.2 we prove that ElGamal encryption scheme satisfies this notion of security under the DDH assumption (Definition 16).

Definition 7 (Indistinguishability against Chosen-Plaintext Attacks: IND-CPA)**.** *A public-key encryption scheme,* $\mathsf{PKE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$*, is* indistinguishable against chosen-plaintext attacks *if the following advantage function is negligible in* λ *for any p.p.t. stateful adversary* $\mathcal{A}$*:*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ m_0, m_1 \leftarrow \mathcal{A}(\mathsf{pk}) \\ \beta \xleftarrow{\$} \{0,1\};\ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m_\beta) \\ \widehat{\beta} \leftarrow \mathcal{A}(\mathsf{ct}) \end{array} : \ \widehat{\beta} = \beta \right] - \frac{1}{2}$$

Intuitively, the adversary chooses two messages, m_0 and m_1 and it is given the encryption of one of them at random. The encryption scheme is said to be IND-CPA if the adversary cannot tell which message was encrypted (actually, not much better than just guessing at random).

Note how strong this definition of security is. It guarantees that not even a single bit of information is leaked from a ciphertext, even for messages that are chosen by the attacker. Furthermore, observe that in many practical scenarios, attackers will not choose the messages that are encrypted.

There are stronger notions of security where the adversary is allowed to query *decryption oracles*, that decrypt ciphertexts of its choice, like IND-CCA:

Definition 8 (Indistinguishability against Adaptive Chosen-Ciphertext Attacks: IND-CCA2)**.** *A public-key encryption scheme,* $\mathsf{PKE} = \{\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}\}$*, is* indistinguishable against adaptive chosen-ciphertext attacks *if the following advantage function is negligible in* λ *for any p.p.t. stateful adversary* $\mathcal{A}$*:*

$$\mathrm{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr\left[\begin{array}{l} (\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ m_0, m_1 \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk}) \\ \beta \xleftarrow{\$} \{0,1\};\ \mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m_\beta) \\ \widehat{\beta} \leftarrow \mathcal{A}^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{ct}) \end{array} : \ \widehat{\beta} = \beta \right] - \frac{1}{2}$$

It has been shown that the above definition is equivalent to *non-malleability*, namely, for a scheme that is IND-CCA2, it is unfeasible to transform a given ciphertext into another ciphertext which decrypts to a related plaintext.

Actually, this notion of security is one of the most accepted and desired for encryption schemes.

2.1.4 Identity-Based Encryption

Identity-Based Encryption (IBE) is the simplest example of Attribute-Based Encryption (see Section 2.1.5), introduced by [179]. It can be seen as an instance of *Predicate Encryption* for the predicate $\mathsf{P}(x, y) = 1$ iff $x = y$, $x, y \in \mathbb{Z}_p$. In an Identity-Based Encryption scheme, a user, Alice, can send a message to another user, Bob, by only considering some public parameters and Bob's identity (a pre-existing identifier, e.g., an email address), unlike traditional public-key encryption, where Bob would need to communicate his public key to Alice. In general, IBE simplifies the key management of certificate-based public-key infrastructure.

A major use case for IBE is email encryption, where it allows pairwise email encryption, that is, Alice can send an encrypted email directly to Bob without Bob's involvement. This technology is being adopted in real-life applications. In fact, early IBE schemes are being standardized in IEEE P1363.3 and RFC 5091.

2.1.5 Attribute-Based Encryption

Attribute-Based Encryption (ABE) [174] is a form of of public-key encryption that supports fine-grained access control for encrypted data. In attribute-based encryption, everyone can create ciphertexts while keys can only be created by the master key owner. ABE schemes use predicates to model (potentially complex) access control policies, and attributes are attached to ciphertexts or secret keys. In the Key-Policy ABE (KP-ABE) version of Attribute-Based Encryption, keys are associated to access policies, while ciphertexts are associated to sets of attributes. On the other hand, in the Ciphertext-Policy ABE (CP-ABE) version, keys are associated to attributes, while ciphertexts are associated to policies.

Predicate Encryption (PE) [66, 138] generalizes Attribute-Based encryption. A predicate encryption scheme for a predicate P guarantees that decryption of a ciphertext ct_x with a secret key sk_y is allowed if, and only if, the attribute x associated to the ciphertext ct and the attribute y associated to the secret key sk verify the predicate P, i.e., $\mathsf{P}(x, y) = 1$. Predicate encryption schemes exist for several useful predicates, such as Zero Inner-Product Encryption (ZIPE), where attributes are vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and the predicate $\mathsf{P}(\boldsymbol{x}, \boldsymbol{y})$ is defined as $\boldsymbol{x}^\top \boldsymbol{y} = 0$. Predicate encryption subsumes several previously defined notions of public-key encryption. For example, Identity-Based Encryption (IBE) [179] can be obtained by defining $\mathsf{P}(x, y)$ as $x = y$. As mentioned before, Attribute-Based Encryption (ABE) [174] can also be instantiated as a predicate encryption scheme similarly. More concretely, for Key-Policy ABE, the attribute x is a boolean vector, the attribute y is a boolean function, and the predicate $\mathsf{P}(x, y)$ is defined as $y(x) = 1$. For Ciphertext-Policy ABE, the roles of the attributes x and y are swapped.

The differences between Predicate Encryption and Attribute-Based Encryption are not very clear in the literature, however, it is accepted that PE subsumes ABE. We can argue that in PE, x or y do not need to be sets of attributes or policies, but can me more abstract objects. However, the main different between the PE and ABE is syntactical, in the sense that in PE, the Dec algorithm does not require value x (associated to the ciphertext) as an input, unlike in ABE. That opens the possibility of defining *attribute-hiding* PE schemes, where value x is not only not provided for decryption, but it cannot be inferred from the rest of the ciphertext ct_x.

Attribute-Based Encryption and Predicate Encryption are increasingly being applied to many areas and recent efficiency improvements make them feasible for real-life applications [16, 31, 129, 149].

Definition 9 (Attribute-Based Encryption)**.** *Given a predicate* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$*, an Attribute-Based Encryption scheme is tuple of polynomial-time algorithms* $\{\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec}\}$ *where:*

- $\mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}) \to (\mathsf{mpk}, \mathsf{msk})$ *gets as input the security parameter* λ*, the attribute universe* $\mathcal{X}$*, the predicate universe* $\mathcal{Y}$*. It outputs a master secret key* msk *and a master public key* mpk*, defining the key space* $\mathcal{K}$*.*

- $\mathsf{Enc}(\mathsf{mpk}, x) \to (\mathsf{ct}_x, \kappa)$ *takes as input* mpk *and an attribute* $x \in \mathcal{X}$*. It outputs a ciphertext* ct_x *and a symmetric encryption key* $\kappa \in \mathcal{K}$*.*

- $\mathsf{KeyGen}(\mathsf{msk}, y) \to \mathsf{sk}_y$ *gets as input* msk *and a value* $y \in \mathcal{Y}$ *returning a secret key* sk_y*.*

- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x, x) \to \kappa$ *gets as input* sk_y *and* ct_x *such that* $\mathsf{P}(x, y) = 1$*. It outputs a symmetric key* κ*.*

Correctness

An attribute Encryption scheme is correct if the decryption, of a genuinely generated ciphertext, ct_x with a valid secret key sk_y, holds whenever $\mathsf{P}(x, y) = 1$.

Definition 10 (Correctness of Attribute-Based Encryption). *For all $x \in \mathcal{X}$, and all $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 1$,*

$$\Pr[\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x, x) = \kappa] = 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y})$*, and where* $(\mathsf{ct}_x, \kappa) \leftarrow \mathsf{Enc}(\mathsf{mpk}, x)$*,* $\mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, y)$*.*

Security

For security we require the scheme be such that, given a ciphertext ct_x produced from Enc, it is hard for an adversary to distinguish between the symmetric key κ produced with ct_x from a uniformly sampled symmetric key. Even in the presence of secret keys sk_y (for values y such that $\mathsf{P}(x, y) = 0$). More formally:

Adaptive security. For any stateful adversary $\mathcal{A}$ and security parameter λ, we define the advantage function:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) := \Pr\left[\begin{array}{r} (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}) \\ x^\star \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{mpk}) \\ (\mathsf{ct}_{x^\star}, \kappa) \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^\star) \\ \beta \xleftarrow{\$} \{0,1\};\ K_0 := \kappa;\ K_1 \xleftarrow{\$} \mathcal{K} \\ \widetilde{\beta} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}_{x^\star}, K_\beta) \end{array} : \widetilde{\beta} = \beta \right] - \frac{1}{2}$$

with the restriction that all queries y that $\mathcal{A}$ makes to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ must satisfy $\mathsf{P}(x^\star, y) = 0$ (that is, the secret keys cannot directly decrypt the challenge ciphertext).

Definition 11 (Adaptive security of Attribute-Based Encryption). *An* ABE *scheme is* adaptively secure *if there exists a* negligible *function* negl *such that for all p.p.t. adversaries* $\mathcal{A}$ *and all sufficiently large* $\lambda \in \mathbb{N}$, $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{ABE}}(\lambda) \leqslant \mathsf{negl}(\lambda)$.

Selective security. We also consider a weaker security notion than the one above, that captures selective attacks, where the adversary selects a challenge $x^\star$ independently of its view. Namely, for any stateful adversary $\mathcal{A}$ and security parameter λ, we define the advantage function:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{sel\text{-}ABE}}(\lambda) := \Pr\left[\begin{array}{r} x^\star \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}) \\ (\mathsf{ct}_{x^\star}, \kappa) \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^\star) \\ \beta \xleftarrow{\$} \{0,1\};\ K_0 := \kappa;\ K_1 \xleftarrow{\$} \mathcal{K} \\ \widetilde{\beta} \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot)}(\mathsf{ct}_{x^\star}, K_\beta) \end{array} : \widetilde{\beta} = \beta \right] - \frac{1}{2}$$

with the restriction that all queries y that $\mathcal{A}$ makes to $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$ must satisfy $\mathsf{P}(x^\star, y) = 0$.

Definition 12 (Selective security of Attribute-Based Encryption)**.** *An* ABE *scheme is* selectively secure *if there exists a* negligible *function* negl *such that for all p.p.t. adversaries $\mathcal{A}$ and all sufficiently large $\lambda \in \mathbb{N}$,* $\mathrm{Adv}_{\mathcal{A}}^{\mathsf{sel\text{-}ABE}}(\lambda) \leqslant \mathsf{negl}(\lambda)$.

2.1.6 Monotone Access Structures

It is common to consider *access structures* that are *monotonic.* An access structure $f : \mathcal{X} \to \{0,1\}$ is said to be monotonic if there exists a partial order $\preceq$ on $\mathcal{X}$ such that

$$(x \preceq x') \wedge f(x) = 1 \;\;\Rightarrow\;\; f(x') = 1$$

Intuitively, if the elements of $\mathcal{X}$ are sets of attributes and we define $\preceq$ as the set inclusion, having more attributes will never turn a monotonic formula from 1 to 0.

It can be shown that monotonic formulas are boolean formulas that are formed with $\wedge$ and $\vee$ gates exclusively. The following is an algebraic characterization of monotonic formulas, widely used for building Predicate Encryption schemes.

Definition 13 (Access Structure [52, 137])**.** *A* (monotone) access structure *for attribute universe $\mathcal{U}$ is a pair $(\mathbf{M}, \rho)$ where $\mathbf{M} \in \mathbb{Z}_p^{\ell \times \tilde{\ell}}$ and $\rho : [\ell] \to \mathcal{U}$. Given $\Gamma \subseteq \mathcal{U}$, we say that*

$$\Gamma \textit{ satisfies } (\mathbf{M}, \rho) \textit{ iff } \mathbf{1}^\top \in \mathsf{span}_{\mathsf{row}}(\mathbf{M}_\Gamma),$$

Here, $\mathbf{1} := (1, 0, \ldots, 0) \in \mathbb{Z}^{\tilde{\ell}}$ is a row vector; $\mathbf{M}_\Gamma$ denotes the collection of vectors $\{\mathbf{M}_j : \rho(j) \in \Gamma\}$ where $\mathbf{M}_i$ denotes the i'th row of $\mathbf{M}$; and $\mathsf{span}_{\mathsf{row}}$ refers to linear span of collection of (row) vectors over $\mathbb{Z}_p$.

Intuitively, Γ satisfies $(\mathbf{M}, \rho)$ iff there exists constants $\omega_1, \ldots, \omega_\ell \in \mathbb{Z}_p$ such that

$$\sum_{\rho(j) \in \Gamma} \omega_j \mathbf{M}_j = \mathbf{1}^\top$$

Observe that the constants $\{\omega_i\}$ can be computed in polynomial time in the size of matrix $\mathbf{M}$ via *Gaussian elimination.*

2.2 Standard Model in Cryptography

The standard model in cryptography is the most widely used and accepted model for provable-security. It consists of isolating specific hard problems and proving the security of cryptographic constructions provided that the selected problems are actually hard. Nonetheless, unless new mathematical techniques are developed and discovered, the hardness of such problems will need to be assumed, in an act of faith, supported by the big efforts and unsuccessful attempts made by the research community on solving them. These widely used hard problems are known as *hardness assumptions*.

2.2.1 Hardness assumptions

In this section we present some of the most widely used and better accepted hardness assumptions. It is desirable to reduce the security of new cryptographic primitives to the hardness of one of these problems. However, it is common that in practice, new *ad hoc* assumptions are used for making such security reductions. In such cases, it is crucial to validate the new assumptions. The GGM (Section 2.3) is an ideal tool to gain confidence about the algebraic soundness of these unusual problems.

Computational problems are about calculating some values satisfying certain properties, while *decisional problems* are about distinguishing between two or more situations.

Computational assumptions.

Definition 14 (Discrete logarithm, DLOG)**.** *Let $\mathbb{G}$ be a finite group. The discrete logarithm problem consists of, given $g, h \in \mathbb{G}$, finding $x \in \mathbb{N}$ such that $g^x = h$.*

The *discrete logarithm problem* is said to be hard for a group $\mathbb{G}$ if there exists a negligible function negl such that for every p.p.t. algorithm $\mathcal{A}$ it holds

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{DLOG}} := \Pr\left[\, g, h \xleftarrow{\$} \mathbb{G};\; x \leftarrow \mathcal{A}(g,h) \,:\, g^x = h \,\right] \leqslant \mathsf{negl}(\lambda)$$

where $2^\lambda \approx |\mathbb{G}|$.

Definition 15 (Computational Diffie-Hellman, CDH)**.** *Let $\mathbb{G}$ be a finite group or order $n \in \mathbb{N}$ and let $g \in \mathbb{G}\backslash\{1_{\mathbb{G}}\}$. Let $a, b \in \mathbb{N}$. The CDH problem consists of, given g, g^a, g^b, computing g^{ab}.*

The *CDH assumption* is said to hold for group $\mathbb{G}$ if there exists a negligible function negl such that for every p.p.t. algorithm $\mathcal{A}$,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{CDH}} := \Pr\left[\, g \xleftarrow{\$} \mathbb{G}\backslash\{1_{\mathbb{G}}\};\; a, b \xleftarrow{\$} \mathbb{Z}_n;\; h \leftarrow \mathcal{A}(g, g^a, g^b) \,:\, h = g^{ab} \,\right] \leqslant \mathsf{negl}(\lambda)$$

where $2^\lambda \approx |\mathbb{G}|$.

Decisional assumptions.

Definition 16 (Decisional Diffie-Hellman, DDH)**.** *Let $\mathbb{G}$ be a finite group of order $n \in \mathbb{N}$ and let $g \in \mathbb{G}\backslash\{1_{\mathbb{G}}\}$. Let $a, b \in \mathbb{N}$. The DDH problem consists of, given g, g^a, g^b and $C \in \mathbb{G}$, deciding whether $C = g^{ab}$ or not.*

The *DDH assumption* is said to hold for group $\mathbb{G}$ if there exists a negligible function negl such that for every p.p.t. algorithm $\mathcal{A}$,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{DDH}} := \Pr\left[\begin{array}{ll} g \xleftarrow{\$} \mathbb{G}\backslash\{1_{\mathbb{G}}\}; & a, b \xleftarrow{\$} \mathbb{Z}_n; \\ C_0 = g^{ab}; & C_1 \xleftarrow{\$} \mathbb{G}; \\ \beta \xleftarrow{\$} \{0,1\}; & \widetilde{\beta} \leftarrow \mathcal{A}(g, g^a, g^b, C_\beta) \end{array} : \widetilde{\beta} = \beta \right] - \frac{1}{2} \leqslant \mathsf{negl}(\lambda)$$

where $2^\lambda \approx |\mathbb{G}|$.

The DDH assumption has led to many efficient cryptographic systems with robust security properties. It has been an extremely important invention for cryptography and, actually, Dan Boneh describes it as a *gold mine* [60].

Subgroup membership assumptions. There are many variants of subgroup membership assumptions. They are all defined over groups of *composite order*, where the order is usually a product of two or three "large" primes. The problem states that it is hard for a polynomial adversary to classify a randomly chosen group element in the different subgroups that the group harbors.

Definition 17 (A subgroup membership problem)**.** *Let $\mathbb{G}$ be a cyclic group of order $N \in \mathbb{N}$, where $n = pq$ for p and q primes, and let g be a generator of $\mathbb{G}$. This subgroup membership problem consists of, given N, g and $h \in \mathbb{G}$, deciding whether h has order p or it has order pq.*

This *subgroup membership assumption* is said to hold for group $\mathbb{G}$ if there exists a negligible function negl such that for every p.p.t. algorithm $\mathcal{A}$,

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{SM}} := \Pr\left[\begin{array}{ll} h \xleftarrow{\$} \mathbb{G}; & \\ h_0 = h; & h_1 = h^q; \\ \beta \xleftarrow{\$} \{0,1\}; & \widetilde{\beta} \leftarrow \mathcal{A}(N, g, h_\beta) \end{array} : \widetilde{\beta} = \beta \right] - \frac{1}{2} \leqslant \mathsf{negl}(\lambda)$$

where p and q are λ-bit primes. Observe that h^q is always an element of the subgroup of order p, while h is an element of the subgroup of order pq with overwhelming probability. Note that factoring N would allow to recover the correct β. Therefore, we say that this assumption is *stronger* than factoring; or, to be very precise, we could say the assumption is *not weaker* than factoring.

Subgroup membership assumptions result extremely useful for making security reductions in the standard model. For example, the *Petit IBE* [191] has been proven adaptively secure in the composite order (under subgroup membership assumptions), but no alternative proof exists (so far) in the prime order setting. In Chapter 5 we provide a proof of its *generic security* in the prime order setting, found with our tool.

2.2.2 Example of proof in the Standard Model

ElGamal encryption scheme was created by Taher Elgamal in 1985 [100]. It is a simple and elegant public-key encryption scheme that relies on the hardness of computing discrete logarithms. In this section we show that ElGamal encryption scheme is IND-CPA under the DDH assumption.

Definition 18 (ElGamal encryption scheme)**.** ElGamal encryption *is given by the following three algorithms:*

```
$\mathsf{KeyGen}(1^\lambda)$ :
    Define a group $\mathbb{G}$
      of prime order $p \approx 2^\lambda$
    Pick a generator $g \in \mathbb{G}$
    $v \xleftarrow{\$} \mathbb{Z}_p$
    $\mathsf{pk} := (\mathbb{G}, g, g^v)$
    $\mathsf{sk} := (\mathbb{G}, v)$
    return $(\mathsf{pk}, \mathsf{sk})$

$\mathsf{Enc}(\mathsf{pk} = (\mathbb{G}, g, V), M \in \mathbb{G})$ :
    $r \xleftarrow{\$} \mathbb{Z}_p$
    $\mathsf{ct} := (g^r, M \cdot V^r)$
    return $\mathsf{ct}$

$\mathsf{Dec}(\mathsf{sk} = (\mathbb{G}, v), (\mathsf{ct}_1, \mathsf{ct}_2) \in \mathbb{G}^2)$ :
    return $\mathsf{ct}_2 / \mathsf{ct}_1^v$
```

Theorem 1. *If the Decisional Diffie-Hellman assumption holds, then for all p.p.t. adversaries $\mathcal{A}$ the advantage $\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{ElGamal,\mathcal{A}}(\lambda)$ is bounded by a negligible function in λ.*

Proof. We show that every adversary $\mathcal{A}$ attacking the IND-CPA experiment of ElGamal can be transformed into an adversary $\mathcal{B}$ against the DDH problem succeeding with a very similar probability as $\mathcal{A}$ (it fact, the advantage of $\mathcal{B}$ is half the advantage of $\mathcal{A}$). Because all efficient adversaries against DDH are assumed to have a negligible advantage, it must be that the advantage of $\mathcal{A}$ against the IND-CPA of ElGamal is negligible.

Without loss of generality, we assume that $\mathcal{A}$ always outputs a bit in the IND-CPA experiment. Now, algorithm $\mathcal{B}$ is given an instance of a DDH problem, say (g, g^a, g^b, C) for certain group $\mathbb{G}$, $g \in \mathbb{G}$ and for some (unknown to $\mathcal{B}$) $a, b \in \mathbb{Z}_p$ and where C is either g^{ab} or uniformly random. Algorithm $\mathcal{B}$ simulates the KeyGen algorithm by defining $\mathsf{pk} = (\mathbb{G}, g, g^a)$. It sends pk to algorithm $\mathcal{A}$. Eventually, $\mathcal{A}$ will reply back with two messages (two group elements) $M_0, M_1 \in \mathbb{G}$. Now, $\mathcal{B}$ samples a bit $\beta \xleftarrow{\$} \{0,1\}$ and sends the ciphertext $(g^b, M_\beta \cdot C)$ to $\mathcal{A}$. When $\mathcal{A}$ returns a bit $\widetilde{\beta}$, $\mathcal{B}$ returns 0 (real, i.e., $C = g^{ab}$) when $\widetilde{\beta} = \beta$, otherwise it returns 1 (random).

Note that, if $C = g^{ab}$, the produced ciphertext would be distributed as a genuine ciphertext. On the other hand, if C were selected at random, the ciphertext $(g^b, M_\beta \cdot C)$ would contain no information about β. That is because for every $M \in \mathbb{G}$, the following distributions are identical:

$$U \xleftarrow{\$} \mathbb{G};\ \mathsf{return}\, U \quad \equiv \quad U \xleftarrow{\$} \mathbb{G};\ \mathsf{return}\, M \cdot U \ .$$

We have,

$$\begin{aligned}
&\mathrm{Adv}_{\mathcal{B}}^{\mathsf{DDH}}(\lambda) \\
&\quad = \Pr\left[\beta = \widetilde{\beta} \,|\, C = g^{ab}\right] \Pr\left[C = g^{ab}\right] + \Pr\left[\beta \neq \widetilde{\beta} \,|\, C \xleftarrow{\$} \mathbb{G}\right] \Pr\left[C \xleftarrow{\$} \mathbb{G}\right] - \frac{1}{2} \\
&\quad = \left(\mathrm{Adv}_{\mathit{ElGammal},\mathcal{A}}^{\mathsf{IND\text{-}CPA}} + \frac{1}{2}\right) \Pr\left[C = g^{ab}\right] + \frac{1}{2} \Pr\left[C \xleftarrow{\$} \mathbb{G}\right] - \frac{1}{2} \\
&\quad = \left(\mathrm{Adv}_{\mathit{ElGammal},\mathcal{A}}^{\mathsf{IND\text{-}CPA}} + \frac{1}{2}\right) \frac{1}{2} + \frac{1}{4} - \frac{1}{2} \\
&\quad = \frac{\mathrm{Adv}_{\mathit{ElGammal},\mathcal{A}}^{\mathsf{IND\text{-}CPA}}}{2}
\end{aligned}$$

where $\Pr\left[\beta \neq \widetilde{\beta} \,|\, C \xleftarrow{\$} \mathbb{G}\right]$ was simplified to $1/2$ because, the input to $\mathcal{A}$ does not contain any information about β and therefore, the sampling of β can be delayed until $\widetilde{\beta}$ was chosen by $\mathcal{A}$. $\square$

2.3 Generic Group Model

The Generic Group Model is an idealized cryptographic model, first introduced by Nechaev [160], that allows to prove the absence of efficient attacks that *do not exploit the representation of the group.*

In cryptography, especially in *public-key cryptography*, it is very common to define primitives over an algebraic group. For example, the RSA encryption scheme is defined over a group of composite order $N = pq$ where p and q are "*large primes*", so that it is hard to factor N. In RSA, messages are group elements and both encryption and decryption consist of taking exponentiations in the group.

It is common to define primitives for over such a group $\mathbb{G}$, ignoring the group implementation. In practice, this group needs to be carefully selected so that it is *hard* to solve certain problems over it. Note that the representation of the group plays an essential role when reasoning about the *hardness* of problems. In particular, there are problems that are easy for certain representations and intractable (with our current knowledge) for others.

For example, the group of integers modulo $p-1$ for a prime p with the addition, $(\mathbb{Z}_{p-1}, +)$, is isomorphic to the group of positive integers modulo p with the product, $(\mathbb{Z}_p^*, \cdot)$. However, the discrete logarithm problem (Definition 14) is *easy* in the first representation and *hard* in the other. Note that for every $g, h \in \mathbb{N}$, the discrete logarithm in the first representation is equivalent to solving a linear congruence, $gx \equiv h \pmod{p-1}$, which can be efficiently done by the *extended Euclid's algorithm.* On the other hand, the best known algorithms for solving the problem under the second representation are sub-exponential (Joux et al. [35, 133]), therefore, the problem is considered intractable for groups of sufficiently large cardinality.

Analyzing security in the Generic Group Model is asking whether certain primitive can be attacked regardless of the representation of the group. An attack that works for every representation is called *generic* and a primitive is said to be *generically secure* if there are no p.p.t. generic attacks against it.

Generic attacks are restricted to basic group operations, such as the *group-law*, equality checks and possibly some additional operations. There exist different models in the literature, that try to formalize this notion of generic attacks. The most popular ones are the model by Shoup [182] and the one by Maurer [156], which where proven to be equivalent [131].

In this thesis, we adopt Maurer's model, where groups are implemented by a third party and group operations are done via oracle access to this implementation using handles.

2.3.1 Example of proof in the Generic Group Model

We illustrate the ideas behind Maurer's model by proving the generic security of the CDH problem (Definition 15).

In this section we will argue the generic hardness of the CDH assumption, i.e., we will show that there does not exist a p.p.t. algorithm $\mathcal{A}$, that solves CDH with *significant* probability for every implementation of group $\mathbb{G}$.

For simplicity, let $p \in \mathbb{N}$ be prime and let $\mathbb{G}$ be a group[1] of order p. We consider an oracle $\mathcal{O}$ that implements $\mathbb{G}$ by using an arbitrary representation. More precisely, $\mathcal{O}$ samples g uniformly from $\mathbb{G}\backslash\{1_{\mathbb{G}}\}$ and a, b uniformly from $\mathbb{Z}_p$. It creates three handles, $\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3$ pointing to g, g^a, g^b respectively and provides $\mathcal{A}$ with these handles. Algorithm $\mathcal{A}$ is allowed to perform group operations, by querying two handles and getting a new handle pointing to a group element that corresponds to the sum of the contents of the provided handles. For example, $\mathcal{A}$ can ask for the group-law operation on handles h_1 and h_3, $\mathcal{O}$ will see that they contain g, g^b and will return to $\mathcal{A}$ a fresh handle, h_4, pointing to g^{b+1}.

[1] We could say "let $\mathbb{G}$ be *the* group of order p", because there exists only one (modulo isomorphism), however, the group admits many different representations.

Additionally, $\mathcal{A}$ is allowed to perform equality checks, e.g., it can ask for a comparison between handles h_3 and h_4 and $\mathcal{O}$ will answer back with "*false*", because the content of such handles is different, i.e., $g^b \neq g^{b+1}$.

In the so-called *generic experiment*, after several calls to the group operation oracle and several equality checks, $\mathcal{A}$ will finish the game by outputting a handle h^*. Algorithm $\mathcal{A}$ wins the experiment if h^* points to g^{ab}.

Now, consider a slightly modified experiment, that we call *symbolic experiment*, where the implementation of the group is done in a different way. Instead of creating pointers to group elements, the new oracle, $\mathcal{O}^{\mathsf{sym}}$ is creating pointers to polynomials in $\mathbb{Z}_p[A, B]$. In particular, the three original handles that $\mathcal{A}$ receives, $\mathsf{h}_1, \mathsf{h}_2, \mathsf{h}_3$, point to polynomials $1, A, B$ respectively. When a group operation is asked between two handles, $\mathcal{O}^{\mathsf{sym}}$ creates a new handle to the *sum of the polynomials* corresponding to the content of the handles. On the other hand, when an equality check is performed, the answer is given based on the comparison of polynomials over $\mathbb{Z}_p[A, B]$.

Note that this modification is imperceptible for $\mathcal{A}$, i.e., if it were interacting with $\mathcal{O}^{\mathsf{sym}}$ instead of $\mathcal{O}$, $\mathcal{A}$ would not have a way of recognizing that situation, except for a small detail: *equality checks*. It is possible that an equality check holds in the *generic experiment* and not in the *symbolic* one[2]. For example, an equality check between handles h_2 and h_3 could hold in the generic experiment if it happened the unlikely event of sampling a and b equal, uniformly from $\mathbb{Z}_p$. However, the same equality check does not hold in the symbolic experiment, because polynomial A is different from polynomial B. The only way for $\mathcal{A}$ to distinguish between an interaction with $\mathcal{O}$ and an interaction with $\mathcal{O}^{\mathsf{sym}}$ is to produce a *bad event*, that is, to perform an equality check that holds and should not hold in the symbolic model (in that case $\mathcal{A}$ would know it is interacting with $\mathcal{O}$). Also, note that $\mathcal{A}$ can predict by itself the equality checks from the symbolic model and detect the previous situation.

It is common to relate the $\mathcal{A}$'s ability of winning the generic experiment to the one of winning the symbolic experiment by the Schwartz-Zippel lemma.

Lemma 1 (Schwartz-Zippel). *Let $\mathbb{F}$ be a field and $f \in \mathbb{F}[X_1, \dots, X_n]$ be a non-zero polynomial in the variables $X_1, \dots, X_n$ with coefficients in $\mathbb{F}$. Let $d \geqslant 0$ be the degree of f and let S be a finite subset of $\mathbb{F}$. Then,*

$$\Pr\left[\, r_1, \dots, r_n \xleftarrow{\$} S : f(r_1, \dots, r_n) = 0 \,\right] \leqslant \frac{d}{|S|} \ .$$

Note that the lemma, applied to the field $\mathbb{F} = \mathbb{Z}_p$ and set $S = \mathbb{Z}_p$ tells us that the probability of any two linear polynomials in $\mathbb{Z}_p[A, B]$ being instantiated to the same value on uniformly sampled inputs is upper-bounded by $1/p$.

[2]Note that the converse cannot happen, i.e., all equality checks that hold in the *symbolic experiment* must hold in the *generic experiment*.

Now, let $\mathcal{A}$ be an algorithm performing q group-law queries to its oracle $\mathcal{O}$ (or $\mathcal{O}^{\mathsf{sym}}$). It will produce at most $q + 3$ different handles. In the case it performed all possible equality checks between these handles, that would be precisely $(q+3)(q+2)/2$ equality checks and therefore, by the *union-bound*, the *bad event* happening is upper-bounded by $(q + 3)(q + 2)/2p$.

Finally, note that no algorithm can win the symbolic experiment, since all operations preserve the degree of the polynomials stored in the handles. Therefore, from handles pointing to $1, A, B$, it is impossible to create a handle to polynomial AB.

We conclude that any p.p.t. generic algorithm performing q group-law operations in $\mathbb{G}$ has a probability of solving the CDH problem that is upper bounded by

$$\frac{q^2 + 5q + 6}{2p}$$

which is negligible λ, where $2^\lambda \approx |\mathbb{G}| = p$, because q must be polynomial in λ.

2.3.2 Why use the Generic Group Model

Informally, a problem is said to be *hard* if no *efficient* algorithm can solve it with *significant probability*. Algorithms are considered to be "*efficient*" if they run in polynomial time in the problem's size, while they are considered to succeed with "*significant probability*" when their probability of success is lower-bounded by the inverse of some polynomial in the problem's size.

If a problem is hard, sufficiently large instances of the problem are intractable with the existing algorithms, even with enormous computational power. There are some problems believed to be hard and widely used in cryptography, e.g., factoring integers, computing the discrete logarithm over elliptic curves... However, proving lower complexity bounds for these problems is extremely challenging with our currently available mathematical knowledge. Actually, proving meaningful lower bounds on the complexity of *hard problems* used in cryptography would require to solve the Millennium Prize problem **P** vs **NP**, formulated by Cook in 1971 [83], which results acutely difficult, as evidenced by the lack of progress in the last decades.

The GGM is a **natural** and **well-defined** framework that allows to prove security of cryptographic constructions without relying on hardness assumptions. Note that the security level provided by a prove in the GGM is at most as high as the security level that a proof in the standard model offers. That is because, a proof in the standard model must rely on assumptions that are *generically secure* and therefore, a proof in the standard model implies *generic security* of the cryptographic primitive.

It is common that proofs in the standard model rely on non-standard hardness assumptions. In such cases, it is crucial to argue that these hardness assumptions meet certain **minimal requirements**. The accepted method for arguing the validity of such non-standard assumptions is to prove their security in the Generic Group Model.

Additionally, note that for certain implementations of groups (like some elliptic curves) it is unknown how to exploit the algebraic structure of the group representation. In such situations, the **best known algorithms** for facing certain problems **are generic**. When cryptographic primitives are implemented in these groups, security in the GGM *can be considered* enough for practical use.

The GGM is an especially amenable framework for proving lower bounds on the size and performance of secure cryptographic schemes. For example, it has been applied to derive several lower bounds on the complexity of Structure-Preserving Signatures [3, 6, 7, 44].

Finally, another incentive for considering the GGM is the fact that constructions in this model are usually **smaller** or **more efficient** than secure constructions in the standard model, where additional components are commonly needed to argue security. An example is the *Petit IBE* [191], which was proven adaptively secure in the composite order setting of bilinear groups. The additional structure that the composite order group provides makes it possible to reduce the security of the Petit IBE to the hardness of a subgroup membership assumption. However, without such a structure, the same argument does not extend to the prime order case, which is preferable for size and efficiency reasons. In Chapter 5 we provide a proof of security of the prime-order version of the Petit IBE in the Generic Group Model. Proving its security in the standard model remains as an open problem. Structure-Preserving Signatures are other example where constructions in the GGM are more efficient. We refer to [1, Table 1] for a comparison between lower and upper bounds for different classes of SPS schemes. For example, the authors show that secure SPS schemes for bilateral messages in the Type $\mathbb{III}$ setting must contain at least 6 group elements in the signature to be proven secure in the *standard model* (under non-interactive assumptions). On the other hand, there exist constructions with only 3 group elements in the signature that achieve *generic security*.

3
Unbounded Analysis of Constructions in the Generic Group Model

Programming is the art of algorithm design
and the craft of debugging errant code.

Ellen Ullman, 2013

In this chapter, we develop a new method to automatically prove security statements in the Generic Group Model as they occur in actual papers. We start by defining a *general language* to describe security definitions, a *class of logical formulas* that characterize how an adversary can win, and a *translation* from security definitions to such formulas. We prove a Master Theorem that relates the security of the construction to the existence of a solution for the associated logical formulas. Moreover, we define a constraint solving algorithm that proves the security of a construction by proving the absence of solutions.

We implement our approach in a fully automated tool, the gga^{∞} tool, and use it to verify different examples from the literature. The results improve on the tool by Barthe et al. [43, 44]: for many constructions, gga^{∞} succeeds in proving standard (unbounded) security, whereas Barthe's tool is only able to prove security for a small number of oracle queries.

3.1 Introduction

The Generic Group Model provides an algebraic setting for describing a wide class cryptographic primitives and assumptions, and is supported by Master Theorems that give a purely algebraic condition that ensures the security of primitives in this model. Very roughly, the proof of the Master Theorems uses the Schwartz-Zippel lemma to prove a security reduction between the Generic Group Model and a Symbolic Generic Group Model, in which the security experiment is purely deterministic. Security in the Symbolic Generic Group Model is trivially equivalent to a purely algebraic condition. For instance, the algebraic condition for a decisional assumption requires to prove that the two sets of polynomials extracted from the left and right games have the same linear dependencies. Therefore, and unavoidably, the difficulty of checking the algebraic condition increases as the assumption becomes more complex, as witnessed by unfortunate failures [105, 128, 185]. For some recent hypotheses, several pages of error-prone calculations are required for proving that the algebraic condition holds, and several authors have used computer algebra systems to carry part of the verifications. These examples suggest the importance of building general tools to assist proofs of security assumptions in the Generic Group Model. One such tool is the Generic Group Analyzer [43], which uses SMT solvers and computer algebra systems to analyze DDH-like assumptions. The tool takes as input a description of an assumption and either returns an algebraic attack or a concrete probability bound if the assumption is secure. The Generic Group Analyzer primarily works for non-interactive assumptions, in which the adversary can only call the oracles which perform the algebraic operations.

The Generic Group Model can also be used for proving the security of cryptographic constructions, such as Digital Signature schemes and algebraic Message Authentication Codes, against algebraic attacks. In this context, the adversary has access to oracles for performing signatures, verification, *etc.* The Generic Group Analyzer also provides support for such problems, but is inherently limited to oracles which do not take handles to group elements as inputs. This support can be used for analyzing simple interactive assumptions. Subsequent extensions of the Generic Group Analyzer overcome this limitation by providing support for oracles that take handles as inputs, and by allowing adversaries to make a bounded number of oracle queries [44]. Using this extension, Barthe et al. [44] synthesize (in the Type $\mathbb{II}$ setting) structure-preserving signatures that are secure against adversaries that can make a bounded number of signing queries. Their approach is based on an algebraic characterization of security, using a vector space whose dimension increases by one for each query. Therefore, their approach is limited to a small number of queries, and an alternative approach must be used for proving security notions which do not impose a bound on the number of queries.

Building on these two contributions, we implement an analyzer which subsumes the Generic Group Analyzer for interactive assumptions and is able to analyze many cryptographic constructions, including signatures and message authentication codes.

3.1.1 Technical overview

In more detail, our contributions are as follows.

First, we define a language to express security experiments in the Generic Group Model where the adversary can make an unbounded number of queries to oracles; moreover, our model allows oracles to take group values as inputs. In addition, we define a rich language of winning conditions. We then establish a Master Theorem, which states that a generic algorithm is secure with respect to a security goal expressed using our language of winning conditions, if the constraint system extracted from the security experiment, given by the algorithm and the winning condition, has no computable solution. Informally, the notion of computable solution provides an algebraic counterpart to the notion of deducibility used in the symbolic (a.k.a. Dolev-Yao) approach to cryptography; more technically, this notion is based on an inductive definition of the adversary's knowledge throughout execution of the algorithm. From a broader perspective, our Master Theorem provides a novel light on the relationship between different cryptographic models, by showing a general relationship between the Generic Group Model and the symbolic model. Note that, for the sake of simplicity, we focus on group settings with bilinear pairings; however, we believe that our model and Master Theorem can naturally extend to the case of multilinear maps.

Second, we define an automated method for proving the absence of computable solutions of constraint systems. Our language of constraints supports algebraic expressions that are generally not considered by prior work on the symbolic model. Therefore, we cannot use previous constraint-solving methods developed for reasoning about cryptographic protocols in the symbolic model. Rather, we define a specialized method which combines general purpose algebraic computations and specialized steps. The algebraic computations are performed using Gröbner bases, whereas the specialized steps include simplifications related to big operators and case distinctions. The latter can be used to add new equations to constraint systems and thus to trigger new simplifications. Case distinctions are an essential ingredient for the success of our method: they yield compact proofs that follow the structure of pen-and-paper arguments found in the literature. Of course, the use of case distinctions is not new in automated deduction; it is at the core of Staalmarck's method, an empirically successful method for propositional logic. However, its use in our setting appears to be new.

Third, we implement our method and evaluate its effectiveness on a sizable set of case studies. Our tool uses off-the-shelf computer algebra systems to perform Gröbner bases computations. However, it draws its efficiency from a finely tuned heuristics for carrying case distinctions. We evaluate our tool on structure-preserving signatures, in all settings (Type I, Type II and Type III). Our tool is able to prove unbounded security of many structure-preserving signatures from the literature, as well as of the algebraic MACs from Chase, Meiklejohn and Zaverucha [74], and of the short randomizable signatures from Pointcheval and Sanders [167]. Furthermore, it also proves unbounded security for most of the examples proved 2-time secure in [44] (these examples were generated automatically using synthesis techniques). Moreover, we also adapt the synthesis tool from [44] to generate structure-preserving signatures in the Type III setting and use our tool to prove security for more than a 100 such schemes.

3.1.2 Related work

The Generic Group Model was introduced by Nechaev [160], Shoup [182] and Maurer [156], following distinct but equivalent approaches [131]. The original approach by Nechaev and Shoup lets the adversary access a randomly selected representation of group elements; in contrast, Maurer's approach requires the adversary to perform all algebraic operations via oracles, and uses handles as symbolic representations of group elements known to the adversary. We opt for the second approach, for its distinctively symbolic flavour. These works establish lower complexity bounds for the generic discrete logarithms and the generic hardness of Diffie-Hellman like assumptions. As for us, they use the Schwartz-Zippel lemma for transforming their original problem into an algebraic one. This approach was extended by Boneh, Boyen and Goh [63]. First, their Generic Group Model focuses on bilinear groups. Second, they consider a general class of assumptions, and provide the first Master Theorem, which provides a systematic method for extracting algebraic conditions of security from assumptions. Their Master Theorem was subsequently extended in many directions. The most relevant works are those that involve the use of computer tools for verifying algebraic conditions. Notably, Freeman [104] verifies the hardness of two assumptions using Magma.

Shoup [182] and Schnorr and Jakobsson [176, 177] were among the first to use the Generic Group Model for proving the security of crytographic constructions. Specifically, Shoup proves (generic) security of an identification scheme, whereas Schnorr and Jakobsson consider signed ElGamal encryption and blind discrete log signatures. More recently, the Generic Group Model has also become an important tool for analyzing the security of pairing-based cryptographic constructions. Chase, Meiklejohn and Zaverucha [74] propose a class of algebraic MACs and prove their generic security. Several authors use the Generic Group Model for proving the generic security of structure-preserving signatures [2]. Groth [118] proposes new fully-structure-preserving signatures [8] and proves their generic security. Similarly, Fuchsbauer, Hanser and Slamanig [106] define a structure-preserving signature on equivalence classes and prove its generic security. Furthermore, the Generic Group Model gives a convenient setting for establishing lower bounds on the complexity of structure-preserving signatures [3, 6, 7, 44]. In a similar spirit, the Generic Group Model has been used for proving the correctness of translations of signature schemes from Type I to Type Ⅲ [5, 7, 12].

It is also worth pointing to a recent examination of the efficiency of pairing-based implementations. Based on a practical evaluation of the efficiency of state-of-the-art implementations of pairings, Chatterjee and Menezes [75] argue that Type Ⅲ pairings are more efficient than their Type Ⅱ counterparts, and should be favoured in implementations. Their observation justifies the need to transpose existing results and tools for the Type Ⅱ setting to the Type Ⅲ setting, and has motivated the application of our methods to the latter.

Several works have developed or used tools for reasoning about the Generic Group Model. As already mentioned, the Generic Group Analyzer [43] implements an automated method for analyzing assumptions. Moreover, a subsequent extension of the analyzer [44] supports the automated analysis of security of structure-preserving security against adversaries that make a bounded number of queries. In practice, the tool only terminates for small bounds on the number of queries. While these works are the most closely related to ours, there have been previous works that apply computer tools to the Generic Group Model. Barthe, Cederquist and Tarento [40, 48] were the first to use formal verification tools for analyzing the security of hardness assumptions and cryptographic constructions in the Generic Group Model. Their work uses the Coq proof assistant, providing no support for automation. Freeman [104] reports on using computer algebra systems to prove the validity of new hardness assumptions in the Generic Group Model. Beyond the GGM, there exist several tools for synthesizing constructions, such as encryption schemes, modes of operations, tweakable blockciphers, and structure-preserving signatures in the Type Ⅱ setting [42, 44, 125, 155], automated transformation of existing constructions, including signature schemes [5, 12, 14], and verification of security proofs [46, 47, 58]. In particular, [47] introduce AutoG& P, a highly automated framework for proving the security of pairing-based cryptographic primitives; the focus of [47] is on encryption schemes, but their methods are also applicable to signatures and MACs. AutoG& P and gga^{∞} are complementary in two different ways. First, gga^{∞} focuses on full automation in the GGM while AutoG& P provides partial automation in the Standard model. Second, and more interestingly, some of our techniques for equational reasoning could be used to achieve more automation in AutoG& P, whereas it could be possible to use techniques from AutoG& P as a fallback solution when full automation fails in gga^{∞}.

3.1.3 Notation

For a set S, we define $aS = \{as \mid s \in S\}$ and $SS' = \{ss' \mid s \in S \wedge s' \in S'\}$. We write S^* to denote vectors of elements in S. We use $\boldsymbol{v}$ to denote a vector and $\boldsymbol{v}_{(i)}$ to denote the i-th element. We assume given a set of *uniform variables* UVar, a set of *handle variables* $\mathsf{HVar} = \mathsf{HVar}_1 \uplus \mathsf{HVar}_2 \uplus \mathsf{HVar}_{\mathsf{t}}$, a set of *parameter variables* PVar, and a set of *index variables* IVar. We use $ty(h) \in \{1, 2, \mathsf{t}\}$ to denote the type of a handle variable, i.e., $ty(h) = i$ iff $h \in \mathsf{HVar}_i$.

We use $R[\boldsymbol{X}^{\pm 1}]$ to denote the set of *Laurent polynomials* over the ring R with variables in $\boldsymbol{X}$. We also use the shorthand $R[\boldsymbol{Y}, \boldsymbol{X}^{\pm 1}]$ for $(R[\boldsymbol{Y}])[\boldsymbol{X}^{\pm 1}]$ to denote nested polynomial rings. We use a similar notation $Mon[\boldsymbol{X}^{\pm 1}, \boldsymbol{Y}]$ for *Laurent monomials*. We write $\mathsf{deg}_V(M)$ to denote the *degree* of V in the Laurent monomial M. We write $\mathsf{coeff}_M(F)$ to denote the coefficient of the Laurent monomial M in the Laurent polynomial F.

For a term t possibly containing variables, we write $t[x \mapsto t']$ to denote the result of substituting all occurrences of the variable x in t with t'. A context C is a term with a distinguished variable which denotes a hole that can be filled in by an arbitrary term. We assume the hole occurs exactly once in a context. We use $C[t]$ to denote the term obtained by plugging t into $C's$ hole.

3.2 Translating Security Experiments into Constraints

In this section, we first present a language to define security experiments in the Generic Group Model. Next, we define the language of winning constraints. Winning constraints are formulas that characterize if an adversary can win a security experiment. Finally, we present a translation procedure from security experiments to winning constraints.

3.2.1 Security Experiment Definition

We first present the language that we use to define security experiments. Afterwards, we define the corresponding games in the GGM and the symbolic group model (see [43]). We will exploit that the generic and symbolic games are indistinguishable and use the symbolic game to perform our analysis.

Definition 19. *(Security experiment) A* security experiment *is defined by a tuple $SE = (t, ainp, odef, wcond)$ where*

- *the* group type *is defined by $t \in \{\mathrm{I}, \mathrm{II}, \mathrm{III}\}$,*
- *the* adversary input *is defined by $ainp = (\boldsymbol{X}, (\boldsymbol{F}_1, \boldsymbol{F}_2, \boldsymbol{F}_{\mathsf{t}}))$ for*

- ◦ global uniform variables $\boldsymbol{X} \in \mathsf{UVar}^*$ *and*
- ◦ input polynomials $\boldsymbol{F_i} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}]^*$,

- *the* oracle *is defined by odef* $= (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_\mathsf{t}}))$ *for*
  - ◦ arguments $\boldsymbol{a} \in \mathsf{PVar}^*$ *and* oracle handles $\boldsymbol{h} \in \mathsf{HVar}^*$,[1]
  - ◦ oracle uniform variables $\boldsymbol{R} \in \mathsf{UVar}^*$, *and*
  - ◦ oracle polynomials $\boldsymbol{H_i} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}]^*$, *and*

- *the* winning condition *is defined by wcond* $= (\widehat{\boldsymbol{a}}, \widehat{\boldsymbol{h}}, \boldsymbol{W}^{=}, \boldsymbol{W}^{\neq})$ *for*
  - ◦ winning arguments $\widehat{\boldsymbol{a}} \in \mathsf{PVar}^*$ *and* winning handles $\widehat{\boldsymbol{h}} \in \mathsf{HVar}^*$, *and*
  - ◦ winning (in)equalities $\boldsymbol{W}^{=}, \boldsymbol{W}^{\neq} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}, \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{h}}]^*$.

Intuitively, the *adversary input* represents the values given initially to the adversary. This usually includes the public parameters and the public keys. The *oracle* is defined by *arguments* and *oracle handles* that represent the oracle input; *uniform variables* that denote randomness sampled by the oracle; and *oracle polynomials* that denote the oracle response. Finally, the *winning condition* is defined by *winning arguments* that represent the forgery that the adversary must produce; and *winning (in)equalities* that characterize valid forgeries.

We define the corresponding generic group game $\mathsf{G}^{\mathsf{gen}}(SE)$ as follows:

1. Sample the vector $\boldsymbol{x} \in (\mathbb{F}_p^\times)^{|\boldsymbol{X}|}$, compute the adversary inputs $[\![\boldsymbol{F_i}(\boldsymbol{x})]\!]_i \in \mathbb{G}_i^{|\boldsymbol{F_i}|}$ (for $i \in \{1, 2, \mathsf{t}\}$), and call the adversary $\mathcal{A}$ with the corresponding handles.

2. The adversary $\mathcal{A}$ can perform q_g queries to perform group operations (for group type t), an unbounded number of equality queries, and q queries to an oracle that implements *odef*. The oracle for *odef* takes scalars $\boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|}$ for $\boldsymbol{a}$ and a vector of handles to group elements $\boldsymbol{U}$ for $\boldsymbol{h}$. We use $\boldsymbol{u}$ to denote the discrete logarithms of $\boldsymbol{U}$, i.e., for all $j \in [|\boldsymbol{h}|]$, $[\![\boldsymbol{u}_{(j)}]\!]_i = \boldsymbol{U}_{(j)}$ where $i = ty(\boldsymbol{h}_{(j)})$. Then it samples $\boldsymbol{r} \in (\mathbb{F}_p^\times)^{|\boldsymbol{R}|}$ and returns handles to $[\![\boldsymbol{H_i}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{r})]\!]_i \in \mathbb{G}_i^{|\boldsymbol{H_i}|}$. We use $\boldsymbol{v}^{(j)}$, $\boldsymbol{u}^{(j)}$, $\boldsymbol{r}^{(j)}$ to denote the corresponding values used in the j-th query.

3. The adversary $\mathcal{A}$ returns scalars $\widehat{\boldsymbol{v}} \in \mathbb{F}_p^{|\widehat{\boldsymbol{a}}|}$ for $\widehat{\boldsymbol{a}}$ and handles to group elements $\widehat{\boldsymbol{U}}$ for $\widehat{\boldsymbol{h}}$. Again, we denote the discrete logarithms of $\widehat{\boldsymbol{U}}$ with $\widehat{\boldsymbol{u}}$. The adversary wins if for $\bowtie \in \{=, \neq\}$, $w \in \boldsymbol{W}^{\bowtie}$, and $j \in [q]$, it holds that $w(\boldsymbol{x}, \boldsymbol{r}^{(j)}, \boldsymbol{v}^{(j)}, \boldsymbol{u}^{(j)}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}) \bowtie 0$.

[1]Handle variables are typed, i.e., for all $j \in [|\boldsymbol{h}|]$, it holds that $ty(\boldsymbol{h}_{(j)}) \in \{1, 2, \mathsf{t}\}$.

Note that additional care must be taken to ensure that the oracles and winning conditions are efficiently computable using scalar multiplication, addition, application of isomorphisms, and application of bilinear maps. For example, it is possible to specify an oracle that takes a handle to an element $[\![v]\!]_{\mathsf{t}} \in \mathbb{G}_{\mathsf{t}}$ and returns $[\![v]\!]_1 \in \mathbb{G}_1$, which cannot be efficiently computed in most bilinear groups of interest.

The symbolic game $\mathsf{G}^{\mathsf{sym}}(SE)$ is defined similarly, but internally uses Laurent polynomials $f(\boldsymbol{X})$ instead of group elements $[\![f(\boldsymbol{x})]\!]_i$. It is completely deterministic since it uses formal variables $\boldsymbol{X}$ to represent the initially sampled values and indexed formal variables $\boldsymbol{R}^{(j)}$ to represent the values sampled in the oracle.

Formally, we define $\mathsf{G}^{\mathsf{sym}}(SE)$ as follows:

1. Store the polynomials $\boldsymbol{F_i}(\boldsymbol{X}) \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}]^{|\boldsymbol{F_i}|}$ in the list for the group $\mathbb{G}_i$ (for $i \in \{1, 2, \mathsf{t}\}$) and call the adversary $\mathcal{A}$ with the corresponding handles.

2. The oracles for group operations and equality checks provide the same interface as in the generic model, but perform all computations in the ring of Laurent polynomials. The oracle for *odef* takes (in the j-th query) scalars $\boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|}$ for $\boldsymbol{a}$ and handles to polynomials
$$\boldsymbol{u} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(j-1)})^{\pm 1}]^{|\boldsymbol{h_i}|}$$
for $\boldsymbol{h}$. It returns handles to polynomials
$$\boldsymbol{H_i}(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{R^{(j)}}) \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(j)})^{\pm 1}]^{|\boldsymbol{H_i}|}.$$

3. The adversary $\mathcal{A}$ returns scalars $\widehat{\boldsymbol{v}} \in \mathbb{F}_p^{|\widehat{\boldsymbol{a}}|}$ for $\widehat{\boldsymbol{a}}$ and handles to polynomials
$$\widehat{\boldsymbol{u}} \in \mathbb{Z}[\boldsymbol{X}^{\pm 1}, (\boldsymbol{R}^{(1)})^{\pm 1}, \ldots, (\boldsymbol{R}^{(q)})^{\pm 1}]^{|\widehat{\boldsymbol{h}}_i|}$$
for $\widehat{\boldsymbol{h}}$. He wins if for $\bowtie\ \in \{=, \neq\}$, $w \in \boldsymbol{W}^{\bowtie}$, and $j \in [q]$, it holds that $w(\boldsymbol{X}, \boldsymbol{R}^{(j)}, \boldsymbol{v}^{(j)}, \boldsymbol{u}^{(j)}, \widehat{\boldsymbol{v}}, \widehat{\boldsymbol{u}}) \bowtie 0$.

Example 1. *We can formalize the* EUF-CMA *security of the scheme in Figure 3.1 using the security experiment $SE = (t, ainp, odef, wcond)$ defined as follows:*

- *the group type is $t = \mathrm{III}$*
- *the adversary input is $ainp = (\boldsymbol{X}, (\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_{\mathsf{t}}}))$ where*
    - ◦ $\boldsymbol{X} = (v, w)$ *(for $v, w \in \mathsf{UVar}$), $\boldsymbol{F_1} = (1, v, w)$, $\boldsymbol{F_2} = (1)$, $\boldsymbol{F_{\mathsf{t}}} = (1)$*
- *the* oracle *is $odef = (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_{\mathsf{t}}}))$ where*
    - ◦ $\boldsymbol{a} = ()$, $\boldsymbol{h} = (m)$ *(for $m \in \mathsf{HVar}_2$),*

Setup $\mathcal{P}(1^\lambda)$: Return $PP = (p, \mathbb{G}_1, \mathbb{G}_2, [\![1]\!]_1, [\![1]\!]_2, \mathbb{G}_t, e) \leftarrow \mathcal{G}(1^\lambda)$ where $\mathcal{G}$ is a polynomial time algorithm that on input 1^λ returns a description of a bilinear map in the Type III setting with groups of order p for a λ-bit prime p.

Key generation $\mathcal{K}(PP)$:
Choose $v, w \leftarrow \mathbb{F}_p^\times$ and compute $VK = (PP, V, W)$ and $SK = (PP, v, w)$ as

$$V \leftarrow [\![v]\!]_1 \text{ and } W \leftarrow [\![w]\!]_1 .$$

Signing $\mathcal{S}_{SK}(M)$:
For $M = [\![m]\!]_2 \in \mathbb{G}_2$ choose $r \leftarrow \mathbb{F}_p^\times$ and compute the signature (T_1, T_2, S) as

$$T_1 \leftarrow [\![r]\!]_1, \ T_2 = [\![r]\!]_2, \text{ and } S \leftarrow [\![m\, v + w + r^2]\!]_2 .$$

Verification $\mathcal{V}_{VK}(M, (T_1, T_2, S))$:
Accept if, and only if, $T_1 \in \mathbb{G}_1$, $M, T_2, S \in \mathbb{G}_2$,

$$e([\![1]\!]_1, S) = e(V, M) + e(W, [\![1]\!]_2) + e(T_1, T_2), \text{ and } e(T_1, [\![1]\!]_2) = e([\![1]\!]_1, T_2).$$

Figure 3.1: SPS-scheme from [75] in Type III setting.

- ◦ $\boldsymbol{R} = (r)$ *(for* $r \in \mathsf{UVar}$*),*
- ◦ $\boldsymbol{H_1} = (r)$, $\boldsymbol{H_2} = (r,\ mv + w + r^2)$, $\boldsymbol{H_t} = ()$

- *the winning condition is* $wcond = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{h}}, \boldsymbol{W}^=, \boldsymbol{W}^{\neq})$ *where*
  - ◦ $\hat{\boldsymbol{a}} = ()$, $\hat{\boldsymbol{h}} = (\hat{m}, \hat{t}_1, \hat{t}_2, \hat{s})$ *and (for* $\hat{t}_1 \in \mathsf{HVar}_1$ $\hat{m}, \hat{t}_2, \hat{s} \in \mathsf{HVar}_2$*),*
  - ◦ $\boldsymbol{W}^= = (\hat{s} - \hat{m}v - w - \hat{t}_1\hat{t}_2,\ \hat{t}_1 - \hat{t}_2)$, $\boldsymbol{W}^{\neq} = (\hat{m} - m^{(j)})$. ■

3.2.2 Winning Constraints

We first define the language of winning constraints, a class of formulas that can be used to characterize if an adversary can win the symbolic game $\mathsf{G}^{\mathsf{sym}}(SE)$. Then we define the set of solutions of a winning constraint and present a set of simplification rules that preserve the set of solutions.

Definition 20. *(Winning constraints) The language of winning constraints is defined by the grammar given in Figure 3.2. We distinguish between* bound index variables *and* free index variables *depending on whether they are bound by* $\forall/\Sigma$*. We write* $ivars(\mathcal{C})$ *to denote the free index variables in the constraint* $\mathcal{C}$*.*

$\mathcal{C} ::= \exists i \notin K.\mathcal{C} \mid \mathcal{C}$	constraint
$\mathcal{C}' ::= \mathcal{C}' \wedge \mathcal{C}' \mid \forall k \notin K.\mathcal{C}' \mid \mathcal{E} = 0 \mid \mathcal{E} \neq 0$	non-existential constraint
$\mathcal{E} ::= \mathcal{E} + \mathcal{E} \mid \mathcal{E} * \mathcal{E} \mid -\mathcal{E} \mid \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$ $\mid \sum_{k \notin K} \mathcal{E} \mid \mathcal{R} \mid \mathcal{R}^{-1} \mid \mathcal{P} \mid \mathcal{V} \mid 1 \mid 0$	expression
$\mathcal{M} ::= \mathcal{M} * \mathcal{M} \mid \mathcal{R} \mid \mathcal{R}^{-1} \mid 1$	monomial over uniform variables
$\mathcal{R} ::= R_{[k]} \mid R$	(indexed) uniform variable ($R \in \mathsf{UVar}$)
$\mathcal{P} ::= \rho_{[k]} \mid \rho$	(indexed) parameter ($\rho \in \mathsf{PVar}$)
$\mathcal{V} ::= Y_{[k]}$	indexed handle variable ($Y \in \mathsf{HVar}$)

Figure 3.2: Grammar for winning constraints (for $k \in \mathsf{IVar}, K \subset \mathsf{IVar}$). For every $\mathsf{Coeff}(\mathcal{E})$, $\mathcal{E}$ does not contain the symbol Coeff.

Intuitively, atomic constraints $\mathcal{E} = 0$ represent polynomial equalities. In the quantifications $\forall k \notin K$ and $\sum_{k \notin K}$, the index variable k ranges over all elements in $[q]$ except for the valuations of the index variables in K. Uniform variables $R/R_{[k]}$ are treated like formal variables, parameters $\rho/\rho_{[k]}$ can be instantiated with integers, handle variables $Y_{[k]}$ can be instantiated with Laurent polynomials over uniform variables, and the arithmetic operations are interpreted in the ring of Laurent polynomials over $\mathbb{F}_p$ for a prime p. An expression $\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$ represents the coefficient of the monomial $\mathcal{M}$ in the expression $\mathcal{E}$ after the parameters and handle variables in $\mathcal{E}$ are instantiated. The resulting Laurent polynomial after instantiation contains only (indexed) uniform variables. Formally, the set of solutions of a winning constraint is defined as follows.

Definition 21. *(Solutions of winning constraints) A* structure $s = (p, q, \sigma, \delta, \chi, \xi)$ *for a prime number p, a natural number q, a valuation $\sigma : \mathsf{IVar} \to [q]$ for (free) index variables, valuations $\delta : \mathsf{PVar} \to \mathbb{F}_p$ and $\chi : \mathsf{PVar} \times [q] \to \mathbb{F}_p$ for the parameters, and a valuation $\xi : \mathsf{HVar} \times [q] \to \mathbb{F}_p[\mathsf{UVar}^{\pm 1}, \mathsf{UVar}_{[1]}^{\pm 1}, \ldots, \mathsf{UVar}_{[q]}^{\pm 1}]$ for the handle variables is a* solution *for a winning constraint $\mathcal{C}$ if $eval_s(\mathcal{C}) = true$ for the function eval defined in Figure 3.3.*

3.2.3 Translation from Security Experiments to Winning Constraints

We define the translation function to convert a security experiment definition into winning constraints. The translation is sound and complete with respect

$$
eval_s(c) = \begin{cases}
eval_{s_{k,K,1}}(c') \vee \ldots \vee eval_{s_{k,K,q-|K|}}(c') & \text{for} \quad c = \exists k \notin K.\, c' \\
eval_{s_{k,K,1}}(c') \wedge \ldots \wedge eval_{s_{k,K,q-|K|}}(c') & \text{for} \quad c = \forall k \notin K.\, c' \\
eval_s(c_1) \otimes eval_s(c_2) & \text{for} \quad c = c_1 \otimes c_2 \\
eval_{s_{k,K,1}}(c') + \ldots + eval_{s_{k,K,q-|K|}}(c') & \text{for} \quad c = \sum_{k \notin K} c' \\
\delta(\rho) & \text{for} \quad c = \rho \\
\chi(\rho, \sigma(i)) & \text{for} \quad c = \rho_{[i]} \\
\xi(Y, \sigma(i)) & \text{for} \quad c = Y_{[i]} \\
R^e & \text{for} \quad c = R^e, e \in \{+1, -1\} \\
R^e_{[\sigma(i)]} & \text{for} \quad c = R^e_{[i]}, e \in \{+1, -1\} \\
\mathsf{coeff}_{\sigma(\mathcal{M})}(eval_s(\mathcal{E})) & \text{for} \quad c = \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) \\
c & \text{for} \quad c \in \{0, 1\}
\end{cases}
$$

Figure 3.3: Definition of the evaluation function $eval_s$ for $s = (p, q, \sigma, \delta, \chi, \xi)$, where $R \in \mathsf{UVar}$, $\otimes \in \{=, \neq, \wedge, *, +\}$ are interpreted as the corresponding boolean operations/arithmetic operations in the ring of Laurent polynomials over $\mathbb{F}_p$ and $s_{k,K,i}$ defined as follows. Let $\{v_1, \ldots v_{q-|K|}\} = [q] \backslash \sigma(K)$, then $s_{k,K,i} = (p, q, \sigma', \delta, \chi, \xi)$ where $\sigma' = \sigma[k \mapsto v_i]$ for $i \in \{1, \ldots, q - |K|\}$.

to a certain class of solutions. Roughly, this means that there is an efficient attacker[2] on the security experiment in the Generic Group Model with non-negligible winning probability iff there is a solution for the translated winning constraints where handle variables are instantiated with "computable" Laurent polynomials.

To simplify the presentation, we assume that for all security experiments in Type II, it holds that $\boldsymbol{F_2} \subseteq \boldsymbol{F_1}$ and $\boldsymbol{H_2} \subseteq \boldsymbol{H_1}$ which allows us to ignore the isomorphism Ψ. Similarly, we assume for Type I that $\boldsymbol{F_1} = \boldsymbol{F_2}$ and $\boldsymbol{H_1} = \boldsymbol{H_2}$ which allows us to ignore that $\mathbb{G}_1 = \mathbb{G}_2$.

First, note that $\boldsymbol{W}^{\bowtie} \subset \mathbb{Z}[\boldsymbol{X}^{\pm 1}, \boldsymbol{R}^{\pm 1}, \boldsymbol{a}, \boldsymbol{h}, \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{h}}]$ where $\boldsymbol{X}, \boldsymbol{R} \in \mathsf{UVar}^*$, $\boldsymbol{a}, \widehat{\boldsymbol{a}} \in \mathsf{PVar}^*$, and $\boldsymbol{h}, \widehat{\boldsymbol{h}} \in \mathsf{HVar}^*$. For an index variable $j \in \mathsf{IVar}$, we write $\boldsymbol{R}_{[j]}$ to denote the vector $(\boldsymbol{R}_{(1)[j]}, \ldots, \boldsymbol{R}_{(|\boldsymbol{R}|)[j]})$ of indexed uniform variables. Similarly, we write $\boldsymbol{a}_{[j]}$ and $\boldsymbol{h}_{[j]}$. For our translation, we instantiante each winning handle variable $\widehat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar}_1 \cup \mathsf{HVar}_2$ with a linear combination of polynomials in the adversary input and in the oracle output. Formally, we define the vector $\boldsymbol{E}$ of expressions as follows. For $u \in [|\widehat{\boldsymbol{h}}|]$ such that $\widehat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar}_1$ and $l = |\boldsymbol{H_1}|$, we

[2]More precisely, an attacker that performs a polynomial number of queries q_g and q.

define

$$\begin{aligned}\boldsymbol{E}_{(u)} = {} & \rho^{(1,u,1)}\boldsymbol{F_1}_{(1)}(\boldsymbol{X}) + \ldots + \rho^{(1,u,|\boldsymbol{F_1}|)}\boldsymbol{F_1}_{(|\boldsymbol{F_1}|)}(\boldsymbol{X}) + \\ & \sum_k \tau_{[k]}^{(1,u,1)}\boldsymbol{H_1}_{(1)}(\boldsymbol{X}, \boldsymbol{R}_{[k]}, \boldsymbol{a}_{[k]}, \boldsymbol{h}_{[k]}) + \ldots + \\ & \sum_k \tau_{[k]}^{(1,u,l)}\boldsymbol{H_1}_{(l)}(\boldsymbol{X}, \boldsymbol{R}_{[k]}, \boldsymbol{a}_{[k]}, \boldsymbol{h}_{[k]})\end{aligned}$$

where $\rho^{(1,u,n)}$ and $\tau^{(1,u,n)}$ are distinct fresh parameter variables. For $u \in [|\widehat{\boldsymbol{h}}|]$ such that $\widehat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar}_2$, we define $\boldsymbol{E}_{(u)}$ analogously. For $u \in [|\widehat{\boldsymbol{h}}|]$ such that $\widehat{\boldsymbol{h}}_{(u)} \in \mathsf{HVar}_\mathsf{t}$, we define $\boldsymbol{E}_{(u)}$ analogously additionally taking products of polynomials from $\mathbb{G}_1$ and $\mathbb{G}_2$ into account. We define the winning constraint derived from *SE* as

$$\mathit{toConstr}(SE) = \bigwedge_{w \in \boldsymbol{W}^{\bowtie}} \forall j. \left(w(\boldsymbol{X}, \boldsymbol{R}_{[j]}, \boldsymbol{a}_{[j]}, \boldsymbol{h}_{[j]}, \widehat{\boldsymbol{a}}, \boldsymbol{E}) \bowtie 0\right) \ .$$

A priori, the notion of solution for winning constraints does not restrict the set of Laurent polynomials that can be used to instantiate the handle variables in $\boldsymbol{h}_{[j]}$. Since we are only interested in solutions where the instantiations of handle variables are computable, we now define the notion of constrained solution.

Definition 22. *(Constrained solutions of winning constraints) A solution is* constrained *by sequences of sets $\{K_j^{(i)}\}_{j\in\mathbb{N}}$ of Laurent polynomials (for $i \in \{1, 2, \mathsf{t}\}$) if for all $i \in \{1, 2, \mathsf{t}\}$, $Y \in \mathsf{HVar}_i$, and $j \in [q]$, it holds that $\xi(Y, j) \in K_j^{(i)}$.*

Since we are interested in solutions constrained by computable Laurent polynomials, we next define the sequences of computable polynomials. We use $\langle S\rangle$ to denote the vector space over $\mathbb{F}_p$ generated by S.

Definition 23. *(Computable polynomials) The sequences of* computable polynomials *for a security experiment*

$$SE = (t, \boldsymbol{X}, (\boldsymbol{F_1}, \boldsymbol{F_2}, \boldsymbol{F_t})), (\boldsymbol{a}, \boldsymbol{h}, \boldsymbol{R}, (\boldsymbol{H_1}, \boldsymbol{H_2}, \boldsymbol{H_t})), \mathit{wcond})$$

are defined as follows:

$$\begin{aligned}\mathcal{K}_0^{SE,(i)} &= \langle \mathit{toSet}(\boldsymbol{F_i})\rangle && \text{for } i \in \{1, 2\} \\ \mathcal{K}_0^{SE,(\mathsf{t})} &= \langle \mathit{toSet}(\boldsymbol{F_t}) \cup (\mathcal{K}_0^{SE,(1)} * \mathcal{K}_0^{SE,(2)})\rangle\end{aligned}$$

$$\begin{aligned}
\mathcal{K}^{SE,(i)}_{j+1} &= \langle \mathcal{K}^{SE,(i)}_{j} \cup & \textit{for } j \geqslant 0, i \in \{1,2\}\\
&\quad \{H(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{E}, \boldsymbol{R}^{(j+1)}) \mid H \in \boldsymbol{H}_i \wedge \\
&\quad\quad \boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|} \wedge |\boldsymbol{E}| = |\boldsymbol{h}| \wedge \boldsymbol{E}_{(u)} \in \mathcal{K}^{SE,(ty(\boldsymbol{h}_{(u)}))}_{j} \}\rangle \\
\mathcal{K}^{SE,(\mathsf{t})}_{j+1} &= \langle \mathcal{K}^{SE,(\mathsf{t})}_{j} \cup (\mathcal{K}^{SE,(1)}_{j+1} * \mathcal{K}^{SE,(2)}_{j+1}) \cup & \textit{for } j \geqslant 0\\
&\quad \{H(\boldsymbol{X}, \boldsymbol{v}, \boldsymbol{E}, \boldsymbol{R}^{(j+1)}) \mid H \in \boldsymbol{H}_{\mathsf{t}} \wedge \\
&\quad\quad \boldsymbol{v} \in \mathbb{F}_p^{|\boldsymbol{a}|} \wedge |\boldsymbol{E}| = |\boldsymbol{h}| \wedge \boldsymbol{E}_{(u)} \in \mathcal{K}^{SE,(ty(\boldsymbol{h}_{(u)}))}_{j} \}\rangle \ .
\end{aligned}$$

The definition is always valid for Type III. *For Types* I *and* II, *it is valid under the previously stated assumptions on* $\boldsymbol{F_i}$ *and* $\boldsymbol{H_i}$. *We say a solution* s *is an* SE-computable solution *if it is constrained by* $(\mathcal{K}^{SE,(i)}_{j})_{j,i}$.

Theorem 2 (Soundness and Completeness of Translation)**.** *Let* $p \approx 2^\lambda$ *and* q_g, q *polynomial in* λ. *Then the winning probability in the generic group game* $\mathsf{G}^{\mathsf{gen}}(SE)$ *with a group of order* p *is negligible in* λ *for all adversaries that perform at most* q_g *(resp.* q*) queries iff there is no* SE*-computable solution for* $\mathit{toConstr}(SE)$.

Sketch. For all concrete values of q_g, q, and SE we can use the master theorem for interactive assumptions from [43] (more precisely, the extended version for handles from [102]) to obtain an algebraic criterion that is equivalent to the security of the construction. By unfolding the definitions of $\mathit{toConstr}$ and eval, we can verify that the criterion is true for all bounds on the number of oracle-queries iff there is no SE-computable solution for $\mathit{toConstr}(SE)$. □

Example 2. *The translation of the security experiment for the example in Figure 3.1 to winning constraints is*

$$\widehat{S} - \widehat{M} * V - W - \widehat{T}_1 * \widehat{T}_2 = 0 \quad \wedge \quad \widehat{T}_1 - \widehat{T}_2 = 0 \quad \wedge \quad \forall k.\ \widehat{M} - M_{[k]} \neq 0$$

where $V, W, R \in \mathsf{UVar}$ *and* $M \in \mathsf{HVar}_2$, $\mu, \mu', \mu'', \rho, \rho', \rho'', \rho''', \tau, \tau', \tau'', \gamma, \gamma', \gamma'' \in \mathsf{PVar}$, *and* $\widehat{M}, \widehat{S}_1, \widehat{S}_2, \widehat{S}_3$ *are defined as*

$$\begin{aligned}
\widehat{M} &= \mu + \sum_k \mu'_{[k]} * R_{[k]} + \sum_k \mu''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}),\\
\widehat{T}_1 &= \rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W,\\
\widehat{T}_2 &= \tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}), \ \textit{and}\\
\widehat{S} &= \gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \ .
\end{aligned}$$

We first outline the sequence of computable monomials for $\mathbb{G}_1$*:*

$$\begin{aligned}
\mathcal{K}_0^{SE,(2)} &= \langle 1, V, W\rangle\\
\mathcal{K}_1^{SE,(2)} &= \langle \mathcal{K}_0^{SE,(2)} \cup \{R_{[1]}\}\rangle\\
\mathcal{K}_2^{SE,(2)} &= \langle \mathcal{K}_1^{SE,(2)} \cup \{R_{[2]}\}\rangle\\
&\dots
\end{aligned}$$

For $\mathbb{G}_2$*, the sequence looks as follows:*

$$\begin{aligned}
\mathcal{K}_0^{SE,(2)} &= \langle 1\rangle\\
\mathcal{K}_1^{SE,(2)} &= \langle \mathcal{K}_0^{SE,(2)} \cup \{1, R_{[1]}, \overbrace{V + W + R_{[1]}^2}^{:=f_1}\}\rangle\\
\mathcal{K}_2^{SE,(2)} &= \langle \mathcal{K}_1^{SE,(2)} \cup \{R_{[2]}, R_{[1]} * V + W + R_{[2]}^2, f_1 * V + W + R_{[2]}^2\}\rangle\\
&\dots
\end{aligned}$$

For $\mathbb{G}_t$*, only the first line of the definition (computable earlier or product of computable in* $\mathbb{G}_1$ *and computable in* $\mathbb{G}_2$*) is non-empty.* ■

3.3 Constraint Solving

In this section, we define an algorithm that takes a winning constraint and tries to derive a contradiction thereby showing that the winning constraint has no solution. Our algorithm uses constraint solving rules to perform a complete search for solutions using simplification rules and case distinctions. We first give the rules and then describe a strategy to apply the rules in Section 3.4. We begin by describing a set of simplification rules for constraints that exploit logical equivalences to bring a constraint into a simplified form. Next, we describe a set of rules for introducing and simplifying Coeff constraints. Then, we describe our rules for performing case distinctions followed by describing a procedure for equational simplification based on Gröbner Basis techniques. We conclude by giving a worked out example.

3.3.1 Constraint solving rules and soundness

We use the notation $\mathcal{C} \rightsquigarrow_{SE} \mathcal{C}_1 \vee \ldots \vee \mathcal{C}_k$ to denote the constraint solving rule that "simplifies" the constraint $\mathcal{C}$ into the disjunction of constraints $\mathcal{C}_1, \ldots, \mathcal{C}_k$. The constraint solving rule might depend on the security experiment SE. Our rules are sound in the following sense: If there exists an SE-solution s for $\mathcal{C}$, then there is an $i \in \{1, \ldots, k\}$ such that there exists an SE-solution s' for $\mathcal{C}_i$. The solution s' is usually very similar to s, but might, for example, perform an

$\mathcal{C}_{simp} ::= \exists k \notin K. \mathcal{C}_{simp} \mid \mathcal{C}^{\wedge}$	existential quantification
$\mathcal{C}^{\wedge} ::= \mathcal{C}^{\forall} \wedge \ldots \wedge \mathcal{C}^{\forall}$	conjunction
$\mathcal{C}^{\forall} ::= \forall k \notin K. \mathcal{C}^{\forall} \mid \mathcal{C}^{eq}$	universal quantification
$\mathcal{C}^{eq} ::= \mathcal{E}^{+} = 0 \mid \mathcal{E}^{+} \neq 0$	(in)equality
$\mathcal{E}^{+} ::= \mathcal{E}^{\Sigma} + \ldots + \mathcal{E}^{\Sigma} \mid 0$	sum
$\mathcal{E}^{\Sigma} ::= \sum_{k \notin K} \mathcal{E}^{\Sigma} \mid -\mathcal{E}^{*} \mid \mathcal{E}^{*} \mid \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}^{*})$	symbolic sum
$\mathcal{M} ::= \mathcal{M} * \mathcal{M} \mid R^{\pm 1} \mid R^{\pm 1}_{[k]} \mid 1$	monomial over uniform variables
$\mathcal{E}^{*} ::= \mathcal{E}^{pv} * \ldots * \mathcal{E}^{pv} \mid 1$	monomials
$\mathcal{E}^{pv} ::= \rho_{[k]} \mid \rho \mid R^{\pm 1} \mid R^{\pm 1}_{[k]} \mid Y_{[k]}$	parameter/variable

Figure 3.4: Grammar for simplified winning constraints where $\rho \in \mathsf{PVar}, R \in \mathsf{UVar}, Y \in \mathsf{HVar}, k \in \mathsf{IVar}$. Conjunctions, sums, and products cannot by empty, but they can have a single argument. All bound variables must occur in the body. A monomial never contains a uniform variable and its inverse and never contains 1 unless it is equal to 1.

additional query with trivial parameters. We use $\mathcal{C} \rightsquigarrow_{SE} \perp$ to denote that $\mathcal{C}$ can be simplified to the empty disjunction, which is equivalent to false.

We say a constraint $\mathcal{C}$ is contradictory if there is either a rule $\mathcal{C} \rightsquigarrow_{SE} \perp$ or there is a rule $\mathcal{C} \rightsquigarrow_{SE} \mathcal{C}_1 \vee \ldots \vee \mathcal{C}_k$ such that for all $i \in \{1, \ldots, k\}$, the constraint $\mathcal{C}_i$ is contradictory. Since all rules are sound, we obtain that if $\mathcal{C}$ is contradictory, then $\mathcal{C}$ has no solution.

3.3.2 Simplification rules

To exploit the equivalence $e = e'$ given in Figure 3.5, we define a corresponding constraint solving rule $C[e] \rightsquigarrow_{SE} C[e']$ for each of them. The rules up to and including the equivalences for Coeff can be used to bring every winning constraint into simplified form (see Figure 3.4). Additionally, we assume given rules for the axioms of commutative rings with respect to $0, 1, *$ and $+$.

The remaining rules are useful to enable the application of other rules. The first remaining set of rules allows to swap binders, which might be required before applying rules that expect a certain binder to be in outermost position. To preserve the well-formedness of constraints, we adapt the index exception sets K as shown below. The second remaining set of rules allows us to add exceptions to binders. This might also benefit the applicability of other rules.

$$
\begin{aligned}
(\forall k \notin K. \mathcal{C}_1 \wedge \mathcal{C}_2) &= (\forall k \notin K. \mathcal{C}_1) \wedge (\forall k \notin K. \mathcal{C}_2) && && \textsf{(equiv-1)}\\
(\forall k \notin K. \mathcal{C}_1) &= \mathcal{C}_1 && \text{if } k \notin \mathit{ivars}(\mathcal{C}_1) && \textsf{(equiv-2)}\\
\mathcal{E}_1 * \mathcal{E}_2 &= \mathcal{E}_2 * \mathcal{E}_1 && && \textsf{(equiv-3)}\\
-(\mathcal{E}_1 + \mathcal{E}_2) &= (-\mathcal{E}_1) + (-\mathcal{E}_2) && && \textsf{(equiv-4)}\\
\sum_{k \notin K} (\mathcal{E}_1 + \mathcal{E}_2) &= (\sum_{k \notin K} \mathcal{E}_1) + (\sum_{k \notin K} \mathcal{E}_2) && && \textsf{(equiv-5)}\\
(\sum_{k \notin K} \mathcal{E}_1) * \mathcal{E}_2 &= (\sum_{k \notin K} \mathcal{E}_1 * \mathcal{E}_2) && && \textsf{(equiv-6)}\\
-(\sum_{k \notin K} \mathcal{E}) &= (\sum_{k \notin K} -\mathcal{E}) && && \textsf{(equiv-7)}\\
((-\mathcal{E}_1) * \mathcal{E}_2) &= -(\mathcal{E}_1 * \mathcal{E}_2) && && \textsf{(equiv-8)}\\
-(-\mathcal{E}) &= \mathcal{E} && && \textsf{(equiv-9)}\\
\mathcal{R} * \mathcal{R}^{-1} &= 1 && && \textsf{(equiv-10)}\\
\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_1 + \mathcal{E}_2) &= \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_1) + \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}_2) && && \textsf{(equiv-11)}\\
\mathsf{Coeff}_{\mathcal{M}}(\sum_{k \notin K} \mathcal{E}) &= \sum_{k \notin K} \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) && \text{if } \mathit{ivars}(\mathcal{M}) \subseteq K && \textsf{(equiv-12)}\\
\mathsf{Coeff}_{\mathcal{M}}(-\mathcal{E}) &= -\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) && && \textsf{(equiv-13)}
\end{aligned}
$$

$$
\begin{aligned}
\exists k_1 \notin K_1. \exists k_2 \notin K_2. \mathcal{C} &= \exists k_2 \notin K_2'. \exists k_1 \notin K_1'. \mathcal{C} && \textsf{(swap-1)}\\
\forall k_1 \notin K_1. \forall k_2 \notin K_2. \mathcal{C} &= \forall k_2 \notin K_2'. \forall k_1 \notin K_1'. \mathcal{C} && \textsf{(swap-2)}\\
\sum_{k_1 \notin K_1} \sum_{k_2 \notin K_2} \mathcal{E} &= \sum_{k_2 \notin K_2'.} \sum_{k_1 \notin K_1'} \mathcal{E} && \textsf{(swap-3)}
\end{aligned}
$$

$$
\begin{aligned}
\forall k \notin K. \mathcal{C} &= (\forall k \notin K \cup \{k^*\}. \mathcal{C}) \wedge \mathcal{C}[k \mapsto k^*] && \text{if } k^* \notin K && \textsf{(split-1)}\\
C[\sum_{k \notin K} \mathcal{E}] &= C[(\sum_{k \notin K \cup \{k^*\}} \mathcal{E}) + \mathcal{E}[k \mapsto k^*]] && \text{where C} && \textsf{(split-2)}\\
& && \text{defines } k^* \neq k' && \\
& && \text{forall } k' \in K &&
\end{aligned}
$$

Figure 3.5: Equivalences for simplifying constraints where K_2' is defined as $K_2 \backslash \{k_1\}$ and K_1' is defined as $K_1 \cup \{k_2\}$ if $k_1 \in K_2$ and K_1 otherwise.

$$C[\mathcal{E} = 0] \rightsquigarrow_{SE} C[\mathcal{E} = 0 \wedge (\forall i_1 \notin K_1, \dots, i_l \notin K_l.\, \mathsf{Coeff}_{\mathcal{M}}(\mathcal{E}) = 0)]$$
$$\text{if } \{i_1, \dots, i_l\} \cap \mathit{ivars}(\mathcal{E}) = \varnothing \wedge \mathcal{E} \text{ does not contain } \mathsf{Coeff} \quad \text{(coeff-1)}$$

$$C[\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})] \rightsquigarrow_{SE} C[\mathit{pmon}(\mathcal{E})] \quad \text{if } \mathit{hmon}(\mathcal{E}) = 1 \text{ and } \mathcal{M} = \mathit{umon}(\mathcal{E}) \quad \text{(coeff-2)}$$

$$C[\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})] \rightsquigarrow_{SE} C[0] \quad \text{if } \mathsf{contMon}_{\mathcal{M}/\mathit{umon}(\mathcal{E})}(\mathit{hmon}(\mathcal{E})) \rightsquigarrow_{SE} \bot \text{ and } C \text{ assures } \mathit{ivars}(\mathcal{M}) \cap \mathit{ivars}(\mathcal{E}) = \varnothing \quad \text{(coeff-3)}$$

Figure 3.6: Rules for introducing and simplifying Coeff expressions.

3.3.3 Introducing and simplifying Coeff constraints

In this section, we describe how to introduce and simplify constraints that involve Coeff expressions. To define our constraint solving rules, we define three functions that filter variables in monomials.

The functions

- $\mathit{umon} : \mathit{Mon}[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to \mathit{Mon}[\mathsf{UVar}^{\pm 1}]$,
- $\mathit{hmon} : \mathit{Mon}[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to \mathit{Mon}[\mathsf{HVar}]$, and
- $\mathit{pmon} : \mathit{Mon}[\mathsf{UVar}^{\pm 1}, \mathsf{HVar}, \mathsf{PVar}] \to \mathit{Mon}[\mathsf{PVar}]$.

keep the exponents for the desired type of variables and set the exponents of all other variables to zero.

The constraint solving rules are given in Figure 3.6. The first rule exploits that if a polynomial is equal to zero, then when interpreting the polynomial as a polynomial over uniform variables, the coefficients for all monomials must be zero. The remaining two rules allow to simplify Coeff expressions. The first rule deals with the case where $\mathcal{E}$ does not contain any handle variables and $\mathcal{M}$ is equal to the monomial over uniform variables contained in $\mathcal{E}$. The second rule deals with the case where it is possible to prove that there is no (SE-computable) instantiation of the handle variables in $\mathcal{E}$ such that the resulting Laurent polynomial contains the monomial $\mathcal{M}$. The rule makes uses the contMon constraint. We will present the rules for showing that such a constraint is contradictory in the next section.

Example 3. *Consider the constraint* Γ *such that*

$$\Gamma = (\sum_j \rho_{[j]} R_{[j]} = 0) \wedge \Gamma' \ .$$

We can simplify the constraint as follows:

$$\begin{aligned}
\Gamma &\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_{j} \rho_{[j]} R_{[j]}) = 0 && [\textit{coeff-1}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}((\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \rho_{[i]} R_{[i]}) = 0 && [\textit{split-2}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \mathsf{Coeff}_{R_{[i]}}(\rho_{[i]} R_{[i]}) = 0 && [\textit{equiv-11}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \mathsf{Coeff}_{R_{[i]}}(\sum_{j \notin \{i\}} \rho_{[j]} R_{[j]}) + \rho_{[i]} = 0 && [\textit{coeff-2}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, (\sum_{j \notin \{i\}} \mathsf{Coeff}_{R_{[i]}}(\rho_{[j]} R_{[j]})) + \rho_{[i]} = 0 && [\textit{equiv-12}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, (\sum_{j \notin \{i\}} 0) + \rho_{[i]} = 0 && [\textit{coeff-3}]\\
&\rightsquigarrow_{SE} \Gamma \wedge \forall i.\, \rho_{[i]} = 0 && [\textit{equiv-ring}]
\end{aligned}$$

For the step using [coeff-3], we exploit that $\mathsf{contMon}_{R_{[i]}/R_{[j]}}(1) \rightsquigarrow_{SE} \perp$ *and that* $j \notin \{i\}$ *ensures that these index variables will never be instantiated with the same value in the given context. We will give the required rules in the next section. Then, our Gröbner-Basis based simplification algorithm will replace* $\rho_{[j]}$ *by* 0 *in* Γ *for arbitrary index variables* j. ■

3.3.3.1 Proving Coeff to be zero for all *SE* solutions.

In this section, we describe a method to check if $\mathsf{Coeff}_{\mathcal{M}}(\mathcal{E})$ can be simplified to 0, i.e., for all *SE*-computable solutions $s = (p, q, \sigma, \delta, \chi, \xi)$, it holds that $\mathsf{coeff}_{\sigma(\mathcal{M})}(eval_s(\mathcal{E})) = 0$. As in previous sections, we describe our approach for Type III, but stress that it can be adapted to Type I and Type II, e.g., by transforming the security experiment to make the isomorphisms redundant. We assume that the oracle definitions are efficiently computable and only returnc handles to elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. Furthermore, we assume that the winning condition only uses handles to elements of $\mathbb{G}_1$ and $\mathbb{G}_2$. This covers most cryptographic constructions of interest (including all SPS schemes). In this case, we never have to deal with handle variables from $\mathsf{HVar}_{\mathsf{t}}$ and for $i \in \{1, 2\}$, the polynomials $\boldsymbol{H}_i$ defining the oracle return values contain only handle variables from HVar_i. We distinguish three cases for $\mathsf{contMon}_{\mathcal{M}}(\mathcal{E})$: (i) $\mathsf{deg}(\mathcal{E}) = 0$, (ii) $\mathsf{deg}(\mathcal{E}) = 1$, and (iii) $\mathsf{deg}(\mathcal{E}) > 1$.

Case (i): We use the rule

$$\mathsf{contMon}_{\mathcal{M}}(1) \rightsquigarrow_{SE} \perp \quad \text{if } \mathcal{M} \neq 1 \ .$$

Here, we require that distinct index variables must be instantiated with distinct values, which is ensured by the side condition of the Coeff-(3) rule.

Case (ii): We have $\mathcal{E} = Y_{[j]}$ for $Y \in \mathsf{HVar}_i$, $j \in \mathsf{IVar}$, and $i \in \{1, 2\}$. We must prove that the monomial $\mathcal{M}$ is not computable in i before query j, i.e., it is impossible (in the symbolic group model) to obtain a handle h for $\mathbb{G}_i$ that points to a polynomial F with $m \in \mathit{mons}(F)$ before the j-th oracle query. We perform a proof by contradiction that covers all cases on how a given monomial $\mathcal{M}$ can be computed. We write $\mathsf{canMult}_{i,\{j_1,\ldots,j_n\}}(m)$ if it is possible to perform the multiplication of a given monomial with m using oracle queries with query-indices distinct from $\{j_1, \ldots, j_n\}$. For example, if we have an oracle that returns a handle to $Y * R_{[j]} + W$ in $\mathbb{G}_1$ (where $Y \in \mathsf{HVar}_1, R, W \in \mathsf{UVar}$), then $\mathsf{canMult}_{1,\{j_1\}}(R_{[j_2]} * R_{[j_3]})$ is true since we can call the oracle for indices j_2 and j_3 to perform a multiplication with $R_{[j_2]}$ and $R_{[j_3]}$. In contrast, $\mathsf{canMult}_{1,\{j_1\}}(R_{[j_1]} * R_{[j_2]} * R_{[j_3]})$ is false because we cannot multiply with $R_{[j_1]}$ if using the oracle for query index j_1 is forbidden. To formalize this reasoning, we define a set of rules to reduce a constraint $\mathsf{contMon}_m(Y_{[j]})$ to a disjunction of constraints $\mathsf{canMult}_{i,J}(m)$ such that $\mathit{ivars}(m) = \emptyset$.

We define the set $\mathcal{SM}_i^{SE}$ of *start monomials for a security experiment SE and group index i* as $\mathit{mons}(\boldsymbol{F}_i) \cup (\mathit{mons}(\boldsymbol{H}_i) \cap \mathit{Mon}[\mathsf{UVar}^{\pm 1}])$ where the $\boldsymbol{H}_i$ are considered as polynomials over handle and uniform variables. We define the set $\mathcal{TM}_i^{SE}$ of *transformation monomials for a security experiment SE and a group index i* as $\{m \mid Y * m \in \mathit{mons}(\boldsymbol{H}_i) \wedge Y \in \mathsf{HVar}_i\} \subseteq \mathit{Mon}[\mathsf{UVar}^{\pm 1}]$. For both sets, we partition the previously defined sets into $\mathcal{SM}_i^{SE} = \mathcal{SM}_{i,glob}^{SE} \uplus \mathcal{SM}_{i,orcl}^{SE}$ and $\mathcal{TM}_i^{SE} = \mathcal{TM}_{i,glob}^{SE} \uplus \mathcal{TM}_{i,orcl}^{SE}$ where the *glob*-sets contain all monomials that contain only global uniform variables and the *orcl*-sets contain all monomials that contain at least one oracle uniform variable. For monomials m, we write $m[j]$ to denote the monomial where all oracle uniform variables Y are replaced with their indexed versions $Y_{[j]}$. We also use the same notation for sets of monomials.

We can now define the rules given in Figure 3.7. The first rule captures that to compute the monomial $\widetilde{m}$ in i before query j, the adversary must start with a monomial m' (in $m_1, \ldots, m_l, \widehat{m_1}[j_1], \ldots$) and then use oracle queries to achieve an indirect multiplication of m' by $\widetilde{m}/m'$. Here, the monomials m_i are either monomials included in the adversary input or monomials included in the oracle return values that do not depend on handles and do not contain oracle uniform variables. The monomials $\widehat{m_i}[j_u]$ are monomials included in the oracle return values that do not depend on handles and that contain oracle uniform variables. The set of forbidden query indices for the indirect multiplication takes into account that j can never be used and that j_u cannot be used if a monomial with index j_u is used as the start monomial.

$$\mathsf{contMon}_{\widetilde{m}}(Y_{[j]}) \rightsquigarrow_{SE} \qquad \text{[contMon-1]}$$
$$\mathsf{canMult}_{i,\{j\}}(\widetilde{m}/m_1) \vee \ldots \vee \mathsf{canMult}_{i,\{j\}}(\widetilde{m}/m_l) \vee$$
$$\mathsf{canMult}_{i,\{j,j_1\}}(\widetilde{m}/\widehat{m}_1[j_1]) \vee \ldots \vee \mathsf{canMult}_{i,\{j,j_1\}}(\widetilde{m}/\widehat{m}_{\hat{l}}[j_1]) \vee$$
$$\ldots \vee$$
$$\mathsf{canMult}_{i,\{j,j_n\}}(\widetilde{m}/\widehat{m}_1[j_n]) \vee \ldots \vee \mathsf{canMult}_{i,\{j,j_n\}}(\widetilde{m}/\widehat{m}_{\hat{l}}[j_n])$$

if $Y \in \mathsf{HVar}_i$, $\{m_1, \ldots, m_l\} = \mathcal{SM}^{SE}_{i,glob}$,
$\{\widehat{m}_1, \ldots, \widehat{m}_{\hat{l}}\} = \mathcal{SM}^{SE}_{i,orcl}$, and $\{j_1, \ldots, j_n\} = ivars(\widetilde{m}) \backslash \{j\}$.

$$\mathsf{canMult}_{i,J}(\widetilde{m}) \rightsquigarrow_{SE} \qquad \text{[contMon-2]}$$
$$\mathsf{canMult}_{i,J\cup\{j\}}(\widetilde{m}/m_1[j]) \vee \ldots \vee \mathsf{canMult}_{i,J\cup\{j\}}(\widetilde{m}/m_l[j])$$

if $\{m_1, \ldots, m_l\} = \mathcal{TM}^{SE}_{i,orcl}$ and $j \in ivars(\widetilde{m}) \backslash J$.

$$\mathsf{canMult}_{i,J}(\widetilde{m}) \rightsquigarrow_{SE} \bot \qquad \text{[contMon-3]}$$

if $J \cap ivars(\widetilde{m}) \neq \varnothing$

Figure 3.7: Rules for dealing with contMon. We use m/m' to denote the corresponding reduced Laurent monomial

The second rule is applicable whenever $\widetilde{m}$ contains an indexed uniform variable $R_{[j]}$ such that $j \notin J$. In this case, the j-th query must be used to perform an indirect multiplication that cancels out $R_{[j]}$ and we perform a case distinction on all monomial multiplications containing oracle uniform variables that can be performed by the oracle. For all cases where this step does not cancel out *all* variables indexed with j, we can use the third rule that formalizes the following fact: If the j-th query is forbidden, there is no way to cancel out a uniform variable with index j.

It is not hard to see that we can reduce all constraints to $\mathsf{canMult}_{i,J}(\widetilde{m})$ such that $ivars(\widetilde{m}) = \varnothing$: If $ivars(\widetilde{m})$ non-empty, then either there is a $j \in ivars(\widetilde{m}) \cap J$ and we can conclude with the last rule or we can apply the second rule and add an index $j \in ivars(\widetilde{m})$ to J. To check if a constraint $\mathsf{canMult}_{i,J}(\widetilde{m})$ with $ivars(\widetilde{m}) = \varnothing$ is unsatisfiable, we translate the constraint into a system of linear equations that formalizes the following idea. Let $\{m_1, \ldots, m_l\} = \mathcal{TM}^{SE}_{i,glob}$, then all indirect multiplications that do not introduce indexed uniform variables are of the form

$$m_1^{\delta_1} * \ldots * m_l^{\delta_l}$$

for $\delta_i \in \mathbb{N}$. This corresponds to using the i-th transformation δ_i times to achieve a multiplication with $m_i^{\delta_i}$. To check if there exist $\delta_1, \ldots, \delta_l \in \mathbb{N}$ such that

$$\widetilde{m} = m_1^{\delta_1} * \ldots * m_l^{\delta_l}$$

we check if the linear system of equations

$$\begin{aligned}
&\mathsf{deg}_{V_1}(\widetilde{m}) = \mathsf{deg}_{V_1}(m_1) * \delta_1 + \ldots + \mathsf{deg}_{V_1}(m_l) * \delta_l \\
&\ldots \\
&\mathsf{deg}_{V_n}(\widetilde{m}) = \mathsf{deg}_{V_n}(m_1) * \delta_1 + \ldots + \mathsf{deg}_{V_n}(m_l) * \delta_l
\end{aligned}$$

has a solution over $\mathbb{N}$ where $\{V_1, \ldots, V_n\}$ is the set of uniform variables that occur in $\widetilde{m}, m_1, \ldots, m_l$.

Case (iii): The last case can be handled by generalizing the previous case. We sketch how to achieve this, we have, $\mathcal{E} = (Y_1)_{[j_1]} * \ldots * (Y_n)_{[j_n]}$ for $Y_u \in \mathsf{HVar}_{i_u}$, $j_u \in \mathsf{IVar}$, and $i_u \in \{1, 2\}$. To extend the method from Case (ii), we use adapted set of start monomials and transformation monomials that take cancellations between these values for the different handles into account. For example, the set of transformation monomials is the product of transformation monomial sets for $j_1, \ldots, j_n$ also allowing any set to be replaced by $\{1\}$.

Example 4. *We will show that* $\mathsf{contMon}_{R_{[i]}/V}(M_{[k]})$ *is contradictory for the security experiment SE defined in Example 1. Note that* $M_{[k]} \in \mathsf{HVar}_2$ *and the monomial sets for this group are:*

$$\begin{aligned}
\mathcal{SM}_{2,glob}^{SE} &= \{1, W\} & \mathcal{SM}_{2,orcl}^{SE} &= \{R, R^2\} \\
\mathcal{TM}_{2,glob}^{SE} &= \{V\} & \mathcal{TM}_{2,orcl}^{SE} &= \emptyset
\end{aligned}$$

By applying the first rule in Figure 3.7 we have:

$$\begin{aligned}
&\mathsf{contMon}_{R_{[i]}/V}(M_{[k]}) \rightsquigarrow_{SE} \\
&\quad \mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}) \lor \mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}W^{-1}) \lor \quad \textit{(div. by } 1 \textit{ and } W\textit{)} \\
&\quad \mathsf{canMult}_{2,\{k,i\}}(V^{-1}) \lor \mathsf{canMult}_{2,\{k,i\}}(V^{-1}R_{[i]}^{-1}) \quad \textit{(div. by } R_{[i]} \textit{ and } R_{[i]}^2\textit{)}
\end{aligned}$$

Now, since $\mathcal{TM}_{2,orcl}^{SE} = \emptyset$*, the second rule in Figure 3.7 gives us:*

$$\begin{aligned}
&\mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}) \rightsquigarrow_{SE} \bot \\
&\mathsf{canMult}_{2,\{k\}}(R_{[i]}V^{-1}W^{-1}) \rightsquigarrow_{SE} \bot
\end{aligned}$$

$$C[\mathcal{E}_1 * \mathcal{E}_2 = 0] \leadsto_{SE} C[\mathcal{E}_1 = 0] \vee C[\mathcal{E}_2 = 0] \qquad \textsf{[dist-1]}$$

$$C[\exists i \notin K.\,\mathcal{C}'] \leadsto_{SE} \begin{array}{ll} & C[\exists i \notin K \cup \{j\}.\,\mathcal{C}'] \\ \vee & C[\mathcal{C}'[i \mapsto j]] \end{array} \qquad \text{if } j \notin K \qquad \textsf{[dist-2]}$$

$$C[\mathcal{C}'] \leadsto_{SE} \begin{array}{ll} & C[\mathcal{C}' \wedge \mathcal{E} = 0] \\ \vee & C[\mathcal{C}' \wedge \mathcal{E} \neq 0] \end{array} \qquad \text{with } \mathcal{E} \text{ arbitrary} \qquad \textsf{[dist-3]}$$

$$\exists \Delta.\,\mathcal{C}' \leadsto_{SE} \begin{array}{ll} & \exists \Delta.\,(\forall i \notin K.\,\rho_{[i]} = 0) \wedge \mathcal{C}' \\ \vee & \exists \Delta, j \notin K.\,\rho_{[j]} \neq 0 \wedge \mathcal{C}' \end{array} \qquad \begin{array}{l} \text{with } K \text{ arbitrary} \\ \text{and } j \notin \mathit{ivars}(\Delta) \\ \cup\, \mathit{ivars}(\mathcal{C}') \end{array} \qquad \textsf{[dist-4]}$$

$$C[c = 0] \leadsto_{SE} \bot \qquad \text{if } c \in \mathbb{Z}\backslash\{0\} \qquad \textsf{[false-1]}$$

$$C[0 \neq 0] \leadsto_{SE} \bot \qquad \textsf{[false-2]}$$

Figure 3.8: Rules for performing case distinctions and contradictions.

Additionally,

$$\textsf{canMult}_{2,\{k,i\}}(V^{-1}R_{[i]}^{-1}) \leadsto_{SE} \bot$$

because $\{k,i\} \cap \mathit{ivars}(V^{-1}R_{[i]}^{-1}) \neq \varnothing$*. Our problem has been reduced to compute*

$$\textsf{canMult}_{2,\{k,i\}}(V^{-1})$$

so we define the system of equations:

$$\textsf{deg}_V(V^{-1}) = \textsf{deg}_V(V) * \delta_1$$

where $\delta_1 \in \mathbb{N}$*. The equation is* $-1 = 1 * \delta_1$ *and it reduces to* $\bot$*. This analysis proves that* $\textsf{contMon}_{R_{[i]}/V}(M_{[k]}) \leadsto_{SE} \bot$*, i.e., the handle variable* $M_{[k]}$ *cannot contain the monomial* $R_{[i]}/V$*.*

3.3.4 Case distinctions and contradictions

The rules for case distinctions and contradictions are given in Figure 3.8. The first rule is applicable whenever we can express the left-hand-side of an equality with 0 as a product of the two factors $\mathcal{E}_1$ and $\mathcal{E}_2$. Since we reason about elements of an integral domain, we can conclude that at least one of the factors must be equal to 0. The second rule formalizes that if $\mathcal{C}'$ is true for some i, then it it is either true for some $i \neq j$ or it is true for $i = j$. The third rule formalizes that for all expressions $\mathcal{E}$, the expression is either equal to 0 or not. We only apply this rule with an $\mathcal{E}$ that already occurs as a subterm of C. In most cases $\mathcal{E} = \rho$ for $\rho \in \mathsf{PVar}$. The final case distinction rule deals with indexed parameter variables $\rho_{[i]}$. Either $\rho_{[i]}$ is equal to zero for all indices not in K or there is an index j not in K such that $\rho_{[j]}$ is not zero. The rule uses Δ to denote all existential bindings in the constraint.

The two contradiction rules are straightforward. The first rule states that a non-zero constant c is not equal to zero. We keep track of applications of this rule to obtain a lower bound on the the prime p for which our proof is valid. The second rule just formalizes that zero is always equal to itself.

3.3.5 Gröbner Basis simplification

Before applying the Gröbner Basis simplification, we ensure that all $\forall$-quantifiers use the same binders Δ and that all index exception sets are maximal for Δ. This might require renaming of variables, extending the index exception sets, and introducing unused variables. For the $\sum$-binders $\widehat{\Delta}_u$, we assume for all u, v that (i) $\widehat{\Delta}_u = \widehat{\Delta}_v$, (ii) $\widehat{\Delta}_u$ is a prefix of $\widehat{\Delta}_v$, or (iii) vice versa.

The resulting constraint system can be rearranged to have the following form

$$\begin{aligned}\exists\nabla.\, &(\forall\Delta.\, \mathcal{E}_1 = 0) \wedge \ldots \wedge (\forall\Delta.\, \mathcal{E}_l = 0) \wedge \\ &(\forall\Delta.\, \widehat{\mathcal{E}}_1 \bowtie_1 0) \wedge \ldots \wedge (\forall\Delta.\, \widehat{\mathcal{E}}_{\hat{l}} \bowtie_{\hat{l}} 0)\end{aligned}$$

where the $\mathcal{E}_u$ are expressions that do not contain handle variables, uniform variables, or Coeff expressions, which we call parameter equality polynomials. The $\widehat{\mathcal{E}}_u$ denote the remaining expressions. We want to move all the $\mathcal{E}_u$ under a single quantifier for simplification. To take renamings of the bound variables into account, we ensure beforehand that for all $\mathcal{E}_u$ and all permutations of the $\forall$-bound variables, the resulting expression is already included. For example, given

$$\forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_1]} * \rho'_{[j_2]} = 0 \wedge \forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_2]} * \rho'_{[j_1]} - \alpha = 0$$

it is usually useful to add at least the permutation

$$\forall j_1, j_2 \notin \{j_1\}.\, \rho_{[j_1]} * \rho'_{[j_2]} - \alpha = 0$$

before moving moving everything under a common quantifier since this yields the shared monomial $\rho_{[j_1]} * \rho'_{[j_2]}$. After moving the parameter equality polynomials under the same quantifier, we get:

$$\begin{aligned}\exists\nabla.\, &(\forall\Delta.\, \mathcal{E}_1 = 0 \wedge \ldots \wedge \mathcal{E}_l = 0) \wedge \\ &(\forall\Delta.\, \widehat{\mathcal{E}}_1 \bowtie_1 0) \wedge \ldots \wedge (\forall\Delta.\, \widehat{\mathcal{E}}_{\hat{l}} \bowtie_{\hat{l}} 0)\ .\end{aligned}$$

Now, we move non-indexed parameters in monomials out of the $\sum$-binder and consistently replace non-bound parameters and $\sum$-expressions with variables X_v. We call the corresponding mapping σ and use g_u to denote polynomial resulting from $\mathcal{E}_u$. We can revert this abstraction process by applying σ, i.e., $\sigma(g_u) = \mathcal{E}_u$. Next, we compute the Gröbner Basis (over $\mathbb{Z}$) of the ideal $\langle g_1, \ldots, g_l \rangle$ which we denote with $I = \langle g'_1, \ldots, g'_{l'} \rangle$. By the properties of the Gröbner Basis, we know that

$$(g_1 = 0 \wedge \ldots \wedge g_l = 0) \Leftrightarrow (g'_1 = 0 \wedge \ldots \wedge g'_{l'} = 0)$$

and hence

$$(\mathcal{E}_1 = 0 \wedge \ldots \wedge \mathcal{E}_l = 0) \Leftrightarrow (\mathcal{E}'_1 = 0 \wedge \ldots \wedge \mathcal{E}'_{l'} = 0)$$

for $\mathcal{E}'_u = \sigma(g_u)$ which we exploit to simplify the parameter equality polynomials. For computing the Gröbner Basis, we use a monomial order that prefers to eliminate abstracted $\sum$ expressions. Next, we use the Gröbner Basis to simplify the expressions $\forall\Delta.\, \widehat{\mathcal{E}}_u \bowtie_u 0$. If $\widehat{\mathcal{E}}_u$ uses all variables in Δ, we use an extension σ' of σ to abstract $\widehat{\mathcal{E}}_u$ to the polynomial f and define f' as the result of reducing f modulo the Gröbner Basis I. As before, we define the simplified $\widehat{\mathcal{E}}'_u$ as $\sigma'(f')$. Often, it is very useful to also simplify below $\sum$-binders. We use an example to illustrate how this works.

Example 5. *Assume $\nabla = j_1$, $\Delta = j_2 \notin \{j_1\}$, $I = \langle X_1 * X_2 \rangle$, $\sigma = \{X_1 \mapsto \rho_{[j_1]}, X_1 \mapsto \rho'_{[j_2]}\}$, and*

$$\mathcal{E}_1 = (\sum_{j_3 \notin \{j_1\}} \rho_{[j_1]} * \rho'_{[j_3]} = 0) \ .$$

Then we use $\forall j_2 \notin \{j_1\}.\, \rho_{[j_1]} * \rho'_{[j_2]} = 0$ to rewrite $\rho_{[j_1]} * \rho'_{[j_3]}$ to 0 below $\sum_{j_3 \notin \{j_1\}}$ by instantiating j_2 with j_3 (both have the same exception j_1). ■

3.3.6 Example: Proof of EUF-CMA for SPS

In this section show how our constraint solving rules can be used to prove (unbounded) EUF-CMA security of the signature scheme in Figure 3.1. The winning constraints for the associated security experiment *SE* are already given in Example 2. To prove EUF-CMA security in the Generic Group Model, we must show that the following constraint has no *SE*-computable solution

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \\
&- ((\tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R^2_{[k]})) \\
&\quad * (\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W) + \widehat{M} * V + W) = 0 \qquad (3.1)
\end{aligned}$$

$$\begin{aligned}
\wedge\quad &\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W \\
&- (\tau + \sum_k \tau'_{[k]} * R_{[k]} + \sum_k \tau''_{[k]} * (M_{[k]} * V + W + R^2_{[k]})) = 0 \qquad (3.2)
\end{aligned}$$

$$\wedge\quad \forall k.\, \widehat{M} - M_{[k]} \neq 0 \qquad (3.3)$$

where $\widehat{M}$ is defined as

$$\widehat{M} = \mu + \sum_k \mu'_{[k]} * R_{[k]} + \sum_k \mu''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \ .$$

Instead of immediately simplifying everything using the equivalences in Figure 3.5, we first apply the rule [coeff-1] where $\mathcal{M} = R^2_{[i]}$ and $\mathcal{E}$ is the equation (3.2). After simplifying the resulting Coeff expressions (see Example 4), we get the new equation $\forall i. -\tau''_{[i]} = 0$. Our Gröbner Basis simplification replaces every occurrence of τ''_i by 0. This results in the following new constraint:

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + W + R^2_{[k]}) \\
&- ((\tau + \sum_k \tau'_{[k]} * R_{[k]}) * (\rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W) \\
&\quad + \widehat{M} * V + W) = 0 \qquad (3.1)
\end{aligned}$$

$$\wedge\quad \rho + \sum_k \rho'_{[k]} * R_{[k]} + \rho'' * V + \rho''' * W - (\tau + \sum_k \tau'_{[k]} * R_{[k]}) = 0 \qquad (3.2)$$

$$\wedge\quad \forall k.\, \widehat{M} - M_{[k]} \neq 0 \qquad (3.3)$$

Now, we can apply the rule [coeff-1] where $\mathcal{E}$ is the left hand side of equation (3.2) and for different monomials $\mathcal{M}$, we obtain the following new equations:

$$\begin{aligned}
&\rho - \tau = 0 && \text{for } \mathcal{M} = 1\\
&\forall k.\, \rho'_{[k]} - \tau'_{[k]} = 0 && \text{for } \mathcal{M} = R_{[k]}\\
&\rho'' = 0 && \text{for } \mathcal{M} = V\\
&\rho''' = 0 && \text{for } \mathcal{M} = W
\end{aligned}$$

After this, we basically got rid of equation (3.2) and our Gröbner Basis simplification yields:

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} + \sum_k \gamma''_{[k]} * (M_{[k]} * V + R^2_{[k]} + W)\\
&- (\tau^2 + (2\sum_k \tau * \tau'_{[k]} * R_{[k]}) + \sum_{k,k' \notin \{k\}} \tau'_{[k]} * \tau'_{[k']} * R_{[k]} * R_{[k']}\\
&\quad + \sum_k \tau'^2_{[k]} * R^2_{[k]} + \widehat{M} * V + W) = 0 \qquad (3.1)
\end{aligned}$$

$$\wedge \quad \forall k.\, \widehat{M} - M_{[k]} \neq 0 \qquad (3.3)$$

We now apply the rule [coeff-1] where $\mathcal{E}$ is expression in equation (3.1) obtaining the following new equations:

$$\begin{aligned}
&\wedge \quad \sum_k \gamma''_{[k]} - 1 = 0 && \text{for } \mathcal{M} = W && (3.4)\\
&\wedge \quad \forall k.\, \gamma''_{[k]} - \tau'^2_{[k]} = 0 && \text{for } \mathcal{M} = R^2_{[k]} && (3.5)\\
&\wedge \quad \forall k.\, \forall k' \notin \{k\}.\, 2 * \tau'_{[k]} * \tau'_{[k']} = 0 && \text{for } \mathcal{M} = R_{[k]}R_{[k']} && (3.6)
\end{aligned}$$

Then, we apply the rule [dist-4] with $K = \varnothing$ to perform a case distinction on the parameter τ':

$$\begin{aligned}
&\forall k.\, \tau'_{[k]} = 0 \wedge \Gamma && \text{(case 1)}\\
\vee \quad &\exists k^*.\, \tau'_{[k^*]} \neq 0 \wedge \Gamma && \text{(case 2)}
\end{aligned}$$

Here, Γ represents the conjunction of our previous five equations. In case 1, the Gröbner Basis simplification results in the system

$$\begin{aligned}
&\gamma + \sum_k \gamma'_{[k]} * R_{[k]} - \tau^2 - \widehat{M} * V - W = 0 && (3.1)\\
\wedge \quad &\forall k.\, \widehat{M} - M_{[k]} \neq 0 && (3.3)\\
\wedge \quad &-1 = 0 && (3.4)
\end{aligned}$$

which simplifies to $\perp$ after applying rule [false-1] to equation (3.4).
In case 2, Gröbner Basis simplification yields:

$$\begin{aligned}
&\exists k^*. \\
&\qquad \gamma + \sum_k \gamma'_{[k]} * R_{[k]} + M_{[k*]} * V - \tau^2 - 2\tau R_{[k*]} - \widehat{M} * V && (3.1)\\
&\wedge \quad \forall k.\, \widehat{M} - M_{[k]} \neq 0 && (3.3)
\end{aligned}$$

We apply the rule [coeff-1] where $\mathcal{E}$ is the left hand side of equation (3.1) for different monomials as $\mathcal{M}$, obtaining:

$$\begin{aligned}
&\gamma - \tau^2 = 0 && \text{for } \mathcal{M} = 1\\
&\forall k \notin \{k^*\}.\, \gamma'_{[k]} = 0 && \text{for } \mathcal{M} = R_{[k]}\\
&\gamma'_{[k*]} - 2\tau = 0 && \text{for } \mathcal{M} = R_{[k*]}
\end{aligned}$$

After simplifying the system, we obtain:

$$\begin{aligned}
&M_{[k*]} * V - \widehat{M} * V = 0 && (3.1)\\
\wedge\quad &\forall k.\, \widehat{M} - M_{[k]} \neq 0 && (3.3)
\end{aligned}$$

Applying the rule [dist-1] to equation (3.1) we obtain two cases:

$$\begin{array}{ccc}
\begin{aligned}
&\exists k^*. \\
&\qquad V = 0 \\
&\wedge \qquad \forall k.\, \widehat{M} - M_{[k]} \neq 0
\end{aligned}
& \bigvee &
\begin{aligned}
&\exists k^*. \\
&\qquad M_{[k*]} - \widehat{M} = 0 \\
&\wedge \qquad \forall k.\, \widehat{M} - M_{[k]} \neq 0
\end{aligned} \\
\text{(case 2.1)} & & \text{(case 2.2)}
\end{array}$$

In case 2.1, after applying [coeff-1] for $\mathcal{M} = V$ to the first equation and simplifying, we obtain the equation $1 = 0$ that reduces to $\perp$ according to rule [false-1].
Finally, in case 2.2 we apply the rule [split-2] and we get the system:

$$\begin{aligned}
&M_{[k*]} - \widehat{M} = 0\\
\wedge\quad &\forall k \notin \{k^*\}.\, \widehat{M} - M_{[k]} \neq 0\\
\wedge\quad &\widehat{M} - M_{[k*]} \neq 0
\end{aligned}$$

Our Gröbner Basis simplification will reduce it to,

$$0 \neq 0 \wedge (\forall k \notin \{k^*\}.\, \widehat{M} - M_{[k]} \neq 0)$$

which reduces to $\perp$ according to rule [false-2].

```
group_setting 3.

sample V,W.
input [V,W] in G1.

oracle o1(M:G2) =
  sample R;
  return [ R ] in G1,
         [ R, M*V + R^2 + W] in G2.

win (wM:G2, wT1:G1, wT2:G2, wS:G2) =
 ((forall i: wM <> M_i) /\ wT1 = wT2 /\ wS = V*wM + wT1*wT2 + W).
```

Figure 3.9: Input file for the Type III re-randomizable SPS scheme from Figure 3.1.

3.4 Implementation and Case Studies

We have implemented the described algorithm in the gga$^{\infty}$ tool[3] and have evaluated its effectiveness and performance on cryptographic constructions from the literature (presented in Table 3.1) and automatically synthesized schemes (presented in Table 3.2). The source code is written in OCaml and uses the computer algebra system SAGE [186] for Gröbner Basis computations and the SMT solver Z3 [94] for checking the satisfiability of linear equations over the natural numbers. Although the code reproduces the algorithm as it is described in this chapter, it also implements some optimizations and additional rules to derive contradictions.

The tool takes an input file such as the one shown in Figure 3.9 and performs a proof search using our constraint solving rules guided by a heuristic. If the search is successful, the tool returns a representation of the proof tree. To ensure termination, we establish a timeout of 1000 seconds.

3.4.1 Case studies

We analyze the security of cryptographic constructions from the literature and collect the results in Table 3.1. All the experiments were executed on a 8-core machine with 2.40GHz Intel Core i7-3630QM CPU and 8GB of RAM.

The first five entries do not require support for oracles that take handles and

[3]Source code and case studies at https://github.com/generic-group-analyzer/gga-unbounded.

Reference	Scheme	Property	Time
Lysyanskaya et al. '99 [154]	LRSW assumption	Valid	2 s
Abe et al. '11 [7]	One-time SPS in Type I	OT-EUF-CMA	1 s
Pointcheval et al. '15 [167]	Assumption 1	Valid	1 s
Pointcheval et al. '15 [167]	Assumption 2	Valid	1 s
Pointcheval et al. '15 [167]	Multi-message sign. scheme ($r = 3$)	EUF-CMA	1 s
Chase et al. '13 [74]	$\text{MAC}_{\texttt{GGM}}$ (messages length $\leqslant 3$)	UF-CMVA	1 s
Chase et al. '13 [74]	$\text{MAC}_{\texttt{DDH}}$ (messages length $\leqslant 3$)	UF-CMVA	3 s
Abe et al. '11 [3]	SPS scheme, messages in $\mathbb{G}_1 \times \mathbb{G}_2$	sEUF-CMA	22 s
Abe et al. '14 [6]	Re-random. SPS for msg. in $\mathbb{G}_2$	EUF-CMA	6 s
Abe et al. '14 [7]	Unified SPS scheme	sEUF-CMA	5 s
Abe et al. '14 [7]	Unified SPS scheme (with tokens)	EUF-CMA	11 s
Chatterjee et al. '15 [75]	Type III randomizable SPS	EUF-CMA	3 s
Barthe et al. '15 [44]	Re-randomizable SPS in Type III	EUF-CMA	6 s
Groth '15 [118]	Fully comb. $\text{SPS}_{b=0}$ ($m, n = 1$)	EUF-CMA	8 s
Groth '15 [118]	Fully comb. $\text{SPS}_{b=1}$ ($m, n = 1$)	sEUF-CMA	8 s

Table 3.1: Case studies (last column denotes time for fully automated proof).

are therefore also in the scope of the tool presented in [43]. For the first four entries, both the tool from [43] and gga^{∞} prove unbounded security. For the fifth example, gga^{∞} succeeds, whereas the tool from [43] fails to find a proof.

The remaining examples are all outside the scope of the tool from [43]. First, we analyze the Message Authentication Codes proposed in [74]. They propose two MACs (instead of public key signatures) as the basis for their anonymous credential system. One of them is proven secure in the Generic Group Model and the other under the Decisional Diffie-Hellman (DDH) assumption (Definition 16). Our tool confirms the first proof and finds a proof in the Generic Group Model for the second construction[4].

We also prove security for a number of Structure-Preserving Signature schemes. First, we analyze the scheme proposed in [3] for bilinear groups of Type III. Then, we analyze the re-randomizable scheme from [6] for Type II and Type III. Next, we prove sEUF-CMA security of the unified SPS signature scheme proposed in [7], which is secure in all three settings. We also prove EUF-CMA security of its re-randomizable version (randomization tokens are given to the adversary). Later, we analyze the translation of the scheme for Type III proposed in [75]. We also consider the Type II scheme from [44]. Finally, we analyze two instances of fully structure-preserving signature schemes proposed in [118].

To evaluate our tool on a wider range of examples, we also make use of

[4]This is of course implied by the pen-and-paper proof under the DDH assumption.

	Search Space		Results	
	Verification equations	First signature elements	2-secure	∞-secure
II	$s_3 = f(r,v,w,m)$	$S_2 = [\![r]\!]_2$	1	1
	$s_3 s_2 = f(r,v,w,m)$	$S_2 = [\![r]\!]_2$	12	9
	$s_3(s_2 - w) = f(r,v,w,m)$	$S_2 = [\![r+w]\!]_2$	14	8
III	$s_1 = s_2 \wedge s_3 = f(r,v,w,m)$	$S_1 = [\![r]\!]_1,\ S_2 = [\![r]\!]_2$	2	2
	$s_1 = s_2 \wedge s_1 s_3 = f(r,v,w,m)$	$S_1 = [\![r]\!]_1,\ S_2 = [\![r]\!]_2$	117	75
	$s_1 s_2 = 1 \wedge s_1 s_3 = f(r,v,w,m)$	$S_1 = [\![r]\!]_1,\ S_2 = [\![r^{-1}]\!]_2$	39	22
			185	117

Table 3.2: Synthesis results for SPS schemes in Type II and Type III with $r, v, w \xleftarrow{\$} \mathbb{Z}_p$, verification keys $V = [\![v]\!]_1, W = [\![w]\!]_1 \in \mathbb{G}_1$, message $M = [\![m]\!]_2 \in \mathbb{G}_2$ and signatures $S_1 = [\![s_1]\!]_1 \in \mathbb{G}_1$, $S_2 = [\![s_2]\!]_2, S_3 = [\![s_3]\!]_2 \in \mathbb{G}_2$.

the synthesis tool for structure-preserving signature schemes presented in [44]. We take the existing results for Type II from [44] and use our tool to analyze (unbounded) EUF-CMA-security for all schemes where the the tool from [44] succeeds to prove 2-EUF-CMA security. We also extend the synthesis tool to generate new schemes in Type III and apply our tool to those schemes that can be proven 2-EUF-CMA secure with the tool from [44]. The results for both Type II and Type III are summarized in Table 3.2. We classify the schemes in different groups, depending on the shape of the verification equations (first column). The column 2-*secure* represents the number of schemes of each group that are proven 2-EUF-CMA secure using the tool from [44], while the column ∞-*secure* represents the number of schemes of each group that are proven EUF-CMA secure using our tool (for all bounds that are polynomial in the security parameter).

4

Attribute-Based Encryption: Algebraic Characterization of Privacy

L'essentiel est invisible pour les yeux.

Antoine de Saint-Exupéry, 1943

Predicate encodings [76, 190] are symmetric primitives that can be used for building predicate encryption schemes. We give an algebraic characterization of the notion of privacy from predicate encodings, and explore several of its consequences. Specifically, we propose more efficient predicate encodings for boolean formulae and arithmetic span programs, and generic optimizations of predicate encodings. We define new constructions to build boolean combination of predicate encodings. We formalize the relationship between predicate encodings and pair encodings [24], another primitive that can be transformed generically into predicate encryption schemes, and compare our constructions for boolean combinations of pair encodings with existing similar constructions from pair encodings. Finally, we demonstrate that our results carry to tag-based encodings [132].

4.1 Introduction

Predicate Encryption (PE) [66, 138] is a form of public-key encryption (PKE) that supports fine-grained access control for encrypted data. We refer to Section 2.1.5 from Chapter 2 for a formal definition.

Modular approaches for PE. In 2014, two independent works by Wee [190] and Attrapadung [24] proposed generic and unifying frameworks for obtaining efficient fully secure PE schemes for a large class of predicates. Both works use the dual system methodology introduced by Lewko and Waters [145, 188] and define a compiler that takes as input a relatively simple symmetric primitive and produces a fully secure PE construction. Wee introduced so-called *predicate encodings*, an information-theoretic primitive inspired from linear secret sharing. Attrapadung introduced so-called *pair encodings* and provided computational and information-theoretic security notions. These approaches greatly simplify the construction and analysis of predicate encryption schemes and share several advantages. First, they provide a good trade-off between *expressivity* and *performance*, while the security relies on standard and well studied assumptions. Second, they unify existing constructions into a single framework, i.e., previous PE constructions can be seen as instantiations of these new compilers with certain encodings. Third, building PE schemes by analyzing and building these simpler encodings is much easier than building PE schemes directly. Compared to full security for PE, the encodings must verify much weaker security requirements. The power of pair and predicate encodings is evidenced by the discovery of new constructions and efficiency improvements over existing ones. However, both approaches were designed over *composite order* bilinear groups. In 2015, Chen, Gay and Wee [76] and Attrapadung [25] respectively extended the predicate encoding and pair encoding compiler to the *prime order* setting. Next, Agrawal and Chase [9] improved on Attrapadung's work by relaxing the security requirement on *pair encodings* and thus, capturing new constructions. In addition, their work also brings both generic approaches closer together, because like in [76], the new compiler relies (in a black-box way) on Dual System Groups (DSG) [77, 78]. Additionally, Kim, Susilo, Guo, and Au [132] recently introduced a new generic framework for modular design of predicate encryption that improves on the performance of existing compilers. Their core primitive, *tag-based encodings*, is very similar to *predicate encodings*.

4.1.1 Prior work

Predicate encodings have been introduced in [190] and we use a refined version that is defined in [76] as our starting point. Both variants use an information-theoretic definition of the hiding while we show that there is an equivalent algebraic definition. Another related work is [111], initiating a systematic study of the communication complexity of the so-called conditional secret disclosure primitive, which is closely related to predicate encodings.

Other works also optimize existing predicate encryption schemes, for example many works focus on going from composite order constructions to the more efficient prime order ones [25, 76, 143]. In [76] they also propose performance improvements on dual system groups. We believe our optimizations via predicate encodings complement other possible enhancements of predicate encryption.

Boolean combinations of predicates have also been considered in the setting of pair encodings. Attrapadung [28, 30] proposes generic transformations for conjunction and for the dual predicate (see Definition 28), but neither for negation nor disjunction. We propose new transformations for conjunction and dual in the framework of predicate encodings and we also deal with negation and disjunction.

The main advantage of DP-ABE is the possibility of considering policies over *objective* attributes (associated to data) and policies over *subjective* attributes (associated to user credentials) at the same time. DP-ABE has been considered by Attrapadung in the pair encoding framework [28, 30]. To the best of our knowledge, we are the first to provide DP-ABE in the predicate encoding and tag-based encoding frameworks.

Revocation is a desirable property for PE and ABE schemes that has also been considered by many works in the literature. Revocation allows to invalidate a user's secret key in such a way that it becomes useless, even if its associated attribute satisfies the policy associated to the ciphertext. Some attempts [173] propose *indirect* revocation that requires that the master secret owner periodically updates secret keys for non-revoked users. Other attempts achieve *direct* revocation [27, 134, 152, 153], but either rely on strong assumptions or provide only selectively security. Our construction not only allows to achieve revocation in a *fully secure* framework, but it allows to add revocation to arbitrary predicate encodings.

Policy hiding is another property of PE, and ABE in particular, that has been broadly studied. In this context, policies associated to ciphertexts are not attached to them and therefore, unauthorized users will only learn the fact that their key does not satisfy the policy, but nothing else. Policy Hiding has been considered in several works [66, 138]. The security of our construction improves on earlier works, thanks to the compiler from [76]. Our observation extends the expressivity of attribute-hiding predicate encryption for ZIPE proposed in [76] to support policy-hiding for boolean formulas.

In [76], the authors introduce the notion of *spatial encryption* predicate encodings. We generalize this notion and introduce a transformation that makes delegation possible for every predicate encoding.

Several works evaluate the suitability of ABE for different applications. For example, ABE has been used and benchmarked to enforce privacy of Electronic Medical Records (EMR) [15], in a system where healthcare organizations export EMRs to external storage locations. Other examples are Sieve [187] or Streamforce [97], systems that provide enforced access control for user data and stream data in untrusted clouds. In contrast to these works, we are the first to evaluate predicate encryption and ABE based on modern modular approaches such as the predicate encoding and pair encoding frameworks. The resulting schemes also satisfy a stronger security notion (full vs. selective security) compared to the previously mentioned evaluations. We focus on synthetic case studies, while other works analyze more realistic settings and integration of ABE into bigger systems. Combining our methods with these more practical case studies is a very interesting direction for future work.

4.1.2 Comparison with Agrawal and Chase (EUROCRYPT 2017)

Concurrently and independently, Agrawal and Chase [11] introduce a new security notion, which they call *symbolic property*, for pair encodings. They adapt previous generic frameworks [9, 25] to define a compiler that takes pair encodings satisfying the symbolic property and produces fully secure predicate encryption schemes under the *q-ratio assumption*—a new assumption that is implied by some q-type assumptions proposed in [24, 148]. Moreover, they introduce the notion of *trivially broken* pair encoding and show that any not trivially broken pair encoding must satisfy their symbolic property. Their definitions of symbolic property and trivially broken for pair encodings are closely related to our algebraic characterization of privacy of predicate encodings. However, the two results are incomparable: although pair encodings are more general than predicate encodings (see Section 4.5.1 for a more detailed comparison), their results rely of *q-type* assumption, whereas our results build on previous frameworks that rely on weaker assumptions (Matrix-DH or k-LIN).

4.1.3 Notation

For a predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, we use $(x,y) \in \mathsf{P}$ as a shorthand for $\mathsf{P}(x,y) = 1$. We use the same conventions for matrix-representations of linear maps on finite-dimensional spaces. We define vectors $\boldsymbol{v} \in \mathbb{F}^n$ as column matrices and denote the *transpose* of a matrix A by $A^\top$. We use $\mathsf{diag}(\boldsymbol{v})$ to denote the diagonal matrix with main diagonal $\boldsymbol{v}$. We denote the identity matrix of dimension n by I_n, a zero vector of length n by $\mathbf{0}_n$ and a zero matrix of m rows and n columns by $\mathbf{0}_{m,n}$. Let S be a set of indices and A be a matrix. A_S denotes the matrix formed from the set of columns of A with indices is in S. We define the *colspan* of a matrix $M \in \mathbb{F}^{m \times n}$ as the set of all possible linear combinations columns of M. That is $\underset{span}{\overset{col}{}}\langle M \rangle = \{M\boldsymbol{v} : \boldsymbol{v} \in \mathbb{F}^n\} \subseteq \mathbb{F}^m$. We analogously define the *rowspan* of a matrix. We abuse of notation and write $[\![\boldsymbol{v}]\!]$ to denote $([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$ for vector $\boldsymbol{v} \in \mathbb{Z}_p^n$.

4.2 Background

In this section, we define *predicate encodings*, *tag-based encodings* and *pair encodings* the three primitives used in the three different modular frameworks for predicate encryption.

4.2.1 Predicate Encodings

Predicate encodings are information-theoretic primitives that can be used for building predicate encryption schemes [190]. We adopt the definition from [76], but prefer to use matrix notation.

Definition 24 (Predicate encoding)**.** *Let* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *be a binary predicate over finite sets* $\mathcal{X}$ *and* $\mathcal{Y}$*. Given a prime* $p \in \mathbb{N}$*, and* $s, r, w \in \mathbb{N}$*, a* (s,r,w)*-predicate encoding for* P *consists of five deterministic algorithms* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$*: the* sender encoding algorithm sE *maps* $x \in \mathcal{X}$ *into a matrix* $\mathsf{sE}_x \in \mathbb{Z}_p^{s \times w}$*, the* receiver encoding algorithm rE *maps* $y \in \mathcal{Y}$ *into a matrix* $\mathsf{rE}_y \in \mathbb{Z}_p^{r \times w}$*, the* key encoding algorithm kE *maps* $y \in \mathcal{Y}$ *into a vector* $\mathsf{kE}_y \in \mathbb{Z}_p^{r}$*, while the* sender *and* receiver decoding algorithms*, respectively* sD *and* rD*, map a pair* $(x,y) \in \mathcal{X} \times \mathcal{Y}$ *into vectors* $\mathsf{sD}_{x,y} \in \mathbb{Z}_p^{s}$ *and* $\mathsf{rD}_{x,y} \in \mathbb{Z}_p^{r}$ *respectively. We require that the following properties are satisfied:*

reconstructability: *for all* $(x,y) \in \mathsf{P}$, $\mathsf{sD}_{x,y}^{\top}\mathsf{sE}_x = \mathsf{rD}_{x,y}^{\top}\mathsf{rE}_y$ *and* $\mathsf{rD}_{x,y}^{\top}\mathsf{kE}_y = 1$*;*

α**-privacy:** *for all* $(x,y) \notin \mathsf{P}, \alpha \in \mathbb{Z}_p$*,*

$$\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^{w};\ \textsf{return}\ (\mathsf{sE}_x\boldsymbol{w},\ \mathsf{rE}_y\boldsymbol{w} + \alpha \cdot \mathsf{kE}_y) \ \equiv\ \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^{w};\ \textsf{return}\ (\mathsf{sE}_x\boldsymbol{w},\ \mathsf{rE}_y\boldsymbol{w})$$

where $\equiv$ *denotes equality of distributions.*

Reconstructability allows to recover α from $(x, y,\ \mathsf{sE}_x\boldsymbol{w},\ \mathsf{rE}_y\boldsymbol{w} + \alpha \cdot \mathsf{kE}_y)$ if $(x,y) \in \mathsf{P}$. *Privacy* ensures that α is perfectly hidden for such tuples if $(x,y) \notin \mathsf{P}$.

Example 6 (IBE predicate encoding)**.** *Let* $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p$ *and let* $s = r = 1$*,* $w = 2$*. We define the encoding functions as follows:*

$$\begin{aligned} \mathsf{sE}_x &= \begin{pmatrix} x & 1 \end{pmatrix} & \qquad \mathsf{sD}_{x,y} &= \begin{pmatrix} 1 \end{pmatrix} \\ \mathsf{rE}_y &= \begin{pmatrix} y & 1 \end{pmatrix} & \qquad \mathsf{rD}_{x,y} &= \begin{pmatrix} 1 \end{pmatrix} \\ \mathsf{kE}_y &= \begin{pmatrix} 1 \end{pmatrix} \end{aligned}$$

The above is a predicate encoding for Identity-Based Encryption, i.e., for the predicate $\mathsf{P}(x,y) = 1$ *iff* $x = y$*. Note that* $\begin{pmatrix} x & 1 \end{pmatrix} = \begin{pmatrix} y & 1 \end{pmatrix}$ *when* $x = y$*, so reconstructability is satisfied. On the other hand,* α*-privacy follows from the fact that if* $x \neq y$*,* $x \cdot w_1 + w_2$ *and* $y \cdot w_1 + w_2$ *are pair-wise independent.* ■

Predicate encryption from predicate encodings. We try to provide some intuition on how predicate encodings are compiled to predicate encryption schemes by the compiler from [76]. We consider a simplified compiler (see explanations below). The master keys, ciphertexts and secret keys have the following form:

$$\begin{aligned}\mathsf{msk} &= [\![\alpha]\!]_2\\ \mathsf{ct}_x &= ([\![s]\!]_1, [\![s \cdot \mathsf{sE}_x \boldsymbol{w}]\!]_1, [\![\alpha s]\!]_{\mathsf{t}} \cdot M)\\ \mathsf{mpk} &= ([\![1]\!]_1, [\![\boldsymbol{w}]\!]_1, [\![1]\!]_2, [\![\boldsymbol{w}]\!]_2, [\![\alpha]\!]_{\mathsf{t}})\\ \mathsf{sk}_y &= ([\![r]\!]_2, [\![\alpha \cdot \mathsf{kE}_y + r \cdot \mathsf{rE}_y \boldsymbol{w}]\!]_2)\end{aligned}$$

The encrypted message $M \in \mathbb{G}_t$ is blinded by a random factor $[\![\alpha s]\!]_{\mathsf{t}}$. The so-called *reconstruction* property of predicate encodings ensures that this blinding factor can be recovered for a pair $(\mathsf{ct}_x, \mathsf{sk}_y)$ if $\mathsf{P}(x, y) = 1$. More concretely, for all pairs (x, y) such that $\mathsf{P}(x, y) = 1$, because multiplying by matrices $\mathsf{sD}_{x,y}$,$\mathsf{rD}_{x,y}$ is a linear operation, it is possible operate in the exponent and compute

$$[\![s \cdot \mathsf{sD}_{x,y}^{\top} \mathsf{sE}_x \boldsymbol{w}]\!]_1 \quad \text{and} \quad [\![\mathsf{rD}_{x,y}^{\top}(\alpha \cdot \mathsf{kE}_y + r \cdot \mathsf{rE}_y \boldsymbol{w})]\!]_2,$$

obtaining $[\![s\beta]\!]_1$ and $[\![\alpha + r\beta]\!]_2$ for $\beta = \mathsf{sD}_{x,y}^{\top} \mathsf{sE}_x \boldsymbol{w} = \mathsf{rD}_{x,y}^{\top} \mathsf{rE}_y \boldsymbol{w}$ (note that knowing the value of β is not necessary). Now, it is simple to recover $[\![\alpha s]\!]_{\mathsf{t}}$ from $e([\![s]\!]_1, [\![\alpha + r\beta]\!]_2)$ and $e([\![s\beta]\!]_1, [\![r]\!]_2)$. Security is ensured by the α-*privacy* property of the encoding together with decisional assumptions about dual system groups. Intuitively, the α-privacy property states that given certain values derived from the output of the encoding functions for random input, α remains information-theoretic hidden.

Note that the following is a simplification of their compiler, where we avoid DSG for simplicity. The real scheme produced by their compiler would have twice as many group elements (under SXDH) or three times as many (under DLIN).

4.2.2 Tag-based encodings

Tag-based encodings are new primitives defined in a very recent work [132] that defines a new generic framework (using prime order groups) for modular design of predicate encryption.

Definition 25 (Tag-based encoding)**.** *Let* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *be a binary predicate over finite sets* $\mathcal{X}$ *and* $\mathcal{Y}$*. Given a prime* $p \in \mathbb{N}$*, and* $c, k, h \in \mathbb{N}$*, a* (c, k, h)*-tag-based encoding encoding for* P *consists of two deterministic algorithms* $(\mathsf{cE}, \mathsf{kE})$*: the* ciphertext encoding algorithm cE *maps* $x \in \mathcal{X}$ *into a matrix* $\mathsf{cE}_x \in \mathbb{Z}_p^{c \times h}$ *and the* key encoding algorithm kE *maps* $y \in \mathcal{Y}$ *into a matrix* $\mathsf{kE}_y \in \mathbb{Z}_p^{k \times h}$*. We require that the following properties are satisfied:*

reconstructability: *for all $(x, y) \in \mathsf{P}$, there exists an efficient algorithm that on input (x, y) computes vectors $\boldsymbol{m}_c \in \mathbb{Z}_p^c$, $\boldsymbol{m}_k \in \mathbb{Z}_p^k$ such that*

$$\boldsymbol{m}_c^\top \mathsf{cE}_x = \boldsymbol{m}_k^\top \mathsf{kE}_y \neq \mathbf{0}_h^\top$$

***h*-hiding:** *for all $(x, y) \notin \mathsf{P}$,*

$$\boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^h;\ \textsf{return}\ (\mathsf{cE}_x \boldsymbol{h},\ \mathsf{kE}_y \boldsymbol{h}) \quad \approx_s \quad \boldsymbol{h}, \boldsymbol{h}' \xleftarrow{\$} \mathbb{Z}_p^h;\ \textsf{return}\ (\mathsf{cE}_x \boldsymbol{h},\ \mathsf{kE}_y \boldsymbol{h}')$$

where $\approx_s$ denotes negligible statistical distance between distributions.

The compiler proposed in [132] uses similar ideas to the one for predicate encodings. However, it does not rely on dual system groups and can be instantiated with symmetric bilinear maps. The message is blinded and ciphertexts and keys contain a set of group elements that are enough to recover the blinding factor only when the predicate is true. This compiler has the advantage that some elements of ciphertexts and keys are $\mathbb{Z}_p$ values and not group elements, which reduces the storage size.

4.2.3 Pair Encodings

Attrapadung [24, 25] proposes an independent modular framework for predicate encryption, based on a primitive called *glspair-encoding*. For our purposes, it suffices to consider a more restrictive, information-theoretic, notion of pair encodings.

Definition 26 (Information-theoretic pair encoding)**.** *Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a binary predicate over finite sets $\mathcal{X}$ and $\mathcal{Y}$. Given a prime $p \in \mathbb{N}$, and $c, k, l, m, n \in \mathbb{N}$, let $\boldsymbol{h} = (h_1, \dots, h_n)$, $\boldsymbol{s} = (s_0, s_1, \dots, s_l)$ and $\boldsymbol{r} = (\alpha, r_1, \dots, r_m)$ be sets of variables. An information-theoretic (c, k, n)-pair encoding scheme for P consists of three deterministic algorithms $(\mathsf{Enc1}, \mathsf{Enc2}, \mathsf{Pair})$: the* ciphertext encoding algorithm $\mathsf{Enc1}$ *maps a value $x \in \mathcal{X}$ into a list of polynomials $\boldsymbol{c}_x \in \mathbb{Z}_p[\boldsymbol{s}, \boldsymbol{h}]^c$, the* key encoding algorithm $\mathsf{Enc2}$ *maps a value $y \in \mathcal{Y}$ into a list of polynomials $\boldsymbol{k}_y \in \mathbb{Z}_p[\boldsymbol{r}, \boldsymbol{h}]^k$ and the* decoding algorithm Pair *maps a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ into a matrix $E_{x,y} \in \mathbb{Z}_p^{c \times k}$. We require that the following properties are satisfied:*

polynomial constraints:

- *For every $x \in \mathcal{X}$ and every $f \in \mathsf{Enc1}(x)$, $f = f(\boldsymbol{s}, \boldsymbol{h})$ only contains monomials of the form s_i or $s_i h_j$, $i \in [0, l]$, $j \in [n]$.*
- *For every $y \in \mathcal{Y}$ and every $f \in \mathsf{Enc2}(y)$, $f = f(\boldsymbol{r}, \boldsymbol{h})$ only contains monomials of the form α, r_i or $r_i h_j$, $i \in [m]$, $j \in [n]$.*

reconstructability: *for all* $(x, y) \in \mathsf{P}$ *and all* $\boldsymbol{c}_x \leftarrow \mathsf{Enc1}(x)$, $\boldsymbol{k}_y \leftarrow \mathsf{Enc2}(y)$, $E_{x,y} \leftarrow \mathsf{Pair}(x, y)$, *the following polynomial equality holds* $\boldsymbol{c}_x^\top E_{x,y} \boldsymbol{k}_y = \alpha s_0$.

perfect security: *for all* $(x, y) \notin \mathsf{P}$ *and all* $\boldsymbol{c}_x \leftarrow \mathsf{Enc1}(x)$, $\boldsymbol{k}_y \leftarrow \mathsf{Enc2}(y)$,

$$\begin{array}{lll} \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^n;\ \boldsymbol{r} \xleftarrow{\$} (\mathbb{Z}_p^*)^m;\ \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{l+1}; & \textsf{return } (\boldsymbol{c}_x(\boldsymbol{s}, \boldsymbol{h}),\ \boldsymbol{k}_y(0, \boldsymbol{r}, \boldsymbol{h})) & \equiv \\ \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^n;\ \boldsymbol{r} \xleftarrow{\$} (\mathbb{Z}_p^*)^m;\ \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^{l+1}; \alpha \xleftarrow{\$} \mathbb{Z}_p; & \textsf{return } (\boldsymbol{c}_x(\boldsymbol{s}, \boldsymbol{h}),\ \boldsymbol{k}_y(\alpha, \boldsymbol{r}, \boldsymbol{h})) & \end{array}$$

where $\equiv$ *denotes equality of distributions.*

The compiler from pair encodings follows similar ideas to the other compilers. The message is blinded by a random factor and ciphertexts and keys contain all the information necessary to recover this blinded factor, only when the predicate holds. The compiler from pair encodings requires to compute a polynomial number of pairings during decryption, unlike the compilers for predicate encodings and tag-based encodings that need[1] 6 and 8 pairings respectively.

4.3 Predicate encodings: properties and consequences

In this section, we present a purely algebraic (and independent of α) characterization of the α-privacy property. It simplifies both the analysis and the construction of predicate encodings. In particular, we use our characterization to define and prove a new optimization of predicate encodings, i.e., a transformation that makes the encoding functions smaller while preserving the predicate. Additionally, we unify the reconstructability and privacy properties and show that they are mutually exclusive and complementary, i.e., for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, one and only one of the two conditions holds. This unified treatment facilitates the construction and study of predicate encodings.

4.3.1 Algebraic properties of predicate encodings

The following theorem captures two essential properties of predicate encodings: first, privacy admits a purely algebraic characterization (furthermore independent of α) given in terms of existence of solutions of a linear system of equations. Second, reconstructability and privacy, when viewed as properties of a single

[1]Decryption in the framework of predicate encodings needs 4 pairings under SXDH assumption or 6 under DLIN, in the framework of tag-based encodings decryption requires 8 pairings and the assumption is DLIN.

pair (x, y), negate each other; i.e. a pair (x, y) always satisfies exactly one of the two properties.

Theorem 3 (Algebraic characterization of privacy)**.** *Let $p \in \mathbb{N}$ be a prime, let $s, r, w \in \mathbb{N}$ and let $S \in \mathbb{Z}_p^{s\times w}$, $R \in \mathbb{Z}_p^{r\times w}$, $\boldsymbol{k} \in \mathbb{Z}_p^r$. The following are equivalent:*

- ***α-privacy.*** *For every $\alpha \in \mathbb{Z}_p$,*
$$\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (S\boldsymbol{w},\ R\boldsymbol{w} + \alpha\cdot\boldsymbol{k}) \quad\equiv\quad \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (S\boldsymbol{w},\ R\boldsymbol{w})$$

- ***(algebraic) privacy.*** *There exists a vector $\boldsymbol{w} \in \mathbb{Z}_p^w$ such that $S\boldsymbol{w} = \boldsymbol{0}_s$ and $R\boldsymbol{w} = \boldsymbol{k}$*

- ***non-reconstructability.*** *For every pair of vectors $\boldsymbol{s} \in \mathbb{Z}_p^s$ and $\boldsymbol{r} \in \mathbb{Z}_p^r$, either $\boldsymbol{s}^\top S \neq \boldsymbol{r}^\top R$ or $\boldsymbol{r}^\top\boldsymbol{k} \neq 1$.*

Proof. We first prove that α-privacy is equivalent to algebraic privacy. Note that the fact that $\forall\, \alpha \in \mathbb{Z}_p$,

$$\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (S\boldsymbol{w},\ R\boldsymbol{w} + \alpha\cdot\boldsymbol{k}) \quad\equiv\quad \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (S\boldsymbol{w},\ R\boldsymbol{w})$$

is equivalent to the existence of a bijection ρ_α such that for all $\boldsymbol{w} \in \mathbb{Z}_p^w$, $S\boldsymbol{w} = S\cdot\rho_\alpha(\boldsymbol{w}) \ \wedge\ R\boldsymbol{w} + \alpha\cdot\boldsymbol{k} = R\cdot\rho_\alpha(\boldsymbol{w})$. By linearity, it can be rewritten as

$$S(\rho_\alpha(\boldsymbol{w}) - \boldsymbol{w}) = \boldsymbol{0}_s \quad\wedge\quad \alpha\cdot\boldsymbol{k} = R(\rho_\alpha(\boldsymbol{w}) - \boldsymbol{w})$$

Now, the existence of such a bijection is equivalent to the existence of a solution for the following (parametric in α) linear system on $\boldsymbol{w}$: $S\boldsymbol{w} = \boldsymbol{0}_s \ \wedge\ R\boldsymbol{w} = \alpha\cdot\boldsymbol{k}$. To see this, note that if ρ_α is such a bijection, $\rho_\alpha(\boldsymbol{w}_0) - \boldsymbol{w}_0$ is a solution of the system for every $\boldsymbol{w}_0 \in \mathbb{Z}_p^w$. On the other hand, if $\boldsymbol{w}^*$ is a solution of the system, the bijection $\rho_\alpha(\boldsymbol{w}) = \boldsymbol{w} + \boldsymbol{w}^*$ satisfies the required identities. To conclude, note that the above system has a solution iff the following (independent of α) does:

$$S\boldsymbol{w} = \boldsymbol{0}_s \quad\wedge\quad R\boldsymbol{w} = \boldsymbol{k}$$

Next, we prove the equivalence between algebraic privacy and non-reconstructability. We use the following helping lemma from [53, Claim 2]: for every field $\mathbb{F}$, let $A \in \mathbb{F}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{F}^n$ be matrices with coefficients in $\mathbb{F}$, the following two statements are equivalent:

- for every $\boldsymbol{a} \in \mathbb{F}^m$, $\boldsymbol{b}^\top \neq \boldsymbol{a}^\top A$;

- there exists $\boldsymbol{z} \in \mathbb{F}^n$ such that $\boldsymbol{z}^\top\boldsymbol{b} = 1$ and $A\boldsymbol{z} = \boldsymbol{0}_m$.

Assume that algebraic privacy does not hold, i.e., for every $\boldsymbol{w} \in \mathbb{Z}_p^w$, either $S\boldsymbol{w} \neq \mathbf{0}_s$ or $R\boldsymbol{w} \neq \boldsymbol{k}$. Equivalently, for every $\boldsymbol{w} \in \mathbb{Z}_p^w$

$$\begin{pmatrix} \mathbf{0}_s \\ \boldsymbol{k} \end{pmatrix} \neq \begin{pmatrix} \text{-}S \\ R \end{pmatrix} \boldsymbol{w}$$

which is equivalent (by our helping lemma) to the existence of $(\boldsymbol{z}_1, \boldsymbol{z}_2) \in \mathbb{Z}_p^s \times \mathbb{Z}_p^r$ such that

$$\begin{pmatrix} \boldsymbol{z}_1^\top & \boldsymbol{z}_2^\top \end{pmatrix} \begin{pmatrix} \mathbf{0}_s \\ \boldsymbol{k} \end{pmatrix} = 1 \quad \wedge \quad \begin{pmatrix} \boldsymbol{z}_1^\top & \boldsymbol{z}_2^\top \end{pmatrix} \begin{pmatrix} \text{-}S \\ R \end{pmatrix} = \mathbf{0}_w^\top$$

That is, there exists $\boldsymbol{z}_1 \in \mathbb{Z}_p^s$, $\boldsymbol{z}_2 \in \mathbb{Z}_p^r$ such that $\boldsymbol{z}_1^\top S = \boldsymbol{z}_2^\top R \ \wedge \ \boldsymbol{z}_2^\top \boldsymbol{k} = 1$, which is exactly reconstructability. The proof follows from the fact all the steps are equivalences. □

Our next result is a representation theorem. It is based on the notion of partial encoding; informally, a partial encoding consists of the first three algorithms of a predicate encoding; it is not attached to any specific predicate, nor is required to satisfy any property.

Definition 27 (Partial encoding). *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $p \in \mathbb{N}$ be a prime and $s, r, w \in \mathbb{N}$. A (s, r, w)-partial encoding is given by three deterministic algorithms* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$*:* sE *maps $x \in \mathcal{X}$ into a matrix* $\mathsf{sE}_x \in \mathbb{Z}_p^{s\times w}$*, and* rE, kE *map $y \in \mathcal{Y}$ into a matrix* $\mathsf{rE}_y \in \mathbb{Z}_p^{r\times w}$ *and a vector* $\mathsf{kE}_y \in \mathbb{Z}_p^r$ *respectively.*

The representation theorem shows that there exists an embedding from partial encodings to predicate encodings, and that every predicate encoding lies the image of the embedding.

Theorem 4 (Representation theorem). *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $p \in \mathbb{N}$ be a prime and $s, r, w \in \mathbb{N}$. Every (s, r, w)-partial encoding* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$ *for $\mathcal{X}$ and $\mathcal{Y}$ induces a predicate encoding* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *for the following predicate (henceforth coined implicit predicate):*

$$\mathsf{P}(x, y) \triangleq \ \forall \boldsymbol{w} \in \mathbb{Z}_p^w, \quad \mathsf{sE}_x\boldsymbol{w} \neq \mathbf{0}_s \ \vee \ \mathsf{rE}_y\boldsymbol{w} \neq \mathsf{kE}_y$$

Moreover, if $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *is a predicate encoding for* P*, then for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$,* $\mathsf{P}(x, y) \Leftrightarrow \mathsf{P}(x, y)$*.*

Proof. The proof follows from Theorem 3 and the observation that reconstructability of predicate encodings is equivalent to P, while privacy of predicate encodings is equivalent to $\neg\,\mathsf{P}$. □

Example 7 (Implicit predicate of IBE predicate encoding)**.** *If we consider the following partial encoding functions corresponding to the encoding presented in Example 6:*

$$\mathsf{sE}_x = \begin{pmatrix} x & 1 \end{pmatrix} \qquad \mathsf{rE}_y = \begin{pmatrix} y & 1 \end{pmatrix} \qquad \mathsf{kE}_y = \begin{pmatrix} 1 \end{pmatrix}$$

our Theorem 4 guarantees that it is a valid predicate encoding for the implicit predicate:

$$\mathsf{P}(x,y) = 1 \;\textit{ iff }\; \forall (w_1, w_2) \in \mathbb{Z}_p^2,\;\; x \cdot w_1 + w_2 \neq 0 \;\;\vee\;\; y \cdot w_1 + w_2 \neq 1$$

A simple analysis shows that the above predicate is equivalent to $x = y$. ■

A consequence of Theorem 4 is that a predicate P over $\mathcal{X}$ and $\mathcal{Y}$ can be instantiated by a (s,r,w)-predicate encoding iff there exist $\mathcal{X}$-indexed and $\mathcal{Y}$-indexed matrices $S_x \in \mathbb{Z}_p^{s\times w}$ and $R_y \in \mathbb{Z}_p^{r\times w}$ and $\mathcal{Y}$-indexed vectors $\boldsymbol{k}_y \in \mathbb{Z}_p^r$ such that:

$$\mathsf{P}(x,y) = 1 \;\text{ iff }\; \begin{pmatrix} \mathbf{0}_s \\ \boldsymbol{k}_y \end{pmatrix} \notin \underset{\mathit{span}}{\mathit{col}} \left\langle \begin{matrix} S_x \\ R_y \end{matrix} \right\rangle$$

That is helpful to analyze the expressivity of predicate encodings of certain size.

Example 8. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets, let $n \in \mathbb{N}$, we will characterize all the predicates that can be achieved from a $(1,1,n)$-partial encoding, say $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE})$. Note that for every pair (x,y), sE_x and rE_y are vectors of length n, while kE_y is a single element. Say,*

$$\mathsf{sE}_x = (f_1(x), \dots, f_n(x)) \qquad \mathsf{rE}_y = (g_1(y), \dots, g_n(y)) \qquad \mathsf{kE}_y = h(y)$$

for certain functions $f_i : \mathcal{X} \to \mathbb{Z}_p$, $g_i, h : \mathcal{Y} \to \mathbb{Z}_p$ for every $i \in [n]$. Theorem 4 guarantees that the above is a valid predicate encoding for the predicate

$$\mathsf{P}(x,y) = 1 \;\textit{ iff }\; h(y) \neq 0 \;\wedge\; \Big(\exists \beta \in \mathbb{Z}_p : \bigwedge_{i\in[n]} f_i(x) = \beta g_i(y)\Big)$$

It can be shown that the predicate $\mathsf{P}((x_1,x_2),y) = 1$ iff $(x_1 = y) \vee (x_2 = y)$ cannot be captured by $(1,1,n)$-predicate encodings, while on the contrary, the predicate $\mathsf{P}((x_1,x_2),y) = 1$ iff $(x_1 = y) \wedge (x_2 = y)$ could be instantiated.

4.3.2 Optimizing predicate encodings

In this section, we show that the efficiency of predicate encodings can be improved by pre- and post-processing. Specifically, we show that an (s,r,w)-encoding $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ for a predicate P can be transformed into a (s',r',w')-encoding $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}', \mathsf{sD}', \mathsf{rD}')$ for the same predicate, by applying a linear transformation to the matrices induced by $\mathsf{sE}, \mathsf{rE}, \mathsf{kE}$.

More precisely, if we define $\mathsf{sE}'_x = A\mathsf{sE}_x$, $\mathsf{rE}'_y = B\mathsf{rE}_y$ and $\mathsf{kE}'_y = B\mathsf{kE}_y$ for two matrices A and B, the privacy of the encoding will be preserved, but reconstructability may be destroyed. On the contrary, when we consider the partial encoding $\mathsf{sE}'_x = \mathsf{sE}_x C$, $\mathsf{rE}'_y = \mathsf{rE}_y C$ and $\mathsf{kE}'_y = \mathsf{kE}_y$ for a matrix C, reconstructability is automatically guaranteed, but privacy could not hold (for the same predicate). Intuitively, this occurs because *reconstructability* depends on the *rowspan* of the matrices $\mathsf{sE}_x, \mathsf{rE}_y$, while *privacy* depends on their *colspan.* Our following theorem imposes conditions on these matrices A, B and C so that the resulting predicate encoding is equivalent to the original one.

Theorem 5. *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $p \in \mathbb{N}$ be a prime, $s, r, w, s', r', w' \in \mathbb{N}$, and let $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ be a (s, r, w)-predicate encoding for $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. Let A be a function that maps every element $x \in \mathcal{X}$ into a matrix $\mathsf{A}_x \in \mathbb{Z}_p^{s' \times s}$, B be a function that maps $y \in \mathcal{Y}$ into a matrix $\mathsf{B}_y \in \mathbb{Z}_p^{r' \times r}$ and let $\mathsf{C} \in \mathbb{Z}_p^{w \times w'}$ be a matrix. There exists a (s', r', w')-partial encoding $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}', \mathsf{sD}', \mathsf{rD}')$ for P, where*

$$\mathsf{sE}'_x = \mathsf{A}_x \mathsf{sE}_x \mathsf{C} \qquad \mathsf{rE}'_y = \mathsf{B}_y \mathsf{rE}_y \mathsf{C} \qquad \mathsf{kE}'_y = \mathsf{B}_y \mathsf{kE}_y$$

provided the following conditions hold:

- *For all $(x, y) \in \mathsf{P}$, $\mathsf{sD}_{x,y}^\top \in \underset{span}{\overset{row}{}}\langle \mathsf{A}_x \rangle$ and $\mathsf{rD}_{x,y}^\top \in \underset{span}{\overset{row}{}}\langle \mathsf{B}_y \rangle$;*
- *For all $(x, y) \notin \mathsf{P}$, there exists $\boldsymbol{w} \in \underset{span}{\overset{col}{}}\langle \mathsf{C} \rangle$ s.t. $\mathsf{sE}_x \boldsymbol{w} = \mathbf{0}_s$ and $\mathsf{rE}_y \boldsymbol{w} = \mathsf{kE}_y$.*

Proof. To see correctness of the new encoding, note that for all $(x, y) \in \mathsf{P}$, since

$$\mathsf{sD}_{x,y}^\top \in \underset{span}{\overset{row}{}}\langle \mathsf{A}_x \rangle \ \wedge\ \mathsf{rD}_{x,y}^\top \in \underset{span}{\overset{row}{}}\langle \mathsf{B}_y \rangle$$

there exist $\mathsf{sD}'^\top_{x,y}$ and $\mathsf{rD}'^\top_{x,y}$ such that

$$\mathsf{sD}_{x,y}^\top = \mathsf{sD}'^\top_{x,y} \mathsf{A}_x \ \wedge\ \mathsf{rD}_{x,y}^\top = \mathsf{rD}'^\top_{x,y} \mathsf{B}_y$$

Therefore,

$$\begin{aligned} &\mathsf{sD}'^\top_{x,y}(\mathsf{A}_x \mathsf{sE}_x \mathsf{C}) = (\mathsf{sD}_{x,y}^\top \mathsf{sE}_x)\mathsf{C} = (\mathsf{rD}_{x,y}^\top \mathsf{rE}_y)\mathsf{C} = \mathsf{rD}'^\top_{x,y}(\mathsf{B}_y \mathsf{rE}_y \mathsf{C}) \\ &\mathsf{rD}'^\top_{x,y}(\mathsf{B}_y \mathsf{kE}_y) = \mathsf{rD}_{x,y}^\top \mathsf{kE}_y = 1 \end{aligned}$$

To see privacy, note that for every $(x, y) \notin \mathsf{P}$, there exists $\boldsymbol{w} \in \underset{span}{\overset{col}{}}\langle \mathsf{C} \rangle$ such that $\mathsf{sE}_x \boldsymbol{w} = \mathbf{0}_s \wedge \mathsf{rE}_y \boldsymbol{w} = \mathsf{kE}_y$. Therefore, there also exists $\boldsymbol{w}' \in \mathbb{Z}_p^{w'}$ such that $\boldsymbol{w} = \mathsf{C}\boldsymbol{w}'$. Note that,

$$\begin{aligned} \mathsf{sE}'_x \boldsymbol{w}' &= (\mathsf{A}_x \mathsf{sE}_x \mathsf{C})\boldsymbol{w}' = \mathsf{A}_x \mathsf{sE}_x \boldsymbol{w} = \mathsf{A}_x \mathbf{0}_s = \mathbf{0}_{s'} \\ \mathsf{rE}'_y \boldsymbol{w}' &= (\mathsf{B}_y \mathsf{rE}_y \mathsf{C})\boldsymbol{w}' = \mathsf{B}_y \mathsf{rE}_y \boldsymbol{w} = \mathsf{B}_y \mathsf{kE}_y = \mathsf{kE}'_y \end{aligned}$$

so algebraic privacy is satisfied. □

This transformation is useful to make predicate encodings simpler and more efficient in different manners. For instance, it can be used to make the matrices corresponding to encoding and decoding functions become sparser. That is, if we consider A and B as functions that apply matrix *Gaussian elimination*[2] to the matrices associated to sE and rE, kE, many entries from these matrices will be zero. Hence, fewer group operations will be performed during encryption and key generation, improving the performance. Moreover, the transformation can be used to reduce the size of mpk, ct_x and sk_y. If $w' < w$, the number of elements in mpk will decrease. This will also improve the performance of encryption and key generation (both depend directly on mpk). Additionally, if $s' < s$ or $r' < r$, the number of elements in ct_x and sk_y will also decrease respectively.

Note that a simplification from the right (multiplying by C) changes the structure of the encoding and may open the possibility of left-simplifications that were not available before and vice versa. Example 9 illustrates this idea. We optimize a predicate encoding that corresponds to the result of applying our negation transformation (from next section, Theorem 8) to the predicate encoding from Example 6.

Example 9. *Let $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p$ and consider the $(2,3,4)$-predicate encoding $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ for $\mathsf{P}(x,y) = 1$ iff $x \neq y$, defined as*

$$\mathsf{sE}_x = \begin{pmatrix} x & \text{-1} & 0 & 0 \\ 1 & 0 & \text{-1} & 0 \end{pmatrix} \qquad \mathsf{rE}_y = \begin{pmatrix} 0 & 1 & 0 & y \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathsf{kE}_y = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\mathsf{sD}_{x,y}^\top = \begin{pmatrix} \frac{-1}{x-y} & \frac{x}{x-y} \end{pmatrix} \qquad \mathsf{rD}_{x,y}^\top = \begin{pmatrix} \frac{1}{x-y} & \frac{-x}{x-y} & 1 \end{pmatrix}$$

Note that for every pair $(x,y) \notin \mathsf{P}$, i.e. $x = y$, the single solution of the system $\mathsf{sE}_x \boldsymbol{w} = \mathbf{0}_2 \wedge \mathsf{rE}_y \boldsymbol{w} = \mathsf{kE}_y$ is $\boldsymbol{w}^\top = \begin{pmatrix} \text{-1} & -y & \text{-1} & 1 \end{pmatrix}$, thus the matrix

$$\mathsf{C} = \begin{pmatrix} \text{-1} & 0 & \text{-1} & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}^\top$$

satisfies the conditions of Theorem 5. Therefore, the $(2,3,2)$-partial encoding $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}')$, where

$$\mathsf{sE}'_x = \mathsf{sE}_x\mathsf{C} = \begin{pmatrix} -x & \text{-1} \\ 0 & 0 \end{pmatrix} \quad \mathsf{rE}'_y = \mathsf{rE}_y\mathsf{C} = \begin{pmatrix} y & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \quad \mathsf{kE}'_y = \mathsf{kE}_y = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

[2]Note that if matrices A_x, B_y or C are invertible, they always satisfy their respective requirements.

induces a predicate encoding for the same predicate. The previous simplification, opens the possibility of applying again the theorem, with matrices A_x *and* B_y*, obtaining a* $(1, 2, 2)$*-predicate encoding for* $\mathsf{P}(x, y) = 1$ *iff* $x \neq y$*. Concretely,*

$$\mathsf{A}_x = \begin{pmatrix} \text{-1} & 0 \end{pmatrix} \qquad \mathsf{sE}''_x = \begin{pmatrix} x & 1 \end{pmatrix} \qquad \mathsf{rE}''_y = \begin{pmatrix} y & 1 \\ 1 & 0 \end{pmatrix} \qquad \mathsf{rE}''_y = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\mathsf{B}_y = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathsf{sD}''^{\top}_{x,y} = \begin{pmatrix} \frac{1}{x-y} \end{pmatrix} \qquad \mathsf{rD}''^{\top}_{x,y} = \begin{pmatrix} \frac{1}{x-y} & 1 \end{pmatrix}$$

■

The above simplifications can be successfully applied to actual predicate encodings proposed in [76]. In Section 4.6.2.1 we propose improved predicate encodings for *monotonic boolean formulas* and *arithmetic span programs.*

4.3.3 Combining predicates

Using the new characterization of predicate encodings from the previous section, we define transformations to combine predicate encodings into new predicate encodings for more complex predicates. In particular, we define predicate encoding transformations for disjunction, conjunction, negation and the dual predicate. These combinations are useful to create new schemes that inherit different properties from the more basic building blocks. In Section 4.6, we propose several constructions that rely on these transformations.

4.3.3.1 Disjunction

We present a method to build a predicate encoding for the disjunction of P_1 and P_2 from predicate encodings for P_1 and P_2. Observe that the predicate encryption scheme obtained from the resulting predicate encoding is more efficient than the predicate encryption scheme obtained by compiling the predicate encodings of P_1 and P_2 separately, and then applying a generic transformation that builds predicate encryption schemes for a disjunction from predicate encryption schemes of its disjuncts.

Theorem 6 (Disjunction of predicate encodings)**.** *For every* (s_1, r_1, w_1)*-predicate encoding* $(\mathsf{sE}^1, \mathsf{rE}^1, \mathsf{kE}^1, \mathsf{sD}^1, \mathsf{rD}^1)$ *for* $\mathsf{P}_1 : \mathcal{X}_1 \times \mathcal{Y}_1 \to \{0, 1\}$ *and every* (s_2, r_2, w_2)*-predicate encoding* $(\mathsf{sE}^2, \mathsf{rE}^2, \mathsf{kE}^2, \mathsf{sD}^2, \mathsf{rD}^2)$ *for* $\mathsf{P}_2 : \mathcal{X}_2 \times \mathcal{Y}_2 \to \{0, 1\}$*, there exists a* $(s_1+s_2, r_1+r_2, w_1+w_2)$*-predicate encoding* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *for the predicate* $\mathsf{P} : (\mathcal{X}_1, \mathcal{X}_2) \times (\mathcal{Y}_1, \mathcal{Y}_2) \to \{0, 1\}$ *such that:*

$$\mathsf{P}((x_1, x_2), (y_1, y_2)) \Leftrightarrow \mathsf{P}_1(x_1, y_1) \vee \mathsf{P}_2(x_2, y_2)$$

Concretely,

$$\mathsf{sE}_{(x_1,x_2)} = \begin{pmatrix} \mathsf{sE}^1_{x_1} & \mathbf{0}_{s_1,w_2} \\ \mathbf{0}_{s_2,w_1} & \mathsf{sE}^2_{x_2} \end{pmatrix} \ \mathsf{rE}_{(y_1,y_2)} = \begin{pmatrix} \mathsf{rE}^1_{y_1} & \mathbf{0}_{r_1,w_2} \\ \mathbf{0}_{r_2,w_1} & \mathsf{rE}^2_{y_2} \end{pmatrix} \ \mathsf{kE}_{(y_1,y_2)} = \begin{pmatrix} \mathsf{kE}^1_{y_1} \\ \mathsf{kE}^2_{y_2} \end{pmatrix}$$

$$\begin{aligned} \mathsf{sD}^\top_{(x_1,x_2),(y_1,y_2)} &= \textit{if } \mathsf{P}_1(x_1,y_1) \textit{ then } \begin{pmatrix} \mathsf{sD}^{1\top}_{x_1,y_1} & \mathbf{0}^\top_{s_2} \end{pmatrix} \textit{ else } \begin{pmatrix} \mathbf{0}^\top_{s_1} & \mathsf{sD}^{2\top}_{x_2,y_2} \end{pmatrix} \\ \mathsf{rD}^\top_{(x_1,x_2),(y_1,y_2)} &= \textit{if } \mathsf{P}_1(x_1,y_1) \textit{ then } \begin{pmatrix} \mathsf{rD}^{1\top}_{x_1,y_1} & \mathbf{0}^\top_{r_2} \end{pmatrix} \textit{ else } \begin{pmatrix} \mathbf{0}^\top_{r_1} & \mathsf{rD}^{2\top}_{x_2,y_2} \end{pmatrix} \end{aligned}$$

Proof. Reconstructability can be seen by a simple check based on the reconstructability of the original encodings.
To see privacy, note that $\mathsf{P}_1(x_1,y_1) \vee \mathsf{P}_2(x_2,y_2) = 0$ implies $\mathsf{P}_1(x_1,y_1) = 0$ and $\mathsf{P}_2(x_2,y_2) = 0$ implies. Let $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ be witnesses of privacy of predicate encodings 1 and 2 respectively. It is easy to check that $\boldsymbol{w}^\top = \begin{pmatrix} \boldsymbol{w}_1^\top & \boldsymbol{w}_2^\top \end{pmatrix}$ is a witness of privacy of the transformed encoding. □

Note that it is possible to obtain sharing between attributes, e.g., if $\mathcal{X}_1 = \mathcal{X}_2$ and the sender uses only the subset $\{(x,x) \mid x \in \mathcal{X}_1\}$ of $\mathcal{X}_1 \times \mathcal{X}_2$, the predicate becomes $\mathsf{P}(x,(y_1,y_2)) = 1$ iff $\mathsf{P}_1(x,y_1) \vee \mathsf{P}_2(x,y_2)$.

4.3.3.2 Conjunction

In contrast to disjunction, the naive solution that just concatenates secret keys fails. Given keys for attribute pairs (y_1,y_2) and (y_1',y_2'), it would be possible to recombine the components and obtain a key for (y_1,y_2') leading to collusion attacks. Our predicate encoding transformation deals with this problem by "tying" the two components together with additional randomness.

Theorem 7 (Conjunction of predicate encodings)**.** *For every (s_1,r_1,w_1)-predicate encoding $(\mathsf{sE}^1,\mathsf{rE}^1,\mathsf{kE}^1,\mathsf{sD}^1,\mathsf{rD}^1)$ for $\mathsf{P}_1 : \mathcal{X}_1 \times \mathcal{Y}_1 \to \{0,1\}$ and every (s_2,r_2,w_2)-predicate encoding $(\mathsf{sE}^2,\mathsf{rE}^2,\mathsf{kE}^2,\mathsf{sD}^2,\mathsf{rD}^2)$ for $\mathsf{P}_2 : \mathcal{X}_2 \times \mathcal{Y}_2 \to \{0,1\}$, there exists a $(s_1+s_2,\ r_1+r_2,\ w_1+w_2+1)$-predicate encoding $(\mathsf{sE},\mathsf{rE},\mathsf{kE},\mathsf{sD},\mathsf{rD})$ for the predicate $\mathsf{P} : (\mathcal{X}_1,\mathcal{X}_2) \times (\mathcal{Y}_1,\mathcal{Y}_2) \to \{0,1\}$ such that:*

$$\mathsf{P}((x_1,x_2),(y_1,y_2)) \Leftrightarrow \mathsf{P}_1(x_1,y_1) \wedge \mathsf{P}_2(x_2,y_2)$$

Concretely,

$$\begin{aligned} \mathsf{sE}_{(x_1,x_2)} &= \begin{pmatrix} \mathsf{sE}^1_{x_1} & \mathbf{0}_{s_1,w_2} & \mathbf{0}_{s_1} \\ \mathbf{0}_{s_2,w_1} & \mathsf{sE}^2_{x_2} & \mathbf{0}_{s_2} \end{pmatrix} & \mathsf{sD}_{(x_1,x_2),(y_1,y_2)} &= \frac{1}{2}\begin{pmatrix} \mathsf{sD}^1_{x_1,y_1} \\ \mathsf{sD}^2_{x_2,y_2} \end{pmatrix} \\ \mathsf{rE}_{(y_1,y_2)} &= \begin{pmatrix} \mathsf{rE}^1_{y_1} & \mathbf{0}_{r_1,w_2} & \mathsf{kE}^1_{y_1} \\ \mathbf{0}_{r_2,w_1} & \mathsf{rE}^2_{y_2} & \text{-}\mathsf{kE}^2_{y_2} \end{pmatrix} & \mathsf{rD}_{(x_1,x_2),(y_1,y_2)} &= \frac{1}{2}\begin{pmatrix} \mathsf{rD}^1_{x_1,y_1} \\ \mathsf{rD}^2_{x_2,y_2} \end{pmatrix} \\ \mathsf{kE}_{(y_1,y_2)} &= \begin{pmatrix} \mathsf{kE}^1_{y_1} \\ \mathsf{kE}^2_{y_2} \end{pmatrix} \end{aligned}$$

Proof. A simple check shows reconstructability. To see privacy, $\mathsf{P}_1(x_1, y_1) \wedge \mathsf{P}_2(x_2, y_2) = 0$ implies $\mathsf{P}_1(x_1, y_1) = 0$ or $\mathsf{P}_2(x_2, y_2) = 0$. If the first holds, let $\boldsymbol{w}_1$ be a witness of privacy of the first encoding. Then, $\boldsymbol{w}^\top = \begin{pmatrix} 2\boldsymbol{w}_1^\top & \mathbf{0}_{w_2}^\top & \text{-}1 \end{pmatrix}$ is a witness of the algebraic privacy of the transformed encoding. If the second holds, let $\boldsymbol{w}_2$ be a witness of privacy of the second encoding. A valid witness now is $\boldsymbol{w}^\top = \begin{pmatrix} \mathbf{0}_{w_2}^\top & 2\boldsymbol{w}_2^\top & 1 \end{pmatrix}$. □

Note that it is possible to combine Theorems 6 and 7 to create a predicate encoding for $\mathsf{P}_1 \bowtie \mathsf{P}_2$, where the placeholder $\bowtie \in \{\vee, \wedge\}$ can be part of keys or ciphertexts. See Section 4.7 for more details about this encoding.

4.3.3.3 Negation

To obtain a functionally complete set of boolean predicate encoding transformers, we now define a transformation for negation. Our transformation unifies negated predicates like Non-zero Inner-Product Encryption (NIPE) and Zero Inner-Product Encryption (ZIPE). In Section 4.6.2.1 we use this transformation to build optimized predicate encodings. The technique works for predicate encodings where the negation transformation yields a predicate encoding that can be further simplified (using our method from Section 4.3.2).

Theorem 8 (Negation of predicate encodings)**.** *For every (s, r, w)-predicate encoding $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ for $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ there exists a $(w, w+1, s+w+r)$-predicate encoding $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}', \mathsf{sD}', \mathsf{rD}')$ for the predicate $\mathsf{P}' : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ such that $\mathsf{P}'(x, y) \Leftrightarrow \neg\mathsf{P}(x, y)$. Concretely,*

$$\mathsf{sE}'_x = \begin{pmatrix} \mathsf{sE}_x^\top & \text{-}I_w & \mathbf{0}_{w,r} \end{pmatrix} \quad \mathsf{rE}'_y = \begin{pmatrix} \mathbf{0}_{w,s} & I_w & \mathsf{rE}_y^\top \\ \mathbf{0}_s^\top & \mathbf{0}_w^\top & \mathsf{kE}_y^\top \end{pmatrix} \quad \mathsf{kE}'_y = \begin{pmatrix} \mathbf{0}_w \\ 1 \end{pmatrix}$$

$$\mathsf{sD}'_{x,y} = \boldsymbol{w}_{x,y} \qquad\qquad \mathsf{rD}'_{x,y} = \begin{pmatrix} \text{-}\boldsymbol{w}_{x,y} \\ 1 \end{pmatrix}$$

where for a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $\mathsf{P}(x, y) = 0$, $\boldsymbol{w}_{x,y}$ is defined as the witness for algebraic privacy, i.e., a vector such that

$$\mathsf{sE}_x \boldsymbol{w}_{x,y} = \mathbf{0}_s \quad \wedge \quad \mathsf{rE}_y \boldsymbol{w}_{x,y} = \mathsf{kE}_y$$

Note that such a vector always exists when $\mathsf{P}(x, y) = 0$. Moreover, sD and rD do not need to be defined when $\mathsf{P}'(x, y)$ is not 1, that is, when $\mathsf{P}(x, y)$ is not 0.

Proof. It is not difficult to check reconstructability. Privacy holds because when $\mathsf{P}(x, y) = 1$, we can define $\boldsymbol{w}^\top = \begin{pmatrix} \text{-}\mathsf{sD}_{x,y}^\top & \text{-}\mathsf{sD}_{x,y}^\top \mathsf{sE}_x & \mathsf{rD}_{x,y}^\top \end{pmatrix}$ which can be checked to be a witness of the algebraic privacy of the transformed predicate encoding. □

A similar construction has been considered in a posterior work [20] to this work. Specifically, they show how to transform a *conditional disclosure of secrets* (CDS) for f into a CDS for $\bar{f}$ (the complement of f).

4.3.3.4 Dual

In the literature, the notions of KP-ABE and CP-ABE are considered separately. In fact, many works are only valid for one of the two versions of Attribute Based Encryption. Our transformation unifies the notion of KP-ABE and CP-ABE in the framework of predicate encodings. In this context they should not be considered separately, because our transformation provides a Ciphertext-Policy predicate encoding from any Key-Policy predicate encoding and vice versa.

Definition 28 (Dual predicate). *Let* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *be a predicate, the* dual predicate *of* P *is* $\mathsf{P}' : \mathcal{Y} \times \mathcal{X} \to \{0,1\}$ *such that* $\mathsf{P}'(y,x) \Leftrightarrow \mathsf{P}(x,y)$*.*

Theorem 9 (Dual predicate encoding)**.** *For every* (s,r,w)*-predicate encoding* $(\mathsf{sE},\mathsf{rE},\mathsf{kE},\mathsf{sD},\mathsf{rD})$ *for* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ *there exists a* $(r, s+1, w+1)$*-predicate encoding* $(\mathsf{sE}',\mathsf{rE}',\mathsf{kE}',\mathsf{sD}',\mathsf{rD}')$ *for the predicate* $\mathsf{P}' : \mathcal{Y} \times \mathcal{X} \to \{0,1\}$ *such that* $\mathsf{P}'(y,x) \Leftrightarrow \mathsf{P}(x,y)$*. Concretely,*

$$\mathsf{sE}'_y = \begin{pmatrix} \mathsf{rE}_y & \mathsf{kE}_y \end{pmatrix} \qquad \mathsf{rE}'_x = \begin{pmatrix} \mathsf{sE}_x & \mathbf{0}_s \\ \mathbf{0}_w^\top & 1 \end{pmatrix} \qquad \mathsf{kE}'_x = \begin{pmatrix} \mathbf{0}_s \\ 1 \end{pmatrix}$$

$$\mathsf{sD}'_{y,x} = \mathsf{rD}_{x,y} \qquad\qquad \mathsf{rD}'_{y,x} = \begin{pmatrix} \mathsf{sD}_{x,y} \\ 1 \end{pmatrix}$$

Proof. A simple check is enough to verify reconstructability. For privacy, note that when $\mathsf{P}'(y,x) = 0$, we have $\mathsf{P}(x,y) = 0$. Let $\boldsymbol{w}$ be a witness of the algebraic privacy of the original encoding. Now, $\boldsymbol{w}'^\top = \begin{pmatrix} -\boldsymbol{w}^\top & 1 \end{pmatrix}$ is a witness of the dual predicate encoding. □

4.4 Tag-based Encodings

We show that our techniques for *predicate encodings* can be extended to the framework of *tag-based encodings*. In particular, we show a similar result to our Theorem 3, which establishes that $\boldsymbol{h}$-hiding and reconstructability are mutually exclusive and complementary.

Theorem 10. *Let* $p \in \mathbb{N}$ *be a prime, let* $k,c,h \in \mathbb{N}$ *and let* $C \in \mathbb{Z}_p^{c\times h}$*,* $K \in \mathbb{Z}_p^{k\times h}$*. The following are equivalent:*

- ***h*-hiding:** $\boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^h$; *return* $(C\boldsymbol{h},\, K\boldsymbol{h}) \;\equiv\; \boldsymbol{h}, \boldsymbol{h}' \xleftarrow{\$} \mathbb{Z}_p^h$; *return* $(C\boldsymbol{h},\, K\boldsymbol{h}')$
- ***non-reconstructability*** *For every* $\boldsymbol{m}_c \in \mathbb{Z}_p^c$ *and very* $\boldsymbol{m}_k \in \mathbb{Z}_p^k$*, either* $\boldsymbol{m}_c^\top C \neq \boldsymbol{m}_k^\top K$ *or* $\boldsymbol{m}_c^\top C = \boldsymbol{0}_h^\top$.

where $\equiv$ *denotes equality of distributions.*

Proof. The proof follows directly from the following lemma and the observation that *i)* is equivalent to $\boldsymbol{h}$-hiding, while *iii)* is non-reconstructability (take $A = C$ and $B = K$). □

Lemma 2. *Let* $A \in \mathbb{Z}_p^{m\times n}$ *and* $B \in \mathbb{Z}_p^{l\times n}$ *be matrices. Let* $C \in \mathbb{Z}_p^{(m+l)\times n}$ *be the concatenation of* A *and* B *by rows. The following three statements are equivalent:*

i) $$\forall \boldsymbol{a} \in \mathbb{Z}_p^m, \forall \boldsymbol{b} \in \mathbb{Z}_p^l,\quad \Pr_{\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_p^n}[A\boldsymbol{x} = \boldsymbol{a} \,|\, B\boldsymbol{x} = \boldsymbol{b}] = \Pr_{\boldsymbol{x} \xleftarrow{\$} \mathbb{Z}_p^n}[A\boldsymbol{x} = \boldsymbol{a}]$$

ii) $$\mathrm{rank}(C) = \mathrm{rank}(A) + \mathrm{rank}(B)$$

iii) $$\forall \boldsymbol{a} \in \mathbb{Z}_p^m, \forall \boldsymbol{b} \in \mathbb{Z}_p^l,\quad \boldsymbol{a}^\top A \neq \boldsymbol{b}^\top B \;\vee\; \boldsymbol{a}^\top A = \boldsymbol{0}_n^\top$$

Of the Lemma. Note that *i)* holds for every $\boldsymbol{a} \in \mathbb{Z}_p^m, \boldsymbol{b} \in \mathbb{Z}_p^l$ such that $A\boldsymbol{x} = \boldsymbol{a}$ or $B\boldsymbol{x} = \boldsymbol{b}$ have no solution. Let $\boldsymbol{a} \in \mathbb{Z}_p^m, \boldsymbol{b} \in \mathbb{Z}_p^l$ be such that the systems $A\boldsymbol{x} = \boldsymbol{a}$ and $B\boldsymbol{x} = \boldsymbol{b}$ have individually at least one solution (note that such $\boldsymbol{a}$ and $\boldsymbol{b}$ always exist). We define the sets $\Gamma_A = \{\boldsymbol{x} \in \mathbb{Z}_p^n : A\boldsymbol{x} = \boldsymbol{a}\}$, $\Gamma_B = \{\boldsymbol{x} \in \mathbb{Z}_p^n : B\boldsymbol{x} = \boldsymbol{b}\}$, $\Gamma_{AB} = \{\boldsymbol{x} \in \mathbb{Z}_p^n : A\boldsymbol{x} = \boldsymbol{a} \;\wedge\; B\boldsymbol{x} = \boldsymbol{b}\}$. By the Rouché-Capelli Theorem,

$$|\Gamma_A| = p^{n-\mathrm{rank}(A)} \qquad |\Gamma_B| = p^{n-\mathrm{rank}(B)} \qquad |\Gamma_{AB}| = p^{n-\mathrm{rank}(C)}$$

Note that *i)* can be expressed as $\frac{|\Gamma_{AB}|}{p^n} = \frac{|\Gamma_A|}{p^n} \cdot \frac{|\Gamma_B|}{p^n}$ which is equivalent to the equation $p^n \cdot |\Gamma_{AB}| = |\Gamma_A| \cdot |\Gamma_B|$, and therefore, $p^n \cdot p^{n-\mathrm{rank}(C)} = p^{n-\mathrm{rank}(A)} \cdot p^{n-\mathrm{rank}(B)}$ if, and only if, $\mathrm{rank}(C) = \mathrm{rank}(A) + \mathrm{rank}(B)$ which is *ii)*.

Now, note that $\mathrm{rank}(C) = \mathrm{rank}(A) + \mathrm{rank}(B)$ if, and only if, there is not a non-zero linear combination of rows of A that can be expressed as a linear combination of rows of B, which is equivalent to statement *iii)*. □

A consequence of Theorem 10 is that every valid tag-based encoding is perfectly hiding, or equivalently, there cannot exist a tag-based encoding where the two distributions from $\boldsymbol{h}$-hiding are negligibly close but not identical.

Thanks to the above theorem, it is possible to define *disjunction* and *conjunction* transformations for tag-based encodings along the lines of predicate encodings. We were not able to design a negation transformation for tag-based encodings and leave it for future work. On the other hand, the dual transformation is straightforward in this framework, as mentioned in [132], because the encoding primitives are completely symmetric.

Expressivity of tag-based encodings vs predicate encodings. We try to improve our understanding of the differences between these two primitives by providing a transformation that produces valid predicate encodings from valid tag-based encodings for the same predicate.

Theorem 11. *Given a $(c, 1, h)$-tag-based encoding* $(\mathsf{cE}, \mathsf{kE})$ *for* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$*, the $(c, 1, h)$-partial predicate encoding* $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}')$ *defined as* $\mathsf{sE}'_x = \mathsf{cE}_x$*,* $\mathsf{rE}'_y = \mathsf{kE}_y$*,* $\mathsf{kE}'_y = \begin{pmatrix} 1 \end{pmatrix}$*, induces a predicate encoding for* P.

Proof. According to our Theorem 4, the partial encoding $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}')$ induces a predicate encoding for the predicate $\mathsf{P}(x, y) = 1$ iff $\exists \boldsymbol{s} \in \mathbb{Z}_p^c, r \in \mathbb{Z}_p^1$ s.t. $\boldsymbol{s}^\top \mathsf{sE}'_x = r \cdot \mathsf{rE}'_y$ and $r \cdot \mathsf{kE}'_y = 1$, or equivalently, $\exists \boldsymbol{s} \in \mathbb{Z}_p^c$ s.t. $\boldsymbol{s}^\top \mathsf{cE}_x = \mathsf{kE}_y$, which is equivalent to the reconstructability of the tag-based encoding $(\mathsf{cE}, \mathsf{kE})$. According to Theorem 10 it is also equivalent to the predicate P. □

Note that because of the symmetry of tag-based encodings, Theorem 11 can be also applied to $(1, k, h)$-tag-based encodings. All the tag-based encodings proposed in [132] (except one) have either $c = 1$ or $k = 1$, so the above theorem can be applied to them.

4.5 Pair Encodings

In this section we provide an embedding that transforms every predicate encoding into an information-theoretic pair encoding. Consequently, we can see predicate encodings as a subclass of pair encodings. This opens the possibility of reusing the conjunction and dual transformation proposed by Attrapadung [28, 30] for pair encodings, to create combinations of predicate encodings via our embedding. We show that this alternative method is fundamentally different from our direct conjunction and dual transformations on predicate encodings, where our combinations produce more efficient encodings.

4.5.1 Embedding Predicate Encodings into Pair Encodings

In this section we provide an embedding that produces a valid *information-theoretic pair encoding* from every valid predicate encoding (see Definitions 24 and 26 for predicate encodings and pair encodings respectively).

Definition 29 (Embedding to Pair Encodings)**.** *Given a (s, r, w)-predicate encoding pe = $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$, we define the embedding* $\mathsf{Emb}(pe) = (\mathsf{Enc1}_{pe}, \mathsf{Enc2}_{pe}, \mathsf{Pair}_{pe})$ *as follows:*

- $\mathsf{Enc1}_{pe}(x) = (c_0, \boldsymbol{c})$, *where* $c_0(s_0, \boldsymbol{h}) = s_0$, $\boldsymbol{c}(s_0, \boldsymbol{h}) = s_0 \cdot \mathsf{sE}_x \boldsymbol{h}$
- $\mathsf{Enc2}_{pe}(y) = (k_0, \boldsymbol{k})$, *where* $k_0(\alpha, r_1, \boldsymbol{h}) = r_1$, $\boldsymbol{k}(\alpha, r_1, \boldsymbol{h}) = \alpha \cdot \mathsf{kE}_y + r_1 \cdot \mathsf{rE}_y \boldsymbol{h}$
- $\mathsf{Pair}_{pe}(x, y) = \begin{pmatrix} 0 & \mathsf{rD}_{x,y}^\top \\ \text{-}\mathsf{sD}_{x,y} & \mathbf{0}_{s,r} \end{pmatrix}$

All variables $\boldsymbol{s} = (s_0)$ and $\boldsymbol{r} = (r_1)$ appear in the clear in the $\mathsf{Enc1}$ and $\mathsf{Enc2}$ polynomials respectively. This simplifies the pair encoding's information-theoretical security notion into one equivalent to the privacy of the predicate encoding (see proof of Theorem 12).

Theorem 12 (Correctness of the embedding)**.** *If pe = $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ is a valid (s, r, w)-predicate encoding for* P*, then* $\mathsf{Emb}(pe)$ *is a valid information theoretic $(s + 1,\ r + 1,\ w)$-pair encoding for* P*.*

Proof. Verifying correctness of the pair encoding is a simple check. For perfect security we need to check that, when $(x, y) \notin \mathsf{P}$, the following two distributions are identical:

$$\alpha, s_0 \xleftarrow{\$} \mathbb{Z}_p;\ r_1 \xleftarrow{\$} \mathbb{Z}_p^*;\ \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^w;\ \mathsf{return}\ (s_0,\ s_0 \cdot \mathsf{sE}_x \boldsymbol{h},\ r_1,\ r_1 \cdot \mathsf{rE}_y \boldsymbol{h}) \quad \equiv$$
$$s_0 \xleftarrow{\$} \mathbb{Z}_p;\ r_1 \xleftarrow{\$} \mathbb{Z}_p^*;\ \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^w;\ \mathsf{return}\ (s_0,\ s_0 \cdot \mathsf{sE}_x \boldsymbol{h},\ r_1,\ r_1 \cdot \mathsf{rE}_y \boldsymbol{h} + \alpha \cdot \mathsf{kE}_y)$$

Since both distributions provide s_0 and r_1 in the clear, the above checking is equivalent to the following:

$$\boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^w;\ \mathsf{return}\ (\mathsf{sE}_x \boldsymbol{h},\ \mathsf{rE}_y \boldsymbol{h}) \quad \equiv$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p;\ r_1 \xleftarrow{\$} \mathbb{Z}_p^*;\ \boldsymbol{h} \xleftarrow{\$} \mathbb{Z}_p^w;\ \mathsf{return}\ (\mathsf{sE}_x \boldsymbol{h},\ \mathsf{rE}_y \boldsymbol{h} + \alpha / r_1 \cdot \mathsf{kE}_y)$$

but those distributions are identical due to the α-privacy of the predicate encoding[3]. □

Our embedding shows that every predicate encoding can be transformed into a perfectly secure pair encoding. In fact, after applying the compiler from [9] to the embedding of a predicate encoding, we get the same predicate encryption scheme that the one provided by the compiler from [76].

We conclude that *predicate encodings* can be transformed into a very special class of *pair encodings*: encodings that allow decryption with 2 pairings and have only one element of randomness in both, ciphertexts and secret keys (what makes them very efficient).

[3]Note that $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $r_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and therefore, α / r_1 distributes uniformly over $\mathbb{Z}_p$, so we can apply the α-privacy property from the predicate encoding.

4.5.2 Comparison between encoding transformations

Attrapadung proposed generic transformations of pair encodings [28, 30]. In particular, he proposed the conjunction and dual transformations. In this section we compare these transformations with the ones proposed in this work. For this, we compare the conjunction of two pair encodings, (embedded from predicate encodings) with the embedding of the conjunction of a (s_1, r_1, w_1)-predicate encoding $pe^1 = (\mathsf{sD}^1, \mathsf{rE}^1, \mathsf{kE}^1, \mathsf{sD}^1, \mathsf{rD}^1)$ and a (s_2, r_2, w_2)-predicate encoding $pe^2 = (\mathsf{sD}^2, \mathsf{rE}^2, \mathsf{kE}^2, \mathsf{sD}^2, \mathsf{rD}^2)$, i.e.,

$$\mathsf{Emb}(pe^1 \wedge_{pred} pe^2) \quad vs \quad \mathsf{Emb}(pe^1) \wedge_{pair} \mathsf{Emb}(pe^2)$$

where $\wedge_{pred}$ and $\wedge_{pair}$ are the conjunction of predicate encodings and pair encodings respectively. Note that $\wedge_{pred}$ corresponds to the transformation from our Theorem 7. On the other hand, for $\wedge_{pair}$ we use the conjunction proposed in [30].

$$\mathsf{Emb}(pe^1 \wedge_{pred} pe^2) = \begin{cases} \mathsf{Enc1}((x_1, x_2)) = (c_0, \boldsymbol{c}_1, \boldsymbol{c}_2) \\ \mathsf{Enc2}((y_1, y_2)) = (k_0, \boldsymbol{k}_1, \boldsymbol{k}_2) \\ \mathsf{Pair}((x_1, x_2), (y_1, y_2)) = E_{(x_1,x_2),(y_1,y_2)} \end{cases}$$

where $\boldsymbol{h} = (h_0, \boldsymbol{h}_1, \boldsymbol{h}_2)$ and

$$\begin{aligned} c_0(s_0, \boldsymbol{h}) &= s_0 \\ \boldsymbol{c}_1(s_0, \boldsymbol{h}) &= s_0 \cdot \mathsf{sE}^1_{x_1} \boldsymbol{h}_1 \\ \boldsymbol{c}_2(s_0, \boldsymbol{h}) &= s_0 \cdot \mathsf{sE}^2_{x_2} \boldsymbol{h}_2 \end{aligned} \qquad \begin{aligned} k_0(\alpha, r_1, \boldsymbol{h}) &= r_1 \\ \boldsymbol{k}_1(\alpha, r_1, \boldsymbol{h}) &= (\alpha + h_0) \cdot \mathsf{kE}^1_{y_1} + r_1 \cdot \mathsf{rE}^1_{y_1} \boldsymbol{h}_1 \\ \boldsymbol{k}_2(\alpha, r_1, \boldsymbol{h}) &= (\alpha - h_0) \cdot \mathsf{kE}^2_{y_2} + r_1 \cdot \mathsf{rE}^2_{y_2} \boldsymbol{h}_2 \end{aligned}$$

$$E_{(x_1,x_2),(y_1,y_2)} = \frac{1}{2} \begin{pmatrix} 0 & \mathsf{rD}^{1\top}_{x_1,y_1} & \mathsf{rD}^{2\top}_{x_2,y_2} \\ \text{-}\mathsf{sD}^1_{x_1,y_1} & \mathbf{0}_{s_1,r_1} & \mathbf{0}_{s_1,r_2} \\ \text{-}\mathsf{sD}^2_{x_2,y_2} & \mathbf{0}_{s_2,r_1} & \mathbf{0}_{s_2,r_2} \end{pmatrix}$$

$$\mathsf{Emb}(pe^1) \wedge_{pair} \mathsf{Emb}(pe^2) = \begin{cases} \mathsf{Enc1}((x_1, x_2)) = (c_0, \boldsymbol{c}_1, \boldsymbol{c}_2) \\ \mathsf{Enc2}((y_1, y_2)) = (k_0, \boldsymbol{k}_1, k_2, \boldsymbol{k}_3) \\ \mathsf{Pair}((x_1, x_2), (y_1, y_2)) = E_{(x_1,x_2),(y_1,y_2)} \end{cases}$$

where $\boldsymbol{h} = (\boldsymbol{h}_1, \boldsymbol{h}_2)$ and

$$\begin{aligned} c_0(s_0, \boldsymbol{h}) &= s_0 \\ \boldsymbol{c}_1(s_0, \boldsymbol{h}) &= s_0 \cdot \mathsf{sE}^1_{x_1} \boldsymbol{h}_1 \\ \boldsymbol{c}_2(s_0, \boldsymbol{h}) &= s_0 \cdot \mathsf{sE}^2_{x_2} \boldsymbol{h}_2 \end{aligned} \qquad \begin{aligned} k_0(\alpha, (r_1, r_2, r_3), \boldsymbol{h}) &= r_1 \\ \boldsymbol{k}_1(\alpha, (r_1, r_2, r_3), \boldsymbol{h}) &= r_3 \cdot \mathsf{kE}^1_{y_1} + r_1 \cdot \mathsf{rE}^1_{y_1} \boldsymbol{h}_1 \\ k_2(\alpha, (r_1, r_2, r_3), \boldsymbol{h}) &= r_2 \\ \boldsymbol{k}_3(\alpha, (r_1, r_2, r_3), \boldsymbol{h}) &= (\alpha - r_3) \cdot \mathsf{kE}^2_{y_2} + r_2 \cdot \mathsf{rE}^2_{y_2} \boldsymbol{h}_2 \end{aligned}$$

$$E_{(x_1,x_2),(y_1,y_2)} = \begin{pmatrix} 0 & \mathsf{rD}^{1\top}_{x_1,y_1} & 0 & \mathsf{rD}^{2\top}_{x_2,y_2} \\ \text{-}\mathsf{sD}^1_{x_1,y_1} & \mathbf{0}_{s_1,r_1} & \mathbf{0}_{s_1} & \mathbf{0}_{s_1,r_2} \\ \mathbf{0}_{s_2} & \mathbf{0}_{s_2,r_1} & \text{-}\mathsf{sD}^2_{x_2,y_2} & \mathbf{0}_{s_2,r_2} \end{pmatrix}$$

The resulting pair encodings are different. The first one (result of our conjunction) does not introduce new random variables and does not increase the number of pairings for decryption. On the other hand, the second conjunction adds new random variables to key generation and increases the number of pairings needed during decryption. This overhead will be amplified if nested conjunctions are used.

We now present a comparison for the dual transformation. Let $pe = (\mathsf{sD}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ be a (s, r, w)-predicate encoding. We compare

$$\mathsf{Emb}(\mathsf{Dual}_{pred}(pe)) \quad vs \quad \mathsf{Dual}_{pair}(\mathsf{Emb}(pe))$$

where $\mathsf{Dual}_{pair}(\cdot)$ represents the dual conversion for pair encodings proposed in [30, Section 4], while $\mathsf{Dual}_{pred}(\cdot)$ corresponds to the transformation from our Theorem 9.

$$\mathsf{Emb}(\mathsf{Dual}_{pred}(pe)) = \begin{cases} \mathsf{Enc1}(y) = (c_0, \boldsymbol{c}_1) \\ \mathsf{Enc2}(x) = (k_0, k_1, \boldsymbol{k}_2) \\ \mathsf{Pair}(y, x) = E_{y,x} \end{cases}$$

where $\boldsymbol{h} = (h_0, \boldsymbol{h}_1)$ and

$$\begin{aligned} c_0(s_0, \boldsymbol{h}) &= s_0 \\ \boldsymbol{c}_1(s_0, \boldsymbol{h}) &= h_0 \cdot s_0 \cdot \mathsf{kE}_y + s_0 \mathsf{rE}_y \boldsymbol{h}_1 \end{aligned} \qquad \begin{aligned} k_0(\alpha, r_1, \boldsymbol{h}) &= \alpha + h_0 \\ k_1(\alpha, r_1, \boldsymbol{h}) &= r_1 \\ \boldsymbol{k}_2(\alpha, r_1, \boldsymbol{h}) &= \mathsf{sE}_x \boldsymbol{h}_1 \end{aligned}$$

$$E_{y,x} = \begin{pmatrix} 1 & 0 & \mathsf{sD}_{x,y}^\top \\ \mathbf{0}_r & \text{-}\mathsf{rD}_{x,y} & \mathbf{0}_{r,s} \end{pmatrix}$$

$$\mathsf{Emb}(\mathsf{Dual}_{pair}(pe)) = \begin{cases} \mathsf{Enc1}(y) = (c_0, c_1, \boldsymbol{c}_2) \\ \mathsf{Enc2}(x) = (k_0, k_1, \boldsymbol{k}_2) \\ \mathsf{Pair}(y, x) = E_{y,x} \end{cases}$$

where $\boldsymbol{h} = (h_0, \boldsymbol{h}_1)$ and

$$\begin{aligned} c_0((s_0, s_1), \boldsymbol{h}) &= s_0 \\ c_1((s_0, s_1), \boldsymbol{h}) &= s_1 \\ \boldsymbol{c}_2((s_0, s_1), \boldsymbol{h}) &= h_0 \cdot s_0 \cdot \mathsf{kE}_y + s_1 \mathsf{rE}_y \boldsymbol{h}_1 \end{aligned} \qquad \begin{aligned} k_0(\alpha, r_1, \boldsymbol{h}) &= \alpha + h_0 \cdot r_1 \\ k_1(\alpha, r_1, \boldsymbol{h}) &= r_1 \\ \boldsymbol{k}_2(\alpha, r_1, \boldsymbol{h}) &= r_1 \cdot \mathsf{sE}_x \boldsymbol{h}_1 \end{aligned}$$

$$E_{y,x} = \begin{pmatrix} 1 & 0 & \mathbf{0}_s^\top \\ 0 & 0 & \mathsf{sD}_{x,y}^\top \\ \mathbf{0}_r & \text{-}\mathsf{rD}_{x,y} & \mathbf{0}_{r,s} \end{pmatrix}$$

4.6 Constructions

We provide new instances of predicate encodings to achieve predicate encryption schemes with new properties or better performance.

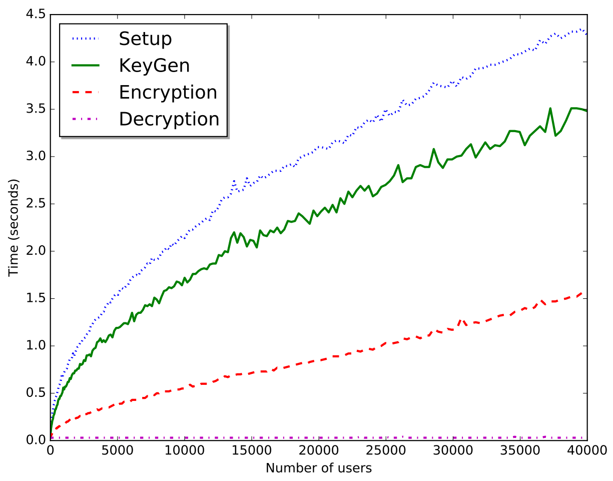


Figure 4.1: Scalability of the PE for revocation.

4.6.1 Combining predicates

4.6.1.1 Dual-Policy ABE

Dual-Policy Attribute Based Encryption [28, 30] has already been considered in the *pair encodings* framework. It combines KP-ABE and CP-ABE into a single construction that simultaneously allows two access control mechanisms. The main advantage is the possibility of considering policies over *objective* attributes (associated to data) and policies over *subjective* attributes (associated to user credentials) at the same time.

Our combinations of predicate encodings allow us to create predicate encryption constructions for Dual-Policy ABE in the framework of pair encodings and tag-based encodings. In particular, given an arbitrary predicate encoding for $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, applying Theorems 9 and 7 we get an encoding for DP-ABE, i.e., for the predicate $\mathsf{P}^\star : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{Y} \times \mathcal{X}) \to \{0,1\}$ defined as

$$\mathsf{P}^\star((x,y),(y',x')) = 1 \text{ iff } \mathsf{P}(x,y) \wedge \mathsf{P}(y',x')$$

4.6.1.2 Revocation

Another application of our combinations is predicate encryption with revocation, by combining a *boolean formula predicate encoding* with a *broadcast encryption predicate encoding*. The former is used to encode the actual policy of the scheme, while the latter takes care of revocation.

Broadcast encryption has been considered in the literature to approach revocation [110, 134, 152]. In broadcast encryption, a broadcasting authority encrypts a message in such a way that only authorized users will be able to decrypt

P	$\bar{\mathsf{P}}$
attributes $= \{a, b, c, d\}$	attributes $= \{\bar{a}, \bar{b}, \bar{c}, \bar{d}\}$
$x = (a \wedge c) \vee d$	$x = (\bar{a} \vee \bar{c}) \wedge \bar{d}$
$y = \{a, c\}$	$y = \{\bar{b}, \bar{d}\}$
$\mathsf{P}(x, y) = 1$ iff $x(y)$	$\bar{\mathsf{P}}(x, y) = 1$ iff $\neg x(y)$

Figure 4.2: Equivalent encodings of a policy using P (CP-ABE) on the left and $\bar{\mathsf{P}}$ (negated CP-ABE) on the right.

it. This can be expressed with the predicate $\mathsf{P}(\boldsymbol{x}, i) = 1$ if, and only if, $\boldsymbol{x}_i = 1$, where $\boldsymbol{x} \in \mathcal{X} = \{0,1\}^n$ and $i \in \mathcal{Y} = [n]$. A drawback is that the number of users in the system, n, is polynomial size. Figure 4.1 shows the performance of predicate encryption built from a predicate encoding that combines boolean formulas with broadcast encryption. The system supports thousands of users in reasonable time.

4.6.2 Improved predicate encodings

In this section we propose new predicate encodings that are more efficient than some of the encodings proposed previously in [76] and implement a general library[4] for predicate encryption with support for predicate encodings and pair encodings to evaluate their performance.

Our encodings are built by applying Theorem 8 to obtain negated encodings and applying Theorem 5 to simplify the negated versions into (some times) more efficient encodings. However, the predicate associated to these new encodings is negated, but if inputs are also negated, the predicate will be equivalent. Figure 4.2 illustrates this idea. On the left, there is a boolean formula CP-ABE for 4 attributes $\{a, b, c, d\}$. On the right side, secret keys and policies are modified so that the negated version is equivalent. The attribute universe is formed by the *negated attributes*, secret keys are formed by all *negated attributes* do not appear in the original key as normal attributes, policies are negated and expressed in Negation Normal Form (NNF).

4.6.2.1 Boolean formulas

In [76], the authors propose two predicate encoding (KP and CP versions) for monotonic boolean formulas. The predicate they consider is a particular case of a Linear Secret Sharing scheme [137]. Let $\mathcal{X} = \{0,1\}^n, \mathcal{Y} = \mathbb{Z}_p^{n \times k}$ for some $n, k \in \mathbb{N}$,

$$\mathsf{P}(\boldsymbol{x}, M) = 1 \text{ iff } \begin{pmatrix} 1 & 0 & \overset{k\text{-}1}{\cdots} & 0 \end{pmatrix} \in \underset{span}{\overset{row}{}} \langle M_{\boldsymbol{x}} \rangle$$

[4]Source code available at https://github.com/miguel-ambrona/abe-relic.

where $M_{\boldsymbol{x}}$ denotes the matrix M filtered by $\boldsymbol{x}$, i.e., $M_{\boldsymbol{x}}$ includes the i-th row of M iff $\boldsymbol{x}_i = 1$.

It has been shown [146] that for every[5] monotonic boolean formula f with attributes from $\mathcal{X}$ there exists a matrix $M \in \mathcal{Y}$ such that for every $\boldsymbol{x} \in \mathcal{X}$, $f(\boldsymbol{x}) \Leftrightarrow \mathsf{P}(\boldsymbol{x}, M)$. The key-policy predicate encoding from [76] is the following,

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & \mathbf{0}_{n,k-1} \end{pmatrix} \quad \mathsf{rE}_M = \begin{pmatrix} I_n & M_{\{2,\dots,k\}} \end{pmatrix} \quad \mathsf{kE}_M = \begin{pmatrix} M_{\{1\}} \end{pmatrix}$$

where $M_{\{1\}}$ denotes the first column of matrix M, $M_{\{2,\dots,k\}}$ represents the rest of the matrix. We do not include explicit decryption functions sD and rD, but they can be computed efficiently by *Gaussian elimination*.

In the above encoding, the number of elements in secret keys and ciphertexts is always maximal, it equals the number of (possibly duplicated) attributes, even for small policies. Furthermore, the maximum number of *and-gates* in a policy must be fixed a priori (because it is related the the number of columns in the matrix).

We propose the following improved predicate encoding for (negated) key-policy monotonic boolean formulas[6], which is an equivalent predicate if instantiated with negated inputs. Let $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \mathbb{Z}_p^{n \times k}$,

$$\mathsf{sE}_{\boldsymbol{x}} = I_n - \mathsf{diag}(\boldsymbol{x}) \qquad \mathsf{rE}_M = M^\top \qquad \mathsf{kE}_M = \begin{pmatrix} 1 & 0 & \overset{k-1}{\cdots} & 0 \end{pmatrix}^\top$$

In our encoding, the number of columns has been reduced up to half[7]. Furthermore, the size of secret keys is proportional to the complexity of policies. In particular, it is equal to the number of *and-gates* in the policy (or equivalently, the number of *or-gates* in the non-negated version). Note that our improvement also works in the ciphertext-policy case.

In Figure 4.3 we present a comparison between our improved encoding for *key-policy monotonic boolean formulas* and the original one. To this end, we generate random boolean formulas for different sizes, starting from a random set of leaf nodes and combining them with boolean operators $\vee$ and $\wedge$. Our tables report on the average time for each algorithm. Our encoding needs 50% less time than the original algorithms for setup, encryption and key generation. For decryption the performance is similar. All the analyzed schemes were instantiated with the same compiler, therefore all achieve the same level of security (under SXDH assumption). In terms of secret key size, our encoding is smaller in general (in the worst case, when all the gates in the policy are *or-gates*, key sizes are equal).

[5] Where every attribute appears at most once and the number of *and-gates* is lower than k (one could overcome the one-use restriction by considering duplicated attributes).

[6] See Section 4.8 for more details about how we obtained this encoding.

[7] Being half when the bound on the number of *and-gates* is maximal.

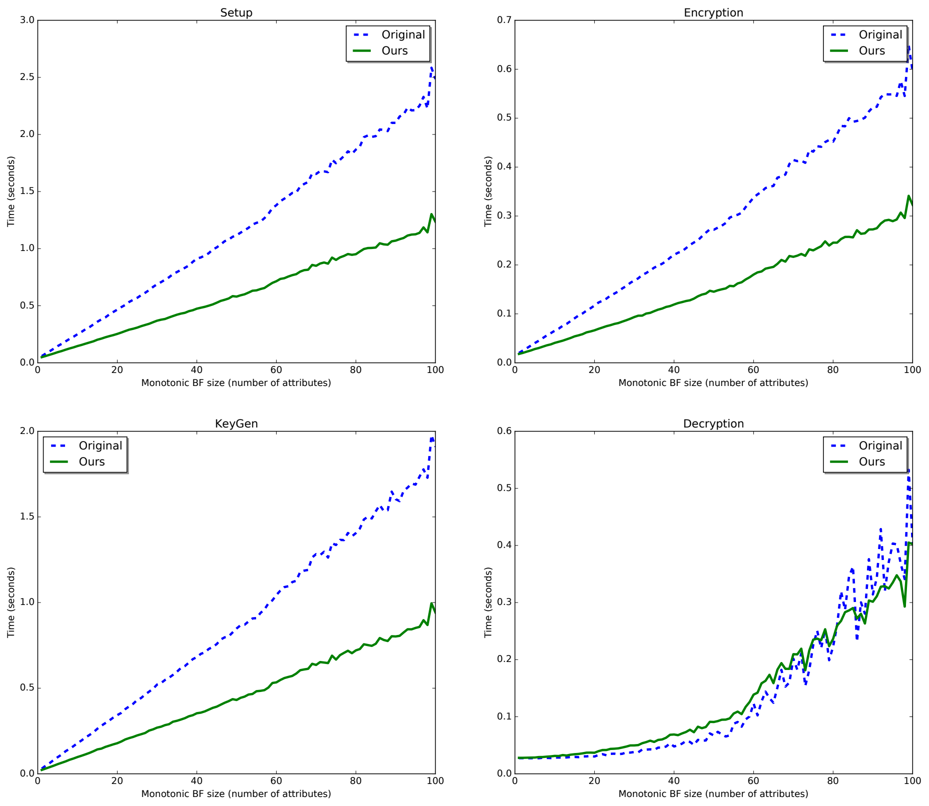


Figure 4.3: Improved predicate encoding for boolean formulas vs original encoding.

4.6.2.2 Arithmetic span programs

Chen et al. proposed in [76] a predicate encoding for *Arithmetic Span Programs (ASP)*. That is, an encoding for the predicate P defined as follows. Let $\mathcal{X} = \mathbb{Z}_p^n$, $\mathcal{Y} = \mathbb{Z}_p^{n\times k} \times \mathbb{Z}_p^{n\times k}$, for some $n, k \in \mathbb{N}$; for every $\boldsymbol{x} \in \mathcal{X}$ and every $(Y, Z) \in \mathcal{Y}$,

$$\mathsf{P}(\boldsymbol{x}, (Y, Z)) = 1 \text{ iff } \begin{pmatrix} 1 & 0 & \overset{k\text{-}1}{\cdots} & 0 \end{pmatrix} \in \substack{\mathit{row}\\ \mathit{span}} \langle \mathsf{diag}(\boldsymbol{x})Y + Z \rangle$$

In [130], Ishai and Wee show how to relate *Arithmetic Span Programs* computations of polynomial functions over a finite field $\mathbb{F}$, i.e., functions $f : \mathbb{F}^n \to \mathbb{F}$ that only use addition and multiplication over the field. Therefore, the above predicate can be seen as $f(\boldsymbol{x}) = 0$, where f is the polynomial function induced by (Y, Z). Let $\mathcal{X} = \mathbb{Z}_p^n$, $\mathcal{Y} = \mathbb{Z}_p^{n\times k} \times \mathbb{Z}_p^{n\times k}$, the original predicate encoding for arithmetic span programs proposed in [76] is the following:

$$\mathsf{sE}_x = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & I_n & \mathbf{0}_{n,k-1} \end{pmatrix}$$

$$\mathsf{rE}_{(Y,Z)} = \begin{pmatrix} I_n & \mathbf{0}_{n,n} & Y_{\{2,\dots,l\}} \\ \mathbf{0}_{n,n} & I_n & Z_{\{2,\dots,l\}} \end{pmatrix} \quad \mathsf{kE}_{(Y,Z)} = \begin{pmatrix} Y_{\{1\}} \\ Z_{\{1\}} \end{pmatrix}$$

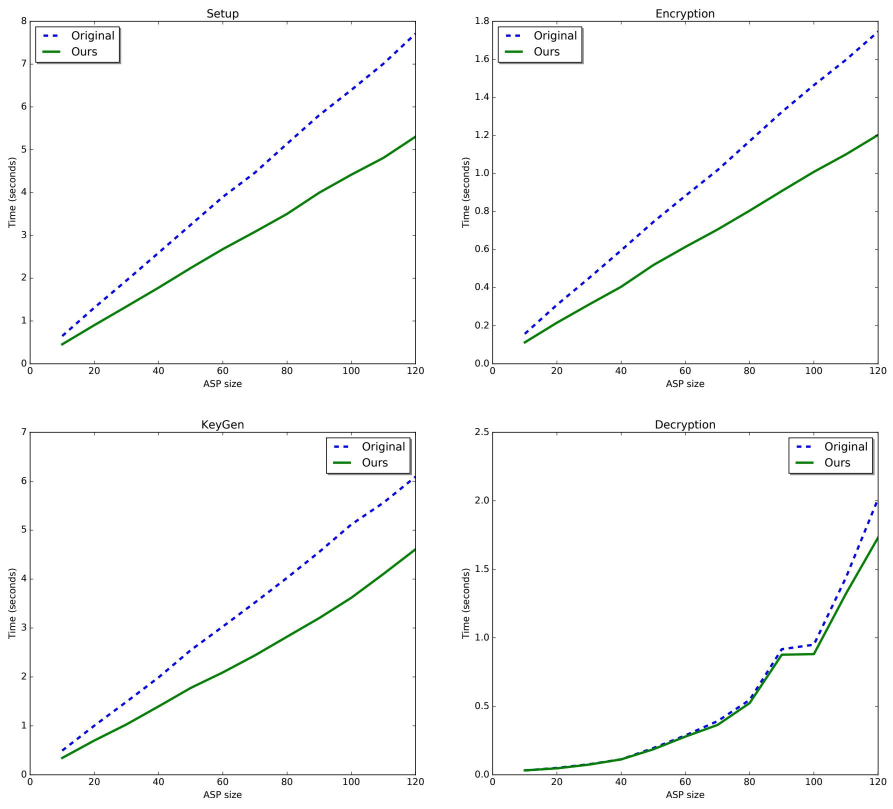


Figure 4.4: Improved predicate encoding for ASP vs original encoding.

We present a more efficient encoding for (negated[8]) arithmetic span programs:

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & \text{-}I_n \end{pmatrix} \quad \mathsf{rE}_{(Y,Z)} = \begin{pmatrix} Z^\top & Y^\top \end{pmatrix} \quad \mathsf{kE}_{(Y,Z)} = \begin{pmatrix} 1 & 0 & \overset{k\text{-}1}{\cdots} & 0 \end{pmatrix}^\top$$

Figure 4.4 shows the performance of our new encoding for KP-ABE for Arithmetic Span Programs compared to the original encoding from [76]. As we expected, our encoding needs 66% of the time required for the original encoding for setup, encryption and key generation. Additionally, secret key size is halved with our encoding.

4.6.3 Extra features

In this section we consider new theoretical results that can be proved thanks to our algebraic characterization of α-privacy and can be used to produce new predicate encodings enhanced with extra properties.

[8]In [130] there is a modification of their algorithm that produces matrices (Y, Z) such that the predicate associated is $f(\boldsymbol{x}) \neq 0$ (the double negation will cancel out).

4.6.3.1 Attribute-hiding for boolean formulas

Chen et al. proposed an extension of the compiler in [76] to build weakly attribute-hiding predicate encryption schemes [66, 138]. In a weakly attribute-hiding scheme, the ciphertext attribute x remains secret for unauthorized users, that only learn the fact that their secret keys are not valid. This additional compiler needs to be instantiated with *predicate encodings* satisfying additional properties. The following is a definition from [76].

Definition 30 (Attribute-Hiding Encodings)**.** *A* (s, r, w)*-predicate encoding,* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *for* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ *is* attribute-hiding *if it verifies the additional requirements:*

x-oblivious reconstruction: $\mathsf{sD}_{x,y}$ *and* $\mathsf{rD}_{x,y}$ *are independent of* x*.*

attribute-hiding: *for all* $(x, y) \notin \mathsf{P}$*,*

$$\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (\mathsf{sE}_x\boldsymbol{w},\ \mathsf{rE}_y\boldsymbol{w}) \quad \equiv \quad \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^s;\ \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^r;\ \textsf{return}\ (\boldsymbol{s},\ \boldsymbol{r})$$

where $\equiv$ *denotes equality of distributions.*

The following theorem relates the second property with our alternative definition of predicate encodings:

Theorem 13 (Algebraic characterization of attribute-hiding)**.** *Let* $p \in \mathbb{N}$ *be a prime, let* $s, r, w \in \mathbb{N}$ *and let* $S \in \mathbb{Z}_p^{s \times w}$*,* $R \in \mathbb{Z}_p^{r \times w}$*,* $\boldsymbol{k} \in \mathbb{Z}_p^r$*. The following are equivalent:*

- $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^w;\ \textsf{return}\ (S\boldsymbol{w},\ R\boldsymbol{w}) \ \equiv\ \boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^s;\ \boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^r;\ \textsf{return}\ (\boldsymbol{s},\ \boldsymbol{r})$
- $\operatorname{rank}\begin{pmatrix} S \\ R \end{pmatrix} = s + r$

Proof. Given $(\boldsymbol{s}, \boldsymbol{r}) \in \mathbb{Z}_p^s \times \mathbb{Z}_p^r$, we define $\Gamma_{\boldsymbol{s},\boldsymbol{r}} = \{\boldsymbol{w} \in \mathbb{Z}_p^w : S\boldsymbol{w} = \boldsymbol{s} \ \wedge\ R\boldsymbol{w} = \boldsymbol{r}\}$. The condition on the second bullet holds iff $w - s - r \geqslant 0$ and the cardinality of $\Gamma_{\boldsymbol{s},\boldsymbol{r}}$ is p^{w-s-r}. Additionally, $|\Gamma_{\boldsymbol{s},\boldsymbol{r}}|$ is independent from $\boldsymbol{r}$ and $\boldsymbol{s}$ iff the two distributions from the first bullet are identical. $\square$

Note that for every (s, r, w)-predicate encoding $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ that is attribute-hiding, there exists an equivalent $(s, 1, w)$-predicate encoding. This is because rD is independent from x and thus, we can apply our optimization Theorem 5 with matrices $\mathsf{B}_y = \mathsf{rD}_{x,y}^\top \in \mathbb{Z}_p^{1 \times w}$, $\mathsf{A}_x = I_s$, $\mathsf{C} = I_w$. Therefore, the class of predicates that can be built from attribute-hiding encodings is included in the class of predicates achieved from $(s, 1, w)$-predicate encodings.

Further, note that our *disjunction* and *conjunction* combinations for predicate encodings (Theorems 6 and 7 respectively) preserve the notion of attribute-hiding[9]. Exploiting this fact, we propose a Policy-Hiding ABE scheme for non-monotonic boolean formulas expressed in DNF (Disjunctive Normal Form). The inner product can be used to encode conjunctions [138]. More concretely, let $\boldsymbol{y} \in \{0,1\}^n \subseteq \mathbb{Z}_p^n$. We establish that the i-th attribute a_i appears in a secret key for $\boldsymbol{y}$ iff $\boldsymbol{y}_i = 1$. Let $S, \bar{S} \subseteq \{a_i\}_{i=1}^n$ be sets such that $S \cap \bar{S} = \varnothing$,

$$\bigwedge_{a \in S} a \ \wedge \ \bigwedge_{a \in \bar{S}} \bar{a} \ \Leftrightarrow^{10} \ \boldsymbol{x}^\top \boldsymbol{y} = |S| \quad \text{where } \forall i \in [n], \ \boldsymbol{x}_i = \begin{cases} 1 & \text{if } a_i \in S \\ \text{-1} & \text{if } a_i \in \bar{S} \\ 0 & \text{otherwise} \end{cases}$$

Note that the ZIPE predicate encoding from [76, Appendix A.1] can be modified into an attribute-hiding encoding for the predicate $\mathsf{P}((\boldsymbol{x}, \gamma), \boldsymbol{y}) = 1$ iff $\boldsymbol{x}^\top \boldsymbol{y} = \gamma$ (see Section 4.9.1).

Therefore, with a disjunction of k predicate encodings like the former we can encode boolean formulas that have at most k disjuncts. Note that the resulting encoding is *attribute-hiding* but it is not *x-oblivious*. However, without the knowledge of the policy x, one can guess for the disjunct his secret key satisfies (if any). In this way, a valid key will be enough to decrypt after at most k decryption tries (one for every disjunct).

4.6.3.2 Delegation

Delegation of keys is a desirable property for every predicate encryption scheme. Roughly, it allows the owner of a secret key to weaken his key creating a new one that is less powerful than the original one. This property can be used to achieve *forward secrecy* (see [72] for an application to ABE that supports delegation), where past sessions are protected from the compromise of future secret keys. To achieve this goal, users can periodically weaken their secret keys and destroy the more powerful ones. In this way, past ciphertexts cannot be decrypted any more and thus, they are protected against the theft of future keys. Delegation is also required for Hierarchical Identity Based Encryption (HIBE). More formally, we say that a predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ supports delegation if there is a partial ordering $\preceq$ on $\mathcal{Y}$ such that for every $x \in \mathcal{X}$, if $\mathsf{P}(x, y) = 1$ and $(y \preceq y')$, then $\mathsf{P}(x, y') = 1$.

Delegation has been considered in [76] as the property of some predicate encodings. We propose a generic method to convert any predicate encoding into one supporting delegation.

[9]Conjunction also preserves x-oblivious reconstruction, while disjunction does not.
[10]This equivalence holds when $|S| < p$, but in practice p is a large prime.

Let $\mathcal{U} = \{a, b, c\}$ be the set of attributes. We consider the predicate encoding for monotonic boolean formulas from [76]. Let $\mathcal{X} = \{0,1\}^3, \mathcal{Y} = \mathbb{Z}_p^{3\times 2}$,

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & \mathbf{0}_{3,2} \end{pmatrix} \qquad \mathsf{rE}_M = \begin{pmatrix} I_3 & M_{\{2\}} \end{pmatrix} \qquad \mathsf{kE}_M = \begin{pmatrix} M_{\{1\}} \end{pmatrix}$$

The following is the encoding of a key for the formula $(a \vee c) \wedge b$, enhanced for delegation according to Theorem 14 (with $k = 1$),

$$\mathsf{rE}_M = \left(\begin{array}{cccc|c} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right) \qquad \mathsf{kE}_M = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Let's assume we want to weaken this key to one for the formula $a \wedge b \wedge c$. Note that in this case we want to make an update of the matrix M:

$$M = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \text{ encodes } (a \vee c) \wedge b \qquad M' = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \text{ encodes } a \wedge b \wedge c$$

It can be done by multiplying rE_M from the left by A

$$\mathsf{rE}'_M = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathsf{rE}_M} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathsf{kE}'_M = A \cdot \mathsf{kE}_M = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Figure 4.5: Example of delegation of keys for monotonic boolean formulas. Since A is a linear function, it can be computed in the exponent from the given key.

Theorem 14 (Delegation). *For every (s, r, w)-predicate encoding* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *for* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*, for every $k \in \mathbb{N}$,* $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}', \mathsf{sD}', \mathsf{rD}')$ *defined below is a valid $(s, r+k, w+k)$-predicate encoding for* P*.*

$$\mathsf{sE}'_x = \begin{pmatrix} \mathsf{sE}_x & \mathbf{0}_{s,k} \end{pmatrix} \qquad \mathsf{rE}'_y = \begin{pmatrix} \mathsf{rE}_y & \mathbf{0}_{r,k} \\ \mathbf{0}_{k,w} & I_k \end{pmatrix} \qquad \mathsf{kE}'_y = \begin{pmatrix} \mathsf{kE}_y \\ \mathbf{0}_k \end{pmatrix}$$

$$\mathsf{sD}'_{x,y} = \mathsf{sD}_{x,y} \qquad \mathsf{rD}'_{x,y} = \begin{pmatrix} \mathsf{rD}_{x,y} \\ \mathbf{0}_k \end{pmatrix}$$

Proof. Correctness can be easily checked. For privacy, let $(x, y) \notin \mathsf{P}$ and let $\boldsymbol{w} \in \mathbb{Z}_p^w$ be such that $\mathsf{sE}_x\boldsymbol{w} = \mathbf{0}_s$ and $\mathsf{rE}_y\boldsymbol{w} = \mathsf{kE}_y$. Note that $\boldsymbol{w}'^\top = \begin{pmatrix} \boldsymbol{w}^\top & \mathbf{0}_k^\top \end{pmatrix}$ is a witness of privacy for $(\mathsf{sE}', \mathsf{rE}', \mathsf{kE}', \mathsf{sD}', \mathsf{rD}')$. $\square$

The additional set of not-null rows in rE'_y can be used to weaken the linear span of rE_y, what directly modifies the predicate. In particular, this method works really well for monotonic boolean formulas (see Figure 4.5 for an example).

4.7 Flexible boolean combinations

Boolean combinations of predicate encodings can be applied dynamically, i.e., they can be combined by leaving placeholders $\mathsf{P} \bowtie \mathsf{P}'$ that will be chosen during encryption or key generation. The following theorem shows how to dynamically combine two predicates making the combinator part of the secret key sk_y.

Theorem 15 (Flexible combination of predicate encodings)**.** *For every* (s_1, r_1, w_1)*-predicate encoding* $(\mathsf{sE}^1, \mathsf{rE}^1, \mathsf{kE}^1, \mathsf{sD}^1, \mathsf{rD}^1)$ *for* $\mathsf{P}_1 : \mathcal{X}_1 \times \mathcal{Y}_1 \to \{0,1\}$ *and every* (s_2, r_2, w_2)*-predicate encoding* $(\mathsf{sE}^2, \mathsf{rE}^2, \mathsf{kE}^2, \mathsf{sD}^2, \mathsf{rD}^2)$ *for* $\mathsf{P}_2 : \mathcal{X}_2 \times \mathcal{Y}_2 \to \{0,1\}$*, there exists a* $(s_1+s_2,\ r_1+r_2,\ w_1+w_2+1)$*-predicate encoding* $(\mathsf{sE}, \mathsf{rE}, \mathsf{kE}, \mathsf{sD}, \mathsf{rD})$ *for the predicate* $\mathsf{P} : (\mathcal{X}_1, \mathcal{X}_2) \times (\mathcal{Y}_1, \mathcal{Y}_2, \{\vee, \wedge\}) \to \{0,1\}$ *such that:*

$$\mathsf{P}((x_1,x_2),(y_1,y_2,\bowtie)) \Leftrightarrow \mathsf{P}_1(x_1,y_1) \bowtie \mathsf{P}_2(x_2,y_2)$$

Concretely,

$$\mathsf{sE}_{(x_1,x_2)} = \begin{pmatrix} \mathsf{sE}^1_{x_1} & \mathbf{0}_{s_1,w_2} & \mathbf{0}_{s_1} \\ \mathbf{0}_{s_2,w_1} & \mathsf{sE}^2_{x_2} & \mathbf{0}_{s_2} \end{pmatrix} \qquad \mathsf{sD}_{(x_1,x_2),(y_1,y_2,\bowtie)} = \frac{1}{2}\begin{pmatrix} \mathsf{sD}^1_{x_1,y_1} \\ \mathsf{sD}^2_{x_2,y_2} \end{pmatrix}$$

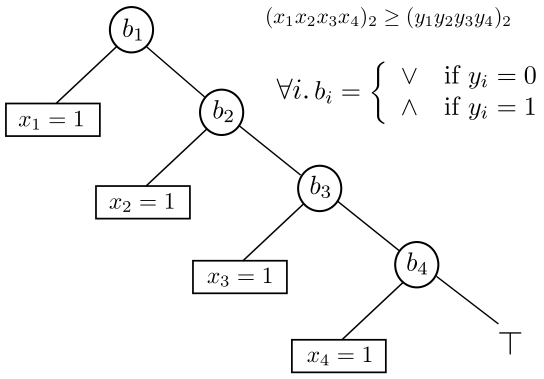
$$\mathsf{rE}_{(y_1,y_2,\bowtie)} = \begin{pmatrix} \mathsf{rE}^1_{y_1} & \mathbf{0}_{r_1,w_2} & f_{\bowtie} \cdot \mathsf{kE}^1_{y_1} \\ \mathbf{0}_{r_2,w_1} & \mathsf{rE}^2_{y_2} & -f_{\bowtie} \cdot \mathsf{kE}^2_{y_2} \end{pmatrix} \qquad \mathsf{rD}_{(x_1,x_2),(y_1,y_2,\bowtie)} = \frac{1}{2}\begin{pmatrix} \mathsf{rD}^1_{x_1,y_1} \\ \mathsf{rD}^2_{x_2,y_2} \end{pmatrix}$$

$$\mathsf{kE}_{(y_1,y_2,\bowtie)} = \begin{pmatrix} \mathsf{kE}^1_{y_1} \\ \mathsf{kE}^2_{y_2} \end{pmatrix}$$

where $f_{\bowtie}$ *is defined as* 1 *if* $\bowtie = \wedge$ *and* 0 *if* $\bowtie = \vee$*.*

Proof. This Theorem follows straightforwardly from Theorems 6 and 7. □

Note that our Theorem 9 gives us an equivalent version of the above theorem, where the placeholder is part of the ciphertext ct_x. Figure 4.6 presents a possible application of a flexible fixed-structure combination of boolean operators. It encodes the predicate $\mathsf{P}(x,y) = 1$ iff $x \geqslant y$, where $\mathcal{X} = \mathcal{Y} = \{0,1\}^4$ (4-bit strings). Note that the leaf nodes are IBE predicate encodings (one of the simplest predicate encodings).



$(x_1x_2x_3x_4)_2 \geq (y_1y_2y_3y_4)_2$
$\forall i.\, b_i = \begin{cases} \vee & \text{if } y_i = 0 \\ \wedge & \text{if } y_i = 1 \end{cases}$
b_1
b_2
b_3
b_4
$x_1 = 1$
$x_2 = 1$
$x_3 = 1$
$x_4 = 1$
$\top$

Figure 4.6: Example of fixed structure for inequalities.

4.8 Obtaining more efficient encodings

We recall the encoding from [76] for KP-monotonic boolean formulas (mentioned in out Section 4.6.2.1). Let $\mathcal{X} = \{0,1\}^n, \mathcal{Y} = \mathbb{Z}_p^{n\times k}$ for some $n, k \in \mathbb{N}$,

$$\mathsf{P}(\boldsymbol{x}, M) = 1 \text{ iff } \begin{pmatrix} 1 & 0 & \overset{k\text{-}1}{\cdots} & 0 \end{pmatrix} \in \underset{span}{\overset{row}{}} \langle M_{\boldsymbol{x}} \rangle$$

where $M_{\boldsymbol{x}}$ denotes the matrix M filtered by $\boldsymbol{x}$, i.e., $M_{\boldsymbol{x}}$ includes the i-th row of M iff $\boldsymbol{x}_i = 1$.

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & \mathbf{0}_{n,k-1} \end{pmatrix} \quad \mathsf{rE}_M = \begin{pmatrix} I_n & M_{\{2,\dots,k\}} \end{pmatrix} \quad \mathsf{kE}_M = \begin{pmatrix} M_{\{1\}} \end{pmatrix}$$

where $M_{\{1\}}$ denotes the first column of matrix M, $M_{\{2,\dots,k\}}$ represents the rest of the matrix. We can directly apply our negation transformation from Theorem 8 and get the following encoding for (negated) monotonic boolean formulas:

$$\mathsf{sE}'_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x}) & \text{-}I_n & \mathbf{0}_{n,k-1} & \mathbf{0}_{n,n} \\ \mathbf{0}_{k-1,n} & \mathbf{0}_{k-1,n} & \text{-}I_{k-1} & \mathbf{0}_{k-1,n} \end{pmatrix}$$

$$\mathsf{rE}'_M = \begin{pmatrix} \mathbf{0}_{n,n} & I_n & \mathbf{0}_{n,k-1} & I_n \\ \mathbf{0}_{k-1,n} & \mathbf{0}_{k-1,n} & I_{k-1} & (M_{\{2,\dots,k\}})^\top \\ \mathbf{0}_n^\top & \mathbf{0}_n^\top & \mathbf{0}_{k-1}^\top & (M_{\{1\}})^\top \end{pmatrix} \quad \mathsf{kE}'_M = \begin{pmatrix} \mathbf{0}_n \\ \mathbf{0}_{k-1} \\ 1 \end{pmatrix}$$

We will simplify the above encoding by using our Theorem 5. We choose the identity matrix for $\mathsf{A}_{\boldsymbol{x}}$ and B_M and certain matrix C that satisfies the conditions of the theorem, i.e., for every pair[11] $(x, y) \notin \bar{\mathsf{P}}$, the column span of C contains at least one witness $\boldsymbol{w}$ such that $\mathsf{sE}'_{\boldsymbol{x}}\boldsymbol{w} = \mathbf{0}_{n+k-1}$ and $\mathsf{rE}'_M\boldsymbol{w} = \mathsf{kE}'_M$. Let

[11]We denote by $\bar{\mathsf{P}}$ the negated predicate of P, i.e., $(\boldsymbol{x}, M) \in \bar{\mathsf{P}} \Leftrightarrow (\boldsymbol{x}, M) \notin \mathsf{P}$.

$(\boldsymbol{x}, M) \notin \bar{\mathsf{P}}$. Let $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4) \in \mathbb{Z}_p^n \times \mathbb{Z}_p^n \times \mathbb{Z}_p^{k-1} \times \mathbb{Z}_p^n$. For $\boldsymbol{w}$ to be a valid witness, it needs to verify

$$\begin{cases} \mathsf{diag}(\boldsymbol{x})\boldsymbol{w}_1 - \boldsymbol{w}_2 = \mathbf{0}_{n+k-1} \\ -\boldsymbol{w}_3 = \mathbf{0}_{k-1} \\ \boldsymbol{w}_2 + \boldsymbol{w}_4 = \mathbf{0}_n \\ \boldsymbol{w}_3 + (M_{\{2,\ldots,k\}})^\top \boldsymbol{w}_4 = \mathbf{0}_{k-1} \\ (M_{\{1\}})^\top \boldsymbol{w}_4 = 1 \end{cases}$$

Since $(\boldsymbol{x}, M) \notin \bar{\mathsf{P}}$, there exist $\boldsymbol{a} \in \mathbb{Z}_p^n$ such that $\boldsymbol{a}^\top \mathsf{diag}(\boldsymbol{x}) M = \begin{pmatrix} 1 & 0 & \overset{k-1}{\ldots} & 0 \end{pmatrix}$. Note that if we choose $\boldsymbol{w}_1 = \boldsymbol{w}_2 = \mathsf{diag}(\boldsymbol{x})\boldsymbol{a}$, $\boldsymbol{w}_3 = \mathbf{0}_{k-1}$, $\boldsymbol{w}_4 = -\mathsf{diag}(\boldsymbol{x})\boldsymbol{a}$, then $\boldsymbol{w}$ is a valid witness, since all the equations are satisfied[12]. This implies that the matrix

$$\mathsf{C} = \begin{pmatrix} I_n \\ I_n \\ \mathbf{0}_{k-1,n} \\ -I_n \end{pmatrix}$$

contains the valid witness $\boldsymbol{w}$ described above and can be used to simplify the encoding into $\mathsf{sE}''_{\boldsymbol{x}} = \mathsf{sE}'_{\boldsymbol{x}}\mathsf{C}$, $\mathsf{rE}''_M = \mathsf{rE}'_M\mathsf{C}$, $\mathsf{kE}''_M = \mathsf{kE}'_M$. That is,

$$\mathsf{sE}''_{\boldsymbol{x}} = \begin{pmatrix} \mathsf{diag}(\boldsymbol{x})\text{-}I_n \\ \mathbf{0}_{k-1,n} \end{pmatrix} \qquad \mathsf{rE}''_M = \begin{pmatrix} \mathbf{0}_{n,n} \\ -(M_{\{2,\ldots,k\}})^\top \\ -(M_{\{1\}})^\top \end{pmatrix} \qquad \mathsf{kE}''_M = \begin{pmatrix} \mathbf{0}_n \\ \mathbf{0}_{k-1} \\ 1 \end{pmatrix}$$

Now, the matrices

$$\mathsf{A}_{\boldsymbol{x}} = \begin{pmatrix} I_n & \mathbf{0}_{n,n} \end{pmatrix} \qquad\qquad \mathsf{B}_M = \begin{pmatrix} \mathbf{0}_{n,n} & \mathbf{0}_{k-1,k-1} & -1 \\ \mathbf{0}_{n,n} & I_{k-1} & 0 \end{pmatrix}$$

verify the conditions of our Theorem 5 and when applied to the encoding sE'', rE'', kE'' the encoding presented in Section 4.6.2.1 for negated monotonic boolean formulas.

4.9 Concrete encodings

4.9.1 Predicate encoding for γ-inner product (modified from [76])

Let $n \in \mathbb{N}$ and let $\mathcal{X} = \mathbb{Z}_p^n \times \mathbb{Z}_p$, $\mathcal{Y} \in \mathbb{Z}_p^n$. The following is a $(n+1, 1, n+2)$-valid predicate encoding for the predicate $\mathsf{P}((\boldsymbol{x}, \gamma), \boldsymbol{y}) = 1$ iff $\boldsymbol{x}^\top \boldsymbol{y} = \gamma$,

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} \boldsymbol{x} & I_n & \mathbf{0}_n \\ \gamma & \mathbf{0}_n^\top & 1 \end{pmatrix} \qquad \mathsf{rE}_{\boldsymbol{y}} = \begin{pmatrix} 0 & \boldsymbol{y}^\top & -1 \end{pmatrix} \qquad \mathsf{kE}_{\boldsymbol{y}} = \begin{pmatrix} 1 \end{pmatrix}$$

[12]Observe that $\mathsf{diag}(\boldsymbol{x})\mathsf{diag}(\boldsymbol{x}) = \mathsf{diag}(\boldsymbol{x})$ because $\boldsymbol{x} \in \{0,1\}^n$.

$$\mathsf{sD}_{\boldsymbol{x},\boldsymbol{y}}^{\top} = \begin{pmatrix} \boldsymbol{y}^{\top} & \text{-}1 \end{pmatrix} \qquad\qquad \mathsf{rD}_{\boldsymbol{x},\boldsymbol{y}} = \begin{pmatrix} 1 \end{pmatrix}$$

It is an *attribute-hiding predicate encoding.*

4.9.2 Generalized predicate encoding for broadcast encryption

We generalize the predicate encoding for broadcast encryption from [76]. Let $\mathcal{X} = \mathbb{Z}_p^n$, $\mathcal{Y} = [n] \times \mathbb{Z}_p$, we consider the predicate $\mathsf{P}(\boldsymbol{x}, (i, \gamma)) = 1$ iff $\boldsymbol{x}_i = \gamma$. As in [76], it is convenient to express the above predicate as follows:
$\mathcal{X} = (\mathbb{Z}_p^{t_2})^{t_1}$, $\mathcal{Y} = [t_1] \times [t_2] \times \mathbb{Z}_p$ and

$$\mathsf{P}((\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t_1}), (i_1, i_2, \gamma)) = 1 \text{ iff } \boldsymbol{x}_{i_1}^{\top} \boldsymbol{e}_{i_2} = \gamma$$

where $n = t_1 t_2$ and (i_1, i_2) is the unique pair of integers satisfying $i = (i_1\text{-}1) \cdot t_2 + i_2$ and $0 < i_2 \leqslant t_2$. Also, $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{t_2})$ is the standard basis of $\mathbb{Z}_p^{t_2}$. The following is a valid $(t_1, t_2, t_1 + t_2)$-predicate encoding for the above predicate:

$$\mathsf{sE}_{\boldsymbol{x}} = \begin{pmatrix} I_{t_1} & \begin{matrix} \boldsymbol{x}_1^{\top} \\ \vdots \\ \boldsymbol{x}_{t_1}^{\top} \end{matrix} \end{pmatrix} \qquad \mathsf{rE}_{(i_1,i_2,\gamma)} = \begin{pmatrix} \mathbf{0}_{t_2,(i_1-1)} & \boldsymbol{e}_{i_2} & \mathbf{0}_{t_2,(t_1-i_1)} & \gamma \cdot I_{t_2} \end{pmatrix}$$

$$\mathsf{kE}_{(i_1,i_2,\gamma)} = \boldsymbol{e}_{i_2} \qquad \mathsf{sD}_{\boldsymbol{x},(i_1,i_2,\gamma)} = \boldsymbol{e}_{i_1} \qquad \mathsf{rD}_{\boldsymbol{x},(i_1,i_2,\gamma)} = \gamma^{-1} \cdot \boldsymbol{x}_{i_1}$$

This encoding can be used to perform 2-dimensional broadcast encryption. That is, users are divided in n groups and every user i has a unique identifier γ_i. Encryption can be done in such a way that at most one user from every group will be able to decrypt.

4.9.3 Tag-based encoding for root sharing of polynomials

Let $m, n \in \mathbb{N}$ and let $\mathcal{X}, \mathcal{Y} \subset \mathbb{Z}_p[T]$ be the sets of polynomials of degree m and n respectively with coefficients over $\mathbb{Z}_p$. For $f(t) \in \mathcal{X}$ and $g(t) \in \mathcal{Y}$, let P be the predicate defined as $\mathsf{P}(f, g) = 1$ iff $\exists t_0 \in \mathbb{Z}_p : f(t_0) = g(t_0) = 0$. The following is a valid $(n, m, m + n)$-tag-based encoding for P,

$$\mathsf{cE}_f = \begin{pmatrix} a_m & \ldots & a_0 & 0 & \overset{n-1}{\cdots} & 0 \\ 0 & a_m & \ldots & a_0 & 0 \ \overset{n-2}{\cdots} & 0 \\ & & \ddots & & \ddots & \\ 0 & \overset{n-1}{\cdots} & 0 & a_m & \ldots & a_0 \end{pmatrix} \quad \mathsf{kE}_g = \begin{pmatrix} b_n & \ldots & b_0 & 0 & \overset{m-1}{\cdots} & 0 \\ 0 & b_n & \ldots & b_0 & 0 \ \overset{m-2}{\cdots} & 0 \\ & & \ddots & & \ddots & \\ 0 & \overset{m-1}{\cdots} & 0 & b_n & \ldots & b_0 \end{pmatrix}$$

where $f(t) = a_m t^m + \cdots + a_1 t + a_0$ and $g(t) = b_n t^n + \cdots + b_1 t + b_0$.

5 Attribute-Based Encryption in the Generic Group Model

Never trust a computer you can't throw out a window.

Attributed to Steve Wozniak

In this work, we propose, implement, and evaluate automated methods for proving security of ABE in the Generic (Bilinear) Group Model, for analyzing the security of cryptographic assumptions and pairing-based schemes.

Our method is applicable to Rational-Fraction Induced ABE, a large class of ABE that contains most of the schemes from the literature, and relies on a Master Theorem, which reduces security in the GGM to a (new) notion of symbolic security, which is amenable to automated verification using constraint-based techniques. We relate our notion of symbolic security for Rational-Fraction Induced ABE to prior notions for Pair Encodings. Finally, we present several applications, including automated proofs for new schemes.

5.1 Introduction

Our results and tools are specialized to prime order, asymmetric (Type III) bilinear groups, with a pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_t$ are prime order groups. This setting is a natural choice to consider because it supports more efficient and compact implementations.

Attribute-Based-Encryption. Following several recent works [24, 190], we focus on schemes where the ciphertext for x is of the form $[\![c_x(s,b)]\!]_1, [\![sa]\!]_t \cdot M$ and the secret key for y is of the form $[\![k_y(a,b,r)]\!]_2$. For correctness, we require that whenever $\mathsf{P}(x, y) = 1$, there should exist a degree 2 function of $c_x(S, B), k_y(A, B, R)$ that outputs SA, where S, B, A, R are formal variables corresponding to the inputs s, b, a, r of c_x, k_y; the degree 2 function allows us to compute $[\![sa]\!]_\mathsf{t}$ given $[\![c_x(s, b)]\!]_1, [\![k_y(a, b, r)]\!]_2$.

We propose ABE schemes based on encodings c_x, k_y defined in terms of rational fractions of polynomials, which allows us to capture larger classes of schemes. An example is the "*petit IBE*" [191], where $c_x(S, B) = (B + x)S$, $k_y(A, B, R) = A/(B + y)$ and P corresponds to the equality predicate. To prove adaptive security of these ABE in GGM, we require that the ABE satisfy a strengthening of the symbolic security from Agrawal-Chase [11] to the many-key setting, namely that there exists no degree two function of $c_x(S, B), \{k_y(A, B, R) : \mathsf{P}(x, y) = 0\}$ that outputs SA. Looking ahead, note that many-key symbolic security is a purely algebraic criterion, and therefore particularly amenable to analysis using automated tools.

Next, we prove that if we restrict c_x, k_y to polynomials that satisfy some structural requirements as in prior works [11, 26] and that the ABE satisfies the (one-key) symbolic security from [11], then the ABE is adaptively secure in GGM. This means that it suffices for the automated tool to check the one-key symbolic security criterion instead of the many-key variant. We note that a similar result was shown in Agrawal-Chase [11], where they first apply a transformation to the ABE scheme which blows up the ciphertext and key sizes by a factor of 2, and showed that the ensuing ABE is adaptively secure in the standard model; in contrast, we prove adaptive security of the ABE *as is* in GGM. Compared to the latter schemes, our schemes are simpler and twice as efficient in terms of encryption time, decryption time, ciphertext and key sizes, but we only achieve security in the idealized GGM model. We note that all known non-trivial attacks on bilinear groups in use today are captured by GGM. For this reason, we believe that our ABE schemes provide a compelling alternative to less efficient standard model schemes in practical applications where performance is paramount.

Formally, we obtain both results in a unified manner by showing that for ABE captured by restricted polynomials c_x, k_y as in the latter, symbolic security implies many-key symbolic security. We note that a few of the ABE schemes captured by our framework have been informally claimed to be adaptively secure in GGM (e.g. [124, footnote 1 (Chapter 6)]), but to the best of our knowledge, our work provides the first formal treatment of adaptive security in GGM for a broad class of schemes satisfying a simple algebraic criterion.

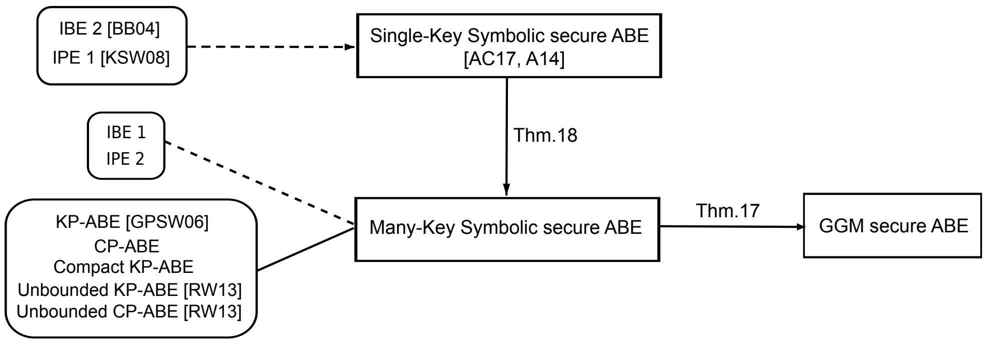


Figure 5.1: Roadmap of our results. The statements marked with dotted arrows were performed fully automatically with our tool (see Section 5.5), while plain arrows denotes proofs by hand. We provide explicit proofs for all our results.

5.1.1 Approach

While we do not advocate proving security in the GGM over the standard model, there are several reasons for our approach.

First, the GGM captures most algebraic attacks, making automated analysis in the GGM desirable for providing cryptographers early feedback during the design of new constructions. Second, the Generic Group Model often admits schemes that are simpler, more efficient, and ultimately more likely to be deployed in real-world systems. Third, existing proofs of adaptive security of ABE in the standard model are very challenging and full automation remains beyond the state-of-the-art, despite recent progress [47]. Finally, GGM proofs are generally considered to be fairly mechanical and sometimes claims are made without proofs, e.g. [124, footnote 1 (Chapter 6)]; this makes GGM proofs a useful target and test-bed for automated proofs.

5.1.2 Related work

Our work builds upon several areas, including ABE, GGM, and computer-aided cryptography.

ABE. Designing adaptively secure and efficient ABE schemes is hard, and has been the focus of many prior works [144, 145, 162, 163, 188]. In 2014, Wee [190] and Attrapadung [24] propose simpler primitives called encoding and cryptographic compilers that turn secure encodings into adaptively secure attribute-based encryption schemes for a broad range of predicates. Their work is initially carried in the composite-order setting; in, Chen, Gay and Wee [76], Agrawal Chase [10], and Attrapadung [26] adapt the compiler to the prime order setting, using the notion of Dual System Groups (DSG) [77, 78]. More recently, Agrawal and Chase [11] propose a notion of symbolic security for pair encodings, and show that every symbolically secure pair encoding is compiled to an attribute-based encryption scheme that achieves full security under a q-type assumption. In Chapter 4 we have provided an algebraic characterization of the information-theoretic notion of α-privacy for predicate encodings. Agrawal and Chase's work and ours leave open the possibility of building fully automated tools for checking symbolic security or the algebraic characterization of privacy.

GGM. The Generic Group Model was introduced in [160, 182] to reason about lower bounds for computing discrete logarithms and related problems. Maurer [156] gives an alternative presentation; while the two presentations are essentially equivalent, Maurer's presentation is more convenient for formalizing the Master Theorem and as a basis for formal verification. The GGM has been used for analyzing a broad variety of assumptions and constructions.

Master Theorems for bilinear groups were introduced by Boneh, Boyen and Goh in [63, 68]. There exist many others instances of Master Theorems; in particular, previous works on automated analyses in the GGM (detailed below) come with their own Master Theorem.

Computer-aided proofs. Barthe, Cederquist and Tarento [41] use the Coq proof assistant for building machine-checked proofs of security in the Generic Group Model. Their formalization is restricted to very simple examples.

Barthe and co-workers [43] develop an automated tool for analyzing security assumptions in the GGM. Their tool is justified by a Master Theorem which reduces security in the Generic Group Model to a weaker notion of symbolic security. However, their Master Theorem and their tool is primarily targeted to analyze assumptions, rather than schemes. A follow-up [45] considers the case of Structure-Preserving Signatures [2, 3, 4, 7, 44, 75] and harnesses the automated analyzer with a synthesis algorithm, which is used to discover new schemes. However, the tool is limited to prove security against a restricted class of adversaries. In Chapter 3 we have extended prior Master Theorems to a more general class of security experiments and provided constraint-solving for proving symbolic security. However, our method in Chapter 3 does not consider rational functions.

Beyond this, prime focus of computer-aided cryptography is to support proofs in the standard model. Prior work uses a highly automated tool called AutoGP for proving security of several IBE in the standard model [47]. However, we are not aware of any prior work that uses computer-aided tools for reasoning about ABE. It could be possible to use existing computer-aided tools such as EasyCrypt [46] for building machine-checked proofs of security of ABE in the standard model; however, it would be very challenging to automate existing proofs for the composite order case, and even more so for the prime order case.

Finally, there have been efforts to integrate formal verification in tool-assisted cryptographic engineering approaches for pairing-based cryptography [14]. There exist some similarities between our constraint-based method for proving symbolic security and the techniques they use. However, the goals of the two methods, and their justification, are fundamentally different.

5.1.3 Notation

We denote by $\mathsf{append}(L, x)$ the act of adding an element x to the list L, and for any $i \in \mathbb{N}$, we denote by $L[i]$ the i'th element of the L if it exists (lists are indexed from index 1 on), or $\perp$ otherwise. For any $s \in \{1, 2, \mathsf{t}\}$, we adopt the convention $[\![\perp]\!]_s = \perp$.

5.2 Rational-Fraction Induced ABE

In this section we define a special class of so-called Rational-Fraction Induced ABE (RFI-ABE), that captures all previous dual system ABE, but also allows inversion in the exponent, thereby capturing ABE's that fall out of the scope of dual system encryption, most notably the IBE from [191], as well as new ABE described in Section 5.4.

We prove the adaptive security of RFI-ABE in the generic group model, where it is assumed that no attack can make use of the algebraic structure of the particular bilinear group that is used. As it is common in the literature, we prove security in two steps. First, we prove a Master Theorem (Theorem 17) that bounds the probability of distinguishing between the *generic* and the *symbolic* models. Second (Lemma 4), we show that the advantage of any adversary in the symbolic model is zero, provided some algebraic condition on the ABE is satisfied (this condition is defined as the symbolic security of the ABE). For the sake of simplicity, our Master Theorem is specialized to capture the security experiment of RFI-ABE, however, it can be generalized to capture more general security games[1].

5.2.1 Rational fractions

Polynomials. Let p be a prime, $n \in \mathbb{N}$. The set of multi-variate polynomials over $\mathbb{Z}_p$ with indeterminates $X_1, \ldots, X_n$ is denoted by $\mathbb{Z}_p[X_1, \ldots, X_n]$. For a polynomial $P \in \mathbb{Z}_p[X]$ and a formal variable Y, we write $P[X \to Y]$ to denote

[1]Note that a more general master theorem could require a looser bound.

the polynomial in $\mathbb{Z}_p[Y]$ where X is replaced by Y. We generalize this notation for multivariate polynomials.

Rational fractions. Let p be a prime, $n \in \mathbb{N}$. A rational fraction is a pair $(f, g) \in \mathbb{Z}_p[X_1, \dots, X_n] \times \mathbb{Z}_p[X_1, \dots, X_n]^*$. We use f/g to denote the rational fraction (f, g) and we use f to denote $(f, 1)$. For any $\boldsymbol{x} \in \mathbb{Z}_p^n$ such that $g(\boldsymbol{x}) = 0$, we denote $f(\boldsymbol{x})/g(\boldsymbol{x}) = \perp$. We define for all $a \in \mathbb{Z}_p$, $\perp + a = a + \perp = \perp$ and $a \cdot \perp = \perp \cdot a = \perp$ and we define the degree of a rational fraction f/g as

$$\mathsf{deg}(f/g) := \max\{\mathsf{deg}(f), \mathsf{deg}(g)\}$$

where $\mathsf{deg}(f)$ and $\mathsf{deg}(g)$ denote the degree of polynomials f and g, respectively.

Equivalence relation. We define an equivalence relation $\sim_{\mathsf{rf}}$ between rational fractions by the clause

$$f_1/g_1 \sim_{\mathsf{rf}} f_2/g_2 \Leftrightarrow f_1 \cdot g_2 = f_2 \cdot g_1$$

where f_1/g_1 and f_2/g_2 are arbitrary rational fractions.

Operators. For any two rational fractions $f_1/g_1, f_2/g_2$ and $\alpha \in \mathbb{Z}_p$, let $\widehat{g} = \mathsf{lcm}(g_1, g_2)$ be the least common multiple of polynomials g_1 and g_2. We define,

- Addition: $f_1/g_2 +_{\mathsf{rf}} f_1/g_2 := (f_1 \cdot \frac{\widehat{g}}{g_1} + f_2 \cdot \frac{\widehat{g}}{g_2})/\widehat{g}$.
- Scalar multiplication: $\alpha \cdot (f/g) := (\alpha \cdot f)/g$.
- Product: $f_1/g_1 \cdot_{\mathsf{rf}} f_2/g_2 := (f_1 \cdot f_2)/(g_1 \cdot g_2)$.

Note that the set of rational fractions equipped with addition, scalar multiplication and product is an algebra over $\mathbb{Z}_p$. In particular, rational fractions verify the associative property with $+_{\mathsf{rf}}$, we write

$$\sum_{i \in [n]}^{\mathsf{rf}} \alpha_i \cdot f_i/g_i := \alpha_1 \cdot f_1/g_1 +_{\mathsf{rf}} \dots +_{\mathsf{rf}} \alpha_n \cdot f_n/g_n$$

for $\alpha_1, \dots, \alpha_n \in \mathbb{Z}_p$, and rational fractions $f_1/g_1, \dots, f_n/g_n$. This is called a linear combination of the rational fractions $f_1/g_1, \dots, f_n/g_n$. For any set of rational fractions Γ, we denote by $\langle \Gamma \rangle$ the set of all linear combinations of rational fractions in Γ.

For any set of formal variables S_1 and S_2, $f_1/g_2 \in \mathbb{Z}_p[S_1]$, and $f_2/g_2 \in \mathbb{Z}_p[S_2]$, we naturally extend the operators $f_1/g_1 +_{\mathsf{rf}} f_2/g_2$ and $f_1/g_1 \cdot_{\mathsf{rf}} f_2/g_2$ to obtain rational fractions in $\mathbb{Z}_p[S_1 \cup S_2]$.

Very roughly, the following Lemma guarantees that a polynomial number of summations of rational fractions, produces a rational fraction whose degree is polynomially large.

Lemma 3. *For any two rational fractions f_1/g_1, f_2/g_2, it holds*

$$\mathsf{deg}(f_1/g_1 +_{\mathsf{rf}} f_2/g_2) \leqslant \mathsf{deg}(f_1/g_1) + \mathsf{deg}(f_2/g_2)\ .$$

Proof. Let $\widetilde{g}_1$ and $\widetilde{g}_2$ be such that $g_1 \cdot \widetilde{g}_1 = \widehat{g}$ and $g_2 \cdot \widetilde{g}_2 = \widehat{g}$. We have,

$$\begin{aligned}
&\mathsf{deg}(f_1/g_1 +_{\mathsf{rf}} f_2/g_2)\\
&\qquad = \mathsf{deg}((f_1 \cdot \widetilde{g}_1 + f_2 \cdot \widetilde{g}_2)/\widehat{g})\\
&\qquad \leqslant \max\{\mathsf{deg}(f_1 \cdot \widetilde{g}_1), \mathsf{deg}(f_2 \cdot \widetilde{g}_2), \mathsf{deg}(\widehat{g})\}\\
&\qquad = \max\{\mathsf{deg}(f_1) + \mathsf{deg}(\widetilde{g}_1), \mathsf{deg}(f_2) + \mathsf{deg}(\widetilde{g}_2), \mathsf{deg}(\widehat{g})\}\\
&\quad (*) \leqslant \max\{\mathsf{deg}(f_1) + \mathsf{deg}(g_2), \mathsf{deg}(f_2) + \mathsf{deg}(g_1), \mathsf{deg}(g_1) + \mathsf{deg}(g_2)\}\\
&\qquad \leqslant \max\{\mathsf{deg}(f_1), \mathsf{deg}(g_1)\} + \max\{\mathsf{deg}(f_2), \mathsf{deg}(g_2)\}\\
&\qquad = \mathsf{deg}(f_1/g_1) + \mathsf{deg}(f_2/g_2)
\end{aligned}$$

where $(*)$ is due to the fact that $\mathsf{deg}(\widetilde{g}_1) \leqslant \mathsf{deg}(g_2)$ and $\mathsf{deg}(\widetilde{g}_2) \leqslant \mathsf{deg}(g_1)$. □

5.2.2 RFI-ABE

Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be predicate, p be a prime, $n \in \mathbb{N}$ and the following deterministic poly-time algorithms (rational fractions are considered over $\mathbb{Z}_p$):

- $\mathsf{sE}(x) \to \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B})$. On input $x \in \mathcal{X}$, the sender encoding algorithm sE outputs a vector of *polynomials* $\boldsymbol{c} = (c_1, \ldots, c_{|\mathsf{ct}_x|})$ in the variables $\boldsymbol{S} = (S_0, \ldots, S_w)$ and the common variables $\boldsymbol{B} = (B_1, \ldots, B_n)$. Without loss of generality, we assume that the polynomials do not contain any monomial B_i or any constant term. We the sake of computability, we also make the assumption that the degree of variables B_i is not greater than 1 in any monomial[2].

- $\mathsf{rE}(y) \to \boldsymbol{k}(\boldsymbol{R}, \boldsymbol{B}, A)$. On input $y \in \mathcal{Y}$, the receiver encoding algorithm rE outputs a vector of *rational fractions* $\boldsymbol{k} = (k_1, \ldots, k_{|\mathsf{sk}_y|})$ in the variables $\boldsymbol{R} = (R_1, \ldots, R_m)$, and the common variables $\boldsymbol{B}$ and A.

- $\mathsf{Pair}(x, y) \to \mathbf{E}$. On input $x \in \mathcal{X}$, $y \in \mathcal{Y}$, the Pair algorithm outputs a matrix $\mathbf{E} \in \mathbb{Z}_p^{|\mathsf{ct}_x| \times |\mathsf{sk}_y|}$,

We say an ABE is $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$-RFI if it is as described in Figure 5.2.

[2]Observe that, under these assumptions, all polynomials $\boldsymbol{c}$ evaluate to zero for $\boldsymbol{S} = \mathbf{0}_{w+1}$.

```
Setup($1^\lambda, \mathcal{X}, \mathcal{Y}$):
    $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_p^n$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$
    Outputs $\mathsf{msk} := (\boldsymbol{b}, \alpha)$, $\mathsf{mpk} := ([\![\boldsymbol{b}]\!]_1, [\![\alpha]\!]_\mathsf{t}) \in \mathbb{G}_1^n \times \mathbb{G}_\mathsf{t}$

Enc($\mathsf{mpk}, x \in \mathcal{X}$):
    $\boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}) \leftarrow \mathsf{sE}(x)$, $\boldsymbol{s} := (s_0, \dots, s_w) \xleftarrow{\$} \mathbb{Z}_p^{w+1}$
    Outputs $\mathsf{ct}_x := [\![\boldsymbol{c}(\boldsymbol{s}, \boldsymbol{b})]\!]_1 \in \mathbb{G}_1^{|\mathsf{ct}_x|}$, $\kappa := [\![\alpha s_0]\!]_\mathsf{t} \in \mathbb{G}_\mathsf{t}$

KeyGen($\mathsf{mpk}, \mathsf{msk} := (\boldsymbol{b}, \alpha), y \in \mathcal{Y}$):
    $\boldsymbol{k}(\boldsymbol{R}, \boldsymbol{B}, A) \leftarrow \mathsf{rE}(y)$, $\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^m$
    Outputs $\mathsf{sk}_y := [\![\boldsymbol{k}(\boldsymbol{r}, \boldsymbol{b}, \alpha)]\!]_2 \in \mathbb{G}_2^{|\mathsf{sk}_y|}$

Dec($\mathsf{mpk}, \mathsf{ct}_x := [\![\boldsymbol{c}]\!]_1, \mathsf{sk}_y := [\![\boldsymbol{k}]\!]_2$):
    $\mathbf{E} \leftarrow \mathsf{Pair}(x, y)$
    Outputs $[\![\boldsymbol{c}^\top \mathbf{E} \boldsymbol{k}]\!]_\mathsf{t}$
```

Figure 5.2: $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$-RFI-ABE.

Degree of a RFI-ABE

We define the degree of a RFI-ABE as the maximum degree over all the polynomials that can be created by multiplying a polynomial from $\mathsf{sE}(x)$ with a polynomial from $\mathsf{rE}(y)$ for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. The degree of a RFI-ABE allows to bound the probability[3] of inconsistent equality check between the generic model and the symbolic model. More formally, given a $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$-RFI-ABE, let

$$\begin{aligned} d_c &:= \max \{ \mathsf{deg}(c_i) \mid i \in [|\mathsf{ct}_x|], \boldsymbol{c} \leftarrow \mathsf{sE}(x), x \in \mathcal{X} \} \\ d_k &:= \max \{ \mathsf{deg}(k_i) \mid i \in [|\mathsf{sk}_y|], \boldsymbol{k} \leftarrow \mathsf{rE}(y), y \in \mathcal{Y} \} \end{aligned}$$

The degree of the pair encoding is defined by $d := d_c \cdot d_k$.

Correctness

The following theorem gives a sufficient condition for a RFI-ABE to be correct according to the definition of correctness from Section 2.1.5.

[3]Note that, by Schwartz-Zippel, the probability that two different polynomials over $\mathbb{Z}_p$, evaluated uniformly at random, give the same output is bounded by the maximum degree of the two, divided by p.

Theorem 16 (Correctness)**.** *Let* ABE *be a* $(p, n, \mathsf{rE}, \mathsf{sE}, \mathsf{Pair})$*-RFI-ABE for predicate* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$*. If for all* $x \in \mathcal{X}$*,* $y \in \mathcal{Y}$ *such that* $\mathsf{P}(x, y) = 1$*, it holds that* $\boldsymbol{c}^\top \mathbf{E} \boldsymbol{k} \sim_{\mathsf{rf}} AS_0$*, where* $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$*,* $\boldsymbol{k} \leftarrow \mathsf{rE}(y)$ *and* $\mathbf{E} \leftarrow \mathsf{Pair}(x, y)$*, then,* ABE is correct *in particular, for all* $x \in \mathcal{X}$*,* $y \in \mathcal{Y}$*,*

$$\Pr\left[\begin{array}{r} (\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{X}, \mathcal{Y}) \\ (\mathsf{ct}_x, \kappa) \leftarrow \mathsf{Enc}(\mathsf{mpk}, x) \\ \mathsf{sk}_y \leftarrow \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, y) \end{array} : \mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_y, \mathsf{ct}_x) \neq \kappa \right] \leqslant \frac{d\,|\mathsf{sk}_y|}{p}$$

where d *is the degree of* ABE *and the probability is taken over the coins of algorithms* Setup*,* Enc*,* KeyGen *and* Dec*.*

Proof. For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, let $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$, $\boldsymbol{k} \leftarrow \mathsf{rE}(y)$, $\mathbf{E} \leftarrow \mathsf{Pair}(x, y)$, and let $f/g := \boldsymbol{c}^\top \mathbf{E} \boldsymbol{k}$. Note that f and g are polynomials in $\mathbb{Z}_p[\boldsymbol{S}, \boldsymbol{R}, \boldsymbol{B}, A]$ and without loss of generality, we can assume g does not contain variables in $\boldsymbol{S}$ because such variables appear only in $\boldsymbol{c}$, which is a vector of *polynomials* and not arbitrary *rational fractions*. If $\mathsf{P}(x, y) = 1$, we have that $f/g \sim_{\mathsf{rf}} AS_0$. In particular, for all $\boldsymbol{s} \in \mathbb{Z}_p^{w+1}$, $\boldsymbol{r} \in \mathbb{Z}_p^m$, $\boldsymbol{b} \in \mathbb{Z}_p^n$, $\alpha \in \mathbb{Z}_p$ such that $g(\boldsymbol{r}, \boldsymbol{b}, \alpha) \neq 0$, we have $f(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{b}, \alpha)/g(\boldsymbol{r}, \boldsymbol{b}, \alpha) = \alpha s_0$, and the key computed by Dec corresponds to the key computed by Enc on x. To finish the proof, we need to bound the probability of g evaluating to 0 on a random point. Observe that, thanks to Lemma 3, if d is the degree of ABE, we have $\mathsf{deg}(g) \leqslant d\,|\mathsf{sk}_y|$. Therefore, the probability of a failed decryption is bounded the probability of g evaluating to 0 on a random point, which by Schwartz-Zippel (Lemma 1), is upper-bounded by $d\,|\mathsf{sk}_y|/p$. □

5.2.3 Symbolic security.

We present an algebraic condition on RFI-ABE that is sufficient to make it secure in the generic group model, as shown in Lemma 4 and Theorem 17.

Definition 31 (Symbolic security of RFI-ABE)**.** *We say a* $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$*-RFI-ABE is* symbolically secure *if for all* $x \in \mathcal{X}$*, there does not exist* $\{\mathbf{E}_y\}_{y \in \mathcal{Y}_x}$*, where for every* $y \in \mathcal{Y}_x$*,* $\mathbf{E}_y \in \mathbb{Z}_p^{|\mathsf{ct}_x| \times |\mathsf{sk}_y|}$*, such that*

$$\sum_{y \in \mathcal{Y}_x} \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B})^\top \mathbf{E}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) \sim_{\mathsf{rf}} AS_0$$

where $\boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}) \leftarrow \mathsf{sE}(x)$*,* $\mathcal{Y}_x \subseteq \mathcal{Y}$ *is the set of all* $y \in \mathcal{Y}$ *such that* $\mathsf{P}(x, y) = 0$*, and for all* $y \in \mathcal{Y}_x$*,* $\boldsymbol{R}_y := (R_{y,1}, \ldots, R_{y,m})$ *is a vector of fresh variables and* $\boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) \leftarrow \mathsf{rE}(y)(\boldsymbol{R} \to \boldsymbol{R}_y)$*.*

We show in the following lemma that the symbolic security above implies a seemingly stronger security notion which, roughly, allows to go from security in the private-key setting, to a public key setting.

Lemma 4 (From public to private key)**.** *Let* ABE *be a* $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$*-RFI-ABE. If* ABE *is symbolically secure, then, for all* $x \in \mathcal{X}$*, there does not exist* $\{\mathbf{E}_y\}_{y \in \mathcal{Y}_x}$ *and* $\gamma \in \mathbb{Z}_p$*, where for every* $y \in \mathcal{Y}_x$*,* $\mathbf{E}_y \in \mathbb{Z}_p^{(n+|\mathsf{ct}_x|)\times|\mathsf{sk}_y|}$*,such that*

$$\sum_{y \in \mathcal{Y}_x} (\boldsymbol{B}, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}))^\top \mathbf{E}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) + \gamma A \sim_{\mathsf{rf}} AS_0$$

where $\boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}) \leftarrow \mathsf{sE}(x)$*,* $\mathcal{Y}_x \subseteq \mathcal{Y}$ *is the set of all* $y \in \mathcal{Y}$ *such that* $\mathsf{P}(x, y) = 0$*, and for all* $y \in \mathcal{Y}_x$*,* $\boldsymbol{R_y} := (R_{y,1}, \ldots, R_{y,m})$ *is a vector of fresh variables and* $\boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) \leftarrow \mathsf{rE}(y)(\boldsymbol{R} \to \boldsymbol{R}_y)$*.*

Proof. By contradiction, suppose there is $x \in \mathcal{X}$, $\{\mathbf{E}_y\}_{y \in \mathcal{Y}_x}$ and $\gamma \in \mathbb{Z}_p$ such that $\sum_{y \in \mathcal{Y}_x} (\boldsymbol{B}, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}))^\top \mathbf{E}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) + \gamma A \sim_{\mathsf{rf}} AS_0$. By evaluating the formal variable S_i for $i \in [0, w]$ to 0, we obtain $\sum_{y \in \mathcal{Y}_x} (\boldsymbol{B}, \mathbf{0}_{|\mathsf{ct}_x|})^\top \mathbf{E}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) + \gamma A \sim_{\mathsf{rf}} 0$. Therefore, it must be $\sum_{y \in \mathcal{Y}_x} (\mathbf{0}_n, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}))^\top \mathbf{E}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) \sim_{\mathsf{rf}} AS_0$, and thus, the $|\mathsf{ct}_x|$ lower rows of matrices $\mathbf{E}_y$, say $\underline{\mathbf{E}}_y \in \mathbb{Z}_p^{|\mathsf{ct}_x|\times|\mathsf{sk}_y|}$ for all $y \in \mathcal{Y}_x$ are such that, $\sum_{y \in \mathcal{Y}_x} \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B})^\top \underline{\mathbf{E}}_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) \sim_{\mathsf{rf}} AS_0$, which contradicts the symbolic security of ABE. $\square$

5.2.4 Security in the Generic Group Model.

Let ABE be a $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$-RFI-ABE for $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, and $\mathcal{A}$ be an adversary. For $\mathtt{xxx} \in \{\mathsf{GGM}, \mathsf{SM}\}$, we define the experiments $\mathsf{Exp}^{\mathtt{xxx}}_{\mathsf{ABE}}(1^\lambda, \mathcal{A})$ in Figure 5.3. We define the advantages:

$$\mathcal{A}^{\mathtt{xxx}}_{\mathsf{ABE},\mathcal{A}}(\lambda) := \left| \frac{1}{2} - \Pr\left[\mathsf{Exp}^{\mathtt{xxx}}_{\mathsf{ABE}}(1^\lambda, \mathcal{A}) \to 1)\right] \right| .$$

Definition 32 (Adaptive security in the Generic Group Model)**.** *We say* ABE *is* adaptively secure in the generic group model *if there exists a negligible function* negl *such that for all p.p.t.adversaries* $\mathcal{A}$ *and for sufficiently large values of* $\lambda \in \mathbb{N}$*:* $\mathsf{Adv}^{\mathsf{GGM}}_{\mathsf{ABE},\mathcal{A}}(\lambda) \leqslant \mathsf{negl}(\lambda)$*.*

Very roughly, experiment $\mathsf{Exp}^{\mathtt{xxx}}(1^\lambda, \mathcal{A})$ is the security game where adversary $\mathcal{A}$ is trying to break the ABE (see Definition 11). However, there is a third party who implements the group, so that the adversary can only access to the group via handles. Internally, this third party keeps track of both, a symbolic representation of group elements and a real one (by sampling random values when required). The difference between $\mathsf{Exp}^{\mathsf{GGM}}$ and $\mathsf{Exp}^{\mathsf{SM}}$ is in equality checks, that are answered by using the generic representation and the symbolic representation of group elements respectively. Our next theorem bounds the probability of any distinguisher between $\mathsf{Exp}^{\mathsf{GGM}}$ and $\mathsf{Exp}^{\mathsf{SM}}$. Approximately, the only chance of distinguishing is that a *bad event*[4] occurs. Theorem 17 bounds the probability of a bad event happening.

[4]Equality checks in the symbolic representation and the generic representation differ.

```
$\mathsf{Exp}_{\mathsf{ABE}}^{\boxed{\mathsf{GGM}}\boxed{\mathsf{SM}}}(1^\lambda, \mathcal{A})$:
    $\mathtt{cnt} := 0$, $L_1^{\mathsf{eq}} = L_2^{\mathsf{eq}} = L_{\mathsf{t}}^{\mathsf{eq}} = L_1^{\sim} = L_2^{\sim} = L_{\mathsf{t}}^{\sim} := [1]$, $\mathcal{Q}_{\mathsf{chal}} = \mathcal{Q}_{\mathsf{sk}} := \varnothing$,
    $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_p^n$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $\boldsymbol{B} = (B_1, \ldots, B_n)$, $\beta \xleftarrow{\$} \{0,1\}$,
    $\mathsf{append}(L_1^{\mathsf{eq}}, \boldsymbol{b})$, $\mathsf{append}(L_{\mathsf{t}}^{\mathsf{eq}}, \alpha)$, $\mathsf{append}(L_1^{\sim}, \boldsymbol{B})$, $\mathsf{append}(L_{\mathsf{t}}^{\sim}, A)$,
    $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{add}}, \mathcal{O}_{\mathsf{pair}}, \boxed{\mathcal{O}_{\mathsf{cmp}}^{\mathsf{eq}}} \boxed{\mathcal{O}_{\mathsf{cmp}}^{\sim}}, \mathcal{O}_{\mathsf{chal}}(\alpha,\beta,\cdot), \mathcal{O}_{\mathsf{sk}}}(1^\lambda, p)$,
    if $\beta' = \beta$ and $\mathsf{P}(x,y) = 0$ for all $x \in \mathcal{Q}_{\mathsf{chal}}$, $y \in \mathcal{Q}_{\mathsf{sk}}$, output 1;
    otherwise, output 0

$\mathcal{O}_{\mathsf{add}}(s \in \{1, 2, \mathsf{t}\}, (i,j) \in \mathbb{N}^2)$:
    $\mathsf{append}(L_s^{\sim}, L_s^{\sim}[i] +_{\mathsf{rf}} L_s^{\sim}[j])$, $\mathsf{append}(L_s^{\mathsf{eq}}, L_s^{\mathsf{eq}}[i] + L_s^{\mathsf{eq}}[j])$

$\mathcal{O}_{\mathsf{pair}}((i,j) \in \mathbb{N}^2)$:
    $\mathsf{append}(L_{\mathsf{t}}^{\sim}, L_1^{\sim}[i] \cdot_{\mathsf{rf}} L_2^{\sim}[j])$, $\mathsf{append}(L_{\mathsf{t}}^{\mathsf{eq}}, L_1^{\mathsf{eq}}[i] \cdot L_2^{\mathsf{eq}}[j])$

$\mathcal{O}_{\mathsf{chal}}(\alpha \in \mathbb{Z}_p, \beta \in \{0,1\}, x \in \mathcal{X})$:
    $\boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}) = \mathsf{sE}(x)$, $\boldsymbol{S} := (S_0, \ldots, S_w)$,
    $\kappa_0^\star := AS_0$, $\kappa_1^\star := U$ where $U$ is a fresh formal variable,
    $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^w$, $v_0^\star := \alpha s_0$, $v_1^\star := u \xleftarrow{\$} \mathbb{Z}_p$,
    $\mathsf{append}\big(L_1^{\sim}, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B})\big)$, $\mathsf{append}(L_{\mathsf{t}}^{\sim}, \kappa_\beta^\star)$, $\mathsf{append}\big(L_1^{\mathsf{eq}}, \boldsymbol{c}(\boldsymbol{s}, \boldsymbol{b})\big)$, $\mathsf{append}(L_{\mathsf{t}}^{\mathsf{eq}}, v_\beta^\star)$,
    $\mathcal{Q}_{\mathsf{chal}} := \mathcal{Q}_{\mathsf{chal}} \cup \{x\}$

$\mathcal{O}_{\mathsf{sk}}(y \in \mathcal{Y})$:
    $\boldsymbol{R}_{\mathtt{cnt}} := (R_{\mathtt{cnt},1}, \ldots, R_{\mathtt{cnt},m})$, $\boldsymbol{k}(\boldsymbol{R}, \boldsymbol{B}, A) = \mathsf{rE}(y)$, $\boldsymbol{r}_{\mathtt{cnt}} \xleftarrow{\$} \mathbb{Z}_p^m$,
    $\mathsf{append}(L_2^{\sim}, \boldsymbol{k}(\boldsymbol{R}_{\mathtt{cnt}}, \boldsymbol{B}, A))$, $\mathsf{append}\big(L_2^{\mathsf{eq}}, \boldsymbol{k}(\boldsymbol{r}_{\mathtt{cnt}}, \boldsymbol{b}, \alpha)\big)$,
    $\mathtt{cnt} := \mathtt{cnt}+1$, $\mathcal{Q}_{\mathsf{sk}} := \mathcal{Q}_{\mathsf{sk}} \cup \{y\}$

$\mathcal{O}_{\mathsf{cmp}}^{\mathsf{eq}}(s \in \{1, 2, \mathsf{t}\}, (i,j) \in \mathbb{N}^2)$:
    Output 1 if $L_s^{\mathsf{eq}}[i] = L_s^{\mathsf{eq}}[j]$, otherwise output 0

$\mathcal{O}_{\mathsf{cmp}}^{\sim}(s \in \{1, 2, \mathsf{t}\}, (i,j) \in \mathbb{N}^2)$:
    Output 1 if $L_s^{\sim}[i] \sim_{\mathsf{rf}} L_s^{\sim}[j]$, otherwise output 0
```

Figure 5.3: Experiments $\mathsf{Exp}_{\mathsf{ABE}}^{\boxed{\mathsf{GGM}}\boxed{\mathsf{SM}}}(1^\lambda, \mathcal{A})$ We require that $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{chal}}$ at most once, and that $\mathsf{P}(x,y) = 0$ for $x \in \mathcal{Q}_{\mathsf{chal}}$ and all $y \in \mathcal{Q}_{\mathsf{sk}}$. In each procedure, the components inside a colored frame are only present in the games marked by the same frame.

```
$\mathrm{Game}_k(1^\lambda, \mathcal{A})$:
    $\mathtt{cnt} := 0,\ L_1^{\mathsf{eq}} = L_2^{\mathsf{eq}} = L_{\mathsf{t}}^{\mathsf{eq}} = L_1^{\sim} = L_2^{\sim} = L_{\mathsf{t}}^{\sim} := [1],\ \mathcal{Q}_{\mathsf{chal}} = \mathcal{Q}_{\mathsf{sk}} := \varnothing,$
    $\boldsymbol{b} \xleftarrow{\$} \mathbb{Z}_p^n,\ \alpha \xleftarrow{\$} \mathbb{Z}_p,\ \boldsymbol{B} = (B_1, \ldots, B_n),\ \beta \xleftarrow{\$} \{0,1\},$
    $\mathsf{append}(L_1^{\mathsf{eq}}, \boldsymbol{b}),\ \mathsf{append}(L_{\mathsf{t}}^{\mathsf{eq}}, \alpha),\ \mathsf{append}(L_1^{\sim}, \boldsymbol{B}),\ \mathsf{append}(L_{\mathsf{t}}^{\sim}, A),$
    $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{add}}, \mathcal{O}_{\mathsf{pair}}, \mathcal{O}_{\mathsf{cmp}}(k,\cdot,\cdot), \mathcal{O}_{\mathsf{chal}}(\alpha,\beta,\cdot), \mathcal{O}_{\mathsf{sk}}}(1^\lambda, p)$
    if $\beta' = \beta$ and $\mathsf{P}(x,y) = 0$ for all $x \in \mathcal{Q}_{\mathsf{chal}}$, $y \in \mathcal{Q}_{\mathsf{sk}}$, output 1;
    otherwise, output 0

$\mathcal{O}_{\mathsf{cmp}}(k \in \mathbb{N} \cup \{0\}, s \in \{1, 2, \mathsf{t}\}, (i,j) \in \mathbb{N}^2)$:
    On the $\nu$-th query:
      • if $\nu < k$: output 1 if $L_s^{\sim}[i] \sim_{\mathsf{rf}} L_s^{\sim}[j]$, otherwise output 0
      • if $\nu \geqslant k$: output 1 if $L_s^{\mathsf{eq}}[i] = L_s^{\mathsf{eq}}[j]$, otherwise output 0
```

Figure 5.4: Game_k for $k \in [0, q_{\mathsf{eq}}]$, for the proof of Theorem 17. We require that $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{chal}}$ at most once, and that $\mathsf{P}(x,y) = 0$ for $x \in \mathcal{Q}_{\mathsf{chal}}$ and all $y \in \mathbb{Q}_{\mathsf{sk}}$. We refer to Figure 5.3 for the description of oracles $\mathcal{O}_{\mathsf{add}}, \mathcal{O}_{\mathsf{pair}}, \mathcal{O}_{\mathsf{chal}}$ and $\mathcal{O}_{\mathsf{sk}}$.

Security proof of the generic construction.

Our next result establishes that symbolically secure RFI-ABE are also secure in the GGM.

Theorem 17 (From symbolic to generic security)**.** *Let* ABE *be a symbolically secure* $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$*-RFI-ABE for* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$*. Let* $\lambda \in \mathbb{N}$ *be the security parameter, and let* $\mathcal{A}$ *be an adversary that on input* $(1^\lambda, p)$*, makes* q_{sk}*,* q_{add}*,* $q_{\mathsf{pair}} \in \mathbb{N}$ *calls to oracles* $\mathcal{O}_{\mathsf{sk}}$*,* $\mathcal{O}_{\mathsf{add}}$*,* $\mathcal{O}_{\mathsf{pair}}$ *respectively, and exactly one call to* $\mathcal{O}_{\mathsf{chal}}$*. We have:*

$$\mathcal{A}^{\mathsf{GGM}}_{\mathsf{ABE},\mathcal{A}}(\lambda) \leqslant \frac{6d(n + |\mathsf{ct}| + q_{\mathsf{sk}}\,|\mathsf{sk}| + q_{\mathsf{add}} + q_{\mathsf{pair}})^4}{p},$$

where d *is the degree of* ABE*,* $|\mathsf{ct}|$ *and* $|\mathsf{sk}|$ *are upper-bounds on the size of any ciphertext and any secret key respectively.*

Proof. The proof of Theorem 17 proceeds in two steps. First, we show that the advantages $\mathcal{A}^{\mathsf{GGM}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ and $\mathcal{A}^{\mathsf{SM}}_{\mathsf{ABE},\mathcal{A}}(\lambda)$ are negligibly different. Then, we argue that $\mathcal{A}^{\mathsf{SM}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = 0$, using the symbolic security of ABE.

For the first step, we proceed via a hybrid argument, using Game_k for $k \in [0, q_{\mathsf{cmp}}]$, defined in Figure 5.4, and we denote by Adv_k the advantage of $\mathcal{A}$ in Game_k. It is clear that Game_0 is the same as $\mathsf{Exp}^{\mathsf{GGM}}_{\mathsf{ABE}}(1^\lambda, \mathcal{A})$, and $\mathrm{Game}_{q_{\mathsf{cmp}}}$ is the same as $\mathsf{Exp}^{\mathsf{SM}}_{\mathsf{ABE}}(1^\lambda, \mathcal{A})$. We show how to transition from Game$_k$ to Game$_{k+1}$ for all $i \in [0, q_{\mathsf{cmp}} - 1]$ in Lemma 6, but before, we prove the following helper lemma.

Lemma 5. *For all $s \in \{1, 2, \mathsf{t}\}$, $f/g \in L_s^{\sim}$,*

$$\mathsf{deg}(f/g) \leqslant 2d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big).$$

Proof of Lemma 5. First, note that for all $s \in \{1, 2, \mathsf{t}\}$, $f/g \in L_s^{\sim}$ is a linear combination of rational fractions in $\widehat{L}_s$ where

$$\begin{aligned}
\widehat{L}_1 &:= \{B_j \mid j \in [n]\} \cup \{\mathsf{sE}(x) \mid x \in \mathcal{Q}_{\mathsf{chal}}\} \\
\widehat{L}_2 &:= \{\mathsf{rE}(y) \mid y \in \mathcal{Q}_{\mathsf{sk}}\} \\
\widehat{L}_{\mathsf{t}} &:= \{A, \kappa^\star_\beta\} \cup \{f_1/g_1 \cdot_{\mathsf{rf}} f_2/g_2 \mid f_1/g_1 \in \widehat{L}_1, f_2/g_2 \in \widehat{L}_2\}
\end{aligned}$$

Thus, for $s \in \{1, 2\}$, $f/g \in L_s^{\sim}$, by Lemma 3, we have $\mathsf{deg}(f/g) \leqslant d\,|\widehat{L}_s|$. Furthermore, for $f/g \in L_{\mathsf{t}}^{\sim}$ we have $\mathsf{deg}(f/g) \leqslant 2d\,|\widehat{L}_{\mathsf{t}}|$. We conclude the proof using the fact that $|\widehat{L}_1| \leqslant n + |\mathsf{ct}|$, $|\widehat{L}_2| \leqslant q_{\mathsf{sk}}\,|\mathsf{sk}|$, $|\widehat{L}_s| \leqslant 2 + |\widehat{L}_1|\,|\widehat{L}_2|$. ☐

Lemma 6. *For all $k \in [0, q_{\mathsf{cmp}}]$,*

$$|\mathsf{Adv}_{k+1} - \mathsf{Adv}_k| \leqslant \frac{12d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big)}{p}\ .$$

Proof of Lemma 6. We bound $|\mathsf{Adv}_{k+1} - \mathsf{Adv}_k|$, by using the fact that Game$_{k+1}$ and Game$_k$ only differ on the $k+1$-th query to $\mathcal{O}_{\mathsf{cmp}}$. In particular, the output of $\mathcal{O}_{\mathsf{cmp}}(k, s, (i, j))$, is different in these two games if

(1) $L_s^{\sim}[i] \sim_{\mathsf{rf}} L_s^{\sim}[j]$ but $L_s^{\mathsf{eq}}[i] \neq L_s^{\mathsf{eq}}[j]$, or

(2) $L_s^{\mathsf{eq}}[i] = L_s^{\mathsf{eq}}[j]$ but not $L_s^{\sim}[i] \sim_{\mathsf{rf}} L_s^{\sim}[j]$.

When considering event (1), we write $L_s^{\sim}[i] = f_i/g_i$ and $L_s^{\sim}[j] = f_j/g_j$. Since $f_i/g_i \sim_{\mathsf{rf}} f_j/g_j$, we have that for all points $\boldsymbol{x}$, it holds $f_i(\boldsymbol{x}) \cdot g_j(\boldsymbol{x}) = f_j(\boldsymbol{x}) \cdot g_i(\boldsymbol{x})$. Moreover, if $g_i(\boldsymbol{x}), g_j(\boldsymbol{x})$ are not null, then $(f_i/g_i)(\boldsymbol{x}) = (f_j/g_j)(\boldsymbol{x})$, that is, $L_s^{\mathsf{eq}}[i] = L_s^{\mathsf{eq}}[j]$. Therefore, it suffices to bound the probability that g_i or g_j evaluate to 0 on a random point. By Lemma 1 (Schwartz Zippel) and Lemma 5, we get an upper-bound of $4d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big)/p$.

Now, we consider event (2). If $L_s^{\mathsf{eq}}[i] = L_s^{\mathsf{eq}}[j] \neq \perp$, event (2) corresponds to the case when $f_i \cdot g_j - f_j \cdot g_i \neq 0$, and $(f_i \cdot g_j - f_j \cdot g_i)(\boldsymbol{x}) = 0$, for a random point $\boldsymbol{x}$. By Schwartz-Zippel (Lemma 1) and Lemma 5, we get an upper-bound of $4d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big)/p$. Finally, using the same argument as for event (1), we can bound the probability that $L_s^{\mathsf{eq}}[i'] = L_s^{\mathsf{eq}}[j'] = \perp$ by $4d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big)/p$.

Summing up all three upper-bounds, we obtain the lemma. ☐

Lemma 7. $\mathcal{A}^{\mathsf{SM}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = 0$.

Proof of Lemma 7. We show that the view of any adversary $\mathcal{A}$, playing the experiment $\mathsf{Exp}^{\mathsf{SM}}_{\mathsf{ABE}}(1^\lambda, \mathcal{A})$, is independent of β.

The only information that might be leadked about α can be due to the output of $\mathcal{O}^{\sim}_{\mathsf{cmp}}$ on queries of the form $(\mathsf{t}, (i, j))$, for some $(i, j) \in \mathbb{N}^2$. We know that $L_\mathsf{t}[i]$ and $L_\mathsf{t}[j]$ are linear combinations of rational fractions in $\widehat{L}_\mathsf{t}$, where $\widehat{L}_\mathsf{t}$ is defined as in Lemma 5. Namely, we write:

$$L_\mathsf{t}[i] = \sum_{\ell \in [|\widehat{L}_\mathsf{t}|]}^{\mathsf{rf}} \alpha^{(i)}_\ell \cdot f_\ell / g_\ell \qquad \text{and} \qquad L_\mathsf{t}[j] = \sum_{\ell \in [|\widehat{L}_\mathsf{t}|]}^{\mathsf{rf}} \alpha^{(j)}_\ell \cdot f_\ell / g_\ell$$

for $\alpha^{(i)}_1, \alpha^{(j)}_1, \ldots, \alpha^{(i)}_{|\widehat{L}_\mathsf{t}|}, \alpha^{(j)}_{|\widehat{L}_\mathsf{t}|} \in \mathbb{Z}_p$, where for all $\ell \in [|\widehat{L}_\mathsf{t}|]$, $\widehat{L}_\mathsf{t}[\ell] := f_\ell / g_\ell$. Now, let $\ell^\star \in \mathbb{N}^*$ be such that $f_{\ell^\star}/g_{\ell^\star} = \kappa^\star_\beta$.

If $\alpha^{(i)}_{\ell^\star} = \alpha^{(j)}_{\ell^\star}$, then $L_\mathsf{t}[i] \sim_{\mathsf{rf}} L_\mathsf{t}[j] \Leftrightarrow \sum^{\mathsf{rf}}_{\ell \neq \ell^\star} \alpha^{(i)}_\ell \cdot f_\ell/g_\ell \sim_{\mathsf{rf}} \sum^{\mathsf{rf}}_{\ell \neq \ell^\star} \alpha^{(j)}_\ell \cdot f_\ell/g_\ell$, which is independent of β.

If $\alpha^{(i)}_{\ell^\star} \neq \alpha^{(j)}_{\ell^\star}$, then

$$L_\mathsf{t}[i] \sim_{\mathsf{rf}} L_\mathsf{t}[j] \Rightarrow \kappa^\star_\beta \sim_{\mathsf{rf}} \sum^{\mathsf{rf}}_{\ell \neq \ell^\star} \frac{\alpha^{(j)}_\ell - \alpha^{(i)}_\ell}{\alpha^{(i)}_{\ell^\star} - \alpha^{(j)}_{\ell^\star}} \cdot f_\ell / g_\ell$$

and thus, there exist $x \in \mathcal{X}$, $\{\mathbf{E}^*_y\}_{y \in \mathcal{Q}_{\mathsf{sk}}}$ and $\gamma \in \mathbb{Z}_p$ such that for $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$ and $\boldsymbol{k}_y \leftarrow \mathsf{rE}(y)$, $\sum_{y \in \mathcal{Q}_{\mathsf{sk}}} (\boldsymbol{B}, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{B}))^\top \mathbf{E}^*_y \boldsymbol{k}_y(\boldsymbol{R}_y, \boldsymbol{B}, A) + \gamma A \sim_{\mathsf{rf}} \kappa^\star_\beta$. When $\beta = 0$, $\kappa^\star_0 = AS_0$, which, together with Lemma 4, contradicts the symbolic security of ABE. On the other hand, when $\beta = 1$, $\kappa^\star_1 = U$, which cannot be a linear combination of rational fractions on a disjoint set of formal variables. We conclude that $L_\mathsf{t}[i] \not\sim_{\mathsf{rf}} L_\mathsf{t}[j]$, regardless of β. □

Summing everything up and applying the *union bound* on the bound given by the hybrid step (Lemma 6), we obtain,

$$\mathcal{A}^{\mathsf{GGM}}_{\mathsf{ABE},\mathcal{A}}(\lambda) \leqslant \frac{12d\big(2 + (n + |\mathsf{ct}|)q_{\mathsf{sk}}|\mathsf{sk}|\big)}{p} q_{\mathsf{cmp}} \ .$$

We conclude the proof of Theorem 17 using the fact that

$$q_{\mathsf{cmp}} \leqslant \frac{(|L^{\mathsf{eq}}_1| + |L^{\mathsf{eq}}_2| + |L^{\mathsf{eq}}_\mathsf{t}|)^2}{2} \leqslant \frac{(2 + n + |\mathsf{ct}| + q_{\mathsf{sk}}\,|\mathsf{sk}| + q_{\mathsf{add}} + q_{\mathsf{pair}})^2}{2} \ .$$

Note that without loss of generality we have assumed that no comparison is performed twice, and thus, the number of performed comparisons is bounded by the number of pairs of different elements. □

5.3 Pair Encodings

In this section, we recall the definition of pair encodings, which have been originally introduced in [24, 190] as a useful abstraction to build ABE whose security proof rely on the Dual System Encryption techniques [188] (roughly speaking, a pair encoding is a private-key, one-time secure variant of ABE). We show in Theorem 18 that any pair encoding that is symbolically secure, as defined in [11] (this is the weakest possible notion of security for pair encoding), yields a symbolically secure RFI-ABE via the construction presented in Figure 5.2. The RFI-ABE obtained are roughly twice more efficient that those obtained via previous dual system frameworks, albeit relying on the Generic Group Model.

For convenience, we adopt the last definition of *pair encodings* given by Agrawal and Chase in [11]. Note that it differs from the original definition by Attrapadung [24] (recalled in Section 4.2.3) since the polynomials are assumed to be of a more restrictive form. However, Agrawal and Chase argue [11, Appendix A] that their restricted formulation of pair encodings does not lose generality, by providing a transformation from the previous formulation to the more restricted one, that does not affect the security of the resulting Predicate Encryption scheme.

Pair encodings.

Let p be a prime, $n \in \mathbb{N}$. A (p, n)-pair-encoding for predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ consists of the following deterministic poly-time algorithms (polynomials are considered over $\mathbb{Z}_p$):

- $\mathsf{sE}(x) \to \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})$. On input $x \in \mathcal{X}$, the sender encoding algorithm, sE, outputs a vector of polynomials, $\boldsymbol{c}$, in the non-lone variables $\boldsymbol{S} = (S_0, \ldots, S_w)$, the lone variables $\widehat{\boldsymbol{S}} = (\widehat{S}_1, \ldots, \widehat{S}_{\widehat{w}})$, and the common variables $\boldsymbol{B} = (B_1, \ldots, B_n)$, where we require every polynomial be a linear combination of the monomials $\{\widehat{S}_i,\ S_j B_k \mid i \in [\widehat{w}], j \in [0, w], k \in [n]\}$.

- $\mathsf{rE}(y) \to \boldsymbol{k}(\boldsymbol{R}, \widehat{\boldsymbol{R}}, \boldsymbol{B}, A)$. On input $y \in \mathcal{Y}$, the receiver encoding algorithm, rE, outputs a vector of polynomials, $\boldsymbol{k}$, in the non-lone variables $\boldsymbol{R} = (R_1, \ldots, R_m)$, the lone variables $\widehat{\boldsymbol{R}} = (\widehat{R}_1, \ldots, \widehat{R}_{\widehat{m}})$, A, and the common variables $\boldsymbol{B}$, where every polynomial be a linear combination of the monomials $\{A,\ \widehat{R}_i,\ R_j B_k \mid i \in [\widehat{m}], j \in [m], k \in [n]\}$.

- $\mathsf{Pair}(x, y) \to \mathbf{E}, \widehat{\mathbf{E}}$. On input $x \in \mathcal{X}$, $y \in \mathcal{Y}$, the Pair algorithm outputs matrices $\mathbf{E} \in \mathbb{Z}_p^{(w+1)\times|\boldsymbol{k}|}$, and $\widehat{\mathbf{E}} \in \mathbb{Z}_p^{|\boldsymbol{c}|\times m}$.

Correctness.

A pair encoding is *correct* if for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 1$, it holds $\boldsymbol{S}^\top \mathbf{E} \boldsymbol{k} + \boldsymbol{c}^\top \mathbf{E} \boldsymbol{R} = AS_0$, where $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$, $\boldsymbol{k} \leftarrow \mathsf{rE}(y)$, $(\mathbf{E}, \widehat{\mathbf{E}}) \leftarrow \mathsf{Pair}(x, y)$.

Symbolic security [11].

A pair encoding is *symbolically secure* if for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 0$, there is no matrix $\mathbf{E}^* \in \mathbb{Z}_p^{(1+w+|\boldsymbol{c}|)\times(m+|\boldsymbol{k}|)}$ such that $(\boldsymbol{S}, \boldsymbol{c})^\top \mathbf{E}^* (\boldsymbol{R}, \boldsymbol{k}) = AS_0$, where $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$, $\boldsymbol{k} \leftarrow \mathsf{rE}(y)$ and $\boldsymbol{S} = (S_0, \dots, S_w)$ and $\boldsymbol{R} = (R_1, \dots, R_m)$ are the variables appearing in polynomials $\boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})$ and $\boldsymbol{k}(\boldsymbol{R}, \widehat{\boldsymbol{R}}, \boldsymbol{B}, A)$.

Our next theorem shows that any symbolically secure (p, n)-pair-encoding $(\mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$ [11] yields a symbolically secure $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$-RFI-ABE.

Theorem 18. *[Symbolically secure pair encoding ⇒ symbolically secure RFI-ABE] Let* $(\mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$ *be a* (p, n)*-pair-encoding for predicate* $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$*. The construction described in Figure 5.2 is a symbolically secure,* $(p, n, \mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$*-RFI-ABE.*

Proof. We prove the symbolic security of the ABE by contradiction. Suppose there is $x \in \mathcal{X}$, and matrices $\{\mathbf{E}_y^*\}_{y \in \mathcal{Y}_x}$ such that

$$\sum_{y \in \mathcal{Y}_x} \big(\boldsymbol{S}, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})\big)^\top \mathbf{E}_y^* \big(\boldsymbol{R}_y, \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, A)\big) = AS_0,$$

where $\boldsymbol{c} \leftarrow \mathsf{sE}(x)$, $\mathcal{Y}_x \subseteq \mathcal{Y}$ is the set of all $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 0$, and for all $y \in \mathcal{Y}_x$, $\boldsymbol{R_y} := (R_{y,1}, \dots, R_{y,m})$, $\widehat{\boldsymbol{R}}_{\boldsymbol{y}} := (\widehat{R}_{y,1}, \dots, \widehat{R}_{y,\widehat{m}})$, $\boldsymbol{k}_y = \mathsf{rE}(y)$.

First, notice that the matrices $\{\underline{\mathbf{E}}_y^*\}_{y \in \mathcal{Y}_x}$ that result from removing the 2nd to the $(w+1)$-th rows of $\{\mathbf{E}_y^*\}_{y \in \mathcal{Y}_x}$ satisfy,

$$\sum_{y \in \mathcal{Y}_x} \big(S_0, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})\big)^\top \underline{\mathbf{E}}_y^* \big(\boldsymbol{R}_y, \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, A)\big) = AS_0 \ . \tag{5.1}$$

That is because those rows would contribute to the whole summation with monomials that result from the combination of variables $(S_1, \dots, S_w)$ with $(\boldsymbol{R}_y, \boldsymbol{k}_y)$, i.e., monomials of the form $S_i R_{y,j}, S_i A, S_i \widehat{R}_{y,\widehat{j}}, S_i R_{y,j}, B_k$ for $i \in [w]$, $j \in [m]$, $\widehat{j} \in [\widehat{m}]$, $k \in [n]$ and $y \in \mathcal{Y}_x$. Note that these monomials do not appear anywhere else and therefore their contribution can be ignored.

Now, for all $y \in \mathcal{Y}_x$, we evaluate the polynomials from (5.1) in $\boldsymbol{R}_{\tilde{y}} = \mathbf{0}_m$ and $\widehat{\boldsymbol{R}}_{\tilde{y}} = \mathbf{0}_{\widehat{m}}$, for all $\tilde{y} \in \mathcal{Y}_x \backslash \{y\}$, and $A = 0$. We obtain,

$$\big(S_0, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})\big)^\top \underline{\mathbf{E}}_y^* \big(\boldsymbol{R}_y, \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, 0)\big) = 0 \ . \tag{5.2}$$

Now, we show that there exists $y^\star \in \mathcal{Y}_x$ and a constant $\rho \in \mathbb{Z}_p^*$ such that $(S_0, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B}))^\top \rho \underline{\mathbf{E}}^*_{y^\star}(\mathbf{0}_m, \boldsymbol{k}_{y^\star}(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, A)) = AS_0$, which, together with (5.2), implies[5] $(S_0, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B}))^\top \rho \underline{\mathbf{E}}^*_{y^\star}(\boldsymbol{R}_{y^\star}, \boldsymbol{k}_{y^\star}(\boldsymbol{R}_{y^\star}, \widehat{\boldsymbol{R}}_{y^\star}, \boldsymbol{B}, A)) = AS_0$, thereby contradicting the symbolic security of $(\mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$.

We do so in two steps, where in Step 1 (Lemma 8), we show that for all $y \in \mathcal{Y}_x$, we can assume some structural properties of the matrix $\underline{\mathbf{E}}^*_y$. In step 2, we show how these structural properties combined with (5.1) and (5.2) allow us to derive the desired $y^\star \in \mathcal{Y}_x$.

Lemma 8 (Step 1)**.** *For all $x \in \mathcal{X}$, $y \in \mathcal{Y}_x$ and let $\boldsymbol{e}_y^{(1)} \in \mathbb{Z}_p^m$, $\boldsymbol{e}_y^{(2)} \in \mathbb{Z}_p^{|\boldsymbol{k}|}$, $\mathbf{E}_y^{(3)} \in \mathbb{Z}_p^{|\boldsymbol{c}| \times m}$, $\mathbf{E}_y^{(4)} \in \mathbb{Z}_p^{|\boldsymbol{c}| \times |\boldsymbol{k}|}$ satisfying (5.2), when setting*

$$\underline{\mathbf{E}}^*_y := \begin{pmatrix} \boldsymbol{e}_y^{(1)\top} & \boldsymbol{e}_y^{(2)\top} \\ \mathbf{E}_y^{(3)} & \mathbf{E}_y^{(4)} \end{pmatrix}$$

then, it also holds for $\boldsymbol{e}_y^{(1)} = \mathbf{0}_m$ and $\mathbf{E}_y^{(4)} = \mathbf{0}_{|\boldsymbol{c}|,|\boldsymbol{k}|}$.

Proof of Lemma 8. We first show that $\boldsymbol{c}(S_0, \widehat{\boldsymbol{S}}, \boldsymbol{B})^\top \mathbf{E}_y^{(4)} \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, 0) = 0$. Observe, by definition of a pair encoding (see Section 5.3), that the polynomial $\boldsymbol{c}(S_0, \widehat{\boldsymbol{S}}, \boldsymbol{B})^\top \mathbf{E}_y^{(4)} \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, 0)$ is a linear combination of monomials of the form:

$$\widehat{S}_{\widehat{i}} \widehat{R}_{y,\widehat{j}}, \; \widehat{S}_{\widehat{i}} B_k R_{y,j}, \; S_0 B_k \widehat{R}_{y,\widehat{j}}, \quad \text{or} \quad S_0 B_k B_{k'} R_j \tag{5.3}$$

for $\widehat{i} \in [\widehat{w}]$, $j \in [m]$, $\widehat{j} \in [\widehat{m}]$, $k, k' \in [n]$.

Now note that, by (5.2) we have

$$\begin{aligned} &\boldsymbol{c}(S_0, \widehat{\boldsymbol{S}}, \boldsymbol{B})^\top \mathbf{E}_y^{(4)} \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, 0) = \\ &\qquad - S_0 \boldsymbol{e}_y^{(1)\top} \boldsymbol{R}_y - S_0 \boldsymbol{e}_y^{(2)\top} \boldsymbol{k}_y(\boldsymbol{R}_y, \widehat{\boldsymbol{R}}_y, \boldsymbol{B}, 0) - \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})^\top \mathbf{E}_y^{(3)} \boldsymbol{R}_y \ . \end{aligned}$$

On the right-hand side of the previous equation, none of the monomials from (5.3) appear, therefore, both, the left-hand side and the right-hand side of the previous equation, must be zero.

Finally, by evaluating (5.2) on $\widehat{\boldsymbol{S}} = \mathbf{0}_{\widehat{w}}$, $\widehat{\boldsymbol{R}}_y = \mathbf{0}_{\widehat{m}}$, $\boldsymbol{B} = \mathbf{0}_n$, we obtain $S_0 \boldsymbol{e}_y^{(1)\top} \boldsymbol{R}_y = 0$. □

Step 2. Substracting equation (5.2) from (5.1) for every $y \in \mathcal{Y}_x$, we obtain,

$$\sum_{y \in \mathcal{Y}_x} \big(S_0, \boldsymbol{c}(\boldsymbol{S}, \widehat{\boldsymbol{S}}, \boldsymbol{B})\big)^\top \underline{\mathbf{E}}^*_y \big(\mathbf{0}_m, \boldsymbol{k}_y(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, A)\big) = AS_0 \ .$$

[5] That is due to linearity and the fact that variable A never appears multiplied by other variables in the polynomials in $\boldsymbol{k}_y$.

Now, evaluating the above equation on $B_i = 0$ and $\widehat{S}_j = 0$ for all $i \in [n]$ and $j \in [\widehat{w}]$, we obtain,

$$\sum_{y \in \mathcal{Y}_x} S_0 \boldsymbol{e}_y^{(2)\top} \boldsymbol{k}_y(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, A) = AS_0 \ .$$

But $\boldsymbol{k}_y(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, A)$ are vectors of polynomials linear in A, therefore we have $\sum_{y \in \mathcal{Y}_x} \boldsymbol{e}_y^{(2)\top} \boldsymbol{k}_y(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, 1) = 1$. And in particular, there exists $y^\star \in \mathcal{Y}_x$ such that $\boldsymbol{e}_{y^\star}^{(2)\top} \boldsymbol{k}_{y^\star}(\mathbf{0}_m, \mathbf{0}_{\widehat{m}}, \mathbf{0}_n, 1) = \mu \neq 0$.

Consequently, the matrix

$$\widetilde{\mathbf{E}} := 1/\mu \begin{pmatrix} \mathbf{0}_m^\top & \boldsymbol{e}_{y^\star}^{(2)^\top} \\ \mathbf{E}_{y^\star}^{(3)} & \mathbf{0}_{|\boldsymbol{c}|,|\boldsymbol{k}|} \end{pmatrix}$$

is such that $\big(\boldsymbol{S}, \boldsymbol{c}(\boldsymbol{S}, \boldsymbol{S}', \boldsymbol{B})\big)^\top \widetilde{\mathbf{E}}\big(\mathbf{0}, \boldsymbol{k}_{y^\star}(\mathbf{0}, \mathbf{0}, \mathbf{0}, A)\big) = AS_0$. Finally, combining this fact with Lemma 8 leads to a contradiction of the symbolic security of $(\mathsf{sE}, \mathsf{rE}, \mathsf{Pair})$. $\square$

Together with Theorem 17, Theorem 18 shows that any pair encoding gives a secure RFI-ABE adaptively secure in the Generic Group Model.

5.4 Concrete RFI-ABE

Focusing on the Generic Group Model allowed us to build schemes that are often simpler and more efficient compared to existing schemes from the literature (see table in Figure 5.1 for a comparison between the most efficient ABE). In this section, we show a selection of schemes that illustrate the versatility of our framework. Our contribution here is threefold:

i) We design new pair encodings, which give new, more efficient RFI-ABE via our framework (Figure 5.2). This is the case of IPE 2, compact KP-ABE, unbounded KP-ABE, CP-ABE, and unbounded CP-ABE.

ii) We use our framework on existing pair encodings, to obtain new, more efficient RFI-ABE, albeit relying on a stronger assumption. This is the case of IBE 1 and IPE 1, whose underlying pair encoding are implicit in the work of [191] and [138] respectively.

iii) We use our framework on existing pair encodings, to prove new security guarantee s on existing RFI-ABE. This is the case of IBE 2 from [62] and KP-ABE from [116]. Here, our framework, when input on the pair encodings implicitly given in [62, 116], outputs exactly the same RFI-ABE present in those papers: there is no efficiency gain. However, we can prove these RFI-ABE *adaptively* secure, under GGM, while they were proved only *selectively* secure, based on standard assumptions.

Overall, our new framework captures previous schemes (contribution iii), and improves upon many others (contributions i and ii), at the price of considering generic security.

KP-ABE	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\mathsf{time}_{\mathsf{Dec}}$	security
GPSW06 [116]	U	ℓ	$\lvert\Gamma\rvert$	$\lvert\Gamma\rvert P+\ell E$	(DBDH, selective) (GGM, adaptive)
RW13 [171] Our uKP-ABE	3 2	3ℓ 2ℓ	$2\lvert\Gamma\rvert+1$ $2\lvert\Gamma\rvert$	$(2\lvert\Gamma\rvert+1)P+3\ell E$ $2\lvert\Gamma\rvert P+2\ell E$	(q-type, selective) (GGM, adaptive)
ALP11 [29] Our cKP-ABE	γ U	$(\gamma+1)\ell$ ℓU	2 2	$2P+\lvert\Gamma\rvert\ell E$ $2P+\lvert\Gamma\rvert\ell E$	(q-type, selective) (GGM, adaptive)

CP-ABE	$\lvert\mathsf{mpk}\rvert$	$\lvert\mathsf{sk}\rvert$	$\lvert\mathsf{ct}\rvert$	$\mathsf{time}_{\mathsf{Dec}}$	security
W11 [189] Our CP-ABE	$U+1$ U	$\lvert\Gamma\rvert+2$ $\lvert\Gamma\rvert+1$	$2\ell+1$ $\ell+1$	$(\lvert\Gamma\rvert+2)P+2\ell E$ $(\lvert\Gamma\rvert+1)P+\ell E$	(q-type, selective) (GGM, adaptive)
RW13 [171] Our uCP-ABE	4 4	$2\lvert\Gamma\rvert+2$ $\lvert\Gamma\rvert+2$	$3\ell+1$ 3ℓ	$(2\lvert\Gamma\rvert+2)P+3\ell E$ $(\lvert\Gamma\rvert+2)P+3\ell E$	(q-type, selective) (GGM, adaptive)

Table 5.1: Comparison of the most efficient existing KP-ABE and CP-ABE schemes for (monotone) boolean span programs, based on prime-order bilinear groups. We denote by $|\Gamma|$ the attribute set size, γ the maximum size for Γ (if bounded), U the size of the attribute universe (if bounded small-universe), ℓ is the size of the access structure. For $|\mathsf{ct}|$, we omit the additive overhead of $O(|\Gamma|)$ *bits* for transmitting the attribute vector (in KP-ABE), or $O(\ell)$ *bits* for transmitting the access structure. We use $\mathsf{time}_{\mathsf{Dec}}$ to denote the decryption time. Numbers in $|\mathsf{mpk}|$, $|\mathsf{sk}|$ and ct columns correspond to the number of group elements from source groups. All mpk and all ct have an additional group element in $\mathbb{G}_{\mathsf{t}}$. We write E to express exponentiation time in source groups and we use P to denote the time of one pairing operation. Decryption algorithms have been optimized taking into account that $P > E$. We use pink to indicate new results.

5.4.1 Identity-Based Encryption (IBE)

We refer to Section 2.1.4 for more details about Identity-Based Encryption. We have $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p$, and the predicate P is defined as: $\mathsf{P}(x, y) = 1$ iff $x = y$.

5.4.1 IBE 1 [191]

This is the prime-order version of the IBE from [191], which uses the Déjà Q framework, introduced in [73].

$$\boldsymbol{B} := B,\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := \varnothing \quad (n = 1, w = 0, m = 0)$$

$$\mathsf{sE}(x) := S_0(B + x) \qquad \mathsf{rE}(y) := A/(B + y) \qquad \mathsf{Pair}(x, y) := 1$$

It is an open problem to translate this framework, which uses *composite-order* bilinear groups, to the more efficient [122] prime order setting. This yields one of the most efficient IBE, as illustrated in the benchmark Figure 5.7. Note that an unpublished manuscript from Eike Kiltz and Gregory Neven, cited in [67, citation 25], already proves adaptive security of IBE 1 in the GGM.

Proof of symbolic security. Suppose there exist $x \in \mathbb{Z}_p$, and $\{e_y \in \mathbb{Z}_p\}_{y \in \mathcal{Y}_x}$ such that

$$\sum_{y \in \mathcal{Y}_x}^{\mathsf{rf}} S_0(B + x) e_y A/(B + y) \sim_{\mathsf{rf}} AS_0 \ .$$

For all $y \in \mathcal{Y}_x$, $y \neq x$, we evaluate the above rational fraction on $B = -x$ to obtain $0 \sim_{\mathsf{rf}} AS_0$, which is a contradiction.

More formally, this corresponds to the application of rules com-den, div-split, eval-var on $B = -x$, and zero-prod, as explained in the example of Section 5.5. □

5.4.1 IBE 2 [62]

This is the Identity-Based Encryption scheme presented in [62], which we prove adaptively secure in the GGM ([62] proved it selectively secure based on DBDH).

$$\boldsymbol{B} := (B_1, B_2),\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := R \quad (n = 2, w = 0, m = 1)$$

$$\mathsf{sE}(x) := (S_0, S_0(B_1 + xB_2)) \qquad \mathsf{rE}(y) := (R, A + R(B_1 + yB_2))$$

$$\mathsf{Pair}(x, y) := \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Proof of symbolic security. The underlying pair encoding of IBE 2 falls under the definition of [11]. Thus, by Theorem 18 we only have to show that the pair encoding is *symbolically secure*, as defined Agrawal and Chase in [11] (see Section 5.3 for more details).

We prove it by contradiction, and using the Lemma 8 (additional structure on the bilinear map): suppose there exist $x, y \in \mathbb{Z}_p$ with $x \neq y$, and $e_1, e_2 \in \mathbb{Z}_p$ such that:

$$e_1 S_0 \left(A + R(B_1 + yB_2)\right) + e_2 S_0(B_1 + xB_2)R = AS_0 \enspace .$$

Evaluating the polynomials on $B_2 = u$ and $B_1 = -xu$, for an arbitrary $u \in \mathbb{Z}_p^*$, we obtain: $e_1 S_0(A + u(y-x)R) = AS_0$. Then, using the rule $\mathsf{extr\text{-}coeff}$ on $S_0 R$, we obtain $(y - x) = 0$, which contradicts $x \neq y$. □

5.4.2 Inner-Product Encryption (IPE)

Inner-Product Encryption (IPE) generalizes IBE, and captures useful classes of predicates, such as CNF and DNF formulas, or predicates that can expressed as polynomials (see [138] for more details).

For IPE, we have $\mathcal{X} = \mathcal{Y} = \mathbb{Z}_p^\ell$ and for any $z \in \mathbb{Z}_p^*$, the predicate P_z is defined as: $\mathsf{P}_z(\boldsymbol{x}, \boldsymbol{y}) = 1$ iff $\boldsymbol{x}^\top \boldsymbol{y} = z$.

5.4.2 IPE 1 [138]

IPE 1 is a prime-order version of [138].

$$\boldsymbol{B} := (U, \boldsymbol{V}),\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := R \qquad (n = 1 + \ell, w = 0, m = 1)$$

$$\mathsf{sE}(x) := (S_0, S_0(U\boldsymbol{x} + \boldsymbol{V})) \qquad \mathsf{rE}(y) := (R, A + R(Uz + \boldsymbol{V}^\top \boldsymbol{y}))$$

$$\mathsf{Pair}(\boldsymbol{x}, \boldsymbol{y}) := \begin{pmatrix} 0 & 1 \\ -\boldsymbol{y} & \mathbf{0}_\ell \end{pmatrix}$$

Via our framework described in Figure 5.2, gives an IPE that is twice shorter than the already existing prime-order version of [138], namely [164]. This is expected because we prove generic security, while the cited works prove security in the standard model.

Proof of symbolic security. The underlying pair encoding of IPE 1 falls under the definition of [11]. Thus, by Theorem 18, we only have to show that the pair encoding is symbolically secure (see in Section 5.3). We prove it by contradiction and using the Lemma 8: suppose there exist $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_p^\ell$ with $\boldsymbol{x}^\top \boldsymbol{y} \neq z$, $e_1 \in \mathbb{Z}_p$ and $\boldsymbol{e}_2 \in \mathbb{Z}_p^\ell$ such that:

$$e_1 S_0 \left(A + R(Uz + \boldsymbol{V}^\top \boldsymbol{y})\right) + S_0(U\boldsymbol{x} + \boldsymbol{V})^\top \boldsymbol{e}_2 R = AS_0 \ .$$

Evaluating the polynomials on $U = u$ and $\boldsymbol{V} = -\boldsymbol{x}u$, for an arbitrary $u \in \mathbb{Z}_p^*$, we obtain $e_1 S_0(A + u(z + \boldsymbol{x}^\top \boldsymbol{y})R) = AS_0$. Now, using the rule extr-coeff on R, we obtain $(z + \boldsymbol{x}^\top \boldsymbol{y}) = 0$, which contradicts $(z + \boldsymbol{x}^\top \boldsymbol{y}) \neq 0$. □

5.4.2 IPE 2

IPE 2 is a new and shorter Inner-Product Encryption that relies on inversions in the exponent, which were not captured by previous frameworks.

$$\boldsymbol{B} := \boldsymbol{B},\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := \varnothing \quad (n = \ell, w = 0, m = 0)$$

$$\mathsf{sE}(x) := S_0(\boldsymbol{x} + \boldsymbol{B}) \qquad \mathsf{rE}(y) := A/(z + \boldsymbol{B}^\top \boldsymbol{y}) \qquad \mathsf{Pair}(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{y}$$

Proof of symbolic security. By contradiction, suppose there exist $\boldsymbol{x} \in \mathbb{Z}_p^\ell$, and $\{\boldsymbol{e}_{\boldsymbol{y}} \in \mathbb{Z}_p^\ell\}_{\boldsymbol{y} \in \mathcal{Y}_x}$ such that

$$\sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}}^{\mathsf{rf}} S_0(\boldsymbol{x} + \boldsymbol{B})^\top \boldsymbol{e}_{\boldsymbol{y}} A/(z + \boldsymbol{B}^\top \boldsymbol{y}) \sim_{\mathsf{rf}} AS_0 \ .$$

Since for all $\boldsymbol{y} \in \mathcal{Y}_x$, $\boldsymbol{x}^\top \boldsymbol{y} \neq z$, we can evaluate the above rational fraction on $\boldsymbol{B} = -\boldsymbol{x}$, to obtain: $0 \sim_{\mathsf{rf}} AS_0$, which is a contradiction. As for the proof of symbolic security of IBE 1 above, this can be handle by our automatic tool, using the rules com-den, div-split, eval-var on $\boldsymbol{B} = -\boldsymbol{x}$, and zero-prod. □

5.4.3 ABE for boolean span programs.

As in Chapter 4 (Section 4.6.2), we define (monotone) access structures using the language of (monotone) span programs [137]. They generalize IBE and IPE, by allowing to embed more complex access policies in ciphertexts or in keys. We refer to Definition 13 from Chapter 2 for the notation and details about monotonic access structures.

Large universe, Unbounded ABE. When $\mathcal{U}$ is of polynomial size, we write $\mathcal{U} := [\gamma]$, and we describe sets $\Gamma \subseteq [\gamma]$ by their characteristic vectors $\boldsymbol{x} \in \{0,1\}^\gamma$, where for all $i \in [\gamma]$, $x_i = 1$ if $i \in \Gamma$, and 0 otherwise. If an ABE supports universes $\mathcal{U}$ of exponential size, we call it *large universe*. If additionally, it does not introduce a bound on the number of attributes per ciphertext, we use the term *unbounded ABE*. For practical purposes, unbounded ABE [147] are preferable, because the setup does not introduce a bound on the number of attributes per ciphertext, and they allow for more versatility since any bit string (once hashed into $\mathbb{Z}_p$) can be used as an attribute.

5.4.3 KP-ABE [116]

Considering security in the Generic Group Model, we prove the *adaptive security* of the KP-ABE from [116], arguably one of the most efficient KP-ABE, while [116] proved its *selective security* based on the DBDH assumption.

Here, $\mathcal{U} := [\gamma]$, $\mathcal{X} := \{0,1\}^\gamma$, $\mathcal{Y} := \mathbb{Z}_p^{\ell \times \tilde{\ell}} \times ([\ell] \rightarrow [\gamma])$.

$$\boldsymbol{B} := \boldsymbol{B},\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := (R_1, \ldots, R_{\tilde{\ell}-1}) \quad (n = \gamma, w = 0, m = \tilde{\ell} - 1)$$

$\mathsf{sE}(\boldsymbol{x}) := (x_1 S_0 B_1, \ldots, x_\gamma S_0 B_\gamma)$

$\mathsf{rE}((\mathbf{M}, \rho)) := \big(\mathbf{M}_1^\top (A, \boldsymbol{R})/B_{\rho(1)}, \ldots, \mathbf{M}_\ell^\top (A, \boldsymbol{R})/B_{\rho(\ell)}\big)$

$\mathsf{Pair}(\boldsymbol{x}, (\mathbf{M}, \rho)) := \mathbf{E} \in \mathbb{Z}_p^{\gamma \times \ell}$, where for all $i \in [\gamma], j \in [\ell]$, it holds $E_{i,j} = w_j$ if $\rho(j) = i$ and 0 otherwise.

Here, coefficients w_j (for $j \in [\ell]$) correspond to the weights from Definition 13 (Chapter 2).

From this point, we use a generalized $\mathsf{extr\text{-}coeff}$ rule for the proof of *symbolic security* of our encodings. More precisely, for polynomials P_1, P_2, P_3, such that $P_2 \neq 0$, and $P_3 \notin I \backslash \{0\}$ where I is the ideal generated by P_2, we have,

$$P_1 P_2 + P_3 = 0 \text{ implies } P_1 = 0 \ \wedge \ P_3 = 0 \ .$$

Proof of symbolic security. By contradiction, suppose there exist $\boldsymbol{x} \in \{0,1\}^\gamma$, and $\{\mathbf{E}_{(\mathbf{M},\rho)} \in \mathbb{Z}_p^{\gamma \times \ell}\}_{(\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}}$ such that

$$\sum := \sum_{(\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}}^{\mathsf{rf}} \mathsf{sE}(\boldsymbol{x})^\top \mathbf{E}_{(\mathbf{M},\rho)} \mathsf{rE}((\mathbf{M},\rho))(\boldsymbol{R} \rightarrow \boldsymbol{R}_{(\mathbf{M},\rho)}) \sim_{\mathsf{rf}} A S_0 \ .$$

We write $\sum = \sum_1 +_{\mathsf{rf}} \sum_2$, where $\sum_1 \in \langle \Upsilon_1 \rangle$ and $\sum_2 \in \langle \Upsilon_1 \rangle$, with:

- $\Upsilon_1 := \left\{x_{\rho(j)} S_0 \mathbf{M}_j^\top (A, \boldsymbol{R}_{(\mathbf{M},\rho)}) : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, j \in [\ell]\right\}$
- $\Upsilon_2 := \left\{x_i S_0 \mathbf{M}_j^\top (A, \boldsymbol{R}_{(\mathbf{M},\rho)}) B_i / B_{\rho(j)} : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, j \in [\ell], i \in [\gamma], \rho(j) \neq i\right\}$

Also, let $\mathcal{V} := \{\rho(j) : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, j \in [\ell], i \in [\gamma], \rho(j) \neq i\}$. We use the rules com-den, div-split and extr-coeff on the monomial $\prod_{k \in \mathcal{V}} B_k$ in the equation $\sum_1 +_{\mathsf{rf}} \sum_2 \sim_{\mathsf{rf}} AS_0$, to obtain $\sum_1 \sim_{\mathsf{rf}} AS_0$.

Now, we write

$$\sum_1 := \sum_{\substack{(\mathbf{M},\rho)\in\mathcal{Y}_{\boldsymbol{x}} \\ j \in [\ell]}} \delta_{(\mathbf{M},\rho),j} x_{\rho(j)} S_0 \mathbf{M}_j^\top (A, \boldsymbol{R}_{(\mathbf{M},\rho)})$$

and for all $(\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}$, we evaluate the equation $\sum_1 \sim_{\mathsf{rf}} AS_0$ on $A = 0$, $S_0 = 1$, and $\boldsymbol{R}_{(\widetilde{\mathbf{M}},\widetilde{\rho})} = \mathbf{0}_{\widetilde{\ell}-1}$ for all $(\widetilde{\mathbf{M}},\widetilde{\rho}) \in \mathcal{Y}_{\boldsymbol{x}} \backslash \{(\mathbf{M},\rho)\}$, to obtain,

$$\forall (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}} \ : \ \sum_{j \in [\ell]} \delta_{(\mathbf{M},\rho),j} x_{\rho(j)} \mathbf{M}_j^\top (0, \boldsymbol{R}_{(\mathbf{M},\rho)}) \sim_{\mathsf{rf}} 0 \ . \tag{5.4}$$

Also, we evaluate the equation $\sum_1 \sim_{\mathsf{rf}} AS_0$ on $A = 1$, $S_0 = 1$, and $\boldsymbol{R}_{(\mathbf{M},\rho)} = \mathbf{0}_{\widetilde{\ell}-1}$ for all $(\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}$and using the rule non-zero-sum, there exists $(\mathbf{M}^\star,\rho^\star) \in \mathcal{Y}_{\boldsymbol{x}}$ such that

$$\sum_{j \in [\ell]} \delta_{(\mathbf{M}^\star,\rho^\star),j} x_{\rho^\star(j)} \mathbf{M}_j^{\star\top} (1, \mathbf{0}_{\widetilde{\ell}-1}) = \mu \neq 0 \tag{5.5}$$

Now, combining Equation (5.4) and (5.5) and using the rule extr-coeff on each variable of $\boldsymbol{R}_{(\mathbf{M}^\star,\rho^\star)}$, we obtain

$$\frac{1}{\mu} \sum_{j \in [\ell]} \delta_{(\mathbf{M}^\star,\rho^\star),j} x_{\rho^\star(j)} \mathbf{M}_j^{\star\top} = \mathbf{1}$$

which contradicts $\mathsf{P}(\boldsymbol{x}, (\mathbf{M}^\star,\rho^\star)) = 0$. □

5.4.3 Compact KP-ABE (cKP-ABE)

We now give a new compact KP-ABE, where the ciphertexts contain 2 group elements, regardless of the number of attributes. This is more efficient that state-of-the-art [29] for which ciphertexts contain 3 group element (although the latter is for large universe).

Here, $\mathcal{U} := [\gamma]$, $\mathcal{X} := \{0,1\}^\gamma$, $\mathcal{Y} := \mathbb{Z}_p^{\ell \times \widetilde{\ell}} \times ([\ell] \to [\gamma])$.

$$\boldsymbol{B} := \boldsymbol{B},\ \boldsymbol{S} := S_0,\ \boldsymbol{R} := \boldsymbol{R} \quad (n = \gamma, w = 0, m = \widetilde{\ell}-1)$$

$$\mathsf{sE}(\boldsymbol{x}) := (S_0 \sum_{i=1}^{\gamma} x_i B_i,\ S_0)$$

$$\mathsf{rE}((\mathbf{M},\rho)) := (k_j, k_{i,j})_{i\in[\gamma], j\in[\ell], \rho(j)\neq i},\ \text{where } k_j := \mathbf{M}_j^\top(A,\boldsymbol{R})/B_{\rho(j)},$$
$$k_{i,j} := \mathbf{M}_j^\top(A,\boldsymbol{R})B_i/B_{\rho(j)}$$

$$\mathsf{Pair}(\boldsymbol{x},(\mathbf{M},\rho)) := \mathbf{E} \in \mathbb{Z}_p^{2\times\gamma\ell}, \text{ such that } \mathsf{sE}(\boldsymbol{x})^\top \mathbf{E}\,\mathsf{rE}((\mathbf{M},\rho)) =$$
$$(S_0 \textstyle\sum_{i=1}^{\gamma} x_i B_i) \cdot_{\mathsf{rf}} \left(\sum_{j\in[\ell]}^{\mathsf{rf}} \omega_j k_j\right) +_{\mathsf{rf}} S_0 \cdot_{\mathsf{rf}} \left(\sum_{j\in[\ell], i\in[\gamma], i\neq\rho(j)}^{\mathsf{rf}} x_i \omega_j k_{i,j}\right)$$

Proof of symbolic security. By contradiction, suppose there exist $\boldsymbol{x} \in \{0,1\}^\gamma$, and $\{\mathbf{E}_{(\mathbf{M},\rho)} \in \mathbb{Z}_p^{2\times\gamma\ell}\}_{(\mathbf{M},\rho)\in\mathcal{Y}_{\boldsymbol{x}}}$ such that

$$\sum := \sum_{(\mathbf{M},\rho)\in\mathcal{Y}_{\boldsymbol{x}}}^{\mathsf{rf}} \mathsf{sE}(\boldsymbol{x})^\top \mathbf{E}_{(\mathbf{M},\rho)} \mathsf{rE}(\mathbf{M},\rho)(\boldsymbol{R} \to \boldsymbol{R}_{(\mathbf{M},\rho)}) \sim_{\mathsf{rf}} AS_0 \ .$$

For $j \in [\ell]$ and $(\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}$ we define $\alpha_j = S_0\mathbf{M}_j^\top(A, \boldsymbol{R}_{(\mathbf{M},\rho)})$. We write $\sum = \sum_1 +_{\mathsf{rf}} \sum_2 +_{\mathsf{rf}} \sum_3$, where for all $k \in [3]$, $\sum_k \in \langle \Upsilon_k \rangle$, with:

- $\Upsilon_1 := \{x_{\rho(j)} S_0 \mathbf{M}_j^\top (A, \boldsymbol{R}_{(\mathbf{M},\rho)}) : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, j \in [\ell]\}$
- $\Upsilon_2 := \{\alpha_j/B_{\rho(j)}, \alpha_j B_i/B_{\rho(j)}, \alpha_j x_i B_i B_{i'}/B_{\rho(j)} : i,i' \in [\gamma], j \in [\ell], \rho(j) \notin \{i,i'\}\}$
- $\Upsilon_3 := \{x_{\rho(j)} S_0 B_i \mathbf{M}_j^\top (A, \boldsymbol{R}_{(\mathbf{M},\rho)}) : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, i \in [\gamma], j \in [\ell]\}$

Let $\mathcal{V} := \{\rho(j) : (\mathbf{M},\rho) \in \mathcal{Y}_{\boldsymbol{x}}, j \in [\ell]\}$. We use the rules com-den, div-split and extr-coeff on the monomial $\prod_{k\in\mathcal{V}} B_k$ to obtain $\sum_1 +_{\mathsf{rf}} \sum_3 \sim_{\mathsf{rf}} AS_0$, from which we obtain $\sum_3 \sim_{\mathsf{rf}} 0$ by using the rule extr-coeff on the monomial B_i for all $i \in [\gamma]$, in the equation $\sum_1 +_{\mathsf{rf}} \sum_3 \sim_{\mathsf{rf}} AS_0$. Therefore, $\sum_1 \sim_{\mathsf{rf}} AS_0$. The rest of the proof goes exactly as for the proof of the KP-ABE [116]. (See the previous encoding for further details.) □

5.4.3 Unbounded KP-ABE (uKP-ABE)

Then, we give an unbounded KP-ABE that improves upon [171] (which is proved *selectively secure* under a q-type assumption), and thereby gives the most efficient unbounded KP-ABE to the best of our knowledge (see the table Figure 5.1 for a precise comparison).

Here, $\mathcal{U} := \mathbb{Z}_p$, $\mathcal{X} := \{\Gamma \subseteq \mathbb{Z}_p\}$, $\mathcal{Y} := \mathbb{Z}_p^{\ell \times \widetilde{\ell}} \times ([\ell] \to \mathbb{Z}_p)$.

$$\boldsymbol{B} := (B_1, B_2),\ \boldsymbol{S} := (S_0, \bar{S}_i)_{i \in \Gamma},\ \boldsymbol{R} := \boldsymbol{R} \quad (n = 2, w = |\Gamma|, m = \widetilde{\ell} - 1)$$

$\mathsf{sE}(\Gamma) := \big(\bar{S}_i(B_1 + iB_2),\ S_0 - \bar{S}_i\big)_{i \in \Gamma}$

$\mathsf{rE}((\mathbf{M}, \rho)) := \big(\mathbf{M}_j^\top(A, \boldsymbol{R})/(B_1 + \rho(j)B_2),\ \mathbf{M}_j^\top(A, \boldsymbol{R})\big)_{j \in [\ell]}$

$\mathsf{Pair}(\Gamma, (\mathbf{M}, \rho)) := \begin{pmatrix} \mathbf{E} & \mathbf{0}_{|\Gamma|,\ell} \\ \mathbf{0}_{|\Gamma|,\ell} & \mathbf{E} \end{pmatrix} \in \mathbb{Z}_p^{2|\Gamma| \times 2\ell}$ where for all $i \in \Gamma$, $j \in [\ell]$, the element of the row associated to i in $\mathbf{E}$, in position j equals w_j if $i = \rho(j)$ and 0 otherwise.

Proof of symbolic security. By contradiction, suppose there exist $\Gamma \subseteq \mathbb{Z}_p$ and $\{\mathbf{E}_{(\mathbf{M},\rho)} \in \mathbb{Z}_p^{2|\Gamma| \times 2\ell}\}_{(\mathbf{M},\rho) \in \mathcal{Y}_\Gamma}$ such that

$$\sum := \sum_{(\mathbf{M},\rho) \in \mathcal{Y}_\Gamma}^{\mathsf{rf}} \mathsf{sE}(\Gamma)^\top \mathbf{E}_{(\mathbf{M},\rho)} \mathsf{rE}(\mathbf{M}, \rho)(\boldsymbol{R} \to \boldsymbol{R}_{(\mathbf{M},\rho)}) \sim_{\mathsf{rf}} AS_0 \ .$$

For $j \in [\ell]$ and $(\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma$, we define $\alpha_j = \mathbf{M}_j^\top(A, \boldsymbol{R}_{(\mathbf{M},\rho)})$. Now, we write $\sum := \sum_1 +_{\mathsf{rf}} \sum_2 +_{\mathsf{rf}} \sum_3$, where for all $k \in [3]$, $\sum_k \in \langle \Upsilon_k \rangle$, with:

- $\Upsilon_1 := \big\{(S_0 - \bar{S}_i)\alpha_j, \bar{S}_{\rho(j)}\alpha_j) : (\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma, j \in [\ell]\big\}$
- $\Upsilon_2 := \big\{\bar{S}_i(B_1 + iB_2)\alpha_j : (\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma, i \in \Gamma\big\}$
- $\Upsilon_3 := \big\{\bar{S}_i(B_1 + iB_2)\alpha_j/(B_1 + \rho(j)B_2) : (\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma, i \in \Gamma, j \in [\ell], \rho(j) \neq i\big\}$
  $\cup \big\{(S_0 - \bar{S}_i)\alpha_j/(B_1 + \rho(j)B_2) : (\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma, i \in \Gamma, j \in [\ell]\big\}$

We show that:

i) $\sum_2 \sim_{\mathsf{rf}} 0$: evaluating the equation $\sum \sim_{\mathsf{rf}} AS_0$ on $B_2 = 0$, then multiplying it by B_1, and using the rule extr-coeff on $\bar{S}_i B_1^2$ for all $i \in \Gamma$.

ii) $\sum_1 \sim_{\mathsf{rf}} AS_0$: using the rule com-den on the equation $\sum_1 +_{\mathsf{rf}} \sum_3 \sim_{\mathsf{rf}} AS_0$, then div-split, and applying the rules extr-coeff on the polynomial $B_1 + \rho(j)B_2$ sequentially for each value $\rho(j)$ such that $(\mathbf{M}, \rho) \in \mathcal{Y}_\Gamma$, and $j \in [\ell]$.

Then, for certain coefficients $\sigma_{(\mathbf{M},\rho)}$, we can write $\sum_1 := \sum_{1.1} + \sum_{1.2}$, where

$$\sum_{1.1} := \sum_{\substack{(\mathbf{M},\rho) \in \mathcal{Y}_\Gamma \\ j : \rho(j) \in \Gamma}} \sigma_{(\mathbf{M},\rho),j} S_0 \alpha_j \qquad \sum_{1.2} := \sum_{\substack{(\mathbf{M},\rho) \in \mathcal{Y}_\Gamma \\ i \in \Gamma, j \in [\ell]}} \sigma_{(\mathbf{M},\rho),j,i}(S_0 - \bar{S}_i)\alpha_j$$

We have $\sum_{1.2} \sim_{\mathsf{rf}} 0$ using $\mathsf{extr\text{-}coeff}$ on S_i for all $i \in \Gamma$.

Finally, we reach a contradiction from $\sum_{1.1} \sim_{\mathsf{rf}} AS$ exactly as in the proof of *symbolic security* of the KP-ABE [116].

5.4.3 CP-ABE

This is a new adaptively secure CP-ABE where ciphertexts are half the size of [189], while the latter was proven *selective secure* based on q-type assumptions.

Here, $\mathcal{U} := [\gamma]$, $\mathcal{X} := \mathbb{Z}_p^{\ell \times \widetilde{\ell}} \times ([\ell] \to [\gamma])$, $\mathcal{Y} := \{0,1\}^\gamma$.

$$\boldsymbol{B} := \boldsymbol{B},\ \boldsymbol{S} := (S_0, \boldsymbol{U}),\ \boldsymbol{R} := R \quad (n = \gamma, w = \widetilde{\ell}-1, m = 1)$$

$$\mathsf{sE}((\mathbf{M}, \rho)) := \left(\left(\mathbf{M}_i^\top (S_0, \boldsymbol{U}) B_{\rho(i)} \right)_{i \in [\ell]},\ S_0 \right)$$

$$\mathsf{rE}(\boldsymbol{y}) := \left((y_j R / B_j)_{j \in [\gamma]},\ A - R \right)$$

$$\mathsf{Pair}((\mathbf{M}, \rho), \boldsymbol{y}) := \begin{pmatrix} \mathbf{E} & \mathbf{0}_\ell \\ \mathbf{0}_\gamma^\top & \mathbf{E} \end{pmatrix} \in \mathbb{Z}_p^{(\ell+1) \times (\gamma+1)} \text{ where for all } i \in [\ell],\ j \in [\gamma],$$

$$E_{i,j} = w_i \text{ if } j = \rho(i) \text{ and } 0 \text{ otherwise.}$$

Proof of symbolic security. Suppose there exist $(\mathbf{M}, \rho) \in \mathbb{Z}_p^{\ell \times \widetilde{\ell}} \times ([\ell] \to [\gamma])$, and $\{\mathbf{E}_{\boldsymbol{y}} \in \mathbb{Z}_p^{(\ell+1) \times (d+1)}\}_{\boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M}, \rho)}}$ such that

$$\sum := \sum_{\boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}}^{\mathsf{rf}} \mathsf{sE}(\mathbf{M}, \rho)^\top \mathbf{E}_{\boldsymbol{y}} \mathsf{rE}(\boldsymbol{y})(R \to R_{\boldsymbol{y}}) \sim_{\mathsf{rf}} AS_0$$

We write $\sum := \sum_1 +_{\mathsf{rf}} \sum_2 +_{\mathsf{rf}} \sum_3$, where for all $k \in [3]$, $\sum_k \in \langle \Upsilon_k \rangle$, with:

- $\Upsilon_1 := \left\{ S_0(A - R_{\boldsymbol{y}}), x_{\rho(i)} R_{\boldsymbol{y}} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) : \boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell] \right\}$
- $\Upsilon_2 := \left\{ x_j S_0 R_{\boldsymbol{y}} / B_j, x_j R_{\boldsymbol{y}} B_{\rho(i)} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) / B_j : \boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}, j \in [\gamma], i \in [\ell], \rho(i) \neq j \right\}$
- $\Upsilon_3 := \left\{ (A - R_{\boldsymbol{y}}) B_{\rho(i)} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) : i \in [\ell] \right\}$

We use the rules $\mathsf{com\text{-}den}$, $\mathsf{div\text{-}split}$ and $\mathsf{extr\text{-}coeff}$ on the monomial $\prod_{k \in [\gamma]} B_k$, to obtain $\sum_1 +_{\mathsf{rf}} \sum_3 \sim_{\mathsf{rf}} AS_0$. Then, we obtain $\sum_3 \sim_{\mathsf{rf}} 0$ using the rule $\mathsf{extr\text{-}coeff}$ on the monomial $B_{\rho(i)}$ for all $i \in [\ell]$, in the equation $\sum_1 +_{\mathsf{rf}} \sum_3 \sim_{\mathsf{rf}} AS_0$. Thus, we get: $\sum_1 \sim_{\mathsf{rf}} AS_0$. Then, we write

$$\sum_1 := \sum_{\boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell]} \sigma_{\boldsymbol{y},i} x_{\rho(i)} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) R_{\boldsymbol{y}} + \sigma_{\boldsymbol{y}} S_0 (A - R_{\boldsymbol{y}})$$

and for all $\boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}$, we evaluate the equation $\sum_1 \sim_{\mathsf{rf}} AS_0$ on $A = 0$, $S_0 = 1$, and $R_{\tilde{\boldsymbol{y}}} = \mathbf{0}$ for all $\tilde{\boldsymbol{y}} \in \mathcal{Y}_{(\mathbf{M},\rho)} \backslash \{\boldsymbol{y}\}$, to obtain:

$$\sum_{i\in[\ell]} \left(\sigma_{\boldsymbol{y},i} x_{\rho(i)} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) R_{\boldsymbol{y}}\right) - \sigma_{\boldsymbol{y}} R_{\boldsymbol{y}} S_0 \sim_{\mathsf{rf}} 0 \tag{5.6}$$

Now, suppose $\sigma_{\boldsymbol{y}} \neq 0$. By evaluating Equation (5.6) on $R_{\boldsymbol{y}} = 1$, we have

$$\sum_{i\in[\ell]} \frac{\sigma_{\boldsymbol{y},i}}{\sigma_{\boldsymbol{y}}} x_{\rho(i)} \mathbf{M}_i^\top (S_0, \boldsymbol{U}) \sim_{\mathsf{rf}} S_0 \quad .$$

Then, using the rule $\mathsf{extr\text{-}coeff}$ on S_0 and all the variables in $\boldsymbol{U}$, we obtain: $\sum_{i\in[\ell]} \frac{\sigma_{\boldsymbol{y},i}}{\sigma_{\boldsymbol{y}}} x_{\rho(i)} \mathbf{M}_i^\top = \mathbf{1}$, which contradicts $\mathsf{P}((\mathbf{M},\rho),\boldsymbol{y}) = 0$. Therefore, for all $\boldsymbol{y} \in \mathcal{Y}_{(\mathbf{M},\rho)}$, we have $\sigma_{\boldsymbol{y}} = 0$. In particular, $\sum_1$ does not contain the formal variable A, which contradicts $\sum_1 \sim_{\mathsf{rf}} AS_0$ (the contradiction is obtained by applying rule $\mathsf{extr\text{-}coeff}$ on A). □

5.4.3 Unbounded CP-ABE (uCP-ABE)

Finally, we give an new, *adaptively secure*, unbounded CP-ABE where secret key size and decryption time are roughly half that state of the art [171], whose selective security is based on q-type assumptions.

Here, $\mathcal{U} := \mathbb{Z}_p$, $\mathcal{X} := \mathbb{Z}_p^{\ell\times\tilde{\ell}} \times ([\ell] \to \mathbb{Z}_p)$, $\mathcal{Y} := \{\Gamma \subseteq \mathbb{Z}_p\}$.

$$\boldsymbol{B} := (B_1, B_2, V, W),\ \boldsymbol{S} := (S_0, \boldsymbol{U}, \bar{S}_i)_{i\in[\ell]},\ \boldsymbol{R} := R$$
$$(n = 4, w = \tilde{\ell}-1+\ell, m = 1)$$

$\mathsf{sE}((\mathbf{M},\rho)) := \left(\bar{S}_i(B_1 + \rho(i)B_2),\ -V\bar{S}_i + W\mathbf{M}_i^\top(S_0,\boldsymbol{U}),\ \mathbf{M}_i^\top(S_0,\boldsymbol{U})\right)_{i\in[\ell]}$

$\mathsf{rE}(\Gamma) := (RV/(B_1 + jB_2), R,\ A - WR)_{j\in\Gamma}$

$\mathsf{Pair}((\mathbf{M},\rho),\Gamma) := \begin{pmatrix} \mathbf{E} & \mathbf{0}_\ell & \mathbf{0}_\ell \\ \mathbf{0}_{\ell,|\Gamma|} & \boldsymbol{e} & \mathbf{0}_\ell \\ \mathbf{0}_{\ell,|\Gamma|} & \mathbf{0}_\ell & \boldsymbol{e} \end{pmatrix} \in \mathbb{Z}_p^{3\ell\times(|\Gamma|+2)}$ where for all $i \in [\ell]$, $j \in [\gamma]$,

$E_{i,j} = w_i$ if $j = \rho(i)$, 0 otherwise; and $e_i = w_i$ if $\rho(i) \in \Gamma$, 0 otherwise.

Proof of symbolic security. Suppose there exist $(\mathbf{M},\rho) \in \mathbb{Z}_p^{\ell\times\tilde{\ell}} \times ([\ell] \to \mathbb{Z}_p)$, and $\{\mathbf{E}_\Gamma \in \mathbb{Z}_p^{3\ell\times(|\Gamma|+2)}\}_{\Gamma\in\mathcal{Y}_{(\mathbf{M},\rho)}}$ such that

$$\sum := \sum_{\Gamma\in\mathcal{Y}_{(\mathbf{M},\rho)}}^{\mathsf{rf}} \mathsf{sE}((\mathbf{M},\rho))^\top \mathbf{E}_\Gamma \mathsf{rE}(\Gamma)(R \to R_\Gamma) \sim_{\mathsf{rf}} AS_0$$

For $i \in [\ell]$ we define $\alpha_i = \mathbf{M}_i^\top(S_0, \boldsymbol{U})$ and $b_j = V/(B_1 + jB_2)$. We write $\sum := \sum_1 +_{\mathsf{rf}} \sum_2 +_{\mathsf{rf}} \sum_3 +_{\mathsf{rf}} \sum_4$, where for all $k \in [4]$, $\sum_k \in \langle \Upsilon_k \rangle$, with:

- $\Upsilon_1 := \left\{ \left(-V\bar{S}_i + W\alpha_i\right) R_\Gamma, \alpha_i(A - WR_\Gamma) : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell] \right\}$
  $\cup \left\{ \bar{S}_i R_\Gamma V : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell], \rho(i) \in \Gamma \right\}$
- $\Upsilon_2 := \left\{ -V\bar{S}_i + W\alpha_i(A - WR_\Gamma), \alpha_i R_\Gamma : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell] \right\}$
- $\Upsilon_3 := \left\{ \bar{S}_i(B_1 + \rho(i)B_2)R_\Gamma, \bar{S}_i(B_1 + \rho(i)B_2)(A - WR_\Gamma) : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell] \right\}$
- $\Upsilon_4 := \left\{ \bar{S}_i(B_1 + \rho(i)B_2)R_\Gamma b_j : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell], j \in \Gamma, \rho(i) \neq j \right\}$
  $\cup \left\{ \left(-V\bar{S}_i + W\alpha_i\right) R_\Gamma b_j, \alpha_i R_\Gamma b_j : \Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}, i \in [\ell], j \in \Gamma \right\}$

We show that:

- $\sum_3 \sim_{\mathsf{rf}} 0$: evaluating the equation $\sum \sim_{\mathsf{rf}} AS_0$ on $B_2 = 0$, then multiplying it by B_1, and using the rule extr-coeff on $\bar{S}_i B_1^2$.
- $\sum_1 +_{\mathsf{rf}} \sum_2 \sim_{\mathsf{rf}} AS_0$: using the rule com-den on the equation $\sum_1 +_{\mathsf{rf}} \sum_2 +_{\mathsf{rf}} \sum_4 \sim_{\mathsf{rf}} AS_0$, then div-split, and applying the rules extr-coeff on the polynomial $B_1 + jB_2$ sequentially for each value $j \in \Gamma$.
- $\sum_2 \sim_{\mathsf{rf}} AS_0$: first, we use the rule extr-coeff on WA and W^2 in the equation $\sum_1 +_{\mathsf{rf}} \sum_2 \sim_{\mathsf{rf}} 0$. Then, we evaluate the equation $\sum_1 +_{\mathsf{rf}} \sum_2 \sim_{\mathsf{rf}} AS_0$ on $W = 0$, $V = 0$, $A = 0$, and use the rule extr-coeff on R_Γ for all $\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}$. Finally, we evaluate the equation $\sum_1 +_{\mathsf{rf}} \sum_2 \sim_{\mathsf{rf}} AS_0$, on $R_\Gamma = 0$ for all $\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}$ and $A = 0$, and we use the rule extr-coeff on $V\bar{S}_i$ for all $i \in [\ell]$.

Summing up, we get: $\sum_1 \sim_{\mathsf{rf}} 0$. We write $\sum_1 := \sum_{1.1} +_{\mathsf{rf}} \sum_{1.2} +_{\mathsf{rf}} \sum_{1.3}$, where

$$\sum_{1.1} := \sum_{\substack{\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)} \\ i : \rho(i) \in \Gamma}} \nu_i^\Gamma A \mathbf{M}_i^\top(S_0, \boldsymbol{U})$$

$$\sum_{1.2} := \sum_{\substack{\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)} \\ i \in [\ell]}} \left(V\bar{S}_i + W\mathbf{M}_i^\top(S_0, \boldsymbol{U})\right)\left(\sigma_i^\Gamma R_\Gamma + \delta_i^\Gamma(A - WR_\Gamma)\right)$$

$$\sum_{1.3} := \sum_{\substack{\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)} \\ i \in [\ell]}} \mathbf{M}_i^\top(S_0, \boldsymbol{U})\left(\eta_i^\Gamma R_\Gamma + \mu_i^\Gamma(A - WR_\Gamma)\right)$$

for certain ν_i^Γ, σ_i^Γ, δ_i^Γ, η_i^Γ, μ_i^Γ, $i \in [\ell]$. We have, for all $\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}$, $i \in [\ell]$, $\sigma_i^\Gamma = 0$ (by using extr-coeff on $V\bar{S}_i R_\Gamma$) and $\delta_i^\Gamma = 0$ (by using extr-coeff on $AV\bar{S}_i$). Therefore, $\sum_{1.2} \sim_{\mathsf{rf}} 0$. Next, note that for all $\Gamma \in \mathcal{Y}_{(\mathbf{M},\rho)}$, $i \in [\ell]$, $\sum_{\Gamma,i} \eta_i^\Gamma \mathbf{M}_i^\top = \mathbf{0}^\top$ (by using extr-coeff on $R_\Gamma S_0$ and $R_\Gamma \boldsymbol{U}$) and $\sum_{\Gamma,i} \mu_i^\Gamma \mathbf{M}_i^\top = \mathbf{0}^\top$ (by using extr-coeff on $WR_\Gamma S_0$ and $WR_\Gamma \boldsymbol{U}$). That implies $\sum_{1.3} \sim_{\mathsf{rf}} 0$. Finally, we have: $\sum_{1.1} \sim_{\mathsf{rf}} AS_0$, which leads to a contradiction, as argued in the symbolic security proof of the CP-ABE. □

5.5 Automated proofs

Our main result entails that symbolic security implies security in the GGM for every RFI-ABE. Conversely, an attack against symbolic security usually represents a generic attack.[6] In this section, we present a constraint-solving method for (dis)proving symbolic security of RFI-ABE. Our method proceeds in two steps: we encode symbolic security as a constraint (written in a fragment of first-order logic); then we use proof rules for proving its (non-)validity. In this section, we present the syntax of constraints and give some proof rules. Then, we show how our method can be used to obtain a proof of symbolic security of the IBE 1 example, and to find a subtle attack. Finally, we present an implementation of the tool, and summarize some experimental results.

Broadly speaking, our algorithms combine simplification rules, which turn systems into simpler ones and case distinctions, which transform one single system into a system of equations, adding to each new system new equations that can trigger further simplifications.

Technically, the main difficulty is to reason about equations and inequations that combine rational fractions and big operators, i.e. expressions of the form $\sum_{i \in \mathcal{Q}} e_i$ or small $\prod_{i \in \mathcal{Q}} e_i$, where $\mathcal{Q}$ is a set of arbitrary size (informally, corresponding to adversary queries). Because neither symbolic computation nor algorithmic verification tools can deal with big operators (the former do not support big operators and the latter operate on a bounded state space), we develop deductive methods for solving systems of equations.

Constraints. We use a rich language of constraints that can express the existence of solutions of systems of equations and inequations between rational expressions. In order to accommodate case analysis, the language also features disjunction at top level. Thus constraints are of the form

$$\exists \boldsymbol{x_1}.\ \mathcal{C}_1 \vee \ldots \vee \exists \boldsymbol{x_k}.\ \mathcal{C}_k$$

where each $\mathcal{C}$ is a finite conjunction of (in-)equations. Due to the presence of big operators, (in-)equations may be universally quantified over arbitrary sets $\mathcal{Q}$. Therefore, and without loss of generality, each $\mathcal{C}$ is a finite conjunction of atoms of the following form:

- equation: $\mathcal{E} = 0$
- inequation: $\mathcal{E} \neq 0$
- universal equation: $\forall\, k \in \mathcal{K}.\, \mathcal{E} = 0$
- universal inequation: $\forall\, k \in \mathcal{K}.\, \mathcal{E} \neq 0$

[6] An attack against symbolic security could potentially require an exponential number of keys, and in that case, it would not correspond to an efficient attack on the scheme.

com-den	$\sum_{i \in K} \mathcal{E}_i / \mathcal{E}'_i \rightsquigarrow \frac{\sum_{i \in K} \mathcal{E}_i \times \prod_{j \in K \setminus \{i\}} \mathcal{E}'_j}{\prod_{i \in K} \mathcal{E}'_i}$
mul-split	$\mathcal{E} * \mathcal{E}' = 0 \rightsquigarrow \mathcal{E} = 0 \lor \mathcal{E}' = 0$
div-split	$\mathcal{E} / \mathcal{E}' = 0 \rightsquigarrow \mathcal{E} = 0 \land \mathcal{E}' \neq 0$
eval-var	$\mathcal{E} = 0 \rightsquigarrow \mathcal{E} = 0 \land \mathcal{E}[v \mapsto \mathcal{E}'] = 0$ for variable v and a closed (variable-free) expression $\mathcal{E}'$
extr-coeff	$\mathcal{E} * v + \mathcal{E}' = 0 \rightsquigarrow \mathcal{E} = 0 \land \mathcal{E}' = 0$ where v is a variable and $\mathcal{E}, \mathcal{E}'$ do not contain v
zero-prod	$\prod_{i \in K} \mathcal{E}_i = 0 \rightsquigarrow \exists j \in K : \mathcal{E}_j = 0$
non-zero-sum	$\sum_{i \in K} \mathcal{E}_i \neq 0 \rightsquigarrow \exists j \in K : \mathcal{E}_j \neq 0$
idx-split	$\exists i \in K. \mathcal{S}_i \rightsquigarrow (\exists i \in K \setminus \{j\}. \mathcal{S}_i) \lor \mathcal{S}_j$

Table 5.2: Selected constraint-solving rules.

where $\mathcal{E}$ ranges over expressions. The syntax of expressions is presented in Figure 5.6. Expressions $\mathcal{E}$ must be well-typed, which we enforce by declaring a type for each variable, and imposing a simple typing discipline on expressions. For example, matrices appearing in our equations are typed with a dimension and we require that these dimensions are consistent for matrix addition and multiplication. Additionally, operators like $\circ$ (pair-wise product) and diag (diagonal matrix) are enforced to be applied to vectors only (matrices with dimension $n \times 1$ or $1 \times n$).

Constraint-solving system. The constraint-solving system consists of two parts: proof rules and proof search.

Proof rules are of the form $\mathcal{D} \rightsquigarrow \mathcal{D}'$. Rules can either be simplification rules or case distinction rules. Simplification rules turn systems into simpler ones. The rules are *sound* in the sense that they preserve solutions, i.e., if the new system contains a contradictory equation like $1 = 0$ or $0 \neq 0$, it is guaranteed that the original system is unsatisfiable. Case distinctions transform one single system into several systems of equations. Soundness is guaranteed because these transformations are such that if the original system has a solution, at least one of the derived new systems will have a solution. In turn, the new equations can trigger further simplifications. Table 5.2 contains some key rules: com-den can

```
sets Q[q].

params forall i in Q: x, y_i, a_i in Zp.
vars                  S0, B, A in Zp.

     forall i in Q:     x <> y_i
  /\ sum(i in Q: S0*(B+x)*a_i*A/(B+y_i)) = A*S0.
```

$$\begin{array}{ll} \text{sets} & \mathcal{Q} = [q]. \\ \text{params} & x^*, y_i, a_i \in \mathbb{Z}_p \quad \forall i \in \mathcal{Q}. \\ \text{vars} & S_0, B, A \in \mathbb{Z}_p. \end{array}$$

$$\begin{array}{ll} & \forall i \in \mathcal{Q} : y_i \neq x^* \\ \wedge & \sum_{i \in \mathcal{Q}} a_i \frac{S_0(B + x^*)A}{B + y_i} = AS_0 \end{array}$$

Figure 5.5: Input file for the symbolic security of IBE 1 and interpretation.

be used to push the division operation outermost, by multiplying and dividing by the common denominator of the summation terms; eval-var exploits the fact that if a polynomial equation is zero, it has to be zero for every evaluation of its variables; eval-coeff uses similar ideas than the previous rule, but is applied to expressions that do not include divisions; zero-prod is semantically sound because $\mathbb{Z}_p$ is an integral domain; finally div-split, mul-split and idx-split (the last two are examples of case-distinction rules) allow to split the system into more restricted cases.

Proof search is a series of heuristics that repeatedly select and apply rules until it is shown that the system has no solution (or on the contrary is solvable). Since all rules are *sound*, the proof search algorithm is *sound*.

Example. We illustrate our constraints solving methodology with an example. Consider the system of equations in Figure 5.5, that corresponds to the symbolic security of the IBE 1 from Section 5.4.1. A solution to such a system consists of concrete values for $q \in \mathbb{N}$ and the parameters $x^*, y_i, a_i \in \mathbb{Z}_p$ for every $i \in [q]$ such that all the equations hold simultaneously treating S_0, B, A as formal variables (note that equality must be treated as the equivalence relation $\sim_{\mathsf{rf}}$ defined in Section 5.2).

The first step consists of getting rid of divisions. To do so, we apply rules com-div and div-split in this case. These rules, combined with other standard simplification rules will transform the system into:

$$\begin{aligned} & \forall i \in \mathcal{Q} : y_i - x^* \neq 0 && (5.7)\\ \wedge \quad & \prod_{i \in \mathcal{Q}} (B + y_i) = \sum_{i \in \mathcal{Q}} \Big(\prod_{j \in \mathcal{Q} \setminus \{i\}} B + y_j \Big) a_i (B + x^*) && (5.8)\\ \wedge \quad & \forall i \in \mathcal{Q} : B + y_i \neq 0 && (5.9) \end{aligned}$$

Now, the application of the rule eval-var to equation (5.8) with variable B and $\mathcal{E}' = -x^*$ will add the equation $\prod_{i \in \mathcal{Q}} (-x^* + y_i) = 0$ to the system, which can be further simplified by zero-prod. The system becomes:

$$\begin{aligned} \exists k \in \mathcal{Q} : & \\ & \forall i \in \mathcal{Q} : y_i - x^* \neq 0 \\ \wedge \quad & \prod_{i \in \mathcal{Q}} (B + y_i) = \sum_{i \in \mathcal{Q}} \Big(\prod_{j \in \mathcal{Q} \setminus \{i\}} B + y_j \Big) a_i (B + x^*) \\ \wedge \quad & \forall i \in \mathcal{Q} : B + y_i \neq 0 \\ \wedge \quad & -x^* + y_k = 0 \end{aligned}$$

which will be reduced to a contradiction after applying standard simplification rules, because the first and the fourth equations are contradictory.

Finding Attacks. Our tool can be used to find attacks for primitives that *look secure*. We present an attack (found by our tool) for the following candidate Unbounded KP-ABE$^{\triangle}$ ($\mathcal{U} := \mathbb{Z}_p$, $\mathcal{X} := \{\Gamma \subseteq \mathbb{Z}_p\}$, $\mathcal{Y} := \mathbb{Z}_p^{\ell \times \tilde{\ell}} \times ([\ell] \to \mathbb{Z}_p)$):

$$\boldsymbol{B} := B,\ \boldsymbol{S} := (S_0, \bar{S}_i)_{i \in \Gamma},\ \boldsymbol{R} := \boldsymbol{R} \qquad (n = 1, w = |\Gamma|, m = \tilde{\ell} - 1)$$

$$\mathsf{sE}(\Gamma) := \big(\bar{S}_i (B + i), : S_0 - \bar{S}_i \big)_{i \in \Gamma}$$

$$\mathsf{rE}((\mathbf{E}, \rho)) := \big(\mathbf{M}_j^{\top} (A, \boldsymbol{R}) / (B + \rho(j)),\ \mathbf{M}_j^{\top} (A, \boldsymbol{R}) \big)_{j \in [\ell]}$$

The attack works as follows: first, the challenger samples $a, b \xleftarrow{\$} \mathbb{Z}_p$ and makes $[\![b]\!]_1, [\![a]\!]_{\mathrm{t}}$ public. The adversary queries a secret key for policy $\mathbf{M} = (1, 0, \ldots, 0), \rho(1) = 3$ which is satisfied iff the set of attributes contains attribute 3. The adversary will be given $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2) = ([\![a/(b+3)]\!]_2, [\![a]\!]_2)$. Then, it picks two messages at random and sends them together with the target set for attributes $\Gamma = \{1, 2\}$. It will receive

$$\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3, \mathsf{ct}_4) = ([\![\bar{s}_1 (b+1)]\!]_1, [\![s_0 - \bar{s}_1]\!]_1, [\![\bar{s}_2 (b+2)]\!]_1, [\![s_0 - \bar{s}_2]\!]_1)$$

$$
\begin{array}{lll}
\mathcal{D} ::= \mathcal{D} \vee \mathcal{D} \mid \mathcal{S} & & \text{disjunction} \\
\mathcal{S} ::= \exists k \in \mathcal{K}.\, \mathcal{S} \mid \mathcal{C} & & \text{symbolic constraint } (k \in \mathsf{Idx}) \\
\mathcal{C} ::= \mathcal{C} \wedge \mathcal{C} \mid \forall\, k \in \mathcal{K}.\, \mathcal{C} & & \text{conjunction } (k \in \mathsf{Idx}) \\
\qquad \mid \mathcal{E} = 0 \mid \mathcal{E} \neq 0 & & \\
\mathcal{E} ::= \mathcal{E} + \mathcal{E} \mid \mathcal{E} * \mathcal{E} \mid \mathcal{E}/\mathcal{E} & & \text{expression } (k \in \mathsf{Idx}) \\
\qquad \mid \mathcal{E} \circ \mathcal{E} \mid \mathsf{diag}(\mathcal{E}) & & \\
\qquad \mid \sum\limits_{k \in \mathcal{K}} \mathcal{E} \mid \prod\limits_{k \in \mathcal{K}} \mathcal{E} & & \\
\qquad \mid -\mathcal{E} \mid \mathcal{E}^\top \mid \mathcal{M} \mid S & & \text{atom } (S \in \mathbb{Z}) \\
\mathcal{K} ::= \Gamma \mid \mathcal{K} \backslash \{k\} & & \text{index set } (k \in \mathsf{Idx}, \Gamma \in \mathsf{Set})
\end{array}
$$

We assume given sets $\mathsf{Var}, \mathsf{Par}, \mathsf{Idx}, \mathsf{Set}$ of *variables*, *parameters*, *indices* and *index sets* respectively. Matrices $\mathcal{M}$ are associated to a name $\rho \in \mathsf{Var} \cup \mathsf{Par}$, a dimension $m \times n$ $(m, n \in \mathbb{N})$ and a domain $\mathbb{Z}_p$ or $\{0, 1\} \subset \mathbb{Z}_p$. Our syntax $\circ$ stands for pair-wise product between vectors. Additionally, for a vector $v \in \mathbb{Z}_p^n$, $\mathsf{diag}(\boldsymbol{v})$ represents the null matrix in $\mathbb{Z}_p^{n \times n}$, where the main diagonal is replaced by vector $\boldsymbol{v}$.

Figure 5.6: Grammar for symbolic constraints.

where $s_0, \bar{s}_1, \bar{s}_2$ are fresh random values in $\mathbb{Z}_p$. Now, the following linear combination

$$
\begin{gathered}
-e(\mathsf{ct}_1, \mathsf{sk}_1) + 2e(\mathsf{ct}_2, \mathsf{sk}_1) - e(\mathsf{ct}_2, \mathsf{sk}_2) + \\
2e(\mathsf{ct}_3, \mathsf{sk}_1) - 2e(\mathsf{ct}_4, \mathsf{sk}_1) + 2e(\mathsf{ct}_4, \mathsf{sk}_2)
\end{gathered}
$$

equals the symmetric key $\kappa = [\![as_0]\!]_{\mathsf{t}}$ derived from encryption. This allows the adversary to fully recover the plaintext and win the experiment. That is due to the following relation:

$$
\begin{aligned}
&-\bar{S}_1 A \frac{B+1}{B+3} + 2A \frac{S_0 - \bar{S}_1}{B+3} - A(S_0 - \bar{S}_1) \\
&+ 2\bar{S}_2 A \frac{B+2}{B+3} - 2A \frac{S_0 - \bar{S}_2}{B+3} + 2A(S_0 - \bar{S}_2) \sim_{\mathsf{rf}} AS_0 \ .
\end{aligned}
$$

Scheme	Time (s)	Proof	Security
IBE 1 [191]	0.016	✓	Many-key
IBE 2 [62]	0.001	✓	One-key*
IPE 1 [138]	0.001	✓	One-key*
IPE 2 (New)	0.027	✓	Many-key
KP-ABE [116]	-	$\times$	-
Compact KP-ABE (New)	-	$\oslash$	-
Unbounded KP-ABE (New)	-	$\oslash$	-
KP-ABE [116]	-	$\times$	-
(fixed-size $\gamma = \ell = \widetilde{\ell} = 2$)	0.046	✓	One-key
(fixed-size $\gamma = \ell = \widetilde{\ell} = 3$)	1.52	✓	One-key
CP-ABE (New)	-	$\times$	-
(fixed-size $\gamma = \ell = \widetilde{\ell} = 2$)	0.212	✓	One-key
(fixed-size $\gamma = \ell = \widetilde{\ell} = 3$)	5.75	✓	One-key
Spatial Encryption [76]	0.005	✓	One-key*
Doubly Spatial Enc. [76]	0.013	✓	One-key*
KP-ABE [76]	0.256	✓	One-key*
CP-ABE [76]	0.206	✓	One-key*
NIPE,ZIPE [76]	0.003	✓	One-key*
CP-ABE for negated bf. [19]	0.084	✓	One-key*
Unbounded KP-ABE$^{\triangle}$	0.006	Attack	Insecure

Table 5.3: Encodings analyzed with our automatic tool. The first group corresponds to the encodings from this Chapter (Section 5.4). ✓ means the tool fully proved the scheme, $\times$ means it could not prove the scheme and $\oslash$ means the scheme cannot be expressed in our grammar. For every scheme we provide the level of symbolic security that was analyzed. For the schemes marked with *, one-key symbolic security is enough to achieve many-key security in the GGM (see Theorem 18).

The above attack can be easily missed when designing the primitive, since it involves a linear combination of six terms on a primitive that at a first sight, looks secure. That is an evidence of the subtleties that inversion in the exponent and the GGM may involve and it justifies the need of rigorous formalization and the design of automated methods for verification.

Implementation and case studies. We have implemented our method in a tool[7] and used the tool on several case studies. Table 5.3 summarizes the results. Our tool is able to prove automatically the symbolic security of our encodings IBE 1, IBE 2, IPE 1, IPE 2 and several encodings from the literature, like CP-ABE's and KP-ABE's from [76], or the CP-ABE for negated boolean formulas from [19]. For the most complex examples, like our CP-ABE and KP-ABE, our tool is only able to prove security for fixed-size dimensions. In some cases this is because it is hard to express the security of the full scheme with our grammar, while in others, our heuristics do not succeed in finding a proof. The tool can also find the attack against the candidate Unbounded KP-ABE$^{\triangle}$ automatically.

Comparison with previous work. We note that our tool follows the approach of the Generic Group Analyzer, gga [43] and the Generic Group Analyzer Unbounded, gga$^{\infty}$ (Chapter 3), and as the later our tool can express systems of equations depending on an unbounded number of terms, which allows to handle many security experiments of interest. Additionally, our tool is defined over a new grammar (described in Figure 5.6) and therefore, it complements previous tools and broadens the class of schemes than can be analyzed with computer-assistance. In particular, our handling of division $/$ and big products $\prod$ suffices to handle many of the primitives proposed in this Chapter.

5.6 Performance evaluation

We have implemented the schemes introduced in the previous section, as well as several Identity-Based Encryption from the literature. Our implementation uses Charm [13] for pairings with a prime-order 224-bits Miyaji, Nakabayashi and Takano elliptic curve [159]. The experiments were executed on a 2.40GHz Intel Core i7-3630QM CPU with 8GB of RAM. We use our implementations to compare the performance between the different schemes (see Figure 5.7). Expectedly, IBE 1 outperforms other constructions, highlighting the usual trade-off between efficiency and security in the standard model. We provide more details below.

For every construction, we evaluate the performance of *setup*, *encryption*, *key generation* and *decryption* on 100 executions, displaying the average time (in milliseconds). *Encryption* and *key generation* take an identity as input, which is chosen uniformly at random (in $\mathbb{Z}_p$) in every execution (this is not considered part of the execution time). We also include the IBE of Boneh and Franklin [64], arguably on of the most efficient IBE (which is proven secure in

[7]Source code and input files available at `https://github.com/miguel-ambrona/ggm-symbolic-solver`.

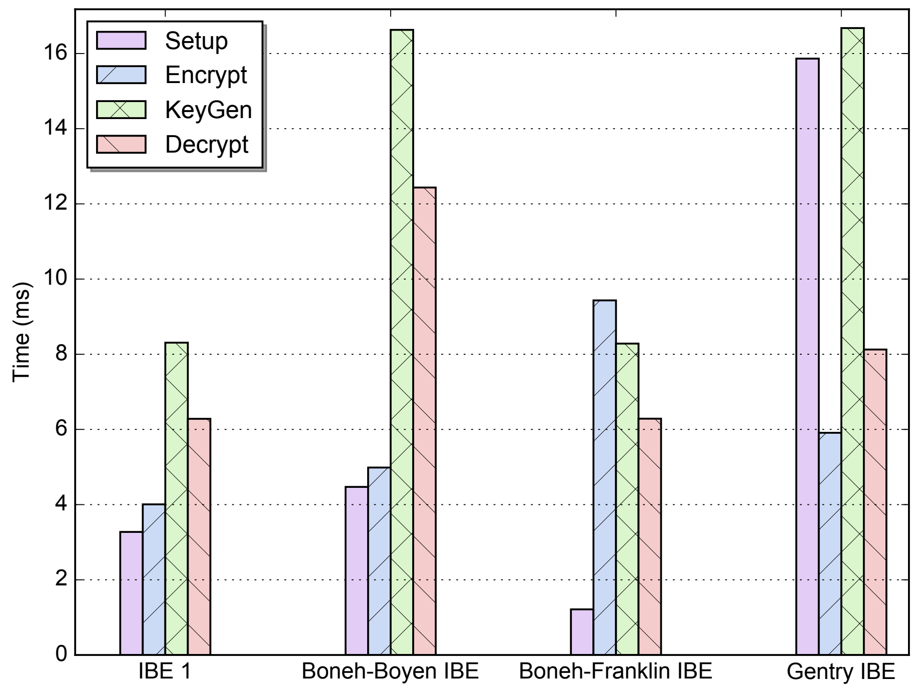


Figure 5.7: Execution time for different IBE schemes.

IBE	mpk	msk	ct	sk
IBE 1	$\mathbb{G}_1 \times \mathbb{G}_t$	$\mathbb{Z}_p^2$	$\mathbb{G}_1$	$\mathbb{G}_2$
BonBoy [61]	$\mathbb{G}_1^2 \times \mathbb{G}_t$	$\mathbb{Z}_p^3$	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$
BonFra [65]	$\mathbb{G}_1 \times (\mathbb{Z}_p \to \mathbb{G}_2)$	$\mathbb{Z}_p$	$\mathbb{G}_1$	$\mathbb{G}_2$
Gentry [112]	$\mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_t$	$\mathbb{Z}_p$	$\mathbb{G}_1 \times \mathbb{G}_t$	$\mathbb{Z}_p \times \mathbb{G}_2$

Table 5.4: Key and ciphertext sizes of IBE algorithms.

the Random Oracle Model). Note that the encryption in IBE 1 is more efficient that in [64], since contrary to the latter, IBE 1 does not require to hash into the source group $\mathbb{G}_1$, and it does not require to compute a pairing. To make the comparison more fair, in our implementation of [64], we consider a naive and efficient hashing from $\mathbb{Z}_p$ into $\mathbb{G}_1$ (performed once in encryption and once in key generation). Note, however, that for security, this hashing cannot be done naively (see [70] for instance).

5.7 Concluding remarks

In this chapter, we presented an automated method for analyzing security of ABE in the Generic Group Model. Our work significantly broadens the scope of automated analyses in the GGM, and nicely complements prior works on proving security of ABE in the standard model. We have shown how our tool can be used for proving automatically security of several schemes, including some variants of previous schemes or new schemes, and for discovering subtle attacks.

6
Automated synthesis of indifferentiability attacks

Children have a very good idea of how to distinguish between fantasies and realities. It is just they are equally interested in exploring both.

Alison Gopnik, 2009

We propose fully automated methods for finding indifferentiability attacks. Our techniques focus on universal distinguishers, for which no simulator exists. We formally define the notion of universal distinguisher and provide methods for proving the universality of a candidate distinguisher. Moreover, we show how our methods can find universal distinguishers given minimal user guidance. We implement our approach and evaluate its effectiveness.

6.1 Introduction

The framework of indifferentiability was introduced by Maurer et al. [157] in 2004. It extends the classical notion of indistinguishability and simplifies the analysis of cryptographic constructions. In particular, the indifferentiability of certain (real) cryptographic component $\mathcal{C}$ from another (ideal) component $\mathcal{R}$ guarantees that the security of a cryptosystem depending on $\mathcal{R}$ is not affected if $\mathcal{R}$ is replaced by $\mathcal{C}$. Coron et al. [84] showed how to apply indifferentiability to the field of hash functions and domain extenders. However, the scope of this indifferentiability framework is limited as showed by Ristenpart et al. [168]. They showed that the features provided by the indifferentiability can be safely applied to single-stage security games, but these techniques do not scale to

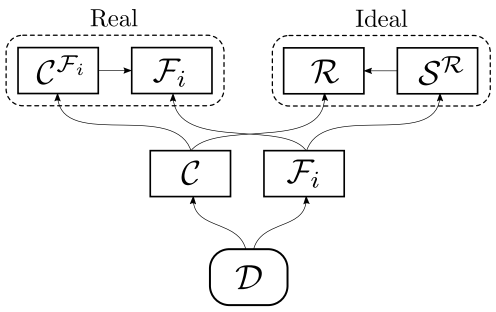


Figure 6.1: Indifferentiability security experiment. $\mathcal{C}^{\mathcal{F}_i}$ represents the real component, dependent on the small ideal components $\mathcal{F}_i$. $\mathcal{R}$ represents an ideal function with the same interface as $\mathcal{C}$, while $\mathcal{S}^{\mathcal{R}}$ is a simulator (with oracle access to $\mathcal{R}$) that simulates the small components $\mathcal{F}_i$. Distinguisher $\mathcal{D}$ must decide whether it is playing in the *real* or in the *ideal* world.

multi-stage[1] security games, since there are examples where the same techniques fail.

The notion of indifferentiability is defined with a security game played by a distinguisher $\mathcal{D}$ and an oracle system (see Figure 6.1). The distinguisher must decide whether it is playing in the real world or in the ideal world. In the real world, it has oracle access to a real component $\mathcal{C}^{\mathcal{F}_i}$ (depending on smaller ideal components) and oracle access to the individual components $\mathcal{F}_i$. On the other hand, in the ideal world, it has oracle access to an ideal random function $\mathcal{R}$ and a simulator $\mathcal{S}^{\mathcal{R}}$ that simulates the small ideal components. Note that the simulator has access to $\mathcal{R}$, but the queries made by $\mathcal{D}$ to the random oracle $\mathcal{R}$ are hidden from the simulator's view. This security game captures the notion of indifferentiability, in particular, the classical notion of indifferentiability establishes that there exists a simulator $\mathcal{S}$ such that the probability of winning the above game is negligible for every (information-theoretic) distinguisher $\mathcal{D}$. Intuitively, indifferentiability guarantees that the real component $\mathcal{C}^{\mathcal{F}_i}$ looks completely random to any observer that has no oracle access to the small components $\mathcal{F}_i$.

The standard approach of showing that certain cryptographic component is *indifferentiable* from a random component is to explicitly build a simulator and prove that any distinguisher would fail in distinguishing between the real and the ideal worlds in the presence of such a simulator. On the other hand, showing that a cryptographic component is *non-indifferentiable* from a random component is done by presenting a distinguisher that would distinguish with non-negligible probability against any possible simulator, this is known

[1]Security games where the distinguisher forgets some information during its attack.

as a *universal* distinguisher. Note that these definitions of *indifferentiability* and *non-indifferentiability* are exclusive, but not exhaustive. A cryptographic primitive could be such that for every distinguisher there exists a successful simulator and that for every simulator there exists a successful distinguisher (see Figure 6.4).

In both cases and for most of the cryptographic primitives of interest, error-prone calculations must be performed and proofs involve very subtle arguments that are hard to verify and in some occasions contain mistakes. For example, Coron et al. showed [87] that the *6-rounds Feistel network* is indifferentiable from a random permutation, while two years later, Holenstein et al. pointed out [126] that the proof from [87] was incorrect, invalidating their result[2]. These examples suggest the importance of developing general tools to assist the analysis of security proofs in the framework of indifferentiability analysis.

6.1.1 Approach

In this work, we propose, implement and evaluate automated methods for analyzing cryptographic components in the framework of indifferentiability, with special emphasis on formalizing and automatically finding universal algebraic distinguishers.

In particular, we introduce the notion of indifferentiability under universal algebraic attacks. A distinguisher is algebraic if it performs operations from a restricted class, in the spirit of the Generic Group Model [156, 160, 182] and the Algebraic Group Model [107]. Roughly, we consider distinguishers that are restricted to perform operations that are used as building blocks of the cryptographic component, thereby, if a primitive is built based on $\oplus$ of n-bit strings and *permutations* $P : \{0,1\}^n \to \{0,1\}^n$, distinguishers are allowed to use these two operations, but they are not allowed to compute other operations such as the bit-wise conjunction of two bit-strings, for example. While we do not advocate considering these restricted adversaries over non-limited ones, there are several reasons for our approach. First, our goal is to *disprove* the indifferentiability of given cryptographic components, and therefore, it is enough to find a successful (possibly restricted) distinguisher, while the absence of arbitrary distinguishers would not imply indifferentiability. Second, our approach is a first step towards the more general setting that considers distinguishers modeled as arbitrary p.p.t. machines. Third, it is not clear whether non-algebraic operations can be useful to perform an attack, because the cryptographic component is not based on such operations. In fact, we review existing attacks from the literature and show that they fall into the class of universal algebraic attacks.

[2]It remains open whether or not the 6-rounds Feistel network is indifferentiable from a random permutation.

In order to capture algebraic distinguishers, we define a *symbolic model* of indifferentiability and prove a Master Theorem which relates indifferentiability under universal algebraic attacks to symbolic indifferentiability (for a similar notion of universal algebraic attacks). The main benefit of the symbolic model is that winning conditions are now expressed in purely algebraic terms.

We develop algorithms (decision procedures) for testing the universality of a distinguisher. These algorithms leverage techniques from unification theory such as *deductibility* or *static equivalence*.

We implement and evaluate our method for several frameworks: *Feistel networks*, *Even-Mansour ciphers*, *Merkle-Damgård* or *confusion-diffusion networks*. We formalize and recover automatically many known attacks from the literature, as well as some new attacks.

6.1.2 Our contributions

First, we define a general language to express security experiments in the framework of indifferentiability. Our language is equipped with a set of simplification rules that define the equational theory of groups of *prime-characteristic*, with uninterpreted function symbols and invertible functions. This language allows to capture security experiments associated to cryptographic primitives such as *Feistel networks*, *Even-Mansour ciphers*, *confusion-diffusion networks*, etc. We then establish a Master Theorem *à la Generic Group Model* [17, 18, 43, 63, 156, 160, 182] which states that our symbolic language is negligibly close to the actual probabilistic model of computation. Roughly, this allows us to say that the conclusions derived from our restricted symbolic model can be extended without a significant loss to the standard probabilistic model. For example, to show the absence of an algorithm winning certain security experiment with significant probability, it is enough to show the absence of an algorithm winning a more restricted algebraic experiment expressed in our language.

Second, we provide a general framework to formally analyze the universality[3] of indifferentiability distinguishers. Specifically we define a grammar for distinguisher (see Figure 6.8) and provide a method to transform such a description into a system of symbolic equalities and inequalities. This system is solvable if, and only if, there exists a simulator for the distinguisher, i.e., the attack is *universal* if, and only if, the system has no solution. We note that our grammar is general enough to capture, to the best of our knowledge, all indifferentiability attacks from the literature. We develop dedicated algorithms to decide whether the system of constraints expressed in our symbolic model has a solution or not.

[3]A universal distinguisher is a p.p.t. algorithm that distinguishes between the real and the ideal scenarios with non-negligible probability against any simulator.

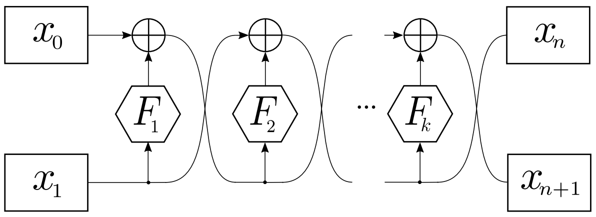


Figure 6.2: Scheme of a k-rounds Feistel network.

Third, we implement our methods and evaluate their effectiveness on actual case studies, formalizing and corroborating known indifferentiability attacks.

As an independent contribution, our framework can be used to automatically find indifferentiability attacks on various cryptographic primitives. We propose two different heuristic approaches in this direction (see Section 6.6.2). We evaluate these heuristics on primitives from the literature and we automatically find attacks for some of them. In the case of *5-rounds Feistel networks* we present a new attack, with a different structure to the one proposed by Coron et al. in [85] found with our tool.

We believe our results complement other works in the framework of indifferentiability. Our tool allows to formalize existing results (and potentially new ones) gaining confidence about their validity and extending the scope of computer assistance to this delicate and very important topic in cryptography.

6.1.3 Motivation: indifferentiability of Feistel networks

A k-rounds Feistel network [103] (Figure 6.2) is a symmetric block cipher that takes as inputs two n-bitstrings $x_0, x_1 \in \{0,1\}^n$, and returns two n-bitstrings $x_k, x_{k+1} \in \{0,1\}^n$, which we will denote by $\mathsf{Feistel}_k(x_0, x_1)$. As evidenced by Figure 6.2, for every $i \in [k]$, x_{i+1} is computed as $x_{i-1} \oplus F_i(x_i)$, where $F_i : \{0,1\}^n \to \{0,1\}^n$ are *round functions*. Feistel networks have the good property that they are invertible. They can be used to build pseudorandom permutations from (non-invertible) pseudorandom functions (as round functions). Note that the inverse can be build by running the network backwards, which can be done by evaluating the round functions in the same direction as for running the network forwards.

A long-standing open problem in cryptography is to determine the minimal number of rounds that are necessary for a Feistel network to become *"equivalent"* to a pseudorandom permutation, when all round functions are random functions. Coron, Patarin and Seurin [87] prove that six rounds are necessary, by providing explicit distinguishers for k-rounds Feistel networks, for $k \leqslant 5$. They also argue that 6 rounds suffice; however, Holenstein, Künzler and Tessaro [127] later identify a flaw in their argument and prove indifferentiability for 14 rounds. A subsequent spate of papers [89, 92, 93] shows that indifferentiability holds for lower number of rounds. Currently, it is known that $k = 8$ suffices, but it is still an open problem whether $k = 6$ or $k = 7$ suffice for indifferentiability.

Formally, the indifferentiability of k-rounds Feistel networks from a random permutation is defined in terms of a game between a distinguisher and a simulator. The distinguisher, tries to distinguish between the real and the random (or simulated) worlds. For the specific case of Feistel networks:

- In the real world, the distinguisher will be querying a real component $\mathsf{Feistel}_k$ corresponding to the Feistel network, based on round functions $\mathcal{F}_i$ which are implemented as true independent random functions.
- In the ideal world, the distinguisher will be querying a random permutation $\mathcal{R}$ and a simulator (Susan) that simulates the round functions and that also has access to $\mathcal{R}$.

We use symbol $\mathcal{C}$ to generically refer to the main oracle, i.e., to $\mathsf{Feistel}_k$ in the real world or $\mathcal{R}$ in the ideal world. In order to separate between the two worlds, the distinguisher will perform a computation c, involving procedure calls and check the validity in both worlds of a system of equalities E between outputs, and possibly intermediate computations, of c. In this article, we consider the special case of universal distinguishers, when the system E, seen as an event, has probability 1 in the real world, whereas for every simulator S, the probability of system E in the ideal world is upper bounded by a negligible function in the security parameter n.

Figure 6.3 presents two distinguishers Daisy and David for a 3-rounds Feistel network. Note that the only difference between the two programs is that their third and fourth instructions are swapped.

We claim that David is a universal distinguisher, whereas Daisy is not. To support this claim, first note that the distinguishing event holds in the real world, since, by definition,

$$\pi_1(\mathsf{Feistel}_3(x_0, x_1)) = x_1 \oplus F_2(x_0 \oplus F_1(x_1))$$

where $\pi_1(\mathsf{Feistel}_3(x_0, x_1))$ stands for the first output of $\mathsf{Feistel}_3(x_0, x_1)$. Now, consider the case of a simulator (Susan) playing against Daisy. In order to make Daisy think that she is playing in the real world, Susan should answer the queries on lines 3 and 4 in such a way that the assertion is satisfied. However, Susan can only answer the queries based on her current knowledge when the query is asked (note that Susan does not see the queries performed by the distinguisher to the main oracle $\mathcal{C}$). For instance, when F_2 is queried, Susan can use the input and output of the query to F_1. In particular, when F_2 is

Daisy:

```
1  $x_1 \leftarrow_\$ \{0,1\}^n$
2  $x_2 \leftarrow_\$ \{0,1\}^n$
3  $f_1 \leftarrow \mathrm{q}(F_1, x_1)$
4  $f_2 \leftarrow \mathrm{q}(F_2, x_2)$
5  $x_0 \leftarrow f_1 \oplus x_2$
6  $(y, y') \leftarrow \mathcal{C}(x_0, x_1)$
```

David:

```
1  $x_1 \leftarrow_\$ \{0,1\}^n$
2  $x_2 \leftarrow_\$ \{0,1\}^n$
3  $f_2 \leftarrow \mathrm{q}(F_2, x_2)$
4  $f_1 \leftarrow \mathrm{q}(F_1, x_1)$
5  $x_0 \leftarrow f_1 \oplus x_2$
6  $(y, y') \leftarrow \mathcal{C}(x_0, x_1)$
```

Distinguishing event: $y \;=\; x_1 \oplus f_2$

Figure 6.3: Distinguishers for 3-rounds Feistel.

queried, Susan can call $\mathcal{R}$ with input $(f_1 \oplus x_2, x_1)$, getting (r, r'), and return $x_1 \oplus r$.

Equivalently, the task of Susan can be captured by a constraints system combining equational constraints and deductibility constraints. These constraints capture both the behavior of the distinguisher and the distinguishing event, and are stated for an equational theory consisting of unary symbols F_1, F_2, F_3 and a binary symbol $\oplus$, with the usual axioms for exclusive or, a binary symbol $\mathcal{C}$ that outputs pairs, and two unary symbols π_1 and π_2 for projections, subject to the usual axioms for projections.

Equational constraints are of the form $e = e'$, where e and e' are either terms of the signature or unification variables α, whereas deductibility constraints are of the form $\{e_1, \dots, e_n\} \vdash \alpha$, stating that there exists an efficient algorithm to compute α from $e_1, \dots, e_n$. A solution to the constraints system is given by a mapping from unification variables to terms, so that all constraints of the system (equational and deductibility) are satisfied. The equational constraints for Daisy and David coincide:

$$\begin{array}{lll} f_1 = F_1(x_1) = \alpha_1 & f_2 = F_2(x_2) = \alpha_2 & x_0 = f_1 \oplus x_2 \\ (y, y') = \mathcal{C}(x_0, x_1) & y \;=\; x_1 \oplus f_2 & \end{array}$$

However, the deductibility constraints differ:

$$\begin{array}{rccc} \text{Daisy's constraints:} & x_1 \vdash \alpha_1 & \wedge & x_1, f_1, x_2 \vdash \alpha_2 \\ \text{David's constraints:} & x_2 \vdash \alpha_2 & \wedge & x_2, f_2, x_1 \vdash \alpha_1 \ . \end{array}$$

The task of Susan is to solve the constraints systems. Note that in the case of Daisy, the system admits a solution:

$$\alpha_2 \mapsto x_1 \oplus \pi_1(\mathcal{C}(\alpha_1 \oplus x_2, x_1)) \ .$$

Note that there is no constraint on the choice of α_1. This substitution validates the equality constraints and the deductibility constraints. On the other hand, the system has no solution in the case of David. To see it, observe that no term t depending only on x_2 can satisfy the equation $y = x_1 \oplus t$.

The above paragraph suggests that checking whether a distinguisher is universal can be reduced to solving constraints systems. In the next sections, we make this intuition formal.

6.2 Preliminaries

6.2.1 Notation

For an arbitrary binary operator $\star$ and $m \in \mathbb{N}$, we denote by $x \star \overset{m)}{\dots} \star x$ the combination through $\star$ of x, m times with itself. By [] we denote the empty list and by $[a]$ a list with single element a. Given two lists, ℓ_1, ℓ_2, their concatenation is denoted by $\ell_1 :: \ell_2$.

6.2.2 Indifferentiability security game

The security experiment for the *indifferentiability* of a real component from an ideal component is a game where a distinguisher (who has oracle access to one of the two) needs to decide whether it has access to the real or the the ideal component.

The real component $\mathcal{C}^{\mathcal{F}_i}$ depends on smaller ideal components $\mathcal{F}_i$, while the ideal component $\mathcal{R}$ does not depend on other components, but there exists a simulator $\mathcal{S}^{\mathcal{R}}$ with access to $\mathcal{R}$ that provides answers to the small components in order to pretend $\mathcal{R}$ is the real component $\mathcal{C}$. More concretely, we recall the standard definition of *indifferentiability* of $\mathcal{C}$ from ideal component $\mathcal{R}$.

Definition 33 (Indifferentiability)**.** *The construction $\mathcal{C}^{\mathcal{F}_i}$ with access to random functions $(\mathcal{F}_1, \dots, \mathcal{F}_k)$ is $(t_\mathcal{S}, q_\mathcal{S}, \epsilon)$*-indifferentiable *from an ideal component $\mathcal{R}$ if for every $q_\mathcal{D} \in \mathbb{N}$, there exists an algorithm $\mathcal{S}$, called* simulator*, running in total time $t_\mathcal{S}$ and making at most $q_\mathcal{S}$ queries to $\mathcal{R}$, such that,*

$$\left|\Pr[\mathcal{D}^{\mathcal{C}_{\mathcal{F}_i}, \mathcal{F}_i} = 1] - \Pr[\mathcal{D}^{\mathcal{R}, \mathcal{S}^{\mathcal{R}}} = 1]\right| \leqslant \epsilon$$

for every (information-theoretic) distinguisher $\mathcal{D}$ that makes at most $q_\mathcal{D}$ queries in total to its oracles.

In general, we say that $\mathcal{C}$ is *indifferentiable* from $\mathcal{R}$ when for every $q_\mathcal{D}$ that is polynomial in the security parameter n, the construction is $(t_\mathcal{S}, q_\mathcal{S}, \epsilon)$-*indifferentiable* from $\mathcal{R}$, where $t_\mathcal{S}$ and $q_\mathcal{S}$ are some polynomials in n and ϵ is certain negligible function in n (note that $t_\mathcal{S}, q_\mathcal{S}, \epsilon$ depend on $q_\mathcal{D}$).

The standard way of arguing that certain component *is not* indifferentiable from an ideal component is to show that there exists a *universal distinguisher*, i.e., an algorithm that can distinguish between the real and the random worlds with significant probability, for every possible simulator.

Definition 34 (Universal distinguisher)**.** *Let $\mathcal{C}^{\mathcal{F}_i}$ be a cryptographic component with access to random functions $(\mathcal{F}_1, \ldots, \mathcal{F}_k)$ and let $\mathcal{R}$ be an ideal component. An oracle algorithm $\mathcal{D}$, that outputs one bit is called a $(q_\mathcal{D}, \delta)$-*universal distinguisher* against the indifferentiability of $\mathcal{C}^{\mathcal{F}_i}$ from $\mathcal{R}$, where $q_\mathcal{D}, \delta$ are real polynomials in certain variable ℓ if for every $\ell \in \mathbb{N}$ and every information-theoretic algorithm $\mathcal{S}$ making at most ℓ queries to its oracles, it holds,*

$$\left|\Pr[\mathcal{D}^{\mathcal{C}_{\mathcal{F}_i}, \mathcal{F}_i} = 1] - \Pr[\mathcal{D}^{\mathcal{R}, \mathcal{S}^{\mathcal{R}}} = 1]\right| \geqslant \frac{1}{\delta(\ell)}$$

where the number of queries made by $\mathcal{D}$ to its oracles is upper-bounded by $q_\mathcal{D}(\ell)$.

In general, we say a distinguisher is universal if there exist polynomials $(q_\mathcal{D}, \delta)$ such that the distinguisher is a $(q_\mathcal{D}, \delta)$-*universal distinguisher.*

6.2.3 The problem statement

In this work, we focus on the problem of deciding whether a distinguisher is universal or not, i.e., given the description of a distinguisher, deciding if it succeeds with significant probability in the presence of any possible simulator.

As mentioned earlier, finding a universal distinguisher is actually the standard approach of arguing that certain cryptographic component is not indifferentiable from an ideal component (such as a random function or a random permutation). However, that is not the only possible way of arguing that certain primitive does not satisfy *indifferentiability.* In Figure 6.4 we divide the set of all cryptographic components into three classes:

- *Indifferentiable components*: these components satisfy a strong guarantee of security, namely, there exists a simulator that wins the indifferentiability game against any distinguisher. As shown by Maurer et al. [157], such components can be used to replace ideal components and the security proofs (in the random oracle model) will not be affected.

- *Limbo components*: these primitives are not *indifferentiable* because there is not a single simulator that wins the indifferentiability game against any distinguisher, but on the other hand, all distinguishers fail to differentiate the real world from the ideal for certain (convenient) simulator. This condition provides them with some security guarantees, we could say that

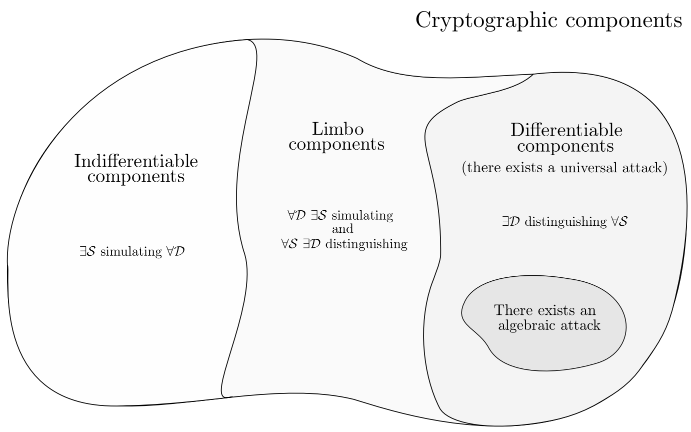

Cryptographic components
Indifferentiable components
∃S simulating ∀D
Limbo components
∀D ∃S simulating and ∀S ∃D distinguishing
Differentiable components (there exists a universal attack)
∃D distinguishing ∀S
There exists an algebraic attack

Figure 6.4: Classification of cryptographic components in the indifferentiability setting.

the output of these primitives *looks random*, but they should not be used to replace a random oracle *unconditionally*. To the best of our knowledge, no cryptographic component has been shown to belong to this class. Proving membership to this class is a very challenging goal and would require different new techniques to the ones used for arguing *indifferentiability*. Actually, the fact 6 or 7 rounds Feistel networks belonged to this class would explain the lack of progress solving the open problem for Feistel.

- *Differentiable components*: these primitives admit a *universal attack* and they *should not* be used in the place of a random oracle. The standard way of arguing that a cryptographic component is not *indifferentiable* is showing that it belongs to this class, by providing a universal distinguisher.

We note that our goal (*developing a method for deciding whether a given distinguisher is universal*) is non-trivial. That is due to the distinguisher-simulator paradigm. Unlike for disproving security in the Generic Group Model (where given the attacker's description, checking that the attacker is successful is a straightforward task), for disproving indifferentiability following this approach, although the description of the distinguisher is given, it is required to prove that such a distinguisher is successful against *all possible simulators*. Such proofs are usually done *ad hoc*, they depend on the specific attack and the primitive that is being analyzed. For example, Coron et al. provide in Section 2 of [85] an attack against the indifferentiability of the *5-rounds Feistel construction* from a random permutation and they show that the attack is universal in Lemma 2.2. Dai et al. show in Section 3 of [90] an attack against the indifferentiability of the *4-rounds iterated Even Mansour cipher* and they prove that the attack is universal in Theorem 1, supported by Lemma 2.

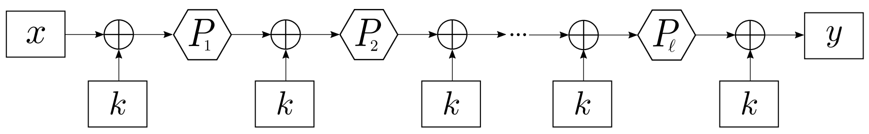


Figure 6.5: Scheme of ℓ-rounds Iterated Even-Mansour

In this work, we propose a general method for analyzing the universality of the distinguishers. Our method does not rely on the primitive that is being analyzed nor the structure of the attack, although it is restricted to *algebraic distinguishers*. To the best of our knowledge, all existing attacks from the literature are algebraic.

6.2.4 Algebraic distinguishers

An *algebraic distinguisher* is restricted to perform operations from a limited set. Roughly, they are limited to the set of operations used for defining the cryptographic component. Figure 6.6 describes the grammar for expressions that we consider. That is, our model captures all cryptographic components that can be described as maps from variables in Var to expressions of grammar $\mathcal{E}_p$. Therefore, our algebraic distinguishers will be somehow limited by this grammar too. We refer to Section 6.3 for a syntactic and semantic description of the algebraic distinguishers that we consider.

Definition 35 (Algebraic universal distinguisher)**.** *We say an algorithm is an* algebraic universal distinguisher *against the indifferentiability of $\mathcal{C}^{\mathcal{F}_i}$ from $\mathcal{R}$ if it is a* universal distinguisher *and it can be expressed as in Definition 37 (Section 6.3).*

6.2.5 Algebraic operations

The algebraic operations that we consider (described in Figure 6.6) have been chosen to capture many primitives of interest. In particular, we consider a grammar for expressions over a commutative group of prime characteristic p and, accordingly, our grammar is equipped with a set of simplification rules (see Figure 6.7).

Names from Name are used to represent constant group elements of the group. Variables from Var are used to define *placeholders* that can be filled with other expressions. Operator $+_p$ is used to represent the group law. Finally, function symbols from Fun are used to represent uninterpreted functions in expressions, and they will be used to represent random oracles or the round-functions on which the cryptographic constructions are based. When such functions are permutations, they admit an inverse.

$\mathcal{E}_p \triangleq \mid 0$	zero
$\mid c$	name, $c \in \mathsf{Name}$
$\mid x$	variable, $x \in \mathsf{Var}$
$\mid \mathcal{E}_p +_p \mathcal{E}_p$	addition
$\mid F_i(\mathcal{L})$	function evaluation, $i \in \mathbb{N}$, $F \in \mathsf{Fun}$
$\mid F_i^{-1}(\mathcal{L})$	inversion, $i \in \mathbb{N}$, $F \in \mathsf{InvFun} \subseteq \mathsf{Fun}$
$\mathcal{L} \triangleq \mathcal{E}_p \mid \mathcal{E}_p, \mathcal{L}$	non-empty list of expressions

Figure 6.6: Grammar $\mathcal{E}_p$ for expressions.

$$\begin{aligned}
(x +_p y) +_p z &= x +_p (y +_p z) && \text{associativity}\\
x +_p y &= y +_p x && \text{commutativity}\\
x +_p \overset{p)}{\ldots} +_p x &= 0 && p\text{-characteristic}\\
x +_p 0 &= x && \text{identity}\\
F_i^{-1}(\boldsymbol{x}, F_1(\boldsymbol{x},\boldsymbol{y}), \ldots, F_m(\boldsymbol{x},\boldsymbol{y})) &= \boldsymbol{y}_i && \text{inversions, } \forall i \in \{1,\ldots,m\}\\
F_i(\boldsymbol{x}, F_1^{-1}(\boldsymbol{x},\boldsymbol{y}), \ldots, F_m^{-1}(\boldsymbol{x},\boldsymbol{y})) &= \boldsymbol{y}_i
\end{aligned}$$

Figure 6.7: Simplification rules for expressions from grammar $\mathcal{E}_p$. For arbitrary expressions x, y, z and vectors of expressions $\boldsymbol{x} \in \mathcal{E}_p^k$, $\boldsymbol{y} \in \mathcal{E}_p^m$ and function $F \in \mathsf{InvFun} \subseteq \mathsf{Fun}$.

Many cryptographic constructions of interest are defined over bitstrings and therefore, can be described in our grammar for $p = 2$ (we use $\oplus$ symbol to refer to $+_2$). For example, a 2-rounds Feistel network can be represented in our grammar as follows:

$$(x_0, x_1) \mapsto (x_0 \oplus F_1(x_1),\ x_1 \oplus F_2(x_0 \oplus F_1(x_1)))$$

where $x_0, x_1 \in \mathsf{Var}$, $F_1, F_2 \in \mathsf{Fun}$. Or the 3-rounds Even Mansour (see Figure 6.5) component can be represented as follows:

$$(k, x) \mapsto P_3(P_2(P_1(x \oplus k) \oplus k) \oplus k) \oplus k$$

where $k, x \in \mathsf{Var}$, $P_1, P_2, P_3 \in \mathsf{InvFun}$.

6.3 Syntax and Semantics for algebraic distinguishers

In this section we describe the syntax for our distinguishers and provide formal semantics for distinguishers and simulators.

6.3.1 Syntax of distinguishers

We first define a system of constraints as a set of symbolic equalities and symbolic inequalities:

Definition 36 (System of constraints). *A system of constraints E is a pair $(E_=, E_{\neq})$, each of which is a set of pairs of (symbolic) expressions from $\mathcal{E}_p$.*

Roughly, an algebraic distinguisher is a computation together with a sequence of distinguishing events:

Definition 37 (Algebraic distinguisher). *An algebraic distinguisher is a pair (c, E) where c is a computation from grammar $\mathcal{C}$ (see Figure 6.8) and E is a system of constraints.*

As described in Figure 6.8, an algebraic distinguisher (c, E) is allowed (in computation c) to sample group elements uniformly from a group $\mathbb{G}$ (of prime characteristic p), invoke oracles (corresponding to both, the oracle for the main component of the indifferentiability game and the round functions on which it is based), and perform algebraic operations (restricted by the grammar of $\mathcal{E}_p$). Distinguishers are also allowed to perform probabilistic choices, denoted by $||$ (see Section 6.5.4.2 for details about this extra feature, also called *branching*). We require computation c be such that on every branching $c_1||c_2$, the variables defined inside c_1 are disjoint from those defined in c_2 (this is without loss of generality). At the end of its computation, the distinguisher will evaluate whether the distinguishing events E hold, and will terminate, outputting value 1 or 0 depending whether all events in E were satisfied or not (respectively).

Below, we formally define the semantics of computation c and give a definition for the satisfiability of E.

6.3.2 Semantics of distinguishers

The probabilistic semantics $[\![\cdot]\!]$ of a computation, map an initial memory Mem (which is a partial map from variables to group elements from a group $\mathbb{G}$) to a

$$\begin{array}{rll}
\mathcal{C} \triangleq & |\ \mathsf{cmd}; \mathcal{C} & \text{sequence of commands} \\
& |\ \mathcal{C} \mid\mid \mathcal{C} & \text{probabilistic choice} \\
& |\ \mathsf{end} & \text{finishing command} \\
\\
\mathsf{cmd} \triangleq & |\ x \leftarrow_{\$} \mathbb{G} & \text{sample random value, } x \in \mathsf{Var} \\
& |\ x \leftarrow \mathrm{q}(F, b, i, \mathcal{L}) & \text{query, } x \in \mathsf{Var}, i \in \mathbb{N}, b \in \{-1,+1\}, F \in \mathsf{Fun} \\
& |\ x := \mathcal{E}_p & \text{assignment, } x \in \mathsf{Var}
\end{array}$$

Figure 6.8: Syntax of algebraic computation for distinguishers.

distribution over final memories.

$$\begin{aligned}
\mathsf{Mem} &: \mathsf{Var} \to \mathbb{G} \\
[\![\cdot]\!]_{\mathcal{O}} &: \mathcal{C} \times \mathsf{Mem} \to D(\mathsf{Mem}) \\
[\![\cdot]\!]^{\mathsf{cmd}}_{\mathcal{O}} &: \mathcal{C} \times \mathsf{Mem} \to \mathsf{Mem}
\end{aligned}$$

$$\begin{aligned}
[\![c_1; c_2]\!]_{\mathcal{O}} &\triangleq \lambda\sigma.\, [\![c_2]\!]_{\mathcal{O}}([\![c_1]\!]^{\mathsf{cmd}}_{\mathcal{O}}(\sigma)) \\
[\![c_1 \mid\mid c_2]\!]_{\mathcal{O}} &\triangleq \lambda\sigma.\, (1/2)\cdot[\![c_1]\!]_{\mathcal{O}}(\sigma) + (1/2)\cdot[\![c_2]\!]_{\mathcal{O}}(\sigma) \\
[\![\mathsf{end}]\!]_{\mathcal{O}} &\triangleq \lambda\sigma.\, \delta(\sigma)
\end{aligned}$$

$$\begin{aligned}
[\![x \leftarrow_{\$} \mathbb{G}]\!]^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \mathrm{Unif}(\mathbb{G})] \\
[\![x \leftarrow \mathrm{q}(F, b, i, \boldsymbol{e})]\!]^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \mathcal{O}(F, b, i, \sigma(\boldsymbol{e}))] \\
[\![x := e]\!]^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \widetilde{\sigma}(e)]
\end{aligned}$$

By $[\![c]\!]_{\mathcal{O}}$ we denote the semantics of the distinguisher's computation c in the presence of the oracles $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. Both, $\mathcal{O}_1, \mathcal{O}_2$ are stateful machines that take on input queries of the form $(F, b, i, \boldsymbol{e})$ and output group elements from $\mathbb{G}$, where F is an identifier for the function, $b \in \{-1, +1\}$ indicates the direction (in case of invertible functions), i corresponds to the index of the output (in case of multi-output functions, i.e., F_i denotes $\pi_i \circ F$ where π_i is the i-th projection) and $\boldsymbol{e}$ is a list of group elements representing the input to the function. By δ we denote the *Dirac delta function*, i.e., the degenerate distribution, and by $\mathrm{Unif}(\mathbb{G})$ we denote a uniform sampling from group $\mathbb{G}$. Finally, $\widetilde{\sigma}$ is interpreted as the natural extension from σ to expressions.

Without loss of generality, we require the distinguisher be such that in every command of the form $x \mapsto \mathcal{O}(F, b, i, \sigma(\boldsymbol{e}))$ or $x := e$, expressions $\boldsymbol{e}$, e do not contain function symbols nor fresh variables (variables that have not been defined in previous calls). That first restriction enforces the distinguisher to be explicit in the order of its oracle calls (what simplifies the analysis as we will see in further sections), while the second restriction enforces the distinguisher to be well-defined. Observe that, under this restriction, $\widetilde{\sigma}(e)$ will be well-defined and it will always be a group element.

Definition 38 (Satisfying systems of constraints)**.** *Given a system of constraints $E = (E_=, E_{\neq})$ and given a map $\sigma : \mathsf{Var} \to \mathbb{G}$, we say that σ* satisfies *E (denoted by $\sigma \models E$) if*

- *for every (e_1, e_2) in $E_=$ s.t. $\mathsf{vars}(e_1, e_2) \subseteq \mathsf{dom}(\sigma)$, it holds $\sigma(e_1) =_{\mathbb{G}} \sigma(e_2)$,*
- *for every $(\widehat{e}_1, \widehat{e}_2)$ in $E_{\neq}$ s.t. $\mathsf{vars}(\widehat{e}_1, \widehat{e}_2) \subseteq \mathsf{dom}(\sigma)$, it holds $\sigma(\widehat{e}_1) \neq_{\mathbb{G}} \sigma(\widehat{e}_2)$.*

In these conditions, a distinguisher (c, E) is interpreted as an algorithm that samples a map $\sigma \leftarrow [\![c]\!]_{\mathcal{O}}(\varnothing)$ and returns 1 if $\sigma \models E$ or 0 otherwise (here, $\varnothing$ denotes the empty map).

6.3.3 Semantics of simulators

We will not give simulators a specific grammar, rather we will specify them based on the way they can respond to queries.

A query is a pair (f, x) of a function name f and an argument x. We are going to model simulators as symbolic and deterministic programs that take a list of queries and return a list of answers of equal length such that the answer to the i-th query is a symbolic combination of the arguments of the first $i-1$ queries. We can justify the restriction to deterministic simulators by the following lemma:

Lemma 9. *If there is a winning probabilistic simulator, there is also a winning deterministic simulator.*

Sketch. The main idea behind the proof is that a probabilistic simulator $\mathcal{S}$ is just a convex combination $\rho_1\mathcal{S}_1 + \rho_2\mathcal{S}_2 + \cdots + \rho_k\mathcal{S}_k$ of deterministic simulators for some $k \in \mathbb{N}$. If $\mathcal{S}$ wins with some probability p, then

$$\mathsf{p} = \mathsf{Pr}[\mathcal{S}] = \rho_1\mathsf{Pr}[\mathcal{S}_1] + \cdots + \rho_k\mathsf{Pr}[\mathcal{S}_k]$$

and therefore there exists some $i \in [k]$ such that $\mathsf{Pr}[\mathcal{S}_i] \geqslant \mathsf{p}$. □

More precisely, we will consider a simulator to be a stateful program that responds to queries from

$$\mathcal{Q} = \{(F, b, i, \boldsymbol{e}) : F \in \mathsf{Fun},\ b \in \{-1, +1\}, i \in \mathbb{N}, \boldsymbol{e} \text{ from } \mathcal{L}\}$$

Because when responding to a query, the simulator can take into account all the previous queries, we can assume that a simulator is a stateless program that, receives a list of queries (corresponding to the history) and responds to the last one:

$$\mathcal{S} : \text{List } \mathcal{Q} \to \mathcal{E}_p$$

Note that expressions returned by simulators may contain function symbols (corresponding to their own access to the random oracle).

6.3.4 Characterization of universal algebraic distinguishers

Before giving a characterization of universal algebraic distinguishers in terms of the defined semantics for distinguishers, we define the notion of *instantiating* a symbolic expression.

Definition 39 (Instantiation of expressions)**.** *Let $\mathbb{G}$ be a group of characteristic p and let e be a ground (without variables) expression of grammar $\mathcal{E}_p$. We define the instantiation of expression e in $\mathbb{G}$, denoted by $\mathsf{inst}_{\mathbb{G}}(e)$, as the group element obtained after the following steps:*

- *Let $\nu : \mathsf{Name} \to \mathbb{G}, \varphi : \mathsf{Fun} \to (\mathbb{G}^* \to \mathbb{G}^*)$ be empty maps.*
- *For every $c \in \mathsf{names}(e)$, overwrite ν with $\nu[x \mapsto \text{Unif}(\mathbb{G})]$.*
- *For every $F \in \mathsf{funs}(e)$, sample f from the set of all functions (from group elements of $\mathbb{G}$ to group elements of $\mathbb{G}$ with the adequate arity (or from the set of all permutations when $F \in \mathsf{InvFun}$), and overwrite φ with $\varphi[F \mapsto f]$.*
- *Interpret expression e by replacing every name and function symbol as dictated by ν and φ respectively. Or more precisely, return $\mathsf{eval}^{\nu,\varphi}(e)$, where*

$$\mathsf{eval}^{\nu,\varphi}(e) = \begin{cases} 0 & \textit{if } e \textit{ matches } 0 \\ \nu(c) & \textit{if } e \textit{ matches } c \in \mathsf{Name} \\ \mathsf{eval}^{\nu,\varphi}(e_1) +_{\mathbb{G}} \mathsf{eval}^{\nu,\varphi}(e_2) & \textit{if } e \textit{ matches } e_1 +_p e_2 \\ (\pi_i \circ \varphi(F)^b \circ \mathsf{eval}^{\nu,\varphi})(\boldsymbol{e}) & \textit{if } e \textit{ matches } F_i^b(\boldsymbol{e}) \end{cases}$$

  where $+_{\mathbb{G}}$ represents the group law of $\mathbb{G}$.

Very roughly, $\mathsf{inst}_{\mathbb{G}}$ of a ground expression is the group element resulting of replacing every name by a uniformly random group element and interpreting every function as a random function. Note that the *instantiation of an expression* can be efficiently implemented by lazy sampling.

We naturally extend the notion of instantiation of an expression to the instantiation of a symbolic oracle $\mathcal{O}$ in $\mathbb{G}$, denoted by $\widetilde{\mathcal{O}}$, which corresponds to the oracle that, instead of returning expressions from grammar $\mathcal{E}_p$, returns group elements by applying $\mathsf{inst}_{\mathbb{G}}$ in a systematic and consistent way.

Definition 40 (Characterization of universal algebraic distinguishers)**.** *A pair (c, E) represents a $(q_{\mathcal{D}}, \delta)$-universal distinguisher if for every symbolic simulator $\mathcal{S}$ (making ℓ queries to its oracles),*

$$\left|\mathsf{Pr}[m \leftarrow [\![c]\!]_{\tilde{\mathcal{C}},\tilde{\mathcal{F}}}(\varnothing)\ :\ m \models E] - \mathsf{Pr}[m \leftarrow [\![c]\!]_{\tilde{\mathcal{R}},\tilde{\mathcal{S}}}(\varnothing)\ :\ m \models E]\right| > \frac{1}{\delta(l)}$$

where $q_{\mathcal{D}}$ is the number of oracle calls in computation c.

In general, a distinguisher is said to be a *universal (probabilistic) distinguisher* if it is a $(q_{\mathcal{D}}, \delta)$-universal distinguisher for certain polynomial δ.

6.4 Symbolic Model and Master Theorem

We define a *symbolic model* in which the security experiment is purely deterministic. The main difference between this model and the previous *probabilistic model* is that in this experiment everything is expressed in terms of symbolic expressions and not group elements.

Expressions from grammar $\mathcal{E}_p$ (Figure 6.6) combined with simplification rules from Figure 6.7 define an equational theory, $\mathcal{T}$. We say two expressions, e_1, e_2, are equivalent in $\mathcal{T}$, denoted by $e_1 =_{\mathcal{T}} e_2$ if one can be rewritten to the other by using the rules from Figure 6.7. In order to describe our symbolic model, we start by defining *symbolic semantics* for distinguishers' programs.

6.4.1 Symbolic semantics of distinguishers

The symbolic semantics $(\!|c|\!)$, of a computation c, map an initial *symbolic memory* to a list of pairs of weights and final symbolic memories. In this case, a symbolic memory is a map from variables to symbolic expressions (from grammar $\mathcal{E}_p$). We assume without loss of generality that, for every variable $x \in \mathsf{Var}$, there exists a corresponding name $\bar{x} \in \mathsf{Name}$.

$$\begin{aligned} \mathsf{SMem} : \mathsf{Var} &\to \mathcal{E}_p \\ (\!|\cdot|\!)_{\mathcal{O}} : \mathcal{C} \times \mathsf{SMem} &\to \mathsf{List}\ (\mathbb{Q}, \mathsf{SMem}) \\ (\!|\cdot|\!)^{\mathsf{cmd}}_{\mathcal{O}} : \mathcal{C} \times \mathsf{SMem} &\to \mathsf{SMem} \end{aligned}$$

$$\begin{aligned} (\!|c_1; c_2|\!)_{\mathcal{O}} &\triangleq \lambda\sigma.\, (\!|c_2|\!)_{\mathcal{O}}((\!|c_1|\!)^{\mathsf{cmd}}_{\mathcal{O}}(\sigma)) \\ (\!|c_1 \,||\, c_2|\!)_{\mathcal{O}} &\triangleq \lambda\sigma.\, \mathrm{split}\,((\!|c_1|\!)_{\mathcal{O}}(\sigma) :: (\!|c_2|\!)_{\mathcal{O}}(\sigma)) \\ (\!|\mathsf{end}|\!)_{\mathcal{O}} &\triangleq \lambda\sigma.\, [(1, \sigma)] \end{aligned}$$

$$\begin{aligned} (\!|x \leftarrow_{\$} \mathbb{F}|\!)^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \bar{x}] \\ (\!|x \leftarrow \mathrm{q}(F, b, i, \boldsymbol{e})|\!)^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \mathcal{O}(F, b, i, \sigma(\boldsymbol{e}))] \\ (\!|x := e|\!)^{\mathsf{cmd}}_{\mathcal{O}} &\triangleq \lambda\sigma.\, \sigma[x \mapsto \sigma(e)] \end{aligned}$$

By $(\!|c|\!)_{\mathcal{O}}$ we denote the semantics of the distinguisher's computation c in the presence of the oracles $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$. Both, $\mathcal{O}_1, \mathcal{O}_2$ are stateful machines that take on input queries of the form $(F, b, i, \boldsymbol{e})$ and output symbolic expressions from $\mathcal{E}_p$. Function "split" takes a list of pairs $(\mathbb{Q}, \mathsf{SMem})$ and divides the first component of every pair by 2. More formally, split $\triangleq \mathsf{map}\ (\lambda(w, \sigma).\, (w/2, \sigma))$.

Definition 41 (Symbolically satisfying systems of constraints)**.** *Given a system of constraints $E = (E_=, E_{\neq})$ and given a map $\sigma : \mathsf{Var} \to \mathcal{E}_p$, we say that σ* symbolically satisfies *E in theory $\mathcal{T}$ (denoted by $\sigma \models_{\mathcal{T}} E$) if*

- *for every $(e_1, e_2) \in E_=$ s.t. $\mathsf{vars}(e_1, e_2) \subseteq \mathsf{dom}(\sigma)$, it holds $\sigma(e_1) =_{\mathcal{T}} \sigma(e_2)$,*
- *for every $(\hat{e}_1, \hat{e}_2)$ in $E_{\neq}$ s.t. $\mathsf{vars}(\hat{e}_1, \hat{e}_2) \subseteq \mathsf{dom}(\sigma)$, it holds $\sigma(\hat{e}_1) \neq_{\mathcal{T}} \sigma(\hat{e}_2)$.*

We define now the symbolic semantics of a distinguisher (c, E) in the presence of a symbolic oracle $\mathcal{O}$, which is a rational number in $[0, 1]$. Roughly, it represents the accumulated probability weight of the branches in c (due to probabilistic choices $||$) in which system E is symbolically satisfied.

Definition 42 (Symbolic semantics of distinguishers)**.** *We define the symbolic semantics of distinguisher (c, E) in the presence of the symbolic oracle $\mathcal{O}$, denoted as $(\!|c, E|\!)_{\mathcal{O}}$ as the rational number computed as:*

$$\sum_{\substack{(w,\sigma) \in (\!|c|\!)_{\mathcal{O}} \\ s.t.\ \sigma \models_{\mathcal{T}} E}} w\ .$$

6.4.2 Universal symbolic distinguishers

In this section we formally define the notion of universal symbolic distinguishers. Because no probabilities or group elements events are involved, but formal symbolic expressions, checking that a distinguisher is actually a universal symbolic distinguisher can be easily done (we refer to Section 6.5 for details). And thanks to Theorem 19, we can extend the absence of symbolic attacks to the absence of probabilistic attacks.

Definition 43 (Universal symbolic distinguisher)**.** *A* universal symbolic distinguisher *is a pair (c, E) of a computation and an event (a system of constraints) such that, for every symbolic simulator $\mathcal{S}$,*

$$|(\!|c, E|\!)_{\mathcal{C},\mathcal{F}} - (\!|c, E|\!)_{\mathcal{R},\mathcal{S}}| > 0 \ .$$

The main theorem we want to show is the following.

Theorem 19. *A pair (c, E) is a universal probabilistic distinguisher if, and only if, it is a universal symbolic distinguisher.*

Sketch. The Theorem follows straightforwardly from the following lemmas (Lemmas 10 and 11) by the triangle inequality and the observation that the difference $|(\!|c, E|\!)_{\mathcal{C},\mathcal{F}} - (\!|c, E|\!)_{\mathcal{R},\mathcal{S}}|$ is always a significant amount (non-negligible) when it is strictly greater than 0 (a constant, actually, because the sizes of c and E are independent of the security parameter). It is also important to note that simulators are not restricted in power, except for the number of queries to their random oracles, which is translated into a polynomial number of function symbols in their symbolic answers. That forces ℓ (from Lemma 11) to be polynomial in the security parameter and therefore (because d from both lemmas is constant), the bounds given by Lemmas 10 and 11 are negligible in the security parameter. □

Lemma 10. *For every distinguisher (c, E) making d oracle calls,*

$$\left|\mathsf{Pr}[m \leftarrow [\![c]\!]_{\tilde{\mathcal{C}},\tilde{\mathcal{F}}} \ : \ m \models E] - (\!|c, E|\!)_{\mathcal{C},\mathcal{F}}\right| \leqslant |E|\frac{d^3 + 27}{|\mathbb{G}|} \ .$$

Lemma 11. *For every distinguisher (c, E) (making d oracle calls), for every symbolic simulator $\mathcal{S}$ (making ℓ queries to its oracles),*

$$\left|\mathsf{Pr}[m \leftarrow [\![c]\!]_{\tilde{\mathcal{R}},\tilde{\mathcal{S}}} \ : \ m \models E] - (\!|c, E|\!)_{\mathcal{R},\mathcal{S}}\right| \leqslant |E|\frac{(d+\ell)^3 + 27}{|\mathbb{G}|} \ .$$

Sketch of proof of Lemmas 10 and 11. The proof follows from the observation that the following two experiments are equivalent

$$\sigma \leftarrow [\![c]\!]_{\tilde{\mathcal{O}}} \,:\, \sigma \models E \quad\equiv\quad \sigma' \leftarrow (\!|c|\!)_{\mathcal{O}} \,:\, (\lambda x.\, (\mathsf{inst}_{\mathbb{G}} \circ \sigma')\, x) \models E$$

combined with our Master Theorem (Theorem 20), that provides a bound on the probability of $\mathsf{inst}_{\mathbb{G}}$ making two (non-equivalent under $\mathcal{T}$) symbolic expressions evaluate to the same group element. □

6.4.3 Master Theorem

The fact that the symbolic model is purely deterministic simplifies the analysis of the security experiments expressed in it. However, as it has been done in other works about symbolic reasoning in cryptography [17, 18, 43, 63, 156, 160, 182], it is required to provide an evidence of the fact that the *symbolic model* is not far from the *real model*, in which instead of symbols there are group elements and probability events. The usual approach of doing so is to provide a *Master Theorem* that guarantees that, for the chosen symbolic theory, the conclusions derived in the symbolic model can be lifted to the probabilistic model.

Roughly, our Master Theorem can be interpreted as the fact that *if two symbolic expressions do not unify in theory $\mathcal{T}$, the probability that they evaluate to the same group element through procedure $\mathsf{inst}_{\mathbb{G}}$ is negligible.*

Theorem 20 (Master Theorem)**.** *Let $\mathbb{G}$ be a group of characteristic p and let e be a ground expression from grammar $\mathcal{E}_p$, containing at most d function symbols and such that $e \neq_{\mathcal{T}} 0$. It holds,*

$$\Pr\left[\mathsf{inst}_{\mathbb{G}}(e) =_{\mathbb{G}} 0_{\mathbb{G}}\right] \leqslant \frac{d^3 + 27}{|\mathbb{G}|}$$

where the probability is taken over the coins inside procedure $\mathsf{inst}_{\mathbb{G}}$ and $0_{\mathbb{G}}$ represents the identity element of $\mathbb{G}$.

Proof. See Section 6.7. □

Note that we do not apply our Master Theorem directly, but we use it to prove our Theorem 19 (see Lemmas 10 and 11), which is a more explicit statement about the relationship between symbolic and probabilistic experiments.

6.5 Analyzing the universality of distinguishers

Given the description (c, E) of a distinguisher, our goal is to show that every possible simulator fails in simulating the *real world* for the distinguisher (see Definitions 33 and 34).

To do so, we will show that (c, E) is a *universal symbolic distinguisher* and we will apply Theorem 19 to reach our goal. In this section we will show how proving that a pair (c, E) is a universal symbolic distinguisher is equivalent to showing that a system of equations has no solution. Arguing the absence of solutions of such a system is done by leveraging techniques from *unification theory* (such as *unification modulo theory*, *deductibility*, *static equivalence*, etc).

6.5.1 Basic definitions

We consider the grammar from Figure 6.6, where Name is a set of names, Var is a set of variables and Fun is a set of function symbols, with a special subset $\mathsf{InvFun} \subseteq \mathsf{Fun}$. We define by $\mathsf{terms}(\mathsf{Name}, \mathsf{Var}, \mathsf{Fun})$ the set of terms over Name, Var and Fun, i.e., the set of expressions from grammar $\mathcal{E}_p$. A substitution σ is a map from Var to $\mathsf{terms}(\mathsf{Name}, \mathsf{Var}, \mathsf{Fun})$. Given a term t and a substitution σ, we denote by $\sigma(t)$ (or simply $t\,\sigma$) the term obtained after substituting every variable $x \in \mathsf{dom}(\sigma)$ in t by the term $\sigma(x)$.

Our simplification rules from Figure 6.7 induce an equational theory that we denote by $\mathcal{T}$. In our theory, two terms or expressions e_1, e_2 are said to be *equivalent* under $\mathcal{T}$, denoted by $e_1 =_{\mathcal{T}} e_2$, if one can be rewritten to the other by the simplification rules from Figure 6.7. A term or expression is said to be *ground* if it contains no variables. A substitution is said to be *ground* if all the terms in its image are ground.

Definition 44 (Unification modulo theory)**.** *A first-order $\mathcal{T}$-unification problem $\Gamma = (\Gamma_=, \Gamma_{\neq})$ is a pair of subsets of* $\mathsf{terms}(\mathsf{Name}, \mathsf{Var}, \mathsf{Fun}) \times \mathsf{terms}(\mathsf{Name}, \mathsf{Var}, \mathsf{Fun})$. *A $\mathcal{T}$-unifier or solution to Γ is a substitution σ such that*

- *for every $(e_1, e_2) \in \Gamma_=$, $e_1\,\sigma =_{\mathcal{T}} e_2\,\sigma$, and*
- *for every $(e_1, e_2) \in \Gamma_{\neq}$, $e_1\,\sigma \neq_{\mathcal{T}} e_2\,\sigma$.*

Example 10. *Consider the theory associated to grammar $\mathcal{E}_2$, where* $\mathsf{InvFun} = \mathsf{Fun} = \{F\}$, $\mathsf{Name} = \{a, b\}$, $\mathsf{Var} = \{x, y\}$. *Let the unfication problem $\Gamma = (\Gamma_=, \Gamma_{\neq})$ be defined as*

$$\Gamma_= = \{(x, F(a, x \oplus y))\} \qquad\qquad \Gamma_{\neq} = \{(x, a)\}\,.$$

It admits unifier $\sigma = \{x \mapsto b,\ y \mapsto b \oplus F^{-1}(a, b)\}$, because $x\,\sigma = b$ and $F(a, x \oplus y)\,\sigma = F(a, F^{-1}(a, b))$ are equivalent under $\mathcal{T}$ while $x\,\sigma = b$ and $a\,\sigma = a$ are not equivalent under $\mathcal{T}$. ■

6.5.2 Deductibility and static equivalence

Deductibility and static equivalence under equational theories are two standard notions in formal security protocol analysis. Roughly, *deductibility* tries to answer to the question of whether certain term can be constructed from a set of given terms from the theory. This notion is related to *computability* and has been used in several occasions for analyzing cryptographic protocols [22, 49, 80, 82], while static equivalence has been applied to the study of *indistinguishability* and off-line guessing attacks [50, 51].

Before defining both concepts, we first introduce the notion of *frame*. Frames are used to organize sequences of terms, representing the *knowledge* of an algorithm.

Definition 45 (Frame)**.** *A* frame *is a pair* $(\tilde{n}, \sigma)$*, written as* $\nu\,\tilde{n}.\,\sigma$*, where* $\tilde{n} \subseteq \mathsf{Name}$ *is a finite set of names and* $\sigma : \mathsf{Var} \to \mathsf{terms}(\mathsf{Name}, \mathsf{Fun})$ *is a map from variables to ground terms.*

Intuitively, names in $\tilde{n}$ are used to represent fresh values generated by third parties and unavailable to the algorithm, while σ represents the algorithm's knowledge.

Deductibility. Given a frame ϕ, representing the available information to an algorithm, we consider the question of whether a given ground term e can be deduced from ϕ in our equational theory $\mathcal{T}$. This concept is expressed by $\phi \vdash e$ and it is axiomatized by the rules:

$$\frac{}{\nu\,\tilde{n}.\,\sigma \vdash c}\; c \notin \tilde{n} \qquad\qquad \frac{}{\nu\,\tilde{n}.\,\sigma \vdash e}\; \text{if } \exists x \in \mathsf{dom}(\sigma)\ :\ x\sigma = e$$

$$\frac{\phi \vdash e_1 \quad \phi \vdash e_2}{\phi \vdash e_1 +_p e_2} \qquad\qquad \frac{\phi \vdash e}{\phi \vdash \hat{e}}\; e =_{\mathcal{T}} \hat{e} \qquad\qquad \frac{}{0}$$

$$\frac{\phi \vdash e_1 \ \ldots\ \phi \vdash e_\ell}{\phi \vdash F_i^b(e_1, \ldots, e_\ell)}\; F \in \mathsf{Fun}, i \in \mathbb{N}, b \in \{-1, +1\}$$

Roughly, deductible terms from $\phi = \nu\,\tilde{n}.\,\sigma$ are those formed by the names that are not forbidden by ϕ, the terms that appear in the image of σ, and all terms that can be built from those by application of function symbols $+_p$ and Fun. This notion captures the notion of algebraic operations (see Section 6.2.4). As is common in the literature [23], we will use a characterization of the notion of deductibility.

Definition 46 (Deductibility)**.** *Let e be a ground expression and let $\nu\,\widetilde{n}.\,\sigma$ be a frame. We have $\nu\,\widetilde{n}.\,\sigma \vdash e$ if, and only if, there exists an expression ρ with $\mathsf{names}(\rho) \cap \widetilde{n} = \varnothing$ and such that $\rho\,\sigma =_{\mathcal{T}} e$.*

Such a term ρ is called a recipe for e. The *deductibility problem* consists of, given a frame ϕ and an expression e, deciding whether $\phi \vdash e$.

Example 11. *Consider our equational theory for $p = 2$ and let $\mathsf{Name} = \{a, b, d, c\}$, $\mathsf{Var} = \{x_1, x_2, x_3\}$, $\mathsf{InvFun} = \mathsf{Fun} = \{F\}$ with $F : \mathbb{G}^2 \to \mathbb{G}$. Consider expression $e = F(a, b) \oplus c$ and the frame $\phi = \nu\,\widetilde{n}.\,\sigma$ with*

$$\widetilde{n} = \{a, c, d\}$$
$$\sigma = \{x_1 \mapsto c \oplus d,\ x_2 \mapsto a \oplus c \oplus d,\ x_3 \mapsto F(a, d)\}\ .$$

It turns out that e is deductible *from ϕ. A possible recipe could be the following $\rho = x_1 \oplus F(x_1 \oplus x_2, b) \oplus F^{-1}(x_1 \oplus x_2, x_3)$.* ■

Static equivalence. Static equivalence tries to capture the notion of two frames being indistinguishable from one another. Note that deductibility is not enough for expressing such a property, because sometimes even when the two frames induce the same set of deductible terms, it is still possible to difference between them, e.g., there may exist equations that would be satisfied in one frame but not in the other.

Definition 47 (Equality modulo a frame)**.** *Let $\phi = \nu\,\widetilde{n}.\,\sigma$ be a frame. We say that expressions e, $\widehat{e}$ are* equal in the frame *ϕ under theory $\mathcal{T}$, denoted by $(e =_{\mathcal{T}} \widehat{e})_\phi$ if $(\mathsf{names}(e) \cup \mathsf{names}(\widehat{e})) \cap \widetilde{n} = \varnothing$ and $e\,\sigma =_{\mathcal{T}} \widehat{e}\,\sigma$.*

The notion of static equivalence is defined in terms of the *equality modulo a frame.* Intuitively, two frames are statically equivalent if whenever two expressions are equal modulo one of the frames, they are also equivalent modulo the other frame.

Definition 48 (Static equivalence)**.** *We say that frames $\phi_1 = \nu\,\widetilde{n}_1.\,\sigma_1$ and $\phi_2 = \nu\,\widetilde{n}_2.\,\sigma_2$ are* statically equivalent *with respect to theory $\mathcal{T}$, denoted by $\phi_1 \approx_{\mathcal{T}} \phi_2$ (or simply $\phi_1 \approx \phi_2$) if $\mathsf{dom}(\sigma_1) = \mathsf{dom}(\sigma_2)$ and for every pair $e, \widehat{e}$ of expressions we have*

$$(e =_{\mathcal{T}} \widehat{e})_{\phi_1} \Leftrightarrow (e =_{\mathcal{T}} \widehat{e})_{\phi_2}\ .$$

Example 12. *Consider our equational theory for $p = 2$ and let $\mathsf{Name} = \{a, b\}$, $\mathsf{Var} = \{x_1, x_2, x_3\}$, $\mathsf{InvFun} = \mathsf{Fun} = \{F\}$ with $F : \mathbb{G}^2 \to \mathbb{G}$. Consider the frames $\phi_1 = \nu\,\widetilde{n}.\,\sigma_1$, $\phi_2 = \nu\,\widetilde{n}.\,\sigma_2$ where*

$$\widetilde{n} = \{a, b\}$$
$$\sigma_1 = \{x_1 \mapsto F(a, b),\ x_2 \mapsto F(b, a),\ x_3 \mapsto a\}$$
$$\sigma_2 = \{x_1 \mapsto F(a, b),\ x_2 \mapsto F(b, a),\ x_3 \mapsto b\}\ .$$

Frames ϕ_1 and ϕ_2 are not *statically equivalent. Observe that, if we take expressions $e = x_3 \oplus F^{-1}(F^{-1}(x_3,\, x_1), x_2)$ and $\widehat{e} = 0$ we have $(e =_{\mathcal{T}} \widehat{e})_{\phi_1}$ but not $(e =_{\mathcal{T}} \widehat{e})_{\phi_2}$.* ■

6.5.3 Deducibility constraints

Deducibility constraints play an essential role for the symbolic analysis of security protocols. Roughly, a deducibility constraint is a problem in which a set of (possibly non-ground) terms and a target term, decide whether there exists a substitution that makes the target term deducible from the given terms. More formally,

Definition 49 (Deducibility constraint)**.** *Let $e_1, \ldots, e_\ell$ be a list of expressions and let u be an expression. Let $\widetilde{n}$ be the set of names appearing in $e_1, \ldots, e_\ell, u$ and let $x_1, \ldots, x_\ell$ be a list of variables that do not appear in $e_1, \ldots, e_\ell, u$. The* deductibility constraint *denoted by $e_1, \ldots, e_\ell \models u$ is the problem of deciding whether there exists a ground substitution ρ such that*

$$\nu\, n.\ \{x_1 \mapsto e_1\,\rho, \ldots, x_\ell \mapsto e_\ell\,\rho\} \vdash u\,\rho\ \ .$$

If such a ρ exists, it is called the *solution* of the deductibility constraint. A *deducibility constraints system* is a finite conjunction of deducibility constraints.

Example 13. *Consider our equational theory for $p = 2$ and let $\mathsf{Name} = \{a, b, c\}$, $\mathsf{Var} = \{y, x_1, x_2\}$, $\mathsf{InvFun} = \mathsf{Fun} = \{F\}$ with $F : \mathbb{G}^2 \to \mathbb{G}$. Consider the expressions $e_1 = F(b \oplus y, a)$, $e_2 = b \oplus c$ and $u = a$. The problem $e_1, e_2 \models u$ has two solutions: $\{y \mapsto b\}$ and $\{y \mapsto c\}$, because*

$$\begin{aligned} \nu\ \{a,b,c\}\,.\ \{x_1 \mapsto F(0,a),\ x_2 \mapsto b \oplus c\} \vdash a \quad &\textit{for } \rho = F^{-1}(0, x_1) \quad \textit{and} \\ \nu\ \{a,b,c\}\,.\ \{x_1 \mapsto F(c \oplus b,a),\ x_2 \mapsto b \oplus c\} \vdash a \quad &\textit{for } \rho = F^{-1}(x_2, x_1)\,. \end{aligned}$$

■

6.5.4 From distinguishers to systems of constraints

In this section, we present a method for analyzing whether a distinguisher (c, E) represents a universal symbolic attack, by checking whether a system of symbolic constraints has a solution.

Without loss of generality, we will focus on distinguishers (c, E) that are not "*fool*" in the sense that all the equations in the event E are symbolically satisfied in the real world. Note that a *fool distinguisher* can be converted into a *non-fool* one by moving the unsatisfied equations from $E_=$ to $E_{\neq}$ and vice versa. Also note that if a distinguisher (c, E) is *non-fool*, it holds:

$$(\!|c, E|\!)_{\mathcal{C},\mathcal{F}} = 1$$

and therefore, any simulator $\mathcal{S}$ that tries to prevent (c, E) from being a symbolic attack, must respond to the queries in such a way that $(\!|c, E|\!)_{\mathcal{R},\mathcal{S}} = 1$, i.e., all the equations in E must be symbolically satisfied. Equivalently, if for all symbolic simulators, it is impossible to symbolically satisfy all the equations in E, we have $(\!|c, E|\!)_{\mathcal{R},\mathcal{S}} < 1$, and thus, (c, E) is a universal symbolic distinguisher.

Note that, in order to show that (c, E) is an attack for all possible simulators, we can consider a simulator who *knows* the distinguisher's strategy, who has the distinguisher's code *hard-wired*.

6.5.4.1 Non-branching distinguishers

We say a distinguisher (c, E) is *non-branching* if computation c does not include the symbol $||$. Note that non-branching symbolic distinguishers are completely deterministic, while branching distinguishers are not.

If the simulator knows the distinguisher's code and the distinguisher (c, E) is non-branching, without loss of generality we can see the simulator as a deterministic machine that on input a frame ϕ returns a symbolic expression e, with $\phi \vdash e$. The simulator's answers must be such that at the end of computation c, all the equations from E are satisfied. In this way, every distinguisher (c, E) can be translated into a system of constraints (Γ, Υ), where $\Gamma = (\Gamma_=, \Gamma_{\neq})$ is a unification problem on variables $\{\alpha_1, \ldots, \alpha_\ell\}$ and $\Upsilon = \{\boldsymbol{e}_1 \models \alpha_1, \ldots, \boldsymbol{e}_\ell \models \alpha_\ell\}$ is a deducibility constraints system. A solution to the system is a substitution σ that is both a solution to problem Γ and a solution to the deducibility constraints system Υ.

In Figure 6.9 we formally describe how to obtain this system of equations from the distinguisher's description (c, E). We assume that, for every variable $x \in \mathsf{Var}$ there exists a corresponding name $\bar{x} \in \mathsf{Name}$ and $\circlearrowleft_{\mathsf{Var}}$ is a procedure that on every call returns a *fresh variable*. Type SMem corresponds to the symbolic memory used to define the symbolic semantics of distinguishers, $\mathsf{SMem} : \mathsf{Var} \to \mathcal{E}_p$. Type K is a list of expressions ($\mathcal{E}_p$ list), while type DC stands for deductibility constraints system. Procedure $\mathsf{trans}_{\sigma,\Upsilon}$ computes a symbolic memory together with a system of deducibility constraints. Roughly, every query to the simulator's oracle produces a fresh variable that is added to the system in form of deducibility constraint, i.e., the value that this variable can take must be computed from the current knowledge of expressions (that is captured by K). The system of constraints (Γ, Υ) associated to distinguisher (c, E) is built as follows:

$$(\sigma, \Upsilon) \leftarrow \mathsf{trans}_{\varnothing,\varnothing}(c)(\varnothing, \varnothing); \;\; \Gamma = (\sigma(E_=), \sigma(E_{\neq})); \;\; \mathrm{return}(\Gamma, \Upsilon) \;\; .$$

Example 14. *Let c be the computation induced by the program associated to Daisy from Figure 6.3, where the distinguishing event is $E_= = \{(y, x_1 \oplus f_2)\}$,*

$$\begin{aligned}
\mathsf{trans}_{\sigma,\Upsilon}(\cdot) &: \mathcal{C} \times \mathsf{K} \to (\mathsf{SMem},\ \mathsf{DC})\\
\mathsf{trans}^{\mathsf{cmd}}_{\Upsilon}(\cdot) &: \mathcal{C} \times (\mathsf{SMem},\ \mathsf{K}) \to (\mathsf{SMem},\ \mathsf{K}, \mathsf{DC})
\end{aligned}$$

$$\begin{aligned}
\mathsf{trans}_{\sigma,\Upsilon}(c_1; c_2) &\triangleq \lambda \boldsymbol{t}.\, \mathsf{trans}_{\widehat{\sigma},\widehat{\Upsilon}}(c_2)(\widehat{\boldsymbol{t}}),\ \text{ where } (\widehat{\sigma}, \widehat{\boldsymbol{t}}, \widehat{\Upsilon}) = \mathsf{trans}^{\mathsf{cmd}}_{\Upsilon}(c_1)(\boldsymbol{t})\\
\mathsf{trans}_{\sigma,\Upsilon}(c_1 \,||\, c_2) &\triangleq \lambda \boldsymbol{t}.\, \mathsf{trans}_{\sigma,\Upsilon}(c_1)(\boldsymbol{t}) \sqcup \mathsf{trans}_{\sigma,\Upsilon}(c_2)(\boldsymbol{t})\\
\mathsf{trans}_{\sigma,\Upsilon}(\mathsf{end}) &\triangleq \lambda \boldsymbol{t}.\, (\sigma, \Upsilon)
\end{aligned}$$

$$\begin{aligned}
\mathsf{trans}^{\mathsf{cmd}}_{\Upsilon}(x \leftarrow_{\$} \mathbb{F}) &\triangleq \lambda(\sigma, \boldsymbol{t}).\, (\sigma[x \mapsto \bar{x}], \boldsymbol{t}, \Upsilon)\\
\mathsf{trans}^{\mathsf{cmd}}_{\Upsilon}(x \leftarrow \mathrm{q}(F, b, i, \boldsymbol{e})) &\triangleq \lambda(\sigma, \boldsymbol{t}).\, (\sigma[x \mapsto y], \boldsymbol{t} :: \boldsymbol{e}), \Upsilon \cup \{\boldsymbol{t} :: \boldsymbol{e} \models y\})\,,\ y \leftarrow \circlearrowleft_{\mathsf{Var}}\\
\mathsf{trans}^{\mathsf{cmd}}_{\Upsilon}(x := e) &\triangleq \lambda(\sigma, \boldsymbol{t}).\, (\sigma[x \mapsto \sigma(e)], \boldsymbol{t}, \Upsilon)
\end{aligned}$$

Figure 6.9: Translation from distinguisher's description to system of constraints. Where $\sqcup$ is defined as $(\sigma_1, \Upsilon_1) \sqcup (\sigma_2, \Upsilon_2) \triangleq (\sigma_1 \cup \sigma_2,\ \Upsilon_1 \cup \Upsilon_2)$.

$E_{\neq} = \varnothing$. *After running procedure* trans *we would obtain* (σ, Υ)*, being*

$$\begin{aligned}
\sigma &= \big\{x_1 \mapsto \bar{x}_1,\ x_2 \mapsto \bar{x}_2,\ f_1 \mapsto \alpha_1,\ f_2 \mapsto \alpha_2,\\
&\qquad x_0 \mapsto \alpha_1 \oplus \bar{x}_2,\ y \mapsto \mathcal{R}_1(\alpha_1 \oplus \bar{x}_2,\ \bar{x}_1),\ y' \mapsto \mathcal{R}_2(\alpha_1 \oplus \bar{x}_2,\ \bar{x}_1)\big\}\\
\Upsilon &= \big\{\bar{x}_1 \models \alpha_1,\ \bar{x}_1, \bar{x}_2 \models \alpha_2\big\}
\end{aligned}$$

where $\bar{x}_1, \bar{x}_2 \in \mathsf{Name}$, $\alpha_1, \alpha_2 \in \mathsf{Var}$, $\mathcal{R} \in \mathsf{InvFun}$. *The unification problem* $\Gamma = (\Gamma_=, \Gamma_{\neq})$ *is obtained after applying* σ *to* E*, that is*

$$\Gamma_= = \{(\mathcal{R}_1(\alpha_1 \oplus \bar{x}_2,\ \bar{x}_1),\ \ \bar{x}_1 \oplus \alpha_2)\} \qquad \Gamma_{\neq} = \varnothing\ .$$

Observe that the system of constraints (Γ, Υ) *derived from Daisy admits a solution, namely, the unifier* $\big\{\alpha_1 \mapsto r,\ \alpha_2 \mapsto \bar{x}_1 \oplus \mathcal{R}_1(r \oplus \bar{x}_2,\ \bar{x}_1)\big\}$*, where* r *is a fresh name. The existence of a solution implies that Daisy* is not *a universal distinguisher.* ■

6.5.4.2 Branching distinguishers

In our grammar for distinguishers (see Figure 6.8) we allow probabilistic choices, syntactically denoted by $c_1 \,||\, c_2$, and interpreted as *a coin will be tossed and depending on the outcome either* c_1 *or* c_2 *will be executed.* Such a *branching* in the program can be extremely useful as evidenced by actual attacks from the literature, like the attack to the *4-rounds iterated Even Mansour cipher* from [90], in which branching plays an essential role. Intuitively, branching is helpful

Computation:

```
1   $x_0 \leftarrow_{\$} \{0,1\}^n$
2   $x_1 \leftarrow_{\$} \{0,1\}^n$
3   $x'_1 \leftarrow_{\$} \{0,1\}^n$
4   $f_1 \leftarrow \mathrm{q}(F_1, x_1)$
5   $f'_1 \leftarrow \mathrm{q}(F_1, x'_1)$
6   $x_2 := x_0 \oplus f_1$
7   $x'_2 := x_0 \oplus f'_1$
8   $f_2 \leftarrow \mathrm{q}(F_2, x_2) \;||\; f'_2 \leftarrow \mathrm{q}(F_2, x'_2)$
```

Distinguishing events:

$$\mathcal{R}_1(x_0, x_1) = x_1 \oplus f_2$$
$$\mathcal{R}_1(x_0, x'_1) = x'_1 \oplus f'_2$$

Figure 6.10: Branching attack on 3-rounds Feistel.

because it creates uncertainty on the simulator's view. Note that, without loss of generality, we are assuming that the simulator knows the distinguisher's code. However, if the distinguisher's code makes probabilistic choices, the simulator might not be able to tell which branch of the code is being executed. The ability of the simulator for distinguishing between different branches is modelled by *static equivalence.*

Example 15. *Consider the computation from Figure 6.10 and observe that line 8 contains a probabilistic choice, where the third query to the simulator may be on value x_2 or x'_2. In this case, the simulator cannot tell the difference between this two branches, because the following two frames are* statically equivalent*:*

$$\nu\,\tilde{n}.\,\{y_1 \mapsto x_1,\, y_2 \mapsto \bar{x}'_1,\, y_3 \mapsto \bar{x}_0 \oplus \bar{f}_1\} \;\approx\; \nu\,\tilde{n}.\,\{y_1 \mapsto x_1,\, y_2 \mapsto \bar{x}'_1,\, y_3 \mapsto \bar{x}_0 \oplus \bar{f}'_1\}$$

where $\bar{x}_0, \bar{x}_1, \bar{x}'_1, \bar{f}_1, \bar{f}'_1 \in \mathsf{Name}$, $y_1, y_2, y_3 \in \mathsf{Var}$ and $\tilde{n} = \{\bar{x}_0, \bar{x}_1, \bar{x}'_1\}$. This situation allows us to further limit the simulator (without loss of generality) by adding an extra constraint. Observe that the system of constraints (Γ, Υ) obtained by our procedure trans *is the following:*

$$\Gamma_= = \{(\mathcal{R}_1(\bar{x}_0,\, \bar{x}_1),\, \bar{x}_1 \oplus \alpha_3),\;\; (\mathcal{R}_1(\bar{x}_0,\, \bar{x}'_1),\, \bar{x}'_1 \oplus \alpha_4)\} \qquad \Gamma_{\neq} = \varnothing$$
$$\Upsilon = \{\bar{x}_1 \models \alpha_1,\;\; \bar{x}_1, \bar{x}'_1 \models \alpha_2,\;\; \bar{x}_1, \bar{x}'_1, \bar{x}_0 \oplus \alpha_1 \models \alpha_3,\;\; \bar{x}_1, \bar{x}'_1, \bar{x}_0 \oplus \alpha_2 \models \alpha_4\}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathsf{Var}$. Note that the problem (Γ, Υ) admits a solution, namely, the unifier

$$\{\alpha_1 \mapsto r,\; \alpha_2 \mapsto s,\; \alpha_3 \mapsto \bar{x}_1 \oplus \mathcal{R}_1(\bar{x}_0,\, \bar{x}_1),\; \alpha_4 \mapsto \bar{x}'_1 \oplus \mathcal{R}_1(\bar{x}_0,\, \bar{x}'_1)\}$$

where $r, s \in \mathsf{Name}$ are fresh names. However, as we have already advanced, we could add an extra constraint to the system to capture the simulators inability of distinguishing between the two branches. The constraint binds variables α_3 and

α_4 (corresponding to the third query in either execution) and it imposes that the production or recipe for α_3 must be identical to the one for α_4. Observe that the above unifier does not satisfy this condition, since the production for α_3 is precisely $y_1 \oplus \mathcal{R}_1(y_3 \oplus r, y_1)$, while the production for α_4 is different, namely, $y_2 \oplus \mathcal{R}_1(y_3 \oplus s, y_2)$. It turns out that in this example, no solution to the problem (Γ, Υ) is such that the additional restriction on α_3 and α_4 is satisfied. We conclude that the distinguisher described in Figure 6.10 is universal. ■

Definition 50 (System of simulating constraints)**.** *A system of simulating constraints is a triple (Γ, Υ, Φ) where $\Gamma = (\Gamma_=, \Gamma_\neq)$ is a unification problem on variables $\{\alpha_1, \dots, \alpha_\ell\}$, $\Upsilon = \{\boldsymbol{e}_1 \models \alpha_1, \dots, \boldsymbol{e}_\ell \models \alpha_\ell\}$ is a deducibility constraints system and Φ is a set of pairs of variables from $\{\alpha_1, \dots, \alpha_\ell\}$. A solution to the system (Γ, Υ, Φ) is a substitution σ such that*

- *σ is a solution to the unification problem Γ,*
- *σ is a solution to the deducibility constraints system Υ, and*
- *for every pair of variables $(\alpha_i, \alpha_j) \in \Phi$ there exists a single recipe ρ such that $(\boldsymbol{e}_i\,\sigma)\,\rho \vdash \sigma(\alpha_i)$ and $(\boldsymbol{e}_j\,\sigma)\,\rho \vdash \sigma(\alpha_j)$.*

Given a distinguisher (c, E), we denote by $\mathsf{branching}(c)$ the set of branching constraints that we can *safely* consider. We do not explicitly define the procedure of computing $\mathsf{branching}(c)$, but give an verbal description of it. At at every probabilistic choice in computation c, and on every query inside the branches (sequentially) a static equivalence problem is considered. Whenever the branches result *indistinguishable*, a new pair of variables (corresponding to those from the queries that are being compared) is added to Φ. This process continues until on some query the frames are *distinguishable*. This idea can be naturally extended to programs containing multiple probabilistic choices.

Our methods for checking whether a distinguisher (c, E) is a universal distinguisher consists of building the system of simulating constraints (Γ, Υ, Φ), obtained as

$$(\sigma, \Upsilon) \leftarrow \mathsf{trans}_{\varnothing,\varnothing}(c)(\varnothing, \varnothing); \quad \Gamma = (\sigma(E_=), \sigma(E_\neq)); \quad \Phi = \mathsf{branching}(c)$$

and checking whether it admits a solution.

6.6 Implementation and automated attacks

We have implemented a general library for indifferentiability analysis. Our library allows to solve *unification, deductibility and static equivalence problems* in the equational theory we consider, for characteristic $p = 2$, (see Figures 6.6

and 6.7), as well as checking that distinguishers expressed in our grammar (Figure 6.8) are universal. We also explore automated synthesis of indifferentiability attacks by proposing two heuristic methods for attacks search. We evaluate our methods on actual primitives from the literature.

All the experiments were executed on a 8-core machine with 2.40GHz Intel Core i7-3630QM CPU and 8GB of RAM. The code is publicly available and open source[4].

6.6.1 Case studies (proving universality of distinguishers)

In this section we present the results of applying our tool to verifying the universality of given attacks. Table 6.1 summarizes our results. On every attack, we indicate how much time our tool took to verify that it is universal. Some of the attacks come from the literature while others correspond to the examples given in this work and some attacks found with our automated attacks search (we refer to next section for details). The code of the missing attacks can be found in Section 6.8.

The first block corresponds to attacks for *Feistel networks*. The first entry is the only one in the table that does not represent an *indifferentiability* attack, but a *indistinguishability against chosen-ciphertext attack*, namely, the distinguisher is not allowed to query the round functions on which the network is based, but it still has access to the main component in both direction. With this example, we want to illustrate the fact that our tool can be used to verify complex relations, like the one proposed in [34], which is tedious to corroborate by *pen-and-paper*. The second entry corresponds to the analysis of the example given in Section 6.1.3, while the rest of components in this block correspond to attacks automatically found with our tool (see Section 6.6.2). Observe that one of them corresponds to the attack described by Coron et al. in [86].

In the next block, we present two attacks against the *iterated Even Mansour*, the first corresponds to an attack found automatically with our tool, the second corresponds to the attack to the 3-rounds IEM (Iterated Even-Mansour) from [141], and the third one corresponds to a very recent attack to the 4-rounds IEM from [91], where the authors show that 5 rounds are sufficient for indifferentiability.

In the third block we analyze the universality of known attacks against the Merkle-Damgård construction (without the prefix-free encodings) and a distinguisher for the 2-rounds confusion-diffusion network, where the public permutation is set (for simplicity) to a swap of blocks. It is future work to extend the expressivity of our distinguishers in order to capture the more general setting of CD networks (see Section 6.6.3).

[4]Source code available at `https://github.com/miguel-ambrona/indiff`.

Reference	Primitive	Attack type	Time
Barbosa and Farshim [34]	3-rounds Feistel	IND-CCA	0.01 ms
Figure 6.3 (David)	3-rounds Feistel	Indiff	0.25 ms
Figure 6.10	3-rounds Feistel	Indiff	3.89 ms
Figure 6.16	4-rounds Feistel	Indiff	10.7 ms
Coron et al. [86]	5-rounds Feistel	Indiff	13.5 ms
Figure 6.17 (new attack)	5-rounds Feistel	Indiff	14.2 ms
Figure 6.13	2-rounds IEM	Indiff	0.15 ms
Lampe and Seurin [141]	3-rounds IEM	Indiff	10.9 ms
Dai et al. [91]	4-rounds IEM	Indiff	179 ms
Prefix attack	Merkle-Damgård	Indiff	0.06 ms
Dodis et al. [98]	2-rounds Conf-Diff.	Indiff	1.17 ms
Figure 6.3 (Daisy)	3-rounds Feistel	Indiff ⊘	0.41 ms
Naïve attack	6-rounds Feistel	Indiff ⊘	0.39 ms

Table 6.1: Case studies (checking universality of distinguishers).

Finally, we include a fourth block presenting *negative examples*, examples of distinguishers that *are not* universal, in order to further test our tool.

6.6.2 Automated attacks search

In this section we describe our heuristic techniques for the automated search of distinguishers. Table 6.2 summarizes our results. We consider two heuristics for fully automated attacks search, these heuristics take as input the description of a cryptographic primitive and try to find a universal distinguisher for it. Figure 6.11 illustrates the input that our heuristics take.

6.6.2.1 Heuristic 1

This first heuristic is based on the idea that it is hard (with a polynomial number of queries to a random oracle) to find values satisfying certain relation, such that their image by a random oracle satisfies a relation too. If that is the case, simulators will be able to do very little to make both relations hold, what can be exploited to derive a distinguishing attack. Roughly, we try to find a non-trivial relation in the structure of the given cryptographic component.

Heuristic 1 first defines a *unification problem*. It starts with a set of variables $(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)$ for certain $\ell \in \mathbb{N}$, where every vector $\boldsymbol{x}_i$ or $\boldsymbol{y}_i$ has a length that matches the arity of the main component. The heuristic *binds* $\boldsymbol{x}_i$ with $\boldsymbol{y}_i$ for every $i \in [\ell]$ by adding equations to the problem that model the fact that $\boldsymbol{y}_i$ *is the output by the main component on input* $\boldsymbol{x}_i$. It then selects two expressions

```
(* 4-rounds Feistel *)

search_attack Indiff {

  rounds F1,F2,F3,F4.

  oracle R (x0,x1) :=
    x2 = x0 + F1(x1);
    x3 = x1 + F2(x2);
    x4 = x2 + F3(x3);
    x5 = x3 + F4(x4);
    return x4, x5.

  oracle R^-1 (x4,x5) :=
    x3 = x5 + F4(x4);
    x2 = x4 + F3(x3);
    x1 = x3 + F2(x2);
    x0 = x2 + F1(x1);
    return x0, x1.
}.
```

```
(* 2-rounds Even-Mansour *)

search_attack Indiff {

  rounds P1,P2 invertible.

  oracle R (k,x) :=
    return P2(P1(x) + k).

  oracle R^-1 (k,y) :=
    return P1^-1(k + P2^-1(y)).

}.
```

Figure 6.11: Input files for the fully automated attack search on 4-rounds Feistel (on the left) and 2-rounds Iterated Even-Mansour, where the first and last round key additions are omitted[6] (on the right).

φ_1, φ_2 in the variables that we have mentioned and it solves the unification problem $\varphi_1(\boldsymbol{x}_1, \dots, \boldsymbol{x}_\ell) = 0$ and $\varphi_2(\boldsymbol{y}_1, \dots, \boldsymbol{y}_\ell) = 0$ together with the equations corresponding to the bindings. The system must have inequalities to guarantee that φ_1 or φ_2 would not be zero in the *random world*.

A solution to the described system can potentially lead to a distinguishing attack. For that, the next step is to *decompile* such relations into programs, with the help of the solution that has been found. This may introduce an exponential blow-up due to potential permutations of sets of instructions, but in practice, these sets of instructions have dependencies between them and, in general, there are not so many possibilities. For every possible program decompiled from the relations, we test with our method from Section 6.5 whether it is a universal distinguisher.

In this first implementation, expressions φ_1, φ_2 are selected by considering all possible expressions of a limited size. In the experiments from Table 6.2 the

[6]Such additions are irrelevant for indifferentiability as noticed in [90].

Description	Primitive	Attack type	Heuristic 1	Heuristic 2
Check tool	2-rounds Feistel	IND-CPA	44.1 ms ✓	0.79 ms ✓
Check tool	3-rounds Feistel	IND-CCA	261 ms ✓	133 ms ✓
Figure 6.16	4-rounds Feistel	Indiff	20.6 s ✓	$>$ 100 s ⊘
-	5-rounds Feistel	Indiff	$>$ 100 s ⊘	$>$ 100 s ⊘
Figure 6.14	2-rounds *pal*-Feistel	IND-CPA	45.3 ms ✓	1.8 ms ✓
Figure 6.15	3-rounds *pal*-Feistel	IND-CCA	201 ms ✓	4.0 ms ✓
Figure 6.15	4-rounds *pal*-Feistel	IND-CCA	33.0 s ✓	7.4 ms ✓
Figure 6.15	5-rounds *pal*-Feistel	IND-CCA	$>$ 100 s ⊘	20.9 ms ✓
Figure 6.15	6-rounds *pal*-Feistel	IND-CCA	$>$ 100 s ⊘	43.3 ms ✓
Figure 6.15	7-rounds *pal*-Feistel	IND-CCA	$>$ 100 s ⊘	165.4 ms ✓
Figure 6.12	1-round IEM	IND-CPA	85.4 ms ×	0.46 ms ✓
Figure 6.13	2-rounds IEM	Indiff	739 ms ×	9.9 ms ✓
-	3-rounds IEM	Indiff	67.6 s ×	$>$ 100 s ⊘
Check tool	2-rounds CD	Indiff	134 ms ✓	17.1 s ✓

Table 6.2: Results on fully automated attacks search. Symbol ✓ is used when the algorithm found an attack, × is used when the algorithm terminates without an attack and ⊘ denotes a timeout (the algorithm did not give an answer after 100 seconds). Here, *pal*-Feistel stands for *palindromic* Feistel network.

size has been limited to 7, i.e. all symbolic expressions that (when interpreted as a tree) contain at most 7 leaves.

6.6.2.2 Heuristic 2

This heuristic follows a similar idea, but executes it in a different way. It basically considers a computation c (see Figure 6.8) that, after being executed, defines a list of symbolic expression. Heuristic 2 solves a linear system on these expressions trying to get a linear combination of them that is symbolically equivalent to 0. Every such linear combination defines a *distinguishing event* that may lead to a universal attack.

Again, in this first implementation we brute-force on all possible computations c of limited size. In our experiments, c is restricted to use at most 4 random samplings and consider equations of size at most 5.

6.6.2.3 Results

Table 6.2 summarizes our experiments on fully automated attacks search. We want to point out that the heuristics have been defined with a general purpose and do not explicitly exploit the properties of any primitive. All the primitives from the table have been analyzed with both heuristics. For some primitives we provide (in Section 6.8) the description of the attack found. When both heuristic found an attack, this description correspond to the attack found by the heuristic that performed better.

It is well-known that *palindromic*-Feistel constructions are insecure constructions, but we have considered these constructions to further test our heuristics.

An attack that does not require to invoke the round functions is IND-CCA, while if it only uses the main component in one direction, it is classified as IND-CPA.

Our new attack on 5-rounds Feistel networks (Figure 6.17) has been found by feeding our Heuristic 1 with specific relations. Namely,

$$\varphi_1 := x_1^{(1)} \oplus x_2^{(1)} \oplus x_3^{(1)} \oplus x_4^{(1)} \quad \text{and} \quad \varphi_2 := y_1^{(0)} \oplus y_2^{(0)} \oplus y_3^{(0)} \oplus y_4^{(0)}$$

where the bindings are $\mathcal{C}(x_i^{(0)}, x_i^{(1)}) = (y_i^{(0)}, y_i^{(1)})$ for every $i \in \{1, 2, 3, 4\}$.

6.6.3 Concluding remarks

In this work we initiate the study of indifferentiability from the perspective of symbolic analysis and automated cryptography. As it has been done in other frameworks, like the GGM, we provide results that guarantee that the symbolic conclusions derived about these systems have a meaningful interpretation in the actual model (without symbols, but concrete implementations of the algebraic structures).

We describe a general method for testing the universality of distinguishers in the context of indifferentiability. This method leverages techniques from unification theory, such as, *unification*, *deductibility*, *static equivalence*.

We provide an implementation of our methods, by solving such problems in the theory that we consider. Our implementation approaches these problems from a combinational point of view, we solve unification, deductibility and static equivalence in two disjoint theories, namely, the theory of $\oplus$ and the theory of function symbols with inverses. We then implement a solution for all problems based on the combination of the individual solutions. Arnaud, Cortier and Delaune [23] guarantee that the combination is *sound* and *complete*. However, our methods also depend on *deductibility constraints*, which, as shown by [71], may be undecidable in some *associative-commutative* theories, like the one we considered. We approach the problem by implementing (we refer to our tool for details) an algorithm for deducibility constraints, but it is future work to prove that our algorithm is sound and complete.

We also leave for future work to improve on the basic heuristics that we have presented for fully automated attacks search.

6.7 Proof of Master Theorem (Theorem 20)

We will prove our Master Theorem in two steps. Very roughly, the first step guarantees that after sampling names uniformly, the expression still does not unify to zero (it still contains function symbols) except with negligible probability. In the second step we prove that if it still contains functions symbols, it will only evaluate to zero with negligible probability.

For the sake of simplicity, the proof provided in this section is for the case when Fun only contains a function F of arity 1 with an inverse. The general case (multiple functions, arbitrary arities and general inverses) can be proven following almost the same argument (with a significant notational overhead). We also focus on the case of characteristic $p = 2$ for simplicity and write $\oplus$ instead of $+_2$ (and without loss of generality, note that all arguments can be extended for an arbitrary p).

In order to divide the proof in two steps, we define an intermediate grammar for expressions without names and variables, $\mathcal{E}_{\mathbb{G}}$, induced by group $\mathbb{G}$ of characteristic p, defined as follows:

$$\begin{array}{rll} \mathcal{E}_{\mathbb{G}} \triangleq & \mid a & \text{group element, } a \in \mathbb{G} \\ & \mid \mathcal{E}_{\mathbb{G}} \boxplus \mathcal{E}_{\mathbb{G}} & \text{addition} \\ & \mid F^{b}(\mathcal{E}_{\mathbb{G}}) & \text{function evaluation, } b \in \{-1, +1\} \end{array}$$

Grammar $\mathcal{E}_{\mathbb{G}}$ is equipped with the following set of simplification rules:

$$\begin{array}{rl} (x \boxplus y) \boxplus z = x \boxplus (y \boxplus z) & \text{associativity} \\ x \boxplus y = y \boxplus x & \text{commutativity} \\ x \boxplus x = 0_{\mathbb{G}} & \text{nilpotence} \\ x \boxplus 0_{\mathbb{G}} = x & \text{identity} \\ F(F^{-1}(x)) = F^{-1}(F(x)) = x & \text{function inverse} \\ a \boxplus b = a +_{\mathbb{G}} b & \text{simplification in } \mathbb{G}\text{, } a, b \in \mathbb{G} \end{array}$$

where $0_{\mathbb{G}}$ represents the identity element of $\mathbb{G}$ and $+_{\mathbb{G}}$ represents the group law in $\mathbb{G}$. The above grammar and rules define an equational theory that we denote by $\mathcal{T}_{\mathbb{G}}$. We say two expressions e_1, e_2 are equivalent in such theory, denoted by $e_1 =_{\mathcal{T}_{\mathbb{G}}} e_2$ if one can be rewritten to the other by the described rewriting rules.

Observe that procedure $\mathsf{inst}_{\mathbb{G}} : \mathcal{E}_2 \to \mathbb{G}$ can be splitted in the composition of two procedures, i.e., $\mathsf{inst}_{\mathbb{G}} = \mathsf{inst}^{\mathsf{f}}_{\mathbb{G}} \circ \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}$, where

$$\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}} : \mathcal{E}_2 \to \mathcal{E}_{\mathbb{G}} \qquad \text{and} \qquad \mathsf{inst}^{\mathsf{f}}_{\mathbb{G}} : \mathcal{E}_{\mathbb{G}} \to \mathbb{G} \ .$$

Roughly, procedure $\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}$ takes a ground expression from grammar $\mathcal{E}_2$ and converts names in group elements from $\mathbb{G}$, getting an expression from grammar

$\mathcal{E}_{\mathbb{G}}$. Then, $\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}$ takes an expression from grammar $\mathcal{E}_{\mathbb{G}}$ and produces a group element by selecting a random function f from the set of all functions from $\mathbb{G}$ to $\mathbb{G}$ and interpreting symbol F as function f.

We divide the proof of Theorem 20 in two helper lemmas (Lemmas 12 and 13).

Lemma 12. *Let $\mathbb{G}$ be a group of characteristic 2 and let e be a ground expression from grammar $\mathcal{E}_2$, containing at most d function symbols and such that $e \neq_{\mathcal{T}} 0$. It holds,*

$$\Pr\left[\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(e) =_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}}\right] \leqslant \frac{1+d}{|\mathbb{G}|}$$

where the probability is taken over the coins inside procedure $\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}$.

Proof. We prove the lemma by induction on d.

- If $d = 0$, expression e does not contain function symbols, but $e \neq_{\mathcal{T}} 0$ and therefore, it must contain names. When such names take a uniform value in $\mathbb{G}$, they will be reduced to an expression of $\mathcal{E}_{\mathbb{G}}$, which can actually be simplified (by the *simplification in $\mathbb{G}$ rule*) to a single group element in $\mathbb{G}$. It is clear that the probability that this element is $0_{\mathbb{G}}$ (that is $\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(e) =_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}}$) is exactly $1/|\mathbb{G}|$.

- If $d > 0$, expression e can be rewritten to an expression of the form:

$$c_1 \oplus \ldots \oplus c_k \oplus F^{b_i}(e_1) \oplus \ldots \oplus F^{b_i}(e_\ell)$$

where $\ell > 0$, $c_i \in \mathsf{Name}$ for $i \in [k]$ are pair-wise different, $b_j \in \{-1, +1\}$ and e_j are expressions of grammar $\mathcal{E}_2$ for $j \in [\ell]$. Note that all expressions can be rewritten to such a form and let us call the expression *homogeneous* when $k = 0$, i.e., when all top-level terms contain a function symbol. An expression without function symbols is said to be *homogeneous* if it is equivalent to 0. The notion of *homogeneity* extends naturally to grammar $\mathcal{E}_{\mathbb{G}}$. Let us consider the following events:

$$\begin{aligned} A &\equiv \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(e) =_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}} & \widehat{A} &\equiv \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(c_1 \oplus \ldots \oplus c_k) =_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}} \\ B &\equiv \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(e) \text{ is } \textit{hom.} & \widehat{B} &\equiv \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(F^{b_1}(e_1) \oplus \ldots \oplus F^{b_\ell}(e_\ell)) \text{ is } \textit{hom.} \end{aligned}$$

Observe that $\Pr[\widehat{B}]$ may not be 1 because there can be cancellations of the top level function symbol of some terms due to the *function inverse* rewriting rule. Also, note that $\Pr[A] \leqslant \Pr[B]$ and $\Pr[B \mid \widehat{B}] = \Pr[\widehat{A} \mid \widehat{B}]$, therefore,

$$\begin{aligned}
\Pr[A] &\leqslant \Pr[B] \\
&= \Pr[B \mid \widehat{B}]\Pr[\widehat{B}] + \Pr[B \mid \neg\widehat{B}]\Pr[\neg\widehat{B}] \\
&\leqslant \Pr[\widehat{A} \mid \widehat{B}]\Pr[\widehat{B}] + \Pr[\neg\widehat{B}] \\
&\leqslant \Pr[\widehat{A}] + \Pr[\neg\widehat{B}] \\
&\leqslant \frac{1+d}{|\mathbb{G}|} \ .
\end{aligned}$$

There, $\Pr[\widehat{A}]$ has been upper-bounded by $1/|\mathbb{G}|$ by a very similar argument to the one given in the case $d = 0$. Additionally, $\Pr[\widehat{B}]$ has been upper-bounded by $d/|\mathbb{G}|$. To see that, for every $j \in [\ell]$ we define the event

$$C_j \quad \equiv \quad \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}(F^{b_j}(e_j)) \text{ is } \textbf{not} \ \textit{homogeneous}$$

and we define d_j as the number of function symbols appearing in e_j. Note that $d_j < d$ for every $j \in [\ell]$. Also, note that if C_j does not occur, it is because e_j unified to $F^{-b_j}(a)$ (for some $a \in \mathbb{G}$) after $\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}$ and thus, $\Pr[\neg C_j]$ is upper-bounded by the probability of a symbolic expression (with d_j or fewer function symbols) evaluating to $0_{\mathbb{G}}$ after $\mathsf{inst}^{\mathsf{n}}_{\mathbb{G}}$. Therefore, by the *induction hypothesis*, $\Pr[C_j] \leqslant (1 + d_j)/|\mathbb{G}|$. Also, note that $\sum_{j\in[\ell]} d_j = d - \ell$. Finally, note that (by the union bound),

$$\Pr[\neg\widehat{B}] \leqslant \sum_{j\in[\ell]} \Pr[C_j] \leqslant \sum_{j\in[\ell]} \frac{1+d_j}{|\mathbb{G}|} = \underbrace{\sum_{j\in[\ell]} \frac{d_j}{|\mathbb{G}|}}_{=(d-\ell)/|\mathbb{G}|} + \underbrace{\sum_{j\in[\ell]} \frac{1}{|\mathbb{G}|}}_{=\ell/|\mathbb{G}|} = \frac{d}{|\mathbb{G}|} \ .$$

$\square$

Before stating the next lemma, observe that any expression e from grammar $\mathcal{E}_{\mathbb{G}}$ containing d function symbols can always be rewritten and seen as a tree with at most $d{+}1$ leaves and where all leaves are elements of $\mathbb{G}$. For example, if $\mathbb{G}$ is implemented as the set of 3-bitstrings with the *exclusive or* operation, the symbolic expression

$$F\,(F\,(\mathsf{110} \boxplus \mathsf{001}) \boxplus F\,(\mathsf{000}) \boxplus \mathsf{110} \boxplus \mathsf{111}) \boxplus \mathsf{101}$$

can be represented by the following tree:

$$\begin{tikzcd}
 & \mathsf{101} & & F \arrow[r] & \mathsf{111} \\
\boxplus \arrow[ur] \arrow[dr] & & & & \\
 & F \arrow[r] & \boxplus \arrow[uur] \arrow[r] \arrow[dr] & F \arrow[r] & \mathsf{000} \\
 & & & \mathsf{001} &
\end{tikzcd}$$

We define $\mathsf{leaves(e)}$ as the set of group elements formed by the leaves of e in this reduced form, again, $|\mathsf{leaves(e)}| \leqslant d{+}1$.

Lemma 13. *Let $\mathbb{G}$ be a group of characteristic 2 and let e be an expression from grammar $\mathcal{E}_{\mathbb{G}}$, containing at most d function symbols and such that $e \neq_{\mathcal{T}_{\mathbb{G}}} 0$. It holds,*

$$\Pr\left[\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}(e) =_{\mathbb{G}} 0_{\mathbb{G}}\right] \leqslant \frac{\frac{1}{3}d^3 + \frac{5}{2}d^2 + \frac{37}{6}d}{|\mathbb{G}|}$$

where the probability is taken over the coins inside procedure $\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}$.

Proof. We will prove the result by induction in d.

- If $d = 0$, we have that $e =_{\mathcal{T}_{\mathbb{G}}} a$ for some $a \in \mathbb{G}$ different from the identity $0_{\mathbb{G}}$. In that case, $\Pr[\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}(e) =_{\mathbb{G}} 0_{\mathbb{G}}] = 0$.

- If $d > 0$, let L be the set of all leaves (in the reduced tree for e) together with $0_{\mathbb{G}}$. Now, consider one of the leaves that has a function symbol at the next level in the tree. Without loss of generality, we can think of this leaf as the next one evaluated by the procedure $\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}$, which will choose uniformly a value $v \in \mathbb{G}$ for function F on this leaf. Let A_L be the event of v being such that for all $a \in L$, $v +_{\mathbb{G}} a \notin L$. Now, observe that if A_L does not occur, the expression e will be transformed into an expression $\widehat{e}$, that cannot be further simplified with the rewriting rules from $\mathcal{E}_{\mathbb{G}}$. Therefore, it is guaranteed that $\widehat{e} \neq_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}}$ and note that $\widehat{e}$ contains at most $d{-}1$ function symbols. The induction hypothesis gives us that

$$\Pr\left[\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}(\widehat{e}) =_{\mathbb{G}} 0_{\mathbb{G}}\right] \leqslant \frac{\frac{1}{3}(d{-}1)^3 + \frac{5}{2}(d{-}1)^2 + \frac{37}{6}(d{-}1)}{|\mathbb{G}|} \ .$$

  Furthermore, note that $|L| = \mathsf{leaves(e)}{+}1 \leqslant d{+}2$ and therefore, the probability of A_L can be upper-bounded by $(d{+}2)^2/|\mathbb{G}|$. We have,

$$\begin{aligned}\Pr\left[\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}}(e) =_{\mathbb{G}} 0_{\mathbb{G}}\right] &\leqslant \frac{\frac{1}{3}(d{-}1)^3 + \frac{5}{2}(d{-}1)^2 + \frac{37}{6}(d{-}1)}{|\mathbb{G}|} + \Pr[A_L] \\ &\leqslant \frac{\frac{1}{3}(d{-}1)^3 + \frac{5}{2}(d{-}1)^2 + \frac{37}{6}(d{-}1)}{|\mathbb{G}|} + \frac{(d{+}2)^2}{|\mathbb{G}|} \\ &= \frac{\frac{1}{3}d^3 + \frac{5}{2}d^2 + \frac{37}{6}d}{|\mathbb{G}|} \ .\end{aligned}$$

□

We are now ready to provide a proof for our Master Theorem.

Proof of Theorem 20. Let e be an expression containing at most d function symbols and such that $e \neq_{\mathcal{T}} 0$. We have,

$$\begin{aligned}
\Pr\left[\mathsf{inst}_{\mathbb{G}}(e) =_{\mathbb{G}} 0_{\mathbb{G}}\right] &= \Pr\left[(\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}} \circ \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}})(e) =_{\mathbb{G}} 0_{\mathbb{G}}\right] \\
\text{(by Lemma 12)} &\leqslant \Pr\left[(\mathsf{inst}^{\mathsf{f}}_{\mathbb{G}} \circ \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}})(e) =_{\mathbb{G}} 0_{\mathbb{G}} \,\middle|\, \mathsf{inst}^{\mathsf{n}}_{\mathbb{G}} \neq_{\mathcal{T}_{\mathbb{G}}} 0_{\mathbb{G}}\right] + \frac{1+d}{|\mathbb{G}|} \\
\text{(by Lemma 13)} &\leqslant \frac{\frac{1}{3}d^3 + \frac{5}{2}d^2 + \frac{37}{6}d}{|\mathbb{G}|} + \frac{1+d}{|\mathbb{G}|} \\
&\leqslant \frac{\frac{1}{3}d^3 + \frac{5}{2}d^2 + \frac{43}{6}d + 1}{|\mathbb{G}|} \\
&\leqslant \frac{d^3 + 27}{|\mathbb{G}|} \ .
\end{aligned}$$

where the last inequality is due to the fact that $\frac{1}{3}d^3 + \frac{5}{2}d^2 + \frac{43}{6}d + 1 \leqslant d^3 + 27$ for every value of $d \in \mathbb{N}$. $\square$

6.8 Distinguishers mentioned in Section 6.6

In this section we present the distinguishers (automatically found with our heuristics) that we refer to in the other sections of this chapter.

Computation:	**Distinguishing event:**
1 $a \leftarrow_{\$} \{0,1\}^n$ 2 $b \leftarrow_{\$} \{0,1\}^n$	$\mathcal{C}(a,a) \oplus a = \mathcal{C}(b,b) \oplus b$

Figure 6.12: Attack (IND-CPA) on 1-round Iterated Even-Mansour (automatically found with our Heuristic 2).

Computation:	**Distinguishing event:**
1 $b \leftarrow_{\$} \{0,1\}^n$ 2 $r_1 \leftarrow \mathrm{q}(P_2, +, 0)$ 3 $r_2 \leftarrow \mathrm{q}(P_1, +, b)$	$r_1 \oplus \mathcal{C}(r_2, b) = 0$

Figure 6.13: Attack (Indiff) on 2-rounds Iterated Even-Mansour (automatically found with our Heuristic 2).

Computation:

```
$a \leftarrow_\$ \{0,1\}^n$
$b \leftarrow_\$ \{0,1\}^n$
```

Distinguishing event:

$$a \oplus b = \mathcal{C}_1(a,b) \oplus \mathcal{C}_1(b,b)$$

Figure 6.14: Attack (IND-CPA) on 2-rounds palindromic Feistel network (automatically found with our Heuristic 2, for all $k \leqslant 7$).

Computation:

```
$a \leftarrow_\$ \{0,1\}^n$
```

Distinguishing event:

$$\mathcal{C}_1(a,a) = \mathcal{C}_1^{-1}(a,a)$$

Figure 6.15: Attack (IND-CCA) on k-rounds palindromic Feistel network (automatically found with our Heuristic 2).

Computation:

```
$r_1 \leftarrow_\$ \{0,1\}^n$
$f_1 \leftarrow \mathrm{q}(F_3, r_1)$
$f_2 \leftarrow \mathrm{q}(F_2, f_1)$
$f_3 \leftarrow \mathrm{q}(F_1, r_1 \oplus f_2)$
$f_4 \leftarrow \mathrm{q}(F_2, r_1 \oplus f_2 \oplus f_1)$
$f_5 \leftarrow \mathrm{q}(F_1, r_1 \oplus f_4)$
$x_0 := f_1 \oplus f_5$
$x'_0 := r_1 \oplus f_1 \oplus f_2 \oplus f_5$
$x''_0 := r_1 \oplus f_1 \oplus f_2 \oplus f_3$
$x_1 := r_1 \oplus f_4$
$x'_1 := x_1$
$x''_1 := r_1 \oplus f_2$
```

Distinguishing events:

$$x_0 \oplus x'_0 \oplus x''_1 = 0$$

$$\mathcal{C}_1(x_0, x_1) \oplus \mathcal{C}_1(x'_0, x'_1) \oplus \mathcal{C}_1(x''_0, x''_1) = 0$$

Figure 6.16: Attack on 4-rounds Feistel network (automatically found with our Heuristic 1).

Computation:

```
1   $a_1 \leftarrow_\$ \{0,1\}^n$
2   $a_2 \leftarrow_\$ \{0,1\}^n$
3   $a_3 \leftarrow_\$ \{0,1\}^n$
4   $f_1 \leftarrow \mathrm{q}(F_3, a_1)$
5   $f_2 \leftarrow \mathrm{q}(F_3, a_2)$
6   $v := f_1 \oplus f_2 \oplus a_3$
7   $f_3 \leftarrow \mathrm{q}(F_2, a_3)$
8   $f_4 \leftarrow \mathrm{q}(F_2, v)$
9   $f_5 \leftarrow \mathrm{q}(F_1, f_3 \oplus a_2)$
10  $f_6 \leftarrow \mathrm{q}(F_1, f_3 \oplus a_1)$
11  $f_7 \leftarrow \mathrm{q}(F_1, f_4 \oplus a_2)$
12  $f_8 \leftarrow \mathrm{q}(F_1, f_4 \oplus a_1)$
13  $x_0 := f_5 \oplus a_3$
14  $x_0' := f_6 \oplus a_3$
15  $x_0'' := f_7 \oplus v$
16  $x_0''' := f_8 \oplus v$
17  $x_1 := f_3 \oplus a_2$
18  $x_1' := f_3 \oplus a_1$
19  $x_1'' := f_4 \oplus a_2$
20  $x_1''' := f_4 \oplus a_1$
```

Distinguishing events:

$$x_1 \oplus x_1' \oplus x_1'' \oplus x_1''' = 0$$

$$\mathcal{C}_1(x_0, x_1) \oplus \mathcal{C}_1(x_0', x_1') \oplus \mathcal{C}_1(x_0'', x_1'') \oplus \mathcal{C}_1(x_0''', x_1''') = 0$$

$$x_1 \neq x_1'$$

$$x_1 \neq x_1''$$

$$x_1 \neq x_1'''$$

Figure 6.17: New attack on 5-rounds Feistel found with our tool, which is structurally different from the attack by Coron et al. [86].

7 Conclusions & Future Work

Amplify, clarify, and punctuate, and let the viewer draw his or her own conclusion.

Keith Jackson

Computer assistance is an emerging practice in the design and verification of cryptographic primitives and protocols. However, the line of research in computer-aided cryptography that focuses on automated analysis is quite recent and fairly unexplored. There exist many open challenges in this topic, which is lagging behind cryptography research by several years. Recent efforts in this field successfully apply automated analysis techniques to actual cryptographic primitives, nonetheless, it remains unclear how automated analysis can be applied to the most recent and advanced constructions.

In this thesis, we present new and relevant results for automated analysis in cryptography. We develop new techniques and tools to broaden the scope of computer-aided cryptography, with special emphasis on pairing-based cryptography.

On the focus on pairing-based constructions. We note that most of our techniques are not limited to the bilinear groups paradigm and could be naturally extended to other settings. However, pairing-based cryptography is a very relevant target, because pairings are often used to build advanced constructions, such as Attribute-Based Encryption or Structure-Preserving Signatures.

Analysis of Cryptographic Constructions in the GGM. The approach that we have adopted for automated analysis in Chapter 3 and Chapter 5 is based on analyzing systems of symbolic constraints. Once a cryptographic primitive and its desired security are specified, proving its security is reduced to deciding whether a system of constraints has a solution or not. Such approach usually involves different non-deterministic choices, guided by a heuristic. More precisely, analyzing such systems of constraints is commonly done through simplification rules that allow to transform the systems into simpler and equivalent ones. These rules do not introduce new solutions to the system and they do not produce new systems with fewer solutions after its application. A security proof consists of a sequence of rule application steps that eventually lead to a final state (from which a solution to the system can be found or no solution exists). Finding the right sequence of rules is a very challenging problem for automation and the designed heuristics sometimes fall short. The tools we develop in this thesis never make mistakes (they are proven to be sound), but can sometimes fail to produce an output (they are not proven to be complete). A very interesting line of future work is to improve the heuristics that we propose, increasing the number of schemes that can be analyzed with our tools.

However, note that most of the problems we focus on (namely, proving security of constructions in our models) are known to be *undecidable*. Therefore, there is no hope to produce *complete* tools in the most general scenarios. It is still a very interesting problem for future work to define restricted classes of primitives or constructions that admit complete methods for analyzing their security.

On the expressivity of Attribute-Based Encryption. One of the main challenges on ABE is expressivity: the state-of-the-art constructions do not achieve a good balance between expressivity and size. Another very challenging problem is direct revocation (see Section 4.6.1.2), which cannot be satisfactorily achieved yet. Our results from Chapter 4 give new insights on addressing these problems, making ABE feasible for real applications. The benchmarks for our improved predicate encodings (see Figure 4.3) show clear run-time improvements for setup, encryption and key generation and minimal improvements for decryption. However, the state-of-the-art for Attribute-Based Encryption is still very far from the performance achieved by other simpler *public-key* cryptosystems (classic Public-Key Encryption, or the simple Identity-Based Encryption). Consequently, the study of this important area of cryptography must continue. We believe our new results open new research directions in this topic.

Analysis of ABE in the Generic Group Model and beyond. On the practical side, it would be very interesting to develop synthesis techniques for exploring the space of Rational-Fraction Induced ABE (RFI-ABE). As in our Chapter 3 or prior works using synthesis [44], it would be interesting to explore classes of constructions systematically, using our tool for finding attacks and proofs. In order to achieve broader coverage (in other words, minimize the number of schemes for which the tool times out), we intend to improve the efficiency of the tool both for finding attacks and proofs. Moreover, it would be desirable to establish mathematical theorems that justify focusing on more restricted, tractable, classes of constructions (else, the search space far exceeds current computing capabilities). Beyond ABE, it seems appealing to explore whether our tool could be used for Structure-Preserving Signatures and, in particular, to synthesize SPSs based on rational fractions (note that the synthesis performed in Chapter 3 has been analyzed with our gga^{∞} , which does support Laurent polynomials in the exponent, but not rational fractions).

On the theoretical side, it would be interesting to prove that RFI-ABE are selectively secure (Definition 12) in the standard model, under a strong q-type hardness assumption.

On our methods for indifferentiability. We note that the results presented in Chapter 6 are still work in progress, that has not been submitted for review yet. Our implemented algorithms for *unification*, *deductibilty* and *static equivalence* are sound and complete, as guaranteed by the results of Arnaud et al. [23]. However, our methods for testing the universality of distinguishers (some times) additionally require analyzing *deducibility constraints*, for which we have implemented an algorithm, that we have not proven sound nor complete. Deducibility constraints is a relatively new topic and not much is known about this problem in *associative-commutative* theories yet (some works give sufficient conditions for the decidability of the problem [71], but extra analysis is required to guarantee that our systems meet such conditions). We leave for future work proving the validity of our algorithm for deducibility constraints or designing one that is sound and complete. However, we note that the value of our work is mainly in our techniques. Our method is sound, but the decidability of the problem we consider depends on an algorithm that is still unknown. Nevertheless, in many cases, unsatisfiablility can be guaranteed by only relying on *unification*, *deductibilty* and *static equivalence*, for which we have decision procedures. Furthermore, it is worth noticing that our heuristics lead to *actual valid* attacks.

On the implementations. Our implementations reproduce the algorithms described in this document and are open-source for reproducibility and public verifiability. However, even though we have made a big effort for faithfully implementing our methods, our tools might contain errors. Using *formal verification* for proving that our implementations meet the theoretical description of our algorithms is a very ambitious and interesting future work direction. Verifying our algorithms would require new approaches, because the nature of our proof techniques is quite unique.

Nonetheless, our techniques have their own value. Implementations are useful to experiment with our methods and to show that our techniques are feasible and applicable to actual examples from the literature. Our tools validate our theoretical results and evidence that they are worth to be explored in more detail. Furthermore, there is no reason to think that our tools misbehave and, even if they are not verified, they are still useful to *play with them*. They can be used to gain confidence about the design of certain primitives or to discard others if attacks were found. Consequently, our tools are not only useful to support our theoretical results, but can be helpful for building new cryptographic primitives. Again, a responsible use of our implementations would require an additional theoretical analysis of the primitives that are discovered with them.

Other research directions. Finally, another very interesting future work is to relate automated analysis in cryptography to the field of Artificial Intelligence (AI). On the one hand, research and new techniques on automated reasoning for cryptography can lead to new approaches to other problems and applications of AI. On the other hand, automated analysis in cryptography can benefit from existing techniques in the area of AI. In particular, machine learning [175] can lead to important advances in the framework of automated reasoning for cryptography. We refer to [169] for a survey of the relationship between the fields of cryptography and machine learning. Some works explore the application of AI to cryptanalysis [121, 142], others explore how neural networks [158] can be used for the design of cryptographic primitives [57, 113]. However, to the best of our knowledge, the application of AI techniques to the area of automated proofs in cryptography remains practically unexplored. A very interesting line of research is to investigate how our heuristics on constraint solving could leverage techniques from machine learning such as neural networks or decision trees [69] to improve their effectiveness.

Cryptography is a very natural candidate for testing new AI techniques, and boosts the research in such an important field, which is the imminent and unavoidable future of our civilization. Who knows whether one day machines will be able to assist in the *creative part* of mathematical proofs.

Funding Acknowledgments

This research was partially supported by:

- Autocrypt: Automation in Cryptology (Ref: N00014-12-1-0914).
- SynCrypt: Automated Synthesis of Cryptographic Constructions (Ref.: N00014-15-1-2750).
- StrongSoft: Sound Technologies for Reliable, Open, New Generation Software (Ref: TIN2012-39391-C04-01). Financiado por el Ministerio de Economía y Competitividad.

Bibliography

[1] M. Abe, M. Ambrona, M. Ohkubo, and M. Tibouchi. Lower bounds on structure-preserving signatures for bilateral messages. In D. Catalano and R. De Prisco, editors, *Security and Cryptography for Networks*, pages 3–22, Cham, 2018. Springer International Publishing.

[2] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, Aug. 2010.

[3] M. Abe, J. Groth, K. Haralambiev, and M. Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Heidelberg, Aug. 2011.

[4] M. Abe, J. Groth, and M. Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, Dec. 2011.

[5] M. Abe, J. Groth, M. Ohkubo, and T. Tango. Converting cryptographic schemes from symmetric to asymmetric bilinear groups. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 241–260. Springer, Heidelberg, Aug. 2014.

[6] M. Abe, J. Groth, M. Ohkubo, and M. Tibouchi. Structure-preserving signatures from type II pairings. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 390–407. Springer, Heidelberg, Aug. 2014.

[7] M. Abe, J. Groth, M. Ohkubo, and M. Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Y. Lindell,

editor, *TCC 2014*, volume 8349 of *LNCS*, pages 688–712. Springer, Heidelberg, Feb. 2014.

[8] M. Abe, M. Kohlweiss, M. Ohkubo, and M. Tibouchi. Fully structure-preserving signatures and shrinking commitments. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 35–65. Springer, Heidelberg, Apr. 2015.

[9] S. Agrawal and M. Chase. *A Study of Pair Encodings: Predicate Encryption in Prime Order Groups*, pages 259–288. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[10] S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 259–288. Springer, Heidelberg, Jan. 2016.

[11] S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In J. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 627–656. Springer, Heidelberg, Apr. / May 2017.

[12] J. A. Akinyele, C. Garman, and S. Hohenberger. Automating fast and secure translations from type-I to type-III pairing schemes. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1370–1381. ACM Press, Oct. 2015.

[13] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.

[14] J. A. Akinyele, M. Green, and S. Hohenberger. Using SMT solvers to automate design tasks for encryption and signature schemes. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 399–410. ACM Press, Nov. 2013.

[15] J. A. Akinyele, C. U. Lehmann, M. D. Green, M. W. Pagano, Z. N. J. Peterson, and A. D. Rubin. Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565, 2010. http://eprint.iacr.org/2010/565.

[16] J. A. Akinyele, M. W. Pagano, M. D. Green, C. U. Lehmann, Z. N. J. Peterson, and A. D. Rubin. Securing electronic medical records using

attribute-based encryption on mobile devices. In X. Jiang, A. Bhattacharya, P. Dasgupta, and W. Enck, editors, *SPSM'11, Proceedings of the 1st ACM Workshop Security and Privacy in Smartphones and Mobile Devices, Co-located with CCS 2011, October 17, 2011, Chicago, IL, USA*, pages 75–86. ACM, 2011.

[17] M. Ambrona, G. Barthe, R. Gay, and H. Wee. Attribute-based encryption in the generic group model: Automated proofs and new constructions. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *ACM CCS 17*, pages 647–664. ACM Press, Oct. / Nov. 2017.

[18] M. Ambrona, G. Barthe, and B. Schmidt. Automated unbounded analysis of cryptographic constructions in the generic group model. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 822–851. Springer, Heidelberg, May 2016.

[19] M. Ambrona, G. Barthe, and B. Schmidt. Generic transformations of predicate encodings: Constructions and applications. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 36–66. Springer, Heidelberg, Aug. 2017.

[20] B. Applebaum, B. Arkis, P. Raykov, and P. N. Vasudevan. Conditional disclosure of secrets: Amplification, closure, amortization, lower-bounds, and separations. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:38, 2017.

[21] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[22] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification*, pages 281–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[23] M. Arnaud, V. Cortier, and S. Delaune. Combining algorithms for deciding knowledge in security protocols. In B. Konev and F. Wolter, editors, *Frontiers of Combining Systems*, pages 103–117, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[24] N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and

more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, Heidelberg, May 2014.

[25] N. Attrapadung. Dual system encryption framework in prime-order groups. Cryptology ePrint Archive, Report 2015/390, 2015. http://eprint.iacr.org/2015/390.

[26] N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, Dec. 2016.

[27] N. Attrapadung and H. Imai. *Conjunctive Broadcast and Attribute-Based Encryption*, pages 248–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[28] N. Attrapadung and H. Imai. *Dual-Policy Attribute Based Encryption*, pages 168–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[29] N. Attrapadung, B. Libert, and E. de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Heidelberg, Mar. 2011.

[30] N. Attrapadung and S. Yamada. Duality in abe: Converting attribute based encryption for dual predicate and dual policy via computational encodings. Cryptology ePrint Archive, Report 2015/157, 2015. http://eprint.iacr.org/2015/157.

[31] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An online social network with user-defined privacy. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 135–146, New York, NY, USA, 2009. ACM.

[32] C. E. Z. Baltico, D. Catalano, and D. Fiore. Practical functional encryption for bilinear forms. *IACR Cryptology ePrint Archive*, 2016:1104, 2016.

[33] C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. *IACR Cryptology ePrint Archive*, 2017:151, 2017.

[34] M. Barbosa and P. Farshim. The related-key analysis of Feistel constructions. In C. Cid and C. Rechberger, editors, *FSE 2014*, volume 8540 of *LNCS*, pages 265–284. Springer, Heidelberg, Mar. 2015.

[35] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 1–16, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[36] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. http://eprint.iacr.org/2005/133.

[37] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli. *Satisfiability modulo theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. 1 edition, 2009.

[38] G. Barthe. High-assurance cryptography: Cryptographic software we can trust. *IEEE Security and Privacy*, 13(5):86–89, Sep.-Oct. 2015.

[39] G. Barthe, J. Cederquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In D. Basin and M. Rusinowitch, editors, *2nd International Joint Conference on Automated Reasoning, IJCAR*, Lecture Notes in Computer Science, pages 385–399, Germany, 7 2004. Springer Verlag. Imported from DIES.

[40] G. Barthe, J. Cederquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In *Automated Reasoning - Second International Joint Conference, IJCAR 2004, Cork, Ireland, July 4-8, 2004, Proceedings*, pages 385–399, 2004.

[41] G. Barthe, J. Cederquist, and S. Tarento. A machine-checked formalization of the generic model and the random oracle model. In D. A. Basin and M. Rusinowitch, editors, *Automated Reasoning - Second International Joint Conference, IJCAR 2004, Cork, Ireland, July 4-8, 2004, Proceedings*, volume 3097 of *Lecture Notes in Computer Science*, pages 385–399. Springer, 2004.

[42] G. Barthe, J. M. Crespo, B. Grégoire, C. Kunz, Y. Lakhnech, B. Schmidt, and S. Zanella Béguelin. Fully automated analysis of padding-based encryption in the computational model. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 1247–1260. ACM Press, Nov. 2013.

[43] G. Barthe, E. Fagerholm, D. Fiore, J. C. Mitchell, A. Scedrov, and B. Schmidt. Automated analysis of cryptographic assumptions in generic group models. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014,*

Part I, volume 8616 of *LNCS*, pages 95–112. Springer, Heidelberg, Aug. 2014.

[44] G. Barthe, E. Fagerholm, D. Fiore, A. Scedrov, B. Schmidt, and M. Tibouchi. Strongly-optimal structure preserving signatures from type II pairings: Synthesis and lower bounds. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 355–376. Springer, Heidelberg, Mar. / Apr. 2015.

[45] G. Barthe, E. Fagerholm, D. Fiore, A. Scedrov, B. Schmidt, and M. Tibouchi. Strongly-optimal structure preserving signatures from type II pairings: Synthesis and lower bounds. In J. Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 355–376. Springer, 2015.

[46] G. Barthe, B. Grégoire, S. Heraud, and S. Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 71–90. Springer, Heidelberg, Aug. 2011.

[47] G. Barthe, B. Grégoire, and B. Schmidt. Automated proofs of pairing-based cryptography. In I. Ray, N. Li, and C. Kruegel:, editors, *ACM CCS 15*, pages 1156–1168. ACM Press, Oct. 2015.

[48] G. Barthe and S. Tarento. A machine-checked formalization of the random oracle model. In *Types for Proofs and Programs, International Workshop, TYPES 2004, Jouy-en-Josas, France, December 15-18, 2004, Revised Selected Papers*, pages 33–49, 2004.

[49] D. Basin, S. Mödersheim, and L. Viganò. Ofmc: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, Jun 2005.

[50] M. Baudet. Deciding security of protocols against off-line guessing attacks, 01 2005.

[51] M. Baudet, V. Cortier, and S. Kremer. Computationally sound implementations of equational theories against passive adversaries. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming*, pages 652–663, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[52] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* Ph.D., Technion - Israel Institute of Technology, 1996.

[53] A. Beimel. *Secret-Sharing Schemes: A Survey*, pages 11–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[54] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, Heidelberg, Aug. 2004.

[55] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.

[56] G. Bellaso. *La cifra del sig. Giovan Battista Belaso.* 1553.

[57] J. Blackledge, S. Bezobrazov, and P. Tobin. Cryptography using artificial intelligence. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6, July 2015.

[58] B. Blanchet. A computationally sound mechanized prover for security protocols. In *2006 IEEE Symposium on Security and Privacy*, pages 140–154. IEEE Computer Society Press, May 2006.

[59] B. Blanchet. Security protocol verification: Symbolic and computational models. In P. Degano and J. D. Guttman, editors, *Principles of Security and Trust*, pages 3–29, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[60] D. Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, Heidelberg, 1998. Invited paper.

[61] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[62] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 443–459. Springer, Heidelberg, Aug. 2004.

[63] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[64] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, Aug. 2001.

[65] D. Boneh and M. K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[66] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, Heidelberg, Feb. 2007.

[67] X. Boyen. Miniature CCA2 PK encryption: Tight security without redundancy. In K. Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 485–501. Springer, Heidelberg, Dec. 2007.

[68] X. Boyen. The uber-assumption family (invited talk). In S. D. Galbraith and K. G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, Heidelberg, Sept. 2008.

[69] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[70] E. Brier, J.-S. Coron, T. Icart, D. Madore, H. Randriam, and M. Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 237–254. Springer, Heidelberg, Aug. 2010.

[71] S. Bursuc, H. Comon-Lundh, and S. Delaune. Associative-commutative deducibility constraints. In W. Thomas and P. Weil, editors, *STACS 2007*, pages 634–645, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[72] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3), 2007.

[73] M. Chase and S. Meiklejohn. Déjà Q: Using dual systems to revisit q-type assumptions. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, Heidelberg, May 2014.

[74] M. Chase, S. Meiklejohn, and G. Zaverucha. Algebraic MACs and keyed-verification anonymous credentials. In G.-J. Ahn, M. Yung, and N. Li, editors, *ACM CCS 14*, pages 1205–1216. ACM Press, Nov. 2014.

[75] S. Chatterjee and A. Menezes. Type 2 structure-preserving signature schemes revisited. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 286–310. Springer, Heidelberg, Nov. / Dec. 2015.

[76] J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, Apr. 2015.

[77] J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, Heidelberg, Aug. 2013.

[78] J. Chen and H. Wee. Dual system groups and its applications — compact hibe and more. Cryptology ePrint Archive, Report 2014/265, 2014. http://eprint.iacr.org/2014/265.

[79] Y. Cherdantseva and J. Hilton. A reference model of information assurance & security. In *Proceedings of the 2013 International Conference on Availability, Reliability and Security*, ARES '13, pages 546–555, Washington, DC, USA, 2013. IEEE Computer Society.

[80] Y. Chevalier, R. Kusters, M. Rusinowitch, and M. Turuani. An np decision procedure for protocol insecurity with xor. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 261–270, June 2003.

[81] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, Dec. 2001.

[82] H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 271–280, June 2003.

[83] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[84] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. In V. Shoup, editor,

CRYPTO 2005, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, Aug. 2005.

[85] J.-S. Coron, T. Holenstein, R. Künzler, J. Patarin, Y. Seurin, and S. Tessaro. How to build an ideal cipher: The indifferentiability of the feistel construction. *Journal of Cryptology*, 29(1):61–114, Jan 2016.

[86] J.-S. Coron, T. Holenstein, R. Künzler, J. Patarin, Y. Seurin, and S. Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, Jan. 2016.

[87] J.-S. Coron, J. Patarin, and Y. Seurin. The random oracle model and the ideal cipher model are equivalent. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 1–20. Springer, Heidelberg, Aug. 2008.

[88] J. M. Crespo. Automation and modularity of cryptographic proofs in the computational model. 2016.

[89] D. Dachman-Soled, J. Katz, and A. Thiruvengadam. 10-round Feistel is indifferentiable from an ideal cipher. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 649–678. Springer, Heidelberg, May 2016.

[90] Y. Dai, Y. Seurin, J. Steinberger, and A. Thiruvengadam. *Indifferentiability of Iterated Even-Mansour Ciphers with Non-idealized Key-Schedules: Five Rounds Are Necessary and Sufficient*, pages 524–555. Springer International Publishing, Cham, 2017.

[91] Y. Dai, Y. Seurin, J. P. Steinberger, and A. Thiruvengadam. Indifferentiability of iterated Even-Mansour ciphers with non-idealized key-schedules: Five rounds are necessary and sufficient. In J. Katz and H. Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 524–555. Springer, Heidelberg, Aug. 2017.

[92] Y. Dai and J. Steinberger. Indifferentiability of 10-round Feistel networks. Cryptology ePrint Archive, Report 2015/874, 2015. http://eprint.iacr.org/2015/874.

[93] Y. Dai and J. Steinberger. Indifferentiability of 8-round feistel networks. In M. Robshaw and J. Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 95–120, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[94] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference*

on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08/ETAPS'08, pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[95] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems. A practical implementation of the timing attack. In J.-J. Quisquater and B. Schneier, editors, *Smart Card Research and Applications*, pages 167–182, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[96] T. Dierks and C. Allen. *RFC 2246 - The TLS Protocol Version 1.0.* Internet Activities Board, Jan. 1999.

[97] T. T. A. Dinh and A. Datta. Streamforce: outsourcing access control enforcement for stream data to the clouds. In *Fourth ACM Conference on Data and Application Security and Privacy, CODASPY'14, San Antonio, TX, USA - March 03 - 05, 2014*, pages 13–24, 2014.

[98] Y. Dodis, M. Stam, J. P. Steinberger, and T. Liu. Indifferentiability of confusion-diffusion networks. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 679–704. Springer, Heidelberg, May 2016.

[99] D. Dolev and A. C.-C. Yao. On the security of public key protocols (extended abstract). In *22nd FOCS*, pages 350–357. IEEE Computer Society Press, Oct. 1981.

[100] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[101] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, Aug. 2013.

[102] E. Fagerholm. *Automated analysis in generic groups.* PhD thesis, University of Pennsylvania, 2015.

[103] H. Feistel. *Cryptography and Computer Privacy.* Scientific American, 1973.

[104] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May / June 2010.

[105] G. Fuchsbauer. Breaking existential unforgeability of a signature scheme from asiacrypt 2014. Cryptology ePrint Archive, Report 2014/892, 2014. http://eprint.iacr.org/2014/892.

[106] G. Fuchsbauer, C. Hanser, and D. Slamanig. EUF-CMA-secure structure-preserving signatures on equivalence classes. Cryptology ePrint Archive, Report 2014/944, 2014. http://eprint.iacr.org/2014/944.

[107] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. Cryptology ePrint Archive, Report 2017/620, 2017. https://eprint.iacr.org/2017/620.

[108] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. Cryptology ePrint Archive, Report 2000/061, 2000. http://eprint.iacr.org/2000/061.

[109] S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. http://eprint.iacr.org/2006/165.

[110] S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *ACM CCS 10*, pages 121–130. ACM Press, Oct. 2010.

[111] R. Gay, I. Kerenidis, and H. Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In R. Gennaro and M. J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 485–502. Springer, Heidelberg, Aug. 2015.

[112] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, Heidelberg, May / June 2006.

[113] A. Ghosh and A. Nath. Cryptography algorithms using artificial neural network. In *International Journal of Advance Research in Computer Science and Management Studies*, volume 2, pages 375–381, 12 2014.

[114] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982.

[115] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. Cryptology ePrint Archive, Report 2006/309, 2006. http://eprint.iacr.org/2006/309.

[116] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. Vimercati, editors, *ACM CCS 06*, pages 89–98. ACM Press, Oct. / Nov. 2006. Available as Cryptology ePrint Archive Report 2006/309.

[117] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, Dec. 2010.

[118] J. Groth. Efficient fully structure-preserving signatures for large messages. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 239–259. Springer, Heidelberg, Nov. / Dec. 2015.

[119] J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016.

[120] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, Apr. 2008.

[121] O. Grošek and Z. Pavol. Automated cryptanalysis. In *Encyclopedia of Artificial Intelligence. IGI Global*, pages 179–185, 10 2009.

[122] A. Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In M. J. Jacobson Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 357–372. Springer, Heidelberg, June 2013.

[123] S. Halevi. A plausible approach to computer-aided cryptographic proofs. *IACR Cryptology ePrint Archive*, 2005:181, 2005.

[124] M. Hamburg. *Spatial Encryption.* Ph.D. Thesis, Stanford University, California, 2011.

[125] V. T. Hoang, J. Katz, and A. J. Malozemoff. Automated analysis and synthesis of authenticated encryption schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 84–95, 2015.

[126] T. Holenstein, R. Künzler, and S. Tessaro. Equivalence of the random oracle model and the ideal cipher model, revisited. *CoRR*, abs/1011.1264, 2010.

[127] T. Holenstein, R. Künzler, and S. Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In L. Fortnow and S. P. Vadhan, editors, *43rd ACM STOC*, pages 89–98. ACM Press, June 2011.

[128] J. Y. Hwang, D. H. Lee, and M. Yung. Universal forgery of the identity-based sequential aggregate signature scheme. In W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, editors, *ASIACCS 09*, pages 157–160. ACM Press, Mar. 2009.

[129] M. Ion, J. Zhang, and E. M. Schooler. Toward content-centric privacy in ICN: attribute-based encryption and routing. In B. Ohlman, G. C. Polyzos, and L. Zhang, editors, *ICN'13, Proceedings of the 3rd, 2013 ACM SIGCOMM Workshop on Information-Centric Networking, August 12, 2013, Hong Kong, China*, pages 39–40. ACM, 2013.

[130] Y. Ishai and H. Wee. *Partial Garbling Schemes and Their Applications*, pages 650–662. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[131] T. Jager and J. Schwenk. On the equivalence of generic group models. In *Provable Security, Second International Conference, ProvSec 2008, Shanghai, China, October 30 - November 1, 2008. Proceedings*, pages 200–209, 2008.

[132] F. G. Jongkil Kim, Willy Susilo and M. H. Au. A tag based encoding: An efficient encoding for predicate encoding in prime order groups. Cryptology ePrint Archive, Report 2016/655, 2016. http://eprint.iacr.org/2016/655.

[133] A. Joux. A new index calculus algorithm with complexity $$l(1/4+o(1))$$ in small characteristic. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, pages 355–379, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[134] P. Junod and A. Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proceedings of the*

Tenth Annual ACM Workshop on Digital Rights Management, DRM '10, pages 13–24, New York, NY, USA, 2010. ACM.

[135] C. S. Jutla and A. Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. *Journal of Cryptology*, 30(4):1116–1156, Oct. 2017.

[136] D. Kahn. *The Codebreakers.* Signet book. Sphere, 1973.

[137] M. Karchmer and A. Wigderson. On span programs. In *In Proc. of the 8th IEEE Structure in Complexity Theory*, pages 102–111. IEEE Computer Society Press, 1993.

[138] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, Apr. 2008.

[139] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, Aug. 1996.

[140] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, Aug. 1999.

[141] R. Lampe and Y. Seurin. How to construct an ideal cipher from a small set of public permutations. In K. Sako and P. Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 444–463. Springer, Heidelberg, Dec. 2013.

[142] E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis. Cryptography and cryptanalysis through computational intelligence. Springer Berlin Heidelberg, 2007.

[143] A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 318–335. Springer, Heidelberg, Apr. 2012.

[144] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 62–91. Springer, Heidelberg, May / June 2010.

[145] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, Feb. 2010.

[146] A. B. Lewko and B. Waters. Decentralizing attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 568–588. Springer, Heidelberg, May 2011.

[147] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 547–567. Springer, Heidelberg, May 2011.

[148] A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, Aug. 2012.

[149] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren. Attribute-based signature and its applications. In D. Feng, D. A. Basin, and P. Liu, editors, *ASIACCS 10*, pages 60–69. ACM Press, Apr. 2010.

[150] B. Libert, T. Peters, M. Joye, and M. Yung. Compactly hiding linear spans - tightly secure constant-size simulation-sound QA-NIZK proofs and applications. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 681–707. Springer, Heidelberg, Nov. / Dec. 2015.

[151] H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In R. Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, Mar. 2012.

[152] Z. Liu and D. S. Wong. *Practical Ciphertext-Policy Attribute-Based Encryption: Traitor Tracing, Revocation, and Large Universe*, pages 127–146. Springer International Publishing, Cham, 2015.

[153] D. Lubicz and T. Sirvent. Attribute-based broadcast encryption scheme made efficient. In S. Vaudenay, editor, *AFRICACRYPT 08*, volume 5023 of *LNCS*, pages 325–342. Springer, Heidelberg, June 2008.

[154] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography*, SAC '99, pages 184–199, London, UK, UK, 2000. Springer-Verlag.

[155] A. J. Malozemoff, J. Katz, and M. D. Green. Automated analysis and synthesis of block-cipher modes of operation. In *IEEE 27th Computer Security Foundations Symposium, CSF 2014, Vienna, Austria, 19-22 July, 2014*, pages 140–152, 2014.

[156] U. M. Maurer. Abstract models of computation in cryptography (invited paper). In N. P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, Dec. 2005.

[157] U. M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, Feb. 2004.

[158] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[159] A. Miyaji, M. Nakabayashi, and S. TAKANO. New explicit conditions of elliptic curve traces for fr-reduction, 2001.

[160] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[161] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the Association for Computing Machinery*, 21(21):993–999, Dec. 1978.

[162] T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231. Springer, Heidelberg, Dec. 2009.

[163] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, Aug. 2010.

[164] T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Heidelberg, Apr. 2012.

[165] A. Petcher and G. Morrisett. The foundational cryptography framework. In R. Focardi and A. Myers, editors, *Principles of Security and Trust*, pages 53–72, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[166] E. A. Poe. *A Few Words on Secret Writing.* 1841.

[167] D. Pointcheval and O. Sanders. Short randomizable signatures. In K. Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, Feb. / Mar. 2016.

[168] T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

[169] R. L. Rivest. Cryptography and machine learning. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '91*, pages 427–439, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[170] J. Robinson. Loveland donald w.. automated theorem proving. a logical basis. fundamental studies in computer science, vol. 6. north-holland publishing company, amsterdam, new york, and oxford, 1978, xiii + 405 pp. 45:629–630, 09 2014.

[171] Y. Rouselakis and B. Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 13*, pages 463–474. ACM Press, Nov. 2013.

[172] A. Rupp, G. Leander, E. Bangerter, A. W. Dent, and A.-R. Sadeghi. Sufficient conditions for intractability over black-box groups: Generic lower bounds for generalized DL and DH problems. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 489–505. Springer, Heidelberg, Dec. 2008.

[173] A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 199–217. Springer, Heidelberg, Aug. 2012.

[174] A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[175] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3):210–229, July 1959.

[176] C. Schnorr. Security of blind discrete log signatures against interactive attacks. In S. Qing, T. Okamoto, and J. Zhou, editors, *Information and Communications Security, Third International Conference, ICICS 2001, Xian, China, November 13-16, 2001*, volume 2229 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2001.

[177] C.-P. Schnorr and M. Jakobsson. Security of signed ElGamal encryption. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 73–89. Springer, Heidelberg, Dec. 2000.

[178] S. Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2,3):111–126, Aug. 2002.

[179] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, Aug. 1984.

[180] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal, Vol 28, pp. 656–715*, Oktober 1949.

[181] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, Oct. 1997.

[182] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[183] Suetonius. *Vita Divi Julii.* AD 121. 56.6.

[184] M. Szydlo. A note on chosen-basis decisional Diffie-Hellman assumptions. In G. Di Crescenzo and A. Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 166–170. Springer, Heidelberg, Feb. / Mar. 2006.

[185] M. Szydlo. A note on chosen-basis decisional diffie-hellman assumptions. In *Financial Cryptography and Data Security*, pages 166–170. Springer, 2006.

[186] The Sage Developers. *Sage Mathematics Software (Version 6.8)*, 2015. `http://www.sagemath.org`.

[187] F. Wang, J. Mickens, N. Zeldovich, and V. Vaikuntanathan. Sieve: Cryptographically enforced access control for user data in untrusted clouds. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 611–626, Santa Clara, CA, Mar. 2016. USENIX Association.

[188] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, Aug. 2009.

[189] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, Heidelberg, Mar. 2011.

[190] H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, Heidelberg, Feb. 2014.

[191] H. Wee. Déjà Q: Encore! Un petit IBE. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 237–258. Springer, Heidelberg, Jan. 2016.

Acronyms

ABE	attribute-based encryption 4, 10, 14, 21, 65, 98, 176
AI	artificial intelligence 178
ASP	arithmetic span program 88
CDH	computational Diffie-Hellman 25
CDS	conditional disclosure of secrets 79
CNF	conjunctive normal form 117
CP-ABE	ciphertext-policy attribute-based encryption 21, 79, 114
CPU	central processing unit 60, 132, 163
DDH	decisional Diffie-Hellman 9, 20, 26, 34
DLIN	decision linear assumption 68
DLOG	discrete logarithm 25
DNA	desoxyribo nucleic acid 4
DNF	disjunctive normal form 91, 117
DP-ABE	dual-policy attribute-based encryption 11, 65, 85
DSG	dual system groups 99
EMR	electronic medical records 65
EUF-CMA	existential unforgeability against chosen-message attacks 18, 40
EUF-RMA	existential unforgeability against random-message attacks 18
GGM	generic group model 4, 14, 25, 31, 37, 98
HIBE	hirarchical identity-based encryption 91
IBE	identity-based encryption 10, 21, 67, 132
IEM	iterated Even-Mansour 163
IND-CCA	indist. against non-adaptive chosen-ciphertext attacks 14, 20
IND-CPA	indistinguishability against chosen-plaintext attacks 14, 20
IPE	inner-product encryption 117
KP-ABE	key-policy attribute-based encryption 21, 79, 114
MAC	message authentication code 14, 61
NIPE	non-zero inner-product encryption 78
NIZK	non-interactive zero-knowledge 17

NNF	negation normal form 86
OAEP	optimal asymmetric encryption padding 4
PE	predicate encryption 9, 21, 63, 65
PKE	public-key encryption 9, 19, 63
PPT	probabilistic polynomial time 7
RAM	random access memory 60, 132, 163
RFI-ABE	rational-fraction induced ABE 12, 101, 177
RSA	Rivest-Shamir-Adleman 4, 28
SMT	satisfiability modulo theories 5, 60
SPS	structure-preserving signature 5, 14, 17, 32, 61, 177
SXDH	symmetric external Diffie-Hellman 68, 87
TLS	transport layer security 4
ZIPE	zero inner-product encryption 65, 78

Index

This page has been added intentionally so that there exists a finite field with as many elements as there are pages in this document.

Miguel Ambrona Castellanos

Madrid, miércoles 26 de septiembre de 2018