Attaining Basically Everything in Attribute-Based Encryption

Simplifying the Design of Practical Schemes via Pair Encodings

Marloes Venema

Radboud University

ISBN: 978-94-6419-866-9
Printed by Gildeprint
Cover design with Jacqueline Eskes-Stöteler
This cover has been designed using images from flaticon.com
The identity-based padlock on the cover was invented by Merel Brandon

Attaining Basically Everything in Attribute-Based Encryption

Simplifying the Design of Practical Schemes via Pair Encodings

Proefschrift ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.H.J.M. van Krieken,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

dinsdag 19 september 2023
om 14:30 uur precies

door

Marloes Trudy Christine Venema
geboren op 13 mei 1993
te Winterswijk

Promotoren:

Prof. dr. L. Batina
Prof. dr. B.P.F. Jacobs

Copromotor:

Dr. G. Alpár

Manuscriptcommissie:

Prof. dr. P. Schwabe (voorzitter)	
Prof. dr. E. Kiltz	Ruhr-Universität Bochum, Duitsland
Prof. dr. A. Lehmann	Universität Potsdam, Duitsland
Prof. dr. ir. B. Preneel	KU Leuven, België
Prof. dr. ir. P.J.M. Veugen	Universiteit Twente, Nederland

Abstract

Attribute-based encryption (ABE) cryptographically implements fine-grained access control on data. In particular, data can be stored by an entity that is not necessarily trusted to enforce access control, or an entity that is not even trusted to have access to the plaintext data at all. Instead, access control can be externally enforced by a trusted authority. Additionally, some multi-authority variants of ABE—which do not have a central authority—can effectively and securely implement access control in multiple-domain settings. Furthermore, ABE is the only cryptographic approach to fine-grained access control that does not require an online trusted third party during access requests, and thus provides better availability properties than more traditional access control mechanisms.

The actual realization of these theoretical advantages in practice depends on whether state-of-the-art ABE schemes support the necessary core properties. In this thesis, we investigate which properties are necessary, and how these properties can be realized efficiently and securely in pairing-based ABE. To this end, we use and build on the *pair encodings framework*, which simplifies the design and analysis of complex ABE schemes by abstracting the schemes to pair encodings. Among other reasons, we use this framework to simplify the cryptanalysis of existing schemes, which shows that several popular multi-authority schemes are broken. Furthermore, we use the common structure of schemes that fit in this framework to fairly benchmark multiple schemes. These benchmarks reveal that existing schemes that support several desirable core properties incur a significant (and fixed) trade-off in the computational efficiency of the encryption and decryption algorithms.

An additional feature of the pair encodings framework is that it allows us to generically realize several core properties that are otherwise difficult to realize. Such properties can be attained by applying transformations to the encodings, which are proven to preserve the security of the original encoding. These transformations are generic, and can therefore be applied to any secure pair encoding. However, the pair encodings framework has two shortcomings. First, it supports only single-authority ABE. To efficiently and securely support multi-authority ABE, we first extend the framework, by generalizing the definition of pair encodings. Second, as our benchmarks also underline, existing schemes in the pair encodings framework do not provide flexible efficiency trade-offs, either incurring high encryption or high decryption costs. Hence, we provide two schemes that allow for a more flexible approach. In particular, practitioners can fine-tune the schemes' efficiency trade-offs based on the computational resources of the encryption and decryption devices. Specifically, the first scheme, GLUE, has a flexible efficiency trade-off between the encryption and decryption algorithms, and can in particular implement an efficient decryption algorithm. The second scheme, TinyABE, has a flexible efficiency trade-off between the size of the master public key and the sizes of the ciphertexts, and can in particular implement an efficient encryption algorithm. We also present a novel conversion technique that generically and efficiently transforms any (passively) secure scheme in the pair encodings framework to a scheme that additionally provides security against chosen-ciphertext attacks. In contrast to existing conversion techniques that achieve this, ours incurs only a small constant overhead in all algorithms. In sum, this thesis helps us to attain the necessary core properties more efficiently for certain settings than was previously possible.

Samenvatting

Op **a**ttributen-ge**b**aseerde **e**ncryptie (ABE) implementeert cryptografisch een fijnmazige toegangscontrole op data. In het bijzonder kan data opgeslagen worden door een entiteit die niet noodzakelijkerwijs vertrouwd wordt om toegangscontrole af te dwingen, of een entiteit die niet eens vertrouwd wordt om zelf toegang tot deze data te hebben. In plaats daarvan kan een externe vertrouwde autoriteit toegangscontrole afdwingen. Daarnaast bestaan er multi-autoriteit varianten van ABE—die geen centrale autoriteit hebben—die effectief en veilig toegangscontrole kunnen implementeren in toepassingen met meerdere domeinen. Daar komt nog bij dat ABE de enige cryptografische oplossing is om toegangscontrole te implementeren die geen online vertrouwde partij nodig heeft bij ieder toegangsverzoek, en daardoor betere beschikbaarheidseigenschappen biedt dan meer traditionele mechanismen voor toegangscontrole.

De daadwerkelijke realisatie van deze theoretische voordelen in de praktijk hangt af van het feit of de nieuwste ABE schema's de noodzakelijke basiseigenschappen hebben. In dit proefschrift onderzoeken we welke eigenschappen nodig zijn, en hoe ze efficient en veilig gerealiseerd kunnen met bilineaire afbeeldingen. Om dit doel te bereiken gebruiken we en bouwen we op het raamwerk voor paarencoderingen, hetgeen het construeren en analyseren van complexe ABE schema's versimpelt door de schema's te abstraheren naar paarencoderingen. We gebruiken dit raamwerk onder andere om de cryptanalyse van bestaande schema's te versimpelen, hetgeen laat zien dat verschillende populaire multi-autoriteit schema's gebroken zijn. We gebruiken ook de gemeenschappelijke structuur van schema's die bevat zijn in dit raamwerk om de efficiëntie van meerdere schema's op een eerlijke manier te vergelijken. Deze vergelijkingen laten zien dat bestaande schema's met meerdere wenselijke basiseigenschappen een significante (en vaste) afweging bieden in computationele efficiëntie van de encryptie- en decryptiealgoritmes.

Een extra kenmerk van het raamwerk voor paarencoderingen is dat het mogelijk is om meerdere basiseigenschappen op een generieke manier te realiseren die normaal gesproken lastig te realiseren zijn. Zulke eigenschappen kunnen dan behaald worden door transformaties toe te passen op de encodering, waarvan bewezen is dat deze de veiligheid van de originele encodering bewaren. Deze transformaties zijn generiek, en kunnen daarom toegepast worden op iedere veilige paarencodering. Echter heeft het raamwerk voor paarencoderigen twee tekortkomingen. Ten eerste is dit raamwerk alleen van toepassing op ABE schema's met een enkele autoriteit. Om toch op een efficiënte en veilige manier meerdere autoriteiten te ondersteunen breiden we eerst het

raamwerk uit, door de bestaande definitie van paarencoderingen te generaliseren. Ten tweede, zoals onze vergelijkingen hebben laten zien, bieden bestaande schema's in dit raamwerk geen flexibele afwegingen in de efficiëntie. Ze hebben ofwel een inefficiente encryptie of decryptie. Daarom introduceren wij twee schema's met een flexibelere aanpak. In het bijzonder kunnen praktisch georienteerde mensen de afweging in efficiëntie van de schema's afstellen zodat ze de rekenkundige middelen van de encryptie- en decryptieapparaten in acht nemen. Het eerste schema, GLUE, heeft een flexibele afweging in efficiëntie tussen de encryptie- en decryptiealgoritmes, en kan daardoor een efficiënt decryptiealgoritme implementeren. Het tweede schema, TinyABE, heeft een flexibele afweging in efficiëntie tussen de groottes van de hoofdsleutel en de cijfertekst, en kan ook een efficiënt encryptiealgoritme implementeren. We introduceren ook een nieuwe conversietechniek die generiek en efficiënt een passief veilige schema in het raamwerk voor paarencoderingen kan transformeren in een schema dat ook veilig is tegen gekozen-cijfertext aanvallen. In tegenstelling tot bestaande conversietechnieken die dit doen zorgt onze transformatie ervoor dat de extra kosten voor alle algoritmes klein en constant zijn. Kortom, dit proefschrift helpt ons om de noodzakelijke basiseigenschappen te behalen op een efficientere manier voor bepaalde situaties dan eerder mogelijk was.

Acknowledgements

Over the course of four years, a PhD student typically accumulates a pretty impressive list of people that contributed to the journey leading up to the student's thesis. To properly thank those people, the best moment is usually at the end: when writing the acknowledgments section of the thesis. Like many PhD students have done before me, I will try to string the right words together to thank all the people that helped me through my academic career.

First and foremost, I would like to thank my supervisor, dr. Greg Alpár, for being a great supervisor. Greg, you provided me with much freedom in the research I conducted, allowed me to ramble about crypto for hours during our meetings and taught me so much about effectively presenting my work (in both my writing and presentations). Moreover, you were also of great support. You helped me stay on the right track and finish things, and kept me from taking too much on my plate. Furthermore, despite the many setbacks and hardships I faced during my PhD, you kept believing in me and my work. I can probably never express properly how much that means to me, but here is my best attempt.

Second, I would like to thank my promotors, prof. dr. Lejla Batina and prof. dr. Bart Jacobs, for giving me much academic freedom, support and valuable advice over the years. Lejla, you taught me much about the crypto community, introduced me to other academic activities such as reviewing papers and organizing conferences, and eventually helped me with finding a postdoc position. Bart, you were supportive of me when things were difficult, and you taught me much about real-world applications of crypto, most notably by inviting me to work with the "Encryption for all" team.

Third, I would like to thank my manuscript committee, consisting of prof. dr. Peter Schwabe, prof. dr. Eike Kiltz, prof. dr. Anja Lehmann, prof. dr. ir. Bart Preneel and prof. dr. ir. Thijs Veugen, for taking the time and effort to read through my ten-chaptered thesis, and provide valuable feedback.

Fourth, I would like to thank my co-authors: dr. Greg Alpár, Leon Botros, dr. Jaap-Henk Hoepman and dr. Antonio de la Piedra. Jaap-Henk, already during my master, you enthused me with your lectures in Advanced Network Security—which helped me to believe I could aspire to do a PhD—and introduced me to ABE with my master thesis—which eventually led to not only a master thesis, but also a PhD thesis about ABE. Antonio and Leon, I have enjoyed working with you two a lot, both because you are as passionate about crypto as I am, and because you taught me much about implementation and other practical aspects of crypto, which eventually helped me (at least, in my view) design better crypto. To all of you: I am proud of the works we have created together.

Fifth, I would like to thank the “Encryption for all” team for treating me as part of the team for several years. It has been both fun and useful to learn about other aspects of deploying crypto in practice. Hanna, Thea, Merel and Jorrit, you taught me much about the usability aspects around crypto, teaching me that making implementations user friendly is just as important as considering other security aspects. Daniel and Leon, we have spent quite some (very useful) hours in front of the whiteboard to come up with creative solutions to solve practical issues that are at play, which theoretical cryptographers may not typically take into account when designing crypto.

Sixth, I would like to thank the Geo Key Manager/Portunus team from Cloudflare for inviting me to be part of the Real-World Crypto talk and subsequent paper (published at USENIX ATC). Watson and Tanya, you taught me so much about deploying ABE in a distributed setting on a large scale and all the practical and industrial hurdles and bottlenecks that need to be overcome to do that. I am proud to have been part of this project and of the paper that we wrote together.

Seventh, I would like to thank dr. Joeri de Ruiter, for seeing through all the badly veiled self-doubt and hiring me as your PhD student. Despite my initial reservations (which manifested in me not applying for the position in the first place and being too honest during my job interview), choosing to do my PhD with you turned out to be one of the best decisions of my life. I am glad that, after you announced that you were going to leave, you persisted in helping me find a supervisor that would help me do the research I truly wanted to do. I am also glad that we stayed in touch over the years.

Eighth, I would like to thank all the (assistant/associate/full) professors that have inspired me to pursue a career in research. Dr. Wieb Bosma, with your talk about “the secret behind the perfect cup of coffee”, you planted the first seed that grew out to be my love for cryptography; you watered this seed when you supervised my bachelor thesis about lattice-based crypto. Dr. Bernd Souvignier, your courses were always some of my favorites, and our meeting about my “future plan for the next five years” was very insightful. I will never cease to cite that, during that meeting, I said that I did not know what I did want to do after my master, but I knew that I was not going to do a PhD. I changed my mind after a couple of months. I would also like to thank prof. dr. Peter Schwabe, for being one of the most enthusiastic teachers, who made a life of research seem accessible, and for helping out with implementation-related questions and finding a postdoc. Another enthusiastic teacher that I would like to thank is dr. Erik Poll. You were there for surprisingly many pivotal moments in my life: together with Peter, you encouraged me to do a computing science master in cyber security, you took part in my job interview with Joeri, and you helped me when I was looking for a job after my PhD. Lastly, I would like to thank dr. Berry Schoenmakers, for introducing me to the world of advanced cryptography and the beauty of theoretically sound security proofs. Ever since following your course, I have known that this particular subfield of cryptography is my passion, and it has motivated me to pursue a PhD in it.

Ninth, I would like to thank all of my office mates. For the longest period of time, I have shared an office with a part of the symmetric-crypto team: Yanis Belkheyar, Jonathan Fuchs, Koustabh Ghosh, Aldo Gunsing, and Alireza Mehrdad. You (and the rest of my reasonably long list of former office mates) have made the long days in the office a little more fun by joking around (among other things, about me being the "public-key crypto weed in the symmetric-crypto garden").

Tenth, I would like to thank all the people with whom I have had interesting discussions about crypto, teaching or other things, which includes many of the aforementioned people, and Łukasz Chmielewski, prof. dr. Cas Cremers, prof. dr. ir. Joan Daemen, Shanley Fijn, dr. Romain Gay, dr. Bernard van Gastel, Irma Haerkens, Thijs Heijligenberg, dr. Engelbert Hubbers, Daniël Kuijsters, Charlotte Lefevre, dr. Veelasha Moonsamy, Krijn Reijnders, dr. Simona Samardjiska, Amber Sprenkels, dr. Monika Trimoska, Janet Versluys, dr. Bas Westerbaan, dr. Bram Westerbaan, Ronny Wichers Schreur, dr. Freek Wiedijk, Thom Wiggers, the "Ask me Anything" crew at Eurocrypt 2021, the organizers of Crossfyre 2021, all students of Introduction to Cryptography and Introduction to Formal Reasoning, all other co-workers at the Digital Security group, and everyone else I may have (okay, probably have) forgotten to mention here. Special thanks go out to Engelbert and Freek, for teaching me a lot about teaching.

Eleventh, I would like to thank my new coworkers at the University of Wuppertal, for welcoming me as a postdoctoral researcher in your group. Tibor, I am so grateful that you have jumped through the necessary bureaucratic hoops to hire me, for all the opportunities that you are giving me to learn about new topics and academic activities, and for your support for my projects. Lin and Máté, thanks for being such great co-lecturers on the course "Provable Security", and Jonas, Pascal and Raphael, thanks for being great tutors. Together with Kai, you have helped me become a better teacher already during the span of one semester. Jan, Raphael and Tobias, thank you for being nice office mates and helping me out with various "new-employee" things. To all of you, as well as Amin, David, Denis, Moritz and Sebastian: I am looking forward to the rest of my postdoc.

Twelfth, I would like to thank some of the friends that I have made along the way: Denisa Greconici, Anna Guinet, Amber Sprenkels, Tanya Verma and more recently, Jonas Lehmann and Doreen Riepel. I appreciate about all of you that we can joke and mess around while also having more serious conversations about things. Especially during the rough parts of an academic career, it is important to have such friendships, and I thank all of you for helping me through those with your support and kindness.

Thirteenth, many special thanks go out to Jan Schoone, for proof-reading my entire (!) thesis, and for appreciating the best tag line for any cryptography course and teaching me about the relevance of Vigenère in this day and age. Usj btol fgvl bugic hbbyi pffhzx vyc agcir dy au TwX. Tealqlep jg eckxf nipfer aemv yvlvygcz hks hykbvyucw, Q qq nycm hk xpex moi qgnwevllxmm ps ibtgr cql dwe gmcgrhdmca, bgjpwugvt pfr bnfzigcgz ti ceb bug ymjkg kj hnniph lfimf, efkeeoerim tgk vcera, npu ddk mkyg whgamplcl fwgndkh wrs zkblrfjfqc.

Last but in no certain way the least, I would like to thank my friends and family, for always being there to support me through any endeavors, including my academic ones. In particular, I mention my aunts and uncles, Jeannette & Luc, and Henk & Paola, and my grandma, “oma Venema”, who sadly passed away in my second year of my bachelor and therefore never gets to see this thesis. *Ik hoop dat je trots bent op me en op wat ik bereikt heb.* Moreover, I would like to thank my brother, Hans, and my parents, Wim & Marie Louise, for (attaining) basically everything. For over thirty years, you have provided me with the right environment, which allowed me to grow up in the way I did: extremely stubborn, endlessly fascinated by how things worked (which may or may not include door handles), constantly being pranked (apparently, I do not have a fourth name), slightly skeptical (probably, because of the aforementioned pranking), and always asking questions. You taught me early on in life that it is important to do what you love, which motivated me to not only pursue a PhD in cryptography but also persist until the end. Without you, this thesis would not exist, so I am dedicating it to you. Thank you!

Lichtenvoorde, the Netherlands
Wuppertal, Germany

Contents

Chapter 1

Introduction

Would it not be great to be able to encrypt a document so that it can only be decrypted and accessed by e.g., all epidemiologists of the Radboud University Medical Center (Radboudumc for short), or any patient who is between 18 and 65 years old, or a specific patient called Alice, who was born on March 14, 2007 and currently resides in Nijmegen? An encryption scheme that provides this functionality would enable individuals to share data in a secure yet flexible way.

Traditional public-key encryption does not effectively provide this functionality. Typically, it is used to allow access to confidential data to one particular entity, which must be known at the time the data are encrypted (by using its public key). Only this entity can later access the data by decrypting the associated ciphertext (by using its secret key). Attribute-based encryption (ABE) [SW05] is a form of public-key encryption in which the key pairs are associated with *attributes* rather than individual users or entities. For instance, in ciphertext-policy ABE [BSW07], the encrypting user can decide who gets access to the data by specifying a policy during encryption, e.g., any "epidemiologist" at the "Radboudumc" or any "patient" with an "age in the range $[18, 65]$". The ability to decrypt the resulting ciphertext is then determined by the attributes owned by the decrypting user, who must be an "epidemiologist" at the "Radboudumc" or be a "patient" with an "age in the range $[18, 65]$". Thus, data that are encrypted with ABE can be accessed by multiple authorized users, making ABE inherently more flexible than traditional public-key encryption.

Based on its functionality, attribute-based encryption can cryptographically implement fine-grained access control on data [GPSW06a, BSW07, PTMW10]. Like most traditional access control mechanisms [SCFY96, HFK$^+$19], it relies on a trusted third party (TTP). This TTP enforces access control on the data, by granting or denying users who wish to access those data. Nevertheless, compared to those traditional access control mechanisms, ABE requires less trust in and less reliance on this TTP. First, because ABE is a cryptographic primitive, the data are encrypted and can thus be stored by an entity that is not necessarily trusted to securely enforce access control or to access the data at all. To ensure that the data are always accessible, they can be stored on (multiple) high-availability servers. In contrast, traditional access control mechanisms typically require the data to be stored by the same TTP that enforces access on those data. This TTP therefore needs to be trusted to enforce access and provide availability. Second, as we will show later, ABE does not necessarily require the TTP to be online during each access request, allowing users to act more autonomously. Importantly, minimizing the role of the TTP in the enforcement of access control in this way also fosters the availability of the data.

In ABE, the key generation authority (KGA) constitutes such a trusted third party. The KGA generates the master public keys and the master secret keys, from which it derives secret keys and issues these to eligible users. Once the users have received secret keys, they can decrypt any ciphertexts for which they have a suitable key. In turn, access to data can be managed by the data owner using encryption. Then, access to these data is indirectly enforced by the KGA, which provides only eligible users with keys that can decrypt the resulting ciphertext. In addition, some variants of ABE, called multi-authority ABE (MA-ABE) [Cha07], support the employment of multiple (possibly mutually distrusting) KGAs. This allows for the secure enforcement of access control in multiple-domain or cross-organizational settings, e.g., electronic health record (EHR) systems involving hospitals and insurance companies. Owing to all these advantages, ABE has attracted much interest from the practical community [ETS18a, ETS18b, KL10, SRGS12, BSS$^+$22, LVV$^+$23].

In particular, we want to highlight the standardization efforts of the European Telecommunications Standards Institute (ETSI). In 2018, ETSI published two technical reports on ABE [ETS18a, ETS18b], describing various use cases in which ABE can be used to cryptographically enforce access control. For each use case, ETSI lists the high-level requirements that ABE needs to satisfy, e.g., regarding the types of policies that need to be supported. Furthermore, the relevant entities and their roles in the use case are described in detail. From this description, the computational resources of each entity can be estimated, which can be subsequently used to set up requirements for the scheme's efficiency. In general, we observe that these requirements may vary. For example, the Internet of Things use case assumes the encryption devices to be resource constrained and therefore benefits from ABE with fast encryption, while the federated WLAN use case may benefit from a fast decryption.

1.1 Our goal

As the title of this thesis suggests, we focus on attaining basically everything in attribute-based encryption. More specifically, we mean that we aspire to achieve all core properties needed for a specific practical setting as efficiently as possible, to maximize the advantages of ABE. We aim to do this by simplifying the design of practical ABE schemes, specifically via pair encodings, and by exploring (possibly hidden) dependencies among various existing constructions. First, we investigate which properties may be required for practice, and which can already be satisfied efficiently (in Chapter 2). This investigation reveals that there is still some room for improvement, especially with regard to the efficiency of existing schemes. We pose several future directions that address these improvements. In the remainder of this thesis, we focus on several of these future directions, helping ABE to become more practical. In particular, we aim to support any property as generically and as efficiently as possible. A property is generically supported if some “basic” scheme with certain security properties can be generically transformed into a provably secure scheme with that property. By supporting properties generically, a suitable scheme can be efficiently constructed for any specific setting, such that it satisfies only the necessary properties. This is more favorable than using a scheme satisfying all properties that could be necessary in practice, because the support of certain properties often incurs a performance penalty. For example, supporting negations in the policies may be costly (Section 2.5.7). Consequently, the scheme satisfying all properties may not be the most efficient choice for that specific setting. In this thesis, we investigate and propose some basic schemes that satisfy several desirable properties. If necessary, these schemes can be securely extended to satisfy other desirable properties that incur a performance penalty. In this way, the most efficient scheme satisfying all necessary properties can be chosen for some specific practical setting.

1.2 Structure

This thesis consists of three parts (Figure 1.1):

- **Part I: Introduction to ABE:** In this part, we introduce ABE and systematize its core properties (Chapter 2), and we provide other necessary theoretical background required for the remainder of this thesis (Chapter 3).

- **Part II: The pair encodings framework:** In this part, we introduce the pair encodings framework (Chapter 4). This is a well-known framework that considers an abstraction of a large class of pairing-based ABE to simplify the design of secure schemes: pair encodings. Additionally, this framework generically supports several properties that are otherwise difficult to attain. We extend the framework with a new compiler, which efficiently transforms any secure pair encoding into a selectively secure ABE scheme. We also use pair encodings to simplify the cryptanalysis of ABE schemes, which we systematically apply to several existing schemes (Chapter 5). Lastly, we use the common structure of schemes that can be captured in the pair encodings framework to simplify the fair benchmarking of multiple ABE schemes (Chapter 6).

- **Part III: New constructions and generic transformations:** In this part, we give several new constructions and generic transformations in the pair encodings framework. First, we introduce GLUE, a new ABE scheme with flexible efficiency trade-offs between the encryption and decryption algorithms that can achieve all necessary core properties (Chapter 7). Practitioners can fine-tune GLUE to achieve the most desirable efficiency trade-off between encryption and

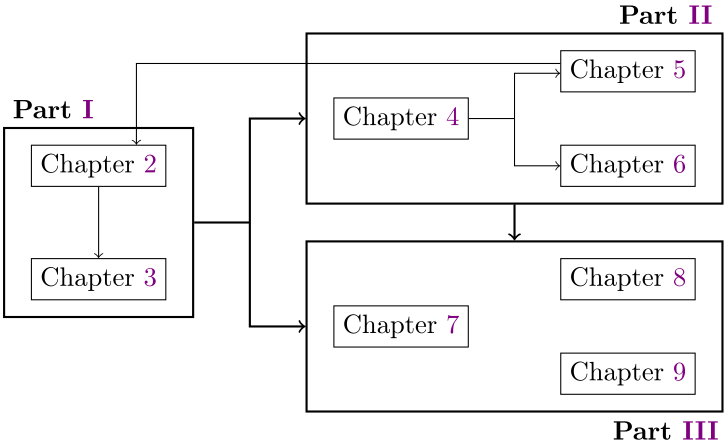


Figure 1.1. Overview of this thesis. The arrows indicate the dependencies among the parts and chapters.

decryption for a specific setting. In particular, this scheme can attain the most efficient decryption compared to existing schemes with the same properties. Second, we introduce TinyABE, another ABE scheme with flexible efficiency trade-offs, but between the sizes of the master public key and the ciphertexts. TinyABE is specifically designed for settings involving the Internet of Things (Chapter 8). In particular, TinyABE can be configured such that the keys and ciphertexts are small enough for the relevant devices, and such that encryption is really fast. Finally, we introduce new generic transformations for chosen-ciphertext security. These transformations efficiently transform any scheme in the pair encodings framework—which are only secure against chosen-plaintext attacks—to a scheme that is secure against chosen-ciphertext attacks, and are subsequently sufficiently secure for practice (Chapter 9).

1.3 Contributions

This thesis is based on multiple papers to which I contributed. Specifically, each chapter corresponds roughly to one paper.

Chapter 2: This chapter is based on the paper

M. Venema, G. Alpár, and J.-H. Hoepman. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. *Des. Codes Cryptogr.*, 91(1):165–220, 2023 of which I am the main author. My co-authors helped me with the presentation of the contents, mostly through discussions and proofreading.

Chapter 4: This chapter is partly based on the paper

> M. Venema. A practical compiler for attribute-based encryption: New decentralized constructions and more. In M. Rosulek, editor, *CT-RSA*, volume 13871 of *LNCS*, pages 132–159. Springer, 2023

of which I am the sole author. Compared to the paper, this chapter contains more information on the pair encodings framework, as well as examples of schemes and proofs.

Chapter 5: This chapter is based on the paper

> M. Venema and G. Alpár. A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. In K. G. Paterson, editor, *CT-RSA*, volume 12704 of *LNCS*, pages 100–125. Springer, 2021

of which I am the main author. My supervisor helped me with the presentation of the contents, mostly through discussions and proofreading. This paper also led to the collaborative implementation of several attacks, which we presented in a briefing at Black Hat Europe:

> A. de la Piedra and M. Venema. Practical attacks against attribute-based encryption. At *Black Hat Europe*, 2021

Chapter 6: This chapter is based on the paper

> A. de la Piedra, M. Venema, and G. Alpár. ABE squared: Accurately benchmarking efficiency of attribute-based encryption. *TCHES*, 2022(2):192–239, 2022

which is a collaborative effort. For this paper, I have provided most of the theoretical contributions, including the type-conversion methods, the specifications of the type-converted schemes and the analysis of the benchmarks.

Chapter 7: This chapter is based on the paper

> M. Venema and G. Alpár. GLUE: generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. In A. Boldyreva and V. Kolesnikov, editors, *PKC*, volume 13940 of *LNCS*, pages 652–682. Springer, 2023

of which I am the main author. My supervisor helped me with the presentation of the contents, mostly through discussions and proofreading.

Chapter 8: This chapter is based on the paper

> M. Venema and G. Alpár. TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. In L. Batina and J. Daemen, editors, *AFRICACRYPT*, volume 13503 of *LNCS*, pages 103–129. Springer, 2022

of which I am the main author. My supervisor helped me with the presentation of the contents, mostly through discussions and proofreading.

Chapter 9: This chapter is based on the paper

> M. Venema and L. Botros. Efficient and generic transformations for chosen-ciphertext secure predicate encryption. Cryptology ePrint Archive, Paper 2022/1436, 2022

which is currently under submission, and of which I am the main author. My co-author proofread the paper, and implemented the schemes. Together, we wrote the performance analysis section.

1.4 Research data management

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands.[1] The code for Chapters 6, 7, 8 and 9 is available in the public domain under an open-source license, and can be accessed at `https://mtcvenema.nl/` or accessed as a single archive at `https://doi.org/10.5281/zenodo.8163274`. The code is also accessible in the following repositories.

Chapter 6: `https://github.com/abecryptools/abe_squared`

Chapter 7: `https://github.com/mtcvenema/glue`

Chapter 8: `https://github.com/mtcvenema/tinyabe`

Chapter 9: `https://github.com/leonbotros/pe_cca`

[1] `https://www.ru.nl/icis/research-data-management/`, last accessed October 27th, 2022.

Part I

Introduction to ABE

Chapter 2

Systematizing core properties of pairing-based ABE to uncover remaining challenges in enforcing access control in practice

Much progress has been made in the last two decades in pairing-based ABE schemes, owing to their versatility and efficiency. In fact, it is possible to support most core properties under strong security guarantees, while incurring acceptable storage and computational costs. It is therefore a good time to ask ourselves whether pairing-based ABE has reached its full practical potential. To answer this question, we provide a comprehensive systematized overview of various existing pairing-based ABE schemes and their core properties in this chapter. We also investigate the relationship between these core properties and real-world access control requirements. We show that a few challenges remain, that must be overcome for ABE to reach its full potential as a mechanism to implement efficient and secure access control in practice.

2.1 Introduction

For the past two decades, the theoretical cryptography community has made much progress in pairing-based ABE [SW05], leading to many publications at prominent conferences, and thus establishing itself as an important and popular research topic. Many schemes have been proposed that vary significantly in the core properties, which determine the basic functionality, efficiency, and security. Some examples of core properties include the level of fine-grainedness of the access policies[1], the performance of the scheme, and the underlying cryptographic assumptions. Some core properties

[1] For example, can the scheme support Boolean formulas such as (“doctor” OR “nurse”) AND “Radboudumc”, and can it support negations, e.g., NOT “oncology department”?

may be more desirable for practical applications than others. Ideally, a scheme that supports most or all of these properties is used for these applications. Nowadays, pairing-based ABE has reached a level of maturity such that most of the desirable core properties can be achieved simultaneously, whilst attaining both strong security guarantees as well as acceptable storage and computational performance [KW19b, Att19, LL20a, AT20]. Given these developments, a natural question that arises is:

To what extent has pairing-based ABE reached its full practical potential?

In this chapter, we work towards answering this question. To this end, we focus on the core properties of ABE, as they strongly influence the basic functionality and efficiency of any pairing-based ABE scheme in practice. Concretely, we identify the main core properties as defined through the years, which we will list briefly in Section 2.1.3 and discuss in more detail in Section 2.3. For these properties, we provide unified definitions, and give an overview of prominent schemes and the properties they satisfy. By considering an example of a practical setting, we argue which properties are desirable. Then, we consider if and how these desirable properties can be achieved simultaneously. To obtain a better understanding of the interplay between these various properties, we analyze how these are realized in the existing pairing-based schemes and whether they are compatible with one another. Furthermore, provided that they are compatible, we consider the effect of satisfying all these properties simultaneously on other practical aspects, such as efficiency and availability. Along the way, we uncover a number of remaining challenges, which we pose as directions for future research. We encourage the (theoretical) community to explore these, as it could make ABE even more practical, mitigating the disadvantages of ABE compared to other primitives for implementing access control, whilst amplifying its advantages.

To place properties specific to ABE in a practical context, we will first describe a large-scale medical scenario in which access control to data is enforced through cryptography. In general, the implemented access control mechanism should support properties such as confidentiality, integrity, and availability [SS94]. In particular, for such large-scale real-world settings, it is important that these properties can be simultaneously guaranteed in the best way possible. On the one hand, properties such as confidentiality and even integrity have been considered at length in the context of ABE. On the other hand, properties such as availability have been treated in much less detail. In this work, we also investigate the relationship between the ABE properties and these three real-world properties. Specifically, we will introduce the new notion of *resilience* in the context of ABE to foster a deeper understanding of availability in real-world settings using ABE. Roughly speaking, resilient ABE minimizes the required interaction between the user and the KGA. By minimizing the required interaction, the user is less dependent on the KGA's availability. Furthermore, we consider how better availability properties can be achieved in multi-authority ABE (MA-ABE) by analyzing existing work. MA-ABE with such availability properties can provide advantages in implementing access control in the multiple-domain setting compared to other solutions.

Ultimately, the goal of our analyses is to help pairing-based ABE reach its full potential. By considering the interplay between the core properties as well as their relationship with real-world (security) properties, we strive to obtain as much functionality, security and efficiency as possible. At the same time, we want to highlight the advantages of ABE in the implementation of access control compared to other solutions. Therefore, we pose several directions for future research that help ABE become even more practical.

2.1.1 Our contribution

Our contribution in this chapter is fourfold.

- **Systematization of knowledge:** We provide an extensive overview and systematization of the core properties of ABE. To this end, we analyze over fifty important pairing-based ABE schemes—each published at a prominent conference—and their properties.
- **Interplay of properties:** We analyze how the core properties are realized to understand whether and how they can be achieved simultaneously. Furthermore, we analyze the influence of these core properties on real-world security properties such as availability.
- **New insights:** Sometimes, this analysis leads to deeper, novel insights, explicitly conveyed as *observations* throughout this chapter.
- **New research directions:** Based on the analysis, the systematization and observations, we identify several *directions* for future research that are relevant to improve the practical advantages of ABE. We encourage the cryptographic community to explore these directions.

To the best of our knowledge, this is the first in-depth overview of pairing-based ABE that discusses the interplay between different core properties as well as their practical impact.

2.1.2 Scope and approach

We focus primarily on pairing-based attribute-based encryption for (non-)monotone span programs (which include Boolean formulas[2]), although the first sections of this chapter are general in the sense that they describe a broader class of ABE. The main reason for our focus on pairing-based schemes is that they are more established, efficient and practical than works based on e.g., multilinear maps [GGH+13] or post-quantum assumptions [Boy13]. While these are interesting in their own right, we

[2] An example of a monotone Boolean formula is (“doctor” OR “nurse”) AND “Radboudumc”. The scheme also supports non-monotone formulas if it supports negations, e.g., NOT “oncology department”.

believe that it would be more suitable to address concerns specific to these subfields once they have reached a similar maturity as pairing-based ABE. Furthermore, we consider ABE for (non-)monotone span programs, because these provide sufficient expressivity that is expected in access control mechanisms. ABE for more fine-grained classes of expressivity such as circuits [GVW15] or ABE that additionally supports inner-product computations [ACGU20] are typically also less efficient or are secure in weaker models. Finally, we focus primarily on the core properties of ABE, and therefore do not extensively discuss other practical extensions, such as revocation (and see more examples in Section 2.10). While these extensions are important and may provide some interesting benefits in practice, we leave any systematized analysis of these for future work. We have also excluded any broken schemes (e.g., Chapter 5) or any schemes that lack a proper security analysis.

In this chapter, we do not necessarily strive for formality. On the contrary, one of our aims is to make the field of ABE more accessible to the practical community. In particular, we have moved away from the heavy notation and complicated formal concepts in the newer works. Nevertheless, we have included their accomplishments. This chapter does contain some formal definitions and models, but only of the most common and established concepts.

For convenience, we refer to the schemes by concatenating the first letter of each author's surname with the last two digits of the year of publication, e.g., the scheme published by Rouselakis and Waters in 2013 [RW13] is referred to as the RW13 scheme.

2.1.3 The core properties of ABE

One of the main components of this chapter is that we explore and analyze the core properties of ABE (defined and discussed in more detail in Section 2.3), which are

- the level of expressivity the scheme supports in its access policies;
- whether the scheme is key-policy or ciphertext-policy based;
- whether the scheme supports small or large universes, i.e., which distinguishes between whether it can support any string as attribute or not;
- whether the scheme is bounded in any of the parameters or not.

In addition, we analyze multi-authority ABE, which employs multiple key generation authorities. We highlight this particular extension of ABE because of its unique advantages in enforcing access control in practice.

2.1.4 Target audience and goal

The target audience for this chapter is wide, and includes practitioners, cryptographic engineers and cryptographers (both newcomers to the field of ABE and experts). In part, the reason for this is that our ultimate goal is to investigate to what extent pairing-based ABE has reached its full practical potential with respect to the core properties. To this end, we also explore the basics of ABE and the requirements for ABE when it is employed in practice. Our goal is therefore threefold:

- introducing new cryptographers to the area of ABE;
- informing practitioners about the potential of ABE;
- encouraging experts in ABE (and related areas) to address the future directions.

Note that some of those future directions may be more general than others, in the sense that they extend to other fields of cryptography as well. For instance, they could also pertain to related areas in pairing-based cryptography, or exploring these directions could require expertise in related fields such as cryptographic engineering.

2.1.5 Organization

Because of the wide range of aspects that this chapter considers, we make the dependencies among the sections explicit. In Figure 2.1, we illustrate several possible ways to read this chapter. In particular,

- **Sections 2.2, 2.3 and 2.4** describe the general concepts of ABE in a practical context, which might be of interest to any readers who want to know more about the practical advantages of ABE in general, including novel (ABE) cryptographers and practitioners. These concepts are general in the sense that they are applicable not only to pairing-based but to every existing ABE scheme;
- **Sections 2.5, 2.6 and 2.7** discuss pairing-based ABE with respect to the general concepts, as well as the storage and computational efficiency of pairing-based ABE, which may be of interest to cryptographers and cryptographic engineers;
- **Section 2.8** focuses on multi-authority ABE, which may be of interest to cryptographers, and practitioners who wish to deploy ABE in the multiple-domain setting. In this section, we re-contextualize the notions of distributed and decentralized, and systematically classify existing schemes;
- **Section 2.9** considers the availability properties of ABE by putting forth the new notion of resilience. This section may be of interest to practitioners and cryptographers;
- **Section 2.10** discusses some additional functionality, which may be of interest to practitioners;
- **Section 2.11** covers our taxonomy, which may be of interest to practitioners and novel cryptographers;
- **Section 2.12** concludes this chapter.

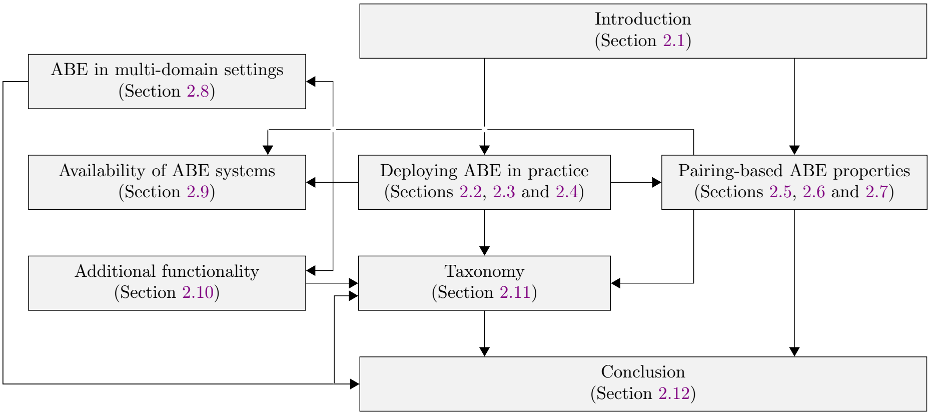

Introduction
(Section 2.1)
ABE in multi-domain settings
(Section 2.8)
Availability of ABE systems
(Section 2.9)
Deploying ABE in practice
(Sections 2.2, 2.3 and 2.4)
Pairing-based ABE properties
(Sections 2.5, 2.6 and 2.7)
Additional functionality
(Section 2.10)
Taxonomy
(Section 2.11)
Conclusion
(Section 2.12)

Figure 2.1. The general structure of this chapter.

2.2 Practical motivation: access control

ABE allows for the secure and practical enforcement of fine-grained access control on data. On the one hand, ABE ensures that the data are encrypted, such that the storing (or transporting) entity cannot read the plaintext data. On the other hand, by its functionality, ABE ensures that access control can be enforced externally by a trusted entity. This allows data to be securely stored and managed on untrusted platforms, such as the cloud or servers managed by other (untrusted) entities. By extension, this also simplifies the enforcement of access control on data in settings where multiple mutually distrusting stakeholders are at play.

Example use case: electronic health records. As an example, consider electronic health record (EHR) systems, planned to be used on a large scale [HSN08]. While the use of EHR systems simplifies the sharing of health records across organizations, jurisdictions or countries, it also increases the risk of infringing upon the privacy rights of individuals [HMCB11]. To address these privacy concerns, existing solutions often use access control to manage access to the data. It varies, though, which access control model is used, who defines and assigns user roles and access policies, and who grants access to the data [ASLT13]. Generally, the most common access control models in such health settings are role-based access control (RBAC) [SCFY96] and attribute-based access control (ABAC) [HFK$^+$19]. In addition, to ensure confidentiality, these solutions require the data to be encrypted, though security problems may still arise. In practice, the keys are frequently stored by the same entity that stores the data [ASLT13]. Effectively, the entity storing the keys enforces access control on the data, and must therefore be highly trusted. Also, this entity needs to be available when access is requested. In multiple-domain settings—in which data pertaining to one individual may be stored or produced by different entities—such a degree of trust may be problematic; especially, if each entity wants to enforce access control regardless of where the data are stored. We show that ABE can provide a secure solution, even in the multiple-domain setting.

Issues in traditional access control mechanisms. We briefly illustrate what such an EHR system may look like, and what availability and scalability issues may occur when traditional access control mechanisms are applied. Consider a medical scenario with a hospital and an insurance company that want to use an EHR system using a traditional form of access control. A patient at the hospital may want to share some of her private data, stored at the hospital, with both her doctor and an employee at her insurance company. Some employee at the insurance company can then request access to the data by contacting the server at the hospital. To grant access, the server needs to know the access policy and contact the insurance company to verify whether the requesting user is an employee. Hence, during an access request, all relevant entities (e.g., the hospital and the insurance company) need to be available. In more complex scenarios, with access policies involving many entities, possibly from different domains, the required interaction among these entities scales up, amplifying any availability issues.

Using ABE to mitigate availability and scalability issues. ABE provides a practical and secure solution, and mitigates the potential availability and scalability issues. Specifically, the KGA indirectly enforces access control by generating the public and secret keys. Furthermore, the ciphertext-policy variant of ABE [BSW07] allows the encrypting users to decide who is allowed access by specifying the access policy. For instance, the patient in our example could encrypt her data under the policy (“hospital” AND “doctor”) OR (“insurance company” AND “employee”) and store the resulting ciphertext on the hospital server. Other users, e.g., the employee in our example, can only successfully decrypt the ciphertext, if they have a set of attributes that satisfies the access policy. In contrast to non-cryptographic access control mechanisms, ABE only requires the enforcing entities, in this case the KGAs, to be available when users request secret keys. A user is typically assumed to request a secret key only once[3]: when the user enters the system (and potentially, when the user obtains new attributes or loses them). From that point forward, the user can access any data for which she is authorized while requiring no interaction with and between any policy-enforcing entities (only the entities who store the data), effectively mitigating any availability issues. In addition, the use of cryptography ensures that the data can be stored anywhere, and thus can be shared across various domains.

[3] Note that, depending on how keys may expire or be revoked (Section 2.10.2), the user may need to request keys more than once.

Multi-authority ABE for multiple-domain settings. In the multiple-domain setting, multi-authority (MA) ABE [Cha07] can be used. In this variant of ABE, the role of the KGA is shared by multiple entities called KGAs or *authorities*. Each KGA securely manages a unique set of attributes. Unfortunately, not all MA-ABE schemes are "decentralized" enough. In particular, users need to interact (at some point) with all KGAs associated with a policy. We illustrate the issue by considering an example in more traditional access control mechanisms. For instance, consider an access policy defined over attributes managed by several TTPs. Ideally, the *access control decision*, i.e., the decision to grant or deny access, is made by verifying with only the relevant TTPs whether the user satisfies the access policy. For example, during the access request of the user in our previous example, the insurance company confirms the employee status to the hospital server, which stores the data. It is then not needed for the server to also check with the hospital whether the user is a doctor, as the access policy is already satisfied. In contrast, most instantiations of MA-ABE require the decrypting user to request keys from all KGAs associated with the access policy, including those KGAs for which the user may not have any attributes. Hence, in the example, the decision to grant access would need to be verified (although indirectly) with both the insurance company and the hospital. Subsequently, the decrypting user may need to interact with possibly many KGAs, which need to be online if the user does not have any keys yet. This negatively impacts the scalability and availability of the system.

Decentralized versus distributed access control decisions. To capture this difference in decision making in the multiple-domain setting—i.e., when multiple entities are associated with a policy—we distinguish concretely between a decentralized and a distributed access control decision. If the access control mechanism allows that the decision to grant access is made by verifying with only the relevant TTPs whether the requesting user satisfies the policy, then we call it decentralized. If all entities need to be contacted—effectively distributing the decision—we call it distributed. Essentially, this distinction between decentralized and distributed is determined by the level of autonomy or independence of the entities. This distinction is roughly in line with the terminology in (algorithmic) decision making in systems [MMM05, dLea10]. In those works, decentralized systems do not require that the nodes have system-wide information or need to communicate with all the nodes in the system to correctly make decisions. The nodes in decentralized systems subsequently enjoy a level of autonomy and independence in this process, which is in line with our definition of decentralized access control decisions. In principle, distributed access control decisions can be seen as the overarching term of any system that allows multiple TTPs to make a decision [dLea10]. However, we shall use the term distributed to clearly distinguish distributed but non-decentralized access control decisions from decentralized access control decisions. In addition, for an MA-ABE scheme to be distributed or decentralized, we require that the KGAs do not need to trust or rely on one another for confidentiality either. This is especially useful in settings in which some entities have conflicting interests. For instance, consider adding another insurance company in our medical scenario.

User independence and authority-dependence minimization. To express some of the properties that we informally discussed, we formulate the following two implicit properties that are generally important for practical access control mechanisms.

- **User independence:** authorized users can obtain access even if not all of the authorities are available at the time of an access request.
- **Authority-dependence minimization:** authorized users only need to rely on the authorities associated with their set of attributes. If it satisfies the access policy, they can gain access without needing to interact with other authorities. In addition, the authorities do not have to trust one another to correctly and securely enforce access control.

In contrast to traditional RBAC and ABAC mechanisms, ABE provides a high level of user independence by its functionality, because users can independently obtain access by decrypting ciphertexts for which they are authorized once they have a secret key, typically retrieved at an earlier point in time. Therefore, they do not need to interact with any policy-enforcing entities, which may be unavailable at the time of an access request. To maximize the level of user independence, we assume that the user only needs to request a secret key once (or possibly once per some time interval, to support revocation (Section 2.10.2)). In Section 2.9, we will put forth the new notion of resilience in the context of ABE to foster such user independence. We also show that some MA-ABE schemes satisfy our strong notion of authority-dependence minimization. We call these schemes decentralized (MA-)ABE schemes. In decentralized ABE, the users only need to interact with those authorities for which they have attributes. In contrast, many MA-ABE schemes require the users have received keys from all authorities associated with the access policy enforced on some ciphertext they are trying to decrypt. If one of these authorities is permanently unavailable, e.g., because an insurance company went bankrupt, the user is not able to decrypt the ciphertext anymore despite possibly satisfying the policy. Decentralized schemes do not have this issue, and are thus especially attractive for implementing access control in the multiple-domain setting.

2.3 Attribute-based encryption – core properties

2.3.1 Attributes and access structures

First, we explain what attributes are, how they are represented, and how they are used as a building block of access structures. An *attribute* is defined as a type-value pair, e.g., the type of the attribute could be "profession" and its value could be "doctor" [HFK+19]. In the examples in this work, we often represent the attributes as strings consisting only of the attribute value (if the type is clear from the context). However, in practice, it is recommendable or even needed (Section 2.5.7) to include the attribute type in the string to avoid confusion. For instance, "doctor" may also refer to a person who holds a PhD in computer science, but in the context of a medical setting, it is unlikely that this meaning is used.

Attributes are an important building block of *access structures*, or *policies*, which specify which attributes need to be possessed by a user in order to be granted access to a certain resource (in our case: data). Typically, access structures are expressed as Boolean formulas (including threshold functions). For instance, the policy "doctor" ∧ "Radboudumc" specifies that access is granted to all doctors who work at the Radboudumc. Access structures can be formally defined as follows.

Definition 2.1: Access structures [Bei96]

Let $\{\mathrm{att}_1, ..., \mathrm{att}_n\}$ be a set of attributes. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{\mathrm{att}_1, ..., \mathrm{att}_n\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets that are not in $\mathbb{A}$ are called the unauthorized sets.

Note that any access policy expressed as a Boolean formula can also be expressed as a set (like in the definition). For instance, consider the formula "(doctor ∨ nurse) ∧ Radboudumc". The associated access structure $\mathbb{A}$ consists of all subsets of attributes in the system that contain the sets $\{\mathrm{doctor}, \mathrm{Radboudumc}\}$ or $\{\mathrm{nurse}, \mathrm{Radboudumc}\}$.

Two important aspects in access structures are the *expressivity* and the *monotonicity*, which collectively determine the level of fine-grainedness of a scheme. Informally speaking, expressivity concerns whether the use of all Boolean formulas consisting of conjunctions, disjunctions and threshold functions [GPSW06a] is allowed. For example, "doctor" AND "Radboudumc" is a conjunction, "doctor" OR "nurse" is a disjunction, and "at least two of the five given genetic markers" is a threshold function. Furthermore, monotonicity concerns whether the use of negations, e.g., NOT "doctor", is allowed. Monotone access structures do not allow the use of negations in the formula, while non-monotone access structures do allow it. In itself, monotonicity can technically be characterized as an expressivity aspect. However, we treat it as a separate feature, because negations are typically supported using different techniques than the other expressivity features.

Definition 2.2: Monotone access structures [Bei96]

An access structure $\mathbb{A} \subseteq 2^{\{\mathrm{att}_1,...,\mathrm{att}_n\}}$ is monotone, if for all $B, C \subseteq \{\mathrm{att}_1, ..., \mathrm{att}_n\}$ holds that if $B \in \mathbb{A}$ and $B \subseteq C$, then also $C \in \mathbb{A}$.

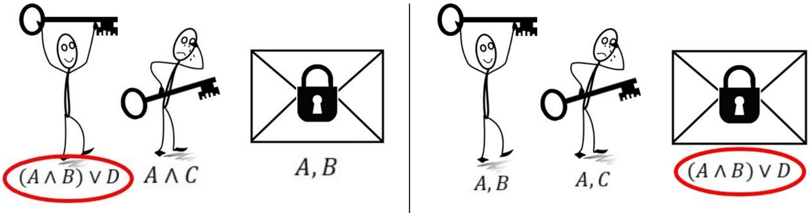
Existing schemes have varying levels of expressivity and monotonicity. The least expressive policies are those that only support *AND-gates* (or: conjunctions) [CN07]. More expressive structures support a single *threshold function*, which consists of a set of attributes and a threshold (smaller than the size of this set) to indicate the minimal number of these attributes that needs to be in the user's possession [SW05]. The most expressive access structures are *(non-)monotone span programs* ((N)MSP), which support any formulas using conjunctions, disjunctions and threshold functions [GPSW06a, BSW07]. Here, the distinction between monotone and non-monotone span programs depends on the monotonicity of the access structures. Although most existing schemes support monotone access structures, it is possible to support non-monotone access structures, which we discuss in more detail in Section 2.5.7. To implement access control on data in line with RBAC [SCFY96] or ABAC [HFK$^+$19], it is paramount that a scheme supports any Boolean formulas, including conjunctions, disjunctions and negations. Furthermore, the support of negations readily allows for the support of revocation (Section 2.10.2). It is therefore desirable that the scheme supports negations.

However, depending on the practical setting and the type of attribute, non-monotonicity may not be realistically achievable. In particular, for some attribute types, it may be difficult to ascertain whether a user does not possess certain values, especially when a user can possess multiple values. For instance, doctors may work at multiple departments or be patients at the hospital where they work. For those attribute types, it could therefore be recommendable not to use negations. To ensure that negations are not used, the KGA could communicate on the application level that those types should not be used by encrypting users.

2.3.2 Key-policy and ciphertext-policy ABE

In ABE, access structures—also known as policies—can be enforced either on the secret keys or the ciphertexts. In *key-policy attribute-based encryption* (KP-ABE) [GPSW06a], the access policies are enforced on the secret keys. These policies are specified by the key generation authority (KGA), and subsequently embedded cryptographically in the secret keys that are distributed to eligible users. The encrypting user can in turn associate a set of attributes with the ciphertext. This ciphertext can only be decrypted by users who have a secret key associated with an access policy satisfied by the set. Conversely, in *ciphertext-policy attribute-based encryption* (CP-ABE) [BSW07], the access policies are enforced on the ciphertexts. In particular, the access policies are specified by the encrypting user. The KGA generates secret keys associated with a set of attributes that the user possesses. Subsequently, a decrypting user can decrypt a ciphertext if she has a secret key that is associated with an attribute set that satisfies the policy. We call such a key an *authorized* secret key. Figure 2.2 illustrates the distinction between KP-ABE and CP-ABE with an example.

We formally define KP-ABE and CP-ABE by defining the notion of *predicate encryption* [KSW08], which is a more general cryptographic primitive that includes



(a) KP-ABE (b) CP-ABE

Figure 2.2. Key-policy versus ciphertext-policy ABE. In the examples, the access structure is either associated with the keys (i.e., the persons holding a key) or the ciphertext (i.e., the locked envelope). In each example, the person on the left is happy, because he can decrypt the ciphertext, while the person on the right is sad, because he cannot.

KP-ABE and CP-ABE as special cases[4]. Specifically, it is defined for any predicate P, where $P\colon \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}$ is a function that takes as input any pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$, and outputs 1 exactly when the predicate is satisfied, i.e., $P(x, y) = 1$. For KP-ABE, $\mathcal{X}$ is the collection of all attributes, and $\mathcal{Y}$ the collection of all policies, and $P(x, y) = 1$ if and only if the set x satisfies the policy y. Conversely, for CP-ABE, $\mathcal{X}$ is the collection of policies and $\mathcal{Y}$ the collection of attribute sets.

Definition 2.3: Predicate encryption (PE) [AC17b]

A predicate encryption scheme for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}$, with some key generation authority (KGA), users and a universe[†] of attributes $\mathcal{U}$ consists of four algorithms:

- $\mathsf{Setup}(\lambda) \rightarrow (\mathrm{MPK}, \mathrm{MSK})$: On input the security parameter λ, this randomized algorithm, executed by the KGA, generates the domain parameters, the master public key MPK and the master secret key MSK.
- $\mathsf{KeyGen}(\mathrm{MSK}, y) \rightarrow \mathrm{SK}_y$: On input the master secret key MSK and $y \in \mathcal{Y}$, this randomized algorithm, executed by the KGA, generates a secret key SK_y.
- $\mathsf{Encrypt}(\mathrm{MPK}, x, M) \rightarrow \mathrm{CT}_x$: On input the master public key MPK, some $x \in \mathcal{X}$ and message M, this randomized algorithm, executed by the encrypting user, generates a ciphertext CT_x.

[4] Our definition of predicate encryption is in line with [AC17b], which is more general than in some other works [KW19a, Att19]. In those works, predicate encryption requires x to be hidden. We will use the notion of attribute hiding (Section 2.10.3) to refer to this additional property.

- Decrypt(MPK, SK_y, CT_x) → M: On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P(x, y) = 1$, then it returns M. Otherwise, it returns an error message.

A scheme is called *correct* if decryption of a ciphertext with an authorized key succeeds with overwhelmingly high probability. The scheme is called *secure* if decryption of a ciphertext with any number of unauthorized keys fails with overwhelmingly high probability. We discuss the notion of security in more detail in Section 2.4.

[†] A universe of attributes is the set of all attributes used in the system (Section 2.3.3).

KP-ABE and CP-ABE allow for the implementation of different types of access control. Roughly, one can determine which of the two is the most appropriate choice as follows. If the particular practical setting requires that policies are enforced based on the attributes associated with the *data*, then KP-ABE is probably the best choice. If the setting requires that policies are enforced based on the attributes associated with the *decrypting user*, then CP-ABE is probably the best choice. More specifically, KP-ABE can implement content-based access control using e.g., tags [PTMW10, APG+11]. When the data are created and subsequently encrypted, it may not be clear what the policies are going to be. However, it may be clear what any tags may constitute, e.g., because they relate to the encrypted data rather than who is authorized to access the data. For instance, in the medical setting, suitable tag types may be the patient's name, date of birth or social security number, and the data type (e.g., results of blood tests or scans). When a doctor wants to access some data, she can contact the hospital's KGA. The KGA can first determine whether the doctor is authorized to access these data. It can then generate a secret key for e.g., the policy "name: Alice" ∧ ("data type: scans" ∨ "data type: blood test"), so that the doctor can access all test results and scans related to Alice. However, note that this type of access control lets the KGA manage access to the data rather than the data owner. It thus does not allow for the implementation of RBAC and ABAC, which allows data owners to specify who gets access to the data, as required in settings as described in Section 2.2. In contrast, CP-ABE allows for the implementation of more fine-grained access control models such as ABAC. In the CP-ABE setting, the KGA (which may be assigned by some health authorities) distributes the keys associated with the attributes of the user. The encrypting user—which may be the data owner—gets to specify the access policy, and is therefore in control of managing access. For these reasons, we consider CP-ABE as the more favorable of the two, so we focus almost solely on CP-ABE in the remainder of this thesis.

2.3.3 The universe of attributes

The set of attributes used in an ABE scheme is called the *universe of attributes*, which can be *small* or *large* [SW05]. In the formal sense, these distinguish between whether the universe is polynomially bounded in the security parameter or not. In small-universe constructions, the master public key—generated in the setup—depends directly on the universe of attributes. Because the KGA needs to explicitly publish a public key for each attribute, the master public key is consequently polynomially bounded. In large-universe constructions, the master public key is independent of the universe. Any user can uniquely generate the public key associated with some attribute from the master public key and the attribute. Because the number of unique public keys is exponential in the security parameter, the number of attributes in the universe is essentially unbounded.

We consider large-universe constructions to be more practical than small-universe constructions for several reasons. In contrast to small-universe constructions, large-universe allow for the generation of public keys from any input strings. As such, the authority does not need to keep a record of all attributes and their public keys (which may be large!), which makes the system more scalable. In turn, users do not have to locate these public keys before encryption. A secondary advantage of this is that this may also be more privacy friendly. For instance, publishing identifiable information such as names or social security numbers reveals that a person is part of a system. Another secondary advantage is that encrypting users can use attributes for which no keys exist yet without first asking the KGA to generate these [PP03], which gives them more autonomy and therefore fosters availability. Finally, attributes can also be added to the universe without any consequences with respect to the public keys and previously generated keys and ciphertexts.

2.3.4 (Completely) unbounded ABE

Sometimes, schemes are bounded in one or more parameters. Indeed, we have already considered the size of the universe, which can be small or large. In addition, schemes can be bounded in the sizes of the sets of attributes or the access policies, and by extension the sizes of the keys or ciphertexts. Furthermore, bounds can be placed on the number of times that an attribute occurs in the policy, which we call *bounded re-use* [LOS+10]. If an attribute may be used only once in each policy, we say that the scheme suffers from a *one-use restriction*. Conversely, a scheme is multi-use if attributes may appear any number of times in the policy.

If schemes are not bounded in any of these parameters, one might argue that they are called *unbounded*. Remarkably, various works describe subtly different definitions of the term "unbounded". Notable examples include:

- LW11b [LW11b]: requires the scheme to support large universes, and to impose no bounds on the attribute sets;

- AY15 [AY15], Att19 [Att19]: require the scheme to impose no bounds on the attribute sets or access policies, including the number of uses of an attribute in the policy. They call a scheme *completely unbounded*, if it also supports large universes;

- CGKW18 [CGKW18]: requires the scheme to impose no bounds on the sets or policies.

We also observe that the term "unbounded" is usually only reserved for schemes that avoid the random oracle model (Section 2.4.4). For instance, large-universe schemes that use a full-domain hash (Section 2.5.5)—which additionally pose no restrictions on any of the discussed parameters—are not typically referred to as unbounded. Presumably, this is because the hash is modeled as a random oracle, which can be regarded as a restriction as well. Therefore, rather than classifying a scheme as unbounded or not, we will consider for each of the aforementioned parameters whether it is unbounded.

Observation 2.1

In general, an obvious disadvantage of requiring bounds on any of these parameters is that it limits some or all parties in the system, for instance, because the policy that they want to use for encryption is larger than the scheme allows. However, there seems to be an additional, more subtle disadvantage in some cases, which is not necessarily caused *directly* by imposing these bounds.

For instance, as we will show in Sections 2.5.5, 2.5.6 and 2.7.3, some methods used to achieve the large-universe property subsequently result in requiring bounds on the policy (or set) associated with the ciphertext. In addition, these methods also affect the efficiency of the scheme. Typically, the public keys and encryption costs grow by a factor that is linear [Wat08] or even quadratic [ALdP11] or cubic [AC16] in this bound. As such, the efficiency of the scheme is directly dependent on the bound. Increasing the bound makes the scheme more flexible, but less efficient, meaning that this bound cannot simply be chosen to be sufficiently large.

As another example, we consider schemes with a bounded re-use of an attribute in a policy. To mitigate the one-use restriction, some works [LOS$^+$10, LW11a] make multiple copies of each attribute. The idea is that, for each use of the same attribute in the policy, another copy of the attribute is used. However, the number of copies is fixed after the setup is run, meaning that it is bounded. Furthermore, the efficiency of the scheme depends directly on this bound [AC17a]. In this case, the public keys and key generation costs grow by a factor that is linear in the bound, and thus yield similar flexibility-efficiency trade-offs as the previous example.

2.4 Security of ABE

2.4.1 Collusion resistance

An important property of ABE is that it is required to be *collusion resistant*. If any number of users are not individually able to decrypt a ciphertext, they should not be able to decrypt collectively either. For example, a doctor who works at the Amsterdam UMC and a nurse who works at the Radboudumc should not be able to individually decrypt a ciphertext with policy “doctor” ∧ “Radboudumc’. They should thus not be able to collude and decrypt the ciphertext together either.

Collusion resistance is important for attribute-based access control. In particular, when such access control is enforced in practice, access should only be granted to authorized users. In traditional systems, the authority ensures this by verifying whether a requesting user possesses a set of attributes that satisfies the policy [HFK+19]. To do this properly, the attributes need to be authenticated. This means that the authority needs to be certain that the attributes are actually in the possession of a single user (and not, say, in the possession of multiple colluding users). To enforce access control with ABE, the employed scheme should ensure this as well, which is the case when it is collusion resistant.

2.4.2 Security models

In the context of ABE, the security models capture security against chosen-plaintext attacks (CPA) and collusion resistance (Section 2.4.1). The strongest notion of security is provided by the *full security* [LOS+10] model, then the *semi-adaptive security* [CW14b] model and then the *selective security* [SW05] model. Other models include co-selective security [AL10] and static security [RW15], but these are used much less often. The basic models consider security against chosen-plaintext attacks but can easily be extended to model chosen-ciphertext attacks (Section 2.4.3). The full security model is formally defined as follows.

Definition 2.4: Full security against chosen-plaintext attacks (CPA)

We define the security game between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first key query phase, we have $P(x^*, y) = 0$, and generates

two messages M_0 and M_1 of equal length in $\mathcal{M}_\lambda$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0,1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*, M_\beta)$, and sends the resulting ciphertext CT_{x^*} to the attacker.

- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}$ such that $P(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game.

In the selective security model, the attacker announces the challenge x^* (in CP-ABE: access structure) before the challenger runs the setup. In the semi-adaptive security model, this happens afterwards, but before the attacker is allowed to query secret keys. While selective and semi-adaptive security certainly prove a level of security—or at least, they inspire some confidence that the scheme is secure—neither does accurately model real-world dangers to security breaches [CKMS16]. It is unreasonable to assume that an attacker is going to announce which e.g., access policies they are going to attack before a system setup.

A natural question would however be: are fully secure schemes truly more secure than selectively secure schemes *in practice*? From a theoretical standpoint, this seems to be true. Intuitively, one could provide a security reduction of a selectively secure scheme in the full security model by protecting against every conceivable access policy, resulting in an exponential security loss [SW05]. In fact, Lewko and Waters [LW14] formally prove this by showing that any such black-box reduction leads to an exponential security loss. Nevertheless, so far, no practical attacks exist that can break any specific selectively secure scheme with a significant advantage over its fully secure variant. As such, it is unclear whether selective security is simply an artifact of the used proof technique, or whether these schemes are truly less secure in practice; and if so, how much less. This gives rise to the following direction.

Direction 2.1: Selective versus full security in practice

To investigate the relationship between specific instantiations of selectively secure ABE schemes and their fully secure counterparts with respect to their security in practice.

2.4.3 Security against chosen-ciphertext attacks

As noted, the basic security models only provide security against chosen-plaintext attacks (CPA). However, in practice, a scheme often also has to be secure against *chosen-ciphertext attacks* (CCA) [RS91]. The model in Definition 2.4 can easily be adapted to model this, i.e., by including a decryption oracle. The attacker is allowed to query this oracle with ciphertexts other than the challenge ciphertext. We give a more formal definition of the CCA-security model in Definition 3.2.

2.4.4 The random oracle model

Some schemes are proven secure in the *random oracle model* (ROM) [BR93], i.e., their security proofs use random oracles. In practice, the random oracles are replaced by cryptographic hash functions [RS04], which are then assumed to behave randomly. While this is an idealized functionality of a hash function, Bellare and Rogaway [BR93] argue that it is sufficiently random for its purpose in most cases. However, Canetti, Goldreich and Halevi [CGH04] show that schemes exist that are secure in the ROM, but for which no implementation of a hash function exists that yields a secure scheme. Although such insecure schemes differ substantially from real-world constructions [LN09, KM15], it is unclear whether such problems may translate to any established ABE schemes, and whether they can be exploited in practice (Direction 2.2).

2.4.5 Complexity assumptions and the generic group model

ABE schemes are typically proven secure by reducing a complexity assumption to their security. Some proofs use *static* assumptions such as the *decisional bilinear Diffie-Hellman* (DBDH) [SW05], the *(symmetric) external Diffie-Hellman* ((S)XDH) [CW14a], *decisional linear* (DLIN) [AC17a] and *subgroup* assumptions [LOS+10]. Other proofs use *parametrized* or *q-type assumptions*, which grow linearly in some parameter q (often dependent on some system parameters). Many of the q-type assumptions used in these proofs can be generalized under the “uber-assumption” [BBG05, Boy08], which is shown to generically hold in the *generic bilinear group model* (GGM) [Sho97]. However, the security level of a scheme may decrease as q increases [Che06], which makes them less attractive than static assumptions. Another difference between static and parametrized assumptions is the analysis of their security. Static assumptions are usually concise, simple to understand and derived from well-understood assumptions. In contrast, parametrized assumptions consist of many inputs and are often tailored to prove security of one specific scheme. As such, each assumption needs to be carefully studied.

Finally, some schemes directly prove security in the GGM [BSW07, ABGW17]. Much like the ROM, the GGM is considered an idealized security model and a proof herein provides only a basic level of confidence. Dent [Den02] shows that schemes exist that are provably secure in the GGM, but can be broken in practice. Regardless, much like in the ROM setting, it is unclear if this is also the case for ABE. In fact, Ambrona et al. [ABGW17] show that a strong relationship exists between generic attacks and the generic security of ABE schemes. (They additionally show that several selectively secure schemes can be proven fully secure in the GGM, with only polynomial security loss, which in part addresses Direction 2.1.) However, this does not include attacks that are non-generic, e.g., that exploit the used hash function (in the ROM) or the underlying group structure (in the GGM).

Direction 2.2: Security of ABE with proofs in idealized models in practice

To analyze the security of ABE schemes that are provably secure in the random oracle model or generic (bilinear) group model in real-world implementations, e.g., with respect to their instantiated hash functions or underlying groups.

2.5 Pairing-based ABE

In this section, we review pairing-based ABE. To this end, we discuss the common structure used in many schemes, and how some of the properties discussed in Section 2.3 can be achieved.

In general, due to the collusion-resistance property (Section 2.4.1), ABE requires security guarantees for both the secret keys and ciphertexts. This is unlike more traditional forms of public-key encryption, such as ElGamal [Gam84], which typically only require security guarantees for the ciphertexts. To ensure security of such encryption schemes, the ciphertexts are defined in groups in which the decisional Diffie-Hellman problem [DH76, Bon98] is assumed to be hard. Roughly, it should then (provably) hold that illegitimately decrypting the ciphertext is just as hard as solving the discrete-log problem. To ensure that security guarantees can be achieved for both the keys and ciphertexts, it makes sense to place both the keys and ciphertexts in such groups, which may, however, complicate decryption. To overcome these difficulties, pairings can be used.

2.5.1 Pairings

A *pairing*—also known as a *bilinear map*—is a map $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ defined over three groups $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$ such that (i) $e(g^a, h^b) = e(g,h)^{ab}$ for all $a, b \in \mathbb{Z}_p$ (bilinearity), (ii) $e(g,h)$ is not the identity in $\mathbb{G}_T$ (non-degeneracy) and (iii) e is efficiently computable. In some schemes, the order of the groups is a composite of three distinct primes instead of a single prime. Because the operations in such groups are one to two orders of magnitude less efficient than in its prime-order counterparts [Gui13], prime-order groups are preferred.

Depending on the relationship between $\mathbb{G}$ and $\mathbb{H}$, different *types* of pairings exist. If $\mathbb{G} = \mathbb{H}$, then the pairing is *symmetric* and called a *type-I* pairing. If not, then the pairing is *asymmetric*. Subsequently, if an efficiently computable homomorphism from $\mathbb{H}$ to $\mathbb{G}$ exists, then it is a *type-II* pairing, and otherwise, it is a *type-III* pairing. While many schemes are designed in the type-I setting, type-III pairings should be used in practice, due to computational efficiency [GPS08] and security issues in the type-I setting [Gal14]. In general, schemes designed in the type-I setting can be securely converted to the type-III setting [AGOT14, AGH15, AHO16a].

2.5.2 Representation of access structures

In pairing-based ABE, (non-)monotone span programs are typically represented by access trees [GPSW06a] or linear secret sharing scheme (LSSS) matrices [GPSW06b]. Most often, LSSS matrices are used, which can be efficiently generated with the methods described in e.g., [LW10a].

Definition 2.5: Access structures represented by LSSS matrices

An access structure can be represented as a pair $\mathbb{A} = (\mathbf{A}, \rho)$, where $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ is an LSSS matrix with $n_1, n_2 \in \mathbb{N}$, and ρ is a function that maps the rows of $\mathbf{A}$ to attributes in the universe. Then, for some vector with randomly generated entries $\mathbf{v} = (s, v_2, ..., v_{n_2}) \in \mathbb{Z}_p^{n_2}$, the i-th secret generated by this matrix is $\lambda_i = \mathbf{A}_i \mathbf{v}^\intercal$, where $\mathbf{A}_i$ denotes the i-th row of $\mathbf{A}$. If $\mathcal{S}$ satisfies $\mathbb{A}$, then a set of rows $\Upsilon = \{i \in \{1, ..., n_1\} \mid \rho(i) \in \mathcal{S}\}$ and coefficients $\varepsilon_i \in \mathbb{Z}_p$ for all $i \in \Upsilon$ exist such that $\sum_{i \in \Upsilon} \varepsilon_i \mathbf{A}_i = (1, 0, ..., 0)$, and by extension, $\sum_{i \in \Upsilon} \varepsilon_i \lambda_i = s$ holds. If $\mathcal{S}$ does not satisfy $\mathbb{A}$, there exists a $\mathbf{w} = (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A}_i \mathbf{w}^\intercal = 0$ for all $i \in \Upsilon$ [Bei96].

Lewko and Waters have devised an algorithm that yields an efficient implementation of the decryption algorithm [LW10a]. First, we translate the access policy to an access tree, such that the leaves correspond to the attributes and the nodes to OR and AND-gates. We create the LSSS matrix recursively as follows:

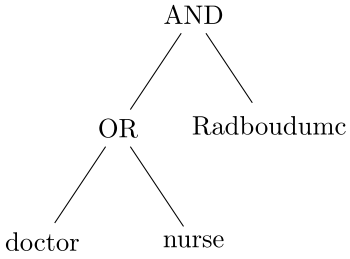
- (Base case) Initialize the root node with vector $\mathbf{x} = (1)$ of length $n_2 = 1$. Note that n_2 is a globally updated counter in the protocol.
- (Inductive step)
  - OR-gate: propagate the node vector $\mathbf{x}$ to both children.
  - AND-gate: split the node vector $\mathbf{x}$ into two vectors: $(\mathbf{x} \| 0^{n_2 - |\mathbf{x}|} \| 1)$ and $(0^{n_2} \| - 1)$ with length $n_2 + 1$, update $n_2 \leftarrow n_2 + 1$ and propagate the vectors to the children.

- (Final step, after all nodes have been propagated) For each leaf, set the vector $\mathbf{x}$ inherited from the parent to $(\mathbf{x} \| 0^{n_2 - |\mathbf{x}|})$ if $|\mathbf{x}| < n_2$.

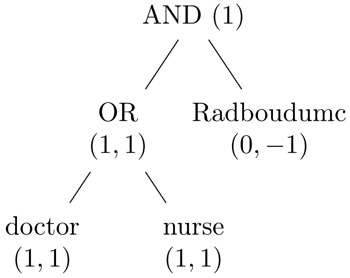
The output of this algorithm is an LSSS matrix $\mathbf{A}$ spanned by the rows in the leaves. We show how the algorithm works with an example.

Example 2.1: Converting an access policy to LSSS matrix

Consider the policy ("doctor" OR "nurse") AND "Radboudumc". We first translate it to an access tree:



Then, we can use this access tree to construct the matrix as follows. We start with the vector (1) in the root (which is an AND-gate in this case). Because the node is an AND-gate, we split the vector into two new vectors of length 2, $(1, 1)$ and $(0, -1)$, which are propagated to the children. On the left side of the tree, the vector in the OR-gate then needs to be propagated to its children, which inherit the same vector due to the OR-gate. This yields the following tree:



Using this tree, we finally obtain the access structure $\mathbb{A} = (\mathbf{A}, \rho)$, where

$$\mathbf{A} = \begin{pmatrix} 0 & -1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad \rho \colon \{1, 2, 3\} \to \{\text{doctor}, \text{nurse}, \text{Radboudumc}\},$$

$$\text{with } \rho(1) = \text{Radboudumc}, \quad \rho(2) = \text{doctor}, \quad \rho(3) = \text{nurse}.$$

2.5.3 Example: the Wat11 scheme

To illustrate what pairing-based ABE schemes look like, we give an example of a CP-ABE scheme. Arguably the simplest CP-ABE scheme is the first Wat11 [Wat11] scheme. It is the CP-ABE variant of the first expressive KP-ABE scheme, i.e., GPSW06 [GPSW06a]. The scheme is originally defined in the prime-order and symmetric setting (and only provides selective security (Section 2.4.2)). This scheme was later improved on in many works [LOS+10, Att14a, Att16, AC17b, KW19b, Att19, LL20a], attaining better security and/or practicality properties.

Construction 2.1: The Wat11 [Wat11] scheme

- Setup(λ): Let $p, \mathbb{G}, \mathbb{G}_T, e, g$ be generated as in Section 2.5.1, such that e is a symmetric pairing and provides sufficient security with respect to security parameter λ. The key generation authority (KGA) also initializes universe $\mathcal{U}$ and randomly generates $\alpha, b, b_{\text{att}} \in_R \mathbb{Z}_p$ for all att $\in \mathcal{U}$. It keeps $\text{MSK} = (\alpha, b, \{b_{\text{att}}\}_{\text{att}\in\mathcal{U}})$ as the master secret key and publishes the master public key

$$\text{MPK} = (p, \mathcal{U}, \mathbb{G}, \mathbb{G}_T, e, g, A = e(g,g)^{\alpha}, B = g^{b}, \{B_{\text{att}} = g^{b_{\text{att}}}\}_{\text{att}\in\mathcal{U}})$$

- KeyGen(MSK, $\mathcal{S}$): For a user that possesses a set of attributes $\mathcal{S}$, the KGA randomly generates $r \in_R \mathbb{Z}_p$, and returns as secret key:

$$\text{SK}_{\mathcal{S}} = (\mathcal{S}, K = g^{\alpha - rb}, K' = g^{r}, \{K_{\text{att}} = g^{rb_{\text{att}}}\}_{\text{att}\in\mathcal{S}}).$$

- Encrypt(MPK, $\mathbb{A}$, M): An encrypting user encrypts message $M \in \mathbb{G}_T$ under access policy $\mathbb{A}$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$, $\rho\colon \{1, ..., n_1\} \to \mathcal{U}$. The user then randomly generates integers $s, s_1, ..., s_{n_1}, v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ and computes the ciphertext as

$$\text{CT}_{\mathbb{A}} = (\mathbb{A}, C = M \cdot A^{s}, C' = g^{s}, \{C_{1,j} = B^{\lambda_j} B_{\rho(j)}^{s_j}, C_{2,j} = g^{s_j}\}_{j\in\{1,...,n_1\}}),$$

such that λ_i denotes the i-th entry of $\mathbf{A} \cdot (s, v_2, ..., v_{n_2})^{\intercal}$.

- Decrypt($\text{SK}_{\mathcal{S}}$, $\text{CT}_{\mathbb{A}}$): Suppose that set $\mathcal{S}$ satisfies policy $\mathbb{A}$, and suppose $\Upsilon = \{j \in \{1, ..., n_1\} \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j\in\Upsilon}$ exist with

$\sum_{i\in\Upsilon}\varepsilon_j\mathbf{A}_j=(1,0,...,0)$. Then the plaintext M is retrieved by computing

$$C/\left(e(C',K)\cdot\prod_{j\in\Upsilon}\left(e(C_{1,j},K')/e(K_{\rho(j)},C_{2,j})\right)^{\varepsilon_j}\right).$$

The scheme is correct, i.e., decryption indeed yields M:

$$\begin{aligned}
&M\cdot e(g,g)^{\alpha s}/\left(e(g^{\alpha-rb},g^s)\cdot\textstyle\prod_{i\in\Upsilon}(e(g^r,B^{\lambda_i}B^{s_i}_{\rho(i)})/e(g^{rb_{\rho(i)}},g^{s_i}))^{\varepsilon_i}\right)\\
&=M\cdot e(g,g)^{\alpha s}/\left(e(g^{\alpha-rb},g^s)\cdot\textstyle\prod_{i\in\Upsilon}(e(g^r,g^{\lambda_i b+s_i b_{\rho(i)}})/e(g^{rb_{\rho(i)}},g^{s_i}))^{\varepsilon_i}\right)\\
&=M\cdot e(g,g)^{\alpha s}/\left(e(g,g)^{\alpha s-rsb}\cdot e(g,g)^{\sum_{i\in\Upsilon}\varepsilon_i r\lambda_i b}\right)\\
&=M\cdot e(g,g)^{\alpha s}/\left(e(g,g)^{\alpha s-rsb}\cdot e(g,g)^{rsb}\right)=M.
\end{aligned}$$

We show how the scheme works with an example.

Example 2.2: Encrypting and decrypting with Wat11

Suppose that we want to encrypt our message $M\in\mathbb{G}_T$ under the policy (“doctor” OR “nurse”) AND “Radboudumc”. Then, we first represent the policy as an LSSS matrix, i.e., $\begin{pmatrix}0&-1\\1&1\\1&1\end{pmatrix}$ (see Example 2.1). We generate $s,s_1,s_2,s_3,v_2\in_R\mathbb{Z}_p$ and compute the shares

$$\boldsymbol{\lambda}=\begin{pmatrix}\lambda_1\\\lambda_2\\\lambda_3\end{pmatrix}=\mathbf{A}\cdot(s,v_2)^{\intercal}=\begin{pmatrix}0&-1\\1&1\\1&1\end{pmatrix}\begin{pmatrix}s\\v_2\end{pmatrix}=\begin{pmatrix}-v_2\\s+v_2\\s+v_2\end{pmatrix}.$$

The ciphertext is then computed as

$$\mathrm{CT}_{\mathbb{A}}=(\mathbb{A},C=M\cdot A^s,C'=g^s,\{C_{1,j}=B^{\lambda_j}B^{s_j}_{\rho(j)},C_{2,j}=g^{s_j}\}_{j\in\{1,2,3\}}).$$

Another user, who has secret keys for the attributes “nurse” and “Radboudumc” can then decrypt the ciphertext as follows. The user matches the attributes corresponding to the first and third row of the policy matrix, i.e., $\Upsilon=\{1,3\}$. They then try to find a linear combination of those rows such that $(1,0)$ can be retrieved, which is when the rows are simply added, i.e., $\varepsilon_1=1$ and $\varepsilon_3=1$. The user then retrieves M by simply computing

$$C/\left(e(C',K)\cdot\prod_{j\in\{1,3\}}\left(e(C_{1,j},K')/e(K_{\rho(j)},C_{2,j})\right)\right)=M.$$

2.5.4 Standard form

Many pairing-based schemes have a similar structure as the Wat11 scheme. In particular, the keys and ciphertexts mainly exist in $\mathbb{G}$ (and $\mathbb{H}$)—with the exception of the first ciphertext component, which is almost always $C = M \cdot e(g,g)^{\alpha s}$—and decryption consists of pairing the appropriate key and ciphertext components. This common structure is explicitly considered in frameworks that consider generic compilers (Section 2.6.2), which abstracts the schemes by analyzing the "exponent space". For instance, the exponent space of the secret keys of the Wat11 scheme (Construction 2.1) can be described as a vector of elements in $\mathbb{Z}_p$, i.e., $\mathrm{sk}_{\mathcal{S}} = (k = \alpha - rb, k' = r, \{k_{\mathrm{att}} = rb_{\mathrm{att}}\}_{\mathrm{att} \in \mathcal{S}})$. In a more general sense, this vector $\mathrm{sk}_{\mathcal{S}}$ can be expressed as a function of the master-key α, some variables $\mathbf{b}$ associated with the public keys, and some variables $\mathbf{r}$ associated with the secret key. Similarly, such a vector can be defined for the ciphertexts. We summarize these findings by defining the *standard form* of predicate encryption (which covers both KP-ABE and CP-ABE (Definition 2.3)), which is derived from the aforementioned generic-compiler frameworks [Wee14, Att14a]. In this definition, we use the following shorthand for generators g and vectors $\mathbf{b} = (b_1, ..., b_n)$: $g^{\mathbf{b}} = (g^{b_1}, ..., g^{b_n})$.

Definition 2.6: Standard form of predicate encryption [Wee14, Att14a]

The standard form of predicate encryption is defined as follows:

- Setup(λ): Taking as input the security parameter λ, the KGA generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. The KGA also defines the universe of attributes $\mathcal{U}$, and generates random $\alpha, b_1, ..., b_n \in_R \mathbb{Z}_p$, where $n \in \mathbb{N}$ is some integer. It outputs $\mathrm{MSK} = (\alpha, \mathbf{b} = (b_1, ..., b_n))$ as its master secret key and publishes the master public key as
$$\mathrm{MPK} = (g, h, e(g,h)^{\alpha}, g^{\mathbf{b}}, h^{\mathbf{b}}).$$
We refer to $\mathbf{b}$ as the common variables, because they occur in both the secret keys and ciphertexts. We refer to α as the master-key, as it can be used to decrypt any ciphertext.

- KeyGen(MSK, y): The KGA generates a secret key for y by generating user-specific random integers $\mathbf{r} = (r_1, r_2, ...) \in_R \mathbb{Z}_p$ and computing the secret key as
$$\mathrm{SK}_y = (y, h^{\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)}),$$
where $\mathbf{k}$ denotes a vector defined over the user-specific random variables, master secret keys and associated set of attributes.

- Encrypt(MPK, x, M): An encrypting user encrypts the message $M \in \mathbb{G}_T$ for x by generating ciphertext-specific randoms $\mathbf{s} = (s, s_1, s_2, ...) \in_R \mathbb{Z}_p$ and computing the ciphertext as

$$\mathrm{CT}_x = (x, M \cdot e(g,h)^{\alpha s}, g^{\mathbf{c}(\mathbf{s},\mathbf{b},x)}),$$

where $\mathbf{c}$ denotes two vectors defined over the ciphertext-specific random variables, master public keys and associated access structure.

- Decrypt(SK, CT): Let $\mathrm{SK} = (y, \mathbf{K} = h^{\mathbf{k}})$ be a secret key and $\mathrm{CT} = (x, C = M \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}})$ a ciphertext such that $P(x,y) = 1$. Define $\mathbf{E}(x,y)$ as the matrix such that we have $\mathbf{c}\mathbf{E}\mathbf{k}^{\intercal} = \alpha s$. Then, we retrieve plaintext M by computing

$$C / \left(\prod_{i,j} e(C_i, K_j)^{\mathbf{E}_{i,j}} \right),$$

where $\mathbf{C} = (C_1, C_2, ...)$ and $\mathbf{K} = (K_1, K_2, ...)$.

Observation 2.2

As suggested by Attrapadung [Att14a], the standard form implies a metric that can be used to measure the "similarity" between two schemes. For instance, by analyzing the LOSTW10 [LOS⁺10] and Wat11 [Wat11] schemes, one would conclude that these two schemes have similar structures. In fact, if one were to abstract both schemes to the vectors **b**, **k** and **c** as in Definition 2.6, they would turn out to be the same. The main difference between the two schemes is the underlying group structure: whereas Wat11 is built on prime-order groups, LOSTW10 is built on composite-order groups. As it turns out, many CP-ABE schemes are structurally similar to the Wat11 scheme, and oftentimes only differ in the underlying group structure [Att14a]. The reason for this is that the Wat11 scheme has an efficient "vector structure"—often referred to as pair encoding (Chapter 4)—compared to other CP-ABE schemes, but it is provably secure in a weaker model and under less established assumptions than would be desirable (Section 2.4). In contrast, the derived schemes [LOS⁺10, Att14a, Att16, AC17b, KW19b, Att19, LL20a] are provably secure in stronger models and under more established assumptions, possibly at the cost of some basic functionality and, importantly, the efficiency.

2.5.5 Supporting large universes

We analyze the methods that are used to support large universes. As argued in Section 2.3.3, from a practical viewpoint, we prefer large-universe constructions over small-universe schemes. Although pairing-based small-universe schemes are simpler to create, they can often be converted into the large-universe setting [Wat11, CGKW18]. This is because many such small-universe constructions use only the master public key associated with an attribute in the key generation (and not some associated secret key). For example, in the Wat11 scheme (Construction 2.1), the generator $B_{\mathrm{att}} = g^{b_{\mathrm{att}}}$ is used in the key generation and encryption algorithms. Two techniques exist that allow this generator $g^{b_{\mathrm{att}}}$ to be generated by a function. First, a full-domain hash (FDH) $\mathcal{H}\colon \{0,1\}^* \rightarrow \mathbb{G}$ can be employed that maps strings directly into the group. This FDH is modeled as a random oracle [GPSW06a]. Second, a collision-resistant hash function can be used to map strings to $\mathbb{Z}_p$, and then an implicit n-degree polynomial is used to map the integer to the group [SW05]. For instance, the KGA can generate an n-degree polynomial $f(\mathrm{att}) = \sum_{i=0}^{n} b_i x_{\mathrm{att}}^i$, include "coefficients" $g^{b_0}, ..., g^{b_n}$ in the master public key, and define the map F as $F(\mathrm{att}) = \prod_{i=0}^{n} (g^{b_i})^{x_{\mathrm{att}}^i} = g^{f(\mathrm{att})}$, where we assume that x_{att} is the hashed representation of att in $\mathbb{Z}_p$. The latter is sometimes also called a "Boneh-Boyen" hash [BB04].

We give an overview of several advantages (+) and disadvantages (−) of the two methods, and their relationship with other properties of ABE. For the "FDH-based" method, we identify the following advantages and disadvantages:

+ It is simple to apply to many small-universe schemes, e.g., [PTMW10, Wat11];

+ It typically yields schemes that attain the same structure, and therefore, at first glance, seem to attain the same efficiency as the small-universe counterpart.

− It may however negatively impact the efficiency of the key generation and encryption algorithms compared to its small-universe variant (Chapter 6);

− Currently, no techniques exist that allow for the additional support of non-monotonicity (Section 2.5.7);

− It cannot benefit from online/offline techniques like polynomial-based large-universe constructions (Section 2.10.1), and therefore cannot significantly improve the efficiency of the key generation and encryption algorithms in practice.

− Its security is proven in the random oracle model (Section 2.4.4).

For the "polynomial-based" method, we identify the following advantages and disadvantages:

+ Techniques exist that allow for the additional support of non-monotonicity (Section 2.5.7);

+ The map is solely determined by group elements g^{b_i}, which may positively influence the efficiency of the key generation and encryption (Chapter 6);

+ The online/offline variants of these schemes (Section 2.10.1) allow for a split of the computational costs into an online and offline phase. The online phase requires almost no computational costs, which significantly improves the efficiency of key generation and encryption in practice;

+ Its security does not rely on the random oracle model (Section 2.4.4).

– Achieving unboundedness is non-trivial (Section 2.5.6);

– Existing schemes typically incur a heavy trade-off in practicality and efficiency, e.g., by being bounded [GPSW06a, Wat08] or by having decryption [LW11b, RW13, CGKW18] or key generation [AHM$^+$16] algorithms that are several factors more computationally costly than the other algorithms (Observation 2.1 and Section 2.7.3).

2.5.6 Achieving unboundedness

As we mentioned in Section 2.3.4, some schemes are bounded in one or more system parameters, such as the attribute sets or policies, or the number of times that a specific attribute may occur in the policy. The most notable reason why any of these parameters may be bounded is related to the amount of randomness that is used in the scheme. For instance, in the second Wat11 [Wat11, Wat08] scheme, the same randomness is used for all attributes in the ciphertext. That is, the ciphertexts in Construction 2.1 are replaced by ciphertexts of the form:

$$\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, C = M \cdot A^s, C' = g^s, \{C_{1,j} = B^{\lambda_j} B^s_{\rho(j)}\}_{j \in [1,n_1]}),$$

i.e., the $B_{\rho(j)}$ is also randomized with s instead of a fresh randomizer s_j. As a result, the number of times that the attribute $\rho(j)$ can occur in the policy is restricted to one. If it is used more than once, B^{λ_j} is not hidden anymore. Similarly, the sets or policies may be bounded due to the use of insufficient randomness in the keys or ciphertexts. For example, in [SW05, GPSW06a], the authors convert the small-universe constructions into large-universe constructions by replacing the generator B_{att} by an implicit n-degree polynomial. However, such a polynomial provides only sufficient randomness for n attributes. Hence, intuitively, plugging it into e.g., the keys in the Wat11 scheme in Construction 2.1, where each key component is randomized with the same randomness r, places a bound[5] on the size of the attribute set: n.

To remove those bounds, more randomness can be used. For example, to lift the one-use restriction[6], one could use a fresh randomizer for each attribute [Wat11] or

[5] Actually, the large-universe construction in Appendix B of the full version [Wat08] is bounded in both the sets and policies, because it uses only one randomizer in the keys, and one in the ciphertexts.

[6] In some areas of ABE, this restriction is not as easily lifted as we suggest here. For example, some proof techniques used in the dual system encryption paradigm (Section 2.6.2) require a one-use restriction for a different reason. This restriction was lifted by Kowalczyk and Wee [KW19b] by introducing novel proof techniques.

for each re-use of one specific attribute [AC17b]. For polynomial-based large-universe schemes, this is more difficult. Intuitively, the reason why this is difficult is in the keys. In general, the keys are tied to one specific user by using the same randomness (e.g., see the discussion in Section 2.8.4). For example, in Wat11, the same r is used for all key components. Hence, replacing B_{att} with a polynomial-based hash $F(\text{att})$ yields a bound on the attribute set associated with the keys, as $F(\text{att})^r$ only provides sufficient randomness for n attributes. Lewko and Waters [LW11b] and Rouselakis and Waters [RW13] overcome this issue by adapting these schemes such that the keys can be tied to one specific user while fresh randomness can be used for each attribute. This subsequently complicates the proof techniques, and fixes the polynomial to the case where the degree $n = 1$.

2.5.7 Supporting non-monotonicity

We analyze the existing methods used to support non-monotonicity. As mentioned, schemes that support non-monotone access structures are more expressive. Moreover, they may simplify the support of e.g., revocation [LSW10] (Section 2.10.2). In general, we observe that two methods exist to support non-monotonicity in ABE. First, the most straightforward way is to include a negative instance of each attribute in the universe [CN07]. For example, the negative instance of the attribute “doctor” is “non-doctor”. During the key generation, a partial secret key is generated for each attribute in the universe: a positive instance for each attribute in the set, and a negative instance for every other attribute in the universe. The advantage of this method is that it can be applied to any small-universe scheme, and the efficiency of the encryption and decryption algorithms are the same as in the monotone setting. The disadvantage of this technique is that it inherently requires the support of small universes only [Att19], and the key generation costs grow in the size of the universe. Furthermore, supporting non-monotonicity in this way causes issues when attributes are added to the universe, which we discuss in more detail in Section 2.9.

Ostrovsky et al. [OSW07] devised another method—hereinafter referred to as OSW-method—which also supports large universes. In particular, their method exploits the structure of large-universe constructions that use the implicit n-degree polynomial (Section 2.5.5), and can thus also be applied to completely unbounded schemes [YAHK14, Att19]. Roughly, this method requires that, during decryption, the entire set of attributes associated with the key is compared with the negated attribute. The decrypting user only satisfies this negation, if all attributes are different from the negated attribute. On a more technical level, this is achieved via the polynomial by exploiting Lagrange interpolation, also frequently used in secret sharing [Sha79]. Intuitively, a partial decryption key is shared among the attributes in the set associated with the key such that one more secret share is needed to recover the decryption key. During decryption with a negated attribute, the decryption key can only be reconstructed if the negated attribute is different from all attributes in the set. The disadvantage of this method is, however, that this comes at the cost of some efficiency. For example, compared to RW13 [RW13], its non-monotone variant in [YAHK14] has keys that are twice as large (and thus the key generation costs are doubled). The decryption costs for negated attributes scale linearly in the size of the set of attributes associated with the key.

In the realm of pairing-based ABE, a special subtype of non-monotonicity—which we shall refer to as "labeled non-monotonicity"—was proposed by Okamoto and Takashima [OT12], and was later further investigated and developed in [TKN20, AT20]. These works explicitly use the attribute labels—corresponding to our notion of attribute types (Section 2.3.1)—in access policies. For instance, the attribute "doctor" can belong to the sub-universe labeled as "profession". Then, the attribute can be negated in two ways: "NOT profession: doctor" or "profession: NOT doctor". In the first case, a set of attributes satisfies the negation if it does not contain the attribute "profession: doctor". In the second case, a set satisfies the negation if it does contain at least one attribute with the label profession, but not with the value "doctor". In particular, the negation is not satisfied if the set does not contain any attributes with the label. We call the latter type of non-monotonicity "labeled non-monotonicity".

Labeled non-monotonicity can be used to securely implement access control that supports negations in dynamic practical settings [TKN20, AT20]. For instance, consider a situation in which a new label, e.g., "profession", is added to a system and used in a negation during encryption. At this point, none of the users have an attribute for this label yet; as such, each user would automatically satisfy the negation "NOT profession: doctor". In contrast, users do not automatically satisfy the negation "profession: NOT doctor", as they need at least one attribute associated with the label "profession". To satisfy it, they would first need to request a new key for a set of attributes that also includes the new label. (Note, however, that users may possibly not possess any attributes associated with the label. In this case, such users could be assigned an empty value, e.g., "profession: none".) We show in Section 2.9.2 that, compared to other techniques achieving non-monotonicity, techniques using labeled non-monotonicity provide the best availability properties without compromising the security of the schemes.

Observation 2.3

Currently, large-universe schemes that support non-monotone access structures incur a significant efficiency trade-off. We identify two underlying reasons:

- Only the second non-monotonicity method, i.e., the OSW-method, can simultaneously support large-universeness and non-monotonicity. As such, the resulting schemes suffer from the same decryption inefficiency as unbounded ABE using the polynomial method, as pointed out in Section 2.7.3;

- The OSW-method requires that, during decryption, the entire set of attributes is compared to the negated attribute, incurring computational costs that are linear in the set.

To some extent, labeled non-monotonicity mitigates both these issues. Most obviously, it mitigates the second issue by only requiring that a subset of attributes associated with the same label as the negated attribute is compared. A less obvious reason is that the attribute labels constitute another "layer" of the universe of attributes [AT20]. For this layer, we do not require non-monotonicity, so we can use both methods to support large-universeness as discussed in Section 2.5.5. In this way, it may be possible to achieve a more desirable efficiency trade-off. For instance, to optimize decryption, we can use an FDH rather than a polynomial. Conversely, to optimize the key generation efficiency or to benefit from online/offline extensions, we can use a polynomial.

For example, the TKN20 [TKN20] scheme supports labeled non-monotonicity by using FDH-based and polynomial-based methods to support large universes. Unlike FDH-based large-universe constructions, the scheme uses an FDH to map the universe labels to the group. Within each labeled universe, it maps the attributes to the group by using the polynomial method (Section 2.5.5). In this way, the non-monotonicity supported with (a simplified version[†] of) the OSW-method ensures that not the entire set of attributes needs to be compared to the negated attribute during decryption. Rather, only the subset of attributes associated with the negated attribute's label needs to be compared. While this yields a more efficient decryption algorithm compared to schemes that support non-labeled non-monotonicity, the use of an FDH may decrease the efficiency of the key generation and encryption algorithms (Chapter 6). On the other hand, if we use the polynomial method to map the labels to the group, as is proposed in [AT20], then decryption requires a linear number of pairing operations per matching attribute. Possibly, a more balanced efficiency can be achieved if Direction 2.6 is explored.

[†] In the case of TKN20 [TKN20], each attribute label may occur only once in the set of attributes. Attrapadung and Tomida [AT20] later lift this restriction by applying the OSW-method in the "attribute layer".

2.6 Security of pairing-based ABE

We review some important techniques and developments in proving security of pairing-based ABE.

2.6.1 Selective security: "program-and-cancel" proofs

In general, the choice of security model depends on the proof strategy. In many early works [SW05, GPSW06a, Wat11], the "program-and-cancel" strategy is used to prove security. In this proof strategy, the challenge access structure is "programmed" in the public keys. During the key query phase, for each set $\mathcal{S}$, the set $\{1, ..., n_1\}$ corresponding to the rows of the access structure is split into two subsets: the set Υ of rows associated with the attributes that are in the set $\mathcal{S}$ associated with the key, and its complement $\overline{\Upsilon}$. For each of the two subsets, a special property is used to ensure that the key components that cannot be programmed are "canceled". This can only be done if the attacker commits to the challenge access structure before the setup (or key query) phase is run. This strategy is therefore mostly used to prove selective (or semi-adaptive) security. Another characteristic of selective security proofs using the program-and-cancel technique is that, in the ciphertext-policy and prime-order setting, the used complexity assumption is oftentimes q-type [Wat11, RW13]. Roughly, this is due to the way in which the policies are embedded in the public keys.

Selective proof techniques can also be utilized in full-security proofs [LW12]. In fact, for some schemes, currently the only way to prove full security is to use selective proof techniques [Att14a, AC17b]. As a consequence, these schemes have, at best, a full security proof under a q-type assumption. We consider these proof techniques in more detail in Chapter 4.

2.6.2 Full security through dual system encryption

Proving full security is difficult but important. As our taxonomy in Table 2.2 shows, the minority of schemes are proven fully secure. This is, in part, because selective security is arguably easier to prove, and typically yields more efficient schemes. Another reason why few schemes in our table are fully secure is that many generic frameworks and transformations exist that do not necessarily aim to build one fully secure ABE scheme—and are therefore not listed in our taxonomy—but generalize existing structures and transformations to simultaneously achieve certain properties. This simplifies the construction of fully secure ABE schemes with many desirable properties while attaining strong security guarantees.

For the past decade, much progress has been made in achieving stronger security guarantees for the existing selectively secure schemes. Currently, the most efficient fully secure versions (e.g., [KW19b, LL20a]) of their selectively secure counterparts (e.g., [GPSW06a, Wat11]) incur roughly twice as much storage and computational cost. These works use and improve the *dual system encryption* methodology introduced by Waters [Wat09]. Interestingly, a vast body of literature exists in this area, e.g., [LOS$^+$10, LW10b, OT10, Fre10, Lew12, CW13, CW14a, CGW15, CG17]. We believe that, in itself, this subfield within pairing-based ABE can benefit from a systematized overview. To avoid heavy, technical explanations without entirely avoiding these accomplishments, we merely mention some interesting recent results.

Generic compilers. To simplify the design and analysis of fully secure schemes, generic frameworks are formulated within the dual system encryption framework, which define generic transformations or *compilers* [Wee14, Att14a, CGW15, AC16, Att16, AC17b]. These compilers facilitate simplified security proofs by proving security of the underlying group structure generically. They ensure that the designer only needs to prove simple notions of security (such as information-theoretic ones) over the exponent space. Notably, [AC17b] (and by extension [ABGW17]) only requires algebraic notions of security, which are derived from selective security proof techniques. Effectively, they prove security generically for any scheme that is not trivially broken. In particular, a scheme is trivially broken if an unauthorized key exists that can decrypt a challenge ciphertext. As a consequence, many selectively secure schemes can be transformed into fully secure schemes (albeit under a q-type assumption). Note that many of these generic frameworks prove stronger notions of security for previously constructed schemes, such as the Wat11 [Wat11] and RW13 [RW13] schemes.

Generic transformations, conversions and compositions. Not only the underlying group structures have been analyzed, but also the structural transformations that are used to achieve certain properties. Several works are dedicated to converting schemes with certain properties into schemes with other properties [AY15, AHY15]. These works are built on the generic compiler frameworks of Attrapadung [Att14a, Att16]. Other works show that certain transformations on the predicates, e.g., conjunctions or negations, preserve security [Att19, AT20, Amb21]. In particular, [Att19, Amb21] are instantiated in the framework of Agrawal and Chase [AC17b]. As a result, schemes satisfying properties such as complete unboundedness, non-monotonicity and constant-size ciphertexts can be constructed, whilst attaining strong security guarantees (e.g., full security under static assumptions in the standard model [AT20, LL20b]). This flexibility in supporting such desirable properties generically is one of the reasons why a large part of this thesis builds on the framework of Agrawal and Chase. We explain this in more detail in Chapter 4.

2.6.3 Conversion from CPA to CCA-security

Most ABE schemes are only proven CPA-secure, though there are some exceptions [CN07, YWRL10, OT10]. Oftentimes, generic conversion methods can be applied, such as methods using non-interactive zero-knowledge proofs [RS91], or key-encapsulation techniques such as the Fujisaki-Okamoto transformation [FO99], which both yield security in the random oracle model.

To avoid random oracles, some conversion methods exploit specific properties of ABE. Yamada et al. [YAHK11] give two methods that use the Canetti-Halevi-Katz transformation [CHK04] to generically obtain CCA-security. The first method considers the *delegatability* of a scheme. A scheme is delegatable if a secret key associated with a set of attributes can be transformed into a secret key associated with a smaller set of attributes. The second method considers the *verifiability* of a scheme. A scheme is verifiable if a user can verify for two sets of attributes whether their associated keys decrypt to the same value. For schemes that are not delegatable or verifiable, Koppula and Waters [KW19a] describe a conversion method that can always be used. For KP-ABE, the most efficient CCA-secure schemes (avoiding random oracles) can be constructed from large-universe constructions with the delegatability method, incurring only a small constant overhead in all algorithms. However, for CP-ABE, all conversion techniques incur a large overhead in at least one of the algorithms. In general, the verifiability method roughly doubles the decryption costs, and the delegatability method for CP-ABE incurs a large constant overhead in all algorithms. Therefore, we formalize the following future direction.

Direction 2.3: Efficient CCA-conversion for CP-ABE

To devise an efficient CCA-conversion for CP-ABE that incurs only a small constant overhead in all algorithms.

2.7 Efficiency of pairing-based ABE

One of the most important practical aspects of any cryptographic primitive is the efficiency. Compared to other primitives that are used to implement access control (as discussed in Section 2.2), ABE generally requires more computational power on the user side. To narrow the efficiency gap, it is paramount that the computational costs of ABE are minimized. In this section, we critically review some aspects related to efficiency of ABE. We also outline some directions related to making ABE more efficient, and properly measuring and comparing efficiency.

2.7.1 The storage costs

In general, the efficiency of an ABE scheme is determined by the computational and storage costs. For the *storage costs*, the sizes of the public and secret keys, as well as the ciphertexts are considered. In practice, it may be more important to optimize the size of the ciphertexts, rather than the keys. For instance, the secret keys are stored on the decryption device, which may not need to be updated frequently after key generation. In contrast, this device may frequently receive ciphertexts to decrypt from other data sources. For mobile devices with limited data subscriptions, it might be problematic to have large ciphertexts. For storage-constrained devices such as sensors and other IoT devices that encrypt data using ABE, such ciphertexts may simply be too large, yielding a problem on the encrypting user's side. Therefore, minimizing the ciphertext size may be desired or even required in these settings.

To support ABE on resource-constrained devices, schemes with constant-size or short ciphertexts [HLR10, AC16, AC17b] can be deployed—possibly alongside another scheme that is more suitable for less constrained devices. As we will show in the taxonomy (Table 2.2), these schemes typically incur various trade-offs in flexibility (imposing bounds on the universe, access policies or sets of attributes) or expressivity (only supporting AND-gates or threshold functions). The only exception is the [AHM+16] scheme, which is expressive, KP-based, achieves complete unboundedness, and has short ciphertexts. The short ciphertexts come with a prominent trade-off: the key size—and by extension the key generation costs—are larger by a factor that is linear in one of the system parameters compared to other popular schemes [Wat11, RW13]. Hence, if the encryption device is powerful, it is more desirable to use one of those popular schemes. In settings in which the encryption devices can be resource constrained or powerful, we recommend that both a scheme with short ciphertexts and a scheme with more balanced sizes and computational costs are deployed. Because CP-ABE with similar features as [AHM+16] does not exist yet, even though this may be useful in practice, we formulate the following direction.

Direction 2.4: CP-ABE with short ciphertexts

To design an unbounded CP-ABE scheme with short ciphertexts that supports expressive access policies.

2.7.2 Computational costs

For the *computational costs*, the performance of some or all of the algorithms—i.e., the setup, key generation, encryption and decryption—is considered. In practice, some of these algorithms may be performed more often than others. Key generation is ideally run only once for each user, while encryption is performed much more often. In turn, because multiple users can decrypt a ciphertext, decryption may be performed more often than encryption. Furthermore, the encryption and decryption devices may have different computational resources, e.g., the average encryption device may be an IoT device while the average decryption device is a personal computer. It is therefore important to take such practical considerations into account when analyzing the efficiency of a scheme.

2.7.3 Theoretical performance considerations

Oftentimes, the computational efficiency of ABE is theoretically analyzed by counting the operations required by the algorithms. In this way, we can gather rough estimates on the efficiency of certain schemes without requiring any knowledge on cryptographic engineering. Although this approach is simple and may already give a good view on how certain schemes compare, they may fall short when the efficiency of the required operations cannot be effectively estimated. As an example, we theoretically analyze the efficiency of various large-universe schemes (Section 2.5.5), which can be divided into two categories: schemes that support this via an FDH or a polynomial.

For FDH-based large-universe schemes, we know that the schemes are relatively close to their small-universe counterpart. Because the most efficient small-universe constructions incur only a constant number of pairing operations during decryption [Wat11], their large-universe counterparts incur a similar decryption efficiency [AC17a]. However, as we show in Chapter 6, the use of an FDH impedes the key generation and encryption efficiency of the scheme when it is implemented in practice.

For polynomial-based schemes, it is less clear what the most efficient construction is. Some bounded schemes using an n-degree polynomial [Wat08] have an efficient decryption algorithm, as they only require a constant number of pairing operations. However, computing the polynomial-based hash requires n exponentiations. Because encryption needs to evaluate the hash for each attribute, the encryption costs grow linearly in not only the number of attributes, but also the degree of the hash. In contrast, in most unbounded schemes [RW13, ABGW17] (which use a 1-degree polynomial), the key generation, encryption and decryption costs grow only in the number of attributes.

As an exception, the unbounded polynomial-based scheme by Attrapadung et al. [AHM$^+$16] provides a slightly different (but flexible!) efficiency trade-off. Essentially, this scheme is the unbounded variant of the (bounded) constant-size scheme by Attrapadung et al. [ALdP11], using techniques of Lewko and Waters [LW11b] to make it unbounded. In this way, the scheme inherits the unbounded features of linear-sized schemes such as [LW11b, RW13], while it can enjoy the potentially desirable efficiency trade-offs provided by bounded schemes such as [ALdP11]. Importantly, it provides this latter feature flexibly, allowing practitioners to choose the efficiency trade-offs. While the key generation costs grow in the number of attributes and some chosen parameter, the pairing operations required during decryption decrease by a factor in this parameter compared to most unbounded schemes. Note, however, that the number of exponentiations required during decryption increases by this factor, as well as the public and secret keys.

In summary, so far, all existing large-universe constructions incur a significant efficiency trade-off. On the one hand, FDH-based schemes typically have less efficient key generation and encryption algorithms, but more efficient decryption algorithms [Wat11, AC17a, TKN20]. On the other hand, polynomial-based schemes may allow for very efficient implementations of key generation and encryption, but have much less efficient decryption algorithms [LW11b, RW13, ABGW17].

Observation 2.4

The polynomial-based method may have the potential to provide a more flexible scheme in terms of efficiency, and perhaps even a generally more efficient scheme in practice. However, to show that this is the case, more research needs to be conducted in this area. First, it needs to be investigated whether a scheme can be designed that combines the techniques of unbounded schemes such as [RW13] and compact[†] bounded schemes such as [Wat08]. Possibly, this can be achieved with a similar approach as in [AHM$^+$16], which combines the unbounded techniques of [LW11b] and bounded techniques of [ALdP11]. Not only may this allow for the design of schemes with a more flexible efficiency trade-off, but also for the design of e.g., unbounded schemes with a balanced efficiency or with an efficient decryption. Furthermore, it seems that the online/offline variants [HW14] of unbounded schemes [LW11b, RW13] rather explicitly exploit the polynomial structure, which can potentially be generalized. In this way, for all polynomial-based large-universe constructions, the key generation and encryption algorithms can be *implemented* efficiently, requiring almost no computational costs in the online phase. As a more ambitious goal, combining these results—splitting an unbounded scheme with an efficient decryption algorithm into an online and offline phase—may lead to a generally very efficient scheme in practice.

[†] Typically, schemes are considered compact if the asymptotic sizes of the keys and ciphertexts depend, in the worst case, linearly on the sizes of the sets or policies [KW19b], e.g., ciphertexts grow only in the policy length.

To this end, we formulate the following directions.

Direction 2.5: Compact unbounded ABE with flexible efficiency

To construct completely unbounded ABE using the polynomial method with flexible efficiency trade-offs, such that the scheme is compact, i.e., its keys and ciphertexts grow only in the set sizes or policy lengths.

Direction 2.6: Unbounded ABE with efficient decryption

To construct unbounded ABE using the polynomial method that minimizes the required number of pairing operations per attribute in the decryption without sacrificing the storage efficiency.

Direction 2.7: Generic online/offline conversions

To formulate a framework that provides generic conversions for any polynomial-based large-universe scheme to the online/offline setting.

2.7.4 Accurately benchmarking efficiency of ABE

The most empirical way to evaluate and compare the computational efficiency is to implement the scheme(s) and analyze the costs for various numbers of attributes.

However, this effort is difficult, as many relevant aspects influence the scheme's efficiency, and choices need to be made. To benchmark schemes more fairly, we have introduced the ABE Squared framework [dlPVA22], which we treat in Chapter 6.

2.8 Multi-authority ABE

In some schemes, the role of the KGA is shared by multiple authorities, which is called *multi-authority ABE* (MA-ABE) [Cha07]. Specifically, the setup and key generation algorithms are performed (jointly or independently) by these authorities. In most schemes, each authority is responsible for its own unique universe of attributes.

2.8.1 Security against corruption

The security models in the multi-authority setting often capture, in addition to CPA-security and collusion (Section 2.4.2), the notion of *corruption*. Roughly, these models require that security is preserved with respect to the honest authorities even if some authorities are corrupted. This "additional" security guarantee is desirable or even required in multiple-domain settings in which authorities do not (necessarily) trust one another, like in the example in Section 2.2.

2.8.2 The goal of MA-ABE

In literature, there seems to be little consensus in what constitutes secure MA-ABE and how independent the authorities must be. Furthermore, the reason that the role of the KGA is shared by multiple authorities may differ. The security requirements may therefore also differ. Some common security objectives are:

(i) To increase confidentiality: even if some authorities are corrupted, as long as some are honest, the scheme is secure. In particular, the KGAs cannot individually decrypt any ciphertexts (by using their master-key) [LCH+11].

(ii) To mitigate availability issues: even if some authorities are unavailable, users can still request keys for the desired set of attributes [LCLS08].

(iii) To increase independence of the (possibly mutually distrusting) authorities: each domain can assign its own trusted authority without requiring to trust the others; encrypting users can securely use attributes managed by one or more authorities [LW11a].

We observe that these objectives determine the level of security of the scheme and the interdependence of the authorities. For instance, if the objective of the designer is to increase confidentiality (and therefore reduce the trust in the authorities), then the authorities may be more dependent on one another in terms of availability [LCH+11]. Conversely, if the objective is to mitigate availability issues, then there may be less security against corruption [LCLS08].

2.8.3 Distributed and decentralized MA-ABE

We define the notions of distributed and decentralized ABE such that they address the goals about increasing confidentiality and the independence of authorities. In general, both distributed and decentralized ABE cover MA-ABE schemes that are secure against corruption. This also means that the schemes should not be centralized in terms of trust, i.e., corruption of one or more authorities should not result in the breach of security towards the other authorities (conform Section 2.8.1). If the scheme is not secure against corruption, the authorities are always and trivially dependent on one another to act honestly, and by extension, the users need to trust all authorities.

Furthermore, we make a clear distinction between decentralized and distributed ABE depending on how access control decisions are made. In particular, the distinction between the two is in whether the scheme supports (albeit indirectly) decentralized access control decisions, as formulated in Section 2.2. In ABE, this means that, for correctness, the decrypting user only needs to request keys for the attributes she possesses and only from the authorities that manage these attributes. In sum, an MA-ABE scheme is decentralized, if it supports decentralized access control decisions, is secure against corruption and does not employ a centralized authority. Therefore, if an MA-ABE scheme is decentralized, it satisfies the authority-dependence minimization (ADM) property (Section 2.2).

Definition 2.7: Distributed and decentralized MA-ABE

Consider an MA-CP-ABE scheme that is secure against corruption, and that does not have a fully trusted central authority. If it supports expressive access policies (i.e., (N)MSPs) and decentralized access control decisions, it is *decentralized*. Otherwise, it is *distributed*.

Specifically, the scheme supports decentralized access control decisions, if for all access policies $\mathbb{A}$ and all sets of attributes $\mathcal{S}$ that satisfy $\mathbb{A}$, such that

- $\mathrm{CT}_{\mathbb{A}}$ is an encryption of a plaintext M under policy $\mathbb{A}$;
- $\mathrm{SK}_{\mathcal{S}}$ is a secret key associated with the set $\mathcal{S}$, generated by only the associated authorities,

decryption of ciphertext $\mathrm{CT}_{\mathbb{A}}$ with key $\mathrm{SK}_{\mathcal{S}}$ yields the original message M.

Partial and full decentralization. The interdependence of the authorities also depends on the setup, in which the global parameters and master-keys are generated. First, the setup includes the generation of one or more master-keys (from which the users' secret keys are derived), which can be generated either completely independently or distributively. If the master-key is generated distributively, then the users need to request keys from each authority, and therefore the scheme is automatically not decentralized. Second, the global parameters are generated in the setup. These global parameters may be associated, albeit implicitly, with secret information on which the (MA-)security of a scheme relies. Therefore, these parameters should not be generated by a single authority if security against corruption is required. Examples of such parameters are a composite-group order N [LW11a], or multiple generators within the same group. Knowing the factorization of N or the relative discrete logarithm between two generators often enables attacks on the scheme. However, the secrets associated with these parameters do not need to be kept after the global setup has finished (unlike the master-keys). It might thus be acceptable that some initial authorities jointly and *distributively* generate these parameters in a secured environment, and afterwards, the associated secret data are destroyed. It is then assumed that these authorities cannot retrieve the secret data later. Because these parameters cannot be generated independently, a scheme cannot be fully decentralized. As such, we call a decentralized scheme that generates global parameters associated with secret information *partially* decentralized, and otherwise, *fully* decentralized.

Decentralization to foster availability. Interestingly, decentralized ABE provides some additional security with respect to the availability of the authorities. We already covered the notion of corruption in the security model (Section 2.8.1), which ensures that security in the form of confidentiality is still preserved with respect to the honest authorities if some authorities are corrupted. In practice, such corruption may also cause issues with respect to the availability of the authorities [MKE08]. If authorities are suddenly unavailable, they cannot issue keys anymore, which may break the entire system. That is, users entering the system after such corruption may not be able to decrypt any old ciphertexts despite being authorized to do so. Because corruption may be easier to instigate, e.g., through denial-of-service attacks, than the retrieval of an authority's secret keys, security against this type of attack is at least as relevant. Similarly, the availability of authorities may depend on whether they want to leave the system voluntarily, or alternatively, new authorities may want to join.

2.8.4 Achieving security against corruption

We briefly review the ways in which security against corruption is achieved in existing schemes. As we have seen in the standard form (Definition 2.6), a scheme consists of various variables that need to be secret. Notably, the master-key α can be used to decrypt any ciphertext, and knowing some common variables $\mathbf{b}$ may also lead to significant breaks (Chapter 5). Most schemes ensure security against corruption by letting each authority hold its own (partial) master-key and its own common variables $\mathbf{b}$ (whereas other schemes distribute the generation of these variables). Furthermore, the user-specific random variables $\mathbf{r}$ introduced in the key generation link the keys to one specific user. It is paramount that this randomness is unique to avert any collusion attacks. In addition, a scheme oftentimes breaks if the randomness is known, so it should not be retrievable by any attackers (which may be a corrupt authority).

Hence, to ensure security against corruption, this randomness is either generated distributively by all authorities [CC09] or provided by a full-domain hash [LW11a].

In general, achieving decentralized ABE is more difficult than achieving distributed ABE. One of the difficulties is in making the scheme collusion resistant. In centralized schemes, encryption uses only one master-key, while in distributed and decentralized schemes, encryption may need to use master-keys managed by several authorities. Roughly, the part that hides the message, i.e., $e(g, h)^{\alpha s}$, in Definition 2.6 is replaced by $e(g, h)^{\alpha_i s}$, where α_i is the master-key of the i-th authority. Then, colluding users should not be able to jointly decrypt by individually computing these "randomized master-keys", therefore partially decrypting the ciphertext. Chase and Chow [Cha07, CC09, Cho16] ensure this by generating the system-wide master-key distributively, and they compute a user's secret key by generating a different sharing of the master-key for each user. As such, a decrypting user only obtains the randomized system-wide master-key, i.e., $e(g, h)^{\alpha s} = \prod_i e(g, h)^{\alpha_i s}$, if her own secret keys are used. However, recall that distributing the master-key also ensures that the ADM property cannot hold, because users trivially need to request keys from each authority.

To overcome this restriction, Lewko and Waters [LW11a] use a different method to tie the partially decrypted ciphertexts to one user. Specifically, they include a zero-sharing associated with the access policy in the ciphertext, which can only be canceled, if it is combined with keys that use the same user randomness during decryption. Because this user randomness needs to be the same for each authority and authorities are not supposed to interact with one another, it is implicitly provided by a full-domain hash. As a trade-off, these schemes [LW11a, RW15] typically have much larger ciphertexts, because a linear number of elements exists in group $\mathbb{G}_T$. In addition, arithmetic in group $\mathbb{G}_T$ is less efficient, and the use of FDHs may further reduce the efficiency of a scheme (Section 2.7). In contrast, schemes that distribute the master-key like Chase and Chow [Cha07, CC09, Cho16] are structurally often closer to the single-authority version of the scheme, and consequently retain a similar efficiency. Importantly, because all decentralized schemes require a full-domain hash for its user-specific randomness, they cannot benefit from an anonymous key issuance protocol [CC09]. Removing the full-domain hash may mitigate or even solve some of these issues. As such, we formulate the following directions.

Direction 2.8: Decentralized ABE conform the standard form

To devise decentralized ABE that is, in structure, closer to single-authority ABE, i.e., conform the standard form (Definition 2.5.4), and thus attains a similar efficiency.

Direction 2.9: Decentralized ABE without full-domain hash

To design a decentralized ABE scheme without requiring full-domain hashes.

Table 2.1. Taxonomy of multi-autority ABE. We list whether the scheme is key-policy (KP) or ciphertext-policy (CP) based, whether they support MSPs, whether they are proven to be secure against corruption (MA-sec), whether the identity is embedded into the keys with a full-domain hash (FDH) or not, whether the scheme is decentralized (Dec), and whether it satisfies the ADM property. For the global parameters (GP), we consider whether the scheme requires the generation of a composite order N or multiple generators (G).

Scheme	KP/CP	MSP	MA-sec	GP	ADM	Dec	No FDH
[Cha07]	KP	✓	$\checkmark_A$	G	✗	✗	✓
[CC09]	KP	✓	✓	-	✗	✗	✓
[LHC$^+$11]	KP	✗	✓	-	✗	✗	✓
[LW11a, LW10a] I	CP	✓	✓	N	✓	◑	✗
[LW11a, LW10a] II	CP	✓	✓	-	✓	●	✗
[LCH$^+$11]	CP	✓	✓	N	✗	✗	✓
[OT13]	CP	✓	✓	G	✓	◑	✗
[RW15]	CP	✓	✓	-	✓	●	✗
[MJ18]	CP	✗	✓	G	✗	✗	✗

$\checkmark_A$ = only for the attribute authorities; ◑ = partially, ● = fully;
N = composite order; G = multiple generators

2.8.5 Taxonomy of MA-ABE schemes

Throughout this section, we have analyzed several MA-ABE schemes. We provide a taxonomy for multi-authority ABE in which we evaluate the properties specific to the multi-authority setting. Table 2.1 lists these results. Note that the table lists much fewer properties than the taxonomy in Section 2.11. For an evaluation of the other properties, we refer to Table 2.2 in Section 2.11.

Table 2.1 shows that only two fully and two partially decentralized schemes with expressive access structures exist. They all employ full-domain hashes to generate the secret randomness for each user. Nevertheless, these four (partially) decentralized ABE schemes satisfy the user independence and authority-dependence minimization properties, and therefore provide practical solutions to enforcing access control.

2.9 Towards (formalizing) resilient ABE

We discuss the resilience of ABE. Many works on ABE have considered the notion of security in the form of confidentiality or integrity, covered by CPA- and CCA-security. However, the notion of availability is also important in practice. We define the notions of attribute resilience and attribute-wise key generation. These two properties capture a level of resilience of a scheme such that availability issues can be mitigated by minimizing the involvement of the authorities.

2.9.1 Attribute resilience

In practice, the universe of attributes may not be static. For instance, as new users join the system, new attributes (values) for e.g., names may have to be added [HFK$^+$19, ETS18a]. In this case, small-universe constructions require that a new public key is generated for each new attribute. This may also impact the previously generated keys and ciphertexts. For example, some schemes associate the keys and ciphertexts with the whole universe, e.g., [CN07, NYO08]. Keys generated before the addition of an attribute and its associated public key may not be able to decrypt a newly encrypted message despite its authorized status. As such, all keys and ciphertexts need to be updated to make the system functional again. Large-universe constructions do not have this problem, because the public keys can be generated from any attribute string, and keys and ciphertexts are therefore never associated with the whole universe.

More generally, we propose the notion of *attribute resilience*, which protects against such problems. It captures the correctness and security of a scheme with respect to the universe of attributes at any given time. That is, at a certain time, the universe of attributes may be larger than at an earlier point in time. In many settings, it is reasonable to require that e.g., ciphertexts encrypted at the earlier point in time are decryptable by a key generated at a later point, provided that it is authorized to do so. However, certain types of schemes are by definition not resilient, and can therefore not be efficiently used in such dynamic settings.

The definition of attribute resilience can be expressed as an extra condition on the correctness and security of the scheme. On the one hand, a ciphertext should be decryptable by an authorized key, regardless of the associated universes of attributes at the time of generation. On the other hand, a ciphertext should *never* be decryptable by an unauthorized key either. Such an additional requirement on the definition, however, requires that the formal definition and security model of the scheme need to be adjusted, e.g., because the universe needs to be taken as input to the algorithms. We formalize attribute resilience and its associated security definition in the full version of the paper on which this chapter is based [VAH21].

2.9.2 On the resilience of ABE supporting non-monotonicity

We observe that ABE that supports non-monotonicity has issues with regard to the attribute resilience. To show this, we discuss the non-resilience that follows from applying the three methods described in Section 2.5.7.

The first method: negative attributes. Schemes supporting non-monotonicity using the first method (Section 2.5.7)—i.e., using negative instances of attributes to implement negations—are rendered incorrect when attributes are added. Suppose that $\mathcal{U}$ and $\mathcal{U}'$ are the universes associated with some key and ciphertext, respectively. Let the access policy of the ciphertext be such that it requires the possession of a negative attribute in $\mathcal{U}' \setminus \mathcal{U}$. Then, the key cannot decrypt the ciphertext, despite possibly not having the attribute and thus satisfying the policy.

The second method: OSW-method. Schemes supporting non-monotonicity using the second method, i.e., the OSW-method, as considered in Section 2.5.7 are rendered insecure when attributes are added. The reason for this is that the OSW-method is combined with the polynomial-based method to support large universes (Section 2.5.5). On the one hand, this allows users to use any conceivable input string as attribute. On the other hand, this lack of control over the universe may yield insecurity in the non-monotonic setting. As we previously discussed, the negations used in the OSW-method require the decrypting user to compare the entire set of attributes associated with the key with the negated attribute. However, depending on how the scheme is implemented in practice, the decrypting user could possess the negated attribute, despite not having the associated key. For example, the KGA may not be able to check whether the user has revealed all relevant attributes in their possession, or a new label is added after the secret keys are generated (as mentioned in Section 2.5.7). It is therefore possible that a decrypting user can decrypt a ciphertext despite not being authorized to do so, and the scheme is consequently insecure. (Note that this issue was also pointed out by Attrapadung and Tomida [AT20], who introduced the notion of labeled non-monotonicity to mitigate such issues.)

The third method: labeled non-monotonicity. Schemes supporting a labeled non-monotonicity with the third method (Section 2.5.7) may provide a more acceptable trade-off in (non-)resilience. As discussed, it is useful that ciphertexts for which the access policy specifies a negation for a label, for which the decrypting user does not have a key yet, cannot be decrypted. This ensures that the owner of the key cannot unjustifiably decrypt a ciphertext despite possessing the negated attribute (value) but not possessing the associated key. In this case, it is reasonable to say that the incorrectness that follows from this lack of attribute(-label) resilience trumps the insecurity that might have otherwise followed. The lack of resilience with respect to the correctness is less disruptive in the third method than in the first method. Whereas this non-resilience can occur at any addition of an attribute (value) in the first method, this only happens when an attribute label is added in the third method.

2.9.3 Attribute-wise key generation

To minimize the required computational power of the KGA, we introduce the notion of *attribute-wise key generation*. An additional benefit of decentralized CP-ABE [LW11a, RW15] is that the user's secret keys can be generated incrementally, i.e., one partial secret key for each attribute. In contrast, existing single-authority schemes require users to request secret keys for the entire set of attributes they possess. The keys are mathematically linked to a single user by using the same user-specific randomness for each attribute such that users cannot collude. In most decentralized schemes, this randomness is provided by a full-domain hash, which deterministically and implicitly generates randomness for any given identity. It therefore allows the KGA to link the new "attribute keys" incrementally to the identity instead of all at once.

In general, the advantage of such an "attribute-wise key generation" is that it is more efficient and practical. Especially in dynamic settings, in which new attribute values, labels or types may be frequently added, attribute-wise key generation may be more efficient. For instance, suppose a recently added attribute (label or value) is used in a ciphertext access policy, and a decrypting user possesses the attribute but not the associated secret key. Then, a partial secret key can be requested for the new attribute (value) only. This reduces computational costs for the key generation authority significantly. By extension, key requests can be handled more quickly, and thus more key requests can be processed, which fosters availability.

The construction of a scheme with an attribute-wise key generation is an open problem. Some CP-ABE schemes seem to have a structure that allows for the construction of such an attribute-wise key generation by using an FDH like in the decentralized setting. However, the main difficulty in constructing such a scheme lies in the security proof. The existing security models consider key queries for sets of attributes (which are thus specified before the key is generated). In contrast, the security model associated with a scheme that supports attribute-wise key generation would have to take into account that keys can be queried gradually for any user and attribute. For instance, selective security proofs (e.g., [Wat11, RW13]) embed the entire set of attributes in all key components, so it is unlikely that these proofs carry over to any such new security model without some adjustments. To solve this, Rouselakis and Waters [RW15] use a static security model. In this model, the sets of attributes for which the attacker is going to query keys are determined during the initialization phase (Definition 2.4). It is unclear, however, if such a scheme can be designed without resorting to weaker models.

Direction 2.10: Attribute-wise key generation

To construct a scheme with attribute-wise key generation (that is secure in a non-static model).

Observation 2.5: Non-monotonicity and attribute-wise key generation

For some methods that are used to support non-monotonicity, attribute-wise key generation is impossible. Intuitively, this follows from the converse of the definition of monotonicity (Definition 2.2). If an access structure $\mathbb{A}$ is not monotone, then B, C exist such that $B \in \mathbb{A}$ and $B \subseteq C$, but not $C \in \mathbb{A}$. Suppose now that $\mathbb{A}$ is used during encryption. Then, some user, who possesses a set of attributes

C, but only has secret keys for the subset B (because she has not requested keys for the rest of the attributes $C \setminus B$ yet), can decrypt the ciphertext even though she does not satisfy the policy.

Fortunately, not all three methods discussed in Section 2.5.7 satisfy this definition of non-monotonicity, and seem to be able to support attribute-wise key generation in some cases. The first method essentially uses monotone structures by pushing the negation to the value, e.g., "profession: non-doctor" instead of "NOT profession: doctor". The second method, on the other hand, does satisfy this definition of non-monotonicity. We also observe that all schemes that use the OSW-method tie all "attribute keys" together to one specific user by distributing the user randomness in a certain way. The KGA can therefore not generate these keys independently of the rest of the attribute keys, and we can thus not define an attribute-wise key generation for these schemes. Because the third method uses a labeled non-monotonicity, e.g., "profession: NOT doctor", it is possible to incrementally generate keys for the labels, but not for the attributes values with the same label. This is however only secure, if the KGA can check whether the user requests keys for all values that she possesses for the label for which she requests keys.

2.10 Additional functionality

Some schemes that we have analyzed support additional functionality, achieving more properties than we have discussed so far. These properties may enhance ABE in certain aspects, which may make them more practical. Since the focus of this chapter is on the core properties of ABE, we introduce some of these properties only briefly.

2.10.1 Online/offline key generation and encryption

Unbounded ABE schemes [LW11b, RW13] based on the polynomial method discussed in Section 2.5.5 can be implemented more efficiently. The key generation and encryption algorithms can be efficiently split into an online and offline phase [HW14], such that the offline phase covers most of the computational costs, while the online phase requires minimal computations. (In contrast, key generation and encryption of FDH-based large-universe schemes could also be split into two phases, but would always require a linear number of exponentiations during online time.) The key generation efficiency is enhanced, because the KGA can securely precompute large batches of secret key material, and needs to perform only few computations in a key request. As a small trade-off, the user needs to put in more computational effort. The encryption efficiency is similarly enhanced, but at the cost of a diminished decryption efficiency. Furthermore, the ciphertexts increase significantly in size. Nevertheless, this implementation of encryption is especially useful for e.g., devices with limited power resources.

2.10.2 Revocation

In access control systems, the access privileges of users may be revoked, e.g., because their credentials have been stolen or have expired. A user's secret keys then need to be disabled such that she cannot decrypt any ciphertexts for which she has lost authorization. For ABE, revocation is significantly more troublesome, because—unlike in traditional public-key encryption—several distinct users might hold secret keys associated with the same attributes [PTMW10]. Many works have addressed revocation by means of different approaches, which can be classified in two categories: user-based [AI09b, IPN^{+}09, YWRL10, LSW10, Cho16, CDLQ16] and time-based [AI09a, SSW12, LYZL18]. Some of these works describe generic approaches to supporting revocation [Cho16, SSW12]. Because of the importance of revocation in practice and the vast body of (non-generic) work in this subfield of ABE, we recommend that any such approaches are systematized and generalized. Then, these methods can be applied to any scheme, generically, rather than to a single scheme.

Direction 2.11: Systematizing and generalizing revocation methods

To systematize and generalize revocation methods such that they can be applied generically to many or even all ABE schemes.

2.10.3 Hidden policies – attribute-hiding ABE

Some schemes are designed such that the access policies can be securely hidden, which is called *attribute-hiding* ABE [BW07, KSW08, NYO08, OT10, Wee17]. Because the predicate (e.g., access policy) associated with a ciphertext may leak information about the sender or receiver, this makes ABE more privacy friendly. Interestingly, the generic framework by Chen, Gay and Wee (CGW15) [CGW15] provides a modular approach to achieving a weaker notion of attribute-hiding called *weakly attribute-hiding*. In this notion of attribute-hiding, the decrypting user is only unable to learn anything about the access policy, if the policy is not satisfied by the set of attributes. In contrast, in fully attribute-hiding schemes, the decrypting does not learn anything about the access policy except whether the set of attributes satisfies it, even if the set does satisfy the policy. Regardless, so far, no attribute-hiding ABE schemes have been proposed that support large universes and are expressive. Presumably, the reason for this is that the expressivity and the (weakly) attribute-hiding properties are seemingly incompatible. To decrypt, the user needs to know how to recover the row $(1, 0, ..., 0)$ from the rows of the policy matrix (Definition 2.5), which can only be done efficiently if the rows are known. Furthermore, because many expressive schemes have ciphertexts that are linear in the size of the policy, the size of the policy is not hidden either. As a "best-of-both-worlds" solution, Wee [Wee17] proposes the notion of *partially-hiding*. This notion applies to a special family of predicates that are composed of two single-input predicates. For example, the first input predicate could be for MSPs and the second for single identities. The partially-hiding property only requires the second input predicate to be hidden, and not the first. Possibly, this approach can be used to construct an ABE scheme for MSPs where the attributes are split into a public part—e.g., the attribute type—and a hidden part—e.g., the attribute value. To devise such a scheme, we formulate the following future direction.

Direction 2.12: Partially-hiding ABE for MSPs

To devise a partially-hiding ABE for MSPs, and to investigate whether they can support practical properties such as large-universeness and non-monotonicity.

2.10.4 Outsourced decryption

To mitigate the decryption costs on the user side, decryption can be securely outsourced to a third party with better computational resources [GHW11]. The approach was later generalized in [Cho16], and may thus be applicable to various existing schemes. This property can also be combined with server-aided revocation techniques [CDLQ16]. While outsourced decryption may solve some issues with the decryption efficiency in practice, we believe that this particular property may somewhat undermine the user independence property—which mitigates availability issues in access control systems—and is one of the main advantages of ABE. As such, it should be carefully considered whether the entity assigned for outsourced decryption is sufficiently reliable in terms of availability.

2.10.5 Traceability

In some settings, users may attempt to share their access privileges with other users (e.g., for financial gain). This can be mitigated with traitor-tracing schemes [LHC$^+$11], and allows the authorities to trace these malicious users. Notably, Lai and Tang [LT18] devised a practical generic approach that can be applied to any existing scheme.

2.11 Taxonomy: classifying existing schemes

Throughout this work, we have analyzed over fifty ABE schemes with respect to the described properties. These schemes and our analysis are summarized in Table 2.2. Each property that we have discussed in Sections 2.3-2.8 and 2.10 is considered in the table. We illustrate the significance of each scheme by highlighting the satisfied properties in green. Note that the table does not include schemes that can be instantiated in the works mentioned in Section 2.6.2, as the underlying choices influence several properties, such as the group order and the level of security.

Table 2.2. Taxonomy: classification of existing schemes

Scheme	Based on	KP/CP	Expr.	LU	L-F	Bound	PO	Type	CS-CT	Sec.	CCA*	SA	No ROM	MA	Add. func.
[SW05] I	-	KP	T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✓	✗	-
[SW05] II	-	KP	T	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✓	✗	-
[GPSW06a] I	[SW05] I	KP	MSP	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✓	✗	-
[GPSW06a] II	[SW05] II	KP	MSP	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✓	✗	-
[Cha07] I	[SW05] I	KP	MA-T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✓	◑	-
[Cha07] II	[GPSW06a] II	KP	MA-MSP	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✓	◑	-
[BSW07]	[GPSW06a] II	CP	MSP	✓	-	-	✓	I	✗	●	$\checkmark_{D,V}$	GGM	✗	✗	-
[CN07]	[GPSW06a] I	KP*	AND	✗	✗	-	✓	I	✗	◑	✓	Static	✓	✗	-
[OSW07]	[GPSW06a] II	KP	NMSP	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✗	◑	$\checkmark_V$	Static	✓	✗	-
[NYO08] I	[CN07] I	KP*	AND	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	Static	✓	✗	AH
[NYO08] II	[BSW07]	CP	AND	✗	✗	-	✓	II	✗	◑	$\checkmark_{D,V}$	GGM	✓	✗	AH
[CC09]	[Cha07]	KP	MA-T	✗	✗	-	✓	I	✗	◑	$\checkmark_V$	$q(\lvert\mathcal{A}\rvert)$	✓	◑	-
[YWRL10]	[CN07] I	KP*	AND	✗	✗	-	✓	I	✗	◑	✓	Static	✓	✗	R
[HLR10]	-	CP	T	✗	✗	-	✓	I	✓	◑	✗	$q(\lvert\mathcal{U}\rvert)$	✓	✗	-
[LOS$^+$10]	[Wat09]	CP	MSP	✗	✓	OU	✗	I	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[OT10]	[LOS$^+$10]	KP,CP	NMSP	✓	-	OU,$\lvert\mathbb{A}\rvert$	✓	III	✗	●	✓	Static	✓	✗	-
[Wat11] I	[LOS$^+$10]	CP	MSP	✗	✓	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	-
[Wat11] II	[LOS$^+$10]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\text{AND})$	✓	✗	-
[Wat11] III	[LOS$^+$10]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✓	✗	-
[LHC$^+$11]	[NYO08, CC09]	KP*	AND	✗	✗	-	✓	II	✗	◑	$\checkmark_V$	$q(\lvert\mathcal{A}\rvert)$	✓	◑	AH, T
[ALdP11]	-	KP	(N)MSP	✓	-	$\lvert\mathcal{S}\rvert$	✓	I	✓	◑	$\checkmark_{D,V}$	$q(\lvert\mathcal{S}\rvert)$	✓	✗	-
[LW11a] I	[LOS$^+$10]	CP	MSP	✗	✗	OU	✗	I	✗	●	✗	Static	✗	●	-
[LW11a] II	[LOS$^+$10]	CP	MSP	✗	✗	OU	✓	I	✗	◑	✗	GGM	✗	●	-
[LW11b]	[LOS$^+$10]	KP	MSP	✓	-	-	✗	I	✗	◑	$\checkmark_{D,V}$	Static	✓	✗	-
[GHW11] I	[Wat11]	CP	MSP	✓	-	-	✓	I	✗	◑	✗	$q(\lvert\mathbb{A}\rvert)$	✗	✗	OD
[GHW11] II	[GPSW06a]	KP	MSP	✓	-	-	✓	I	✗	◑	✗	Static	✗	✗	OD
[LCH$^+$11]	[LOS$^+$10]	CP	MSP	✗	✗	-	✗	I	✗	●	$\checkmark_{D,V}$	Static	✓	◑	-
[LW12]	[LOS$^+$10]	CP	MSP	✗	✓	-	✗	I	✗	●	$\checkmark_{D,V}$	$q(\text{AND})$	✓	✗	-
[SSW12]	[LOS$^+$10]	KP,CP	MSP	✗	✓	OU	✗	I	✗	●	✗	Static	✓	✗	R
[OT12]	[OT10]	KP,CP	NMSP	✓	-	-	✓	III	✗	●	$\checkmark_V$	Static	✓	✗	-
[HW13]	[GPSW06a] II	KP	MSP	✗	✗	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathcal{U}\rvert)$	✓	✗	-
[CCL$^+$13]	[GZC$^+$12]	KP	T	✓	-	$\lvert\mathcal{S}\rvert$	✗	I	✓	●	✗	Static	✓	✗	-
[OT13]	[OT10]	KP,CP	NMSP	✓	-	$\lvert\mathbb{A}\rvert$	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	●	-
[LCL$^+$13]	[SW05]	KP	T	✗	✓	-	✓	I	✗	◑	$\checkmark_{D,V}$	Static	✓	✗	R, OD
[RW13]	[LW11b]	KP,CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	-
[LCW13]	[LW12, GKSW10]	CP	MSP	✗	✓	-	✗	I	✗	●	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	T
[HW14]	[RW13]	KP,CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	OO
[KL15]	[LOS$^+$10]	KP	MSP	✗	✗	-	✗	I	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[RW15]	[LW11a]	CP	MSP	✓	-	-	✓	I	✗	◔	✗	$q(\lvert\mathbb{A}\rvert)$	✗	●	-
[CGW15]	[Wat11] II	KP,CP	MSP	✗	✓	OU,$\lvert\mathbb{A}\rvert$	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[LW15]	[LCW13, RW13]	CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	R, T
[ZGT$^+$16] I	[CW14a, Tak14]	KP	MSP	✗	✗	-	✓	III	✓	◕	$\checkmark_{D,V}$	Static	✓	✗	-
[ZGT$^+$16] II	[CW14a, Tak14]	KP	MSP	✓		$\lvert\mathcal{S}\rvert$	✓	III	✓	◕	$\checkmark_{D,V}$	Static	✓	✗	-
[AHM$^+$16]	[LW11b, Att14a]	KP	MSP	✓	-	-	✗	III	✓	●	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	-
[CDLQ16]	[RW13]	CP	MSP	✓	-	-	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\lvert\mathbb{A}\rvert)$	✓	✗	R, OD
[ABGW17]	-	KP,CP	MSP	✓	-	-	✓	III	✗	●	$\checkmark_{D,V}$	GGM	✓	✗	-
[AC17a]	[CGW15]	KP,CP	MSP	✓	-	OU	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	-
[CGKW18]	[RW13, Att14a]	KP,CP	MSP	✓	-	OU	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[LYZL18]	[Wat11, BBG05]	CP	MSP	✗	✓	OU	✓	I	✗	◑	$\checkmark_{D,V}$	$q(\text{AND})$	✓	✗	R
[MJ18]	[CGW15]	CP	T	✗	✗	-	✓	III	✗	●	✗	Static	✗	◑	AH
[KW19b] I,II	[LW12]	KP,CP	BF	✗	✗	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[KW19b] III	[LW11b]	KP	BF	✓	-	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✓	✗	-
[TKN20]	[AC17a, KW19b]	KP,CP	NM-BF	✓	-	-	✓	III	✗	●	$\checkmark_{D,V}$	Static	✗	✗	-

Core properties

KP = key-policy, CP = ciphertext-policy, KP* = KP implemented as CP, Expr. = expressivity, LU = large-universe, L-F = extendable to LU with FDH (only listed if not already supported), T = threshold function; (NM-)BF = (non-monotone) Boolean formulas; (N)MSP = (non-)monotone span program; OU = one-use, $\mathbb{A}$ = policy, $\mathcal{S}$ = attribute set

Efficiency

PO = prime-order group; CS-CT = constant-size ciphertext

Security

CCA* = security against CCA from using verifiability (V) and delegatability (D); SA = security assumption, ROM = random oracle model, GGM = generic group model; q(par) = q-type assumption depends on par; ● = full, ◕ = semi-adaptive, ◑ = selective, ◔ = static security;

Additional functionality

MA = multi-authority ABE, AH = attribute-hiding, OD = outsourced decryption, T = traceability, R = revocation, OO = online/offline, ● = decentralized, ◑ = distributed

2.12 Future work and conclusion

We showed that over the past two decades, much progress has been made in the context of pairing-based ABE. Especially with respect to security, a wide variety of techniques and frameworks has been developed to construct ABE with the required security guarantees. Selectively secure schemes can now be converted into fully secure schemes, incurring relatively little sacrifice in efficiency. Furthermore, such security guarantees can be achieved whilst retaining or achieving all desirable core properties. From a practical viewpoint, we are therefore optimistic that the existing techniques and frameworks are sufficient in the design of future schemes. However, we may need to settle for full security under parametrized assumptions rather than static ones.

Nevertheless, it seems that the focus on improvements in security has resulted in a declined interest in improving the general efficiency and other practical aspects of ABE within the theoretical community. As we have shown, achieving some desirable properties simultaneously may result in the inability to support other properties. For instance, the simultaneous support of large universes and non-monotone access structures negatively affects the resilience and efficiency of a scheme. To shift the attention to the practical issues outlined in this chapter, we have proposed several directions for future research that we encourage the community to address. In addition, we encourage the practical community to seriously consider ABE in the design of cryptographically-enforced access control. At this point, ABE is already at a reasonably advanced stage and may be beneficial in settings for which no practical and secure alternatives exist.

In particular, we motivated that ABE provides a practical and secure solution to enforcing access control. We explained the problem through a simple use case of an EHR system in the multiple-domain setting. In Section 2.2, we identified two properties to increase availability and scalability: user independence and authority-dependence minimization. These properties minimize the role of the authorities in the enforcement of access control. In general, ABE provides user independence. After the users have received secret keys from the authority, they can decrypt any ciphertexts for which they are authorized without requiring interaction with the authority again. The authority-dependence minimization property is relevant in the multi-authority setting and minimizes the number of authorities with which the user has to interact. In particular, this property ensures that decrypting users do not have to interact with authorities for which they have no relevant attributes. We defined the notion of decentralized ABE such that decentralized schemes satisfy the ADM property. Because decentralized ABE additionally provides security against corruption, it ensures that access control can be securely enforced by various (possibly mutually distrusting) authorities. This makes decentralized ABE especially attractive as a solution in multiple-domain settings. We showed that several schemes are decentralized, but that these can benefit from improvements.

We posed directions for future research (Section 2.12.1) to mitigate the disadvantages of ABE while maintaining and even amplifying its benefits. Currently, the main disadvantage of ABE is the general inefficiency compared to other cryptographic primitives. In contrast, other primitives such as traditional public-key encryption, proxy re-encryption [BBS98] and identity-based proxy re-encryption [GA07] are more efficient on the user side. For this reason, many of our directions for future research address the efficiency of ABE. Furthermore, the other directions address features that ABE provides or can potentially provide that other solutions may not be able to sufficiently address. By exploring these directions, ABE becomes even more practical and secure, reaching its full potential as a mechanism to implement efficient and secure access control in practice.

2.12.1 Future directions addressed in this thesis

In this thesis, we address the following future directions:

- Direction 2.3: Efficient CCA-conversion for CP-ABE
- Direction 2.4: CP-ABE with short ciphertexts
- Direction 2.5: Compact unbounded ABE with flexible efficiency
- Direction 2.6: Unbounded ABE with efficient decryption
- Direction 2.7: Generic online/offline conversions
- Direction 2.8: Decentralized ABE conform the standard form
- Direction 2.10: Attribute-wise key generation
- Direction 2.11: Systematizing and generalizing revocation methods
- Direction 2.12: Partially-hiding ABE for MSPs

In particular,

- Chapter 4 addresses, to some extent, Directions 2.8, 2.10 and 2.11;
- Chapter 7 addresses Directions 2.5 and 2.6, and to some extent, Direction 2.7;
- Chapter 8 addresses Direction 2.4;
- Chapter 9 addresses Direction 2.3.

Chapter 3

Theoretical background

In this chapter, we give further theoretical background needed in the remainder of this thesis. In particular, it contains notations, formal definitions and otherwise important results such as lemmas and theorems.

3.1 Notation

We use λ to denote the security parameter. A negligible function parametrized by λ is denoted as $\mathrm{negl}(\lambda)$. If an element x is chosen uniformly at random from a finite set S, then we denote this as $x \in_R S$. If an element x is produced by running algorithm Alg, then we denote this as $x \leftarrow \mathrm{Alg}$. If a set of attributes $\mathcal{S}$ satisfies some access structure $\mathbb{A}$, we denote this as $\mathbb{A} \models \mathcal{S}$. If not, then we denote this as $\mathbb{A} \not\models \mathcal{S}$. We use $\mathbb{Z}_p := \mathbb{Z}/p\mathbb{Z}$ for the ring of integers modulo p. For integers $a < b$, we denote $[a, b] = \{a, a+1, ..., b-1, b\}$, $[b] = [1, b]$ and $\overline{[b]} = [0, b]$. We use boldfaced variables $\mathbf{M}$ and $\mathbf{v}$ for matrices and vectors, respectively, where $(\mathbf{M})_{i,j}$ denotes the entry of $\mathbf{M}$ in the i-th row and j-th column, and $(\mathbf{v})_i$ denotes the i-th entry of $\mathbf{v}$. We use the following shorthand notation for vectors in the exponent: $g^{\mathbf{v}} = (g^{(\mathbf{v})_1}, g^{(\mathbf{v})_2}, ...)$. If $\mathbf{v}$ is a vector whose entries are polynomial functions over variables $\mathbf{x} = (x_1, ...)$, we sometimes write $\mathbf{v}(\mathbf{x})$ to indicate this. For access policies $\mathbb{A} = (\mathbf{A}, \rho)$, we denote the entries as $A_{j,k}$ (for all $j \in [n_1], k \in [n_2]$). We denote $a \ : \ \mathbf{A}$ to substitute variable a by a matrix $\mathbf{A}$. We define $\mathbf{1}_{i,j}^{d_1 \times d_2} \in \mathbb{Z}_p^{d_1 \times d_2}$ as the matrix with 1 in the i-th row and j-th column, and 0 everywhere else, and similarly $\mathbf{1}_i^{d_1} \in \mathbb{Z}_p^{d_1}$ and $\overline{\mathbf{1}}_i^{d_2} \in \mathbb{Z}_p^{d_2}$ as the row and column vectors with 1 in the i-th entry and 0 everywhere else. If some algorithm yields no output or outputs an error message, then we use $\perp$ to indicate this. We use $a \| b$ to indicate that two strings a and b are concatenated. Sometimes, we use the *implicit representation used for group elements* in [EHK+13]. This representation is quite often used in ABE literature to simplify the description of schemes [CGW15, CGKW18, KW19b, TKN20]. In particular, suppose $g' \in \mathbb{G}'$ is a generator of some group $\mathbb{G}'$, then we use $[x]_{\mathbb{G}'}$ to denote the element g'^x.

3.2 Pairings (or bilinear maps)

Following Section 2.5.1, we define pairings as follows.

Definition 3.1: Pairings

We define a pairing to be a map e on three groups $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ of prime order p, so that $e\colon \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$, with generators $g \in \mathbb{G}, h \in \mathbb{H}$ is such that the following properties hold:

- **Bilinearity:** for all $a, b \in \mathbb{Z}_p$, it holds that $e(g^a, h^b) = e(g, h)^{ab}$;
- **Non-degeneracy:** for $g^a \neq 1_{\mathbb{G}}, h^b \neq 1_{\mathbb{H}}$, it holds that $e(g^a, h^b) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}'}$ denotes the unique identity element of the associated group $\mathbb{G}'$
- **Efficiency:** e is efficiently computable.

We refer to $\mathbb{G}$ and $\mathbb{H}$ as the two *source groups*, and $\mathbb{G}_T$ as the *target group*.

If an efficiently computable non-trivial isomorphism exists between $\mathbb{G}$ and $\mathbb{H}$, i.e., $\mathbb{G} \cong \mathbb{H}$, we call the pairing symmetric or of type I. If $\mathbb{G} \ncong \mathbb{H}$, we call the pairing asymmetric. Specifically, if an efficiently computable non-trivial homomorphism exists from $\mathbb{H}$ to $\mathbb{G}$ (but not from $\mathbb{G}$ to $\mathbb{H}$), we call the pairing of type II, and if there exists no such efficiently computable homomorphism, we call the pairing of type III. If we want to explicitly consider type-I pairings, we denote the pairing as $\hat{e}$. Otherwise, we use simply e.

3.3 Formal definitions and security models

3.3.1 Full security against chosen-ciphertext attacks

The model for security against chosen-ciphertext attacks as introduced in Section 2.4.3 is formally defined as an extension of the CPA-security model in Definition 2.4. In particular, we extend the query phases with the option to make decryption queries for ciphertexts (that are not equal to the challenge ciphertext).

Definition 3.2: Full security against chosen-ciphertext attacks (CCA)

We define the security game IND-CCA(λ) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.

- **First query phase:** The attacker can make two types of queries:
  - **Key query:** The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
  - **Decryption query:** The attacker sends a ciphertext CT_x for $x \in \mathcal{X}$ and some $y \in \mathcal{Y}$ is such that $P(x, y) = 1$, to the challenger, who returns the message $M \leftarrow \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x)$ (where $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$).
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first key query phase, we have $P(x^*, y) = 0$, and generates two messages M_0 and M_1 of equal length in $\mathcal{M}$, and sends these to the challenger. The challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}^*_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*, M_\beta)$, and sends the resulting ciphertext $\mathrm{CT}^*_{x^*}$ to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker cannot query keys for $y \in \mathcal{Y}$ such that $P(x^*, y) = 1$ or make a decryption query for $\mathrm{CT}^*_{x^*}$.
- **Decision phase:** The attacker outputs a guess β' for β.

The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CCA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CCA}} \leq \mathrm{negl}(\lambda)$.

3.3.2 Ciphertext-policy ABE

In Section 2.3.2, we have given formal definitions of KP-ABE and CP-ABE by giving the formal definition of the more general concept of predicate encryption. A special case of PE on which we focus the most in this thesis is CP-ABE. CP-ABE can be instantiated as PE in the following way. Suppose $\mathcal{U}$ is the universe of attributes, then $\mathcal{X} = \{\mathbb{A} \mid \mathbb{A} \subseteq \mathcal{P}(\mathcal{U})\}$ (where $\mathcal{P}(\mathcal{U})$ denotes the power set of $\mathcal{U}$) is the collection of all possible policies over attributes in $\mathcal{U}$, and $\mathcal{Y} = \{\mathcal{S} \mid \mathcal{S} \subseteq \mathcal{U}\}$ is the collection of all possible sets of attributes. Then, the predicate $P_{\text{CP-ABE}}$ is defined as $P_{\text{CP-ABE}}(\mathbb{A}, \mathcal{S}) = (\mathbb{A} \models \mathcal{S})$. For completeness, we also include the formal definition of CP-ABE, and its associated concepts (largely discussed in Chapter 2).

Definition 3.3: Ciphertext-policy ABE (CP-ABE) [BSW07]

A ciphertext-policy attribute-based encryption (CP-ABE) scheme over a message space $\mathcal{M}$ consists of four algorithms:

- $\mathsf{Setup}(\lambda) \rightarrow (\mathrm{MPK}, \mathrm{MSK})$: On input the security parameter λ, this probabilistic algorithm generates the domain parameters (which includes a description of the universe of attributes $\mathcal{U}$), the master public key MPK and the master secret key MSK.
- $\mathsf{KeyGen}(\mathrm{MSK}, \mathcal{S}) \rightarrow \mathrm{SK}_{\mathcal{S}}$: On input the master secret key MSK and set of attributes $\mathcal{S}$, this probabilistic algorithm generates a secret key $\mathrm{SK}_{\mathcal{S}}$.
- $\mathsf{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \rightarrow \mathrm{CT}_{\mathbb{A}}$: On input the master public key MPK, access policy $\mathbb{A}$ and message M, this probabilistic algorithm generates a ciphertext $\mathrm{CT}_{\mathbb{A}}$.
- $\mathsf{Decrypt}(\mathrm{MPK}, \mathrm{SK}_{\mathcal{S}}, \mathrm{CT}_{\mathbb{A}}) \rightarrow M$: On input the master public key MPK, the secret key $\mathrm{SK}_{\mathcal{S}}$, and the ciphertext $\mathrm{CT}_{\mathbb{A}}$, if $\mathbb{A} \models \mathcal{S}$, then it returns M. Otherwise, it returns an error message $\perp$.

Large-universe ABE. The universe of attributes $\mathcal{U}$ can be small or large [SW05]. If in the Setup, a public key is generated for each attribute, the universe is *small*. Conversely, if the size of the master public key does not depend on the size of the universe, the universe is *large*.

Multi-use ABE. The access policies may be restricted in the number of times that one attribute may occur. If an attribute may only occur once, we call the scheme *one-use*. If it allows unlimited occurrences of one attribute, we call it *multi-use*.

Non-monotone ABE. Three types of non-monotonicity have been formalized (see Section 2.5.7): *OSW-type* [OSW07], *OT-type* [OT12], and *OSWOT-type* [AT20]. In the negations considered by Ostrovsky, Sahai and Waters (OSW) [OSW07], e.g., “NOT profession: doctor”, the entire attribute set associated with the secret key, e.g., “{profession: nurse, department: radiology, department: neurology}”, is compared with the negated attribute to establish that the set does not contain it. In ABE implementations, this translates in a decryption cost that grows in not only the size of the policy, but also in size of the attribute sets. Such negations may thus not be efficient if the sets are large. In the negations considered by Okamoto and Takashima (OT) [OT12], e.g., “profession: NOT doctor”, the attribute labels, e.g., “profession”, play a role. In particular, the set must contain an attribute with label “profession” and its value, e.g., “nurse”, must differ from the negated attribute. While this is more efficient than OSW-type negations, the set of attributes is allowed to contain only one attribute for each label, e.g., such negations are not supported for the label “department” in our first example. Thus, schemes supporting this type of negations are bounded in the number of label re-uses, which is not always desirable. For instance, like in our example, users may have multiple attributes for labels such as “departments at a hospital”, “courses followed at a university” or “mail addresses”. As a solution, Attrapadung and Tomida [AT20] introduced OSWOT-type negations, e.g., “department: NOT cardiology”, to extend OT-type negations, such that the negated attribute is compared with all attributes in the set that share the same label, e.g., “{department: radiology, department: neurology}”. In this way, the flexibility of OSW-type negations and the efficiency of OT-type negations can be combined.

3.3.3 Predicate key encapsulation

Although in the security analysis, it is simpler to consider the encryption variant of a scheme, it is preferred to use key encapsulation in practice. In the key-encapsulation variant of predicate encryption (Definition 2.3), which we call predicate KEM (P-KEM), we replace Encrypt by Encaps and Decrypt by Decaps, where Encaps also outputs a symmetric key, and Decaps outputs a symmetric key instead of a plaintext message. This symmetric key is used to symmetrically encrypt the data.

Definition 3.4: Predicate key-encapsulation mechanism (P-KEM)

A predicate key-encapsulation mechanism for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ consists of four algorithms:

- $\mathsf{Setup}(\lambda, \mathrm{par}) \to (\mathrm{MPK}, \mathrm{MSK})$: On input the security parameter λ and parameters par, this probabilistic algorithm generates the domain parameters, the master public key MPK and the master secret key MSK.
- $\mathsf{KeyGen}(\mathrm{MSK}, y) \to \mathrm{SK}_y$: On input the master secret key MSK and some $y \in \mathcal{Y}$, this probabilistic algorithm generates a secret key SK_y.
- $\mathsf{Encaps}(\mathrm{MPK}, x) \to (\mathrm{K}, \mathrm{CT}_x)$: On input the master public key MPK and some $x \in \mathcal{X}$, this probabilistic algorithm generates an encapsulated symmetric key K and a ciphertext CT_x.
- $\mathsf{Decaps}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x) \to \mathrm{K}$: On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P(x,y) = 1$, then it returns the encapsulated symmetric key K. Otherwise, it returns an error message $\perp$.

Correctness. For all $x \in \mathcal{X}$, and $y \in \mathcal{Y}$ such that $P(x,y) = 1$,

$$\Pr[(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathsf{Setup}(\lambda); (\mathrm{K}, \mathrm{CT}_x) \leftarrow \mathsf{Encaps}(\mathrm{MPK}, x);\\ \mathsf{Decaps}(\mathrm{MPK}, \mathsf{KeyGen}(\mathrm{MSK}, y)), \mathrm{CT}_x) \neq \mathrm{K}] \leq \mathrm{negl}(\lambda).$$

Full security against chosen-plaintext attacks. The full security model for P-KEM is defined similarly as that for PE (Definition 2.4). The crucial difference between the two is that the goal of the attacker is to distinguish a symmetric key produced by the encapsulation algorithm from a randomly generated key.

Definition 3.5: CPA-security for P-KEM

We define the security game IND-CPA(λ) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker.
- **First query phase:** The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first key query phase, we have $P(x^*, y) = 0$, and sends it to the challenger. The challenger first encapsulates a key under x^*, i.e., $(\mathrm{K}^*, \mathrm{CT}_{x^*}) \leftarrow \mathrm{Encaps}(\mathrm{MPK}, x^*)$, and then flips a coin $\beta \in_R \{0, 1\}$. If $\beta = 0$, the key K^* is replaced by a value that is selected uniformly at random from the key space. The challenger then sends the resulting encapsulation key K^* and ciphertext CT_{x^*} to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}$ such that $P(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The attacker's advantage is defined as $\mathsf{Adv}_{\text{P-KEM,IND-CPA}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. A scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\text{P-KEM,IND-CPA}} \leq \mathrm{negl}(\lambda)$.

In the selective security model, the attacker commits to the predicate $x^* \in \mathcal{X}$ before the Setup phase. In the co-selective security model, the attacker commits to all $y \in \mathcal{Y}$ before the Setup phase.

3.3.4 Multi-authority PE

We also give a unified formal definition of multi-authority PE, which we already informally discussed in Section 2.8.

Definition 3.6: Multi-authority predicate encryption (MA-PE)

A multi-authority PE scheme for a predicate $P \colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ over a message space $\mathcal{M}$, for authorities $\mathcal{A}_1, ..., \mathcal{A}_{n_{\text{aut}}}$ with $\mathcal{Y}_{\mathcal{A}_i} \subseteq \mathcal{Y}$ and for all $i \neq j$, $\mathcal{Y}_{\mathcal{A}_i} \cap \mathcal{Y}_{\mathcal{A}_j} = \emptyset$, consists of five algorithms:

- $\mathsf{GlobalSetup}(\lambda) \rightarrow \mathrm{GP}$: On input the security parameter λ, this algorithm generates the global domain parameters GP.
- $\mathsf{AuthoritySetup}(\mathrm{GP}) \rightarrow (\mathcal{A}, \mathrm{MPK}_{\mathcal{A}}, \mathrm{MSK}_{\mathcal{A}})$: On input the global domain parameters, this probabilistic algorithm outputs the authority identifier $\mathcal{A}$, the master public key $\mathrm{MPK}_{\mathcal{A}}$ and the master secret key $\mathrm{MSK}_{\mathcal{A}}$.
- $\mathsf{KeyGen}(\mathcal{A}, \mathrm{MSK}_{\mathcal{A}}, \mathrm{GID}, y_{\mathrm{GID},\mathcal{A}}) \rightarrow \mathrm{SK}_{\mathrm{GID},\mathcal{A},y_{\mathrm{GID},\mathcal{A}}}$: On input the authority identifier $\mathcal{A}$, the corresponding master secret key $\mathrm{MSK}_{\mathcal{A}}$ and some $y_{\mathrm{GID},\mathcal{A}} \in \mathcal{Y}_{\mathcal{A}}$, for the user with global identifier GID, this probabilistic algorithm generates a secret key $\mathrm{SK}_{\mathrm{GID},\mathcal{A},y_{\mathrm{GID},\mathcal{A}}}$.
- $\mathsf{Encrypt}(\{\mathcal{A}_i, \mathrm{MPK}_{\mathcal{A}_i}\}_i, x, M) \rightarrow \mathrm{CT}_x$: On input a set of authority identifiers, the associated master public keys $\mathrm{MPK}_{\mathcal{A}_i}$, some $x \in \mathcal{X}$ and message M, this probabilistic algorithm generates a ciphertext $\mathrm{CT}_{\{\mathcal{A}_i\}_i,x}$.
- $\mathsf{Decrypt}(\{\mathcal{A}, \mathrm{MPK}_{\mathcal{A}}, \mathrm{SK}_{\mathrm{GID},\mathcal{A},y_{\mathrm{GID},\mathcal{A}}}\}, \mathrm{CT}_{\{\mathcal{A}_i\}_i,x}) \rightarrow M$: On input a set of authority identifiers, the associated master public keys $\mathrm{MPK}_{\mathcal{A}}$, secret keys $\{\mathrm{SK}_{\mathrm{GID},\mathcal{A},y_{\mathrm{GID},\mathcal{A}}}\}$ and ciphertext $\mathrm{CT}_{\{\mathcal{A}_i\}_i,x}$, if $P(x,y) = 1$ (where $y = \bigcup_{\mathcal{A}} y_{\mathrm{GID},\mathcal{A}}$), it returns M. Otherwise, it returns error message $\perp$.

Security. The security model is similar to that of regular PE (Definition 2.4). In the Setup phase, both the GlobalSetup and AuthoritySetup are run. Furthermore, a set of authorities $\mathfrak{C} \subseteq [n_{\text{aut}}]$ is corrupted before the Setup phase is run (which we call *static corruption*). In the Setup, the master secret keys corresponding to the corrupt authorities are also shared with the attacker. Then, we define $\mathcal{Y}_{\mathfrak{C}} \subseteq \mathcal{Y}$ to be the collective set of key predicates that the attacker controls, by corrupting the authorities and querying secret keys. In the challenge phase and second query phase, we have the additional restriction that the challenge x^* is such that $P(x^*, y) = 0$ for all $y \in \mathcal{Y}_{\mathfrak{C}}$.

Multi-authority ciphertext-policy ABE. A specific instance of multi-authority PE is multi-authority CP-ABE, which is the multi-authority variant of CP-ABE. In this special subtype of CP-ABE, we add another function $\tilde{\rho}$ to the access policy $\mathbb{A} = (\mathbf{A}, \rho, \tilde{\rho})$, which maps the rows of the matrix to the corresponding authority identifiers, i.e., $\tilde{\rho}\colon [n_1] \rightarrow \{\mathcal{A}_i\}_{i \in [n_{\text{aut}}]}$.

3.3.5 Authenticated symmetric encryption

In practice, it is common to combine an encapsulation scheme—to encapsulate a symmetric key—and an (authenticated) *symmetric encryption* scheme—to encrypt the data. In Chapter 9, we show how this can be done efficiently for the CPA-secure schemes analyzed and proposed in this thesis. To this end, we use authenticated symmetric encryption, for which we give the relevant definitions below.

Definition 3.7: Symmetric encryption (SE)

Let λ be the security parameter. A symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc}, \mathrm{Dec})$, with key $\mathrm{K} \in \mathcal{K}(\lambda)$, where $\mathcal{K}(\lambda)$ is some key space of size 2^λ, is defined by

- $\mathrm{Enc}_\mathrm{K}(M) \to \mathrm{CT}_{\mathrm{sym}}$: On input message $M \in \{0,1\}^*$, encryption returns a ciphertext $\mathrm{CT}_{\mathrm{sym}}$.
- $\mathrm{Dec}_\mathrm{K}(\mathrm{CT}_{\mathrm{sym}}) \to M$: On input ciphertext $\mathrm{CT}_{\mathrm{sym}}$, decryption returns a message M or an error message $\perp$.

The scheme is correct if for all keys $\mathrm{K} \in \mathcal{K}(\lambda)$ and all messages $M \in \{0,1\}^*$, we have $\mathrm{Dec}_\mathrm{K}(\mathrm{Enc}_\mathrm{K}(M)) = M$.

Security of symmetric encryption. For symmetric encryption, we introduce the notions of *ciphertext indistinguishability* and *ciphertext authenticity*. Informally, ciphertext indistinguishability ensures that an attacker cannot distinguish between encryptions of any two messages. More formally, it is defined as follows.

Definition 3.8: Ciphertext indistinguishability of SE

Let λ be a security parameter and let $\mathrm{SE} = (\mathrm{Enc}, \mathrm{Dec})$ be an (authenticated) symmetric encryption scheme. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The challenger first picks a key $\mathrm{K} \in \mathcal{K}(\lambda)$. Then, the attacker specifies two messages M_0, M_1 and gives these to the challenger, who flips a coin $\beta \in_R \{0,1\}$ and returns $\mathrm{CT}_{\mathrm{sym}} \leftarrow \mathrm{Enc}_\mathrm{K}(M_\beta)$ to the attacker. The attacker $\mathcal{A}$ outputs a guess β' for β. Then, $\mathrm{SE} = (\mathrm{Enc}, \mathrm{Dec})$ has indistinguishable ciphertexts if for all polynomial-time attackers $\mathcal{A}$ in the game above holds:

$$\mathsf{Adv}_{\mathrm{SE,CIND}} = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

In this work, we assume that $\mathcal{K}(\lambda)$ is the target group $\mathbb{G}_T$. Because most encryption schemes take a key that is a bit string of λ or 2λ bits as input, we use a secure key derivation function $\mathrm{KDF}\colon \mathcal{K}(\lambda) \to \{0,1\}^\lambda$ (or $\{0,1\}^{2\lambda}$) to map the target group elements to strings [CS03].

We also formally define ciphertext authenticity. Informally, ciphertext authenticity ensures that an attacker cannot generate a new valid ciphertext for some key for which one ciphertext is given.

Definition 3.9: Ciphertext authenticity of authenticated encryption

Let λ be a security parameter and let $\mathrm{SE} = (\mathrm{Enc}, \mathrm{Dec})$ be an (authenticated) symmetric encryption scheme. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The challenger first picks a key $\mathrm{K} \in \mathcal{K}(\lambda)$. Then, the attacker specifies one message M and gives it to the challenger, who returns $\mathrm{CT}_{\mathrm{sym}} \leftarrow \mathrm{Enc}_{\mathrm{K}}(M)$ to the attacker. The attacker outputs a ciphertext $\mathrm{CT}'_{\mathrm{sym}}$. Then, the encryption scheme has ciphertext authenticity if for all such attackers holds that $\mathsf{Adv}_{\mathrm{SE},\mathrm{CAUT}} = \Pr[\mathrm{Dec}_{\mathrm{K}}(\mathrm{CT}'_{\mathrm{sym}}) \neq \perp \wedge \mathrm{CT}'_{\mathrm{sym}} \neq \mathrm{CT}_{\mathrm{sym}}] \leq \mathrm{negl}(\lambda)$.

3.3.6 Hash functions

In this thesis, we use three types of hash functions, which can be used to map any arbitrary string into a fixed output domain. For example, we use hash functions to map arbitrary attribute strings to group elements (see e.g., Section 2.5.5).

Definition 3.10: Collision-resistant hash function [Dam89]

A *collision-resistant hash function* is a map $\mathrm{CR}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ that takes an arbitrary-length bit string as input, and outputs a bit string of 2λ bits. The CR is collision resistant if for all attackers, it holds that the advantage $\mathsf{Adv}_{\mathrm{CR}} = \Pr[x \neq x' \wedge \mathrm{CR}(x) = \mathrm{CR}(x')] \leq \mathrm{negl}(\lambda)$, where $x, x' \in \{0,1\}^*$ denote the output of the attacker.

Definition 3.11: Full-domain hash (FDH) in the group $\mathbb{G}'$

A *full-domain hash* (FDH) is a map $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}'$ that takes as input an arbitrarily-long bit string, and outputs an element in the co-domain, in this case, the group $\mathbb{G}'$. It is secure, if it is indistinguishable from a random oracle, i.e., its outputs are generated negligibly close to uniformly at random over the domain $\mathbb{G}'$.

Definition 3.12: Random-prefix collision-resistant hash function (RPC) [ACIK10]

Let λ be a security parameter, and let $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathcal{G} \to \mathcal{Z}$ be a hash function that takes two inputs, one in $\{0,1\}^\lambda$ and one in $\mathcal{G}$, and maps them to an element in $\mathcal{Z}$. Consider the following game between challenger $\mathcal{C}$ and attacker $\mathcal{A}$. The attacker gives the challenger some $g \in \mathcal{G}$. The challenger then picks $k \in \{0,1\}^\lambda$, and gives k and $\mathrm{RPC}(k, g)$ to the attacker. Then, the RPC is *random-prefix collision resistant* if for all such attackers, it holds that the advantage $\mathsf{Adv}_{\mathrm{RPC}} = \Pr[(k', g') \in \{0,1\}^\lambda \times \mathcal{G} \wedge (k', g') \neq (k, g) \wedge \mathrm{RPC}(k', g') = \mathrm{RPC}(k, g)] \leq \mathrm{negl}(\lambda)$.

In this thesis, we use the concrete instantiation given by Abe et al. [ACIK10]. In particular, their instantiation of the RPC hash is a second-preimage resistant hash that takes as input a 128-bit string k and the element in $\mathcal{G}$. Note that second-preimage resistance is sufficient (and that we do not require the stronger notion of collision resistance, as implied by the terminology), because the definition of an RPC hash requires that it is hard to find a second input to the hash that yields the same output as the fixed input given by the challenger.

3.4 Online/offline ABE and PE

As mentioned in Section 2.10.1, Hohenberger and Waters [HW14] introduced the notion of *online/offline ABE*, which can be used to speed up the online execution time of the key generation and encryption algorithms. We adapt the definitions of Hohenberger and Waters to the predicate encryption paradigm. We also emphasize that the online/offline variants of the key generation and encryption algorithms are merely an extension to the key generation and encryption algorithms of the original PE. They can be used interchangeably, i.e., any authorized key generated with the regular or online/offline key generation algorithm can decrypt any ciphertext generated with the regular or online/offline encryption algorithm. Hence, our definition is also presented as an extension to PE rather than as a separate primitive. To account for the possibility that online/offline extensions may exist for one of the two and not the other, we also present the online/offline versions of the algorithms as optional. Furthermore, we include a final step in the online/offline key generation, to be executed after the keys have been received by the user. In this step, the online/offline secret keys are combined such that secret keys can be generated that are indistinguishable from secret keys in a regular run of the key generation algorithm. Therefore, decryption with keys generated by the online/offline algorithms does not incur any additional costs compared to decryption with keys generated by the regular algorithm.

Definition 3.13: Online/offline predicate encryption

A predicate encryption scheme for a predicate $P \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ over a message space $\mathcal{M}$ with optional online/offline key generation and encryption consists of nine algorithms:

- $\mathrm{Setup}(\lambda) \to (\mathrm{MPK}, \mathrm{MSK})$: On input the security parameter λ, this probabilistic algorithm generates the domain parameters, the master public key MPK and the master secret key MSK.

- Regular.KeyGen(MSK, y) → SK_y: On input the master secret key MSK and some $y \in \mathcal{Y}$, this probabilistic algorithm generates a secret key SK_y.
- Offline.KeyGen(MSK) → ISK: On input the master secret key MSK, this optional probabilistic algorithm generates an intermediate secret key ISK.
- Online.KeyGen(MSK, ISK, y) → $\mathrm{OO.SK}_y$: On input the master secret key MSK, intermediate secret key ISK and some $y \in \mathcal{Y}$, this optional probabilistic algorithm generates an online/offline secret key $\mathrm{OO.SK}_y$.
- FinalStep.KeyGen($\mathrm{OO.SK}_y$) → SK_y: On input an online/offline secret key $\mathrm{OO.SK}_y$ for some $y \in \mathcal{Y}$, this optional probabilistic algorithm generates a (regular) secret key SK_y.
- Regular.Encrypt(MPK, x, M) → CT_x: On input the master public key MPK, some $x \in \mathcal{X}$ and message M, this probabilistic algorithm generates a ciphertext CT_x.
- Offline.Encrypt(MPK) → ICT: On input the master public key MPK, this optional probabilistic algorithm generates an intermediate ciphertext ICT.
- Online.Encrypt(MPK, ICT, x, M) → $\mathrm{OO.CT}_x$: On input the master public key MPK, intermediate ciphertext ICT, some $x \in \mathcal{X}$ and message M, this optional probabilistic algorithm generates an online/offline ciphertext $\mathrm{OO.CT}_x$.
- Decrypt(MPK, (OO.)SK_y, (OO.)CT_x) → M: On input the master public key MPK, the (online/offline) secret key (OO.)SK_y, and the (online/offline) ciphertext (OO.)CT_x, if $P(x, y) = 1$, then it returns M. Otherwise, it returns an error message $\perp$.

3.4.1 Security model

We also adjust the security model in [HW14] to match our definition of online/offline predicate encryption. In particular, the attacker can request regular or online/offline keys (but not intermediate keys) during the query phases, and request either a regular or online/offline ciphertext during the challenge phase.

Definition 3.14: Full CPA-security for online/offline PE

We define the security game IND-CPA-OO(λ) between challenger and attacker as follows:

- **Setup phase:** The challenger runs Setup(λ) to obtain MPK and MSK, and sends the master public key MPK to the attacker. The challenger also initializes an empty list L and a counter $c_L = 0$.

- **First query phase:** The attacker can make the following types of queries (in any order, any number of times, polynomially bounded in λ):
    - **Regular key query:** The attacker queries secret keys for $y \in \mathcal{Y}$, and obtains $\mathrm{SK}_y \leftarrow \mathrm{KeyGen}(\mathrm{MSK}, y)$ in response.
    - **Intermediate secret key query:** The attacker queries intermediate secret keys. The challenger generates $\mathrm{ISK} \leftarrow \mathrm{Offline.KeyGen}(\mathrm{MSK})$, stores (c_L, ISK) in list L and updates the counter $c_L \leftarrow c_L + 1$.
    - **Online/offline secret key query:** The attacker queries online/offline secret keys for $y \in \mathcal{Y}$ and set of indices $\mathcal{I} \subseteq [0, c_L]$ such that none of the indices in $\mathcal{I}$ have been queried before. (Otherwise, there is no entry in the table for one or more indices, and then, the challenger returns an error message $\perp$.) The challenger selects intermediate secret keys (i, ISK_i) for all $i \in \mathcal{I}$ from the list, deletes these entries from the list and generates $\mathrm{OO.SK}_y \leftarrow \mathrm{Online.KeyGen}(\mathrm{MSK}, \{\mathrm{ISK}_i\}_{i\in\mathcal{I}}, y)$.
- **Challenge phase:** The attacker specifies some $x^* \in \mathcal{X}$ such that for all y in the first phase, $P(x^*, y) = 0$ holds, and generates two equal-length messages M_0 and M_1. The attacker sends these to the challenger and chooses whether it wants to be queried on a regular or online/offline ciphertext:
    - **Regular challenge:** The challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$, encrypts M_β under x^*, i.e., $\mathrm{CT}_{x^*} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x^*, M_\beta)$, and sends the resulting ciphertext CT_{x^*} to the attacker.
    - **Online/offline challenge:** The challenger first generates intermediate ciphertexts $\mathrm{ICT} \leftarrow \mathrm{Offline.Encrypt}(\mathrm{MPK})$. Then, the challenger flips a coin, i.e., $\beta \in_R \{0, 1\}$, online encrypts M_β under x^* with ICT, i.e., $\mathrm{OO.CT}_{x^*} \leftarrow \mathrm{Online.Encrypt}(\mathrm{MPK}, \mathrm{ICT}, x^*, M_\beta)$, and sends the resulting ciphertext $\mathrm{OO.CT}_{x^*}$ to the attacker.
- **Second query phase:** This phase is identical to the first query phase, with the additional restriction that the attacker can only query $y \in \mathcal{Y}$ such that $P(x^*, y) = 0$.
- **Decision phase:** The attacker outputs a guess β' for β.

The online/offline predicate encryption scheme is fully secure if all polynomial-time attackers have at most a negligible advantage in this security game, i.e., $\mathsf{Adv}_{\mathrm{PE,IND\text{-}CPA\text{-}OO}} = |\Pr[\beta' = \beta] - \frac{1}{2}| \leq \mathrm{negl}(\lambda)$.

3.5 Complexity assumptions

In Section 2.4.5, we mentioned that the security of many ABE schemes depends on the hardness of non-parametrized and parametrized complexity assumptions, such as the symmetric external Diffie-Hellman or the decisional bilinear Diffie-Hellman assumptions and q-type assumptions.

3.5.1 Non-parametrized assumptions

Definition 3.15: The symmetric external Diffie-Hellman (SXDH) assumption

Let λ be the security parameter. Let $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ be a pairing over three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p and let $g \in \mathbb{G}$ and $h \in \mathbb{H}$ be two generators. The challenger generates $\mathsf{x}, \mathsf{y} \in_{\mathsf{R}} \mathbb{Z}_{\mathsf{p}}$, outputs

$$g', g'^{\mathsf{x}}, g'^{\mathsf{y}}, \text{ where } g' \in \{g, h\}.$$

The challenger flips a coin $\beta \in_R \mathbb{Z}_p$ and outputs $T \in_R \mathbb{G}'$ if $\beta = 0$ and $T = g'^{\mathsf{xy}}$ if $\beta = 1$ (where $\mathbb{G}' = \mathbb{G}$ if $g' = g$ and $\mathbb{G}' = \mathbb{H}$ if $g' = h$). The attacker outputs a guess β' for β. The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{DDH},\mathbb{G}'} = |\Pr[\beta' = 1 \mid \beta = 1] - \Pr[\beta' = 1 \mid \beta = 0]|$. The decisional bilinear Diffie-Hellman assumption holds in $\mathbb{G}'$ if all polynomial-time attackers have at most a negligible advantage, i.e., $\mathsf{Adv}_{\mathrm{DDH},\mathbb{G}'} \leq \mathrm{negl}(\lambda)$. The symmetric external Diffie-Hellman assumption holds if the DDH assumption holds in $\mathbb{G}$ and $\mathbb{H}$, i.e., $\mathsf{Adv}_{\mathrm{SXDH}} \leq \mathsf{Adv}_{\mathrm{DDH},\mathbb{G}} + \mathsf{Adv}_{\mathrm{DDH},\mathbb{H}} \leq \mathrm{negl}(\lambda)$.

Definition 3.16: The decisional bilinear Diffie-Hellman (DBDH) assumption

Let λ be the security parameter. Let $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ be a pairing over three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p, and let $g \in \mathbb{G}, h \in \mathbb{H}$ be two generators. The challenger generates $\mathsf{x}, \mathsf{y}, \mathsf{z} \in_{\mathsf{R}} \mathbb{Z}_{\mathsf{p}}$, outputs

$$g, g'^{\mathsf{x}}, g'^{\mathsf{y}}, g'^{\mathsf{z}}, \text{ where } g' \in \{g, h\}.$$

The challenger also flips a coin $\beta \in_R \mathbb{Z}_p$ and outputs $T \in_R \mathbb{G}_T$ if $\beta = 0$ and $T = e(g, h)^{\mathsf{xyz}}$ if $\beta = 1$. The attacker outputs a guess β' for β. The advantage of the attacker is defined as $\mathsf{Adv}_{\mathrm{DBDH}} = |\Pr[\beta' = 1 \mid \beta = 1] - \Pr[\beta' = 1 \mid \beta = 0]|$. The decisional bilinear Diffie-Hellman assumption holds if all polynomial-time attackers have at most a negligible advantage, i.e., $\mathsf{Adv}_{\mathrm{DBDH}} \leq \mathrm{negl}(\lambda)$.

3.5.2 The uber-assumption family

As mentioned in Section 2.4.5, many q-type assumptions can be captured in the uber-assumption framework [BBG05, Boy08]. Specifically, Boneh, Boyen and Goh prove generic lower bounds on the complexity of such q-type assumptions in the GGM.

Definition 3.17: The uber-assumption family [BBG05, Boy08]

Let $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ be a pairing over three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p, and let $g \in \mathbb{G}, h \in \mathbb{H}$ be two generators. Let $n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c \in \mathbb{N}$ be four positive integers. Suppose that, for all $\mathbb{G}' \in \{\mathbb{G}, \mathbb{H}, \mathbb{G}_T\}$, we have vectors of polynomials $\mathfrak{P}_{\mathbb{G}'} \in \mathbb{Z}_p[X_1, ..., X_{n_c}]^{n_{\mathbb{G}'}}$. Let $\mathfrak{P}_T \in \mathbb{Z}_p[X_1, ..., X_{n_c}]$ be another polynomial. The challenger generates $\mathsf{x}_1, ..., \mathsf{x}_{\mathsf{n}_\mathsf{c}} \in_\mathsf{R} \mathbb{Z}_\mathsf{p}$, and outputs

$$g^{\mathfrak{P}_\mathbb{G}(\mathsf{x}_1,\ldots,\mathsf{x}_{\mathsf{n}_\mathsf{c}})}, h^{\mathfrak{P}_\mathbb{H}(\mathsf{x}_1,\ldots,\mathsf{x}_{\mathsf{n}_\mathsf{c}})}, e(g,h)^{\mathfrak{P}_{\mathbb{G}_T}(\mathsf{x}_1,\ldots,\mathsf{x}_{\mathsf{n}_\mathsf{c}})}.$$

The challenger also flips a coin $\beta \in_R \mathbb{Z}_p$ and outputs $T \in_R \mathbb{G}_T$ if $\beta = 0$ and $T = e(g,h)^{\mathfrak{P}_\mathsf{T}(\mathsf{x}_1,\ldots,\mathsf{x}_{\mathsf{n}_\mathsf{c}})}$ if $\beta = 1$. The attacker outputs a guess β' for β. The attacker's advantage is defined as $\mathsf{Adv}_{(n_\mathbb{G},n_\mathbb{H},n_{\mathbb{G}_T},n_c)\text{-DDH}} = |\Pr[\beta' = 1 \mid \beta = 1] - \Pr[\beta' = 1 \mid \beta = 0]|$. The decisional $(n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c)$-Diffie-Hellman ($(n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c)$-DDH) assumption holds if all polynomial-time attackers have at most a negligible advantage, i.e., $\mathsf{Adv}_{(n_\mathbb{G},n_\mathbb{H},n_{\mathbb{G}_T},n_c)\text{-DDH}} \leq \mathrm{negl}(\lambda)$.

Remark 3.1

For type-I pairings, we set $\mathfrak{P}_\mathbb{G} = \mathfrak{P}_\mathbb{H}$.

Boneh, Boyen and Goh [BBG05] show that, if $\mathfrak{P}_T$ is linearly independent of $\mathfrak{P}_{\mathbb{G}_T}$ and all products of the polynomials (in the entries) in $\mathfrak{P}_\mathbb{G}$ with the polynomials in $\mathfrak{P}_\mathbb{H}$, then the decisional $(n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c)$-Diffie-Hellman ($(n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c)$-DDH) assumption holds with the following time complexity [Boy08, §5.2].

Corollary 3.1: Asymptotic lower bound for uber assumptions [Boy08]

Let p, $\mathfrak{P}_{\mathbb{G}'}$ and $\mathfrak{P}_T$ be as in Definition 3.17. Suppose $\mathfrak{P}_T$ is independent of $\mathfrak{P}_{\mathbb{G}_T}$ and all products of the polynomials in $\mathfrak{P}_\mathbb{G}$ with the polynomials in $\mathfrak{P}_\mathbb{H}$. Let $\deg_{\mathbb{G}'}$ be the maximum degree of the polynomials in $\mathfrak{P}_{\mathbb{G}'}$, let $\deg_T$ be the degree of $\mathfrak{P}_T$, and set $\deg = \max(\{\deg_{\mathbb{G}_T}, \deg_T, \deg_\mathbb{G} + \deg_\mathbb{H}\})$. Then, any attacker $\mathcal{A}$ that can solve the decisional $(n_\mathbb{G}, n_\mathbb{H}, n_{\mathbb{G}_T}, n_c)$-Diffie-Hellman problem in the generic group model must take time at least $\mathcal{O}(\sqrt{p/\deg} - n_c)$.

Part II

The pair encodings framework

Chapter 4

The power of pair encodings

The pair encodings framework is an important result in the simplified design of complex ABE and PE schemes. In particular, it reduces the effort of proving security of a scheme to proving security of the associated pair encoding, which can then be transformed into a provably secure pairing-based encryption scheme with a compiler. Especially the symbolic property, as introduced by Agrawal and Chase (Eurocrypt '17), has proven to be a valuable security notion that is both simple to verify and applies to many schemes. Nevertheless, several practical extensions using full-domain hashes or employing multiple authorities cannot be instantiated with this compiler, and therefore still require complicated proof techniques.

In this chapter, we review the current state of the pair encodings framework, and we present the first compiler for ABE and PE that supports such practical extensions. To this end, we generalize the definitions of pair encodings and the symbolic property. With our compiler, we flexibly instantiate any pair encoding scheme that satisfies this extended notion of the symbolic property in any pairing-friendly groups, and generically prove the resulting scheme to be selectively secure. To illustrate the effectiveness of our new compiler, we give several new multi-authority and hash-based constructions.

4.1 Introduction

In 2014, Attrapadung [Att14a] and Wee [Wee14] introduced frameworks for *pair* and *predicate encodings*, respectively, to simplify the design and analysis of complex predicate encryption schemes. Informally speaking, pair and predicate encoding schemes abstract a pairing-based predicate encryption scheme to "what happens in the exponent of the keys and ciphertexts". The idea behind these frameworks is that the designer only needs to prove information-theoretic or algebraic notions of security for these encodings. Then, via a *generic compiler*, Attrapadung and Wee construct predicate encryption schemes by instantiating the encodings in some carefully-constructed pairing-friendly groups. Subsequently, they generically prove full security, using dual system encryption techniques [Wat09], of the resulting PE from the security of the encoding and the security of the groups.

Since its invention, many works have contributed to the pair encodings[1] framework [AY15, CGW15, Att16, AC16, AC17b, ABS17, Att19, Amb21]. Nowadays, many pairing-based schemes can be captured in this framework, ensuring that these efficiently satisfy a strong notion of security. Not only has the pair encodings framework become a powerful tool in the design of new schemes, it is also possible to generically transform or compose existing schemes [AY15, AC17b, ABS17, Att19, Amb21]. As a result, increasingly complex schemes can be constructed without further complicating the security proofs. For example, revocation mechanisms [ABS17, YAE$^+$17] and range attributes [AHO$^+$16b] can be generically and efficiently supported [Att19].

Arguably the most powerful security notion for pair encodings is the *symbolic property*, which was first introduced as such by Agrawal and Chase [AC17b], but builds on several prior works, e.g., [LW12, Att14a, Att16]. Interestingly, the symbolic property is meant to make security proofs easy to verify. In particular, this effort boils down to performing simple linear algebra. This is a much simpler task than verifying complex security reductions that require significant expertise. From a historical perspective, the symbolic property builds on the ideas behind the more classical proofs, called "program-and-cancel" proofs (Section 2.6.1), which were used to prove selective security in the early days [BB04, SW05]. In the selective-security model, the attacker commits to the predicate that they are going to attack before seeing the public keys, which is unreasonable to assume in practice [CKMS16].

Nevertheless, even though the symbolic property is strongly linked [LW12, Att14a, AC17b] to these classical proofs, it is not clear if the symbolic property can be used to prove selective security generically. Of course, this also raises the question of whether we should care about this particularly low-hanging fruit at all. If we can use the symbolic property to build fully secure schemes, then why would we want to use it to build weaker schemes? Our answer to this question is simple: because the resulting schemes are simpler, more efficient, and we may be able to generically build practical schemes that we cannot build with the current full-security compilers yet [Att14a, Att16, AC17b]. Notably, those compilers do not readily support various practical properties, e.g.,

- the employment of multiple authorities (Section 2.8);
- full-domain hashes, e.g., to achieve large-universeness efficiently (Section 2.5.5);
- or flexible instantiations in the pairing-friendly groups [AGH13, AGOT14] (which heavily influences the scheme's efficiency (Chapter 6)).

[1] Because predicate encodings are a subset of pair encodings [ABS17], we will use the term pair encodings throughout the rest of this thesis.

Fully secure schemes that do satisfy such properties [LW11a, AC17a, TKN20] need to resort to more complicated proof techniques (and on a case-by-case basis), and move us further away from the simplicity of the symbolic property again. Moreover, because of this complexity, many schemes that do have such properties have turned out to be broken [VA21]. This is, by any means, much worse than using a scheme that is "only" selectively secure.

In addition, the broader audience seems to have confidence in selectively secure schemes, and considers these to be practical. In particular, selectively secure schemes are typically at least a factor 2 more efficient than similar schemes in the full-security setting [AC17b, VAH21]. Because their descriptions do not require the use of complex structures such as dual system groups [CW13, CW14a], they are also simpler and more intuitive. By extension, they are easier to prototype and analyze for any given practical setting [dlPVA22]. Presumably, these are reasons why many public cryptographic libraries contain many implementations of selectively secure schemes [AGM$^+$13, Zeu20, dlPVA, FEN], or why half of the schemes considered by the European Telecommunications Standards Institute [ETS18b] are selectively secure. All in all, simplifying the design of selectively secure schemes is valuable.

4.1.1 Importance of pair encodings in this thesis

In Parts II and III of this thesis, we rely on the pair encodings framework. In addition to the reasons already mentioned, we do this for two more reasons. First, pair encodings provide a more compact notation, making it easier to cryptanalyze the schemes. Second, the pair encodings framework implies a common structure of ABE schemes (see Definition 2.6) that simplifies the fair analysis of the efficiency of multiple schemes. More specifically,

- we consider pair encodings to cryptanalyze existing schemes more effectively (Chapter 5);
- we use the common structure of the schemes implied by pair encodings to benchmark and compare the efficiency of existing schemes (Chapter 6);
- we use the symbolic security property [AC17b] to construct new provably secure schemes (Chapters 7 and 8);
- we formalize a new composition of pair encodings to create a new generic transformation for CCA-security (Chapter 9).

4.1.2 Our contribution: a new compiler

In addition to reviewing the current state of the pair encodings framework, we also propose a new generic compiler. This compiler uses the symbolic property to generically prove selective security of the resulting predicate encryption scheme. With this new compiler, we are able to achieve properties that cannot be generically supported with existing full-security compilers (yet), i.e.,

- multi-authority extensions;
- full-domain hashes;
- flexible instantiations in the pairing-friendly groups.

To achieve these properties, we generalize the definitions of pair encodings and the symbolic property, and introduce maps that explicitly address the use of hashes and the instantiations of the encodings in the pairing-friendly groups.

As a result of our compiler, we also obtain new CP-ABE schemes supporting MSPs. In particular, we give new constructions for decentralized large-universe ABE (Section 2.8). We also explain how the new compiler can be used to obtain a single-authority ABE with attribute-wise key generation, which addresses Direction 2.10.

Relation to fully secure schemes in the generic group model. Our compiler also strenghtens the connection between selectively and fully secure schemes. Previously, Ambrona et al. [ABGW17] showed that any scheme that is not trivially broken is provably fully secure in the generic group model (GGM) (Section 2.4.5). The class of encoding schemes that they consider overlaps with that of the Agrawal-Chase compiler [AC17b], which is also covered by our compiler. For this class of schemes, we obtain the following result: the compiled scheme is provably fully secure in the GGM (with some non-trivial security loss), and it is provably selectively secure in the standard model under a q-type assumption (which is a type of assumption that grows stronger as q grows).

4.2 Definition of pair encoding schemes

Throughout the years, the notion of *pair encoding schemes* has been defined and refined [Att14a, Att16, AC16, AC17b]. We provide the most refined definition below. This definition is related to the standard form proposed in Definition 2.6. In particular, pair encoding schemes consider the encoding vectors $\mathbf{b}, \mathbf{k}$ and $\mathbf{c}$, which are associated with the master public key, the secret keys and ciphertexts, respectively. Furthermore, this definition considers the specific structure of the entries in the vector, which are polynomials over certain variables. Concretely, the entries of the key encodings $\mathbf{k}$ and the ciphertext encodings $\mathbf{c}$ are polynomials over common variables $\mathbf{b}$ and some key and ciphertext-specific variables. These key and ciphertext-specific variables may occur either together with some common variable, e.g., rb or sb, or alone, e.g., $\hat{r}$ or $\hat{s}$. We call these non-lone and lone variables, respectively.

Definition 4.1: Pair encoding schemes (PES) [AC17b]

A pair encoding scheme for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and prime number p, with optionally some additional parameters par, is given by four deterministic polynomial-time algorithms as described below.

- Param(par) $\to (n, \mathbf{b})$: On input par, the algorithm outputs $n \in \mathbb{N}$ that specifies the number of common variables, which are denoted as $\mathbf{b} = (b_1, ..., b_n)$.
- EncKey$(y, p, \alpha, \mathbf{b}) \to (m_1, m_2, \mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y))$: On input $p \in \mathbb{N}$ and $y \in \mathcal{Y}$, this algorithm outputs a vector of polynomials $\mathbf{k} = (k_1, ..., k_{m_3})$, with $m_3 \in \mathbb{N}$, defined over non-lone variables $\mathbf{r} = (r_1, ..., r_{m_1})$ and lone variables $\hat{\mathbf{r}} = (\hat{r}_1, ..., \hat{r}_{m_2})$. Specifically, the polynomial k_i is expressed as

  $$k_i = \delta_i \alpha + \sum_{j \in [m_2]} \delta_{i,j} \hat{r}_j + \sum_{j \in [m_1], k \in [n]} \delta_{i,j,k} r_j b_k,$$

  for all $i \in [m_3]$, where $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$.
- EncCt$(x, p, \mathbf{b}) \to (w_1, w_2, \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x))$: On input $p \in \mathbb{N}$ and $x \in \mathcal{X}$, this algorithm outputs a vector of polynomials $\mathbf{c} = (c_1, ..., c_{w_3})$, with $w_3 \in \mathbb{N}$, defined over non-lone variables $\mathbf{s} = (s_0 = s, s_1, ..., s_{w_1})$ and lone variables $\hat{\mathbf{s}} = (\hat{s}_1, ..., \hat{s}_{w_2})$. Specifically, the polynomial c_i is expressed as

  $$c_i = \sum_{j \in [w_2]} \eta_{i,j} \hat{s}_j + \sum_{j \in \overline{[w_1]}, k \in [n]} \eta_{i,j,k} s_j b_k,$$

  for all $i \in [w_3]$, where $\eta_{i,j}, \eta_{i,j,k} \in \mathbb{Z}_p$.
- Pair$(x, y, p) \to (\mathbf{E}, \overline{\mathbf{E}})$: On input $p \in \mathbb{N}$, $x \in \mathcal{X}$, and $y \in \mathcal{Y}$, this algorithm outputs two matrices $\mathbf{E} \in \mathbb{Z}_p^{(w_1+1) \times m_3}$ and $\overline{\mathbf{E}} \in \mathbb{Z}_p^{w_3 \times m_1}$.

A PES is correct, if for every $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $P(x,y) = 1$, it holds that $\mathbf{s}\mathbf{E}\mathbf{k}^\intercal + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^\intercal = \alpha s$.

Notation. In this thesis, we use the same representation for corresponding indices in the vectors and matrices. In this way, we can efficiently identify which entries should be combined during, e.g., the Pair algorithm. For example, consider the index 0 of $s = s_0$ in the non-lone variable vector $\mathbf{s} = (s, s_1, ..., s_{w_1})$, which we thus also use as row index in the matrix $\mathbf{E}$. Additionally, we often use alternative notation to represent the row and column indices of the matrices and vectors. We do this to efficiently match these entries with other parameters. For example, consider the index representations

$(1,j)$ and $(2,j,k)$, where $j \in [n_1]$ and $k \in [n_2]$, which are mapped injectively in the interval $[n_1(n_2+1)]$.

4.2.1 Examples of pair encoding schemes

We give three examples of pair encoding schemes, i.e., those implied by the Waters scheme [Wat11, §3] and the Rouselakis-Waters scheme [RW13, §3], and the Agrawal-Chase pair encoding scheme [AC17b, AC17c, §B.1].

Construction 4.1: The PES for [Wat11]

The PES implied by the Waters [Wat11, §3] scheme is defined as follows:

- Param($\mathcal{U}$): On input the universe $\mathcal{U}$, the algorithm outputs $n = |\mathcal{U}|+1$, and $\mathbf{b} = (b, \{b_{\text{att}}\}_{\text{att}\in\mathcal{U}})$.
- EncKey($\mathcal{S}, p$): On input set of attributes $\mathcal{S}$, this algorithm outputs $\mathbf{k} = (k_0 = \alpha + rb, \{k_{\text{att}} = rb_{\text{att}}\}_{\text{att}\in\mathcal{S}})$ defined over non-lone variables $\mathbf{r} = (r)$ and lone variables $\hat{\mathbf{r}} = \emptyset$. Note that $m_1 = 1$ and $m_2 = 0$.
- EncCt($\mathbb{A}, p$): On input access policy $\mathbb{A} = (\mathbf{A}, \rho)$, this algorithm outputs $\mathbf{c} = (\{c_j = \lambda_j + s_j b_{\rho(j)}\}_{j\in[n_1]})$, where $\lambda_j = A_{j,1}sb + \sum_{k\in[2,n_2]} A_{j,k}v_k$, defined over non-lone variables $\mathbf{s} = (s, \{s_j\}_{j\in[n_1]})$ and lone variables $\hat{\mathbf{s}} = (\{v_k\}_{k\in[2,n_2]})$. Note that $w_1 = n_1$ and $w_2 = n_2 - 1$.
- Pair($\mathbb{A}, \mathcal{S}, p$): On input policy $\mathbb{A}$, and set of attributes $\mathcal{S}$, if $\mathbb{A} \models \mathcal{S}$, then this algorithm determines $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j\in\Upsilon}$ such that $\sum_{j\in\Upsilon} \varepsilon_j \lambda_j = sb$ (Definition 2.5), and outputs two matrices $\mathbf{E} = \mathbf{1}_{0,0}^{(w_1+1)\times m_3} + \sum_{j\in\Upsilon} \varepsilon_j \mathbf{1}_{j,\rho(j)}^{(w_1+1)\times m_3}$ and $\overline{\mathbf{E}} = -\sum_{j\in\Upsilon} \varepsilon_j \mathbf{1}_{j,1}^{w_3\times m_1}$.

Construction 4.2: The PES for [RW13]

The PES implied by the Rouselakis-Waters [RW13, §3] scheme is defined as follows:

- Param($\emptyset$): The algorithm outputs $n = 4$, and $\mathbf{b} = (b, b', b_0, b_1)$.
- EncKey($\mathcal{S}, p$): On input set of attributes $\mathcal{S}$, this algorithm outputs $\mathbf{k} = (k_0 = \alpha + rb, \{k_{\text{att}} = rb' + r_{\text{att}}(b_0 + x_{\text{att}}b_1)\}_{\text{att}\in\mathcal{S}})$, where $x_{\text{att}} \in \mathbb{Z}_p$ is the integer representation of att, defined over non-lone variables $\mathbf{r} = (r, \{r_{\text{att}}\}_{\text{att}\in\mathcal{S}})$ and lone variables $\hat{\mathbf{r}} = \emptyset$. Note that $m_1 = |\mathcal{S}|+1$ and $m_2 = 0$.

- EncCt($\mathbb{A}, p$): On input access policy $\mathbb{A} = (\mathbf{A}, \rho)$, this algorithm outputs $\mathbf{c} = (\{c_{1,j} = \lambda_j + s_j b', c_{2,j} = s_j(b_0 + x_{\rho(j)} b_1)\}_{j \in [n_1]})$, where $\lambda_j = A_{j,1} sb + \sum_{k \in [2,n_2]} A_{j,k} v_k$ and x_{att} is the representation of att in $\mathbb{Z}_p$, defined over non-lone variables $\mathbf{s} = (s, \{s_j\}_{j \in [n_1]})$ and lone variables $\hat{\mathbf{s}} = (\{v_k\}_{k \in [2,n_2]})$. Note that $w_1 = n_1$ and $w_2 = n_2 - 1$.

- Pair($\mathbb{A}, \mathcal{S}, p$): On input policy $\mathbb{A}$, and set of attributes $\mathcal{S}$, if $\mathbb{A} \models \mathcal{S}$, then this algorithm determines $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ such that $\sum_{j \in \Upsilon} \varepsilon_j \lambda_j = sb$ (Definition 2.5), and outputs two matrices $\mathbf{E} = \mathbf{1}^{(w_1+1) \times m_3}_{0,0} + \sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}^{(w_1+1) \times m_3}_{j,\rho(j)}$ and $\overline{\mathbf{E}} = -\sum_{j \in \Upsilon} \varepsilon_j \left(\mathbf{1}^{w_3 \times m_1}_{(1,j),0} + \mathbf{1}^{w_3 \times m_1}_{(2,j),\rho(j)} \right)$.

Construction 4.3: The PES for [AC17b, AC17c]

The PES by the Agrawal and Chase [AC17b, AC17c, §B.1] scheme is defined as follows:

- Param($\mathcal{U}$): On input the universe $\mathcal{U}$, the algorithm outputs $n = |\mathcal{U}| + 1$, and $\mathbf{b} = (b, \{b_{\text{att}}\}_{\text{att} \in \mathcal{U}})$.

- EncKey($\mathcal{S}, p$): On input set of attributes $\mathcal{S}$, this algorithm outputs $\mathbf{k} = (k_0 = \alpha + rb, \{k_{\text{att}} = rb_{\text{att}}\}_{\text{att} \in \mathcal{S}})$ defined over non-lone variables $\mathbf{r} = (r)$ and lone variables $\hat{\mathbf{r}} = \emptyset$. Note that $m_1 = 1$ and $m_2 = 0$.

- EncCt($\mathbb{A}, p$): On input access policy $\mathbb{A} = (\mathbf{A}, \rho)$, this algorithm determines a map $\tau \colon [n_1] \to [m]$ (which ensures that the rows of $\mathbf{A}$ corresponding to the same attribute are assigned to different random integers), where $m = \max_{j \in [n_1]} |\rho^{-1}(\rho(j))|$, such that $\rho(j) = \rho(j')$ implies $\tau(j) \neq \tau(j')$ for all $j, j' \in [n_1]$ with $j \neq j'$, and outputs $\mathbf{c} = (\{c_j = \lambda_j + s_{\tau(j)} b_{\rho(j)}\}_{j \in [n_1]})$, where $\lambda_j = A_{j,1} sb + \sum_{k \in [2,n_2]} A_{j,k} v_k$, defined over non-lone variables $\mathbf{s} = (s, \{s_j\}_{j \in [n_1]})$ and lone variables $\hat{\mathbf{s}} = (\{v_k\}_{k \in [2,n_2]})$. Note that $w_1 = n_1$ and $w_2 = n_2 - 1$.

- Pair($\mathbb{A}, \mathcal{S}, p$): On input policy $\mathbb{A}$, and set of attributes $\mathcal{S}$, if $\mathbb{A} \models \mathcal{S}$, then this algorithm determines $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ such that $\sum_{j \in \Upsilon} \varepsilon_j \lambda_j = sb$ (Definition 2.5), and outputs two matrices $\mathbf{E} = \mathbf{1}^{(w_1+1) \times m_3}_{0,0} + \sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}^{(w_1+1) \times m_3}_{\tau(j),\rho(j)}$ and $\overline{\mathbf{E}} = -\sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}^{w_3 \times m_1}_{j,1}$.

4.3 Security of pair encodings

We consider two notions of security for pair encodings: the *symbolic property*, which is an algebraic notion of security, and the *perfect master-key hiding* property, which is an information-theoretic notion of security.

4.3.1 The symbolic property

The symbolic property is a powerful security notion for pair encoding schemes that is purely algebraic. Roughly, the *selective* and *co-selective symbolic property* are based on the classical security notions of selective and co-selective security for PE (Definition 2.4). Recall that, in these models, the attacker commits to the challenge access policy (resp. set of attributes). This is used in "program-and-cancel" proofs [Wat11, RW13], in which the challenger embeds the policy (resp. set) in the public keys. In the simulation of the secret keys and challenge ciphertext, the components are programmed in a specific way, using that the set does not satisfy the policy (resp. policy is not satisfied by the set). Typically, the components that cannot be programmed are canceled by other non-programmable components. In the pair encodings framework, this "programming" corresponds to "substitution", and the "canceling" corresponds to "evaluating to 0". In particular, the symbolic property considers whether the variables in the encodings can be substituted by certain vectors and matrices, such that evaluating the polynomials with these substitutions yields the all-zero vector.

Definition 4.2: Symbolic security property (Sym-Prop) **[AC17b]**

A pair encoding scheme $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ for a predicate $P \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ satisfies the (d_1, d_2)-selective symbolic property for positive integers d_1 and d_2 if there exist deterministic polynomial-time algorithms EncB, EncS, and EncR such that for all p, par, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $P(x,y) = 0$, we have that

- $\text{EncB}(x) \to \mathbf{B}_1, ..., \mathbf{B}_n \in \mathbb{Z}_p^{d_1 \times d_2}$;
- $\text{EncR}(x,y) \to \mathbf{r}_1, ..., \mathbf{r}_{m_1} \in \mathbb{Z}_p^{d_2}, \mathbf{a}, \hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_{m_2} \in \mathbb{Z}_p^{d_1}$;
- $\text{EncS}(x) \to \mathbf{s}_0, ..., \mathbf{s}_{w_1} \in \mathbb{Z}_p^{d_1}, \hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_{w_2} \in \mathbb{Z}_p^{d_2}$;

such that $\mathbf{s}_0 \cdot \mathbf{a}^\intercal \neq 0$, and if we substitute

$$\hat{s}_{i'} : \hat{\mathbf{s}}_{i'} \qquad s_i b_j : \mathbf{s}_i \mathbf{B}_j \qquad \alpha : \mathbf{a}^\intercal \qquad \hat{r}_{k'} : \hat{\mathbf{r}}_{k'}^\intercal \qquad r_k b_j : \mathbf{B}_j \mathbf{r}_k^\intercal,$$

for $i \in [w_1], i' \in [w_2], j \in [n], k \in [m_1], k' \in [m_2]$ in all the polynomials of $\mathbf{k}$ and $\mathbf{c}$ (output by EncKey and EncCt, respectively), they evaluate to $\mathbf{0}$.

Similarly, a PES satisfies the (d_1, d_2)-co-selective symbolic security property if there exist EncB, EncR, EncS that satisfy the above properties but where EncB and EncR only take y as input, and EncS takes x and y as input.

A scheme satisfies the (d_1, d_2)-symbolic property if it satisfies the (d_1', d_2')-selective and (d_1'', d_2'')-co-selective properties for some $d_1', d_1'' \leq d_1$ and $d_2', d_2'' \leq d_2$.

Agrawal and Chase [AC17b] prove that any PES satisfying the (d_1, d_2)-symbolic property can be transformed in a fully secure predicate encryption scheme.

In [Att19], Attrapadung defines a slightly stronger version of the symbolic property, called $\mathsf{Sym\text{-}Prop}^{+}$, which additionally requires that $\mathbf{a} = \mathbf{1}_1^{d_1}$.

4.3.2 Perfect master-key hiding

In some works [Att14a, Att16], the information-theoretic security notion of perfect master-key hiding is used to achieve security under non-parametrized assumptions such as SXDH (Definition 3.15). As the name suggests, this security notion requires that the master-key α is information-theoretically hidden, i.e., the distributions of the keys and ciphertexts are identical for each choice of the master-key. Intuitively, this ensures that the attacker cannot determine which master-key α was used by observing the key and ciphertext distributions.

Definition 4.3: Perfectly master-key hiding (PMH) [Att16]

A pair encoding scheme Γ = (Param, EncKey, EncCt, Pair) for a predicate family $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is perfectly master-key hiding if, for all p, par, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $P(x,y) = 0$, we have that the following distributions are identical:

$$\{\mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y), \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x)\} \text{ and } \{\mathbf{k}(0, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}, y), \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x)\},$$

where all variables $\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}$ are taken uniformly at random from $\mathbb{Z}_p$.

Attrapadung [Att16] and Agrawal and Chase [AC16] prove that any pair encoding scheme that is perfectly master-key hiding can be converted to a fully secure predicate encryption scheme. The resulting scheme is then secure under a static assumption such as SXDH.

4.3.3 Examples of security proofs

Perfect master-key hiding. We give a proof of perfect master-key hiding for Wat11 (Construction 4.1), based on the proofs in [Att14a, Att14b, §9.1]. We also show that the PES for RW13 is not perfectly master-key hiding, which is based on the proofs in [Att14a, Att14b, §7.1]

Lemma 4.1

The PES in Construction 4.1 is perfectly master-key hiding.

Proof. We show that α is information-theoretically hidden by rb. Suppose $\mathbb{A} \not\models \mathcal{S}$. Then, by Definition 2.5, we can find $\mathbf{w} = (w_1, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ with $w_1 = 1$, such that for all $j \in \Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$, it holds that $\sum_{k\in[n_2]} A_{j,k}\mathbf{w}_k = 0$. Hence, for all $j \in \Upsilon$, we have

$$\sum_{k\in[n_2]} A_{j,k}v_k = \sum_{k\in[n_2]} A_{j,k}(v_k + zw_k),$$

where $v_1 = sb$ and $z \in_R \mathbb{Z}_p$. Because we cannot distinguish the left-hand side from the right, sb is information-theoretically hidden. For all $j \in [n_1] \setminus \Upsilon$, we have that $b_{\rho(j)}$ only occurs in the ciphertext encoding, and thus, $s_j b_{\rho(j)}$ information-theoretically hides λ_j. Because sb is hidden and s is not, we know that b is hidden. Therefore, $\alpha + rb$ hides α. □

Lemma 4.2

The PES in Construction 4.2 is not perfectly master-key hiding.

Proof. We show that α depends on the distribution of the key and ciphertext encodings. Let $\mathbb{A}$ and $\mathcal{S}$ be such that $\mathbb{A} \not\models \mathcal{S}$ and $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\} \neq \emptyset$. Then, from $s_j(b_0 + x_{\rho(j)}b_1)$ and s_j, we obtain $b_0 + x_{\rho(j)}b_1$. By extension, we obtain rb' from k_{att} and r_{att}, and from rb' and r, we obtain b'. We use b' and s_j to get λ_j, and as a result, we also get sb. From s and sb, we acquire b, and from b, r and $\alpha + rb$, we obtain α. Hence, the distributions are dependent of α, and thus, the PES is not perfectly master-key hiding. □

Symbolic property. We give symbolic property proofs for all PESs in Constructions 4.1 ,4.2 and 4.3, based on the proofs in [AC17b, AC17c, §B]. Note, however, that we have improved the co-selective proof for the [AC17b] PES (Construction 4.3).

Lemma 4.3

The PESs in Constructions 4.1 and 4.3 satisfy Sym-Prop.

Proof. For the proof of the **selective symbolic property**, we define τ as in Construction 4.3 for Construction 4.1 as well, set $d_1 = m$, $d_2 = n_2$, and

- EncB($\mathbb{A}$): We set $\mathbf{B} = \mathbf{1}_{1,1}^{d_1\times d_2}$, and $\left\{\mathbf{B}_{\text{att}} = -\sum_{j\in\rho^{-1}(\text{att}),k\in[n_2]} A_{j,k}\mathbf{1}_{\tau(j),k}^{d_1\times d_2}\right\}_{\text{att}\in\mathcal{U}}$.

- EncR($\mathbb{A}, \mathcal{S}$): Let $\mathbf{w} = (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ be the vector orthogonal to all $j \in \Upsilon$ (Definition 2.5). We set $\mathbf{a} = \mathbf{1}_1^{d_1}$, and $\mathbf{r} = -\sum_{k \in [n_2]} w_k \overline{\mathbf{1}}_k^{d_2}$.
- EncS($\mathbb{A}$): We set $\mathbf{s} = \mathbf{1}_1^{d_1}, \{\mathbf{v}_k = \overline{\mathbf{1}}_k^{d_2}\}_{k \in [n_2]}$. Further, we set $\{\mathbf{s}_j = \mathbf{1}_{\tau(j)}^{d_1}\}_{j \in [n_1]}$ (for Construction 4.1) and $\{\mathbf{s}_l = \mathbf{1}_l^{d_1}\}_{l \in [m]}$ (for Construction 4.3).

For the **co-selective symbolic property** proof, we set $d_1 = 1$ and $d_2 = 1$, and

- EncB($\mathcal{S}$): We set $\mathbf{B} = 1$, $\{\mathbf{B}_{\text{att}} = 0\}_{\text{att} \in \mathcal{S}}$ and $\{\mathbf{B}_{\text{att}} = -1\}_{\text{att} \in \mathcal{U} \setminus \mathcal{S}}$.
- EncR($\mathcal{S}$): We set $\mathbf{a} = 1$, and $\mathbf{r} = -1$.
- EncS($\mathbb{A}, \mathcal{S}$): Let $\mathbf{w}$ be the vector orthogonal to all $j \in \Upsilon$ (Definition 2.5). We set $\mathbf{s} = w_1$ and $\{\mathbf{v}_k = w_k\}_{k \in [2, n_2]}$, and set $\left\{\mathbf{s}_j = \sum_{k \in [n_2]} A_{j,k} w_k\right\}_{j \in [n_1]}$ (for Construction 4.1) and $\left\{\mathbf{s}_l = \sum_{j \in \tau^{-1}(l), k \in [n_2]} A_{j,k} w_k\right\}_{l \in [m]}$ (for Construction 4.3).

Hence, the two PESs satisfy Sym-Prop. □

To illustrate the simplicity of verifying a symbolic property proof, we verify the proof of the selective property for Construction 4.1. To this end, we have to verify that $\mathbf{a} \cdot \mathbf{s}^\intercal = \mathbf{1}_1^{d_1} \cdot (\mathbf{1}_1^{d_1})^\intercal \neq 0$ and that the polynomials k_0, k_{att} and c_j evaluate to $\mathbf{0}$ when the variables are replaced by the vectors and matrices. For k_0, we have:

$$k_0 = \alpha + rb = \mathbf{1}_1^{d_1} - \sum_{k \in [n_2]} w_k \mathbf{1}_{1,1}^{d_1 \times d_2} \overline{\mathbf{1}}_k^{d_2} = \mathbf{1}_1^{d_1} - \mathbf{1}_1^{d_1} = \mathbf{0}^{d_1}.$$

Note that writing each matrix as a sum of matrices in which only one entry is non-zero simplifies the computations. For example, $\sum_{k \in [n_2]} w_k \mathbf{1}_{1,1}^{d_1 \times d_2} \overline{\mathbf{1}}_k^{d_2}$ is computed easily by first observing that $w_k \mathbf{1}_{1,1}^{d_1 \times d_2} \overline{\mathbf{1}}_k^{d_2} = \mathbf{0}^{d_1}$ for all $k \neq 1$. Similarly, we verify that

$$\begin{aligned} k_{\text{att}} = rb_{\text{att}} &= -\sum_{k \in [n_2]} w_k \overline{\mathbf{1}}_k^{d_2} \left(-\sum_{j \in \rho^{-1}(\text{att}), k' \in [n_2]} A_{j,k'} \mathbf{1}_{\tau(j),k'}^{d_1 \times d_2} \right) \\ &= \sum_{j \in \rho^{-1}(\text{att}), k \in [n_2]} A_{j,k} w_k \mathbf{1}_{\tau(j)}^{d_1} = \sum_{j \in \rho^{-1}(\text{att})} \mathbf{A}_j \mathbf{w}^\intercal \mathbf{1}_{\tau(j)}^{d_1} = \mathbf{0}^{d_1}, \end{aligned}$$

and for c_j, we have

$$c_j = \lambda_j + s_j b_{\rho(j)} = A_{j,1} sb + \sum_{k \in [2, n_2]} A_{j,k} v_k + s_j b_{\rho(j)}$$

$$= A_{j,1} \mathbf{1}_1^{d_1} \mathbf{1}_{1,1}^{d_1 \times d_2} + \sum_{k \in [2, n_2]} A_{j,k} \overline{\mathbf{1}}_k^{d_2} + \mathbf{1}_{\tau(j)}^{d_1} \left(-\sum_{j' \in \rho^{-1}(\rho(j)), k' \in [n_2]} A_{j',k'} \mathbf{1}_{\tau(j'),k'}^{d_1 \times d_2} \right)$$

$$= \sum_{k\in[n_2]} A_{j,k}\overline{\mathbf{1}}^{d_2}_k - \sum_{k\in[n_2]} A_{j,k}\overline{\mathbf{1}}^{d_2}_k = \mathbf{0}^{d_2}.$$

Lemma 4.4

The PES in Construction 4.2 satisfies Sym-Prop.

Proof. For the proof of the **selective symbolic property**, we set $d_1 = n_1 + 1$, $d_2 = (n_1+1)n_2$. For simplicity of notation, we write the indices in $[d_2]$ as a tuple $(1,k)$ or $(2,j,k)$ (with $j \in [n_1], k \in [n_2]$) such that it represents a unique integer in $[d_2]$. For the indices in $[d_1]$, we start counting at 0.

- EncB($\mathbb{A}$): We set

$$\mathbf{B} = \mathbf{1}^{d_1\times d_2}_{0,(1,1)}, \qquad \mathbf{B}' = \sum_{j\in[n_1],k\in[n_2]} A_{j,k}\mathbf{1}^{d_1\times d_2}_{j,(1,k)},$$

$$\mathbf{B}_1 = \sum_{j\in[n_1],k\in[n_2]} A_{j,k}\mathbf{1}^{d_1\times d_2}_{j,(2,j,k)}, \qquad \mathbf{B}_0 = -\sum_{j\in[n_1],k\in[n_2]} A_{j,k}x_{\rho(j)}\mathbf{1}^{d_1\times d_2}_{j,(2,j,k)}.$$

- EncR($\mathbb{A},\mathcal{S}$): Let $\mathbf{w}$ be the vector orthogonal to all $\mathbf{A}_j$ with $j \in \Upsilon$ (Definition 2.5), and let $\overline{\Upsilon} = [n_1]\setminus\Upsilon$. We set

$$\mathbf{r} = \sum_{k\in[n_2]} w_k\overline{\mathbf{1}}^{d_2}_{(1,k)}, \qquad \mathbf{r}_{\text{att}} = \sum_{j\in\overline{\Upsilon},k\in[n_2]} \frac{w_k\overline{\mathbf{1}}^{d_2}_{(2,j,k)}}{x_{\rho(j)} - x_{\text{att}}}, \qquad \mathbf{a} = w_1\mathbf{1}^{d_1}_0$$

- EncS($\mathbb{A}$): We set $\mathbf{s} = \mathbf{1}^{d_1}_0, \quad \left\{\mathbf{s}_j = -\mathbf{1}^{d_1}_j\right\}_{j\in[n_1]}, \quad \mathbf{v}_k = \overline{\mathbf{1}}^{d_2}_{(1,k)}.$

For the proof of the **co-selective symbolic property**, we set $d_1 = d_2 = |\mathcal{S}|+1$, and

- EncB($\mathcal{S}$): We set

$$\mathbf{B} = \mathbf{1}^{d_1\times d_2}_{0,0}, \qquad \mathbf{B}' = \mathbf{1}^{d_1\times d_2}_{0,0},$$

$$\mathbf{B}_1 = -\sum_{\text{att}\in\mathcal{S}} \mathbf{1}^{d_1\times d_2}_{\text{att},\text{att}}, \qquad \mathbf{B}_0 = \sum_{\text{att}\in\mathcal{S}} x_{\text{att}}\mathbf{1}^{d_1\times d_2}_{\text{att},\text{att}} - \mathbf{1}^{d_1\times d_2}_{0,\text{att}}.$$

- EncR($\mathcal{S}$): We set $\mathbf{r} = \overline{\mathbf{1}}^{d_2}_0, \quad \mathbf{r}_{\text{att}} = \overline{\mathbf{1}}^{d_2}_{\text{att}}, \quad \mathbf{a} = \mathbf{1}^{d_1}_0.$

- EncS($\mathbb{A}, \mathcal{S}$): Let $\mathbf{w}$ (with $w_1 = 1$) be the vector orthogonal to all $\mathbf{A}_j$ with $j \in \Upsilon$ (Definition 2.5). We set

$$\mathbf{s} = \mathbf{1}_0^{d_1}, \quad \left\{ \mathbf{s}_j = -\mathbf{A}_j \mathbf{w}^\intercal \left(\mathbf{1}_0^{d_1} + \sum_{\mathrm{att} \in \mathcal{S}} \frac{\mathbf{1}_{\mathrm{att}}^{d_1}}{x_{\mathrm{att}} - x_{\rho(j)}} \right) \right\}_{j \in [n_1]}, \quad \mathbf{v}_k = w_k \overline{\mathbf{1}}_0^{d_2}.$$

Note that, for $\rho(j) \in \mathcal{S}$, we have that $\mathbf{s}_j = \mathbf{0}^{d_1}$, because $\mathbf{A}_j \mathbf{w}^\intercal = 0$.

Similarly as in [AC17c], it follows that all polynomials evaluate to $\mathbf{0}$. Hence, the PES satisfies Sym-Prop. □

4.4 The AC17 generic compiler in a nutshell

Arguably the most powerful generic compiler for full security is the compiler given by Agrawal and Chase [AC17b], which transforms any symbolically secure PES into a fully secure PE. Agrawal and Chase also prove that any scheme that cannot be trivially broken is symbolically secure, meaning that any scheme that can be abstracted to a PES conform Definition 4.1 that is e.g., selectively secure also implies a fully secure scheme in the AC17 framework. Furthermore, in an earlier work [AC16], Agrawal and Chase show that any scheme that is perfectly master-key hiding results in a fully secure scheme with the same compiler, under a more standard, non-parametrized assumption such as the SXDH assumption (Definition 3.15).

The generic compiler by Agrawal and Chase [AC17b] additionally takes dual system groups as input. Dual system groups (DSG) were first introduced by Chen and Wee [CW13, CW14a], and later improved on by Chen, Gay and Wee [CGW15] and Agrawal and Chase [AC16]. DSGs are pairing-based groups that systematize and generalize the dual system encryption techniques (Section 2.6.2) that are often used to prove full security of ABE. As a result, these groups satisfy certain correctness and security properties that can be modularly used in such proofs. They considerably simplify the design of fully secure schemes in prime-order groups.

The most efficient instantiation of the dual system groups in the AC17 compiler relies on the SXDH assumption [CW14a]. The schemes instantiated under these DSGs are roughly a factor 2 more costly than their selectively secure counterparts. For example, the common variables $\mathbf{b} = (b_1, ..., b_n)$ are mapped to group elements $[\mathbf{b}_1]_{\mathbb{G}}, ..., [\mathbf{b}_n]_{\mathbb{G}}$, where $\mathbf{b}_i$ are pairs. For the other variables, the maps are more intricate. To illustrate the complexity of such constructions compared to their selectively secure counterparts, we give an example. In particular, we give the instantiation of RW13, for which we gave a PES in Construction 4.2, in these groups below. We refer to [CW14a] and [AC17b] for a clear description of the groups and how they are instantiated in the AC17 compiler.

Construction 4.4: A fully secure variant of RW13 (RWAC)

The ciphertext-policy attribute-based encryption scheme by Rouselakis and Waters [RW13] is defined in the Agrawal-Chase framework, using the prime-order dual system groups for SXDH in [CW14a], as follows.

- Setup(λ): Taking as input the security parameter λ, the setup generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. The setup also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$. It then generates random integers $\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b'_i \in_R \mathbb{Z}_p$ for all $i \in \{1,2,3\}$, such that $d_1 d_4 \neq d_2 d_3$. It outputs $\mathrm{MSK} = (\alpha_1, \alpha_2, d_1, d_2, d_3, d_4, d_5, b_i, b_{0,i}, b_{1,i}, b'_i)$ as its master secret key and publishes the master public key as

$$\begin{gathered}\mathrm{MPK} = (g, h, A = e(g,h)^{\alpha_1 d_1 + \alpha_2 d_2}, \{g_i = g^{d_i}, B_i = g^{b_1 d_i + b_3 d_{i+2}}, \\ \{B_{l,i} = g^{b_{l,1} d_i + b_{l,3} d_{i+2}}\}_{l \in \{0,1\}}, B'_i = g^{b'_1 d_i + b'_3 d_{i+2}}\}_{i \in \{1,2\}}).\end{gathered}$$

- KeyGen(MSK, $\mathcal{S}$): On input a set of attributes $\mathcal{S}$, the algorithm generates random integers $r, r_{\mathrm{att}} \in_R \mathbb{Z}_p$ for each $\mathrm{att} \in \mathcal{S}$, letting $x_{\mathrm{att}} \in \mathbb{Z}_p$ denote the representation of att in $\mathbb{Z}_p$ and computes the secret key as

$$\begin{gathered}\mathrm{SK}_{\mathcal{S}} = (\{K_i = h^{\alpha_i - r\bar{b}_i}, K'_1 = h^{r d_4 d_6}, K'_2 = h^{-r d_3 d_6}, \\ K_{1,\mathrm{att},i} = h^{-r_{\mathrm{att}}(\bar{b}_{1,i} x_{\mathrm{att}} + \bar{b}_{0,i}) - r\bar{b}'_i}, \\ K_{2,\mathrm{att},1} = h^{r_{\mathrm{att}} d_4 d_6}, K_{2,\mathrm{att},2} = h^{-r_{\mathrm{att}} d_3 d_6}\}_{i \in \{1,2\}, \mathrm{att} \in \mathcal{S}}),\end{gathered}$$

  where for $l \in \{0,1\}$, we have $d_6 = \frac{d_5}{d_1 d_4 - d_2 d_3}$ and

$$\begin{aligned}\bar{b}_1 &= d_6(b_1 d_4 - b_2 d_2), & \bar{b}_2 &= d_6(-b_1 d_3 + b_2 d_1), \\ \bar{b}_{l,1} &= d_6(b_{l,1} d_4 - b_{l,2} d_2), & \bar{b}_{l,2} &= d_6(-b_{l,1} d_3 + b_{l,2} d_1), \\ \bar{b}'_1 &= d_6(b'_1 d_4 - b'_2 d_2), & \bar{b}'_2 &= d_6(-b'_1 d_3 + b'_2 d_1).\end{aligned}$$

- Encrypt(MPK, $\mathbb{A}$, M): A message $M \in \mathbb{G}_T$ is encrypted under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho\colon [n_1] \to \mathcal{U}$ by generating random integers s,s_i,$v_j \in_R \mathbb{Z}_p$ for all $i \in [n_1]$ and $j \in [2, n_2]$, and computes the ciphertext as

$$\begin{gathered}\mathrm{CT}_{\mathbb{A}} = (C = M \cdot A^s, \{C'_i = g_i^s, C_{1,i,j} = B_i^{A_{j,1} s} g_i^{\lambda_j} (B'_i)^{s_j}, \\ C_{2,i,j} = \left(B_{1,i}^{x_{\rho(j)}} B_{0,i}\right)^{s_j}, C_{3,i,j} = g_i^{s_j}\}_{i \in \{1,2\}, j \in [1,n_1]}),\end{gathered}$$

  such that λ_j denotes the j-th entry of $\mathbf{A} \cdot (0, v_2, ..., v_{n_2})^\intercal$.

- Decrypt($\mathrm{SK}_{\mathcal{S}}, \mathrm{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and suppose $\Upsilon = \{j \in [1, n_1] \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$. Then the plaintext M is retrieved by computing

$$C / \prod_{i \in \{1,2\}} \left(e(C'_i, K_i) \cdot e(\prod_{j \in \Upsilon} C_{1,i,j}^{\varepsilon_j}, K'_i) \prod_{j \in \Upsilon} \left(e(C_{2,i,j}^{\varepsilon_j}, K_{2,\rho(j),i}) \cdot e(C_{3,i,j}^{\varepsilon_j}, K_{1,\rho(j),i}) \right) \right).$$

4.4.1 Efficiency consideration: type conversion

The performance penalty of a factor 2 holds only for the case in which we instantiate all key components in $\mathbb{H}$ and all ciphertext components in $\mathbb{G}$ (or vice versa). However, as we will explain in more detail in Chapter 6, it might be desirable to use a different distribution of the key and ciphertext components over the two groups. Furthermore, using FDHs to achieve large-universeness (Section 2.5.5) inevitably requires a different distribution of key and ciphertext components over the two source groups (see, e.g., [AC17a, TKN20]). In these cases, we cannot use the most efficient instantiation for dual-system groups from SXDH anymore, but rather require an instantiation that incurs a performance penalty of a factor 3 compared to the selectively secure counterparts of the schemes.

4.5 Towards a new compiler: generalizing PES

We will also propose a simpler generic compiler, which we will use throughout this thesis. To this end, we first generalize the definitions of pair encodings and the symbolic property, such that they can also capture multi-authority schemes. Furthermore, we introduce functions that capture whether the encoding variables are produced as a result of a full-domain hash or not, and in which groups they are going to be instantiated in the compiler. Before this, we will first explain how the symbolic property and selective security proofs are related.

4.5.1 How the symbolic property and selective security relate

As mentioned, the selective symbolic property and selective security are strongly related in their approaches. More specifically, the evaluation of the polynomials k_i and c_i to 0 after substituting the variables by the vectors and matrices is closely related to the “canceling” part of the “program-and-cancel” strategy used in selective-security proofs. The “programming” part of this proof strategy is related to the complexity assumption that is used in the reduction. Concretely, various input parameters to this complexity assumption are used to program the key and ciphertext components associated with the common and non-lone variables. They are programmed in such a way that the $e(g, h)^{\alpha s}$ part of the scheme can be programmed by the “testing value” of the complexity assumption. For example, consider the keys and ciphertexts of the

Boneh-Boyen [BB04] scheme:

$$\mathrm{SK} = (h^{\alpha + r(b_0 + yb_1)}, h^r), \qquad \mathrm{CT} = (M \cdot e(g,h)^{\alpha s}, g^{s(b_0 + xb_1)}, g^s),$$

where x and y are identities, for which the associated PES is

$$\mathbf{k}(\alpha, r, (b_0, b_1)) = \alpha + r(b_0 + yb_1), \qquad \mathbf{c}(s, (b_0, b_1)) = s(b_0 + xb_1).$$

It satisfies the selective symbolic property, because for $x \neq y$, we can set

$$\mathbf{a} = 1, \qquad \mathbf{r} = \frac{1}{x - y}, \qquad \mathbf{b}_0 = -x, \qquad \mathbf{b}_1 = 1, \qquad \mathbf{s} = 1.$$

Analogously, in the selective security proof, we can make a reduction to the decisional bilinear Diffie-Hellman (DBDH) assumption, i.e., given $g^{\mathsf{x}}, h^{\mathsf{x}}, g^{\mathsf{y}}, h^{\mathsf{y}}, g^{\mathsf{x}}, h^{\mathsf{z}}$, determine whether some testing value T is equal to $e(g,h)^{\mathsf{xyz}}$ or not. We can program the master public key, and the secret key and ciphertext components associated with the non-lone variables in a similar way as in the symbolic property as follows:

$$e(g,h)^{\alpha} = e(g,h)^{\bar{\alpha}} \cdot e(g,h)^{\mathbf{a}\mathsf{xz}}, \qquad g^{b_0} = g^{\bar{b}_0} \cdot g^{\mathbf{b}_0\mathsf{z}}, \qquad g^{b_1} = g^{\bar{b}_1} \cdot g^{\mathbf{b}_1\mathsf{z}},$$
$$h^r = h^{\bar{r}} \cdot h^{\mathbf{r}\mathsf{x}}, \qquad g^s = g^{\bar{s}} \cdot g^{\mathbf{s}\mathsf{y}}.$$

Then, the secret key and ciphertext components associated with the polynomials can be programmed by using the inputs to the DBDH assumption and using that the polynomials evaluate to 0 for those inputs that are not part of the assumption. For example, the key component is simulated as follows:

$$h^{\alpha + r(b_0 + yb_1)} = h^{\bar{\alpha} + \mathbf{a}\mathsf{xz} + (\bar{r} + \mathbf{r}\mathsf{x})(\bar{b}_0 + \mathbf{b}_0\mathsf{z} + y(\bar{b}_1 + \mathbf{b}_1\mathsf{z}))}$$
$$= \underbrace{h^{\bar{\alpha} + \bar{r}(\bar{b}_0 + \mathbf{b}_0\mathsf{z} + y(\bar{b}_1 + \mathbf{b}_1\mathsf{z})) + \mathbf{r}\mathsf{x}(\bar{b}_0 + y\bar{b}_1)}}_{\Delta_1} \cdot h^{\mathbf{a}\mathsf{xz} + \mathbf{r}\mathsf{x}(\mathbf{b}_0\mathsf{z} + y\mathbf{b}_1\mathsf{z})} = \Delta_1 \cdot \underbrace{h^{(\mathbf{a} + \mathbf{r}(\mathbf{b}_0 + y\mathbf{b}_1))\mathsf{xz}}}_{=1},$$

such that Δ_1 can be programmed from $\bar{\alpha}, \bar{r}, \bar{b}_0, \bar{b}_1$ and the inputs to the DBDH assumption, and the remainder associated with h^{xz} (which cannot be part of the assumption) cancels because the polynomial $\alpha + r(b_0 + yb_1)$ evaluates to 0 when α, r, b_0, b_1 are substituted by $\mathbf{a}, \mathbf{r}, \mathbf{b}_0, \mathbf{b}_1$. Lastly, the blinding value is set to $e(g,h)^{\alpha s} = T \cdot e(g,h)^{\bar{\alpha} s} \cdot e(g,h)^{\alpha \bar{s}} \cdot e(g,h)^{\bar{\alpha}\bar{s}}$.

For our compiler, we generalize this approach. Roughly, we associate the public-key variables with (parallel instances of) z, all lone key variables with (parallel instances of) xz, and all non-lone key variables with (parallel instances of) x, so that the key polynomials are associated with (parallel instances of) xz. Similarly, we associate the lone ciphertext variables with (parallel instances of) yz and the non-lone ciphertext variables with (parallel instances of) y, so that the ciphertext polynomials are associated with (parallel instances of) yz. Finally, the blinding value should be associated with xyz, so in the case that this is αs (as in the definition of PES), we require that α and s only use xz and y (and no parallel instances) of the inputs to the complexity assumption. Note that these parallel instances are related to the choices of d_1 and d_2, e.g., we require d_1 parallel instances of y to embed each entry of the substitution vector for a non-lone ciphertext variable. In Section 4.6, we give an assumption with such parallel instances that holds in the generic group model.

4.5.2 Generalizing the definition of pair encoding schemes

In order to cover a larger class of schemes, we also give a more general definition of pair encoding schemes. Notably, decentralized schemes such as [LW11a, RW15] cannot be covered by Definition 4.1. Consequently, we cannot benefit from the generic security as well as the generic conversion techniques that the pair encodings framework provides. Regardless, the proof techniques in [RW15] are strikingly similar to the proof techniques in works in the single-authority setting [Wat11, RW13]. We use this observation to define our more general definitions of pair encoding schemes and the symbolic property. Concretely, for the definition of pair encodings, we extend the master-key α to multiple instances α_i. We also explicitly include ciphertext polynomials that will be instantiated in the target group, and write the blinding value used to mask M in the scheme as a polynomial instead of fixing it to be αs.

Definition 4.4: Generalized pair encoding schemes (GPES)

A generalized pair encoding scheme for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and prime number p, with optionally some additional parameters par, is given by four deterministic polynomial-time algorithms as described below.

- Param(par) $\to (n_\alpha, n_b, \boldsymbol{\alpha}, \mathbf{b})$: On input par, the algorithm outputs $n_\alpha, n_b \in \mathbb{N}$ that specify the number of master key variables and common variables, respectively, which are denoted as $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{n_\alpha})$ and $\mathbf{b} = (b_1, ..., b_{n_b})$, respectively.
- EncKey$(y, p, \boldsymbol{\alpha}, \mathbf{b}) \to (m_1, m_2, \mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}, \mathbf{b}, y))$: On input $p \in \mathbb{N}$, $y \in \mathcal{Y}$, $\boldsymbol{\alpha}$ and $\mathbf{b}$, this algorithm outputs a vector of polynomials $\mathbf{k} = (k_1, ..., k_{m_3})$, with $m_3 \in \mathbb{N}$, defined over non-lone variables $\mathbf{r} = (r_1, ..., r_{m_1})$ and lone variables $\hat{\mathbf{r}} = (\hat{r}_1, ..., \hat{r}_{m_2})$. Specifically, the polynomial k_i is expressed as

$$k_i = \sum_{j \in [n_\alpha]} \delta_{i,j} \alpha_j + \sum_{j \in [m_2]} \hat{\delta}_{i,j} \hat{r}_j + \sum_{j \in [m_1], k \in [n_b]} \delta_{i,j,k} r_j b_k,$$

  for all $i \in [m_3]$, where $\delta_{i,j}, \hat{\delta}_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$.
- EncCt$(x, p, \boldsymbol{\alpha}, \mathbf{b}) \to (w_1, w_2, w_2', c_M, \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x), \mathbf{c}'(\mathbf{s}, \tilde{\mathbf{s}}, \boldsymbol{\alpha}, x))$: On input $p \in \mathbb{N}$, $x \in \mathcal{X}$, $\boldsymbol{\alpha}$ and $\mathbf{b}$, this algorithm outputs a blinding variable c_M

and two vectors of polynomials $\mathbf{c} = (c_1, ..., c_{w_3})$ and $\mathbf{c}' = (c'_1, ..., c'_{w_4})$, with $w_3, w_4 \in \mathbb{N}$, defined over non-lone variables $\mathbf{s} = (s_1, s_2, ..., s_{w_1})$, lone variables $\hat{\mathbf{s}} = (\hat{s}_1, ..., \hat{s}_{w_2})$ and special lone variables $\tilde{\mathbf{s}} = (\tilde{s}_1, ..., \tilde{s}_{w'_2})$. Specifically, the polynomial c_i is expressed as

$$c_i = \sum_{j \in [w_2]} \eta_{i,j} \hat{s}_j + \sum_{j \in [w_1], k \in [n_b]} \eta_{i,j,k} s_j b_k,$$

for all $i \in [w_3]$, where $\eta_{i,j}, \eta_{i,j,k} \in \mathbb{Z}_p$, the polynomial c'_i is expressed as

$$c'_i = \sum_{j \in [n_\alpha], j' \in [w_1]} \eta'_{i,j,j'} \alpha_j s_{j'} + \sum_{j \in [w'_2]} \hat{\eta}'_{i,j} \tilde{s}_j,$$

for all $i \in [w_4]$, where $\eta'_{i,j,j'}, \hat{\eta}'_{i,j} \in \mathbb{Z}_p$, and the variable c_M is expressed as

$$c_M = \sum_{j \in [w'_2]} \zeta_j \tilde{s}_j + \sum_{j \in [n_\alpha], j' \in [w_1]} \zeta_{j,j'} \alpha_j s_{j'},$$

where $\zeta_j, \zeta_{j,j'} \in \mathbb{Z}_p$.

- $\mathsf{Pair}(x, y, p) \to (\mathbf{e}, \mathbf{E}, \overline{\mathbf{E}})$: On input $p \in \mathbb{N}$, $x \in \mathcal{X}$, and $y \in \mathcal{Y}$, this algorithm outputs a vector $\mathbf{e} \in \mathbb{Z}_p^{w_4}$ and two matrices $\mathbf{E}$ and $\overline{\mathbf{E}}$ of sizes $w_1 \times m_3$ and $w_3 \times m_1$, respectively.

A PES is correct for every p, par, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $P(x, y) = 1$, it holds that $\mathbf{e}\mathbf{c}'^{\intercal} + \mathbf{s}\mathbf{E}\mathbf{k}^{\intercal} + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^{\intercal} = c_M$.

4.5.3 Special symbolic property for GPES

To generalize the symbolic property, we also need to find proper substitutions for the new master-key variables and the ciphertext encodings $\mathbf{c}'$. In addition, we need to be able to account for static corruption of certain variables.

For the master-key variables, we first observe that these occur as lone variables in the key encodings and as common variables in the ciphertext encodings $\mathbf{c}'$, meaning that we only have to be able to multiply them with non-lone ciphertext variables. It is thus sufficient to substitute the master-key variables with vectors (rather than matrices, like the common variables). Because the non-lone ciphertext variables are substituted by vectors of length d_1, we therefore also substitute the master-key variables by vectors of length d_1, so that their inner product yields an integer. In addition to products of master-key variables and non-lone variables, the ciphertext encodings consist of special lone variables, which also need to be substituted by integers.

To ensure that we can replace $e(g, h)^{c_M}$ with the testing value T, we additionally require that all master-key variables and non-lone ciphertext variables that occur in

c_M are equal to $\mathbf{1}_1^{d_1}$. In this way, the products of the simulated components do not yield any parallel instances of xyz.

Finally, to support corruption, we need to ensure that none of the corrupted secret values (such as those related to the master-keys or the lone key variables) contains any parameters that cannot be programmed from the inputs to the complexity assumption. We ensure this by setting their corresponding substitution vectors/matrices to all-zero. This yields the following definition.

Definition 4.5: Special selective symbolic property for GPES

A GPES $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ satisfies the (d_1, d_2)-selective symbolic property for positive integers d_1 and d_2 if there exist deterministic polynomial-time algorithms EncB, EncS, and EncR such that for all p, par, and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $P(x,y) = 0$, and optionally, there exist $\mathfrak{a} \subsetneq [n_\alpha]$, $\mathfrak{b} \subsetneq [n_b]$ (which we call corruptable variables), such that we have that

- $\text{EncB}(x, \mathfrak{a}, \mathfrak{b}) \to \mathbf{a}_1, ..., \mathbf{a}_{n_\alpha} \in \mathbb{Z}_p^{d_1}, \mathbf{B}_1, ..., \mathbf{B}_{n_b} \in \mathbb{Z}_p^{d_1 \times d_2}$;
- $\text{EncR}(x, y, \mathfrak{a}, \mathfrak{b}) \to \mathbf{r}_1, ..., \mathbf{r}_{m_1} \in \mathbb{Z}_p^{d_2}, \hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_{m_2} \in \mathbb{Z}_p^{d_1}$;
- $\text{EncS}(x, \mathfrak{a}, \mathfrak{b}) \to \mathbf{s}_1, ..., \mathbf{s}_{w_1} \in \mathbb{Z}_p^{d_1}, \hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_{w_2} \in \mathbb{Z}_p^{d_2}, \tilde{\mathbf{s}}_1, ..., \tilde{\mathbf{s}}_{w_2'} \in \mathbb{Z}_p$;

such that, if we substitute

$$\hat{s}_{i'} : \hat{\mathbf{s}}_{i'} \qquad \tilde{s}_{i''} : \tilde{\mathbf{s}}_{i''} \qquad s_i b_j : \mathbf{s}_i \mathbf{B}_j \qquad \alpha_l : \mathbf{a}_l^\intercal \qquad \hat{r}_{k'} : \hat{\mathbf{r}}_{k'}^\intercal \qquad r_k b_j : \mathbf{B}_j \mathbf{r}_k^\intercal,$$

for $i \in [w_1], i' \in [w_2], i'' \in [w_2'], j \in [n_b], k \in [m_1], k' \in [m_2], l \in [n_\alpha]$ in all the polynomials of $\mathbf{k}$, $\mathbf{c}$ and $\mathbf{c}'$ (output by EncKey and EncCt, respectively), they evaluate to $\mathbf{0}$. Furthermore,

- for all $j \in [n_\alpha] \setminus \mathfrak{a}, j' \in \overline{[w_1]}$ with $\zeta_{j,j'} \neq 0$, we have that $\mathbf{a}_j = \mathbf{s}_{j'} = \mathbf{1}_1^{d_1}$;
- for $j \in [w_2']$ with $\zeta_j \neq 0$, we have that $\tilde{\mathbf{s}}_j = 1$;
- for $j \in \mathfrak{a}$, we have $\mathbf{a}_j = \mathbf{0}^{d_1}$;
- and for $j \in \mathfrak{b}$, we have that $\mathbf{B}_j = \mathbf{0}^{d_1 \times d_2}$.

Remark 4.1

PESs can be captured under our definition of generalized PES. That is, we can simply set $n_\alpha = 1$, $w_2, w_4 = 0$ and $C_M = \alpha s$. Furthermore, most existing PESs (e.g., [AC17b, Att19]) satisfy the special (d_1, d_2)-selective symbolic property, because they satisfy the symbolic property, and $\mathbf{a} = \mathbf{s} = \mathbf{1}_1^{d_1}$. Therefore, these can be securely instantiated in the selective-security setting with our compiler.

4.5.4 Distribution of the encodings

We also give an explicit definition for the distribution of the encodings over the two source groups $\mathbb{G}$ and $\mathbb{H}$, and the target group $\mathbb{G}_T$ when they are instantiated in our new compiler. Such a distribution should ensure that the correctness of the GPES is preserved, such that the correctness of the PE scheme is also guaranteed. In particular, for the correctness of the decryption, we require that each pair of key and ciphertext encodings that needs to be paired has one encoding in $\mathbb{G}$ and one in $\mathbb{H}$. Furthermore, to ensure that encryption can be performed correctly, the master public keys required in computing a ciphertext encoding element need to be in the same group.

Definition 4.6: Distribution of the encodings over $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$

Let $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ be a GPES for a predicate $P \colon \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ and let $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ be three groups. Let $\mathcal{E}$ denote the set of possible encodings and non-lone variables that can be sampled with Param, EncKey and EncCt, and let $\mathcal{E}' \subseteq \mathcal{E}$ denote its subset containing the master key variables $\boldsymbol{\alpha}$ and ciphertext encodings $\mathbf{c}'$. Then, we define $\mathfrak{D} \colon \mathcal{E} \to \{\mathbb{G}, \mathbb{H}, \mathbb{G}_T\}$ to be the distribution of Γ over $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$ such that the correctness of the encoding is preserved. This is the case, if for every p, par, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ such that $P(x, y) = 1$, it holds that

- $\mathfrak{D}(\mathcal{E}') = \{\mathbb{G}_T\}$, and $\mathfrak{D}(\mathcal{E} \setminus \mathcal{E}') = \{\mathbb{G}, \mathbb{H}\}$;
- for all $i \in [m_3]$, $j \in [w_1]$, if $\mathfrak{D}(k_i) = \mathfrak{D}(s_j)$, then $\mathbf{E}_{j,i} = 0$;
- for all $i \in [w_3]$, $j \in [m_1]$, if $\mathfrak{D}(c_i) = \mathfrak{D}(r_j)$, then $\overline{\mathbf{E}}_{i,j} = 0$;
- for all $k \in [n_b]$ for which there exist some $i \in [w_3], j \in [w_1]$ with $\eta_{i,j,k} \neq 0$, we have $\mathfrak{D}(b_k) = \mathfrak{D}(c_i)$.

4.5.5 Full-domain hashes and random oracles

Sometimes, some of the variables are generated implicitly by a full-domain hash (FDH). For example, this is done to support large universes (see Section 2.5.5) or to link the keys together in decentralized schemes (see Section 2.8.4). Our compiler and proof can easily support the use of full-domain hashes. In that case, the security proof requires the hashes to be modeled as random oracles. In particular, the random oracles answer the queries exactly in the way that it does in a proof where the variable is not generated by an FDH. To capture such random oracle queries in the security proof, we also define a function $\mathcal{F}$ that maps each encoding variable to a natural number, indicating which random oracle is queried.

Definition 4.7: FDH-generated encoding variables

Let $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ be a GPES for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. Let $\mathcal{E}$ denote the set of possible encodings and non-lone variables that can be sampled with Param, EncKey and EncCt. Then, we define $\mathcal{F}\colon \mathcal{E} \to \mathbb{N}$ to be the map that assigns whether the encoding variables are generated by an FDH or not. If not, then the encoding variable is mapped to 0. Otherwise, it is mapped to any integer larger than 0. When the FDH is instantiated, it expects the index of the encoding variable as input, e.g., if $\mathcal{F}(b_{\text{att}}) = 1$, then $\mathcal{H}_1$ expects att as input in the scheme, and outputs $[b_{\text{att}}]_{\mathfrak{D}(b_{\text{att}})}$.

Furthermore, to ensure correctness of the scheme, we require the distribution over the two source groups to be such that, for any common variable b_k that is provided implicitly by a hash, and each associated encoding k_i and c_i, it holds that they are placed in the same group. Similarly, we can define such a restriction for the other variables. Furthermore, if a non-lone variable and a common variable occur together in a product in one of the polynomials, then it cannot be the case that both are generated by an FDH. (It is possible to generate at most one with an FDH, by computing, e.g., $\mathcal{H}(\text{att})^r$ or $\mathcal{H}(\text{GID})^{b_{\text{att}}}$, but not both.) We formalize these restrictions as follows.

Definition 4.8: Correctness of variables generated by an FDH

Let $\mathfrak{D}$ be as in Definition 4.6. Then, for any common variable b_k with $\mathcal{F} > 0$ (i.e., generated implicitly by the full-domain hash), it holds that:

- for all $i \in [m_3]$, if $\mathfrak{D}(k_i) \neq \mathfrak{D}(b_k)$, then $\delta_{i,j,k} = 0$ for all $j \in [m_1]$;
- for all $i \in [w_3]$, if $\mathfrak{D}(c_i) \neq \mathfrak{D}(b_k)$, then $\eta_{i,j,k} = 0$ for all $j \in [w_1]$.

For any non-lone variable r_j or s_j with $\mathcal{F}(r_j), \mathcal{F}(s_j) > 0$, it holds that:

- for all $i \in [m_3]$, if $\mathfrak{D}(k_i) \neq \mathfrak{D}(r_j)$, then $\delta_{i,j,k} = 0$ for all $k \in [n]$;
- for all $i \in [w_3]$, if $\mathfrak{D}(c_i) \neq \mathfrak{D}(s_j)$, then $\eta_{i,j,k} = 0$ for all $k \in [n]$;
- for all $i \in [m_3], k \in [n]$, if $\delta_{i,j,k} \neq 0$, then $\mathcal{F}(b_k) = 0$;
- for all $i \in [w_3], k \in [n]$, if $\eta_{i,j,k} \neq 0$, then $\mathcal{F}(b_k) = 0$.

Furthermore, for each $i \in \mathbb{N}$ with $i > 0$, we require that all the encodings that are mapped to it, i.e., $\mathcal{F}^{-1}(i)$, are either all common variables, or all non-lone key variables, or all non-lone ciphertext variables.

4.5.6 Our complexity assumption

The last ingredient to our compiler is the complexity assumption. The assumption that we use to prove security generically is loosely based on the q-type assumptions used in works that prove selective security, e.g., [RW13, §A]. Roughly, this assumption creates several parallel instances of an assumption similar to the DBDH assumption, augmented with some additional inputs.

Definition 4.9: The (d_1, d_2)-parallel DBDH assumption

Let λ be the security parameter. Let $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ be a pairing over three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p, and let $g \in \mathbb{G}, h \in \mathbb{H}$ be two generators. The challenger generates $\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{c}_\mathsf{i}, \mathsf{c}'_\mathsf{j} \in_R \mathbb{Z}_p$ for all $\mathsf{i} \in [2, d_1], \mathsf{j} \in [2, d_2]$, sets $\mathsf{c}_1 = \mathsf{c}'_1 = 1$ and outputs for all $\mathbb{G}' \in \{\mathbb{G}, \mathbb{H}\}$:

$$\begin{array}{llll}
[\mathsf{x}\mathsf{c}_\mathsf{i}]_{\mathbb{G}'}, & \text{for all } \mathsf{i} \in [d_1] & \left[\frac{\mathsf{xzc}_\mathsf{i}}{\mathsf{c}_{\mathsf{i}'}\mathsf{c}'_\mathsf{j}}\right]_{\mathbb{G}'}, & \text{for all } \mathsf{i}, \mathsf{i}' \in [d_1], \mathsf{i} \neq \mathsf{i}', \mathsf{j} \in [d_2] \\
[\mathsf{y}\mathsf{c}'_\mathsf{j}]_{\mathbb{G}'}, & \text{for all } \mathsf{j} \in [d_2] & \left[\frac{\mathsf{yzc}'_\mathsf{j}}{\mathsf{c}_\mathsf{i}\mathsf{c}'_{\mathsf{j}'}}\right]_{\mathbb{G}'}, & \text{for all } \mathsf{i} \in [d_1], \mathsf{j}, \mathsf{j}' \in [d_2], \mathsf{j} \neq \mathsf{j}' \\
\left[\frac{\mathsf{z}}{\mathsf{c}_\mathsf{i}\mathsf{c}'_\mathsf{j}}\right]_{\mathbb{G}'}, & \text{for all } \mathsf{i} \in [d_1], \mathsf{j} \in [d_2]. & &
\end{array}$$

By setting $\mathsf{c}_1 = \mathsf{c}'_1 = 1$, $[\mathsf{x}]_{\mathbb{G}'}, [\mathsf{y}]_{\mathbb{G}'}, [\mathsf{z}]_{\mathbb{G}'}$ are included in these terms. The challenger also flips a coin $\beta \in_R \mathbb{Z}_p$ and outputs $T \in_R \mathbb{G}_T$ if $\beta = 0$ and $T = e(g, h)^{\mathsf{xyz}}$ if $\beta = 1$. The attacker outputs a guess β' for β. The advantage of the attacker is defined as $\mathsf{Adv}_{(d_1,d_2)\text{-pDBDH}} = |\Pr[\beta' = \beta] - \frac{1}{2}|$. The *$(d_1, d_2)$-parallel DBDH assumption* ((d_1, d_2)-pDBDH) holds if all polynomial-time attackers have at most a negligible advantage, i.e., $\mathsf{Adv}_{(d_1,d_2)\text{-pDBDH}} \leq \text{negl}(\lambda)$.

We prove the following lemma similarly as in [Wat11, RW13]. The proof can be found in the full version of this paper [Ven23b]).

Lemma 4.5

The (d_1, d_2)-parallel DBDH assumption holds in the GGM (Section 2.4.5).

Remark 4.2

Interestingly, for $d_1 = d_2 = 1$, the (d_1, d_2)-parallel DBDH assumption is equivalent to the DBDH assumption (Definition 3.16). An advantage of this is that, if the GPES is such that the special selective symbolic property holds for $d_1 = d_2 = 1$, we automatically obtain an instantiation of a scheme whose security relies on DBDH. In contrast, the q-type assumption on which the Agrawal-Chase compiler relies does not satisfy this property.

4.6 Our generic compiler

Our new generic compiler instantiates the GPES into the pairing-friendly groups $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ in the most obvious way. Roughly, the master public key, the secret keys and the ciphertexts have the following form:

$$\mathrm{MPK} = (e(g,h)^{\boldsymbol{\alpha}}, g'^{\mathbf{b}}), \qquad \mathrm{SK} = (g'^{\mathbf{r}}, g'^{\mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}, \mathbf{b}, y)}),$$
$$\mathrm{CT} = (M \cdot e(g,h)^{c_M}, g'^{\mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x)}, e(g,h)^{\mathbf{c}'(\mathbf{s}, \tilde{\mathbf{s}}, \boldsymbol{\alpha}, x)}),$$

(where g' indicates that either $g' = g$ or $g' = h$ for each entry of the vector in the exponent). More concretely, we define our generic compiler as follows.

Definition 4.10: Our generic compiler

Let $\Gamma = (\mathrm{Param}, \mathrm{EncKey}, \mathrm{EncCt}, \mathrm{Pair})$ be a GPES for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, let $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$ be a pairing over three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p, let $g \in \mathbb{G}, h \in \mathbb{H}$ be two generators and let $\mathfrak{D}\colon \mathcal{E} \to \{\mathbb{G}, \mathbb{H}\}$ be a distribution of the encodings over the two source groups $\mathbb{G}$ and $\mathbb{H}$, and let $\mathcal{F}\colon \mathcal{E} \to \mathbb{N}$ be the map that maps the encoding variables to natural numbers. For each $i \in \mathcal{F}(\mathcal{E}) \setminus \{0\}$, let $\mathcal{H}_i\colon \{0,1\}^* \to \mathbb{G}'$ denote a full-domain hash modeled as a random oracle, where $\mathbb{G}' = \mathfrak{D}(\mathcal{F}^{-1}(i))$ is the group to which the associated encoding variables are mapped. Then, we define the predicate encryption scheme for predicate P as follows:

- Setup(λ, par): On input the security parameter λ and parameters par, this algorithm generates $(n_\alpha, n_b, \boldsymbol{\alpha}, \mathbf{b}) \leftarrow \mathrm{Param}(\mathrm{par})$, sets $\mathrm{MSK} = (\boldsymbol{\alpha}, \{b_i \mid i \in [n_b] \wedge \mathcal{F}(b_i) = 0\})$ as the master secret key, and outputs

  $$\mathrm{MPK} = (A = \{[\alpha_i]_{\mathbb{G}_T}\}_{i \in [n_\alpha]}, \{[b_i]_{\mathfrak{D}(b_i)} \mid i \in [n_b] \wedge \mathcal{F}(b_i) = 0\})$$

  as the master public key. The global parameters are $p, e, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h$.

- KeyGen(MSK, y): On input the master secret key MSK and some $y \in \mathcal{Y}$, this algorithm generates $(m_1, m_2, \mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}, \mathbf{b}, y)) \leftarrow \mathrm{EncKey}(y, p)$, and outputs the secret key

$$\mathrm{SK}_y = (y, \{[r_j]_{\mathfrak{D}(r_j)} \mid j \in [m_1] \wedge \mathcal{F}(r_j) = 0\}, \{[k_i]_{\mathfrak{D}(k_i)}\}_{i \in [m_3]})$$

- Encrypt(MPK, x, M): On input the master public key MPK, $x \in \mathcal{X}$ and message $M \in \mathbb{G}_T$, this algorithm generates $(w_1, w_2, w_2', c_M, \mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x), \mathbf{c}'(\mathbf{s}, \tilde{\mathbf{s}}, \boldsymbol{\alpha}, x)) \leftarrow \mathrm{EncCt}(x, p)$, and outputs the ciphertext

$$\begin{gathered} \mathrm{CT}_x = (x, M \cdot e(g,h)^{c_M}, \\ \{[s_j]_{\mathfrak{D}(s_j)} \mid j \in [w_1] \wedge \mathcal{F}(s_j) = 0\}, \{[c_i]_{\mathfrak{D}(c_i)}\}_{i \in [w_3]}, \{[c_i']_{\mathbb{G}_T}\}_{i \in [w_4]}). \end{gathered}$$

- Decrypt(MPK, SK_y, CT_x): On input the master public key MPK, the secret key SK_y, and the ciphertext CT_x, if $P(x,y) = 1$, then it first obtains $(\mathbf{E}, \overline{\mathbf{E}}) \leftarrow \mathrm{Pair}(x, y, p)$, sets

$$\begin{aligned} \mathcal{P} = {} & \{(s_j, k_i, \mathbf{E}_{j,i}) \mid i \in [m_3], j \in [w_1], \mathbf{E}_{j,i} \neq 0 \wedge \mathfrak{D}(s_j) = \mathbb{G}\} \\ & \cup \{(k_i, s_j, \mathbf{E}_{j,i}) \mid i \in [m_3], j \in [w_1], \mathbf{E}_{j,i} \neq 0 \wedge \mathfrak{D}(s_j) = \mathbb{H}\} \\ & \cup \{(r_j, c_i, \overline{\mathbf{E}}_{i,j}) \mid i \in [w_3], j \in [m_1], \overline{\mathbf{E}}_{i,j} \neq 0 \wedge \mathfrak{D}(r_j) = \mathbb{G}\} \\ & \cup \{(c_i, r_j, \overline{\mathbf{E}}_{i,j}) \mid i \in [w_3], j \in [m_1], \overline{\mathbf{E}}_{i,j} \neq 0 \wedge \mathfrak{D}(r_j) = \mathbb{H}\}, \end{aligned}$$

  and then retrieves

$$\prod_{i \in [n_\alpha]} [c_i']_{\mathbb{G}_T}^{\mathbf{e}_i} \prod_{(\mathfrak{l}, \mathfrak{r}, \mathfrak{e}) \in \mathcal{P}} e([\mathfrak{l}]_{\mathbb{G}}, [\mathfrak{r}]_{\mathbb{H}})^{\mathfrak{e}} = e(g,h)^{\mathbf{e}\mathbf{c}'^{\intercal} + \mathbf{s}\mathbf{E}\mathbf{k}^{\intercal} + \mathbf{c}\overline{\mathbf{E}}\mathbf{r}^{\intercal}} = e(g,h)^{c_M}.$$

The correctness of the scheme is preserved under the correctness of the GPES and the preservation-of-correctness property of the distribution (Definition 4.6).

Theorem 4.1

If Γ satisfies the special symbolic property (Definition 4.5), and the (d_1, d_2)-parallel DBDH assumption holds in the groups $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$, then the PE scheme in Definition 4.10 is selectively secure. (If we allow corruption of variables, the scheme is also secure under static corruption of variables.)

Proof. Suppose some attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ exists that can break the scheme in Definition 4.10 with non-negligible advantage ε. We show that it can be used to construct an attacker $\mathcal{A}_{(d_1,d_2)\text{-pDBDH}}$ with non-negligible advantage in a security game with challenger $\mathcal{C}_{(d_1,d_2)\text{-pDBDH}}$ as well.

- **Initialization phase:** Attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ commits to $x^* \in \mathcal{X}$, and corruptable $\mathfrak{a} \subsetneq [n_\alpha]$ and $\mathfrak{b} \subsetneq [n_b]$ as in the special symbolic property, and sends those to challenger $\mathcal{C}_{\text{PE,IND-CPA}}$. (Note that $\mathfrak{a} = \mathfrak{b} = \emptyset$, if we do not allow corruption.) Let $\mathcal{C}_{(d_1,d_2)\text{-pDBDH}}$ be a challenger that sends the terms of the assumption in Definition 4.9, where d_1, d_2 as in the special symbolic property (Definition 4.5). Let $\mathrm{EncB}, \mathrm{EncR}, \mathrm{EncS}$ be the three algorithms that generate the necessary substitutions for the variables in the encodings.

- **Setup phase:** Challenger $\mathcal{C}_{\text{PE,IND-CPA}}$ runs $(\{\mathbf{a}_i, \mathbf{B}_j\}_{i\in[n_\alpha],j\in[n_b]}) \leftarrow \mathrm{EncB}(x^*, \mathfrak{a}, \mathfrak{b})$ to obtain the necessary substitutions for the master public key MPK. It constructs the master public key as

$$\mathrm{MPK} = \Bigg(\{A_j = e(g,h)^{\bar{\alpha}_j} \cdot \prod_{\mathsf{i}\in[d_1]} e([\mathsf{x}]_{\mathbb{G}}^{\mathsf{c_i}}, [\mathsf{z}]_{\mathbb{H}})^{(\mathbf{a}_j)_\mathsf{i}}\}_{j\in[n_\alpha]\setminus\mathfrak{a}},$$
$$\left\{[b_k]_{\mathfrak{D}(b_k)} = [\bar{b}_k]_{\mathfrak{D}(b_k)} \cdot \prod_{\mathsf{i}\in[d_1],\mathsf{j}\in[d_2]} \left[(\mathbf{B}_k)_{\mathsf{i,j}} \frac{\mathsf{z}}{\mathsf{c_i c'_j}}\right]_{\mathfrak{D}(b_k)}\right\}_{k\in\{k'\in[n]\setminus\mathfrak{b}|\mathcal{F}(b_{k'})=0\}},$$
$$\{A_j = e(g,h)^{\bar{\alpha}_j}\}_{j\in\mathfrak{a}}, \left\{[b_k]_{\mathfrak{D}(b_k)} = [\bar{b}_k]_{\mathfrak{D}(b_k)}\right\}_{k\in\mathfrak{b}}\Bigg),$$

  where $\bar{\alpha}, \bar{b}_k \in_R \mathbb{Z}_p$ for all $k \in [n]$. Note that $\left[(\mathbf{B}_k)_{\mathsf{i,j}} \frac{\mathsf{z}}{\mathsf{c_i c'_j}}\right]_{\mathfrak{D}(b_k)}$ can be generated from the terms in the assumption by computing $\left[\frac{\mathsf{z}}{\mathsf{c_i c'_j}}\right]_{\mathfrak{D}(b_k)}^{(\mathbf{B}_k)_{\mathsf{i,j}}}$.

- **Random oracle query phase for $\mathcal{H}_i$:** If attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ queries the random oracle $\mathcal{H}_i$ the input corresponding to common variable b_k (with $\mathcal{F}(b_k) = i$), then it obtains $(\mathbf{a}_1, ..., \mathbf{a}_{n_\alpha}, \mathbf{B}_1, ..., \mathbf{B}_n) \leftarrow \mathrm{EncB}(x^*, \mathfrak{a}, \mathfrak{b})$ and outputs

$$[b_k]_{\mathfrak{D}(b_k)} = \begin{cases} [\bar{b}_k]_{\mathfrak{D}(b_k)} \cdot \prod_{\mathsf{i}\in[d_1],\mathsf{j}\in[d_2]} \left[(\mathbf{B}_k)_{\mathsf{i,j}} \frac{\mathsf{z}}{\mathsf{c_i c'_j}}\right]_{\mathfrak{D}(b_k)} & \text{if } k \in [n] \setminus \mathfrak{b} \\ [\bar{b}_k]_{\mathfrak{D}(b_k)} & \text{if } k \in \mathfrak{b}, \end{cases}$$

  where $\bar{b}_k \in_R \mathbb{Z}_p$. If the oracle is queried implicitly for non-lone variable r_j, it runs $(\mathbf{r}_1, ..., \mathbf{r}_{m_1}, \hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_{m_2}) \leftarrow \mathrm{EncR}(x^*, y, \mathfrak{a}, \mathfrak{b})$ and outputs

$$[r_j]_{\mathfrak{D}(r_j)} = [\bar{r}_j]_{\mathfrak{D}(r_j)} \cdot \prod_{\mathsf{i}\in[d_1]} [(\mathbf{r}_j)_\mathsf{i} \mathsf{x} \mathsf{c_i}]_{\mathfrak{D}(r_j)},$$

and if it is queried for non-lone variable s_j, it runs $(\mathbf{s}_1, ..., \mathbf{s}_{w_1}, \hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_{w_2}) \leftarrow \mathrm{EncS}(x^*, \mathfrak{a}, \mathfrak{b})$ and outputs

$$[s_j]_{\mathfrak{D}(s_j)} = [\bar{s}_j]_{\mathfrak{D}(s_j)} \cdot \prod_{\mathsf{j}\in[d_2]} [(\mathbf{s}_j)_{\mathsf{j}}\mathsf{y}\mathsf{c}'_{\mathsf{j}}]_{\mathfrak{D}(s_j)},$$

where $\bar{r}_j, \bar{s}_j \in_R \mathbb{Z}_p$.

- **First query phase:** Attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ queries secret keys for $y \in \mathcal{Y}$. Challenger $\mathcal{C}_{\text{PE,IND-CPA}}$ generates $(\mathbf{r}_1, ..., \mathbf{r}_{m_1}, \hat{\mathbf{r}}_1, ..., \hat{\mathbf{r}}_{m_2}) \leftarrow \mathrm{EncR}(x^*, y, \mathfrak{a}, \mathfrak{b})$ and $\bar{r}_j \in_R \mathbb{Z}_p$ for all $j \in [m_1]$ and programs the secret key as

$$\mathrm{SK}_y = (y, \{[r_j]_{\mathfrak{D}(r_j)} = [\bar{r}_j]_{\mathfrak{D}(r_j)} \cdot \prod_{\mathsf{i}\in[d_1]} [(\mathbf{r}_j)_{\mathsf{i}}\mathsf{x}\mathsf{c}_{\mathsf{i}}]_{\mathfrak{D}(r_j)}\}_{j\in[m_1]}, \{[k_i]_{\mathfrak{D}(k_i)}\}_{i\in[m_3]}),$$

such that $[k_i]_{\mathfrak{D}(k_i)} = [\sum_{j\in[n_\alpha]} \delta_{i,j}\alpha_j + \sum_{j\in[m_2]} \delta_{i,j}\hat{r}_j + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k} r_j b_k]_{\mathfrak{D}(k_i)}$ is programmed by implicitly setting

$$[\alpha_j]_{\mathfrak{D}(\alpha_j)} = [\bar{\alpha}_j]_{\mathfrak{D}(\alpha_j)} \cdot \prod_{\mathsf{j}\in[d_2]} \left[(\mathbf{a}_j)_{\mathsf{j}} \frac{\mathsf{x}\mathsf{z}}{\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(\alpha_j)} \quad \text{for all } j \in [n_\alpha],$$

$$[\hat{r}_j]_{\mathfrak{D}(\hat{r}_j)} = \prod_{\mathsf{j}\in[d_2]} \left[(\hat{\mathbf{r}}_j)_{\mathsf{j}} \frac{\mathsf{x}\mathsf{z}}{\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(\hat{r}_j)} \quad \text{for all } j \in [d_1],$$

$$[r_j b_k]_{\mathfrak{D}(b_k)} = [\bar{r}_j b_k]_{\mathfrak{D}(b_k)} \cdot [r_j \bar{b}_k]_{\mathfrak{D}(b_k)} \cdot \prod_{\mathsf{i},\mathsf{i}'\in[d_1],\mathsf{j}\in[d_2]} \left[(\mathbf{r}_j)_{\mathsf{i}'} (\mathbf{B}_k)_{\mathsf{i},\mathsf{j}} \frac{\mathsf{z}\mathsf{x}\mathsf{c}_{\mathsf{i}'}}{\mathsf{c}_{\mathsf{i}}\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)}$$

for all $j \in [d_2]$, such that the only terms we cannot program with the terms of the (d_1, d_2)-pDBDH assumption are those with $\frac{\mathsf{x}\mathsf{z}}{\mathsf{c}'_{\mathsf{j}}}$ for all $\mathsf{j} \in [d_2]$, which includes the rightmost terms of $[r_j b_k]_{\mathfrak{D}(b_k)}$ for $\mathsf{i} = \mathsf{i}'$, i.e.,

$$\prod_{\mathsf{i}\in[d_1],\mathsf{j}\in[d_2]} \left[(\mathbf{B}_k)_{\mathsf{i},\mathsf{j}} (\mathbf{r}_j)_{\mathsf{i}} \frac{\mathsf{z}\mathsf{x}\mathsf{c}_{\mathsf{i}}}{\mathsf{c}_{\mathsf{i}}\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)} = \prod_{\mathsf{i}\in[d_1],\mathsf{j}\in[d_2]} \left[(\mathbf{B}_k \mathbf{r}_j^{\intercal})_{\mathsf{j}} \frac{\mathsf{x}\mathsf{z}}{\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)}.$$

For these terms, it follows from the selective property that these are canceled in the simulation of k_i, because

$$k_i = \sum_{j\in[n_\alpha]} \delta_{i,j}\mathbf{a}_j + \sum_{j\in[m_2]} \hat{\delta}_{i,j}\hat{\mathbf{r}}_j + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k}\mathbf{B}_k\mathbf{r}_j^{\intercal} = \mathbf{0}^{d_2},$$

from which it follows that for all $\mathsf{j} \in [d_2]$:

$$\left(\sum_{j\in[n_\alpha]} \delta_{i,j}(\mathbf{a}_j)_{\mathsf{j}} + \sum_{j\in[m_2]} \hat{\delta}_{i,j}(\hat{\mathbf{r}}_j)_{\mathsf{j}} + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k}(\mathbf{B}_k\mathbf{r}_j^\intercal)_{\mathsf{j}}\right)\frac{\mathsf{xz}}{\mathsf{c}'_{\mathsf{j}}}$$
$$= \sum_{j\in[n_\alpha]} \delta_{i,j}(\mathbf{a}_j)_{\mathsf{j}}\frac{\mathsf{xz}}{\mathsf{c}'_{\mathsf{j}}} + \sum_{j\in[m_2]} \hat{\delta}_{i,j}(\hat{\mathbf{r}}_j)_{\mathsf{j}}\frac{\mathsf{xz}}{\mathsf{c}'_{\mathsf{j}}} + \sum_{j\in[m_1],k\in[n]} \delta_{i,j,k}(\mathbf{B}_k\mathbf{r}_j^\intercal)_{\mathsf{j}}\frac{\mathsf{xz}}{\mathsf{c}'_{\mathsf{j}}} = 0.$$

Note that the rest of the terms can be programmed as follows:

$$[\bar{r}_j b_k]_{\mathfrak{D}(b_k)} = [b_k]_{\mathfrak{D}(b_k)}^{\bar{r}_j}, \qquad [r_j \bar{b}_k]_{\mathfrak{D}(b_k)} = [r_j]_{\mathfrak{D}(b_k)}^{\bar{b}_k},$$
$$\prod_{\mathsf{i},\mathsf{i}'\in[d_1],\mathsf{i}\neq\mathsf{i}',\mathsf{j}\in[d_2]} \left[(\mathbf{r}_j)_{\mathsf{i}'}(\mathbf{B}_k)_{\mathsf{i},\mathsf{j}}\frac{\mathsf{zxc}_{\mathsf{i}'}}{\mathsf{c}_{\mathsf{i}}\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)} = \prod_{\mathsf{i},\mathsf{i}'\in[d_1],\mathsf{i}\neq\mathsf{i}',\mathsf{j}\in[d_2]} \left[\frac{\mathsf{xzc}_{\mathsf{i}'}}{\mathsf{c}_{\mathsf{i}}\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)}^{(\mathbf{r}_j)_{\mathsf{i}'}(\mathbf{B}_k)_{\mathsf{i},\mathsf{j}}}.$$

- **Challenge phase:** Attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ sends two messages M_0 and M_1 of equal length in $\mathbb{G}_T$ to challenger $\mathcal{C}_{\text{PE,IND-CPA}}$. The challenger flips a coin, i.e., $\beta \in_R \{0,1\}$, encrypts M_β under x^* and computes the following ciphertext CT_{x^*} to the attacker. First, it runs $(\mathbf{s}_1, ..., \mathbf{s}_{w_1}, \hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_{w_2}) \leftarrow \text{EncS}(x^*, \mathfrak{a}, \mathfrak{b})$, and it sets

  $$\{[s_j]_{\mathfrak{D}(s_j)} = [\bar{s}_j]_{\mathfrak{D}(s_j)} \cdot \prod_{\mathsf{j}\in[d_2]} [(\mathbf{s}_j)_{\mathsf{j}}\mathsf{yc}'_{\mathsf{j}}]_{\mathfrak{D}(s_j)}\}_{j\in[w_1]},$$

  such that $[c_i]_{\mathfrak{D}(c_i)} = [\sum_{j\in[w_2]} \eta_{i,j}\hat{s}_j + \sum_{j\in[w_1],k\in[n]} \eta_{i,j,k}s_j b_k]_{\mathfrak{D}(c_i)}$ can be programmed by implicitly setting

  $$\{[\hat{s}_j]_{\mathfrak{D}(s_j)} = \prod_{\mathsf{i}\in[d_1]} \left[(\hat{\mathbf{s}}_j)_{\mathsf{i}}\frac{\mathsf{yz}}{\mathsf{c}_{\mathsf{i}}}\right]_{\mathfrak{D}(s_j)}\}_{j\in[w_2]},$$
  $$[s_j b_k]_{\mathfrak{D}(c_i)} = [\bar{s}_j b_k]_{\mathfrak{D}(c_i)} \cdot [s_j \bar{b}_k]_{\mathfrak{D}(c_i)} \cdot \prod_{\mathsf{i}\in[d_1],\mathsf{j},\mathsf{j}'\in[d_2]} \left[(\mathbf{s}_j)_{\mathsf{j}'}(\mathbf{B}_k)_{\mathsf{i},\mathsf{j}}\frac{\mathsf{yzc}'_{\mathsf{j}'}}{\mathsf{c}_{\mathsf{i}}\mathsf{c}'_{\mathsf{j}}}\right]_{\mathfrak{D}(b_k)}.$$

  Only the terms with $\frac{\mathsf{yz}}{\mathsf{c}_{\mathsf{i}}}$ (i.e., for $\mathsf{j} = \mathsf{j}'$) cannot be programmed from the terms in the assumption, but are canceled in the simulation of $[c_i]_{\mathfrak{D}(c_i)}$, because

  $$c_i = \sum_{j\in[w_2]} \eta_{i,j}\hat{\mathbf{s}}_j + \sum_{j\in[w_1],k\in[n]} \eta_{i,j,k}\mathbf{s}_j\mathbf{B}_k = \mathbf{0}^{d_1},$$

  from which it follows that, for all $\mathsf{i} \in [d_1]$, we have

  $$\left(\textstyle\sum_{j\in[w_2]} \eta_{i,j}(\hat{\mathbf{s}}_j)_{\mathsf{i}} + \sum_{j\in[w_1],k\in[n]} \eta_{i,j,k}(\mathbf{s}_j\mathbf{B}_k)_{\mathsf{i}}\right)\tfrac{\mathsf{yz}}{\mathsf{c}_{\mathsf{i}}} = \textstyle\sum_{j\in[w_2]} \eta_{i,j}(\hat{\mathbf{s}}_j)_{\mathsf{i}}\tfrac{\mathsf{yz}}{\mathsf{c}_{\mathsf{i}}} + \sum_{j\in[w_1],k\in[n]} \eta_{i,j,k}(\mathbf{s}_j\mathbf{B}_k)_{\mathsf{i}}\tfrac{\mathsf{yz}}{\mathsf{c}_{\mathsf{i}}} = 0.$$

The other terms can be programmed by computing

$$[\bar{s}_j b_k]_{\mathfrak{D}(c_i)} = [b_k]_{\mathfrak{D}(c_i)}^{\bar{s}_j}, \qquad [s_j \bar{b}_k]_{\mathfrak{D}(c_i)} = [s_j]_{\mathfrak{D}(c_i)}^{\bar{b}_k},$$

$$\prod_{\mathsf{i} \in [d_1], \mathsf{j}, \mathsf{j}' \in [d_2], \mathsf{j} \neq \mathsf{j}'} \left[(\mathbf{s}_j)_{\mathsf{j}'} (\mathbf{B}_k)_{\mathsf{i},\mathsf{j}} \frac{\mathsf{yzc}'_{\mathsf{j}'}}{\mathsf{c}_\mathsf{i} \mathsf{c}'_\mathsf{j}} \right]_{\mathfrak{D}(b_k)} = \prod_{\mathsf{i} \in [d_1], \mathsf{j}, \mathsf{j}' \in [d_2], \mathsf{j} \neq \mathsf{j}'} \left[\frac{\mathsf{yzc}'_{\mathsf{j}'}}{\mathsf{c}_\mathsf{i} \mathsf{c}'_\mathsf{j}} \right]_{\mathfrak{D}(b_k)}^{(\mathbf{s}_j)_{\mathsf{j}'} (\mathbf{B}_k)_{\mathsf{i},\mathsf{j}}}.$$

Then, $[c'_i]_{\mathbb{G}_T}$ can be programmed by implicitly setting $\tilde{s}_j = \mathsf{xyz}$, by using that we can compute $\left[\mathsf{xyz} \frac{\mathsf{c}'_{\mathsf{j}'}}{\mathsf{c}'_\mathsf{j}}\right]_{\mathbb{G}_T} = e([\mathsf{xc_i}]_{\mathbb{G}}, \left[\frac{\mathsf{yzc}'_\mathsf{j}}{\mathsf{c_i c}'_{\mathsf{j}'}}\right]_{\mathbb{H}})$ for all $\mathsf{j} \neq \mathsf{j}' \in [d_2]$, and

$$c'_i = \sum_{j \in [n_\alpha], j' \in [w_1]} \eta'_{i,j,j'} \boldsymbol{\alpha}_j \mathbf{s}_{j'}^\intercal + \sum_{j \in [w'_2]} \hat{\eta}'_{i,j} \tilde{\mathbf{s}}_j = 0.$$

Hence,

$$\begin{aligned}
[c'_i]_{\mathbb{G}_T} = & \prod_{j \in [n_\alpha], j' \in [w_1]} [s_{j'}]_{\mathbb{G}_T}^{\eta'_{i,j,j'} \bar{\alpha}_j} \cdot \prod_{j \in [n_\alpha], j' \in [w_1]} [\alpha_j]_{\mathbb{G}_T}^{\eta'_{i,j,j'} \bar{s}_{j'}} \\
& \cdot \prod_{j \in [n_\alpha], j' \in [w_1], \mathsf{j}, \mathsf{j}' \in [d_2]} \left[(\mathbf{a}_j)_\mathsf{j} (\mathbf{s}_{j'})_{\mathsf{j}'} \mathsf{yc}'_{\mathsf{j}'} \frac{\mathsf{xz}}{\mathsf{c}'_\mathsf{j}} \right]_{\mathbb{G}_T}^{\eta'_{i,j,j'}} \cdot \prod_{j \in [w'_2]} [\tilde{\mathbf{s}}\mathsf{xyz}]_{\mathbb{G}_T}^{\hat{\eta}'_{i,j}} \\
= & \prod_{j \in [n_\alpha], j' \in [w_1]} \left([s_{j'}]_{\mathbb{G}_T}^{\eta'_{i,j,j'} \bar{\alpha}_j} \cdot A_j^{\eta'_{i,j,j'} \bar{s}_{j'}} \cdot \prod_{\mathsf{j}, \mathsf{j}' \in [d_2], \mathsf{j} \neq \mathsf{j}'} \left[\mathsf{xyz} \frac{\mathsf{c}'_{\mathsf{j}'}}{\mathsf{c}'_\mathsf{j}} \right]_{\mathbb{G}_T}^{\eta'_{i,j,j'} (\mathbf{a}_j)_\mathsf{j} (\mathbf{s}_{j'})_{\mathsf{j}'}} \right) \\
& \cdot [\mathsf{xyz}]_{\mathbb{G}_T}^{\sum_{j \in [n_\alpha], j' \in [w_1]} \eta'_{i,j,j'} (\mathbf{a}_j)(\mathbf{s}_{j'})^\intercal + \sum_{j \in [w'_2]} \hat{\eta}'_{i,j} \tilde{\mathbf{s}}},
\end{aligned}$$

where $[s_{j'}]_{\mathbb{G}_T}$ can be computed from $[s_{j'}]_{\mathfrak{D}(s_{j'})}$. Furthermore, let $\mathcal{J}_1 = \{(j, j') \in [n_\alpha] \times [w_1] \mid \zeta_{j,j'} \neq 0\}$ and $\mathcal{J}_2 = \{j \in [w_2] \mid \zeta_j \neq 0\}$. Then, we let

$$\begin{aligned}
C &= M_\beta \cdot \prod_{(j,j') \in \mathcal{J}_1} A_j^{\zeta_{j,j'} s_{j'}} \prod_{j \in \mathcal{J}_2} e(g, h)^{\zeta_j \hat{s}_j} \\
&= M_\beta \cdot \prod_{(j,j') \in \mathcal{J}_1} [s_{j'}]_{\mathbb{G}_T}^{\bar{\alpha}_j \zeta_{j,j'}} \cdot \prod_{(j,j') \in \mathcal{J}_1} [\mathsf{xyz}]_{\mathbb{G}_T}^{\zeta_{j,j'}} \prod_{j \in \mathcal{J}_2} [\mathsf{xyz}]_{\mathbb{G}_T}^{\zeta_j} \\
&= M_\beta \cdot \prod_{(j,j') \in \mathcal{J}_1} [s_{j'}]_{\mathbb{G}_T}^{\bar{\alpha}_j \zeta_{j,j'}} \cdot T^{\sum_{(j,j') \in \mathcal{J}_1} \zeta_{j,j'} + \sum_{j \in \mathcal{J}_2} \zeta_j},
\end{aligned}$$

which is well-formed if $T = e(g, h)^{\mathsf{xyz}}$. It outputs the ciphertext as

$$\mathrm{CT}^*_{x^*} = (x^*, C, \{[s_j]_{\mathfrak{D}(s_j)}\}_{j \in [w_1]}, \{[c_i]_{\mathfrak{D}(c_i)}\}_{i \in [w_3]}, \{[c'_i]_{\mathbb{G}_T}\}_{i \in [w_4]})$$

- **Second query phase:** This phase is identical to the first query phase.
- **Decision phase:** Attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ outputs a guess β' for β. If $\beta' = \beta$, then attacker $\mathcal{A}_{(d_1,d_2)\text{-pDBDH}}$ concludes that the ciphertext was well-formed. Thus, it outputs that $T = e(g,h)^{\mathsf{xyz}}$, and otherwise, it outputs that $T \in_R \mathbb{G}_T$.

The probability that attacker $\mathcal{A}_{(d_1,d_2)\text{-pDBDH}}$ guesses correctly when $T = e(g,h)^{\mathsf{xyz}}$ holds corresponds to the success probability of attacker $\mathcal{A}_{\text{PE,IND-CPA}}$, i.e., $\varepsilon + \frac{1}{2}$. If $T \in_R \mathbb{G}_T$ holds, then attacker $\mathcal{A}_{\text{PE,IND-CPA}}$ guesses at random, and thus, attacker $\mathcal{A}_{(d_1,d_2)\text{-pDBDH}}$ also guesses at random. Hence, the advantage $\mathsf{Adv}_{\mathcal{A}_{(d_1,d_2)\text{-pDBDH}}} = \varepsilon + \frac{1}{2} - \frac{1}{2} = \varepsilon$ is non-negligible. □

4.6.1 The new generic compiler in the multi-authority setting

Although our regular compiler can also prove security of multi-authority schemes, it does not explicitly consider multiple authorities. To convert the compiler to the multi-authority setting, we need to split the setup into the global setup and the authority setup, in which a subset of the parameters, associated with some authority, is generated. Furthermore, the key generation should be fragmented across authorities, meaning that it should be possible to split the key generation into independent parts. For this to work properly in practice, any non-lone key variable that occurs across multiple authorities needs to be generated by an FDH. By extension, for any such non-lone variables, the substituted vector as in the (special) symbolic property often depends on the entire $y \in \mathcal{Y}$, rather than only the subset $y_{\mathcal{A}} \subseteq y$ that is relevant for one authority with identifier $\mathcal{A}$. In this case, we require the static security model. For the compiler in the multi-authority setting, we define the following two properties.

Definition 4.11: Independent encodings

Let Γ = (Param, EncKey, EncCt, Pair) be a GPES for a predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, and let $\mathcal{F}$ be the FDH-generated encoding assignment map (Definition 4.7). Let $\mathcal{A}_1, ..., \mathcal{A}_{n_{\text{aut}}}$ be $n_{\text{aut}} \in \mathbb{N}$ authorities, such that $\mathcal{Y}_{\mathcal{A}_i} \subseteq \mathcal{Y}$ denotes the set of predicates managed by $\mathcal{A}_i$, which are disjoint, i.e., $\mathcal{Y}_{\mathcal{A}_i} \cap \mathcal{Y}_{\mathcal{A}_j} = \emptyset$ for all $i \neq j$. The GPES has independent encodings, if the following holds:

- we can find maps $\mathfrak{A}_\alpha\colon [n_\alpha] \to [n_{\text{aut}}]$ and $\mathfrak{A}_b\colon [n_b] \to [n_{\text{aut}}]$, where we have $(n_\alpha, n_b, \boldsymbol{\alpha}, \mathbf{b}) \leftarrow \text{Param}(\text{par})$. Let $\boldsymbol{\alpha}_{|l} = \{\alpha_i \mid i \in \mathfrak{A}_\alpha^{-1}(l)\}$ and $\mathbf{b}_{|l} = \{b_i \mid i \in \mathfrak{A}_b^{-1}(l)\}$ for all authorities $\mathcal{A}_l$;
- for all $y_{\text{GID}} = \{y_{\text{GID},\mathcal{A}_l}\}_{l \in [n_{\text{aut}}]}$, if we obtain $(m_{1,l}, m_{2,l}, \mathbf{k}_l(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}_{|l}, \mathbf{b}_{|l}, y_{\text{GID},\mathcal{A}_l})) \leftarrow \text{EncKey}(y_{\text{GID},\mathcal{A}_l}, p)$ for all $y_{\text{GID},\mathcal{A}_l}$, then it should hold that running $(m_1, m_2, \mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}, \mathbf{b}, y_{\text{GID}})) \leftarrow \text{EncKey}(y_{\text{GID}}, p)$ yields $\mathbf{k}(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}, \mathbf{b}, y_{\text{GID}}))$ that is equivalent to $\{\mathbf{k}_l(\mathbf{r}, \hat{\mathbf{r}}, \boldsymbol{\alpha}_l, \mathbf{b}_l, y_{\text{GID},\mathcal{A}_l})\}_{l \in [n_{\text{aut}}]}$;

- for all $l \in [n_{\text{aut}}]$, let $\mathbf{r}_{|l} \subseteq \mathbf{r}$ and $\hat{\mathbf{r}}_{|l} \subseteq \hat{\mathbf{r}}$ be the subsets of non-lone and lone key variables for which $\mathbf{k}_l$ has a non-zero coefficient. Then, for all $r_j \in \mathbf{r}$ for which $l \neq l'$ exist such that $r_j \in \mathbf{r}_{|l} \cap \mathbf{r}_{|l'}$, it should hold that $\mathcal{F}(r_j) > 0$, and similarly, for $\hat{r}_j \in \hat{\mathbf{r}}$ with $l \neq l'$ such that $\hat{r}_j \in \mathbf{r}_{|l} \cap \mathbf{r}_{|l'}$, we have $\mathcal{F}(\hat{r}_j) > 0$.

Then, we convert the generic compiler in Definition 4.10 to the multi-authority setting as follows.

Definition 4.12: Our multi-authority compiler

Let $\Gamma = (\text{Param}, \text{EncKey}, \text{EncCt}, \text{Pair})$ be a GPES for a predicate $P \colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ as in Definition 4.10, with the additional property that its encodings are independent (Definition 4.11). Then, in the multi-authority setting, almost all algorithms are the same as in Definition 4.10, except that we replace the Setup and KeyGen by:

- GlobalSetup(λ, par): On input the security parameter λ and parameters par, this algorithm outputs the global parameters $\text{GP} = (p, e, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h)$.
- AuthoritySetup(GP): On input the global parameters GP, this probabilistic algorithm outputs the authority identifier $\mathcal{A}_l$, sets $\text{MSK}_{\mathcal{A}_l} \leftarrow (\boldsymbol{\alpha}_{|l}, \{b_i \mid b_i \in \mathbf{b}_{|l} \wedge \mathcal{F}(b_i) = 0\})$, and outputs
$$\text{MPK} = (A = \{[\alpha_i]_{\mathbb{G}_T} \mid \alpha_i \in \boldsymbol{\alpha}_{|l}\}, \{[b_i]_{\mathfrak{D}(b_i)} \mid b_i \in \mathbf{b}_{|l} \wedge \mathcal{F}(b_i) = 0\})$$
as the master public key. Note that $\boldsymbol{\alpha}_{|l}$ and $\mathbf{b}_{|l}$ are as in Definition 4.11.
- KeyGen($\mathcal{A}_l, \text{MSK}_{\mathcal{A}_l}, \text{GID}, y_{\text{GID},\mathcal{A}_l}$): On input the master secret key $\text{MSK}_{\mathcal{A}}$ of authority $\mathcal{A}_l$ and some $y_{\text{GID},\mathcal{A}_l} \in \mathcal{Y}_{\mathcal{A}_l}$ for identifier GID, this algorithm generates $(m_{1,l}, m_{2,l}, \mathbf{k}_l(\mathbf{r}_{|l}, \hat{\mathbf{r}}_{|l}, \boldsymbol{\alpha}_{|l}, \mathbf{b}_{|l}, y_{\text{GID},\mathcal{A}_l})) \leftarrow \text{EncKey}(y_{\text{GID},\mathcal{A}_l}, p)$, and outputs the secret key as
$$\text{SK}_{\text{GID},\mathcal{A}_l,y_{\text{GID},\mathcal{A}_l}} = (y_{\text{GID},\mathcal{A}_l}, \{[r_j]_{\mathfrak{D}(r_j)} \mid r_j \in \mathbf{r}_{|l} \wedge \mathcal{F}(r_j) = 0\}, \{[k_{i,l}]_{\mathfrak{D}(k_{i,l})}\}_{i \in [m_{3,i}]}).$$

The security proof for the multi-authority compiler is almost identical to the proof for Theorem 4.1 (see the full version [Ven23b]). In particular, we can first prove (from the independent encodings property and in the static-security model) that the public and secret keys, and ciphertexts produced by the multi-authority compiler are indistinguishable from the keys and ciphertexts produced by the regular compiler. Then, it follows from the security of the regular compiler that the scheme produced by the multi-authority compiler is also secure.

Theorem 4.2

If Γ has independent encodings and satisfies the special symbolic property (Definition 4.5), and the (d_1, d_2)-parallel DBDH assumption holds in $\mathbb{G}$, $\mathbb{H}$, and $\mathbb{G}_T$, then the scheme in Definition 4.10 is statically secure. The scheme is also secure under static corruption, if the special symbolic property holds for $\mathfrak{a} = \bigcup_{l \in \mathfrak{C}} \boldsymbol{\alpha}_{|l}$ and $\mathfrak{b} = \bigcup_{l \in \mathfrak{C}} \mathbf{b}_{|l}$, where $\mathfrak{C}$ denotes the set of corrupted authorities.

4.7 New schemes

To illustrate the effectiveness of our new compiler, we give several new CP-ABE constructions. In particular, these constructions can be instantiated with our new compiler, while existing full-security compilers cannot instantiate them. In this section, we give several new decentralized large-universe CP-ABE schemes. In the proofs, we use a different technique than the "zero-out lemma" as used in statically-secure decentralized ABE [RW15, DKW23]. We also give large-universe variants of Wat11 (Construction 4.1) and AC17 (Construction 4.3), and investigate whether RW13 (Construction 4.2) and the large-universe variants of Wat11 and AC17 can support an attribute-wise key generation (Section 2.9.3).

For all these schemes, we assume that $\mathcal{F}$ maps the variables to 0 unless otherwise specified. We do not define maps for $\mathfrak{D}$, as the proofs generalize to any such map that is correct. We also let $\mathbf{w}$ (with $w_1 = 1$) be the vector orthogonal to all $\mathbf{A}_j$ with $j \in \Upsilon$ (Definition 2.5). The access policy of each decentralized scheme is extended with another map $\tilde{\rho}\colon [n_1] \to [n_{\text{aut}}]$, which maps each row to an authority, and similarly, we extend the attribute set with a map $\tilde{\rho}_{\mathcal{S}}\colon \mathcal{S} \to [n_{\text{aut}}]$, which maps each attribute in the set to an authority. In the proofs for decentralized ABE and ABE with attribute-wise key generation, we need the entire key set $\mathcal{S}$ to construct the substitution vector of one or more key variables. Therefore, when instantiating those PESs with the multi-authority compiler, the resulting schemes are statically secure.

4.7.1 Efficient decentralized large-universe CP-ABE from FDH

We first give a scheme that is similar to the Rouselakis-Waters decentralized scheme (RW15) [RW15], but has a more efficient decryption. In part, to achieve this, we use the multi-use techniques by Agrawal and Chase [AC17b] (also used in their scheme in Construction 4.3). In particular, we introduce another map $\tau\colon [n_1] \to [m]$ that maps the rows associated with the same attributes to different integers, i.e., $m = \max_{j \in [n_1]} |\rho^{-1}(\rho(j))|$, and τ is injective on the sub-domain $\rho^{-1}(\rho(j)) \subseteq [n_1]$.

Construction 4.5: Decentralized large-universe CP-ABE from FDH

We define the GPES as follows.

- Param($\mathcal{U}$): Let $\{\mathcal{A}_l\}_{[n_{\text{aut}}]}$ be the authorities, and $n_\alpha = n_{\text{aut}}$ and $n_b = 2n_{\text{aut}} + |\mathcal{U}|$, $\boldsymbol{\alpha} = (\{\alpha_l\}_{l \in [n_{\text{aut}}]}$, and $\mathbf{b} = (\{b_l, b'_l\}_{l \in [n_{\text{aut}}]}, \{b_{\text{att}}\}_{\text{att} \in \mathcal{U}})$, where $\mathcal{U}$ denotes the universe. We set $\mathcal{F}(b_{\text{att}}) = 1$ for all att $\in \mathcal{U}$ (where the FDH expects att as input).

- EncKey$((\mathcal{S}, \tilde{\rho}_{\mathcal{S}}), p)$: Set $m_1 = |\tilde{\rho}_{\mathcal{S}}(\mathcal{S})| + 1$, $m_2 = 0$, $\mathbf{k} = (\{k_{1,l} = \alpha_l + r_{\text{GID}} b_l + r_l b'_l\}_{l \in \tilde{\rho}_{\mathcal{S}}(\mathcal{S})}, \{k_{2,\text{att}} = r_{\tilde{\rho}_{\mathcal{S}}(\text{att})} b_{\text{att}}\}_{\text{att} \in \mathcal{S}})$, and $\mathcal{F}(r_{\text{GID}}) = 2$ (where the FDH expects GID as input).

- EncCt$((\mathbf{A}, \rho, \tilde{\rho}, \tau), p)$: We set $w_1 = m + n_1$, $w_2 = n_2 - 1$, $w'_2 = n_2 - 1$, $c_M = \tilde{s}$, $\mathbf{c} = (\{c_{1,j} = \mu_j + s_j b_{\tilde{\rho}(j)}, c_{2,j} = s_j b'_{\tilde{\rho}(j)} + s'_{\tau(j)} b_{\rho(j)}\}_{j \in [n_1]})$ and $\mathbf{c}' = (\{c'_j = \lambda_j + \alpha_{\tilde{\rho}(j)} s_j\}_{j \in [n_1]})$, where $\lambda_j = A_{j,1}\tilde{s} + \sum_{k \in [2,n_2]} A_{j,k} \hat{v}_k$, $\mu_j = \sum_{k \in [2,n_2]} A_{j,k} \hat{v}'_k$ and $\mathbf{s} = (\{s_j\}_{[n_1]}, \{s'_l\}_{l \in [m]})$.

- Pair$((\mathbf{A}, \rho, \tilde{\rho}, \tau), (\mathcal{S}, \tilde{\rho}_{\mathcal{S}}), p)$: On input policy $\mathbb{A} = (\mathbf{A}, \rho, \tilde{\rho}, \tau)$, and set of attributes $(\mathcal{S}, \tilde{\rho}_{\mathcal{S}})$, if $\mathbb{A} \models \mathcal{S}$, then this algorithm determines $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ such that $\sum_{j \in \Upsilon} \varepsilon_j \lambda_j = \tilde{s}$ (Definition 2.5), and outputs the vector $\mathbf{e} = \sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}_j^{w_4}$, and two matrices

$$\mathbf{E} = -\textstyle\sum_{j \in \Upsilon} \varepsilon_j \left(\mathbf{1}^{w_1 \times m_3}_{(2,\tau(j)),(2,\rho(j))} + \mathbf{1}^{w_1 \times m_3}_{(1,j),(1,\tilde{\rho}(j))} \right) \text{ and } \overline{\mathbf{E}} = \textstyle\sum_{j \in \Upsilon} \varepsilon_j \left(\mathbf{1}^{w_3 \times m_1}_{(1,j),\text{GID}} + \mathbf{1}^{w_3 \times m_1}_{(2,j),\tilde{\rho}(j)} \right).$$

Lemma 4.6

The GPES in Construction 4.5 satisfies the special selective symbolic property.

Proof. Let $\mathfrak{C} \subseteq [n_{\text{aut}}]$ be a set of corrupted authorities, and $d_1 = n_1$ and $d_2 = n_2$.

- EncB$((\mathbf{A}, \rho, \tilde{\rho}, \tau), \mathfrak{a}, \mathfrak{b}) \rightarrow (\{\mathbf{a}_l, \mathbf{B}_l, \mathbf{B}'_l\}_{l \in [n_{\text{aut}}]}, \{\mathbf{B}_{\text{att}}\}_{\text{att} \in \mathcal{U}})$, where $\mathbf{a}_l = \mathbf{0}^{d_1}$ and $\mathbf{B}_l, \mathbf{B}'_l, \mathbf{B}_{\text{att}} = \mathbf{0}^{d_1 \times d_2}$ for all $l \in \mathfrak{C}$ and att $\notin \rho([n_1])$, and let $\mathbf{v} \in \mathbb{Z}_p^{n_2}$ (with $v_1 = 1$) be the vector orthogonal to each row $j \in \tilde{\rho}^{-1}(\mathfrak{C})$ associated with a corrupted authority. For all $l \in [n_{\text{aut}}] \setminus \mathfrak{C}$, we set:

$$\mathbf{a}_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [n_2]} A_{j,k} v_k \mathbf{1}_j^{d_1}, \quad \mathbf{B}_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [2,n_2]} A_{j,k} (\mathbf{1}_{j,k}^{d_1 \times d_2} + v_k \mathbf{1}_{j,1}^{d_1 \times d_2}),$$

$$\mathbf{B}'_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [n_2]} A_{j,k} \mathbf{1}_{j,k}^{d_1 \times d_2}, \{\mathbf{B}_{\text{att}} = \sum_{j \in \rho^{-1}(\text{att}), k \in [n_2]} A_{j,k} \mathbf{1}_{\tau(j),k}^{d_1 \times d_2}\}_{\text{att} \in \rho([n_1])}.$$

- $\mathrm{EncR}((\mathbf{A},\rho,\tilde{\rho},\tau),(\mathcal{S},\tilde{\rho}_{\mathcal{S}}),\mathfrak{a},\mathfrak{b}) \to (\mathbf{r}_{\mathrm{GID}},\{\mathbf{r}_l\}_{l\in\tilde{\rho}_{\mathcal{S}}(\mathcal{S})})$, where

$$\mathbf{r}_{\mathrm{GID}} = -\overline{\mathbf{1}}_1^{d_2} + \sum_{k\in[2,n_2]} w_k \overline{\mathbf{1}}_k^{d_2}, \qquad \mathbf{r}_l = -\sum_{k\in[n_2]} w_k \overline{\mathbf{1}}_k^{d_2}.$$

- $\mathrm{EncS}((\mathbf{A},\rho,\tilde{\rho},\tau),\mathfrak{a},\mathfrak{b}) \to (\{\mathbf{s}_j\}_{j\in[n_1]},\{\mathbf{s}'_l\}_{l\in[m]},\{\hat{\mathbf{v}}_k,\hat{\mathbf{v}}'_k\}_{k\in[2,n_2]},\tilde{\mathbf{s}})$, where

$$\tilde{\mathbf{s}} = 1, \qquad \mathbf{s}_j = -\mathbf{1}_j^{d_1}, \qquad \mathbf{s}'_l = \mathbf{1}_l^{d_1}, \qquad \hat{\mathbf{v}}_k = v_k, \qquad \hat{\mathbf{v}}'_k = \overline{\mathbf{1}}_k^{d_2} + v_k\overline{\mathbf{1}}_1^{d_2}.$$

Note that, instead of applying the zero-out lemma to simulate $e(g,h)^{c'_j}$ for all $j \in [n_1]$, we introduce another vector $\mathbf{v}$ that is orthogonal to all rows associated with corrupted authorities, which we embed in $\hat{\mathbf{v}}$. □

4.7.2 Decentralized CP-ABE supporting OT-type negations

We also give a decentralized large-universe CP-ABE scheme that supports OT-type negations. Roughly, it is a decentralized variant of the TKN20 [TKN20] scheme. For this scheme, we define an additional function ρ' that maps the rows of the policy matrix to 1 if the attribute in the policy is not negated and to 2 if it is negated, a function ρ_{lab} that maps the rows of the policy matrix to the label universe, and $\tau\colon [n_1] \to [m]$ is a function that maps each row to associated with the same label to different integers, i.e., $m = \max_{j\in[n_1]} |\rho_{\mathrm{lab}}^{-1}(\rho_{\mathrm{lab}}(j))|$, and τ is injective on the subdomain $\rho_{\mathrm{lab}}^{-1}(\rho_{\mathrm{lab}}(j)) \subseteq [n_1]$.

Construction 4.6: Decentralized CP-ABE with OT-type negations

We define the GPES as follows.

- $\mathrm{Param}(\mathcal{L})$: Let $\{\mathcal{A}_l\}_{[n_{\mathrm{aut}}]}$ be the authorities. On input the label universe $\mathcal{L}$, we set $n_\alpha = n_{\mathrm{aut}}$ and $n_b = (1+2|\mathcal{L}|)n_{\mathrm{aut}}$, where $\boldsymbol{\alpha} = \{\alpha_l\}_{l\in[n_{\mathrm{aut}}]}$, and $\mathbf{b} = (\{b,\{b_{l,\mathrm{lab},0},b_{l,\mathrm{lab},1}\}_{\mathrm{lab}\in\mathcal{L}}\}_{l\in[n_{\mathrm{aut}}]})$. We also set $\mathcal{F}(b_{l,\mathrm{lab},i}) = 2l+i$ for all $l \in [n_{\mathrm{aut}}], i \in \{0,1\}, \mathrm{lab} \in \mathcal{L}$. (The FDH expects $\mathcal{A}_l$ and lab as input.)

- $\mathrm{EncKey}((\mathcal{S},\tilde{\rho}_{\mathcal{S}}),p)$: Assume that, for each $\mathrm{lab} \in \mathcal{L}$, there is at most one $\mathrm{att} \in \mathcal{U}$ such that $(\mathrm{lab},\mathrm{att}) \in \mathcal{S}$. We set $m_1 = |\tilde{\rho}_{\mathcal{S}}(\mathcal{S})| + 1$, $m_2 = 0$, and $\mathbf{k} = (\{k_{1,l} = \alpha_l + r_{\mathrm{GID}}b_l + r_l b'_l\}_{l\in\tilde{\rho}_{\mathcal{S}}(\mathcal{S})}, \{k_{2,(\mathrm{lab},\mathrm{att})} = r_{\tilde{\rho}_{\mathcal{S}}(\mathrm{att})}(b_{\tilde{\rho}_{\mathcal{S}}(\mathrm{att}),\mathrm{lab},0} + x_{\mathrm{att}}b_{\tilde{\rho}_{\mathcal{S}}(\mathrm{att}),\mathrm{lab},1})\}_{(\mathrm{lab},\mathrm{att})\in\mathcal{S}})$, where x_{att} is the representation of att in $\mathbb{Z}_p$.

- EncCt$((\mathbf{A}, \rho, \tilde{\rho}, \rho', \rho_{\text{lab}}, \tau), p)$: We set $w_1 = m + n_1$, $w_2 = n_2 - 1$, $w_2' = n_2 - 1$, $c_M = \tilde{s}$,

$$\mathbf{c} = (\{c_{1,j} = \mu_j + s_j b_{\tilde{\rho}(j)}\}_{j \in [n_1]},$$
$$\{c_{2,j} = s_j b'_{\tilde{\rho}(j)} + s'_{\tau(j)}(b_{\tilde{\rho}(j),\rho_{\text{lab}}(j),0} + x_{\rho(j)} b_{\tilde{\rho}(j),\rho_{\text{lab}}(j),1})\}_{j \in \Psi},$$
$$\{c_{2,j} = s_j b'_{\tilde{\rho}(j)} + s'_{\tau(j)} b_{\tilde{\rho}(j),\rho_{\text{lab}}(j),1}, c_{3,j} = s_{\tau(j)}(b_{\tilde{\rho}(j),\text{lab},0} + x_{\rho(j)} b_{\tilde{\rho}(j),\rho_{\text{lab}}(j),1})\}_{j \in \overline{\Psi}})$$

  and $\mathbf{c}' = (\{c_j' = \lambda_j + \alpha_{\tilde{\rho}(j)} s_j\}_{j \in [n_1]})$, where $\lambda_j = A_{j,1}\tilde{s} + \sum_{k \in [2,n_2]} A_{j,k}\hat{v}_k$, and $\Psi = \{j \in [n_1] \mid \rho'(j) = 1\}$ and $\overline{\Psi} = [n_1] \setminus \Psi$ (i.e., the set of rows associated with the non-negated and negated attributes, respectively), and $\mathbf{s} = (\{s_j\}_{[n_1]}, \{s_l'\}_{l \in [m]})$.

- Pair$((\mathbf{A}, \rho, \tilde{\rho}, \rho', \rho_{\text{lab}}, \tau), (\mathcal{S}, \tilde{\rho}_{\mathcal{S}}), p)$: If $(\mathbf{A}, \rho, \tilde{\rho}, \rho', \rho_{\text{lab}}, \tau) \models \mathcal{S}$, then this algorithm determines $\Upsilon = \{j \in \Psi \mid (\rho_{\text{lab}}(j), \rho(j)) \in \mathcal{S}\}$, $\overline{\Upsilon} = \{j \in \overline{\Psi} \mid (\rho_{\text{lab}}(j), \rho(j)) \notin \mathcal{S} \wedge \exists(\rho_{\text{lab}}(j), \text{att}) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon \cup \overline{\Upsilon}}$ so that $\sum_{j \in \Upsilon \cup \overline{\Upsilon}} \varepsilon_j \lambda_j = \tilde{s}$ (Definition 2.5), and outputs the vector $\mathbf{e} = \sum_{j \in \Upsilon \cup \overline{\Upsilon}} \varepsilon_j \mathbf{1}_j^{w_4}$ and matrices

$$\mathbf{E} = -\sum_{j \in \Upsilon \cup \overline{\Upsilon}} \varepsilon_j \mathbf{1}^{w_1 \times m_3}_{(1,j),(1,\tilde{\rho}(j))} - \sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}^{w_1 \times m_3}_{(2,\tau(j)),(2,\rho(j))}$$
$$-\sum_{j \in \overline{\Upsilon}} \frac{\varepsilon_j}{x_{\text{att}_j} - \rho(j)} \mathbf{1}^{w_1 \times m_3}_{(2,\tau(j)),(2,\rho(j))} \text{ and}$$
$$\overline{\mathbf{E}} = \sum_{j \in \Upsilon \cup \overline{\Upsilon}} \varepsilon_j \left(\mathbf{1}^{w_3 \times m_1}_{(1,j),\text{GID}} + \mathbf{1}^{w_3 \times m_1}_{(2,j),\tilde{\rho}(j)}\right) + \sum_{j \in \overline{\Upsilon}} \frac{\varepsilon_j}{x_{\text{att}_j} - \rho(j)} \mathbf{1}^{w_3 \times m_1}_{(3,j),\tilde{\rho}(j)},$$

  where att_j is such that $(\rho_{\text{lab}}(j), \text{att}_j) \in \mathcal{S}$.

Lemma 4.7

The GPES in Construction 4.6 satisfies the special selective symbolic property.

Proof. Let $\mathfrak{C} \subseteq [n_{\text{aut}}]$ be a set of corrupted authorities, and $d_1 = n_1$ and $d_2 = n_2 + n_1 n_2 |\rho_{\text{lab}}(n_1)|$. For simple notation of the column indices, we use $(1, k)$ and $(2, j, k, \text{lab})$ (for all $j \in [n_1], k \in [n_2], \text{lab} \in \rho_{\text{lab}}(n_1)$), which are mapped injectively in the interval $[d_2]$. We define EncB, EncR, EncS as follows:

- EncB$((\mathbf{A}, \rho, \rho', \tau), \mathfrak{a}, \mathfrak{b}) \to (\{\mathbf{a}_l, \mathbf{B}_l, \mathbf{B}_{l,\text{lab},0}, \mathbf{B}_{l,\text{lab},1}\}_{l \in [n_{\text{aut}}], \text{lab} \in \mathcal{L}})$, where where $\mathbf{a}_l = \mathbf{0}^{d_1}$ and $\mathbf{B}_l, \mathbf{B}_l' = \mathbf{0}^{d_1 \times d_2}$ for all $l \in \mathfrak{C}$, and let $\mathbf{v} \in \mathbb{Z}_p^{n_2}$ (with $v_1 = 1$) be the vector orthogonal to each row $j \in \tilde{\rho}^{-1}(\mathfrak{C})$ associated with a corrupted authority.

For all $l \in [n_{\mathrm{aut}}] \setminus \mathfrak{C}$, we set:

$$\mathbf{a}_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [n_2]} A_{j,k} v_k \mathbf{1}_j^{d_1}, \qquad \mathbf{B}_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [2, n_2]} A_{j,k} (\mathbf{1}_{j,(1,k)}^{d_1 \times d_2} + v_k \mathbf{1}_{j,(1,1)}^{d_1 \times d_2}),$$

$$\mathbf{B}'_l = \sum_{j \in \tilde{\rho}^{-1}(l), k \in [n_2]} A_{j,k} \mathbf{1}_{j,(1,k)}^{d_1 \times d_2},$$

$$\mathbf{B}_{l,\mathrm{lab},0} = \sum_{j \in \Psi_{l,\mathrm{lab}}, k \in [n_2]} A_{j,k} \left(\mathbf{1}_{\tau(j),(1,k)}^{d_1 \times d_2} - x_{\rho(j)} \mathbf{1}_{\tau(j),(2,j,k,\mathrm{lab})}^{d_1 \times d_2} \right) - \sum_{j \in \overline{\Psi}_{l,\mathrm{lab}}, k \in [n_2]} x_{\rho(j)} A_{j,k} \mathbf{1}_{\tau(j),(1,k)}^{d_1 \times d_2},$$

$$\mathbf{B}_{l,\mathrm{lab},1} = \sum_{j \in \Psi_{l,\mathrm{lab}}, k \in [n_2]} A_{j,k} \mathbf{1}_{\tau(j),(2,j,k,\mathrm{lab})}^{d_1 \times d_2} + \sum_{j \in \overline{\Psi}_{l,\mathrm{lab}}, k \in [n_2]} A_{j,k} \mathbf{1}_{\tau(j),(1,k)}^{d_1 \times d_2}$$

where $\Psi_{l,\mathrm{lab}} = \{j \in [n_1] \mid \tilde{\rho}(j) = l \wedge \rho_{\mathrm{lab}}(j) = \mathrm{lab} \wedge \rho'(j) = 1\}$ and $\overline{\Psi}_{l,\mathrm{lab}} = \{j \in [n_1] \mid \tilde{\rho}(j) = l \wedge \rho_{\mathrm{lab}}(j) = \mathrm{lab} \wedge \rho'(j) = 0\}$.

- $\mathrm{EncR}((\mathbf{A}, \rho, \rho', \tau), \mathcal{S}, \mathfrak{a}, \mathfrak{b}) \rightarrow (\mathbf{r}_{\mathrm{GID}}, \{\mathbf{r}_l\}_{l \in \tilde{\rho}_{\mathcal{S}}(\mathcal{S})})$: Let $\mathbf{w} \in (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ be such that $\mathbf{A}_j \mathbf{w}^\intercal = 0$ for all $j \in [n_1]$ with either $(\rho_{\mathrm{lab}}(j), \rho(j)) \in \mathcal{S}$ if $\rho'(j) = 1$ or $(\rho_{\mathrm{lab}}(j), \mathrm{att}) \in \mathcal{S}$ with $\mathrm{att} \neq \rho(j)$ if $\rho'(j) = 0$ (Definition 2.5). Then, set $\mathbf{r}_{\mathrm{GID}} = -\overline{\mathbf{1}}_1^{d_2} + \sum_{k \in [2, n_2]} w_k \overline{\mathbf{1}}_k^{d_2}$ and

$$\mathbf{r}_l = \sum_{k \in [n_2]} w_k \overline{\mathbf{1}}_{(1,k)}^{d_2} + \sum_{j \in \Psi_l \cap \overline{\Upsilon}, k \in [n_2], (\rho_{\mathrm{lab}}(j), \mathrm{att}) \in \mathcal{S}} \frac{w_k}{x_{\rho(j)} - x_{\mathrm{att}}} \overline{\mathbf{1}}_{(2,j,k,\mathrm{lab})}^{d_2},$$

where $\Psi_l = \{j \in \tilde{\rho}^{-1}(l) \mid \rho'(j) = 1\}$ and $\overline{\Upsilon} = \{j \in [n_1] \mid (\rho_{\mathrm{lab}}(j), \rho(j)) \notin \mathcal{S}\}$.

- $\mathrm{EncS}((\mathbf{A}, \rho, \rho', \tau), \mathfrak{a}, \mathfrak{b}) \rightarrow (\{\mathbf{s}_j\}_{j \in [n_1]}, \{\mathbf{s}'_l\}_{l \in [m]}, \{\hat{\mathbf{v}}_k, \hat{\mathbf{v}}'_k\}_{k \in [2, n_2]}, \tilde{\mathbf{s}})$, where

$$\tilde{\mathbf{s}} = 1, \quad \mathbf{s}'_l = -\mathbf{1}_l^{d_1}, \quad \mathbf{s}_j = \mathbf{1}_j^{d_1}, \quad \hat{\mathbf{v}}_k = v_k, \quad \hat{\mathbf{v}}'_k = \overline{\mathbf{1}}_{(1,k)}^{d_2} + v_k \overline{\mathbf{1}}_{(1,1)}^{d_2}. \qquad \square$$

Remark 4.3

This is the first decentralized large-universe CP-ABE scheme that simultaenously supports any type of negations and that is almost completely unbounded. (The only aspect in which it is bounded is the number of re-uses of a single label in the keys.) In contrast, the only other decentralized scheme that supports negations is the scheme by Okamoto and Takashima [OT13], which also supports OT-type negations and is even fully secure, but is bounded in the label universe and the number of label re-uses in both the keys and ciphertexts.

4.7.3 Large-universe variants of Wat11 and AC17

With our compiler, we can easily transform the PESs of Wat11 (Construction 4.1) and AC17 (Construction 4.3) to the large-universe setting, i.e., by setting $\mathcal{F}(b_{\text{att}}) = 1$ for each att $\in \mathcal{U}$ (and setting $\mathcal{F}$ to 0 for all other encodings) (Definition 4.7). Note that our large-universe variant of the PES for Wat11 is implied by the fourth scheme in [Wat11, Wat08, §A] (using random oracles), which is why we name it Wat11-IV instead of Wat11-LU.

Construction 4.7: The PES for Wat11-IV

The PES for the large-universe variant of Wat11 (Wat11-IV) is defined almost the same as Construction 4.1), except that the Param function also sets $\mathcal{F}(b_{\text{att}}) = 1$ for each att $\in \mathcal{U}$ (where the FDH expects att as input).

Construction 4.8: The PES for AC17-LU

The PES for the large-universe variant of AC17 (AC17-LU) is defined almost the same as Construction 4.3), except that the Param function also sets $\mathcal{F}(b_{\text{att}}) = 1$ for each att $\in \mathcal{U}$ (where the FDH expects att as input).

Note that it follows readily from the small-universe counterparts of these schemes that the large-universe instantiations with our compiler are selectively secure.

4.7.4 Schemes with an attribute-wise key generation

Lastly, we consider whether the large-universe schemes Wat11-IV (Construction 4.7), RW13 (Construction 4.2) and AC17-LU (Construction 4.8) can support an attribute-wise key generation (Direction 2.10). For some schemes, this can be achieved easily by generating the user-specific random r with an FDH (Section 2.9.3), and instantiating the scheme with the multi-authority compiler (where the number of authorities is 1). (Note that we subsequently also acquire schemes that are proven statically secure and not selectively secure.) For RW13, this indeed works: we can simply generate r with the FDH. Because none of the other variables are generated with an FDH, the correctness properties in Definition 4.8 are preserved. However, for Wat11-IV and AC17-LU, this cannot be done as easily, because b_{att} is already generated by an FDH. Because the user-specific random r occurs together with b_{att} in a product, they cannot be both generated with an FDH. Hence, the correctness property cannot be preserved. To attain an attribute-wise key generation for these schemes, we require a more intricate approach, possibly closer to the decentralized scheme from FDH in Construction 4.5. By slightly adapting that scheme (i.e., by introducing a fresh random integer r_l for each subset of $\mathcal{S}$ for which a key is requested) and by setting the number of authorities to 1, we can obtain a single-authority scheme with an attribute-wise key generation that is somewhat similar to AC17-LU. Unfortunately, this variant of the scheme incurs considerably higher encryption and decryption costs than AC17-LU. Possibly, by combining the designs, a more efficient variant may be obtained (which we leave for future work).

4.8 Future work

Our new compiler gives room for further improvements in the simplified design of practical PE schemes. Most obviously, it could be investigated whether the approaches used for our compiler also carry over to full-security compilers. Furthermore, since our new complexity assumption is structurally closer to the DBDH assumption, it would be valuable to investigate whether it can be reduced to DBDH and other well-studied non-parametrized assumptions such as the symmetric external Diffie-Hellman assumption. Lastly, our decentralized schemes could be used as inspiration for generic constructions of decentralized schemes, similarly as in the single-authority setting [Att19]. In this way, we can efficiently achieve certain complex properties such as non-monotonicity [Amb21] in decentralized ABE.

4.9 Conclusion

We have reviewed the current state of the pair encodings framework, and we have introduced a new practical compiler for PE and ABE, which uses the symbolic property to simplify the security proofs. In general, we have shown that the pair encodings framework is very powerful, and captures many schemes. Furthermore, it allows us to create larger schemes with additional functionality (such as revocation mechanisms (Section 2.10.2)) without requiring additional (complicated!) security proofs. We have also proposed a new compiler that benefits from the simplicity of the symbolic property. Although in contrast to existing full-security compilers [Att14a, Att16, AC16, AC17b], ours proves selective security generically, it supports full-domain hashes, flexible instantiations in the pairing-friendly groups and multi-authority extensions. These properties are widely considered attractive for practice. To illustrate the effectiveness of our compiler, we have given several new CP-ABE schemes—including the first decentralized large-universe CP-ABE scheme that supports any type of negations and that is almost completely unbounded—whose proofs are much less sizable and arguably simpler to verify than the security proofs of similar schemes [RW15, TKN20].

Chapter 5

A bunch of broken schemes: a simple yet powerful linear approach to analyzing security of ABE

Verifying security of advanced cryptographic primitives such as attribute-based encryption (ABE) is often difficult. In this chapter, we show how to break eleven schemes: two single-authority and nine multi-authority ABE (MA-ABE) schemes. Notably, we break DAC-MACS, a highly-cited multi-authority scheme, published at TIFS. The fact that its security issues have not been noticed for so long suggests that verifying security of complex schemes is complicated, and may require simpler tools. The multi-authority attacks also illustrate that mistakes are made in transforming single-authority schemes into multi-authority ones. To simplify the verification of security, we systematize our methods to a linear approach to analyzing generic security of ABE. Our approach is not only useful in analyzing existing schemes, but can also be applied during the design and reviewing of new schemes. As such, it can prevent the employment of insecure (MA-)ABE schemes in the future.

5.1 Introduction

Proving and verifying security of new schemes are difficult, and, perhaps unsurprisingly, several schemes turn out to be broken. Some schemes were shown to be generically broken with respect to the basic functionality, and are therefore insecure. Others were only broken with respect to additional functionality. Table 5.1 shows that many of these schemes have been published at venues that include cryptography in their scope. This suggests that, even for cryptographers, it is difficult to verify security of ABE. In addition, many of these schemes are highly cited due to their focus on practical applications. This popularity shows that the claimed properties of these schemes are high in demand. It is thus important to simplify the security analysis.

Table 5.1. Attacks on existing schemes. For each scheme, we list in which work it was broken, which functionality was attacked, and whether it was later fixed. Also, we provide the venue and number of citations for these schemes according to Google Scholar. These results were retrieved on 18 November 2020.

Scheme	Broken in	Attacked functionality	Fixed?	Venue	Cit.
[LRZW09] [ZCL+13] [XFZ+14]	[LHC+11] [CDM15]	Private access policies	[LHC+11]	ISC AsiaCCS NC	203 104 46
[HSMY12]	[GZZ+13]	Basic	U	NC	176
[YJR+13]	[HXL15] [WJB17]	Revocation	[WJB17]	TIFS	474
[HSM+14] [HSM+15]	[WZC15]	Basic	U	ESORICS TIFS	30 128
[JLWW15]	[MZY16]	Distributed key generation	[JLWW16]	TIFS	161

NC = non-crypto venue/journal; U = unknown

To simplify the design and analysis of complex primitives such as ABE, the pair encodings framework provides a powerful tool, also because it considers the common structure of many schemes. Furthermore, Agrawal and Chase [AC17b] show that fully secure schemes can be constructed from pair encodings that are provably symbolically secure. Using this, they show that any scheme that is not trivially broken implies a fully secure scheme. Later, Ambrona et al. [ABGW17] expand their framework to a broader class of schemes, and devise automated tools to prove symbolic security, subsequently yielding provably secure schemes in the GGM. However, operating these tools still requires a considerable expertise (and in a different field). Additionally, these frameworks do not support practical extensions of ABE such as MA-ABE.

In any case, these works illustrate that proving generic security of a scheme provides a meaningful first step in the analysis of a new scheme, and may even imply stronger notions of security. Conversely, showing that a scheme is *not* generically secure provides overwhelming evidence that a scheme is insecure, regardless of the underlying group structure or accompanying security proofs. As such, devising *manual* tools and heuristics to effectively analyze the generic (in)security of schemes may further contribute to these frameworks. That is, finding a generic attack—assuming that one exists—is often much simpler than finding a mistake in a security proof. In fact, it is often the first step that an experienced cryptographer takes when designing a new scheme.

5.1.1 Our contribution

We focus on simplifying the search for generic attacks (provided that they exist). In a broader context, our goal is not necessarily to attack existing schemes, but to propose a framework that simplifies the analysis—and by extension, design—of secure ABE schemes. We do this by systematizing a simple heuristic approach to finding attacks. Our contribution in this endeavor is twofold. First, we show that eleven schemes are vulnerable to generic attacks, rendering them (partially) insecure. Five of these are insecure in the basic security model. The other six are insecure in the multi-authority security model—which also allows for the corruption of one or more authorities—but are possibly secure if all authorities are assumed to be honest. Essentially, these six schemes provide a comparable level of security as single-authority schemes. Second, we systematize our methods to a linear approach to generic security analysis of ABE based on the common structure of many schemes. Similarly as the aforementioned frameworks, we consider the pair encodings of the schemes. To this end, we also formalize such pair encodings for multi-authority schemes. Furthermore, we describe three types of attacks, which model the implicit security requirements on the keys and ciphertexts, and simplify the search for generic attacks. They model whether the master key of the/an authority can be recovered, or whether users can collude and decrypt ciphertexts that they cannot individually decrypt. In the multi-authority setting, we also model the notion of corruption.

5.1.2 Technical details

A brief overview of the attack models. We propose three types of attacks, which all imply attacks on the security model for ABE. This model considers chosen-plaintext attacks (CPA) and collusion of users. Two of our attack models only consider the secret keys issued in the first key query phase of the security model, while the third model also considers the challenge ciphertext. Informally, the attacks are:

- **Master-key attack (MK):** The attacker can extract the KGA's master key, which can be used to decrypt any ciphertext.
- **Attribute-key attack (AK):** The attacker can generate a secret key for a set $\mathcal{S}'$ that is strictly larger than each set $\mathcal{S}_i$ associated with an issued key.
- **Decryption attack (D):** The attacker can decrypt a ciphertext for which no authorized key was generated.

In addition, we distinguish complete from conditional decryption attacks. Conditional attacks can only be performed when the collective set of attributes possessed by the colluding users satisfies the access structure. In contrast, complete attacks allow any ciphertext to be decrypted. Figure 5.1 illustrates the relationship between the attacks, and how the attacks relate to the security model. We consider the first key query phase and the challenge phase, which output the secret keys for a polynomial number of sets of attributes, and a ciphertext associated with an access structure such that all keys are unauthorized, respectively.

The security models in the multi-authority setting are similar, but include the notion of corruption. The attacker is allowed to corrupt one or more authorities in

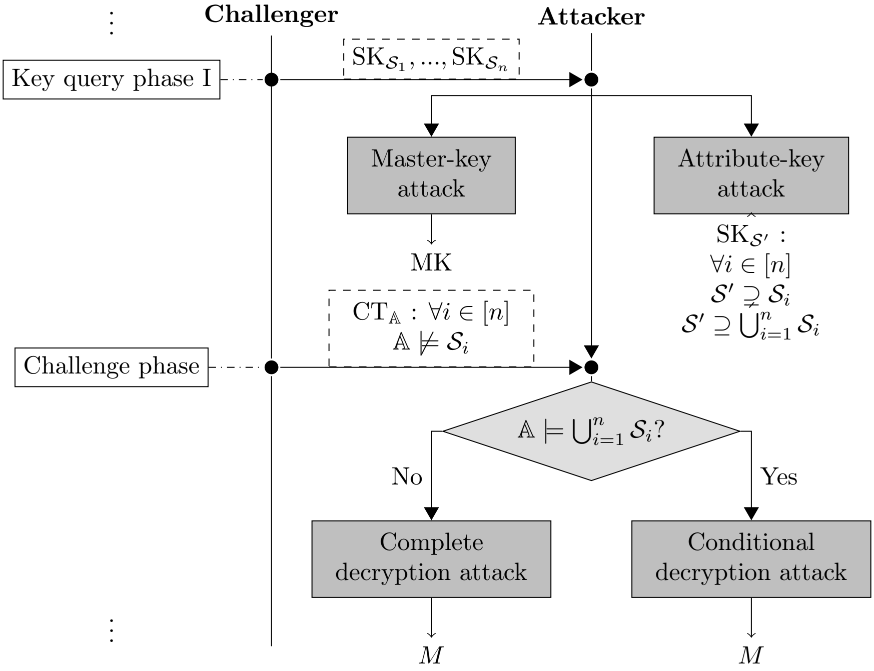

Challenger
Attacker
Key query phase I
$\mathrm{SK}_{\mathcal{S}_1}, ..., \mathrm{SK}_{\mathcal{S}_n}$
Master-key attack
Attribute-key attack
MK
$\mathrm{SK}_{\mathcal{S}'}$: $\forall i \in [n]$ $\mathcal{S}' \supsetneq \mathcal{S}_i$ $\mathcal{S}' \supseteq \bigcup_{i=1}^{n} \mathcal{S}_i$
$\mathrm{CT}_{\mathbb{A}}$: $\forall i \in [n]$ $\mathbb{A} \not\models \mathcal{S}_i$
Challenge phase
$\mathbb{A} \models \bigcup_{i=1}^{n} \mathcal{S}_i$?
No
Yes
Complete decryption attack
Conditional decryption attack
M
M

Figure 5.1. The general attacks and how they relate to one another.

an attack, which should not yield sufficient power to enable an attack against the honest authorities. Sometimes, schemes employ a central authority (CA) in addition to employing multiple attribute authorities. This CA is assumed to perform the algorithms as expected, though sometimes, it may be corruptable. In this work, we show how to model the corruption of attribute authorities and corruptable CAs, and how the additional knowledge (e.g., the master secret keys) gained from corrupting an authority can be included in the attacks.

Finally, we observe that sometimes it is unclear whether a multi-authority scheme is supposed to provide security against corruption. Initially, multi-authority ABE was designed to be secure against corruption [Cha07, LW11a]. Not only does this protect honest authorities from corrupt authorities, but it also increases security from the perspective of the users. Conversely, not allowing corruption in the security model provides a comparable level of security as single-authority ABE. In some cases, the informal description of a scheme is ambiguous on whether it protects against corruption. For instance, schemes are compared with other multi-authority schemes that are secure against corruption, while the proposed scheme is not, even though this is not explicitly mentioned [PO17, MGZ19].

Table 5.2. The schemes for which we provide attacks. For each scheme, we indicate on which scheme it is based, which type of attack we apply to it and whether it is complete, whether it uses collusion or corruption, whether the attack explicitly contradicts the model in which the scheme is claimed to be secure. We also list the conference or journal in which the scheme was published and how many times the paper is cited according to Google Scholar. These results were retrieved on 18 November 2020.

	Scheme	Based on	CD	Att.	Col.	Cor.	Con.	Venue	Cit.
	[ZH10b, ZH10a] [ZHW15]	-	✗	AK	2	-	✓	NC NC	112 123
	[NDCW15]	[Wat11]	✓	D	-	-	✓	ESORICS	46
Multi-authority ABE	[YJ12]	-	✓	MK	-	$\mathcal{A}$	✓	NC	155
	[YJRZ13, YJR$^+$13] [WJB17]	-	✓	D	-	-	✓	NC, TIFS NC	474 28
	[JLWW13] [JLWW15]	[BSW07]	✗	AK	2	-	✓	NC TIFS	174 161
	[QLZ13]	-	✓	MK	-	-	✓	ICICS	42
	[YJ14]	-	✓	D	-	$\mathcal{A}$	✓	NC	240
	[CM14]	-	✓	D	-	$\mathcal{A}$	U	NC	42
	[LXXH16] [MST17]	[Wat11]	✓	MK	-	CA	✓ U	NC AsiaCCS	110 25
	[PO17]	-	✓	D	-	$\mathcal{A}$	U	SACMAT	16
	[MGZ19, §3.2]	[LW11a]	✓	MK	-	CA	U	Inscrypt	4

CD = complete decryption attack, Att = attack, MK = master-key attack,
AK = attribute-key attack, D = decryption attack; Col = collusion,
Cor = corruption, Con = contradicts proposed security model, U = unclear,
NC = not published at peer-reviewed crypto venue/journal,
CA = central authority, $\mathcal{A}$ = key generation authority

Finding attacks, generically. We evaluate the generic (in)security of a scheme by considering the pair encodings of a scheme (Chapter 4). On a high level, generic security of a scheme is evaluated by considering whether αs can be retrieved from a ciphertext encoding $\mathbf{c}(\mathbf{s}, \mathbf{b})$ and an unauthorized key encoding $\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b})$. By multiplying the entries of $\mathbf{k}$ and $\mathbf{c}$, we emulate the pairing operations made in the scheme. By linearly combining the resulting values (for which we require additions), we emulate the other available group operations. As a result, such a "combination" of a key and ciphertext encoding can be denoted by a matrix multiplication, i.e., $\mathbf{E}$ for which $\mathbf{k}\mathbf{E}\mathbf{c}^\intercal = \alpha s$. Pair encoding schemes allow us to evaluate the generic security of any scheme that satisfies this structure, regardless of the underlying group structure. Unfortunately, the structure of most multi-authority schemes differs from this structure. Therefore, we extend the existing definitions to additionally support these multi-authority schemes. Furthermore, we split the key and ciphertext encodings into two parts, so we can separately evaluate the stronger attacks, i.e., master-key and complete decryption attacks, and the weaker attacks, i.e., attribute-key and conditional decryption attacks. This further simplifies the analysis of schemes.

The attacked schemes. Table 5.2 lists the schemes for which we have found attacks (see the paper [VA21, VA20] for the attacks). Many of these schemes are published at venues that include cryptography in their scope, or have been highly cited. Hence, even though many researchers have studied these schemes, mistakes in the security proofs have gone unnoticed. These attacks also illustrate that systematizing any generic attacks may actually have merit. Not only does it provide designers with simple tools to test their own schemes with respect to generic attacks, but also reviewers and practitioners. Because most schemes are broken with respect to the strongest attacks, i.e., master-key and complete decryption attacks, formalizing these models—which are stronger but easier to verify—simplifies the search for generic attacks.

5.1.3 Definition of (multi-authority) ciphertext-policy ABE

We slightly adjust the more traditional definition of CP-ABE (Definition 3.3) and its multi-authority variant [LW11a]. Specifically, we split the generation of the keys into two parts: the part that is dependent on an attribute and the part that is not. These are relevant distinctions in the definitions of various attack models.

Definition 5.1: More fine-grained definition of ciphertext-policy ABE

A CP-ABE scheme with some authorities $\mathcal{A}_1, ..., \mathcal{A}_n$ (where $n \in \mathbb{N}$)—such that each $\mathcal{A}_i$ manages universe $\mathcal{U}_i$—users and a universe of attributes $\mathcal{U} = \bigcup_{i=1}^{n} \mathcal{U}_i$ consists of the following algorithms.

- $\mathsf{GlobalSetup}(\lambda) \rightarrow \mathrm{GP}$: The global setup is a randomized algorithm that takes as input the security parameter λ, and outputs the public global system parameters GP (independent of any attributes).
- $\mathsf{MKSetup}(\mathrm{GP}) \rightarrow (\mathrm{GP}, \mathrm{MK})$: The master-key setup is a randomized algorithm that takes as input the global parameters GP, and outputs the (secret) master key MK (independent of any attributes) and updates the global parameters by adding the public key associated with MK.
- $\mathsf{AttSetup}(\mathrm{MK}, \mathrm{GP}, \mathrm{att}) \rightarrow (\mathrm{MSK}_{\mathrm{att}}, \mathrm{MPK}_{\mathrm{att}})$: The attribute-key setup is a randomized algorithm that takes as input possibly the master key and the global parameters and an attribute, and outputs a master secret $\mathrm{MSK}_{\mathrm{att}}$ and public key $\mathrm{MPK}_{\mathrm{att}}$ associated with attribute att.
- $\mathsf{UKeyGen}(\mathrm{MK}, \mathrm{GP}, \mathrm{id}) \rightarrow \mathrm{SK}_{\mathrm{id}}$: The user-key generation is a randomized algorithm that takes as input the master key MK, the global parameters GP and the identifier id, and outputs the secret key $\mathrm{SK}_{\mathrm{id}}$ for id.

- $\mathsf{AttKeyGen}(\mathsf{GP}, \mathsf{MK}, \mathsf{SK}_{\mathrm{id}}, \{\mathsf{MSK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}, \mathcal{S}) \rightarrow \mathsf{SK}_{\mathrm{id},\mathcal{S}}$: The attribute-key generation is a randomized algorithm that takes as input set of attributes $\mathcal{S}$ possessed by some user with identifier id, and the global parameters GP, the master key MK, the secret key $\mathrm{SK}_{\mathrm{id}}$ and master secret key $\mathrm{MSK}_{\mathrm{att}}$, and outputs a user-specific secret key $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{\mathrm{SK}_{\mathrm{id},\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}$.

- $\mathsf{Encrypt}(\mathsf{GP}, \{\mathsf{MPK}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}}, \mathbb{A}, M) \rightarrow \mathsf{CT}_{\mathbb{A}}$: This randomized algorithm is run by any encrypting user and takes as input a message M, access structure $\mathbb{A}$ and the relevant public keys. It outputs the ciphertext $\mathrm{CT}_{\mathbb{A}}$.

- $\mathsf{Decrypt}(\mathsf{SK}_{\mathrm{id},\mathcal{S}}, \mathsf{CT}_{\mathbb{A}}) \rightarrow M$: This deterministic algorithm takes as input a ciphertext $\mathrm{CT}_{\mathbb{A}}$ and secret key $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id},\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}$ associated with a set $\mathcal{S}$. It outputs plaintext message M if $\mathcal{S}$ is authorized. Otherwise, it aborts.

- $\mathsf{MKDecrypt}(\mathsf{MK}, \mathsf{CT}) \rightarrow M$: This deterministic algorithm takes as input a ciphertext CT and the master key MK, and outputs message M.

The scheme is called correct if decryption outputs the correct message for a secret key associated with a set of attributes that satisfies the access structure.

In the single-authority setting (i.e., where $n = 1$), the GlobalSetup, MKSetup and AttSetup are described in one Setup, and the UKeyGen and AttKeyGen have to be run in one KeyGen. In the multi-authority setting (i.e., where $n > 1$), the GlobalSetup is run either jointly or by some central authority. MKSetup can either be run distributively or independently by each $\mathcal{A}_i$. AttSetup can be run distributively or individually by $\mathcal{A}_i$ for the managed attributes $\mathcal{U}_i$. UKeyGen is run either distributively, individually for each $\mathcal{A}_i$, or implicitly (e.g., by using a hash). AttKeyGen is run by the $\mathcal{A}_i$ managing the set of attributes.

5.1.4 The security model and our attack models

The associated security definition for our fine-grained definition of CP-ABE (Definition 5.1) is similar to the security model for multi-authority PE (Section 3.3.4) (see [VA21] for a formal treatment). The only difference is that, instead of running the AuthoritySetup, the MKSetup and AttSetup are run, and instead of running KeyGen, the UKeyGen and AttKeyGen are run. We formally define our attack models in line with Figure 5.1, such that CPA-security also implies security against these attacks. Conversely, the ability to find such attacks implies insecurity in this model.

Definition 5.2: Master-key attacks (MKA)

We define the game between challenger and attacker as follows. First, the initialization, setup and first key query phases are run as in the full security model for multi-authority PE (Section 3.3.4). Then:

- **Decision phase:** The attacker outputs MK'.

The attacker wins the game if for all messages M, decryption of ciphertext $\mathrm{CT} \leftarrow \mathrm{Encrypt}(..., M)$ yields $M' \leftarrow \mathrm{MKDecrypt}(\mathrm{MK}', \mathrm{CT})$ such that $M = M'$.

Definition 5.3: Attribute-key attacks (AKA)

We define the game between challenger and attacker as follows. First, the initialization, setup and first key query phases are run as in the full security model for multi-authority PE (Section 3.3.4). Then:

- **Decision phase:** The attacker outputs $\mathrm{SK}_{\mathcal{S}'}$, where $\mathcal{S}' \supsetneq \mathcal{S}_j$ for all $j \in [n]$.

The attacker wins the game if $\mathrm{SK}_{\mathrm{id}',\mathcal{S}'}$ is a valid secret key for some arbitrary identifier id' and set $\mathcal{S}'$.

Definition 5.4: Decryption attacks (DA)

We define the game between challenger and attacker as follows. First, the initialization, setup, first key query and challenge phases are run as in the full security model for multi-authority PE (Section 3.3.4). Then:

- **Decision phase:** The attacker outputs message M'.

The attacker wins the game if $M' = M$. A decryption attack is *conditional* if $\mathbb{A} \models \bigcup_{j=1}^{n} \mathcal{S}_j$. Otherwise, it is *complete*.

5.2 Warm-up: attacking DAC-MACS

We first give an example of how an attack can be found effectively by attacking the YJR+13 [YJRZ13, YJR$^+$13] scheme, also known as DAC-MACS. DAC-MACS is a popular multi-authority scheme that supports key revocation. This functionality was already broken in [HXL15, WJB17], but a fix for its revocation functionality was proposed in [WJB17]. We show that even the basic scheme—which matches the "fixed version" [WJB17]—is vulnerable to a complete decryption attack. We review a stripped-down version of the global and master-key setups, the user-key generation and encryption. In particular, we consider only the parts that are not dependent on any attributes. Also note that we use a slightly different notation for the variables: $(a, \alpha_k, \beta_k, z_j, u_j, t_{j,k}) \mapsto (b, \alpha_i, b_i, x_1, x_2, r_i)$.

- GlobalSetup: The central authority generates a pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ over groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with generator $g \in \mathbb{G}$, chooses a random integer $b \in_R \mathbb{Z}_p$ and publishes the global parameters $\mathrm{GP} = (p, e, \mathbb{G}, \mathbb{G}_T, g, g^b)$;
- MKSetup: Authority $\mathcal{A}_i$ chooses random $\alpha_i, b_i \in_R \mathbb{Z}_p$, and outputs the master secret key $\mathrm{MSK}_i = (\alpha_i, b_i)$ and the master public key $\mathrm{MPK}_i = (e(g,g)^{\alpha_i}, g^{1/b_i})$;
- UKeyGen: Upon registration, the user receives partial secret key $\mathrm{SK} = (x_1, g^{x_2})$ from the central authority, with a certificate that additionally includes x_2. To request a key from authority $\mathcal{A}_i$, the user sends this certificate. The attribute-independent part of a user's secret key provided by authority $\mathcal{A}_i$ is $\mathrm{SK}'_i = (g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}, g^{r_i b_i/x_1}, g^{r_i b})$, where $r_i \in_R \mathbb{Z}_p$;
- Encrypt: A message M is encrypted by picking random $s \in_R \mathbb{Z}_p$ and computing: $\mathrm{CT} = (M \cdot \left(\prod_i e(g,g)^{\alpha_i}\right)^s, g^s, g^{s/b_i}, ...)$.

Note that an authority $\mathcal{A}_i$ can individually generate $g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}$, if x_2 is known to the authority. In the specification of DAC-MACS, the central authority generates a certificate containing x_2 and the identifier of the user, such that these are linked. In the conference version [YJRZ13], this certificate is encrypted, and can be decrypted only by the authorities. However, in the journal version [YJR$^+$13], this certificate is not explicitly defined to be hidden from the user. We assume that x_2 is therefore also known to the user. Then, after receiving the certificate from the user, x_2 is used by the authority $\mathcal{A}_i$ to link the secret keys to this particular user. However, we show that knowing exponents x_1, x_2 enables an attack. That is, any decrypting user is trivially able to decrypt any ciphertext, without even needing to consider the attribute-dependent part of the keys and ciphertexts. First, we show that we cannot perform a master-key attack, i.e., retrieve α_i. In particular, the partial secret keys are of the form $\mathrm{SK} = (x_1, g^{x_2}, x_2, g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}, g^{r_i b_i/x_1}, g^{r_i b})$. We observe that master key α_i only occurs in $g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}$. Now, we can cancel out $g^{x_2 b}$, because x_2 is known and g^b is a global parameter. Unfortunately, we cannot cancel out $g^{r_i b/b_i}$.

Subsequently, we show that it is possible to perform a decryption attack. For this, we also consider $\mathrm{CT} = (M \cdot e(g,g)^{\alpha_i s}, g^s, g^{s/b_i}, ...)$. To retrieve $e(g,g)^{\alpha_i s}$, we start by pairing $g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}$ and g^s, and compute

$$e(g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}, g^s)^{x_1} = \underbrace{e(g,g)^{\alpha_i s}}_{\substack{\uparrow \\ \text{blinding value}}} + \overbrace{e(g,g)^{\underset{\substack{\uparrow \\ e(g^b,g^s)^{x_1 x_2}}}{x_1 x_2 s b} + \underset{\substack{\uparrow \\ e(g^{r_i b}, g^{s/b_i})^{x_1}}}{x_1 r_i s b/b_i}}}^{\text{to cancel}}$$

Hence, $e(g,g)^{\alpha_i s}$ can be retrieved and thus the ciphertext can be decrypted. Resisting this attack is not trivial. The main issue is that x_2 is known to the user, because x_2 needs to be known by the authority to generate $g^{\alpha_i/x_1 + x_2 b + r_i b/b_i}$. Otherwise, it cannot generate $g^{x_2 b}$. To avoid the attack, the CA could encrypt the certificate containing x_2—like in the conference version [YJRZ13]—so only the authorities $\mathcal{A}_i$ can decrypt it, and the user does not learn x_2. The attacker can however corrupt any authority, learn x_2 and perform the attack. This still breaks the scheme, because of its claimed security against corruption of authorities $\mathcal{A}_i$.

This attack illustrates two things. First, it shows the simplicity of finding a master-key or complete decryption attack—the two strongest attacks—provided that one exists. In particular, in the analysis, we only have to consider the parts of the keys that are not related to the attributes or additional functionality. This strips away a significantly more complicated part of the scheme. Second, we can systematically focus on the the goal of retrieving g^{α_i} or $e(g,g)^{\alpha_i s}$. Due to the structure of the scheme, we can directly analyze the exponent space of the key and ciphertext components. The pairing operation effectively allows us to compute products of these values "in the exponent". Therefore, we do not have to consider the underlying group structure. Instead, we can attempt to retrieve $\alpha_i s$ by linearly combining the products of the exponent spaces of the key and ciphertext components. In addition, we can use the explicit knowledge of certain variables "in the exponent" by using these variables in the coefficients.

Not only is finding such a generic attack simpler than verifying a security proof, it may also help finding the mistake in the proof. As shown, the main reason that our attack works is that x_2 is known to the user. We use this observation to find the mistake in the security proof in the journal version [YJR$^+$13], which is loosely based on the selective security proof by Waters [Wat11]. In the proof, the challenger and attacker play the security game in Definition 2.4. The attacker is assumed to be able to break the scheme with non-negligible advantage. The challenger uses this to break the complexity assumption by using the inputs to the assumption in the simulation of the keys and challenge ciphertext. Roughly, the challenger embeds the element that needs to be distinguished from a random element in the complexity assumption in the challenge ciphertext component $e(g,g)^{\alpha_i s}$. To ensure that $e(g,g)^{\alpha_i s}$ cannot be generated trivially from e.g., g^{α_i} and g^s, the challenger cannot simulate the master secret key g^{α_i}. To simulate the key $g^{\alpha_i/x_1+x_2 b+r_i b/b_i}$, the part with g^{α_i} is canceled out by the $g^{x_2 b}$ part. By extension, the challenger cannot fully simulate $g^{x_2 b}$. Because g^b needs to be simulated (as it is part of the public key), it is not possible to simulate the secret in x_2. In [YJR$^+$13], the authors attempt to solve this issue by generating x_2 randomly, and by implicitly writing it as the sum of the non-simulatable secret and another random integer x_2' (which is thus unknown to the challenger). While this allows the simulation of x_2, this causes an issue in the simulation of $g^{\alpha_i/x_1+x_2 b}$. Because the secret part in x_2 is meant to cancel out the non-simulatable part, $g^{\alpha_i/x_1+x_2 b}$ needs to be simulated by computing $g^{x_2' b}$. This is not possible, since x_2' is unknown to the challenger.

5.3 Systematizing our methodology

Our methodology consists of a systematized approach to finding attacks. It consists of a more concise notation implied by the common structure of many ABE schemes (Section 5.3.1). We model how learning explicit values "in the exponent", e.g., by corrupting an authority, can be used in the attacks (Section 5.3.2). We give our attack models in the concise notation (Sections 5.3.3, 5.3.4). Finally, we describe a heuristic approach that simplifies the effort of finding attacks (Section 5.3.5).

5.3.1 The common structure implies a more concise notation

As mentioned in Chapter 4, many schemes have a similar structure, and can be captured in the pair encodings framework. We adapt Definition 4.1 to match our more fine-grained definition of CP-ABE (Definition 5.1).

Definition 5.5: Extended pair encoding implied by CP-ABE

Let authorities $\mathcal{A}_1,\ldots,\mathcal{A}_n$ manage universes $\mathcal{U}_i$ for each i, and set $\mathcal{U} = \bigcup_{i=1}^{n} \mathcal{U}_i$ as the collective universe.

- GlobalSetup(λ): This algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and a pairing $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It may also select *common variables* $\mathbf{b} \in_R \mathbb{Z}_p$. It publishes the global parameters

  $$\mathrm{GP} = (p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, g, h, \mathcal{U}, g^{\mathbf{gp}(\mathbf{b})}),$$

  where we refer to $\mathbf{gp}$ as the *global parameter encoding*.

- MKSetup(GP): This algorithm selects $\alpha \in_R \mathbb{Z}_p$, sets master key $\mathrm{MK} = \alpha$ and publishes master public key $\mathrm{MPK} = e(g,h)^{\alpha}$.

- AttSetup(MK, GP, att): This algorithm selects integers $\mathbf{b}_{\mathrm{att}} \in_R \mathbb{Z}_p$, sets as master secret keys $\mathbf{MSK}_{\mathrm{att}} = \mathbf{b}_{\mathrm{att}}$, and publishes

  $$\mathbf{MPK}_{\mathrm{att}} = g^{\mathbf{mpk}_a(\mathbf{b}_{\mathrm{att}}, \mathbf{b})},$$

  where we refer to $\mathbf{mpk}_a$ as the *master attribute-key encoding*.

- UKeyGen(MK, GP, id): This algorithm selects user-specific random integers $\mathbf{r}_u \in_R \mathbb{Z}_p$ and computes partial user-key

  $$\mathrm{SK}_{\mathrm{id}} = h^{\mathbf{k}_u(\mathrm{id}, \alpha, \mathbf{r}_u, \mathbf{b})},$$

  where we refer to $\mathbf{k}_u$ as the *user-key encoding*.

- AttKeyGen(GP, MK, $\mathrm{SK}_{\mathrm{id}}, \{\mathrm{MSK}_{\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}, \mathcal{S}$): Let $\mathrm{SK}_{\mathrm{id}} = (h_{\mathrm{id},1}, h_{\mathrm{id},2}, ...)$. This algorithm selects user-specific random integers $\mathbf{r}_a \in_R \mathbb{Z}_p$ and computes a key $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{\mathrm{SK}_{\mathrm{id},\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}}$, such that for all att $\in \mathcal{S}$

$$\mathrm{SK}_{\mathrm{id},\mathrm{att}} = (h_{\mathrm{id},1}^{\mathbf{k}_{a,1}(\mathrm{att},\mathbf{r}_a,\mathbf{b},\mathbf{b}_{\mathrm{att}})}, h_{\mathrm{id},2}^{\mathbf{k}_{a,2}(\mathrm{att},\mathbf{r}_a,\mathbf{b},\mathbf{b}_{\mathrm{att}})}, ...),$$

  where we refer to $\mathbf{k}_{a,i}$ as the *user-specific attribute-key encodings*.

- Encrypt(GP, $\{\mathrm{MPK}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}}, \mathbb{A}, M$): This algorithm picks ciphertext-specific randoms $\mathbf{s} = (s, s_1, s_2, ...) \in_R \mathbb{Z}_p$ and outputs the ciphertext

$$\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, M \cdot e(g,h)^{\alpha s}, g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})}),$$

  where we refer to $\mathbf{c}$ as the *attribute-independent ciphertext encoding*, and $\mathbf{c}_a$ the *attribute-dependent ciphertext encoding*.

- Decrypt(($\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id},\mathcal{S}}), \mathrm{CT}_{\mathbb{A}}$): Let $\mathrm{SK}_{\mathrm{id}} = h^{\mathbf{k}_u(\mathrm{id},\alpha,\mathbf{r}_u,\mathbf{b})} = (h_{\mathrm{id},1}, h_{\mathrm{id},2}, ...)$, $\mathrm{SK}_{\mathrm{id},\mathcal{S}} = \{(h_{\mathrm{id},1}^{\mathbf{k}_{a,1}(\mathrm{att},\mathbf{r}_{a,i},\mathbf{b},\mathbf{b}_{\mathrm{att}})}, h_{\mathrm{id},2}^{\mathbf{k}_{a,2}(\mathrm{att},\mathbf{r}_{a,i},\mathbf{b},\mathbf{b}_{\mathrm{att}})}, ...)\}_{i\in\{1,...,n\},\mathrm{att}\in\mathcal{S}\cap\mathcal{U}_i}$, and $\mathrm{CT}_{\mathbb{A}} = (\mathbb{A}, C = M \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, \mathbf{C}_a = g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})})$. Define $\mathcal{S}_{\mathbb{A}} = \{\mathrm{att} \sim \mathbb{A} \mid \mathrm{att} \in \mathcal{S}\}$, and matrices $\mathbf{E}$, $\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}$ for each att $\in \mathcal{S}$ such that

$$\mathbf{c}\mathbf{E}\mathbf{k}_u^{\intercal} + \sum_{\mathrm{att}\in\mathcal{S}_{\mathbb{A}}} (\mathbf{c}, \mathbf{c}_a)\mathbf{E}_{\mathrm{att},\mathcal{S},\mathbb{A}}(\mathbf{k}_u, \mathbf{k}_a)^{\intercal} = \alpha s.$$

  Then, the plaintext message M can be retrieved by recovering $e(g,h)^{\alpha s}$ from $\mathbf{C}, \mathbf{C}_a$ and $\mathrm{SK}_{\mathrm{id}}, \mathrm{SK}_{\mathrm{id},\mathcal{S}}$, and $M = C/e(g,h)^{\alpha s}$.

- MKDecrypt(MK, CT): Let MK $= \alpha$, $\mathrm{MK}' = h^{\mathrm{mk}(\alpha,\mathbf{b})}$ and CT $= (C = m \cdot e(g,h)^{\alpha s}, \mathbf{C} = g^{\mathbf{c}(\mathbb{A},\mathbf{s},\mathbf{b})}, \mathbf{C}_a = g^{\mathbf{c}_a(\mathbb{A},\mathbf{s},\mathbf{b},\{\mathbf{b}_{\mathrm{att}}\}_{\mathrm{att}\sim\mathbb{A}})})$. Define a vector $\mathbf{e}$ such that $\mathbf{c}\mathbf{e}^{\intercal}\mathrm{mk} = \alpha s$. Then, m can be retrieved by computing

$$C / \prod_{\ell} e(C_\ell, \mathrm{MK}')^{\mathbf{e}_\ell},$$

  where C_ℓ and $\mathbf{e}_\ell$ denote the ℓ-th entry of $\mathbf{C}$ and $\mathbf{e}$, respectively.

Each encoding $\mathbf{enc}$(var) denotes a vector of polynomials over variables var. Generators constructed by hash functions [BSW07] are covered by this definition by assuming that $\mathcal{H}(\mathrm{att}) = g^{b_{\mathrm{att}}}$ for some implicit b_{att}. Depending on the scheme, MKSetup may be run distributively or by a single CA (in which case there is only one public key $e(g,h)^{\alpha}$ associated with the master keys), or independently and individually by multiple authorities $\mathcal{A}_i$ (in which case there are multiple public keys $e(g,h)^{\alpha_i}$, and we replace the blinding value $e(g,h)^{\alpha s}$ by $e(g,h)^{\sum_{i\in\mathcal{I}}\alpha_i s}$).

5.3.2 Modeling knowledge of exponents – extending $\mathbb{Z}_p$

The previously defined notation describes the relationship between the various variables "in the exponent" of the keys and ciphertexts. The explicit values of most variables are unknown to the attacker. In multi-authority ABE, authorities provide the inputs to some encodings, and therefore know these values, as well as their (part of the) master key. Hence, corruption of authorities results in the knowledge of some explicit values "in the exponent". If the values provided by honest authorities are not well-hidden, it might enable an attack on them.

We model the "knowledge of exponents" in attacks by extending the space from which the entries of $\mathbf{E}$ and $\mathbf{E}_{\text{att},\mathcal{S},\mathbb{A}}$ are chosen: $\mathbb{Z}_p$ (or some extension with variables associated with $\mathcal{S}$ and $\mathbb{A}$). In fact, the entries of these matrices may be any fraction of polynomials over $\mathbb{Z}_p$ and the known exponents. Let $\mathfrak{K}$ be the set of known exponents, then the extended field of rational fractions $\mathbb{Z}_p(\mathfrak{K})$ is defined as the quotient field of $\mathbb{Z}_p[\mathfrak{K}]$, where $\mathbb{Z}_p[\mathfrak{K}]$ denotes the polynomial ring in variables $\mathfrak{K}$. We write the elements in $\mathbb{Z}_p(\mathfrak{K})$ as $ab^{-1} \pmod{p}$, where $a, b \in \mathbb{Z}_p[\mathfrak{K}]$ and $b \neq 0$.

5.3.3 Formal definitions of the attacks in the concise notations

We formally define our attack models (conform Definitions 5.6–5.8, depicted in Figure 5.1) in the concise notation. For each attack, $\mathfrak{K} \subseteq \{x, x_1, x_2, ...\}$ denotes the set of known variables. We use the following shorthand for a key encoding for a user id with set $\mathcal{S}$ and for a ciphertext encoding for access structure $\mathbb{A}$:

$$\begin{aligned}\mathbf{k}_{\text{id},\mathcal{S}} &:= (\mathbf{gp}(\mathbf{b}), \mathbf{mpk}_a(b_{\text{att}}, \mathbf{b}), \mathbf{k}_u(\text{id}, \alpha, \mathbf{r}_u, \mathbf{b}), \mathbf{k}_{a,1}(\text{att}, \mathbf{r}_a, \mathbf{b}, \mathbf{b}_{\text{att}}), ...),\\ \mathbf{c}_{\mathbb{A}} &:= (\mathbf{gp}(\mathbf{b}), \mathbf{mpk}_a(b_{\text{att}}, \mathbf{b}), \mathbf{c}(\mathbb{A}, \mathbf{s}, \mathbf{b}), \mathbf{c}_a(\mathbb{A}, \mathbf{s}, \mathbf{b}, \{\mathbf{b}_{\text{att}}\}_{\text{att}\sim\mathbb{A}})).\end{aligned}$$

We first define the master-key attacks. In these attacks, the attacker has to retrieve master key $\text{mk}(\alpha, \mathbf{b})$, so any ciphertext can be decrypted conform MKDecrypt. In many schemes, it holds that master key mk is α (i.e., h^{α}), though in others, recovering e.g., $\text{mk}_i = \alpha_i / b_i$ for authorities $\mathcal{A}_i$ is required to decrypt all ciphertexts. This is because ciphertext encoding $\mathbf{c}$ often contains s or sb_i.

Definition 5.6: Master-key attacks

A scheme is vulnerable to a master-key attack if there exist $(\text{id}_1, \mathcal{S}_1), ..., (\text{id}_n, \mathcal{S}_n)$ and the associated key encodings $\mathbf{k}_{\text{id}_i,\mathcal{S}_i}$, and there exist $\mathbf{e}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i}$, where $\ell_i = |\mathbf{k}_{\text{id}_i,\mathcal{S}_i}|$ denotes the length of the i-th key encoding, such that $\sum_i \mathbf{k}_i \mathbf{e}_i^{\intercal} = \text{mk}(\alpha, \mathbf{b}) \in \mathbb{Z}_p(\alpha, \mathbf{b})$. Then, it holds that for all attribute-independent ciphertext encodings $\mathbf{c}$ there exists $\mathbf{e}' \in \mathbb{Z}_p^{\ell'}$ (with $|\mathbf{c}| = \ell'$) such that $\text{mk} \cdot \mathbf{e}' \cdot \mathbf{c}^{\intercal} = \alpha s$.

We formally define attribute-key attacks. In an attribute-key attack, the attacker has to generate a secret key associated with a set $\mathcal{S}'$ that is strictly larger than any of the sets $\mathcal{S}_i$ associated with the issued keys.

Definition 5.7: Attribute-key attacks

A scheme is vulnerable to an attribute-key attack if $(\mathrm{id}_1, \mathcal{S}_1), ..., (\mathrm{id}_n, \mathcal{S}_n)$ exist such that for the key encodings $\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}$, it holds that a valid key $\overline{\mathbf{k}}_{\mathrm{id}',\mathcal{S}'}$ (with user-specific randoms $\overline{\mathbf{r}}_u$ and $\overline{\mathbf{r}}_a$ constructed linearly from the other user-specific randoms) can be computed such that $\bigcup_{i=1}^n \mathcal{S}_i \subseteq \mathcal{S}'$ and $\mathcal{S}_i \subsetneq \mathcal{S}'$ for all $i \in [n]$. We say that $\overline{\mathbf{k}}_{\mathrm{id}',\mathcal{S}'}$ can be computed, if there exist $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \overline{\ell}}$, where $\overline{\ell} = |\overline{\mathbf{k}}_{\mathrm{id}',\mathcal{S}'}|$ and $\ell_i = |\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}|$, for all $\mathcal{S}_i$ such that $\overline{\mathbf{k}}_{\mathrm{id}',\mathcal{S}'} = \sum_i \mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}\mathbf{E}_i$.

We formally define the complete and conditional decryption attacks. In a decryption attack, the attacker decrypts a ciphertext for which it only has unauthorized keys. The attack is conditional if the collective set of attributes satisfies the access structure associated with the ciphertext. Otherwise, it is complete.

Definition 5.8: Complete/conditional decryption attacks

A scheme is vulnerable to a decryption attack if there exist $(\mathrm{id}_1, \mathcal{S}_1), ..., (\mathrm{id}_n, \mathcal{S}_n)$ and $\mathbb{A}$ such that $\mathbb{A} \not\models \mathcal{S}_i$ for all i, associated ciphertext encoding $\mathbf{c}_{\mathbb{A}}$ and key encodings $\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}$, for which there exist $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \ell'}$, where $\ell_i = |\mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}|$ and $\ell' = |\mathbf{c}_{\mathbb{A}}|$, such that $\sum_i \mathbf{k}_{\mathrm{id}_i,\mathcal{S}_i}\mathbf{E}_i\mathbf{c}_{\mathbb{A}}^{\intercal} = \alpha s$. The attack is conditional if it holds that $\mathbb{A} \models \bigcup_i \mathcal{S}_i$. Otherwise, it is complete.

It readily follows that master-key and attribute-key attacks imply decryption attacks. Specifically, master-key attacks and attribute-key attacks for which $\bigcup_{i=1}^n S_i \subsetneq \mathcal{S}'$ holds imply complete decryption attacks.

5.3.4 Definitions of multi-authority-specific attacks

The multi-authority setting yields two additional difficulties in the design of secure schemes. First, the corruption of authorities yields extra knowledge about the exponent space. Second, the distributed nature of the master key may enable new attacks. Formally, we define attacks under corruption as follows.

Definition 5.9: Attacks under corruption

A scheme is vulnerable to attacks under corruption if an attacker can corrupt a subset $\mathcal{I} \subsetneq \{1, ..., m\}$ of authorities $\mathcal{A}_1, ..., \mathcal{A}_m$ and thus obtain knowledge of variables $\mathfrak{K}$ consisting of all variables and (partial) encodings generated by the corrupt authorities, enabling an attack conform Definitions 5.6, 5.7 or 5.8.

Oftentimes, the master key is generated distributively by the authorities. Hence, the blinding value is of a distributed form, e.g., $e(g,h)^{\alpha s} = e(g,h)^{\sum_i \alpha_i s}$. If each partial blinding value e.g., $e(g,h)^{\alpha_i s}$ can be recovered independently of the user's randomness, then the scheme is vulnerable to a multi-authority-specific decryption attack under collusion. For instance, suppose the blinding value is defined as $(\alpha_1 + \alpha_2)s$. If one user can recover $\alpha_1 s$ (but not $\alpha_2 s$) and another user can recover $\alpha_2 s$ (but not $\alpha_1 s$), then the scheme is vulnerable to a multi-authority-specific decryption attack. They can collectively recover $(\alpha_1 + \alpha_2)s$, while clearly, they cannot do this individually. This type of attack was also performed by Wang et al. [WZC15] on the HSM+14 [HSM$^+$14] and HSM+15 [HSM$^+$15] schemes.

Definition 5.10: Multi-authority-specific (MAS) decryption attacks

Suppose the blinding value of the message is of the form $\sum_i \mathrm{bv}_i(\alpha_i, \mathbf{s}, \mathbf{b})$, where α_i denotes the master key of authority $\mathcal{A}_i$, and bv_i represent elements in $\mathbb{G}_T$. A scheme is vulnerable to a MAS-decryption attack if there exist a ciphertext encoding $\mathbf{c}_{\mathbb{A}}$ and sets $\mathcal{S}_i \subseteq \mathcal{U}_i$ with key encodings $\mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i}$ for which there exist $\mathbf{E}_i \in \mathbb{Z}_p(\mathfrak{K})^{\ell_i \times \ell'}$, where $\ell_i = |\mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i}|$ and $\ell' = |\mathbf{c}_{\mathbb{A}}|$, such that $\mathbf{k}_{\mathrm{id}_i, \mathcal{S}_i} \mathbf{E}_i \mathbf{c}_{\mathbb{A}}^{\intercal} = \mathrm{bv}_i$.

A MAS-decryption attack is also a decryption attack conform Definition 5.8. The blinding value can be retrieved, while the individual sets are not authorized to decrypt the ciphertext. Conversely, because such attacks do not exist in the single-authority setting, they are weaker than regular decryption attacks.

5.3.5 Our heuristic approach

We devise a targeted approach, which can be applied manually (or automatically), to finding attacks. As the definitions in the previous section imply, finding an attack is equivalent to finding a suitable linear combination—where the linear coefficients are the entries of $\mathbf{e}$ or $\mathbf{E}$—of all products of the key and ciphertext entries. While finding such coefficients is relatively simple, we note that finding suitable inputs to the attacks may be more difficult. In particular, the number of colluding users and the number of attributes associated with the keys and ciphertexts are effectively unbounded. However, we observe that it often suffices to consider a limited number of inputs, and that for some attacks, only the user-key and attribute-independent ciphertext entries need to be considered. Specifically, Table 5.3 describes these inputs in terms of encodings, the sets of attributes, and the access policy. Depending on the maximum number of monomials consisting of common variables in any key entry, the attacker might need multiple secret keys for the same set of attributes to recover certain coefficients. For instance, suppose the attacker wants to retrieve α from $\alpha + r_1 b_{\mathrm{att}_1} + r'_1 b'_{\mathrm{att}_1}$, where r_1 and r'_1 are known, user-specific random variables, and b_{att_1} and b'_{att_1} denote the common variables associated with attribute att_1. Because of the three unknown, linearly independent monomials, this can only be done if the attacker has three distinct keys for attribute att_1. In general, the maximum number

Table 5.3. The inputs of the attacks, and which encodings are needed.

Attack	Secret keys			Ciphertexts		
	UK	AK	$\mathcal{S}$	AI	AD	$\mathbb{A}$
Master-key	✓	✗	-	✗	✗	-
Attribute-key	✓	✓	$\mathcal{S}_1 = \{\mathrm{att}_1\}, \mathcal{S}_2 = \{\mathrm{att}_2\}$	✗	✗	-
Complete decryption	✓	✗	-	✓	✗	-
Conditional decryption	✓	✓	$\mathcal{S}_1 = \{\mathrm{att}_1\}, \mathcal{S}_2 = \{\mathrm{att}_2\}$	✓	✓	$\mathbb{A} = \mathrm{att}_1 \wedge \mathrm{att}_2$

UK, AK = user-, attribute-key; AI, AD = attribute-independent, -dependent

Table 5.4. The number of required honest authorities n and the attribute universes $\mathcal{U}_1$ and $\mathcal{U}_2$ managed by authorities $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, in the multi-authority setting.

Attack	n	$\mathcal{U}_1$	$\mathcal{U}_2$
Master-key	1	✗	✗
Attribute-key	1	$\{\mathrm{att}_1, \mathrm{att}_2\}$	✗
Complete decryption	1	✗	✗
Conditional decryption	1	$\{\mathrm{att}_1, \mathrm{att}_2\}$	✗
MAS-decryption	2	$\{\mathrm{att}_1\}$	$\{\mathrm{att}_2\}$

of keys with the same set of attributes can be determined in this way, i.e., by counting the maximum number of linearly independent monomials for each entry.

Similarly, the inputs to multi-authority specific attacks can be limited. First, we consider the attacks under corruption. Corruption of any number of authorities results in the additional knowledge of some otherwise hidden exponents, i.e., the master keys and any random variables generated by these authorities. For most schemes, it should be sufficient to consider one corrupted and one honest authority in the attacks, though depending on how e.g., the master key α is shared, the number of corrupted authorities may need to be increased. Further, we use the same descriptions of the inputs to the attacks as in the single-authority setting, with the additional requirement that the input attributes are managed by the honest authority. Second, we consider multi-authority specific (MAS) decryption attacks. Corruption is not necessary in this setting, so we assume that all authorities are honest. Additionally, we require at least two honest authorities as input to finding any attack, so we let each authority manage one attribute. Table 5.4 summarizes the additional inputs to the attacks in Table 5.3. Finally, it may be possible that a corruptable central authority (CA) is part of the scheme, in which case we also consider whether corruption of this CA enables an attack.

We describe a more targeted approach to finding an attack, i.e., the linear coefficients $\mathbf{e}$ and $\mathbf{E}$, given the input encodings. The approach to finding an attack is linear, as we attempt to retrieve the desired output (conform Definitions 5.6, 5.7 and 5.8) by making linear combinations of products of encodings. The simplest attacks are the master-key and complete decryption attacks, as we only need to consider the attribute-independent parts of the keys and ciphertexts. For these attacks, the goal is to retrieve master key α, or blinding value αs. Typically, α occurs only in one entry of the keys, while s occurs only in one entry of the ciphertext. Instead of trying all combinations of the key entries with the ciphertext, we formulate a more targeted approach. First, consider the monomials to be canceled, and then which combinations of the key and ciphertext entries can make these monomials. In canceling the previous monomials, it might be that new monomials are added, meaning that these in turn also need to be canceled. This process repeats until all monomials are canceled, and α or αs remains, unless such an attack does not exist. For attribute-key attacks, this effort is considerably more difficult, as the target is less clear. However, it often suffices to consider whether the same monomial occurs more than once in the key encoding. For conciseness, we will only provide the non-zero coefficients in an attack.

5.4 Examples of attacks demonstrating the approach

Using examples of attacks that we have found, we illustrate the way in which our heuristic approach can be applied. In particular, this suggests the simplicity of only considering the exponent space rather than also considering the underlying group structure. Furthermore, in our strongest attack models (i.e., master-key and complete decryption), we often only need to consider the attribute-independent variables, which strips away a large and significantly more difficult part of the scheme. Because many schemes are broken in these models, we assert that it has merit to manually analyze schemes with respect to these models.

5.4.1 Example without corruption: DAC-MACS

We perform the attack on DAC-MACS [YJRZ13, YJR+13] in Section 5.2 in the concise notations.

- **Type of attack:** Complete decryption attack;
- **Global parameters:** $\mathbf{gp} = (\mathrm{gp}_1, ...) = (b, ...)$;
- **Master keys $\mathcal{A}_i$: $\mathbf{mpk}_i$** $= b_i$;
- **User-key:** $\mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b}) = (\alpha_i/x_1 + x_2 b + r_i b/b_i, r_i b_i/x_1, r_i b)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, s/b_i)$;
- **Blinding value:** $\alpha_i s$;
- **Known exponents:** $\mathfrak{K} = \{x_1, x_2\}$ (by definition);

Note that this notation is not only more concise, it is also more structured. In particular, it is clearly denoted what the goal is (i.e., retrieve the blinding value), and what the relevant keys and ciphertexts look like without considering any information about the underlying groups or attribute-dependent variables. Furthermore, this allows us to strip away any additional functionality that further complicates the structure—and by extension, the analysis—of the scheme.

Due to the concise notations, the previous attack can also be found more simply than before. First, we sample a user-key $(k_1, k_2, k_3) \leftarrow \mathbf{k}_u(\alpha, \mathbf{r}, \mathbf{b})$, and ciphertext $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$. To retrieve $\alpha_i s$, we start by pairing k_1 with c_1:

$$x_1k_1c_1 = \overset{\text{blinding value}}{\overset{\downarrow}{\alpha_i s}} + \overbrace{\underset{\underset{x_1x_2\mathrm{gp}_1c_1}{\uparrow}}{x_1x_2sb} + \underset{\underset{x_1k_3c_2}{\uparrow}}{x_1r_isb/b_i}}^{\text{to cancel}},$$

which yields two monomials to cancel. Subsequently, we can combine the other components and our explicit knowledge of x_1 and x_2 in such a way that these monomials can be canceled. This attack can be formulated in matrix notations:

$$\begin{aligned}\alpha_i s &= \underbrace{(k_1, k_2, k_3, \mathrm{gp}_1)}_{\mathbf{k}_u} \underbrace{\begin{pmatrix} x_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -x_1 & 0 \\ -x_1x_2 & 0 & 0 \end{pmatrix}}_{\mathbf{E}} \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ \mathrm{gp}_1 \end{pmatrix}}_{\mathbf{c}} \\ &= x_1k_1c_1 - x_1k_3c_2 - x_1x_2\mathrm{gp}_1c_1.\end{aligned}$$

Because most of the entries of $\mathbf{E}$ are zero, we will only write the non-zero entries of $\mathbf{E}$ in further attacks. Note that attacks found in the concise notations also translate back to the original description, e.g., compare this attack with that in Section 5.2. More generally, computing $\mathbf{k}_j\mathbf{E}_{i,j}\mathbf{c}_i$ in terms of pair encodings corresponds to computing $e(g^{\mathbf{c}_i}, h^{\mathbf{k}_j})^{\mathbf{E}_{i,j}}$ in the original description of the scheme.

5.4.2 Example with corruption: the [YJ14] scheme

The YJ14 [YJ14] scheme is somewhat similar to the YJR+13 [YJR$^+$13] scheme in the secret keys. However, the decrypting user knows fewer exponents: instead of sharing x_2 in YJR+13 with the user, it is shared with the authorities $\mathcal{A}_i$. Regardless, corruption of one authority leads to the knowledge of x_2, and thus enables an attack. We define the encodings and attack as follows.

- **Type of attack:** Complete decryption attack, under corruption of one $\mathcal{A}_i$;
- **Global parameters:** $\mathbf{gp} = (b, b')$;

- **Master secret key $\mathcal{A}_i$:** $\mathfrak{msk}_i = (\alpha_i, x)$;
- **User-key:** $\mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b}) = (\alpha_i + xb + rb', r)$;
- **Attribute-independent ciphertext:** $\mathbf{c}(\mathbf{s}, \mathbf{b}) = (s, sb', ...)$;
- **Blinding value:** $(\sum_i \alpha_i)s$;
- **Known variables:** $\mathfrak{K} = \{x\}$ (by corrupting $\mathcal{A}'$);
- **The goal:** Recover $\alpha_i s$ from $(k_{1,i}, k_{2,i}) \leftarrow \mathbf{k}_u(\alpha_i, \mathbf{r}, \mathbf{b})$, $(c_1, c_2) \leftarrow \mathbf{c}(\mathbf{s}, \mathbf{b})$;
- **The attack:** $\alpha_i s = k_{1,i}c_1 - k_{2,i}c_2 - x\text{mpk}_1 c_1$.

5.4.3 Example without corruption: the [JLWW13] scheme

We also give an example of a conditional attribute-key attack enabled by two colluding users. This illustrates the increased difficulty of executing more general attacks, as they require us to evaluate the entire key. An additional difficulty of executing an attribute-key attack is in finding an appropriate target key encoding. However, our possibilities as an attacker are considerably limited, as we can only linearly combine the key components, and not multiply them. In fact, as Table 5.2 shows, we could only find attribute-key attacks if a key consists of recurring monomials. While it is difficult to prove that an attribute-key attack does not exist, it is easy to verify whether a key consists of recurring monomials.

We attack the [JLWW13] and [JLWW15] schemes—also known as AnonyControl—which have the same key generation. The JLWW15 scheme is different from JLWW13 in the encryption. It is however incorrect, because a value of a single user's secret key is used. The encodings are defined as follows.

- **Type of attack:** Conditional attribute-key attack, collusion of two users;
- **Global parameters:** $\mathbf{gp} = (b, b'), \mathbf{mpk}_a(\text{att}_i) = b_{\text{att}_i}$;
- **Secret keys:** $\mathbf{k}_u(\alpha, r, \mathbf{b}) = (\alpha + r)$, $\mathbf{k}_a(\text{att}_i, r, r_i, \mathbf{b}) = (r_i b_{\text{att}_i} + r, r_i)$;

We show that the recurrence of r as a monomial in the user-key and attribute-key encoding enables an attack. While it is relatively simple to show that this cannot be exploited in a single-user setting, we show that sampling two keys for two different sets of attributes $\mathcal{S}_1 = \{\text{att}_1\}$ and $\mathcal{S}_2 = \{\text{att}_2\}$ (as in Table 5.3) enables the generation of a third key for both attributes, i.e., $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$. For $\mathcal{S}_1 = \{\text{att}_1\}$, we sample $k \leftarrow \mathbf{k}_u(\alpha, r, \mathbf{b})$, and $(k_1, k_2) \leftarrow \mathbf{k}_a(\text{att}_1, r, r_1, \mathbf{b})$. For $\mathcal{S}_2 = \{\text{att}_2\}$, we sample $k' \leftarrow \mathbf{k}_u(\alpha, r', \mathbf{b})$, and $(k'_1, k'_2) \leftarrow \mathbf{k}_a(\text{att}_2, r', r_2, \mathbf{b})$.

The goal is to compute a key for set $\mathcal{S}_3 = \{\text{att}_1, \text{att}_2\}$. We aim to generate attribute-keys for the user-key associated with $\mathcal{S}_1$, i.e., k, which links the keys together with r. As such, to create a key for $\mathcal{S}_3$, we need to generate an attribute-key for att_2. We do this by computing: $\mathbf{k}_a(\text{att}_2, r, r_2, \mathbf{b}) = (k'_1 + k - k', k'_2)$.

5.5 Future work and conclusion

We have presented a linear, heuristic approach to analyzing security—consisting of a more concise notation—and applied it to existing schemes. This approach simplifies manually finding generic attacks provided that they exist. For future work, it would be valuable to extend the approach to be provably exhaustive, such that it follows with [ABGW17] that the scheme also implies a provably secure scheme. In addition, it would be valuable to automatize finding attacks for the multi-authority encodings like [ABGW17] does in the single-authority setting. To demonstrate the effectiveness of our approach, we have shown that several existing schemes are vulnerable to our attacks, either rendering them fully or partially insecure (see the paper [VA20, VA21] for more attacks). Most of the attacks are similar in that they either exploit that one monomial occurs more than once in the keys, or known exponents yield sufficient knowledge to enable an attack. In general, schemes for which we found an attack without requiring corruption are structurally more complicated than the single-authority schemes on which they are (loosely) based. Schemes that are insecure against corruption are generally closer to their (provably secure) single-authority variants, but knowing certain exponents enables an attack. Possibly, distributively generating these exponents may prevent this. For future work, it may be interesting to consider whether this yields secure schemes.

Chapter 6

ABE Squared: accurately benchmarking efficiency of ABE

Measuring efficiency is difficult. In the last decades, several works have contributed in the quest to successfully determine and compare the efficiency of pairing-based ABE. However, many of these works are limited. In this chapter, we present a framework for accurately benchmarking efficiency of ABE: ABE Squared. In particular, we focus on uncovering the multiple layers of optimization that are relevant to the implementation of ABE schemes. Moreover, we focus on making any comparison fairer by considering the influence of the potential design goals on any optimizations. On the lowest layer, we consider the available optimized arithmetic provided by state-of-the-art cryptographic libraries. On the higher layers, we consider the choice of elliptic curve, the order of the computations, and importantly, the instantiation of the scheme on the chosen curves. To compare schemes more transparently, we develop this framework, in which ABE schemes can be justifiably optimized and compared by taking into account the possible goals of a designer. To illustrate the effectiveness of ABE Squared, we implement several schemes and provide all relevant benchmarks. These show that the design goal influences the optimization approaches, which in turn affect the overall efficiency of the implementations. Importantly, these demonstrate that the schemes also compare differently than existing works previously suggested.

6.1 Introduction

Since attribute-based encryption was introduced, much progress has been made in the development of pairing-based ABE schemes. As is common in the field of cryptography, whenever a new scheme is presented, its efficiency is compared to that of other state-of-the-art schemes. For ABE, the Charm framework [AGM$^+$13] is used in many cases [RW13, RW15, AC17a, ABGW17], which simplifies the prototyping of new pairing-based schemes and provides benchmarking tools. However, because Charm mainly aims at usability in this endeavor, it uses several abstraction layers between the schemes and the necessary arithmetic. As a result, not all available optimizations can be used in benchmarking efforts, even though these might be significant in any comparisons. Furthermore, by default, the Charm framework builds on the PBC library [Lyn13], which only supports outdated elliptic curves that have been proven not to provide 128 bits of security [KB16, BD19]. Consequently, many implementations and efficiency comparisons use these outdated curves. By extension, those implementations do not provide realistic estimates of computational costs in practice. When implemented for practice, curves that currently provide 128 bits of security should be used. Because these might provide different trade-offs in efficiency, the implementations may incur different computational costs than the curves used in the old benchmarks [Ara17, CDS20].

Oftentimes, works that do not use Charm in their efficiency analyses have similar issues. For instance, they may not use all, if any, optimized arithmetic or other lower-level optimization techniques [Zeu20, AHM$^+$16, TKN20, PRMV21]. Such techniques allow for faster computations of exponentiations, such as fixed-base exponentiations or multiple-base exponentiations [Sco11, Möl01]. These are often used when elliptic-curve schemes are deployed in practice, and provide a significant computational advantage over regular variable-base exponentiations. Other implementations may be targeted for specific platforms such as certain embedded devices [SR13, WZSI14, MTP$^+$21]. Hence, they are difficult to use in future efficiency comparisons without implementing the schemes for those specific devices. On the other hand, software implementations that do optimize the arithmetic used in the schemes [ZPM$^+$15] have implemented all underlying arithmetic for some specific elliptic curve, and are therefore difficult to adapt to other, more up-to-date, curves. This is problematic, since this particular curve, e.g., the BN254 curve [BN05], may turn out to provide, e.g., only 100 bits of security [SKSW20].

Another common denominator of these implementations is the absence of a clearly-defined design rationale when the schemes are instantiated in pairing-friendly elliptic-curve groups. For instance, choosing a suitable pairing-friendly group providing 128 bits of security [Gui20b] is important for the overall efficiency [Sco11, CDS20]. However, not every curve may be a good choice for every scheme. Moreover, at the protocol level, schemes are often designed in the symmetric, type-I setting [Sco11]. On the other hand, in practice, it is better to use asymmetric, type-III pairings due to their efficiency and security (Section 2.5). While such schemes can be converted from the type-I to the type-III setting [RCS12], existing works that facilitate this [AGH13, AGOT14, AGH15, AHO16a] are often not used in the implementation of ABE schemes.

Nevertheless, such a design rationale determines the groups in which the computations are performed during key generation, encryption and decryption, and is therefore crucial when analyzing the efficiency of the scheme. This rationale heavily influences the choice of pairing-friendly groups and the type conversion, and by extension, the efficiency of the scheme. For instance, operations in $\mathbb{G}$ are generally more efficient than those in $\mathbb{H}$ [Ara17, AGM$^+$, CDS20]. Consequently, if a designer places all ciphertext components in $\mathbb{G}$, then the encryption efficiency is optimized at the expense of the key generation efficiency. Another designer might want to optimize the key generation efficiency, and therefore places all key components in $\mathbb{G}$, and the ciphertext components in $\mathbb{H}$. Because not all implementations take into account and specify these considerations, they cannot be effectively and meaningfully compared [VAH21]. In fact, a somewhat unethical cryptographer who, for instance, wants to promote their new scheme's fast encryption algorithm could place all ciphertext components of their own scheme in $\mathbb{G}$, while they place the compared scheme's ciphertext components in $\mathbb{H}$. As a result, their own scheme might outperform the other scheme, even though the other scheme would have outperformed the new scheme if its ciphertext components had also been placed in $\mathbb{G}$. In summary, for various efficiency goals, a different distribution of the key and ciphertext components over the two source groups may be optimal.

In this chapter, we aim to resolve the aforementioned issues. In particular, we provide a framework for benchmarking and comparing efficiency of ABE schemes that takes into account important features such as optimized arithmetic and conversion techniques. Along the way, we introduce novel conversion techniques to obtain, e.g., a type conversion with an optimized decryption algorithm. We also show how this framework can be applied to existing schemes by implementing and benchmarking them. Lastly, we illustrate how these benchmarks can be compared fairly, by comparing the variants that are optimized in the same way, e.g., the variants with an optimized decryption algorithm.

6.1.1 Our contribution

We set up ABE Squared, a general framework for accurately benchmarking efficiency of ABE. This framework describes optimization approaches in the implementation of ABE schemes based on various design goals, by unifying multiple established areas in optimization. By choosing one design goal, multiple schemes can be optimized in a uniform way, and thus be fairly compared. Concretely,

- we identify four *optimization layers* that are important in the implementation of the schemes:
  - the used arithmetic and group operations;
  - the choice of pairing-friendly curve;
  - the order of computations;
  - the type-conversion techniques;
- we formulate various *optimization approaches* for several clearly defined design goals:

  - optimized key generation;
  - optimized encryption;
  - optimized decryption;
  - balanced (in any combination of the algorithms, e.g., balanced key generation/encryption, balanced encryption/decryption);

- as part of the optimization approaches, we introduce new heuristic, manual *conversion techniques* from the type-I to the type-III setting, which take into account the other optimization layers. This is especially important for optimizing the decryption algorithm, for which the existing frameworks fall short.

To illustrate the effectiveness of our framework, we provide the *implementations* of several important CP-ABE schemes: Wat11 (Construction 4.1), RW13 (Construction 4.2) and AC17 (Construction 4.3).

6.1.2 Background

We provide further background information and motivate our choices and some of the features of the new framework.

RELIC. Our framework uses the RELIC toolkit [AGM$^+$], which is a cryptographic library that can be used for building elliptic-curve and pairing-based cryptographic schemes. In particular, it implements pairing-friendly elliptic curves that provide at least 128 bits of security, such as BLS12-381 and BLS12-446 [BLS02, Bow, BD19, WB19]. It provides high-speed implementations of frequently used arithmetic and group operations. In general, using a library for the lowest level of optimizations, i.e., arithmetic and groups operations, allows us to easily instantiate the schemes with new—possibly more secure or more efficient—curves, in the case that it is necessary to do so. We chose RELIC because it is actively maintained, and compared to other libraries such as MIRACL [Sco03], it supports more pairing-friendly curves providing 128 bits of security. This allows us to compare the efficiency of the implemented schemes on multiple curves with the same security level. In this way, we can investigate which trade-offs the various curves incur.

BLS12-381. A specific elliptic curve of interest supported by RELIC is the BLS12-381 curve [Bow, WB19], which is an instantiation of the curves designed by Barreto, Lynn and Scott (BLS) [BLS02]. In the last few years, BLS12-381 has established itself as a popular curve. For example, it is used in the Algorand blockchain [GHM$^+$17, BKLS02] and by the privacy-oriented Zcash blockchain in the implementation of zk-SNARKS [BCCT12, ZCa21].

Other pairing-friendly curves. We show that the chosen pairing-friendly curve influences the overall efficiency of the scheme, and is thus an important aspect in the optimization. To illustrate this, we benchmark the schemes on various curves, at the same security levels (see Section 6.2.2 for a selection of the curves). As part of the optimization approaches, we pick the best curve, given the chosen design goal.

OpenABE. In addition to Charm, we also compare our implementations with the OpenABE [Zeu20] implementation of the scheme by Waters [Wat11, Wat08]. It is a publicly available software implementation of ABE that is actively maintained, and that is specifically designed for practical application rather than for benchmarking efforts. Furthermore, OpenABE also relies on RELIC for the curve arithmetic. However, it is specifically configured to only support the BN254 curve.

The implemented schemes. We implement several variants of three schemes: Wat11 (Construction 4.1), RW13 (Construction 4.2) and AC17 (Construction 4.3). Specifically, we consider two versions of Wat11 and AC17: their small-universe and large-universe variants. We have chosen these schemes, in part, due to their popularity in follow-up work. By benchmarking and comparing these constructions, we obtain a better understanding of their efficiency. Compared to Charm, their efficiency also compares differently, which illustrates that our framework truly leads to significant improvements in the accuracy of the benchmarks.

Our type-conversion techniques. As part of our framework, we introduce novel heuristic and manual techniques to convert the schemes from the type-I to the type-III setting. Although type conversion has been extensively studied, culminating in various works that even automate this effort [AGH13, AGOT14, AGH15, AHO16a], these do not sufficiently optimize the schemes for all of the design goals that we consider in our optimization approaches. In particular, some of these conversion techniques focus, in terms of efficiency, mainly on the sizes of the parameters [AGOT14, AGH15, AHO16a]. These are not sufficient in optimizing the type-converted scheme for, e.g., the optimized decryption approach. In contrast, [AGH13] also allows for the optimization of the computational costs, but only takes into account the number of group operations, e.g., exponentiations, and not the computational costs of the various group operations. This does not allow us to distinguish between the different group operations within the same group, even though they may incur different computational costs. Furthermore, [AGH13] does not explicitly allow the choice of pairing-friendly curve or the order of the computations to be taken into account. Hence, we develop techniques to consider these different computational costs, by also measuring these costs for various suitable curves, and by determining the most efficient order of computations. In this way, we can minimize the computational costs of the algorithm(s) more accurately, based on the chosen design goal and associated optimization approach.

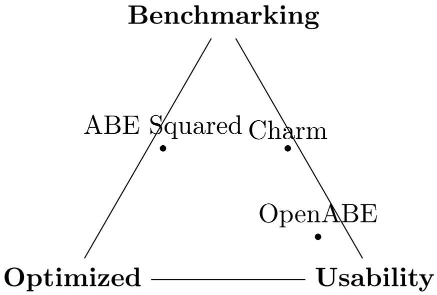


Figure 6.1. Rough overview of the position of our framework compared to Charm and OpenABE, with respect to the goals: benchmarking, optimization and usability.

Code. Our implementations of the schemes are available in the public domain, and can be found at `https://github.com/abecryptools/abe_squared`.

Positioning our framework. Our framework aims to bridge a gap in the benchmarking of ABE schemes, as described above. Notable software implementations that are built on libraries such as RELIC and that provide benchmarking utilities—and that are still maintained—are Charm and OpenABE. However, their goals are arguably different from ours (see Figure 6.1). Charm and OpenABE are both focused on usability, albeit in different ways. Charm aims to be usable in the prototyping and benchmarking of schemes, so that cryptographers can implement new schemes without having to know implementation details. OpenABE aims to be usable for practical applications, providing ready-to-use ABE implementations for practitioners. As a result, neither of their implementations uses all available optimized arithmetic. In contrast, our framework, ABE Squared, focuses on optimization rather than usability, such that a more accurate view of the efficiency of ABE schemes can be obtained. Although the implementations can be used by any cryptographer to benchmark and compare the schemes, they are not immediately suitable for practical applications like OpenABE. Furthermore, our implementations do not aim to provide a platform to readily implement new schemes, like Charm does, since it requires a significant engineering expertise and some familiarity with RELIC.

6.2 Our framework: ABE Squared

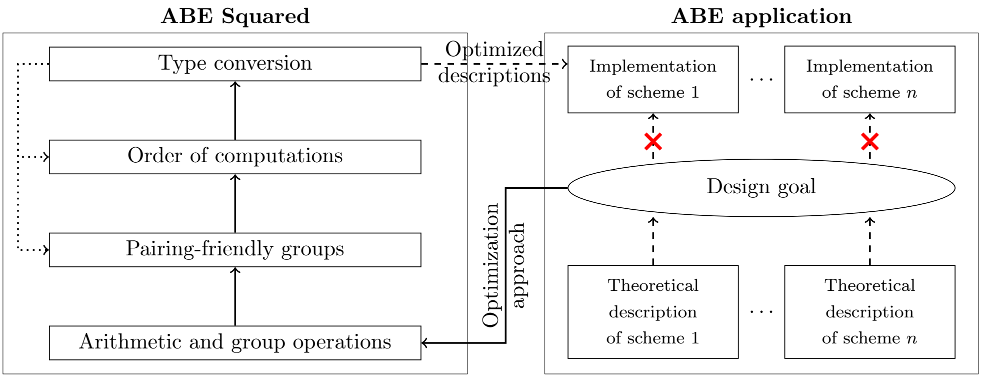
We introduce our framework, ABE Squared, in this section. Concretely, we identify several layers of optimization: the arithmetic and group operations, the pairing-friendly groups that are used, the order of the computations and the type conversion (see Figure 6.2). The goal of our framework is to optimize the theoretical description of the scheme. As such, we want to obtain a description of the scheme that directly yields the most efficient implementation. In this process, the design goal associated with a practical application is crucial: some applications may require an optimized encryption while others require an optimized decryption algorithm. In order to achieve this goal, these four layers need to be optimized. We do this by devising optimization approaches based on these design goals.

These optimization approaches consist of several steps. In particular, we first analyze the efficiency of the arithmetic and group operations used in the schemes by benchmarking their efficiency in the pairing-friendly groups that can be used. Subsequently, we show how the order of computations can be optimized, given the efficiency of the available algorithms for arithmetic in the chosen pairing-friendly groups. (Note, however, that the optimal order may depend on the choice of pairing-friendly groups and the distribution of the key and ciphertext components, i.e., in which groups these live. Because type conversion—which determines this distribution—is the next step, we may need to adjust the order at a later stage in the optimization approach.) Finally, we show how the schemes can be instantiated in these groups to obtain the best possible efficiency, given the design goal. To this end, we devise new manual and heuristic techniques to convert the scheme from the type-I to the type-III setting. Possibly, the choices that are made during this type conversion might require that we circle back to the choice of pairing-friendly groups or the order of computations. For instance, it may not be clear what the best choice of pairing-friendly group is without simply benchmarking the schemes for all of them.

6.2.1 Optimized arithmetic and group operations

We analyze the efficiency of the arithmetic that may be required to perform the algorithms of a scheme. Many efficient algorithms exist to perform arithmetic in groups $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$, including optimized algorithms for certain combinations of arithmetic. Furthermore, depending on the fixed use of certain variables, the use of precomputation may significantly speed up the computations.

- **Variable-base exponentiation (VBE):** an exponentiation of the form g^x, in which the variable base g varies in each execution of the algorithm;
- **Fixed-base exponentiation (FBE):** an exponentiation of the form g^x, in which the base g is fixed after the setup and is the same in each execution of the algorithm;
- **Multiple-base exponentiation (MBE):** a product of multiple exponentiations [Möl01], typically of the form $\prod_{i \in \mathcal{I}} g_i^{x_i}$ such that g_i are bases and x_i are


ABE Squared
ABE application
Type conversion
Order of computations
Pairing-friendly groups
Arithmetic and group operations
Optimized descriptions
Optimization approach
Implementation of scheme 1
. . .
Implementation of scheme n
Design goal
Theoretical description of scheme 1
. . .
Theoretical description of scheme n

The arrows have the following meaning:
$a \longrightarrow b$ = "a influences b"
$a \cdots\rightarrow b$ = "a may require adjustment in b"
$a \dashrightarrow b$ = "a is input to b"/"b is output of a"

Figure 6.2. Overview of the ABE Squared framework and its relationship with ABE applications. In particular, the diagram describes the steps needed between the theoretical descriptions and the implementations of (possibly multiple) ABE schemes. Instead of moving from a design goal to the implementation of a scheme directly, we first optimize the theoretical description of the scheme for the chosen design goal.

exponents for each $i \in \mathcal{I}$ with $|\mathcal{I}| \geq 2$. Note that RELIC refers to these as simultaneous exponentiations[1] instead, and has two functions for this algorithm: `_mul_sim`, a two-base variant and `_mul_sim_lot`, a multi-base variant;

- **Multi-pairing:** a product of pairing operations can be executed more efficiently [GS06]. In general, a pairing computation consists of a Miller loop [Mil04] and a final exponentiation. In a pairing product, the final exponentiation can be shared, i.e., it only needs to be performed once. In this way, only the Miller loop needs to be executed for each additional pairing operation in the product;

- **Fixed-argument pairing:** a pairing operation can be computed more efficiently if the first argument is fixed. For instance, [CS10] speeds up the Miller loop by 37%. RELIC does not support fixed-argument pairings, however;

[1] Actually, RELIC refers to multiple-base exponentiations as simultaneous multiplications, where 'multiplication' refers to a scalar multiplication. A scalar multiplication is an additive operation in an elliptic-curve group analogous to an exponentiation in a multiplicative group.

- **Hashing into the group:** a map from the set of arbitrary-length strings $\{0,1\}^*$ to a group. RELIC supports these, including a more optimized variant for the BLS12-381 curve [WB19].

6.2.2 Optimal choices of pairing-friendly groups

Another aspect that influences the efficiency of the algorithms is the choice of the pairing-friendly group [GPS08, Ara17]. In general, many pairing-friendly groups exist that provide 128 bits of security [Gui20a], currently the recommended minimum security level for cryptography [Bar20]. These groups typically consist of elliptic-curve groups, such as the BLS [BLS02] and BN [BN05] curves. Some of the curves listed in [Gui20a] provide more than 128 bits of security, and therefore, they will still likely yield sufficient security if the most novel attacks are slightly improved [KB16, BD19, Gui20a]. In contrast, other curves provide slightly fewer than 128 bits of security and may not provide sufficient security if these attacks are improved. RELIC [AGM$^+$] supports two curves in the $[125,128]$-range, i.e., BLS12-381 and BN382, and three curves with security levels in the $[129,135]$-range, i.e., BN446, BLS12-446 and BLS12-455. On the one hand, curves with a higher security level provide less efficient arithmetic [GPS08]. On the other hand, these curves provide more than 128 bits of security. This might also be beneficial, because most ABE schemes decrease a few bits in security as some of the parameters, e.g., the size of the access policies, grow [Wat11, RW13, AC17b]. If curves with a security level in the $[125,128]$-range are used, the implementations of these schemes provide even fewer than 128 bits of security. For instance, BLS12-381 currently provides roughly 126 bits of security [GMT20], and the schemes that we have selected lose an additional 4-7 bits for the maximum policy sizes that we will use. Therefore, the implementations provide 119-122 bits of security. In contrast, BLS12-446 and BN446 provide 132 bits of security [GS19], and thus, the implementations provide 128 bits of security. In this thesis, we focus only on the $[125,128]$-bit security range. We refer to the paper [dlPVA22] for a performance analysis of the results in the $[129,135]$-bit range.

6.2.3 Benchmarks of the group operations on various curves

To choose a suitable curve, it is important to know how efficiently the group operations perform. Table 6.1 lists the performances of various algorithms on the elliptic curves that we will use in our benchmarks in Section 6.3. The table shows that, at the same security level, BLS12 curves outperform the BN curves in almost all algorithms except hashing and multiple-base exponentiations with large numbers of bases. Therefore, we expect that for most, if not all, schemes, the BLS12 curves are better choices than the BN curves. The table also shows that the arithmetic in $\mathbb{G}$ is generally faster than the arithmetic in $\mathbb{H}$, which in turn is faster than the arithmetic in $\mathbb{G}_T$. Furthermore, performing an additional pairing operation—whose costs are slightly lower than the

Table 6.1. The computational costs of various algorithms on the elliptic curves used in our comparison, expressed in the number of 10^3 clock cycles. For each pair of curves with the same security level, the lowest costs are typeset in **bold**. These benchmarks were run on an AMD Ryzen 7 PRO 4750 processor, with power management disabled and throttle to max. frequency (one single core) at 4.1 GHz. Note that TBE and MBE denote a two-base exponentiation (run with `_mul_sim`) and multi-base exponentiation with two bases (run with `_mul_sim_lot`), respectively, and MP a multi-pairing with two pairings.

	Costs of the algorithms in $\mathbb{G}$					**Costs of the algorithms in $\mathbb{H}$**					**In $\mathbb{G}_T$**	**Pairing costs**	
Curve	**VBE**	**FBE**	**TBE**	**MBE**	**Hash**	**VBE**	**FBE**	**TBE**	**MBE**	**Hash**	**VBE**	**Pair**	**MP**
BN254	91	51	135	160	58	157	112	339	273	167	236	425	585
BLS12-381	**184**	**102**	**266**	**319**	225	**324**	**247**	**729**	**548**	548	**496**	**1245**	**1618**
BN382	266	153	382	473	**151**	487	372	1105	837	**480**	745	1405	1963

costs incurred by a Miller loop—is more costly than exponentiating in $\mathbb{G}$ and $\mathbb{H}$, while it is less costly than exponentiating in $\mathbb{G}_T$.

6.2.4 Optimizing the order of computations

The order of the computations can also be optimized. The most notable example of an optimized order of computations is to share a pairing operation when several components share an argument on the other side of the pairing [PTMW10]. From this point forward, we will refer to this kind of product as a shared-argument pairing product. For instance, rather than computing $\prod_{i\in\Upsilon} \hat{e}(K, C_i)$, one can compute $\hat{e}(K, \prod_{i\in\Upsilon} C_i)$, which only requires one pairing operation and $|\Upsilon|$ multiplications in one of the source groups instead of one $|\Upsilon|$-multi-pairing. Similar optimizations can be done by allowing the key generation authority to generate components such as $h^{r_{\text{att}}(b_1 x_{\text{att}}+b_0)+rb'}$ by first computing $r_{\text{att}}(b_1 x_{\text{att}} + b_0) + rb'$ in $\mathbb{Z}_p$ and then exponentiating h with the result, rather than computing this as $h^{r_{\text{att}}(b_1 x_{\text{att}}+b_0)+rb'} = (h^{b_1})^{r_{\text{att}} x_{\text{att}}} (h^{b_0})^{r_{\text{att}}} (h^{b'})^r$. While the former only costs one exponentiation and three multiplications in $\mathbb{Z}_p$, the latter requires a three-base exponentiation, which is generally much less efficient. In optimizing the order of computations, it is important to know the efficiency of the operations in the various groups. For instance, $\prod_{i\in\Upsilon} (\hat{e}(K_{1,i}, C_{1,i}) \cdot \hat{e}(K_{2,i}, C_{2,i}))^{\varepsilon_i}$ is often optimized to $\prod_{i\in\Upsilon} \hat{e}(K_{1,i}^{\varepsilon_i}, C_{1,i}) \cdot \hat{e}(K_{2,i}^{\varepsilon_i}, C_{2,i})$, because two exponentiations in $\mathbb{G}$ are more efficient than one exponentiation in $\mathbb{G}_T$. While this is the case for the curves considered in this work, it might be the case that, for some curves, it is more efficient to do one exponentiation in $\mathbb{G}_T$ instead of two in $\mathbb{G}$. Furthermore, we show in Section 6.2.6 that the optimal order may depend on the distribution of the key and ciphertext components over the two source groups.

6.2.5 Our optimization approaches for specific design goals

In optimizing the ABE schemes, we consider various approaches based on specific real-world design goals. In particular, this influences the conversion of the schemes to the type-III setting, but possibly also the choice of an elliptic curve. For pairing-based schemes in general, such conversions from the type-I to the type-III setting were previously considered in [AGH13, AGOT14, AGH15, AHO16a], which all automate this effort and which focus mostly on other predicate encryption primitives such as identity-based encryption [Sha84]. However, these frameworks only optimize the parameter sizes, and not necessarily the computational costs of the algorithms. While, e.g., optimizing the ciphertext size also results in an optimized efficiency of the encryption algorithm, such approaches might not necessarily lead to an optimized decryption algorithm. Furthermore, depending on the application in which ABE is going to be deployed, a practitioner may prefer a more balanced approach, in which the total costs of, e.g., the encryption and decryption algorithms are optimized rather than either one of them. To this end, we define the following optimization approaches based on design goals.

- **Optimized key generation (OK):** optimize the efficiency of the key generation algorithm;
- **Optimized encryption (OE):** optimize the efficiency of the encryption algorithm;
- **Optimized decryption (OD):** optimize the efficiency of the decryption algorithm;
- **Balanced key generation/encryption (BKE):** optimize the average costs of the key generation and encryption algorithms;
- **Balanced encryption/decryption (BED):** optimize the average costs of the encryption and decryption algorithms.

A practitioner can also devise optimization approaches for other design goals, e.g., "balanced key generation/decryption". In general, a practitioner can specify any goal in which the average/total costs of any subset of algorithms is minimized. Even though these may be useful in practice, the conversion techniques used in these approaches are likely similar to those used in the aforementioned approaches.

The importance of computational-efficiency focused approaches. In contrast to most conversion frameworks [AGOT14, AGH15, AHO16a], we do not necessarily describe our approaches in terms of the sizes of the public keys, secret keys or ciphertexts, but rather in terms of the computational costs of the algorithms like [AGH13]. However, due to the fact that the smallest group $\mathbb{G}$ also provides the most efficient arithmetic, we estimate that our optimized encryption and key generation approaches coincide with the optimized ciphertext and secret key size approaches in [AGH15]. The other three design goals, on the other hand, do not seem to match with any of the approaches in these conversion frameworks [AGOT14, AGH15, AHO16a], even though these may be of interest to practitioners. For instance, optimizing either the key generation or encryption algorithm may result in a heavy performance penalty on the other algorithms, while a balanced approach would ensure that an algorithm can perform efficiently while only requiring minimal sacrifice in efficiency on the other algorithms. Furthermore, we show that an optimal key or ciphertext size does not necessarily imply an optimal decryption cost, but requires a more intricate, in-depth analysis of the decryption algorithm and the arithmetic provided by the chosen groups and pairing. At a high level, our approach is thus closer to [AGH13], which focuses on optimizing the computational efficiency of the scheme, albeit in an automated way. Another common denominator between [AGH13] and our work is that the converted scheme is not automatically secure, though we argue that the converted schemes are secure nonetheless. An advantage of our type-conversion techniques over [AGH13] is that we take into account the costs of the arithmetic and group operations in our optimizations.

6.2.6 Our type-conversion methods

We describe our type-conversion methods, which can be used to convert a scheme from the type-I to the type-III setting given some specific design goal (as discussed in Section 6.2.5). We assume that the scheme to be converted is given in the type-I setting, and that it can be somewhat freely converted to the type-III setting without breaking its security. This is often the case: schemes are predominantly designed in the type-I setting [Sco11], but the symmetry of the pairing in many cases is not needed for their security [AGH13]. The symmetry is, on the other hand, to some extent important for the correctness. Specifically, the key and ciphertext components that are paired during decryption need to live in different source groups. If these two paired components live in the same group, then this yields incorrectness of decryption, for the simple reason that they cannot be paired. In addition, when full-domain hashes are used, we are slightly more limited, since the components involving these need to be placed in the same source groups (see Remark 6.1). In sum, while we have much freedom in how we convert from the type-I to the type-III setting, we are bounded by the correctness of the scheme.

Furthermore, as we mentioned, conversion from the type-I to the type-III setting is not trivial, as any conversion heavily influences the efficiency of a scheme. Hence, we ideally want to apply this conversion in the optimal way considering the optimization approach (associated with the chosen design goal) and the correctness of the decryption algorithm. For instance, if we want to convert some scheme to the type-III setting such that it has the most efficient encryption algorithm (i.e., as in the OE approach), then we attempt to place as many ciphertext components in the first group $\mathbb{G}$ as possible. This consequently means that the key components that are paired with these ciphertext components need to be placed in the second group $\mathbb{H}$. For the other approaches, the conversion is often more intricate, and requires knowledge of the computational costs in the groups $\mathbb{G}$, $\mathbb{H}$ and $\mathbb{G}_T$.

For each optimization approach, we follow the same steps:

(1) We first list the secret key and ciphertext components, and order them in such a way, that it is clear which components are paired during decryption such that we can maintain correctness of decryption;

(2) We specify for each key-ciphertext component pair whether they need to be exponentiated and whether they occur in a product during decryption;

(3) We determine the computational costs, for each key and ciphertext component, of the key generation and encryption algorithm;

(4) To determine the computational costs of the decryption algorithm, the order of the computations needs to be optimized (Section 6.2.4), which depends on the curve, and possibly the distribution of the components;

(5) Based on this information, we can determine the best possible distribution of the key and ciphertext components over the two source groups for a specific optimization approach. We describe how this can be done below.

Remark 6.1: Full-domain hashes

Because no full-domain hashes $\mathcal{H}_1 \colon \{0,1\}^* \rightarrow \mathbb{G}$ and $\mathcal{H}_2 \colon \{0,1\}^* \rightarrow \mathbb{H}$ exist such that $e(\mathcal{H}_1(\text{att}), h) = e(g, \mathcal{H}_2(\text{att}))$ holds [GPS08], we need to place the key and ciphertext components involving the FDH in the same group. As a consequence, we have less flexibility in optimizing the schemes using FDHs for its large-universeness.

Optimized encryption. Given the list of paired key-ciphertext components, the strategy is simple: we place as many ciphertext components in the first source group as possible. Because we need to place the ciphertext components involving the FDH in $\mathbb{G}$, we also place the key components involving an FDH in $\mathbb{G}$, and thus place the ciphertext components paired with these key components in $\mathbb{H}$.

Optimized key generation. Similarly, given the list of paired key-ciphertext components, the strategy is simple: we place as many key components in the first source group as possible. Similarly as in the optimized encryption approach, we always place ciphertext components involving an FDH in $\mathbb{G}$, and thus place the key components paired with these ciphertext components in $\mathbb{H}$.

Optimized decryption. To optimize decryption, we need to take a more careful approach. First, we need to consider whether group elements need to be exponentiated during decryption, because they occur in a shared-argument pairing product (see Section 6.2.4), e.g., $\prod_j \hat{e}(K', C_j)^{\varepsilon_j} = \hat{e}(K', \prod_j C_j^{\varepsilon_j})$. In this case, our conversion consists of placing the shared argument in $\mathbb{H}$ and the other—which needs to be exponentiated—in $\mathbb{G}$. For all key-ciphertext component pairs that do not occur in a shared-argument pairing product, it does not matter whether the key or ciphertext component is placed in $\mathbb{G}$, as long as any potential exponentiation happens in the first source group. In these cases, we will place, by default, the ciphertext component in $\mathbb{G}$, as it is oftentimes more important to optimize the encryption algorithm than the key generation algorithm. If the application allows the use of precomputation tables for all key components, we may also choose to place the key component in $\mathbb{G}$, and perform the exponentiations with a fixed-base exponentiation.

Balanced key generation/encryption. For a balanced efficiency of the key generation and encryption algorithms, we optimize the total key generation and encryption costs. We do this by considering the computational costs for each key-ciphertext component pair. For each pair, we place the component with the highest computational costs in $\mathbb{G}$, and the other in $\mathbb{H}$. For instance, if the pair (K, C) (like in our example) is such that the computation of K only requires a fixed-base exponentiation, and the computation of C requires a multi-base exponentiation, then we place C in $\mathbb{G}$ and K in $\mathbb{H}$. In this approach, it is also important to consider whether the pairs occur in a shared-argument pairing product during decryption. In this case, we place the shared argument in $\mathbb{H}$ and the other components in $\mathbb{G}$. Therefore, computing the shared argument incurs only a constant cost in $\mathbb{H}$ (during key generation or encryption), while computing the arguments on the other side incurs a linear cost in $\mathbb{G}$ (during encryption or key generation), subsequently optimizing the total costs of these computations.

Balanced encryption/decryption. Similarly, for a balanced efficiency of the encryption and decryption algorithms, we optimize the total encryption and decryption costs. This may be a slightly more complicated endeavor than the balanced key generation/encryption approach due to the more complicated nature of the optimized decryption strategy. Like in this strategy, we need to take into account whether a key-ciphertext component pair occurs in a shared-argument pairing product or not. In this case, it is beneficial for the decryption costs to place the shared argument in $\mathbb{H}$ and the other components in $\mathbb{G}$. However, this may more negatively affect the encryption costs than that it positively affects the decryption costs. For instance, suppose that the coefficients ε_j are small, e.g., $\varepsilon_j \in \{0, 1\}$ like in [LW10a]. Then, $\prod_j \hat{e}(C', K_{\rho(j)})^{\varepsilon_j}$ can be computed as $\hat{e}(\prod_j K_{\rho(j)}^{\varepsilon_j}, C')$ to minimize the decryption costs, requiring a linear number of multiplications in $\mathbb{G}$. However, this ensures that C' is in $\mathbb{H}$, and therefore likely costs at least one exponentiation in $\mathbb{H}$ instead of $\mathbb{G}$

Table 6.2. A list of the key-ciphertext component pairs and their costs incurred in computing them during key generation and encryption, in terms of fixed-base exponentiations (FBE) and two-base exponentiations (2-MBE). For the decryption costs, we list whether the pairing is indexed and whether it needs to be exponentiated.

Key component	Costs	Ciphertext component	Costs	Decryption	
				Indices	Exponentiation
K	FBE	C'	FBE	-	-
K'	FBE	$C_{1,j}$ $(j \in [n_1])$	2-MBE	$j \in \Upsilon \subseteq [n_1]$	ε_j
K_{att} $(\text{att} \in \mathcal{S})$	FBE	$C_{2,j}$ $(j \in [n_1])$	FBE	$j \in \Upsilon \subseteq [n_1]$	ε_j

(depending on the computational costs of C'). If the expected average costs incurred by the multiplications needed during decryption is lower than the costs incurred by computing C', we might want to place C' in $\mathbb{G}$ and place $K_{\rho(j)}$ in $\mathbb{H}$.

6.2.7 Example: type-converting Wat11

We explain our type-conversion techniques through an example: by converting the CP-ABE scheme by Waters (Wat11) [Wat11] from the type-I to the type-III setting. We first show how to convert the small-universe version of Wat11, and then argue how these conversions translate to the large-universe version of Wat11. A description of the Wat11 scheme can be found in Construction 2.1.

Listing the key and ciphertext components. To convert the scheme to the type-III setting, we first consider which key components need to be paired with which ciphertext components (see Table 6.2). In this way, we can ensure that each pair has exactly one component in each source group.

Optimized encryption and key generation. For the optimized encryption and key generation approaches, it is clear in which source groups the components need to be placed. Because the scheme does not involve hashing into the group, we have much freedom. For the optimized encryption approach, we can simply place all ciphertext components in $\mathbb{G}$, and the key components in $\mathbb{H}$. Conversely, for the optimized key generation approach, we can place all key components in $\mathbb{G}$ and the ciphertext components in $\mathbb{H}$.

Balanced key generation/encryption. For a more balanced approach in the efficiency of key generation and encryption, we take into account the number of components on the "other side of the pairing" during decryption. For instance, if one places K' in $\mathbb{G}$, then all $C_{1,j}$ need to be placed in $\mathbb{H}$, blowing up the encryption costs considerably. Hence, we place K' in $\mathbb{H}$ and $C_{1,j}$ in $\mathbb{G}$ to make the key generation and encryption costs more balanced. For the $(K_{\text{att}}, C_{1,j})$ key-ciphertext component pair, there is no such trade-off, as both cost one fixed-base exponentiation. In this case, we favor the encryption algorithm (as mentioned in Section 6.2.6), as it is probably run more often than the key generation algorithm. (Note, however, that one may want to take a different approach, and favor the key generation over the encryption algorithm instead.) For this reason, we place $C_{1,j}$ in $\mathbb{G}$ and K_{att} in $\mathbb{H}$. Thus, the optimized encryption and the balanced key generation/encryption efficiency approaches yield the same constructions, since all key components are placed in $\mathbb{H}$.

Optimized decryption. To optimize the decryption algorithm, we need to consider the best order of the operations performed during decryption, i.e.,

$$C / \left(\hat{e}(C', K) \cdot \left(\hat{e}\left(\prod_{j \in \Upsilon} C_{1,j}^{\varepsilon_j}, K' \right) / \prod_{j \in \Upsilon} \hat{e}\left(C_{2,j}^{\varepsilon_j}, K_{\rho(j)} \right) \right) \right).$$

Because a pairing operation is usually one of the most expensive operations, we want to minimize the use of these. Consequently, we use a shared-argument pairing and place the exponentations in $\mathbb{G}$ (Section 6.2.4). To ensure this, it is therefore better to put $C_{1,j}$ in $\mathbb{G}$ and K' in $\mathbb{H}$. For the other product of pairing operations, i.e., $\prod_{j \in \Upsilon} \hat{e}\left(C_{2,j}^{\varepsilon_j}, K_{\rho(j)}\right)$, it does not matter in which groups $K_{\rho(j)}$ and $C_{2,j}$ live, as we can exponentiate in $\mathbb{G}$, regardless of whether $K_{\rho(j)}$ or $C_{2,j}$ is in it. If, on the other hand, one is willing to use precomputation tables for all key components K_{att}, then we can speed up decryption by placing K_{att} in $\mathbb{G}$. Because this may require a large amount of precomputation space, this may, however, not be desirable in practice. Hence, we do not use precomputation, and, as mentioned in Section 6.2.6, we choose to favor the encryption efficiency over the key generation efficiency. We thus place the ciphertext component $C_{2,j}$ in $\mathbb{G}$ and K_{att} in $\mathbb{H}$.

Balanced encryption/decryption. Because the type conversion is the same for the optimized encryption and decryption approaches, it is, by extension, also the same for the balanced encryption/decryption approach. This is because, for each key-ciphertext component pair, we chose the best distribution of the two source groups with respect to the encryption and decryption efficiency. This therefore also yields the best efficiency trade-offs for the two.

Overview of the distributions for each optimization approach. In Table 6.3, we summarize the distributions of the key and ciphertext components for each approach. In particular, it shows that the distributions are the same for the optimized encryption, optimized decryption, balanced key generation/encryption and balanced encryption/decryption approaches.

Wat11-IV: the large-universe variant. The large-universe variant of the Waters scheme (Wat11-IV) [Wat08] replaces the generator $g^{b_{\text{att}}}$ by the output of a hash

Table 6.3. The distributions of the key and ciphertext components of Wat11-I over the groups $\mathbb{G}$ and $\mathbb{H}$, for each optimization approach, i.e., optimized encryption (OE), optimized key generation (OK), optimized decryption (OD), balanced key generation/encryption (BKE) and balanced encryption/decryption (BED).

Key	**Group**					**Ciphertext**	**Group**				
component	OE	OK	OD	BKE	BED	**component**	OE	OK	OD	BKE	BED
K	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	C'	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$
K'	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$C_{1,j}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$
K_{att}	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$C_{2,j}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$

Table 6.4. The distributions of the key and ciphertext components of Wat11-IV over the groups $\mathbb{G}$ and $\mathbb{H}$, for each optimization approach.

Key	**Group**					**Ciphertext**	**Group**				
component	OE	OK	OD	BKE	BED	**component**	OE	OK	OD	BKE	BED
K	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	C'	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$
K'	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$C_{1,j}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$
K_{att}	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$	$\mathbb{G}$	$C_{2,j}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$	$\mathbb{H}$

function, i.e., $\mathcal{H}(\text{att})$, where $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}$ denotes a hash function that maps arbitrary strings randomly in the group $\mathbb{G}$. The advantage of this is that the scheme can support any arbitrary string as attribute without requiring to change the master public key to be updated. Compared to the original, small-universe variant of the scheme, little needs to change. However, the use of a hash into the group gives us a little less freedom in the conversion from the type-I to the type-III setting. By Remark 6.1, we necessarily place the key and ciphertext components involving the hash in the same source group. For the optimized encryption and optimized key generation approaches, it is evident that these therefore need to be placed in the first source group (see Table 6.1). For optimized decryption, it follows from the $(K', C_{1,j})$ key-ciphertext component pair—which occur in a shared-argument pairing product—that the components involving the hash need to be placed in the first source group. That is, K' needs to be placed in $\mathbb{H}$ because it is the shared argument in the shared-argument pairing product, while $C_{1,j}$—which involves the hash—needs to be placed in $\mathbb{G}$. By extension, this requires K_{att} to be placed in $\mathbb{G}$ as well, because it involves a hash. The distribution of the key and ciphertext components is therefore almost entirely fixed for all optimization approaches. We can only choose the distribution of components K and C'. Because these incur a constant cost, the key generation, encryption and decryption costs are almost entirely fixed as well. Table 6.4 describes the distributions of the components over the two source groups of Wat11-IV. As it shows, the distributions are the same for the pairs $(K', C_{1,j})$ and $(K_{\text{att}}, C_{2,j})$. For the pair (K, C'), the distribution is the same as for Wat11-I.

6.2.8 Selecting the best elliptic curve for a specific goal

To fully optimize a scheme, it is important that the best curve is selected for each scheme and for each design goal. In general, this may not be the same curve for each scheme and each design goal, as the different choices of curves provide different trade-offs in efficiency. For instance, BN382 provides efficient hashing in the two source groups, while BLS12-381 provides efficient exponentiations and pairing operations [Ara17]. It may therefore be the case that ABE schemes that require many hashing operations are more efficient on the BN382 curve, while schemes that do not require these perform better on BLS12-381 curves. More generally, curves exist that provide more efficient arithmetic in $\mathbb{G}$ [CDS20] or that provide more efficient products of pairings [GF16]. However, these are unfortunately not supported by RELIC.

To determine the optimal curve for each scheme and each design goal, we compare the efficiency of the scheme on several curves providing the same level of security. To this end, we compare the computational costs of several ABE schemes on the curves providing the same level of security supported by RELIC.

6.3 Benchmarking

We show how our framework can be applied to several existing ABE schemes.

6.3.1 The schemes

In this chapter, we analyze and implement several selectively secure ciphertext-policy ABE schemes. We have motivated our choice to implement CP-ABE schemes in Section 6.1.2, and we will motivate the choice to implement selectively secure schemes below. The schemes that we implement are the previously considered small and large universe variants of Wat11 called Wat11-I and Wat11-IV (Constructions 4.1 and 4.7), RW13 (Construction 4.2), and the small and large-universe variants of AC17 (Constructions 4.3 and 4.8). We use our compiler in Section 4.6 to achieve provably selectively secure schemes. Descriptions of the type-converted variants of the schemes can be found in the paper [dlPVA22].

On the security of these schemes. We briefly discuss the security of the implemented schemes.

Selective versus full security. We consider the selectively secure variants of Wat11, RW13 and AC17, because this yields a cleaner comparison of the schemes on a structural level. In contrast, many fully secure variants of these schemes exist, which are similar to Wat11, RW13 and AC17 on a structural level, but these differ in the underlying groups. For instance, LOSTW10 [LOS$^+$10] is a fully secure variant of Wat11 and is instantiated in composite-order groups. For the same security level, LOSTW10 performs one to two orders of magnitude worse than Wat11, which can be instantiated in a prime-order group [Gui13]. Other fully secure variants of Wat11 [Att19, KW19b], which allow for instantiation in prime-order groups, might simply use different underlying group structures, which may affect the efficiency as well. However, this difference in efficiency might then be (partially) attributed to the choice of underlying groups, and not necessarily the different structures of the schemes. For a fair comparison of two structurally different schemes, one could first compare the efficiency of the selectively secure variants, instantiated in prime-order groups. Then, one can extrapolate the comparison to the full-security setting by considering the efficiency of the chosen underlying groups, which can be chosen the same if all the compared schemes have a fully secure counterpart in the same framework, e.g., [Att14a, Wee14, CGW15, Att16, Att19]. Note that this is the case for Wat11, RW13 and AC17, which have instantiations in the pair encodings framework (Chapter 4).

Security in idealized models. The security of the implemented schemes depends on idealized models, such as the random oracle model (ROM) [BR93] and the generic group model (GGM) [Sho97]. In particular, the large-universe variants of Wat11 and AC17 model the FDH as a random oracle in the proofs. Furthermore, the security of all three schemes depends on a q-type assumption, i.e., the (d_1, d_2)-parallel DBDH assumption (Definition 4.9). As shown in Lemma 4.5, we lose at most $\log_2(\sqrt{d_1 + d_2 + 1})$ bits of security, where d_1 and d_2 are as in the selective proofs of Lemmas 4.3 and 4.4. Because the maximum policy size in this chapter is 100, the maximum security loss is therefore at most 4 bits for Wat11 and AC17 and 7 bits for RW13.

6.3.2 Implementation

We have implemented the schemes in RELIC [AGM$^+$], and benchmarked the implementations using the AMD Ryzen 7 PRO 4750 processor with power management disabled (one single core) and throttle at max. frequency (4.1 GHz). We refer to the paper [dlPVA22] for more information on the implementations.

6.3.3 Performance analysis of our implementations

We analyze the performance of the implementations in our framework and compare it with those in existing frameworks such as Charm [AGM$^+$13] and OpenABE [Zeu20]. Our goal with this comparison is not necessarily to illustrate that our implementations are faster than those of Charm and OpenABE. Rather, we want to show that the choice of optimization approach as well as the use of all available optimized arithmetic influences this analysis. Not consistently and systematically using these may result in an unfair comparison of multiple schemes.

Comparing our framework with Charm and OpenABE. One of the main goals of our framework is to fully optimize the efficiency of all the schemes with respect to

Table 6.5. Comparison of the computational costs in milliseconds (for a processor with a frequency of 4.1 GHz) of the Charm and OpenABE implementations of Wat11 and RW13, and our implementations, on the BN254 curve.

Scheme/ Variant	**Key generation**			**Encryption**			**Decryption**		
	# of attributes			# of attributes			# of attributes		
	1	10	100	1	10	100	1	10	100
Wat11-I									
Charm	1.5	5.1	41.3	4.6	19.4	166.6	27.5	113.6	1010.4
OE	0.1	0.3	2.8	0.1	0.6	5.3	0.2	0.6	5.1
OK	0.0	0.2	1.3	0.2	1.3	11.7	0.2	0.6	6.0
RW13									
Charm	2.4	11.8	104.9	5.0	21.9	189.2	40.3	332.0	3255.5
OE	0.1	0.6	5.8	0.1	0.8	7.0	0.2	1.0	9.7
OK	0.1	0.3	2.8	0.2	1.5	14.5	0.2	1.1	11.0
Wat11-IV									
OpenABE	0.2	0.6	4.7	0.3	1.5	14.0	0.7	2.6	23.0
OE	0.1	0.4	3.9	0.2	0.9	8.3	0.2	0.6	5.0
OK	0.1	0.4	4.0	0.2	0.9	8.4	0.2	0.6	5.2

the same design goal. To show how effective this is compared to existing works, we have run benchmarks of the Charm [AGM$^+$13] implementations of Wat11-I and RW13 as well as the implementations of our optimizations. We have also run benchmarks of the OpenABE [Zeu20] implementation of Wat11-IV as well as the implementations of our two optimizations. Because the implementations in Charm and OpenABE support, at best, the BN254 curve [BN05], we compare the computational costs of the schemes on this curve. The results in Table 6.5 show that our implementations greatly improve on the implementations of Charm, the costs being at least one order of magnitude lower. For decryption, our implementations perform even a factor 100-300 faster than the Charm implementation. In particular, Charm takes several seconds to execute decryption for large policies with RW13, which increases even further if more up-to-date curves such as BLS12-381 are used. In contrast, our implementations never require more than 15 *milliseconds* to execute. Compared to the OpenABE implementation of Wat11-IV, our implementations perform roughly equally efficient in the key generation, a factor 1.6 faster in the encryption algorithm, and a factor 4 faster in the decryption.

In addition, Table 6.5 illustrates that comparing optimized implementations of the schemes yields a different comparison of two schemes. For instance, the benchmarks for Wat11-I and RW13 show that Wat11-I outperforms RW13 in all algorithms. We show in Section 6.3.4 that Wat11-IV—the large-universe variant of Wat11-I—is actually slower than Wat11-I, and is even outperformed by RW13 in the key generation and encryption algorithms for the OK and OE approaches, respectively. This difference in results illustrates that it is important to compare two implementations of schemes with the same properties, which are subsequently optimized with respect to the same goals. If this is not done, then one might unjustifiably draw the conclusion that Wat11-IV is a more efficient scheme (given any design goal) than RW13.

Table 6.6. Comparison of the computational costs expressed in 10^3 clock cycles for each scheme and optimization approach (OA) for 100 attributes, on their most efficient curve in the $[125, 128]$-bit security range. For each scheme, we determine the most efficient variant, and the increase (in percentages) that the other variants incur compared to the most efficient variant. The lowest costs are typeset in **bold**.

Scheme	**OA**	**Curve**	**Key generation**		**Encryption**		**Decryption**	
			Costs	Increase	Costs	Increase	Costs	Increase
Wat11-I	OE	BLS12-381	25653	143.0%	**39951**	-	**58515**	-
	OK	BLS12-381	**10555**	-	101181	153.3%	63151	7.9%
Wat11-IV	OE	BLS12-381	42275	0.3%	**77641**	-	**58290**	-
	OK	BLS12-381	**42135**	-	77898	0.3%	58441	0.3%
RW13	OE	BLS12-381	51401	137.3%	**54491**	-	**112072**	-
	OK	BLS12-381	**21657**	-	128221	135.3%	118998	6.2%
AC17	CP	BLS12-381	10696	0.6%	29352	0.0%	9027	123.8%
	OE	BLS12-381	25471	139.5%	**29348**	-	13736	240.6%
	OK	BLS12-381	**10635**	-	76067	159.2%	13696	239.6%
	OD	BN382	16184	52.2%	41776	42.3%	**4033**	-
AC17-LU	CP	BLS12-381	42632	1.7%	52752	1.1%	9106	124.4%
	OE	BLS12-381	42196	0.7%	**52176**	-	9060	123.3%
	OK	BLS12-381	**41913**	-	52326	0.3%	9076	123.7%
	OD	BN382	45093	7.6%	59276	13.6%	**4058**	-

Comparing schemes for different optimizations. In Section 6.2.5, we explained that the chosen design goal influences the optimization approach, including the conversion strategy from the type-I to the type-III setting. To this end, we have converted each scheme with respect to the different design goals. We illustrate the trade-offs incurred by the conversions in Table 6.6. It shows that, indeed, the variant of a scheme that is optimized with respect to a specific algorithm also outperforms the other variants in this algorithm. Note that, as expected, for the schemes without an FDH, these differences are much more pronounced than the schemes with an FDH.

6.3.4 Proof of concept: comparison of large-universe schemes

We also show how the benchmarks can be used in the comparison of different schemes. We do this by comparing the computational costs of the schemes for the same optimization approaches. Specifically, we compare the large-universe schemes, Wat11-IV, RW13 and AC17-LU with one another to investigate which of the three performs best with respect to some chosen optimization approach. Table 6.7 shows that AC17-LU outperforms the other two in almost all optimizations and the subsequent implementations of the algorithms. The only exception is the optimized key generation approach, where RW13 provides the most efficient key generation algorithm, outperforming the other two schemes by a factor 2. It therefore seems that, currently, RW13 is the best choice when the design goal is to have an optimized key generation algorithm. For the other approaches, it is best to use AC17-LU. Notably, RW13 outperforms

Table 6.7. Comparison of the computational costs for each large-universe scheme and optimization approach (OA) for 100 attributes, on their most efficient curves in the [125, 128]-bit security range. For each optimization approach, we determine the most efficient scheme, and the increase (in percentages) that the other schemes incur compared to the most efficient variant (which costs are typeset in **bold**). The costs are expressed in 10^3 clock cycles.

OA	**Scheme**	**Curve**	**Key generation**		**Encryption**		**Decryption**	
			Costs	Increase %	Costs	Increase %	Costs	Increase %
OE	Wat11-IV	BLS12-381	42275	0.2%	77641	48.8%	58290	543.4%
	RW13	BLS12-381	51401	21.8%	54491	4.4%	112072	1137.1%
	AC17-LU	BLS12-381	**42196**	-	**52176**	-	**9060**	-
OK	Wat11-IV	BLS12-381	42135	94.6%	77898	48.9%	58441	543.9%
	RW13	BLS12-381	**21657**	-	128221	145.0%	118998	1211.2%
	AC17-LU	BLS12-381	41913	93.5%	**52326**	-	**9076**	-
OD	Wat11-IV	BLS12-381	**42275**	-	77641	42.5%	58290	1336.5%
	RW13	BLS12-381	51401	21.6%	**54491**	-	112072	2661.9%
	AC17-LU	BN382	45093	6.7%	59276	8.8%	**4058**	-

Wat11-IV in the OE, OK and BKE (which yields the same computational costs as OD) approaches, and thus constitutes not only a scheme that is interesting for its theoretical properties, but also for its efficiency.

6.4 Future work

This work provides the basis for further research at various levels.

6.4.1 Automating our framework

Our type-conversion methods are heuristic and manual. The reason why they are heuristic is because the conversion methods are inextricably intertwined with the efficiency of the arithmetic not only provided by the chosen curve, but also by the order of the computations and the implementation of the arithmetic (which might in turn depend on the architecture of the processor). Currently, automating our conversion techniques is not trivial. Due to the heuristic nature of our given methods (which requires us to circle back to earlier design choices to see if these need to be adjusted), it may be more difficult to automate than like in [AGOT14, AGH15, AHO16a]. Instead, one could take a different approach. First, one could make a theoretical estimation of the computational costs of the arithmetic and group operations, for all possible pairing-friendly curves at the desired security level. Second, one could make a list of all possible distributions of the key and ciphertext components over the source groups. For each distribution and pairing-friendly curve, one can then determine the most efficient algorithms for the group operations and optimize the order of computations (which is not a trivial effort either). Given some optimization goal, the most efficient distribution can then be selected to be the optimal type conversion.

6.4.2 More pairing-friendly curves

In our analysis, we have only considered two curves in the $[125, 128]$-bit security range and two curves in the $[129, 135]$-bit security range. As we mentioned in the introduction, many pairing-friendly groups exist that provide at least 128 bits of security [Gui20a]. Notably, the KSS16 curves [KSS08] provide efficient arithmetic in the first source group and efficient multi-pairing operations [GF16, Ara17, CDS20]. These might be especially beneficial for schemes such as RW13, which provide much freedom with respect to their type conversion. In order to improve the benchmarks in this framework, more curves need to be supported by RELIC. Alternatively, a framework or library can be set up with the estimated efficiency of frequently-used arithmetic of the curves providing 128 bits of security listed at [Gui20a]. This would also help in any automated efforts.

6.4.3 Improving usability, validity and verifiability

To simplify the accurate comparison of schemes even further, it is important to make the framework more usable for ABE designers. As a result, cryptographers can easily compare their new scheme with existing ones in a transparent way without requiring a deep understanding of cryptographic engineering. One could make the framework more usable by providing a functionality that allows designers to specify, e.g., an encoding of a scheme (rather than a full-fledged description). In this way, it also becomes easier to analyze the schemes with respect to other metrics, such as validity and verifiability, e.g., either manually [VA21] or even automatically [ABGW17].

6.4.4 Implementing fully secure ABE

We have implemented only the selectively secure variants of Wat11, RW13 and AC17. While this provides a reliable comparison of the structure of the schemes, in practice, we prefer the use of a fully secure scheme. Since several frameworks exist that provide efficient generic conversions to the full-security setting (Chapter 4), it would be important to benchmark the underlying groups used in such conversions. In this way, the most efficient security-conversion technique can be selected. Note that those generic conversions are also compatible with the aforementioned encodings (Section 6.4.3).

6.4.5 Using other algorithms for group operations

We have used RELIC for the implementations of the group operations used in the ABE schemes. As mentioned in Section 6.2.1, RELIC does not support all available algorithms, e.g., it does not support fixed-argument pairings. Furthermore, using precomputation tables in multiple-base exponentiations may significantly speed up the encryption algorithm [Möl01]. Conversely, implementing ABE in resource-constrained devices may require the use of different optimizations [FA17].

6.4.6 Other pairing-based ABE, and related primitives

Our methods are mostly targeted at optimizing pairing-based ABE of the specific structure considered in this chapter. While this covers many ABE schemes, some schemes exist that do not have this exact structure, e.g., [LW11a, RW15]. Furthermore, our methods are also applicable to other pairing-based primitives that satisfy the targeted structure [AC17b, Att19], e.g., identity-based encryption [Sha84, BF01] and identity-based broadcast encryption [Del07]. Possibly, our framework can be expanded to cover pairing-based cryptography for other structures (and primitives) as well.

6.5 Conclusion

We have presented ABE Squared, a framework for accurately benchmarking efficiency of attribute-based encryption. Concretely, this framework aims to optimize the theoretical descriptions of ABE schemes for some chosen design goal by considering four optimization layers. These layers consider the arithmetic and group operations, the chosen pairing-friendly group, the order of the computations and the conversion techniques. By taking into account all layers during the optimization of a theoretical description, we are able to attain more efficient implementations. More specifically, we have devised several optimization approaches that aim to accomplish some chosen design goal, e.g., optimized key generation, encryption or decryption. By optimizing multiple schemes with respect to the same goal, they can be compared more fairly. Because existing conversion techniques did not allow us to, e.g., optimize the decryption algorithm, we have given new heuristic and manual techniques that facilitate this. Unlike other existing works, these conversion techniques take into account the other three optimization layers.

To show the effectiveness of our framework, we have optimized and implemented five schemes: Wat11-I, Wat11-IV, RW13, AC17 and AC17-LU. These implementations show that, indeed, the efficiency of the schemes depends heavily on the design goals and subsequent optimization approaches. For example, Charm shows that Wat11-I is generally faster than RW13. This may result in the idea that Wat11-IV, the large-universe variant of Wat11-I, is also faster than RW13, because it is similar. In contrast, we have shown that RW13 outperforms Wat11-IV with respect to the optimized encryption and optimized key generation approaches. This illustrates clearly that taking into account any such design goals in the implementation and benchmarks is crucial in the comparisons as well. Therefore, the ABE Squared framework provides an instrumental contribution in the benchmarking—and eventually, in the deployment—of ABE.

Part III

New constructions and generic transformations

Chapter 7

GLUE: generalizing unbounded ABE for flexible efficiency trade-offs

CP-ABE is a versatile primitive that has been considered extensively to securely manage data in practice. Especially completely unbounded schemes are attractive, because they do not restrict the sets of attributes and policies. So far, any such schemes that support negations in the access policy or that have online/offline extensions have an inefficient decryption algorithm.

In this chapter, we propose GLUE (Generalized, Large-universe, Unbounded and Expressive), which is a novel scheme that allows for the efficient implementation of the decryption while allowing the support of both negations and online/offline extensions. We achieve these properties simultaneously by uncovering an underlying dependency between encryption and decryption, which allows for a flexible trade-off in their efficiency. For the security proof, we devise a new technique that enables us to generalize multiple existing schemes. As a result, we obtain a completely unbounded scheme supporting negations that, to the best of our knowledge, outperforms all existing such schemes in the decryption algorithm.

7.1 Introduction

To securely and efficiently implement access control on data, especially pairing-based CP-ABE proves to be an attractive primitive [BSW07, KL10, SRGS12]. In 2018, ETSI published two technical reports on ABE [ETS18a, ETS18b], which include detailed descriptions of use cases, varying from cloud settings to mobile networks. In such settings, the computational resources of the key generation, encryption and decryption devices may vary. Thus, different use cases may require schemes with different efficiency trade-offs.

According to ETSI, ABE schemes should be efficient and secure. Interestingly, while ETSI proposes ABE to be used to enforce ABAC [HFK$^+$19] on data, it explicitly notes that ABE cannot satisfactorily support it, because ABE cannot support negations efficiently [ETS18b]. Indeed, the decryption algorithm of most ABE schemes supporting negations is incredibly expensive [OSW07, LSW10, YAHK14, Att19]. Recently, some interesting progress was made, yielding significant speed-ups in decryption time [TKN20, AT20]. However, those schemes still have a costly (although less costly) decryption [AT20] or restrict the attribute sets [TKN20].

In this chapter, we introduce a new scheme that enables the realization of the following properties:

(1) **Large-universe:** any string can be used as an attribute;

(2) **Unbounded:** no restrictions on, e.g., the sizes of the policies or attributes sets, or the number of times that an attribute may occur in the policy;

(3) **Expressive:** support of MSPs, ensuring that policies represented as Boolean formulas consisting of conjunctions and disjunctions can be supported;

(4) **Non-monotone:** support of non-monotone span programs, ensuring that the policies can use negations;

(5) **Compact:** the key and ciphertext sizes depend, in the worst case, on the size of the set or the length of the policy.

Additionally, this scheme offers a flexible choice in the encryption/decryption efficiency trade-off during the setup of the parameters. In this way, the scheme can be fine-tuned to take into account the computational resources of the key generation, encryption and decryption devices. In particular, this feature allows for significant speed-ups in the decryption compared to other schemes that also satisfy the listed properties.

Achieving properties (1)-(5) simultaneously. Only a limited number of existing schemes support properties (1)-(5) simultaneously (Chapter 2). In fact, all pairing-based schemes that provide non-monotonicity and large-universeness use a polynomial-based hash—also known as a "Boneh-Boyen (BB) hash" [BB04]—that maps arbitrary attribute strings into the scheme (Sections 2.5.5 and 2.5.7). Of those schemes, the only ones that are completely unbounded [LSW10, YAHK14, Att19, AT20] are based on LW11b (the KP-ABE version) [LW11b] and RW13 (the CP-ABE version) [RW13] (Table 7.1). However, all those schemes have an inefficient decryption compared to ABE schemes that use a full-domain hash (FDH) to achieve large-universeness (Section 6.3.4). For this reason, such schemes are often favored in practice, despite their inability to support negations [ETS18a, ETS18b]. Nevertheless, since supporting negations fosters the expressivity of ABE, we aim at improving the decryption efficiency of schemes using a BB hash.

Table 7.1. Comparison of large-universe schemes supporting (N)MSPs. For each scheme, we list whether it is unbounded (in the sets $\mathcal{S}$ and policies $\mathbb{A}$, and the number of attribute and label re-uses), whether it supports negations or has a provably secure extension that supports negations, and whether it is compact. Note that we have only listed schemes that have a unique associated pair encoding.

Scheme	**KP/CP**	**Unbounded**				**Negations**			**Compact**
		$\lvert\mathcal{S}\rvert$	$\lvert\mathbb{A}\rvert$	ARU	LRU	OSW	OT	OSWOT	
[GPSW06a, §5]	KP	✗	✓	✓	✓	✓ [OSW07]	✗	✗	✓
[BSW07]	CP	✓	✓	✓	✓	✗	✗	✗	✓
[ALdP11]	KP	✗	✓	✓	✓	✓	✗	✗	✗
[Wat11, §6]	CP	✓	✓	✓	✓	✗	✗	✗	✓
[Wat08, §B]	CP	✗	✗	✗	✗	✗	✗	✗	✓
[LW11b, RW13]	KP	✓	✓	✓	✓	✓ [LSW10]	✓ [Att19]	✓ [AT20]	✓
[OT12]	CP	✓	✓	✗	✗	✗	✓	✗	✗
[RW13]	CP	✓	✓	✓	✓	✓ [YAHK14]	✓ [Att19]	✓ [AT20]	✓
[AHM$^+$16]	KP	✓	✓	✓	✓	✓ [ALdP11]	✗	✗	✗
[AC16]	CP	✗	✗	✓	✓	✓ [Att19, Amb21]	✓ [AT20, Amb21]	✓ [AT20, Amb21]	✗
[AC17a]	CP(/KP)	✓	✓	✗	✗	✗	✗	✗	✓
[ABGW17, §5.3]	KP/CP	✓	✓	✓	✓	✗	✗	✗	✓
[Att19, §A-I]	CP(/KP)	✓	✓	✓	✓	✓	✗	✓ [AT20]	✓
[Att19, §A-II]	CP(/KP)	✗	✓	✓	✓	✓	✗	✗	✗
[Att19, §A-III]	CP(/KP)	✓	✗	✓	✓	✓	✗	✗	✗
[TKN20]	KP/CP	✓	✓	✓	✗	✗	✓	✗	✓
GLUE	CP(/KP)	✓	✓	✓	✓	✓	✓	✓[AT20]	✓

Note: ARU = attribute re-use; LRU = label re-use (in the sets and policies)

Improving decryption efficiency of schemes using a BB hash. To determine whether we can improve on the decryption efficiency of the existing schemes satisfying properties (1)-(5), we investigate *all* schemes using a BB hash to achieve large-universeness. In particular, if we consider all such schemes, then we see that a scheme that is unbounded, compact and costs less than one pairing operation per attribute during decryption does not exist yet (Figure 7.1). Because pairing operations are the most expensive operations in pairing-based ABE, it is therefore important to minimize the use of those. In this chapter, we aim to achieve this: we provide a scheme that satisfies properties (1)-(5), while requiring less than one pairing operation per attribute during decryption.

7.1.1 Our contributions

We first give a high-level overview of our contributions. Then, we provide more (technical) details about these contributions.

- **New construction:** We present a new unbounded large-universe scheme using a BB hash (thus avoiding random oracles). Its encryption/decryption efficiency trade-off can be fine-tuned by taking into account the computational resources of the devices.
- **Generalizations:** Concretely, the scheme can be considered a generalization of two large-universe schemes: the Rouselakis-Waters (RW13) scheme [RW13] and

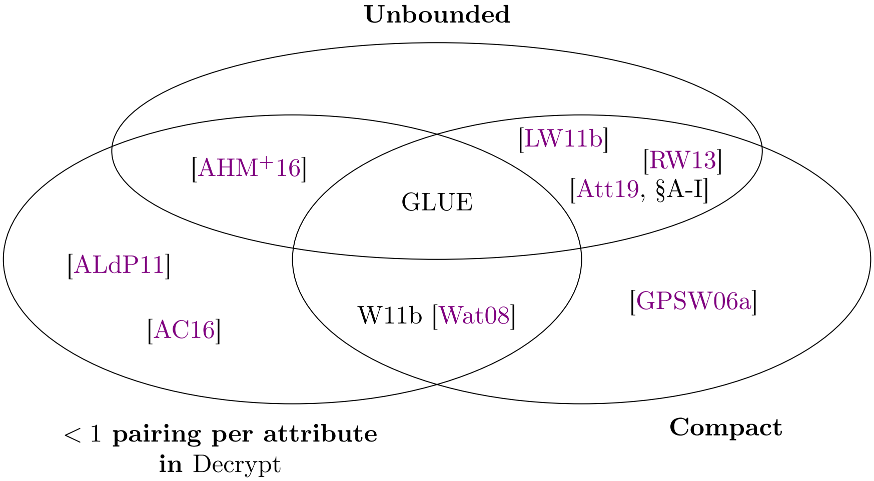


Figure 7.1. Overview of large-universe schemes using a BB hash.

the bounded large-universe scheme without random oracles by Waters (W11b) [Wat08]. This generalization also illustrates a deeper connection among various designs.

- **Security proof:** We develop new proof techniques to ensure that the randomness provided by a BB hash can be simultaneously used for the keys and ciphertexts. To the best of our knowledge, we are the first to achieve this in the unbounded setting, and in the full-security setting.
- **Extensions:** We provide three extensions to the basic scheme: one online/offline [HW14] and two non-monotone extensions supporting OT-type and OSW-type negations, respectively. Notably, we obtain an online/offline ABE scheme and a scheme supporting OSW-type negations with the most efficient decryption algorithms. This enables us to support OSWOT-type negations more efficiently, which is the most desirable in practice.

7.1.2 New construction: GLUE

We focus on three schemes that satisfy at least two out of the three depicted properties (see Figure 7.1): W11b [Wat08], RW13 [RW13] and AHM+16 [AHM+16]. Those three schemes provide a good starting point for GLUE, our new scheme which satisfies all required properties. Intuitively, we apply the partitioning techniques of AHM+16 to combine the unbounded RW13 and the bounded W11b that allows for efficient decryption. However, as we show later, for the secure combination of these techniques, GLUE requires a more intricate approach.

We give a high-level description of the partitioning approach as introduced by AHM+16. First, we partition the attribute sets into smaller subsets. Then, we apply a (bounded) scheme with efficient decryption (in their case, ALP11 [ALdP11]) to each subset. Lastly, we use the unbounded techniques of e.g., RW13 or LW11b [LW11b] to securely connect the subsets. In this way, the decryption costs of the scheme can be decreased. Unfortunately, this comes at a cost. Because bounded schemes typically have a more expensive encryption, the encryption costs are increased. From a broader perspective, this approach creates a scheme with a (flexible) efficiency trade-off feature. As we will show later, this trade-off is determined by some parameter n. The encryption costs are higher by a factor n than those of unbounded schemes such as RW13, whereas the decryption costs are lower by this factor. Because this parameter n can be chosen during setup, it can be fine-tuned for the given practical context. If decryption needs to be efficient (which is often the case), one can choose larger n than in cases in which encryption needs to be efficient.

The main reason why we achieve the compactness property, contrary to AHM+16, is due to the bounded scheme that is used. Because AHM+16 uses ALP11 [ALdP11], a scheme with constant-size ciphertexts and large keys whose sizes depend on the parameter n, its keys are large and its key generation is very expensive. Moreover, although the number of pairing operations required during decryption decreases, the number of exponentiations grows by a factor n for each matching attribute. As we will show, this means that AHM+16 decryption is not much more efficient than unbounded schemes such as RW13. As a solution, we use the W11b scheme, whose decryption costs consist of a constant number of pairing operations and no additional exponentiations. In this way, we achieve a much better speed-up in decryption, and since W11b is compact, the key sizes and key generation costs are not affected.

7.1.3 Generalizing RW13 by generalizing the hash

The main difference between the partitioning approach as applied by AHM+16 and us is that we have to partition both the key sets and the ciphertext policies. The reason for this is that AHM+16 uses ALP11, which is bounded in only the ciphertexts, whereas we use W11b, which is bounded in both the keys and ciphertexts. By extension, we need to apply some technique to connect the resulting key and ciphertext "parts". However, we will show that the security proof of W11b does not generalize to the unbounded setting, meaning that we have to devise a new proof technique for W11b that does generalize. Furthermore, it is not possible to apply the exact same approach as that of AHM+16. In particular, to prove security, they embed the scheme in the fully secure key-policy doubly-spatial encryption [Ham11] scheme in [Att14a], and then, they apply the embedding lemma [AHM^{+}16]. We cannot use this approach, because, to the best of our knowledge, the W11b scheme cannot be embedded in an existing scheme in a similar fashion.

Hence, we take a slightly different approach: we generalize RW13 by generalizing its specific instantiation of the BB hash. A BB hash is a hash as in the polynomial-based approach to support large universes (Section 2.5.5), i.e., $F_n(x_{\text{att}}) = \prod_{i=0}^{n} B_i^{x_{\text{att}}^i}$, where the generators $B_i = g^{b_i}$ implicitly embed the coefficients of the polynomial $f_n(x_{\text{att}}) = \sum_{i=0}^{n} b_i x_{\text{att}}^i$. Where RW13 (and its unbounded derivatives [HW14, Att19]) uses an implicit 1-degree polynomial, we use an implicit n-degree polynomial, like W11b [Wat08]. However, we will show that simply replacing the 1-degree polynomial by some n-degree polynomial does not immediately yield a secure scheme. To solve this, we replace another public-key variable used in the scheme by a polynomial.

7.1.4 Security proof

One of the main difficulties of our scheme is proving the selective and co-selective property. In the first place, proving security is difficult due to the lack of provably secure schemes that use the randomness provided by the BB hash for *both the keys and ciphertexts*. To the best of our knowledge, previously, only W11b [Wat08] used the hash for both the keys and ciphertexts, but only in the bounded setting and in the selective-security model. However, the proof does not seem to readily generalize to the unbounded setting (see the full version [VA22b]). Hence, we develop a novel technique to prove security. We do this, in part, by combining several techniques.

- **Proof techniques using the hash for the keys:** We generalize the proof techniques used by Agrawal and Chase in [AC17b] to prove full security of their scheme in [AC16]: the AC16 [AC16] scheme. AC16 is a CP-ABE scheme with constant-size ciphertexts, in which the randomness provided by the Boneh-Boyen hash is used for the *keys*. In the selective proof, the polynomial embedded in the public keys needs to be used by the secret keys after the public keys are generated. The proof does this by embedding a "reprogrammable" polynomial in the public keys. We call these polynomials to be reprogrammable in the sense that the randomizers of the secret keys can later program it to a suitable target polynomial. We use this general strategy for the keys.

- **Proof techniques using the hash for the ciphertext:** Even though the W11b [Wat08] proof does not generalize to the unbounded setting, we are able to use a part of the proof strategy. In the selective proof, the implicit polynomial embedded in the public key is "programmed" to take into account the attributes that will be used in the challenge ciphertext. We use this general strategy for the ciphertexts.

- **Unbounded proof techniques:** One of the bottlenecks of the two aforementioned strategies is that they are bounded approaches: they use only one randomizer for the keys and one for the ciphertexts. To make them unbounded, we use the general approach of the RW13 [RW13] proof. This proof gives us a rough idea of how the implicit polynomial and the randomizers should be programmed. Furthermore, it shows us how to use the polynomial an unbounded number of times: using layering and individual randomness techniques allows us to select the required instance of the polynomial.

Another bottleneck is that the "programmed" and "reprogrammable" approaches are orthogonal, and can therefore not be used simultaneously in the same polynomial without applying a trick. Presumably, this is also the reason why the W11b proof uses the programmed approach for both the keys and the ciphertexts, and applies an algebraic argument to ensure that everything can be simulated as required. We eliminate this bottleneck and combine all these proof techniques, by splitting the polynomial into the product of two smaller polynomials: one "programmed" polynomial and one "reprogrammable" polynomial. For the selective proof, we use the programmed polynomial for the ciphertexts and the reprogrammable polynomial for the keys. For the co-selective proof, the roles of the polynomials are reversed.

7.1.5 Practical extensions

We provide several extensions to our scheme. Because we prove security in the AC17 [AC17b] framework, some of these extensions are automatically provably secure.

- **The key-policy and dual-policy versions**, by applying [Att19]. The key-policy version can be found in the full version [VA22b];
- **Online/offline extensions**, by generalizing [HW14], in Section 7.5.1. Owing to its generality, these extensions also apply to the following extensions;
- **Non-monotone versions**, by applying [Amb21, Att19, TKN20]:
  - **OT-type:** the PES can be found in the full version [VA22b];
  - **OSW-type:** the PES can be found in Section 7.5.2.

7.1.6 Efficiency comparison with schemes supporting (1)-(5)

We generalize RW13 to achieve a scheme that supports or can support properties (1)-(5) whilst being able to achieve a more efficient decryption algorithm. In Table 7.2, we compare the efficiency of RW13 and its OSW-type non-monotone variant Att19-I-CP with GLUE (which supports MSPs only) and GLUE-N (which additionally supports OSW-type negations). In Section 7.6, we give more concrete estimates for the timings in practice and how they compare to existing schemes.

7.2 Generalizing RW13

We first show how RW13 can be generalized. On a high level, we do this by substituting the implicit 1-degree polynomial in the RW13 keys and ciphertexts with an

Table 7.2. Theoretical efficiency comparison of all (selectively secure) compact unbounded large-universe CP-ABE schemes supporting monotone span programs (that support or can support OSW(OT)-type negations), by analyzing the key generation, encryption and decryption costs with respect to the number of exponentiations c_{exp} and pairings c_{pair}.

Scheme	**Key generation**	**Encryption**	**Decryption**	
	c_{exp}	c_{exp}	c_{exp}	c_{pair}
[RW13]	$2+2\lvert\mathcal{S}\rvert$	$2+5\lvert\mathbb{A}\rvert$	$2\lvert\Upsilon\rvert$	$2+2\lvert\Upsilon\rvert$
[Att19, §A-I]	$6+6\lvert\mathcal{S}\rvert$	$2+8\lvert\mathbb{A}\rvert$	$2\lvert\Upsilon\rvert$	$4+2\lvert\Upsilon\rvert$
GLUE	$2+\lvert\mathcal{S}\rvert+\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil$	$2+\lvert\mathbb{A}\rvert(n_k+2n_c+1)+\left\lceil\frac{\lvert\mathbb{A}\rvert}{n_c}\right\rceil$	$2\lvert\Upsilon\rvert$	$2+\left\lceil\frac{\lvert\Upsilon\rvert}{n_k}\right\rceil+\left\lceil\frac{\lvert\Upsilon\rvert}{n_c}\right\rceil$
GLUE-N	$6+4\lvert\mathcal{S}\rvert+2\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil$	$2+\lvert\mathbb{A}\rvert(n_k+2n_c+4)+\left\lceil\frac{\lvert\mathbb{A}\rvert}{n_c}\right\rceil$	$2\lvert\Upsilon\rvert$	$4+\left\lceil\frac{\lvert\Upsilon\rvert}{n_k}\right\rceil+\left\lceil\frac{\lvert\Upsilon\rvert}{n_c}\right\rceil$

(a) Costs for non-negated policies

Scheme	c_{exp}	c_{pair}
[Att19, §A-I]	$2\lvert\Upsilon\rvert\cdot\lvert\mathcal{S}\rvert$	$4+\lvert\Upsilon\rvert+\min(\lvert\Upsilon\rvert,\lvert\mathcal{S}\rvert)$
GLUE-N (worst case)	$2\lvert\Upsilon\rvert\cdot\lvert\mathcal{S}\rvert+\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil\cdot\lvert\Upsilon\rvert$	$4+\lvert\Upsilon\rvert+\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil$
GLUE-N (best case)	$2\left\lceil\frac{\lvert\Upsilon\rvert}{n_c}\right\rceil\cdot\lvert\mathcal{S}\rvert+\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil\cdot\lvert\Upsilon\rvert$	$4+\left\lceil\frac{\lvert\Upsilon\rvert}{n_c}\right\rceil+\left\lceil\frac{\lvert\mathcal{S}\rvert}{n_k}\right\rceil$

(b) Decryption costs for negated policies

Note: $\mathcal{S}$ = attribute set; $\mathbb{A}$ = access policy; Υ = matching attributes;
n_k, n_c = parameters chosen during the setup

n-degree polynomial. Like W11b, the randomness provided by this n-degree polynomial will be shared between the keys and ciphertexts. That is, suppose that n_k and n_c are positive integers such that $n = n_k + n_c - 1$, then the n-degree polynomial provides enough randomness for $n_k - 1$ attributes in the keys, and n_c attributes in the ciphertext. To optimally use this randomness, we therefore split the keys and ciphertexts into partitions of at most n_k and n_c attributes, respectively. For instance, if $\mathcal{S}$ denotes the set of attributes for which a key is requested, then $\mathcal{S}$ is split into partitions of maximum size n_k, i.e., $\mathcal{S} = \mathcal{S}_1 \cup ... \cup \mathcal{S}_m$ such that $|\mathcal{S}_l| \leq n_k$ for each $l \in [m]$. Then, to avoid boundedness, we apply the RW13 trick by introducing one "randomizer" for each partition (both in the keys and ciphertexts).

7.2.1 The RW13 scheme

The secret keys and ciphertexts of RW13 (Construction 4.2) are of the form

$$\mathrm{SK} = (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\mathrm{att}} = h^{rb' + r_{\mathrm{att}}(x_{\mathrm{att}} b_1 + b_0)}, K_{2,\mathrm{att}} = h^{r_{\mathrm{att}}}\}_{\mathrm{att} \in \mathcal{S}}),$$
$$\mathrm{CT} = (C = M \cdot e(g,h)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot (B')^{s_j},$$
$$C_{2,j} = (B_1^{x_{\mathrm{att}_j}} B_0)^{s_j}, C_{3,j} = g^{s_j}\}_{j \in [n_1]}),$$

where $B = g^b, B_1 = g^{b_1}, B_0 = g^{b_0}$ and $B' = g^{b'}$ denote public keys, and $x_{\mathrm{att}_j} = x_{\rho(j)}$.

7.2.2 First attempt: a naive approach

Our first attempt is to directly replace the 1-degree polynomial, $x_{\mathrm{att}}b_1 + b_0$, by an n-degree polynomial, i.e., $f_n(x_{\mathrm{att}}) = \sum_{i=0}^{n} b_i x_{\mathrm{att}}^i$ (where $n = n_k + n_c - 1$):

$$\mathrm{SK} = (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\mathrm{att}} = h^{rb' + \boxed{r_{\mathrm{att}} f_n(x_{\mathrm{att}})}}, K_{2,\mathrm{att}} = h^{r_{\mathrm{att}}}\}_{\mathrm{att}\in\mathcal{S}}),$$
$$\mathrm{CT} = (C = M \cdot e(g,g)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot (B')^{s_j},$$
$$C_{2,j} = \boxed{F_n(x_{\mathrm{att}_j})^{s_j}} = \left(\prod_{i=0}^{n} B_i^{x_{\mathrm{att}_j}^i}\right)^{s_j}, C_{3,j} = g^{s_j}\}_{j\in[n_1]}),$$

where $B_i = g^{b_i}$ for all $i \in [0, n]$. We split $\mathcal{S}$ into partitions of maximum size n_k, and the rows of $\mathbb{A}$ in partitions of size n_c. We ensure that the same randomizer is used for all attributes in the same partition, i.e., set $r_{\mathrm{att}} = r_{\mathrm{att}'}$ and $s_j = s_{j'}$, if att and att$'$, and att$_j$ and att$_{j'}$ are in the same partitions, respectively.

Unfortunately, the resulting scheme is insecure (see the full version [VA22b] for a concrete attack). Roughly, the reason is that $C_{1,j} = g^{\lambda_j b + s_j b'}$ does not sufficiently hide $\lambda_j b$, because the same s_j is used for all attributes in the same partition. Therefore, we need to introduce more randomness.

7.2.3 Second (successful) attempt

We show how to use another polynomial to introduce enough randomness. Because we only need enough randomness for the ciphertext partitions, we require an $(n_c - 1)$-degree polynomial. This polynomial, $f'_{n_c-1}(x_{\mathrm{att}}) = \sum_{i=0}^{n_c-1} b'_i x_{\mathrm{att}}^i$, will replace the "0-degree polynomial" b'. Because s_j provides randomness for one attribute, and f'_{n_c-1} provides randomness for $n_c - 1$ attributes in the partition, this sufficiently hides λ_j. The resulting scheme is then

$$\mathrm{SK} = (K = h^{\alpha - rb}, K' = h^r, \{K_{1,\mathrm{att}} = h^{\boxed{r f'_{n_c-1}(x_{\mathrm{att}})} + \boxed{r_{\mathrm{att}} f_n(x_{\mathrm{att}})}},$$
$$K_{2,\mathrm{att}} = h^{r_{\mathrm{att}}}\}_{\mathrm{att}\in\mathcal{S}}),$$
$$\mathrm{CT} = (C = M \cdot e(g,h)^{\alpha s}, C' = g^s, \{C_{1,j} = B^{\lambda_j} \cdot \boxed{F'_{n_c-1}(x_{\mathrm{att}_j})^{s_j}},$$
$$C_{2,j} = \boxed{F_n(x_{\mathrm{att}_j})^{s_j}}, C_{3,j} = g^{s_j}\}_{j\in[n_1]}),$$

where $F'_{n_c-1}(x_{\mathrm{att}}) = \prod_{i=0}^{n_c-1} (B'_i)^{x_{\mathrm{att}}^i} = g^{f'_{n_c-1}(x_{\mathrm{att}})}$, and $B'_i = g^{b'_i}$ for all $i \in \overline{[n_c - 1]}$. Note that this scheme generalizes RW13, because setting $n_c = n_k = 1$ yields RW13.

7.2.4 More efficient decryption

Generalizing the polynomial allows for an improved decryption efficiency. To understand why this yields a significant improvement, we briefly review the W11b scheme.

We consider the keys and ciphertexts, which are of the form:

$$\mathrm{SK} = (K = h^{\alpha - rb}, K' = h^r, \{K_{\mathrm{att}} = h^{rf_n(x_{\mathrm{att}})}\}_{\mathrm{att} \in \mathcal{S}}),$$
$$\mathrm{CT} = (C = M \cdot e(g,h)^{\alpha s}, C' = g^s, \{C_j = B^{\lambda_j} F_n(x_{\rho(j)})^s\}_{j \in [n_1]}),$$

where $r, s \in \mathbb{Z}_p$ are randomly chosen integers, $B = g^b$ is a public key, α is the master key. To decrypt, one computes

$$(C/e(C', K)) \cdot \left(\prod_{j \in \Upsilon} e(C_j, K')^{\varepsilon_j} / \prod_{j \in \Upsilon} e(C', K_{\rho(j)})^{\varepsilon_j} \right),$$

where ε_j for $j \in \Upsilon \subseteq [n_1]$ are integers that allow us to reconstruct the secret s. Each such product of pairings can be computed more efficiently by using the bilinearity property on the shared arguments, e.g., $\prod_j e(K', C_j)^{\varepsilon_j}$ can be computed more efficiently by first multiplying C_j and then taking a pairing:

$$C/e\left(C', K \cdot \prod_{j \in \Upsilon} K_{\rho(j)}^{\varepsilon_j}\right) \cdot e\left(\prod_{j \in \Upsilon} C_j^{\varepsilon_j}, K'\right).$$

This requires only two pairing operations instead of $2|\Upsilon|+1$. Roughly, the number of pairing operations grows in the number of randomizers in the keys, and in the ciphertexts. By replacing the 1-degree polynomial in RW13 by an n-degree polynomial, the numbers of randomizers in the keys and ciphertexts are reduced by a factor n_k and n_c, respectively. Thus, the number of pairing operations is reduced similarly.

Note that this also illustrates why it is important that the BB hash is used for both the keys and the ciphertexts. For example, in the GPSW06 large-universe scheme [GPSW06a, §5], the randomness provided by the hash is used only for the ciphertexts. As a result, the keys require a fresh randomizer for each attribute, and therefore, decryption costs at least one pairing operation per attribute.

7.3 Our construction

We now present the complete description of our scheme in the selective security setting (see the full version [VA22b] for a fully secure version). In this scheme, we also introduce the maps ι and τ, which map the attributes of the keys and ciphertexts, respectively, into arbitrary partitions of maximum sizes n_k and n_c.

Construction 7.1: GLUE

GLUE is defined as follows.

- Setup(λ): On input the security parameter λ, the setup generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}, h \in \mathbb{H}$, and chooses a pairing $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It also defines the universe of attributes $\mathcal{U} = \mathbb{Z}_p$, chooses $n_k \in \mathbb{N}$ and $n_c \in \mathbb{N}$ as the maximum partition sizes of the keys and ciphertexts, respectively, and sets $n = n_k + n_c - 1$. It then generates random $\alpha, b, b_i, b'_{i'} \in_R \mathbb{Z}_p$ for all $i \in [0, n], i' \in [0, n_c - 1]$. It outputs $\text{MSK} = (\alpha, b, \{b_i, b'_{i'}\}_{i \in [0,n], i' \in [0, n_c - 1]})$ as its master secret key and publishes the master public key as

$$\text{MPK} = (g, h, A = e(g,h)^{\alpha}, B = g^b, \{B_i = g^{b_i}, B'_{i'} = g^{b'_{i'}}\}_{i \in \overline{[n]}, i' \in \overline{[n_c - 1]}}).$$

- KeyGen(MSK, $\mathcal{S}$): On input set of attributes $\mathcal{S}$, the key generation computes $m = \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$, defines $\iota \colon \mathcal{S} \to [m]$ such that $|\iota^{-1}(l)| \leq n_k$ for each $l \in [m]$, generates random integers $r, r_1, ..., r_m \in_R \mathbb{Z}_p$, and outputs the secret key as

$$\text{SK}_{\mathcal{S}} = (K = h^{\alpha - rb}, K' = h^r, \iota,$$
$$\{K_{1,\text{att}} = h^{r_{\iota(\text{att})}(\sum_{i=0}^{n} b_i x_{\text{att}}^i) + r(\sum_{i=0}^{n_c - 1} b'_i x_{\text{att}}^i)}\}_{\text{att} \in \mathcal{S}}, \{K_{2,l} = h^{r_l}\}_{l \in [m]}).$$

- Encrypt(MPK, $\mathbb{A}$, M): A message $M \in \mathbb{G}_T$ is encrypted under policy $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho \colon [n_1] \to \mathcal{U}$ by computing $m' = \max\left(\left\lceil \frac{n_1}{n_c} \right\rceil, \max_{j \in [n_1]} |\rho^{-1}(\rho(j))|\right)$ and defining $\tau \colon [n_1] \to [m']$ such that $|\tau^{-1}(l')| \leq n_c$ for each $l' \in [m']$ and if $j, j' \in [n_1]$ with $j \neq j'$ such that $\rho(j) = \rho(j')$, then $\tau(j) \neq \tau(j')$, i.e., multiple occurrences of the same attribute are mapped to different partitions. (Note that this works because m' is defined to be at least as large as the maximum number of occurrences of each attribute.) It then generates random integers $s, s_1, ..., s_{m'}, v_2, ..., v_{n_2} \in_R \mathbb{Z}_p$ and outputs the ciphertext as

$$\text{CT}_{\mathbb{A}} = (C = M \cdot A^s, C' = g^s, \tau, \{C_{1,j} = B^{\lambda_j} \cdot \prod_{i=0}^{n_c - 1} (B'_i)^{s_{\tau(j)} \rho(j)^i},$$
$$C_{2,j} = \prod_{i=0}^{n} B_i^{s_{\tau(j)} x_{\rho(j)}^i}\}_{j \in [n_1]}, \{C_{3,l'} = g^{s_{l'}}\}_{l' \in [m']}),$$

such that λ_j denotes the j-th entry of $\mathbf{A} \cdot (s, v_2, ..., v_{n_2})^{\intercal}$.

- Decrypt($\text{SK}_{\mathcal{S}}$, $\text{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, and let $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$, such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$

(Definition 2.5). Then, the plaintext M is retrieved by computing

$$C/\Big(e(C',K)\cdot \textstyle\prod_{j\in\Upsilon}\big(e(C_{1,j},K')/e(C_{3,\tau(j)},K_{1,\rho(j)})\cdot e(C_{2,j},K_{2,\iota(\rho(j))})\big)^{\varepsilon_j}\Big).$$

This can be computed more efficiently as

$$\begin{aligned}C/\Big(e(C',K)\cdot e(\textstyle\prod_{j\in\Upsilon}C_{1,j}^{\varepsilon_j},K')\cdot\Big(\textstyle\prod_{l'\in[m']}e(C_{3,l'},\prod_{j\in\Upsilon\cap\tau^{-1}(l')}K_{1,\rho(j)}^{-\varepsilon_j})\\ \cdot\textstyle\prod_{l\in[m]}e(\prod_{j\in\Upsilon\cap\rho^{-1}(\iota^{-1}(l))}C_{2,j}^{-\varepsilon_j},K_{2,l})\Big)\Big),\end{aligned}$$

which costs, on average, $2+\left\lceil\frac{|\Upsilon|}{n_k}\right\rceil+\left\lceil\frac{|\Upsilon|}{n_c}\right\rceil$ pairing operations.

The scheme is correct, i.e., we have $C/e(C',K)=M\cdot e(g,h)^{\alpha s}\cdot e(g,h)^{-\alpha s+rsb}=M\cdot e(g,h)^{rsb}$ and

$$\begin{aligned}&\prod_{j\in\Upsilon}\big(e(C_{1,j},K')/e(C_{3,\tau(j)},K_{1,\rho(j)})\cdot e(C_{2,j},K_{2,\iota(j)})\big)^{\varepsilon_j}\\ &\qquad=\prod_{j\in\Upsilon}(e(g,h)^{r\lambda_j b+rs_{\tau(j)}\sum_{i=0}^{n_c-1}b'_i x^i_{\rho(j)}}\\ &\qquad\cdot e(g,h)^{-r_{\iota(\rho(j))}s_{\tau(j)}(\sum_{i=0}^{n}b_i x^i_{\rho(j)})-rs_{\tau(j)}\sum_{i=0}^{n_c-1}b'_i x^i_{\rho(j)}}\\ &\qquad\cdot e(g,h)^{r_{\iota(\rho(j))}s_{\tau(j)}\sum_{i=0}^{n}b_i x^i_{\rho(j)}})^{\varepsilon_j}\\ &=\prod_{j\in\Upsilon}e(g,h)^{r\varepsilon_j\lambda_j b}=e(g,h)^{rb\sum_{j\in\Upsilon}\varepsilon_j\lambda_j}=e(g,h)^{rsb},\end{aligned}$$

which yields the plaintext, i.e., $M\cdot e(g,h)^{rsb}/e(g,h)^{rsb}=M$.

Unique representation of attributes. In the scheme, we assume that any attribute string att $\in\{0,1\}^*$ can be uniquely represented in $\mathbb{Z}_p$. In practice, this can be done by using a collision-resistant hash function $\mathcal{H}\colon\{0,1\}^*\to\mathbb{Z}_p$ [SW05].

7.3.1 The associated pair encoding scheme

To prove security, we define the pair encoding scheme associated with our scheme in Construction 7.1, for which we use the variables $n_c, n_k, n,\mathcal{S},\iota,\rho,\tau,\ n_1, n_2, \lambda_i, m, m'$ from Construction 7.1, as follows.

Construction 7.2: PES for GLUE

- Param($\emptyset$): Let $\mathbf{b}=(b,b_0,...,b_n,b'_0,...,b'_{n_c-1})$, where $n=n_c+n_k-1$.

- EncKey($\mathcal{S}$): Let $\mathbf{r} = (r, \{r_l\}_{l\in[m]})$, and $\mathbf{k} = (k', \{k_{1,\text{att}}\}_{\text{att}\in\mathcal{S}})$, where $k' = \alpha - rb$ and $k_{1,\text{att}} = r_{\iota(\text{att})}(\sum_{i=0}^{n} b_i x_{\text{att}}^i) + r(\sum_{i=0}^{n_c-1} b'_i x_{\text{att}}^i)$.
- EncCt($(\mathbf{A}, \rho)$): Let $\mathbf{s} = (s, \{s_{l'}\}_{l'\in[m']})$ and $\hat{\mathbf{s}} = (\hat{v}_2, ..., \hat{v}_{n_2})$, and $\mathbf{c} = (\{c_{1,j}, c_{2,j}\}_{j\in[n_1]})$, where $c_{1,j} = \mathbf{A}_j(sb, \hat{\mathbf{s}})^\intercal + s_{\tau(j)} \sum_{i=0}^{n_c-1} b'_i x_{\rho(j)}^i$ and $c_{2,j} = s_{\tau(j)} \sum_{i=0}^{n} b_i x_{\rho(j)}^i$.

In Section 7.4, we prove security of the PES. It follows from Theorem 4.1 that instantiating the PES with Construction 4.10 is selectively secure.

Theorem 7.1

The PES for GLUE in Construction 7.2 satisfies Sym-Prop (Definition 4.2).

Corollary 7.1

Because Construction 7.2 satisfies Sym-Prop, Construction 7.1 is selectively secure (Theorem 4.1).

7.4 The security proof

While the construction of the scheme already provides some idea on why it may be secure, the proof requires some additional insights. First, we briefly review some important aspects in the Rouselakis-Waters proof, to gain some deeper understanding of the structure of the selective property proof. Then, we show how existing techniques can be combined to generalize the selective proof.

On a high level, the selective proof consists of the splitting of the n-degree polynomial, which provides randomness for the keys *and* ciphertexts, into a product of three polynomials f_1, f'_{n_c-1} and g_{n_k-1}. We use g_{n_k-1} for the keys, and $f_1 f'_{n_c-1}$ for the challenge ciphertext. For the key polynomial g_{n_k-1}, we use Agrawal and Chase's [AC17c] techniques. In their selective proof of the CP-ABE scheme with short ciphertexts, they embed a polynomial in the public keys such that this polynomial can be reprogrammed to some polynomial associated with the set of attributes of the key. We call such polynomials "reprogrammable". For the ciphertext polynomials f_1, f'_{n_c-1}, we use a combination of the proofs of RW13 and W11b. Roughly, in these proofs, they embed polynomials in the public keys, such that these polynomials are associated with the attributes that occur in the challenge access policy. We call such polynomials "programmed". These techniques ensure that the polynomials evaluate to the right values when the set and policy attributes are evaluated. Figure 7.2 depicts the relationship between the existing proofs and ours. A similar approach can be taken in the co-selective proof, by swapping the roles of the two polynomials.

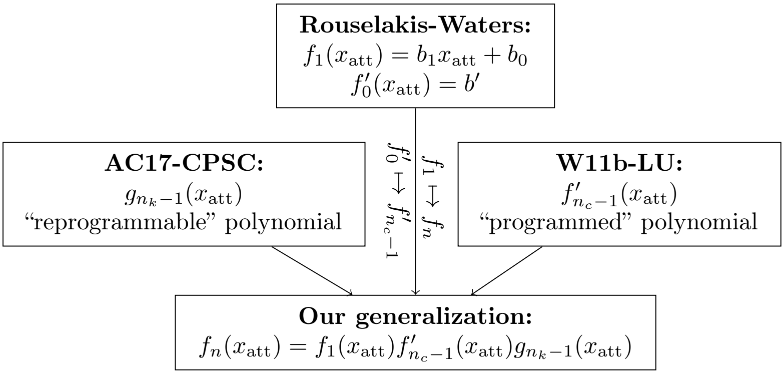

Rouselakis-Waters:
$f_1(x_{\text{att}}) = b_1 x_{\text{att}} + b_0$
$f'_0(x_{\text{att}}) = b'$
AC17-CPSC:
$g_{n_k-1}(x_{\text{att}})$
"reprogrammable" polynomial
$f'_0 \mapsto f'_{n_c-1}$
$f_1 \mapsto f_n$
W11b-LU:
$f'_{n_c-1}(x_{\text{att}})$
"programmed" polynomial
Our generalization:
$f_n(x_{\text{att}}) = f_1(x_{\text{att}}) f'_{n_c-1}(x_{\text{att}}) g_{n_k-1}(x_{\text{att}})$

Figure 7.2. A high-level overview of the polynomials used in the combined proofs and our generalized selective proof.

7.4.1 Generalizing the Rouselakis-Waters proof

We generalize the Rouselakis-Waters proof (see the proof of Lemma 4.4) by layering the policy embedded in the public keys in a partition-wise fashion instead of attribute-wise. In this way, the ciphertext-specific variable $s_{l'}$, which is used for all attributes in the same partition, can select all attributes associated within the l'-th partition. As such, in the computation of $c_{1,j}$ and $c_{2,j}$, we need $s_{\tau(j)} f'_{n_c-1}(x_{\rho(j)})$ to cancel out $\mathbf{A}_j(sb, \hat{\mathbf{s}})^\intercal$ and $s_{\tau(j)} f_n(x_{\rho(j)})$ needs to go to $\mathbf{0}$. To this end, we need to substitute f'_{n_c-1} in such a way that it outputs exactly $-\sum_{k \in [n_2]} A_{j,k} \overline{\mathbf{1}}^{d_2}_{(1,k)}$ when $s_{\tau(j)} f'_{n_c-1}(x_{\rho(j)})$ is computed. Similarly, the key-specific variable r_l needs to be constructed such that $k_{1,\text{att}}$ goes to $\mathbf{0}$, which happens when $r_{\iota(\text{att})} f_n(x_{\text{att}})$ cancels out $r f'_{n_c-1}(x_{\text{att}})$.

To accomplish this, we define f_n and f'_{n_c-1} as mentioned before, i.e., $f_n(x_{\text{att}}) = f_1(x_{\text{att}}) f'_{n_c-1}(x_{\text{att}}) g_{n_k-1}(x_{\text{att}})$. Roughly, we substitute f_1 in the same way as in the Rouselakis-Waters proof, while we use the polynomials f'_{n_c-1} and g_{n_k-1} to ensure that $c_{1,j}$ and $c_{2,j}$, and $k_{1,\text{att}}$ evaluate to $\mathbf{0}$, respectively. Because we are given the challenge access structure a priori, i.e., as input to EncB, we can program these as required in the substitutions of the polynomials f_1 and f'_{n_c-1} in the public keys. Concretely, we substitute $b_0, ..., b_n$ such that

$$\begin{aligned} f_n(x_{\text{att}}) \ &: \sum_{j \in [n_1], k \in [n_2]} A_{j,k} F_{n,j,k}(x_{\text{att}}) \\ = \sum_{j \in [n_1], k \in [n_2]} &A_{j,k} \underbrace{F_{1,j}(x_{\text{att}}) F'_{n_c-1,j}(x_{\text{att}}) \hat{G}_{n_k-1,j,k}(x_{\text{att}})}_{F_{n,j,k}(x_{\text{att}})}, \end{aligned}$$

where $F_{1,j}(x_{\text{att}}) = (x_{\text{att}} - x_{\rho(j)})$ and

$$F'_{n_c-1,j}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} d'_{i,j} x_{\text{att}}^i = \prod_{j' \in \chi_j \setminus \{j\}} \frac{x_{\text{att}} - x_{\rho(j')}}{x_{\rho(j)} - x_{\rho(j')}},$$

with $\chi_j = \{j' \in [n_1] \mid \tau(j') = \tau(j)\}$. We refer to $F_{1,j}$ and $F'_{n_c-1,j}$ as the "programmed" polynomials. These ensure that $F_{n,j}(x_{\rho(j')}) = 0$ for all $j' \in \chi_j$, $F'_{n_c-1,j}(x_{\rho(j)}) = 1$ and $F'_{n_c-1,j'}(x_{\rho(j)}) = 0$ for all $j' \in \chi_j \setminus \{j\}$. Then, $c_{1,j}$ and $c_{2,j}$ evaluate to $\mathbf{0}$, if we substitute

$$f'_{n_c-1}(x_{\text{att}}) \ : \sum_{j \in [n_1], k \in [n_2]} A_{j,k} F'_{n_c-1,j}(x_{\text{att}}) \mathbf{1}^{d_1 \times d_2}_{(1,\tau(j)),(1,k)}.$$

In contrast, the set of attributes associated with a key is given after the public keys have been established, i.e., as input to EncR, so we need to somehow achieve that we can program the polynomial $\hat{G}_{n_k-1,j,k}$ after the public keys are generated. We do this by setting

$$\hat{G}_{n_k-1,j,k}(x_{\text{att}}) = \sum_{i=0}^{n_k-1} \mathbf{1}^{d_1 \times d_2}_{(1,\tau(j)),(2,i,j,k)} x_{\text{att}}^i,$$

such that $\hat{G}_{n_k-1,j,k}$ constitutes a "reprogrammable" polynomial. It can be reprogrammed by ensuring that r_l consists of the coefficients $u_{i,j,l}$ of some target polynomial(s), i.e., by multiplying

$$\left(\sum_{i=0}^{n_k-1} \mathbf{1}^{d_1 \times d_2}_{(1,\tau(j)),(2,i,j,k)} x_{\text{att}}^i\right) \left(\sum_{i=0}^{n_k-1} u_{i,j,l} \overline{\mathbf{1}}^{d_2}_{(2,i,j,k)}\right) = \sum_{i=0}^{n_k-1} u_{i,j,l} \mathbf{1}^{d_1}_{(1,\tau(j))} x_{\text{att}}^i.$$

We use this to "reprogram" the polynomial $\hat{G}_{n_k-1,j,k}(x_{\text{att}})$ for all $j \in \overline{\Upsilon}$, which is well-defined, because $\rho(j) \notin \mathcal{S}$. This then yields $F'_{n_c-1,j}(x_{\text{att}})$ and cancels out the $F'_{n_c-1,j}(x_{\text{att}})$ in the $rf'_{n_c-1}(x_{\text{att}})$ part in $k_{1,\text{att}}$ for all $j \in \overline{\Upsilon}$. Note that, like in Rouselakis-Waters, we have $\mathbf{A}_j \mathbf{w}^\intercal = 0$ for all $j \in \Upsilon$, so those layers automatically go to $\mathbf{0}$ in the computation of $k_{1,\text{att}}$. Hence, for each partition $\Psi_l = \{\text{att} \in \mathcal{S} \mid \iota(\text{att}) = l\}$ with $l \in [m]$, we define the polynomial

$$G_{n_k-1,j,l}(x_{\text{att}}) = \sum_{i=0}^{n_k-1} u_{i,j,l} x_{\text{att}}^i = \sum_{\text{att}' \in \Psi_l} \frac{1}{F_{1,j}(x_{\text{att}'})} \prod_{\text{att}'' \in \Psi_l \setminus \{\text{att}'\}} \frac{x_{\text{att}} - x_{\text{att}''}}{x_{\text{att}'} - x_{\text{att}''}},$$

for each $j \in \overline{\Upsilon}$, such that $G_{n_k-1,j,l}(x_{\text{att}}) = \frac{1}{F_{1,j}(x_{\text{att}})}$ for all att $\in \Psi_l$.

Putting it all together, we substitute $b_0,\ldots,b_n$ with coefficients such that the polynomial f_n is substituted by

$$\sum_{j\in[n_1],k\in[n_2]} A_{j,k}F_{n,j,k}(x_{\text{att}}) = \sum_{j\in[n_1],k\in[n_2]} A_{j,k}\sum_{i=0}^{n} d_{i,j,k}x_{\text{att}}^{i},$$

where $d_{i,j,k} = \sum_{i'\in\overline{[n_k-1]},i''\in\overline{[n_c-1]}:i'+i''=i} d'_{i',j}\mathbf{1}^{d_1\times d_2}_{(1,\tau(j)),(2,i'',j,k)}$.

7.4.2 The selective symbolic property

We prove the selective symbolic property, using m, m', τ, ι as in Section 7.3 and $F_{n,j,k}, d_{i,j,k}, F'_{n_c-1,j}, d'_{i,j}, G_{n_k-1,j,l}, u_{i,j,l}$ and χ_j as in Section 7.4.1. Let $d_1 = n_2+1$ and $d_2 = ((n_k+1)n_1+1)n_2$. For simplicity of notation, we write indices in $[d_2]$ as a tuple $(1,k)$ or $(2,i,j,k)$ (with $i \in \overline{[n_k]}, j \in [n_1], k \in [n_2]$) such that it represents a unique integer in $[d_2]$. For the indices in $[d_1]$, we start counting at 0. The substitutions are, for all $i \in \overline{[n]}, i' \in \overline{[n_k]}, l \in [m], l' \in [m'], k \in [2, n_2]$:

$$b \;:\; \mathbf{1}^{d_1\times d_2}_{0,(0,1)}, \quad b_i \;:\; \sum_{j\in[n_1],k\in[n_2]} A_{j,k}d_{i,j,k}, \quad b'_{i'} \;:\; \sum_{j\in[n_1],k\in[n_2]} A_{j,k}d'_{i',j}\mathbf{1}^{d_1\times d_2}_{(1,\tau(j)),(0,k)},$$

$$s \;:\; \mathbf{1}^{d_1}_{0}, \quad s_{l'} \;:\; -\mathbf{1}^{d_1}_{(1,l')}, \quad \alpha \;:\; \mathbf{1}^{d_1}_{0}, \quad \hat{v}_k \;:\; \overline{\mathbf{1}}^{d_2}_{(0,k)}, \quad r \;:\; \sum_{k'\in[n_2]} w_{k'}\overline{\mathbf{1}}^{d_2}_{(0,k')}$$

$$r_l \;:\; -\sum_{i'\in\overline{[n_k-1]},j'\in\overline{\Upsilon},k'\in[n_2]} w_{k'}u_{i',j',l}\overline{\mathbf{1}}^{d_2}_{(1,i',j',k')}.$$

Then, $c_{1,j}, c_{2,j}, k'$ and $k_{1,\text{att}}$ indeed go to $\mathbf{0}$ (see the full version [VA22b]).

7.4.3 Co-selective symbolic property

We prove that the co-selective symbolic property also holds. For this proof, the roles of the reprogrammable and the programmed polynomial are reversed, because we are allowed to use an attribute set $\mathcal{S}$ in the programming of the public keys and secret keys, and the policy $\mathbb{A}$ only for the ciphertext. Similarly as in the selective case, we use the co-selective proof of RW13 (see the proof of Lemma 4.4) as inspiration. In this proof, we substitute the polynomials with:

$$f_n(x_{\text{att}}) \;:\; \sum_{l\in[m]} \left(\underbrace{\hat{F}_{n_c-1,1,l}(x_{\text{att}})G_{n_k,l}(x_{\text{att}})}_{G_{n,l}(x_{\text{att}})} - \hat{F}_{n_c-1,2,l}(x_{\text{att}}) \right),$$

$$f'_{n_c-1}(x_{\text{att}}) \;:\; \hat{F}_{n_c-1,2,0}(x_{\text{att}}),$$

where we define $G_{n,l}(x_{\text{att}}) = \sum_{i=0}^{n} \tilde{u}_{i,l} x_{\text{att}}^i$, and

$$G_{n_k,l}(x_{\text{att}}) = \sum_{i=0}^{n_k} u_{i,l} x_{\text{att}}^i = \prod_{\text{att}' \in \Psi_l} (x_{\text{att}} - x_{\text{att}'})$$

is a programmed polynomial with $G_{n_k,l}(x_{\text{att}}) = 0$ for att $\in \Psi_l$, and

$$\hat{F}_{n_c-1,1,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \mathbf{1}_{(1,i,l),l}^{d_1 \times d_2} x_{\text{att}}^i \text{ and } \hat{F}_{n_c-1,2,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \mathbf{1}_{(2,i),l}^{d_1 \times d_2} x_{\text{att}}^i$$

are the reprogrammable polynomials, to be reprogrammed to

$$F_{n_c-1,1,j,l}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \tilde{d}_{i,j,l} x_{\text{att}}^i = \frac{1}{G_{n_k,l}(x_{\rho(j)})} F'_{n_c-1,j}(x_{\text{att}})$$

$$\text{and } F'_{n_c-1,j}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} d'_{i,j} x_{\text{att}}^i,$$

respectively, for $j \in \overline{\Upsilon}$. Note that $F_{n_c-1,1,j,l}(x_{\rho(j)}) = \frac{1}{G_{n_k,l}(x_{\rho(j)})}$ if $j \in \overline{\Upsilon}$ and $F_{n_c-1,1,j',l}(x_{\rho(j)}) = 0$ for $j' \in \chi_j$. Concretely, we have

$$\tilde{u}_{i,l} = \sum_{i' \in \overline{[n_c-1]}, i'' \in \overline{[n_k]} : i'+i''=i} u_{i'',l} \mathbf{1}_{(1,i',l),l}^{d_1 \times d_2}.$$

Then, for $i \in \overline{[n_c - 1]}, i' \in [n_c, n], l \in [m], l' \in [m'], k \in [n_2]$ we make the following substitutions:

$$b \ : \ \mathbf{1}_{0,0}^{d_1 \times d_2}, \quad b_i \ : \ \sum_{l \in [m]} \left(\tilde{u}_{i,l} - \mathbf{1}_{(2,i),l}^{d_1 \times d_2} \right), \quad b_{i'} \ : \ \sum_{l \in [m]} \tilde{u}_{i,l}$$

$$b'_i \ : \ \mathbf{1}_{(2,i),0}^{d_1 \times d_2}, \quad \alpha \ : \ \mathbf{1}_0^{d_1}, \quad \hat{v}_k \ : \ w_k \overline{\mathbf{1}}_0^{d_2}, \quad r \ : \ \overline{\mathbf{1}}_0^{d_2}, \quad r_l \ : \ \overline{\mathbf{1}}_l^{d_2}$$

$$s \ : \ \mathbf{1}_0^{d_1}, \quad s_{l'} \ : \ - \sum_{i \in \overline{[n_c-1]}, j \in \hat{\chi}_{l'} \cap \overline{\Upsilon}, k \in [n_2]} A_{j,k} w_k \left(d'_{i,j} \mathbf{1}_{(2,i)}^{d_1} + \tilde{d}_{i,j} \right),$$

where $\tilde{d}_{i,j} = \sum_{l \in [m]} \tilde{d}_{i,j,l} \mathbf{1}_{(1,i,l)}^{d_1}$, $d_1 = n_c(m+1)+1$ and $d_2 = m+1$. For simplicity, we use tuple notations $(1, i, l)$ and $(2, i)$ for the indices in $[2, d_1]$, and index 0 maps to the first row. Then, $c_{1,j}, c_{2,j}, k'$ and $k_{1,\text{att}}$ indeed go to $\mathbf{0}$ (see the full version [VA22b]).

7.5 Extensions

We present an online/offline (Definition 3.13) and a non-monotone extension of GLUE.

7.5.1 Online/offline version of GLUE

In the online/offline version (Definition 3.13) of GLUE, the intermediate keys and ciphertexts are instantiated with random values rather than actual attributes. These random values are also included in the intermediate keys and ciphertexts, so that during online time, the difference between the random value and the used attribute can be computed. This difference is then included in the key or ciphertext, so that a regular key or ciphertext can be generated from the online/offline key or ciphertext.

Construction 7.3: GLUE-OO

- Setup(λ): This algorithm is the same as in Construction 7.1.
- Regular.KeyGen($\mathrm{MSK}, \mathcal{S}$): This algorithm is the same as in Construction 7.1.
- Offline.KeyGen(MSK): The algorithm generates two types of "intermediate secret keys".
  - *First type:* The algorithm generates random integer $r \in \mathbb{Z}_p$ and stores $\mathrm{ISK}_1 = (K = h^{\alpha - rb}, K' = h^r, r)$.
  - *Second type:* The algorithm generates random integers $r', z_j \in_R \mathbb{Z}_p$ for all $j \in [n_k]$, and stores $\mathrm{ISK}_2 = (\{\hat{K}_{1,j} = h^{z_j}, z_j\}_{j \in [n_k]}, K_2 = h^{r'}, r')$.
- Online.KeyGen($\mathrm{MSK}, \mathrm{ISK}, \mathcal{S}$): On input set of attributes $\mathcal{S}$, the algorithm computes $m = \left\lceil \frac{|\mathcal{S}|}{n} \right\rceil$, defines $\iota \colon \mathcal{S} \to [m]$ such that $|\iota^{-1}(l)| \leq n$ for each $l \in [m]$, and further defines $\hat{\iota} \colon \mathcal{S} \to [n_k]$ such that it is injective on each subdomain $\iota^{-l}(l)$ for all $l \in [m]$. It takes one intermediate secret key of the first type, and m of the second type:

  $$(K = h^{\alpha - rb}, K' = h^r, r), (\{\hat{K}_{1,j,l} = g^{z_{j,l}}, z_{j,l}\}_{j \in [n_k]}, K_{2,l} = h^{r_l}, r_l)_{l \in [m]},$$

  sets $\hat{K}_{1,\mathrm{att}} = \hat{K}_{1,\hat{\iota}(\mathrm{att}),\iota(\mathrm{att})}$, then computes

  $$\hat{K}_{3,\mathrm{att}} = \left(r_{\iota(\mathrm{att})} \sum_{i=0}^{n} b_i x_{\mathrm{att}}^i + r \sum_{i=0}^{n_c - 1} b'_i x_{\mathrm{att}}^i \right) - z_{\hat{\iota}(\mathrm{att}),\iota(\mathrm{att})}.$$

  and outputs the secret key as

  $$\mathrm{SK}_{\mathcal{S}} = (K, K', \iota, \{\hat{K}_{1,\mathrm{att}}, \hat{K}_{3,\mathrm{att}}\}_{\mathrm{att} \in \mathcal{S}}, \{K_{2,l}\}_{l \in [m]}).$$

- FinalStep.KeyGen($\mathrm{OO.SK}_{\mathcal{S}}$): The user can generate the secret keys as in Construction 7.1 from $(K, K', \iota, \{\hat{K}_{1,\mathrm{att}}, K_{2,l}, \hat{K}_{3,\mathrm{att}}\}_{\mathrm{att}\in\mathcal{S}, l\in[m]})$, by computing for each $\mathrm{att} \in \mathcal{S}$ the secret key component $K_{1,\mathrm{att}}$ as in Construction 7.1: $K_{1,\mathrm{att}} = \hat{K}_{1,\mathrm{att}} \cdot h^{\hat{K}_{3,\mathrm{att}}}$.

- Regular.Encrypt($\mathrm{MPK}, \mathbb{A}, M$): This algorithm is the same as in Construction 7.1.

- Offline.Encrypt(MPK) $\to \mathrm{CT}_{\mathbb{A}}$: The algorithm generates two types of "intermediate ciphertexts".

  – *First type:* It selects $s \in_R \mathbb{Z}_p$ and stores $(\hat{C} = A^s, C' = g^s, s)$.

  – *Second type:* It selects $s', \hat{\lambda}_1, ..., \hat{\lambda}_{n_c} \in_R \mathbb{Z}_p$, $(\hat{x}_{j,1}, ..., \hat{x}_{j,n}) \in_R \mathbb{Z}_p^n$, sets $\hat{x}_{j,0} = 1$ for all $j \in [n_c]$, and stores

  $(\{\hat{C}_{1,j} = B^{\hat{\lambda}_j} \cdot \prod_{i=0}^{n_c-1} (B'_i)^{s'\hat{x}_{j,i}}, \hat{C}_{2,j} = \prod_{i=0}^{n} B_i^{s'\hat{x}_{j,i}}, \hat{\lambda}_j, \{\hat{x}_{j,i}\}_{i\in\overline{[n]}}\}_{j\in[n_c]}, C_3 = g^{s'}, s')$.

- Online.Encrypt($\mathrm{MPK}, \mathrm{ICT}, \mathbb{A}, M$): On input policy $\mathbb{A} = (\mathbf{A}, \rho)$, the algorithm selects one "intermediate ciphertext" $(\hat{C}, C', s)$ of the first type, and then $m' = \left\lceil \frac{n_1}{n_c} \right\rceil$ "intermediate ciphertexts" of the second type

  $$(\{\hat{C}_{1,j,l'}, \hat{C}_{2,j,l'}, \hat{\lambda}_{j,l'}, \{\hat{x}_{j,i,l'}\}_{i\in[0,n]}\}_{j\in[n_c]}, C_{3,l'}, s_{l'})$$

  (for all $l' \in [m']$). It defines τ and λ_j as in Construction 7.1, and further defines $\hat{\tau}\colon [n_1] \to [n_c]$ such that $\hat{\tau}$ is injective on each subdomain $\tau^{-1}(l')$ with $l' \in [m']$, i.e., each attribute gets mapped to a unique tuple $(\hat{C}_{1,j,l'}, ..., s_{l'})$. It encrypts message M by setting $C = M \cdot \hat{C}$ and sets for all $j \in [n_1], i \in [n]$:

  $$\hat{C}_{1,j} = \hat{C}_{1,\hat{\tau}(j),\tau(j)}, \quad \hat{C}_{2,j} = \hat{C}_{2,\hat{\tau}(j),\tau(j)},$$
  $$\hat{C}_{4,j} = \lambda_j - \hat{\lambda}_{\hat{\tau}(j),\tau(j)}, \quad \hat{C}_{5,j,i} = s_{\hat{\tau}(j)}(x_{\rho(j)}^i - \hat{x}_{\hat{\tau}(j),i,\tau(j)}).$$

  The user publishes the ciphertext as

  $$\mathrm{CT}_{\mathbb{A}} = (C, C', \tau, \{\hat{C}_{1,j}, \hat{C}_{2,j}, \hat{C}_{4,j}, \hat{C}_{5,j,i}\}_{i\in[n], j\in[\hat{n}_1]}, \{C_{3,l'}\}_{l'\in[\hat{m}']}).$$

  Note that the ciphertext increases by $n_1(n+1)$ elements in $\mathbb{Z}_p$ compared to regular ciphertexts.

- Decrypt($\mathrm{MPK}, (\mathrm{OO.})\mathrm{SK}_{\mathcal{S}}, (\mathrm{OO.})\mathrm{CT}_{\mathbb{A}}$): If $\mathcal{S}$ satisfies $\mathbb{A}$, determine ε_j and Υ as in Construction 7.1, set $\Upsilon_l = \{j \in \Upsilon \mid \iota(\rho(j)) = l\}$ for all $l \in [m]$ and compute $C/\left(e(C', K) \cdot C_1 \cdot C_2 \cdot C_3\right)$, where

$$C_1 = e\left(\prod_{j\in\Upsilon} \hat{C}_{1,j}^{\varepsilon_j} \cdot (g^b)^{\sum_{j\in\Upsilon}\varepsilon_j \hat{C}_{4,j}} \cdot \prod_{i=1}^{n_c-1} (g^{b_i})^{\sum_{j\in\Upsilon}\varepsilon_j \hat{C}_{5,j,i}}, K'\right),$$

$$C_2 = \prod_{j\in\Upsilon} e(C_{3,\tau(j)}^{-\varepsilon_j}, K_{1,\rho(j)}), C_3 = \prod_{l\in[m]} e\left(\prod_{j\in\Upsilon_l} \hat{C}_{2,j}^{\varepsilon_j} \cdot \prod_{i=1}^{n}(g^{b_i})^{\sum_{j\in\Upsilon_l}\varepsilon_j \hat{C}_{5,j,i}}, K_{2,l}\right).$$

Correctness. Correctness of the decryption algorithm follows readily by showing that $\hat{C}_{1,j} \cdot (g^b)^{\hat{C}_{4,j}} \cdot \prod_{i=1}^{n_c-1}(g^{b_i})^{\hat{C}_{5,j,i}} = C_{1,j}$, and $\hat{C}_{2,j} \cdot \prod_{i=1}^{n}(g^{b_i})^{\hat{C}_{5,j,i}} = C_{2,j}$, where $C_{1,j}$ and $C_{2,j}$ are as in Construction 7.1 (see the full version [VA22b]). Then, the correctness proof is identical to that in this definition.

Security proof. The security proof for the online/offline version of GLUE is similar to the proof in [HW14], and can be found in the full version [VA22b]. In the proof, we reduce the security of the online/offline version to the security of the regular scheme. In this proof, we use the homomorphic properties of the scheme and the randomness of $\hat{\lambda}_i$ and $\hat{x}_{j,i}$.

7.5.2 GLUE version supporting OSW-type negations

We provide a provably secure generalized unbounded scheme supporting NMSPs with OSW-type negations. We obtain this scheme by applying the generic negation in [Amb21], and the direct sum transformation and ciphertext-policy augmentation (confined to OR) in [Att19] to GLUE (Construction 7.2). Note that this yields a generalized variant of Att19-CP-I [Att19]. The variables $n_c, n_k, n, \mathcal{S}, \rho, \iota, \tau, n_1, n_2, \lambda_i, m, m'$ are as in Construction 7.1. In this definition, we also include a function ρ_2 that maps the row to 1 if the attribute is not negated in the policy, and to 2 if it is negated.

For this scheme, we require Lagrange interpolation. That is, given $n+1$ points $(x, f_n(x))$, with $x \in \mathcal{S}$ and $|\mathcal{S}| = n+1$, on a polynomial. Then, we can reconstruct the the point $f_n(z)$ by computing

$$f_n(z) = \sum_{x\in S} \Lambda_{S,x} f_n(x) \pmod p, \text{ where } \Lambda_{S,x,z} = \prod_{y\in S\setminus\{x\}} \frac{z-y}{x-y} \pmod p.$$

Construction 7.4: GLUE-N

GLUE-N, which supports OSW-type negations, is defined as:

- Param(par): Let

$$\mathbf{b} = (b, b'', b^{(3)}, b_0, ..., b_n, \bar{b}_0, ..., \bar{b}_n, b'_0, ..., b'_{n_c-1}, \bar{b}'_0, ..., \bar{b}'_{n_c-1}),$$

where $n = n_c + n_k - 1$ with $n_k \geq n_c$, and

$$f_n(x_{\text{att}}) = \sum_{i=0}^{n} b_i x_{\text{att}}^i, f'_{n_c-1}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} b'_i x_{\text{att}}^i,$$
$$\bar{f}_n(x_{\text{att}}) = \sum_{i=0}^{n} \bar{b}_i x_{\text{att}}^i, \bar{f}'_{n_c-1}(x_{\text{att}}) = \sum_{i=0}^{n_c-1} \bar{b}'_i x_{\text{att}}^i.$$

- EncKey($\mathcal{S}$): Let $\mathbf{r} = (r, r', \bar{r}, \{r_l, \bar{r}_{\text{att}}, \bar{r}'_l\}_{\text{att}\in\mathcal{S}, l\in[m]})$, and let $\mathbf{k} = (k', k'', \bar{k}'', \{k_{1,\text{att}}, \bar{k}_{1,\text{att}}, \bar{k}_{2,\text{att}}\}_{\text{att}\in\mathcal{S}})$, where $k' = \alpha - r'b$,

$$k'' = r'b'' + rb^{(3)}, \quad \bar{k}'' = r'b'' + \bar{r}b^{(3)}, \quad k_{1,\text{att}} = r_{\iota(\text{att})} f_n(x_{\text{att}}) + r f'_{n_c-1}(x_{\text{att}}),$$
$$\bar{k}_{1,\text{att}} = \bar{r}'_{\iota(\text{att})} \bar{f}'_{n_c-1}(x_{\text{att}}) + \bar{r}_{\iota(\text{att})} \bar{b}_0, \quad \bar{k}_{2,\text{att}} = \bar{r}_{\iota(\text{att})} \bar{f}_n(x_{\text{att}}),$$

such that $\sum_{l\in[m]} \bar{r}'_l = \bar{r}$. Note that we require that each partition is full, i.e., $|\iota^{-1}(l)| = n_k$ for all $l \in [m]$. If needed, this can be done by using dummy attributes [OSW07].

- EncCt($(\mathbf{A}, \rho, \rho_2)$): Let $\Phi = \{j \in [n_1] \mid \rho_2(j) = 1\}$ and $\overline{\Phi} = [n_1] \setminus \Phi$. Let $\mathbf{s} = (s, \{s_{l'}\}_{l'\in[m']}, \{s'_j\}_{j\in[n_1]})$ and $\hat{\mathbf{s}} = (\hat{v}_2, ..., \hat{v}_{n_2})$. We set $\mathbf{c} = \{c_{1,j}, c_{2,j}, c_{3,j}, \bar{c}_{2,j'}, \bar{c}_{3,j'}\}_{j\in\Phi, j'\in\overline{\Phi}}$, where $c_{1,j} = \mathbf{A}_j(sb, \hat{\mathbf{s}})^\intercal + s'_j b''$, and

  – For $j \in \Phi$: $c_{2,j} = s'_j b^{(3)} + s_{\tau(j)} f'_{n_c-1}(x_{\rho(j)}), c_{3,j} = s_{\tau(j)} f_n(x_{\rho(j)})$.

  – For $j \in \overline{\Phi}$: $\bar{c}_{2,j} = s'_j b^{(3)} + s_{\tau(j)} \bar{f}'_{n_c-1}(x_{\rho(j)}), \bar{c}_{3,j} = s_{\tau(j)} \bar{f}_n(x_{\rho(j)})$. We require that each partition that has at least one negated attribute in it is full and only contains negated attributes that occur in a conjunction, i.e., for all $j \in [n_1]$ with $\rho_2(j) = 2$, we have $|\chi_j| = n_c$, where $\chi_j = \{j' \in [n_1] \mid \tau(j') = \tau(j) \wedge \rho_2(j') = 2\}$. If needed, this can be done by using dummy attributes (not issued for keys) [OSW07].

- Pair($\mathcal{S}, (\mathbf{A}, \rho, \rho_2)$): For $\mathcal{S}$ that satisfies $\mathbb{A}$, we have some $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ such that $\{\varepsilon_j \in \mathbb{Z}_p\}_{j\in\Upsilon}$ exists with $\sum_{j\in\Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$ (Definition 2.5). We also split Υ into two subsets $\Upsilon' = \Upsilon \cap \Phi$ and $\overline{\Upsilon}' = \Upsilon \setminus \Upsilon'$. We retrieve αs by first computing for each ciphertext partition

$l' \in [m']$ with some row $j \in \overline{\Upsilon}'$ with $\tau(j) = l'$ and $l \in [m]$:

$$\sum_{\mathrm{att} \in \Psi_l} \Lambda_{\Omega_{j,l}, x_{\mathrm{att}}, 0} s_{\tau(j)} \bar{k}_{2,\mathrm{att}} + \sum_{j' \in \chi_j} \Lambda_{\Omega_{j,l}, \rho(j'), 0} \bar{r}_l \bar{c}_{3,j'} = \bar{r}_l s_{\tau(j)} \bar{f}_n(0),$$

where $\Psi_l = \{\mathrm{att} \in \mathcal{S} \mid \iota(\mathrm{att}) = l\}$, and $\Omega_{j,l} = \{x_{\mathrm{att}} \mid \mathrm{att} \in \Psi_l\} \cup \{\rho(j') \mid j' \in [n_1], \tau(j) = \tau(j')\}$. Then, we use it to retrieve for all att $\in \Psi_l$:

$$s_{\tau(j)} \bar{k}_{1,\mathrm{att}} - \bar{r}_l s_{\tau(j)} \bar{f}_n(0) = s_{\tau(j)} \bar{r}'_{\iota(\mathrm{att})} \bar{f}'_{n_c - 1}(x_{\mathrm{att}}),$$

which we use to recover for each $j \in \overline{\Upsilon}'$ and $l \in [m]$:

$$\sum_{\mathrm{att} \in \Psi_l} \Lambda_{\Psi'_l, x_{\mathrm{att}}, \rho(j)} s_{\tau(j)} \bar{r}'_l \bar{f}'_{n_c - 1}(x_{\mathrm{att}}) = s_{\tau(j)} \bar{r}'_l \bar{f}'_{n_c - 1}(x_{\rho(j)}),$$

where $\Psi'_l = \{x_{\mathrm{att}} \mid \mathrm{att} \in \Psi_l\}$. Then, we retrieve

$$\sum_{l \in [m]} s_{\tau(j)} \bar{r}'_l \bar{f}'_{n_c - 1}(x_{\rho(j)}) = s_{\tau(j)} \bar{r} \bar{f}'_{n_c - 1}(x_{\rho(j)})$$

for each $j \in \overline{\Upsilon}'$, so in turn we can retrieve

$$r' c_{1,j} - s'_j \bar{k}'' + \bar{r} c_{2,j} - s_{\tau(j)} \bar{r} \bar{f}'_{n_c - 1}(x_{\rho(j)}) = r' \mathbf{A}_j (sb, \hat{\mathbf{s}})^\intercal.$$

Then, for all $j \in \Upsilon'$, we compute

$$r' c_{1,j} - s'_j k'' + r c_{2,j} - s_{\tau(j)} k_{1,\rho(j)} + r_{\iota(\rho(j))} c_{3,j} = r' \mathbf{A}_j (sb, \hat{\mathbf{s}})^\intercal.$$

Finally, we retrieve $sk' - \sum_{j \in \Upsilon} \varepsilon_j r' \mathbf{A}_j (sb, \hat{\mathbf{s}})^\intercal = \alpha s$.

Performance analysis of the selectively secure instantiation. If we assume that $|\mathcal{S}|$ can take on any positive value, then it is best to put all non-lone variables in $\mathbb{H}$ and the polynomials in $\mathbb{G}$. If $|\mathcal{S}|$ is always going to be small compared to $|\Upsilon|$, then it is better to put all ciphertext components in $\mathbb{G}$ and the key components in $\mathbb{H}$. The costs are (in the best case, assuming that the negations can be distributed optimally over the ciphertext partitions, see the full version [VA22b]):

- KeyGen: $3 + 2m + |\mathcal{S}|$ exponentiations in $\mathbb{H}$ and $3 + 3|\mathcal{S}|$ exponentiations in $\mathbb{G}$;
- Encrypt: $|\Upsilon| + m' + 1$ exponentiations in $\mathbb{H}$, $n_1(n_c + n + 4)$ exponentiations in $\mathbb{G}$, and 1 exponentiation in $\mathbb{G}_T$;

- Decrypt: $\min\left(am + \left\lceil \frac{|\Upsilon|}{n_c} \right\rceil, a(m + 2|\mathcal{S}|) + \left\lceil \frac{|\Upsilon'|}{n_c} \right\rceil\right) + \left\lceil \frac{|\Upsilon'|}{n_k} \right\rceil + 4$ pairing operations, and $am \left\lceil \frac{|\overline{\Upsilon}'|}{n_c} \right\rceil (2n_k + n_c)$ exponentiations in $\mathbb{G}$, where $a = 1$ if $|\overline{\Upsilon}'| > 0$, and $a = 0$ otherwise.

For encrypt and decrypt, the costs are higher when the attributes associated with the negations cannot be distributed optimally over the ciphertext partitions. In the worst case, each negated attribute incurs 2 exponentiations in $\mathbb{H}$ and $n_c + n + 4$ in $\mathbb{G}$ in Encrypt. For decryption, suppose that the policy consists of negated attributes only, and none of them can be placed in the same partition. Then, the decryption costs are $4 + |\Upsilon| + \left\lceil \frac{|\Upsilon'|}{n_k} \right\rceil$ pairing operations and $2|\Upsilon| \cdot |\mathcal{S}| + \left\lceil \frac{|\Upsilon'|}{n_k} \right\rceil \cdot |\Upsilon|$ exponentiations.

7.6 Performance analysis

We analyze the efficiency of GLUE (Construction 7.1). An important aspect in this analysis is the pair of parameters n_k and n_c, which is chosen during the setup (e.g., by a practitioner). On a high level, the key generation, encryption and decryption of the selectively secure version of GLUE incur the following costs:

- KeyGen: $2 + |\mathcal{S}| + \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$ exponentiations in $\mathbb{H}$;
- Encrypt: 1 exponentiation in $\mathbb{G}_T$, $1 + \left\lceil \frac{n_1}{n_c} \right\rceil$ exponentiations, n_1 MBEs with $n_c + 1$ bases and n_1 MBEs with $n_k + n_c$ bases in $\mathbb{G}$;
- Decrypt: roughly $2 + \left\lceil \frac{|\Upsilon|}{n_k} \right\rceil + \left\lceil \frac{|\Upsilon|}{n_c} \right\rceil$ pairing operations.

The efficiency of these algorithms depends on the one hand on the efficiency of these operations, and on the other hand on the choices of n_k and n_c. By analyzing these rough costs from a mathematical point of view, the trade-off between the encryption and decryption efficiency is optimal when $n_k = n_c$ (which follows from the arithmetic mean-harmonic mean inequality). However, when the set of attributes $\mathcal{S}$ is large, and n_k is small, it may occur that all matching attributes are in different partitions. As such, choosing n_k to be larger, e.g., $n_k = 10$, ensures that the matching attributes are in the same key partitions with a large probability, and therefore the actual number of pairing operations is higher. In general, it holds that, the larger the partition sizes, the fewer pairing operations are needed during decryption. Unfortunately, the drawback is that encryption becomes more expensive, meaning that we may want to use the online/offline version of the scheme in practice. In the full version [VA22b], we give more details on how a suitable partition size may be chosen. For our analysis, we consider three parameter settings: $(n_k, n_c) \in \{(3, 3), (5, 5), (10, 5)\}$. Furthermore, for the variant that supports OSW-type negations, we consider $|\mathcal{S}| \in \{1, 5\}$.

On the comparability of the schemes. For a fair comparision, we optimize all the schemes in the same way when instantiating the schemes in the asymmetric setting (Chapter 6). Specifically, we optimize the decryption and encryption efficiency. For the analysis of [RW13], [HW14], Att19-I-CP and Att19-I-CP-OO [Att19, §A-I] schemes, we have used the performance analysis of our associated schemes for $n_k = n_c = 1$ (which have the same encodings). We also compare our monotone schemes with [AHM$^+$16] and [ABGW17] (see the full version [VA22b] for our instantiations of the schemes). In particular, we have first instantiated the schemes with our generic compiler (Definition 4.10), so they yield the most comparable results. To place the costs based on our theoretical analyses in the fully-secure setting, we multiply the costs for each element and operation in $\mathbb{G}$ and $\mathbb{H}$ by a factor 2. This overhead corresponds to the most efficient instantiation of the schemes in the AC17 framework (Section 4.4). For all schemes, we also assume that the input access policies are Boolean formulas, so that for decryption, it is ensured that $\varepsilon_j \in \{0, 1\}$ [LW11a].

Estimates based on benchmarks in RELIC. We estimate the computational costs of the schemes by obtaining benchmarks of various algorithms and extrapolating the results by analyzing the descriptions of the schemes. We analyze the efficiency in this way for two reasons. First, it allows us to analyze the efficiency of many scheme configurations without having to implement each one, which is a cumbersome and error-prone effort. Second, it allows us to compare the schemes more accurately and more fairly, because we can make estimates[1] based on the ABE Squared approach (Chapter 6) instead of using the Charm framework [AGM$^+$13]. For the performance analysis in this chapter, we have run benchmarks in RELIC [AGM$^+$] on a 1.6 GHz Intel i5-8250U processor for the BLS12-446 curve [BLS02], which can be found in the full version [VA22b].

Comparison. Tables 7.3a and 7.3b show the performances of all unbounded schemes using a BB hash that support MSPs and NMSPs[2]. The tables illustrate that the decryption algorithms of our regular schemes are significantly faster than the established schemes. While the encryption costs increase compared to the other schemes, our online/offline versions also provide a solution in this regard, incurring minimal online costs. This comes with a slight trade-off in the ciphertext size and the decryption efficiency compared to the regular version, but overall, our online/offline schemes outperform the established schemes in all algorithms. Importantly, the decryption of our schemes supporting negations with parameters $n_k = n_c = 5$ outperforms the only other unbounded OSW-type non-monotone scheme. Importantly, decryption is faster by a factor 4 for non-negated attributes, and faster by a factor 4-5 for negated

[1] Although approximated theoretically, we expect our estimates to be close to the costs of actual implementations, see the full version of GLUE [VA22b].

[2] The code used to generate these benchmarks is available as a Jupyter notebook at `https://github.com/mtcvenema/glue`.

Table 7.3. Rough estimates of the storage costs of the secret keys and the ciphertexts in kilobytes (KB), where 1 KB = 1024 bytes, and the (online) computational costs incurred by the key generation, encryption and decryption algorithms of GLUE$_{(n_k,n_c)}$ (and its online/offline (suffixed with "OO") and OSW-non-monotone (suffixed with "N") variants) and the other unbounded CP-ABE schemes, expressed in milliseconds (ms), for 10 and 100 attributes. Note that the offline key generation and encryption costs of each online/offline scheme are equal to the key generation and encryption costs of its regular version.

		Storage costs					**Computational costs**					
			SK		CT		KeyGen		Encrypt		Decrypt	
	Scheme	|MPK|	10	100	10	100	10	100	10	100	10	100
Regular	[RW13]	1.42	4.86	44.58	4.05	33.58	26.0	238.7	32.9	305.9	46.2	375.2
	[AHM+16] ($n_k = 2$)	1.75	5.3	45.02	6.45	55.67	16.5	122.9	40.8	368.3	43.7	317.4
	[ABGW17]	1.42	2.65	22.51	3.94	33.47	14.2	120.5	32.3	305.2	27.9	192.4
	GLUE$_{(3,3)}$	2.08	3.53	30.02	3.39	26.36	18.9	160.7	59.8	571.4	24.3	133.9
	GLUE$_{(5,5)}$	2.74	3.09	26.93	3.17	24.83	16.5	144.2	82.3	800.4	17.0	82.8
	GLUE$_{(10,5)}$	3.28	2.87	24.72	3.17	24.83	15.4	132.3	102.1	998.4	15.1	64.5
O/O	[HW14]	1.42	5.23	48.29	4.79	41.0	0	0	0	0	51.5	416.2
	GLUE$_{(3,3)}$	2.08	3.9	33.73	5.62	48.62	0	0	0	0	33.6	202.6
	GLUE$_{(5,5)}$	2.74	3.46	30.64	6.88	61.94	0	0	0	0	27.6	157.5
	GLUE$_{(10,5)}$	3.28	3.24	28.43	8.74	80.49	0	0	0	0	24.2	123.4

(a) Schemes supporting MSPs only.

		Storage costs					**Computational costs**									
			SK		CT		KeyGen		Encrypt		Decrypt					
	Scheme	|MPK|	10	100	10	100	10	100	10	100	10	100	10	100	10	100
Regular	Att19-I-CP	1.4	11	100	6	56	59	542	67	638	52	380	55	368	216	1779
	GLUE-N$_{(3,3)}$	2.1	7.6	64	5	41	45	386	90	865	30	139	62	375	110	746
	GLUE-N$_{(5,5)}$	2.7	6.5	56	4.6	38	40	353	111	1086	22	88	55	368	55	383
	GLUE-N$_{(10,5)}$	3.3	5.9	50	4.6	38	38	329	131	1284	21	70	79	599	78	614
O/O	Att19-I-CP-OO	1.4	12	111	7.1	63	0	0	0	0	61	461	65	449	226	1860
	GLUE-N$_{(3,3)}$	2.1	8.7	75	7.3	63	0	0	0	0	47	275	79	511	126	881
	GLUE-N$_{(5,5)}$	2.7	7.6	67	8.3	75	0	0	0	0	42	236	75	516	75	530
	GLUE-N$_{(10,5)}$	3.3	7.1	62	10	94	0	0	0	0	37	186	95	715	95	730

(b) Schemes supporting OSW-type negations. The decryption costs are for non-negated, and negated policies with $|\mathcal{S}| \in \{1, 5\}$, respectively.

attributes and $|\mathcal{S}| = 5$, bringing down the costs from almost two seconds to 382 ms. As a result, our schemes could provide a more attractive building block for OSWOT-type non-monotone schemes, as they support more efficient decryption algorithms for negated and non-negated attributes, and for small and large sets of attributes for each label. Furthermore, owing to the online/offline extensions, the key generation and encryption algorithms do not need to suffer from heavy online computations. Instead, encrypting users need to store only 3.17-10.17 kilobytes per one intermediate ciphertext of the first type and sufficient of the second type for ten attributes (depending on the instantiation). This means that, with just a megabyte of space, a user can store at least 100 intermediate ciphertexts for a total of 1000 attributes. For computing devices such as computers and smartphones, which have an abundance of storage space nowadays, this is a more than acceptable trade-off. Similarly, key generation authorities can store intermediate keys for at least 286 users and 2860 attributes with just a megabyte of space. Thus, with gigabytes, an authority can precompute keys for hundreds of thousands of users and millions of attributes.

7.7 Applying multiple instantiations of GLUE

The flexible efficiency trade-offs that GLUE provides can be exploited in practice. In particular, practitioners can choose one suitable instantiation of GLUE, or choose multiple instantiations of GLUE to support different computational devices. Interestingly, by using the direct sum with parameter reuse transformation of Attrapadung [Att19], GLUE would be able to support multiple instances of itself simultaneously, such that the size of the master public key is upper-bounded in the maximum size of the public keys of all instances. This may be useful in settings in which the devices have varying computational resources. For instance, in the WLAN use case considered by ETSI [ETS18a], the decryption devices may be any mobile device in a network, including more constrained devices such as smartwatches. For those devices, it is more beneficial to use a scheme with fast decryption, e.g., $\text{GLUE}_{(5,5)}$, while for faster devices, it is sufficient to employ a scheme with slower decryption, e.g., RW13. In WLAN systems, the access point sends an encrypted e.g., WPA2-PSK key to the connecting device, which can decrypt it if is satisfies the policy. Because this exchange is interactive, the connecting device and access point could first negotiate on the particular instance of GLUE for which the connecting device has a secret key before encrypting the WPA2-PSK key. In non-interactive systems, e.g., cloud settings [ETS18a], it may be more desirable to use multiple instances in parallel. Powerful devices could, for instance, use multiple instances to support less powerful devices that only use the more efficient instances. For example, powerful decryption devices could have keys for both $\text{GLUE}_{(5,5)}$ and RW13, while less powerful encryption devices use RW13 or an online/offline variant of GLUE to encrypt.

7.8 Future work

For future work, it would be valuable to improve the security, efficiency and expressivity of GLUE. First, we have proven security in the AC17 framework, which requires a q-type assumption. Although frameworks exist that prove security generically under static assumptions [Att14a, CGW15, Att16], these use the stronger master-key hiding property (Definition 4.3). Like other ABE using a BB hash, ours does not satisfy this property (Lemma 4.2). To achieve full security, more intricate proof techniques need to be devised, such as [CGKW18]. Second, we have analyzed the efficiency of the schemes on the BLS12-446 curve. Presumably, the encryption and decryption costs can improve if curves such as KSS16-339 [KSS08] are used, which provide faster arithmetic in $\mathbb{G}$ and provide more efficient products of pairing operations [CDS20]. GLUE (and RW13) may also benefit from fixed-base multi-base exponentiations [Möl01], which RELIC does not support. Finally, while we have given the first steps towards realizing more efficient schemes supporting OSWOT-type negations, we have not explicitly specified these schemes, which can be built using GLUE-N.

7.9 Conclusion

We have proposed GLUE, a new unbounded large-universe scheme with flexible efficiency trade-off. This scheme is a generalization of RW13 and W11b, in that it supports polynomials of any degree for the Boneh-Boyen hash. To optimally use the randomness provided by the hash, we use the partitioning approach (previously also used by AHM+16 [AHM$^+$16]), splitting the sets of attributes and the policies into partitions of maximum sizes n_k and n_c, respectively. This allows for a decreased number of pairing operations required during decryption compared to RW13 (and related variants). Roughly, the pairing costs decrease by a factor $n_k = n_c$ (if chosen to be equal). Along the way, we have also introduced new proof techniques. These ensure that the randomness provided by the BB hash can be used for both the keys and ciphertexts in the unbounded setting. Finally, we have shown that our schemes indeed outperform existing schemes using a BB hash in the decryption, and notably, all schemes supporting OSW-type negations. Because our non-monotone schemes are unbounded and faster than 1.2 seconds in all algorithms on a laptop, even for large policies and sets, they are more suitable for practice than existing non-monotone schemes.

Chapter 8

TinyABE: unrestricted CP-ABE for embedded devices and low-quality networks

CP-ABE has attracted much interest from the practical community to enforce access control in distributed settings such as the Internet of Things (IoT). In such settings, encryption devices are often constrained, having small memories and little computational power, and the associated networks are lossy. To optimize both the ciphertext sizes and the encryption speed is therefore paramount. In addition, the master public key needs to be small enough to fit in the encryption device's memory. At the same time, the scheme needs to be expressive enough to support common access control models. Currently, however, the state of the art incurs undesirable efficiency trade-offs. Existing schemes often have ciphertexts whose sizes are linear in the policy, and consequently, the ciphertexts may be too large and encryption may be too slow. In contrast, schemes with small ciphertexts have extremely large master public keys, and are generally computationally inefficient.

In this chapter, we propose TinyABE: a novel CP-ABE scheme that is expressive and can be configured to be efficient enough for settings with embedded devices and low-quality networks. In particular, we demonstrate that our scheme can be configured such that the ciphertexts are small, encryption is fast and the master public key is small enough to fit in memory. From a theoretical standpoint, the new scheme and its security proof are non-trivial generalizations of the expressive scheme with constant-size ciphertexts by Agrawal and Chase (TCC'16, Eurocrypt'17) and its proof to the unbounded setting. By using techniques of Rouselakis and Waters (CCS'13), we remove the restrictions that the Agrawal-Chase scheme imposes on the keys and ciphertexts, making it thus more flexible. In this way, TinyABE is especially suitable for IoT.

8.1 Introduction

CP-ABE has proven to be a valuable primitive in enforcing access control on a cryptographic level [BSW07, KL10, SRGS12]. Recently, ETSI has published two specifications regarding the high-level requirements for ABE [ETS18a], and how ABE can increase data security and privacy [ETS18b]. In these specifications, ETSI focuses on several use cases, one of which considers data access control in the Internet of Things (IoT), in particular. An important requirement that ETSI imposes on ABE is that an IoT device should be able to encrypt, but not necessarily decrypt. To this end, the public keys and ciphertexts should be small, and encryption should be efficient.

According to RFC8576[1], IoT devices and networks are characterized by small memory, low computational power, and high packet loss rates. Unfortunately, many ABE schemes—including those considered by ETSI [ETS18a]—have ciphertexts sizes and encryption costs that grow linearly in the number of attributes [AC17a, ABGW17]. As a result, these schemes are not suitable for IoT applications [ETS20]. First, encryption may simply consume too much time, requiring almost one second per attribute [Sco20]. Second, even for small policies, the ciphertexts may be so large that they have to be fragmented across more than one data packet during transmission. This results in an increased probability that at least one of the packets is dropped, and subsequently increases the expected time that it takes for the message to successfully arrive at the receiver [PST20]. Third, the ciphertext may not fit in memory. The computation of one ciphertext would therefore need to be split into parts, and the partial ciphertexts need to be streamed out of the device, like in [HRS16]. This may further complicate issues with packet loss.

To mitigate difficulties with the size, ABE schemes with sufficiently short ciphertexts can be deployed. Several schemes with constant-size ciphertexts have been proposed [EMN$^+$09, HLR10, CZF11, ALdP11, AHY15, AC16]. However, many of these schemes have restricted policies [EMN$^+$09, HLR10, CZF11], supporting only AND-gates or threshold functions, and therefore have a limited expressivity. Others are bounded [ALdP11, AHY15, AC16], supporting only limited sizes for the sets or policies associated with the ciphertexts. More importantly, the efficiency of these bounded schemes depends heavily on the bounds. Hence, choosing these bounds to be sufficiently high for some given practical setting is not a suitable option either.

In this chapter, we mitigate these limitations by proposing a scheme with a trade-off feature. Upon setup, the system parameters can be chosen such that the desired efficiency trade-off between the sizes of the keys and the ciphertexts, as well as the computational costs of the algorithms can be attained. In particular, one can optimize encryption so that it can be performed on IoT devices. Furthermore, one can configure the ciphertexts to be small enough for a specific setting, i.e., to fit in memory of IoT devices or in one Ethernet packet for a priorly specified maximum policy length, e.g.,

[1] https://tools.ietf.org/html/rfc8576

100. One can also configure the master public key to be small enough to fit in memory. This makes TinyABE especially suitable for IoT.

8.1.1 Our contributions

Our main contribution is TinyABE, a new CP-ABE scheme that simultaneously can satisfy several desirable properties:

(1) **Expressivity:** The scheme supports monotone span programs (MSPs), which includes Boolean formulas consisting of both AND and OR-gates;

(2) **Large-universeness:** Any string can be used as attribute;

(3) **Unboundedness:** No bounds are posed on the parameters, including the attribute sets associated with the keys and the policy lengths;

(4) **Configurable:** The system parameters can be chosen such that the scheme attains the required efficiency, for example

   - **Short ciphertexts:** The scheme can be configured such that the ciphertexts are sufficiently small for scenarios involving low-quality networks;
   - **Efficient encryption:** The scheme can be configured such that encryption is fast, even on resource-constrained devices.

We achieve this by making the expressive CP-ABE scheme with constant-size ciphertexts by Agrawal and Chase (AC16) [AC16] unbounded. As a result, our scheme is parametrized, and can be configured to provide the desired efficiency trade-off. Special cases of our scheme include AC16, and the CP-ABE scheme with constant-size ciphertexts by Attrapadung (Att19) [Att19]. TinyABE can thus be viewed as a generalization of AC16 to the unbounded setting. We also provide two secondary contributions:

- *Security proof:* We generalize Agrawal and Chase's [AC17b] proof for AC16 to the unbounded setting using Rouselakis and Waters' [RW13] techniques;
- *Performance analysis:* We analyze the efficiency of our scheme with a focus on practice. In particular, we obtain the most efficient encryption algorithm compared to other expressive and unbounded schemes;

8.2 High-level overview and details about TinyABE

Our construction. TinyABE is a generalization of the Agrawal-Chase scheme (AC16) [AC16, AC17b] to the unbounded setting, using the partitioning techniques by Attrapadung et al. (AHM+16) [AHM$^+$16], and by using the proof techniques by Rouselakis and Waters (RW13) [RW13]. By generalizing AC16, we can make it more efficient.

Although AC16 supports expressive policies and attains constant-size ciphertexts, it is bounded in parameters N_1 and N_2, where N_1 and N_2 denote the upper bounds on the number of rows and columns of the access structure, respectively. Importantly, the scheme's efficiency depends on these parameters. Whereas the ciphertext sizes are constant, the master public key grows by a factor N_1N_2, and the secret keys grow by a factor $N_1^2N_2$. We show in our performance analysis that, as a result, the master public key is already so large for $N_1 = N_2 = 32$, i.e., 103 kilobytes (KB), that it does not fit in memory of many embedded devices. By making AC16 unbounded, the efficiency depends differently on these factors. We make the AC16 scheme unbounded by using a similar approach as AHM+16. Roughly, we partition the sets of rows and columns in smaller subsets of maximum sizes $\hat{n}_1$ and $\hat{n}_2$, respectively, and apply the AC16 scheme to the partitions. The master public key and secret keys then also grow in factors $\hat{n}_1\hat{n}_2$ and $\hat{n}_1^2\hat{n}_2$, respectively, but $\hat{n}_1$ and $\hat{n}_2$ can be much smaller to attain small ciphertexts. Although our ciphertexts are not constant-size, they shrink by a factor $\mathcal{O}(\min(\hat{n}_1, \hat{n}_2))$ compared to schemes with linear-size ciphertexts, such as RW13. Thus, even for small choices of $\hat{n}_1$ and $\hat{n}_2$, our ciphertexts are much smaller than RW13 ciphertexts. Whereas RW13 ciphertexts might only fit in memory or in one Ethernet packet for a maximum policy length of 33 or 3, respectively, TinyABE can support larger policy lengths. For example, in the same settings, it supports maximum policy lengths of 298 (for $\hat{n}_1 = \hat{n}_2 = 3$), and 100 (for $\hat{n}_1 = \hat{n}_2 = 13$), respectively, while the associated master public keys are only 2.3, and 19 KB, respectively.

Security proof. We formulate our scheme and proofs in the AC17 [AC17b] framework (Chapter 4). In part, we use this framework, because we generalize AC16, and its only proofs in the full-security setting are given in this framework [AC17b, Att19]. In contrast, other expressive CP-ABE schemes with constant-size ciphertexts [AHY15, AT20] have larger keys than AC16 and are therefore less efficient. Furthermore, recall that, because we prove security in the AC17 framework, our scheme can be transformed into a scheme supporting negations in the policies [Att19, Amb21].

Improving the partitioning approach. We improve on the partitioning approach used for the KP-ABE scheme of Attrapadung et al. (AHM+16) [AHM$^+$16], which is unbounded, supports expressive policies, and can be configured to have small ciphertexts. Specifically, AHM+16 generalizes the first expressive KP-ABE scheme with constant-size ciphertexts of Attrapadung et al. (ALP11) [ALdP11] to the unbounded setting. Concretely, their approach consists of the partitioning of the attribute set (to be used during encryption) into subsets of maximum size n_k, where n_k is the bound on the attribute set inherited from ALP11. Before our work, a CP-ABE scheme attaining similar characteristics remained an open problem. In fact, the first expressive CP-ABE schemes with constant-size ciphertexts [AHY15, AC16] were proposed four years after the introduction of ALP11. Presumably, the reason for this delay is the difficulty in simultaneously achieving these properties in the ciphertext-policy setting.

On the one hand, the entire access policy—which is two-dimensional—needs to be embedded in one ciphertext component. On the other hand, the decrypting user—who has an attribute set satisfying the policy—may not have keys for all attributes used in the access policy. These difficulties also translate to the unbounded setting: to make AC16 unbounded, we need to partition in two dimensions instead of one. In addition, we want to embed the entire policy in one ciphertext component, like AC16. This is unlike AHM+16, which embeds each partitioned subset in a separate ciphertext component, and thus still requires a linear number of operations during encryption. In contrast, the costs of computing our ciphertext component embedding the policy are essentially upper-bounded by a constant.

Performance analysis. We show that TinyABE offers advantages over other schemes by analyzing the storage and computational costs. In this analysis, we take into account the limitations of constrained devices and low-quality networks. To this end, we select two configurations of TinyABE, which we compare with RW13 and AC16. Our first configuration provides sufficiently small public keys and ciphertexts for IoT devices, whilst attaining an efficient encryption. For example, in Section 8.5.3, we estimate the encryption costs on some IoT devices. For policies of length 100, encryption with RW13 takes over a minute, while encryption with our scheme takes only 7.6 seconds. Moreover, while the master public key of AC16 is almost a megabyte in size, our master public key is only 2.25 kilobytes, and thus fits easily in memory of constrained devices. Our second configuration ensures that, for policy lengths of up to 100 attributes, the ciphertexts fit in one Ethernet packet, which has a maximum transmission unit of 1500 bytes. In contrast, RW13 ciphertexts are too large.

Expressive, large-universe, unbounded and efficient. TinyABE is simultaneously expressive and unrestricted while it is configurable. Therefore, it can be configured to be efficient enough for practical applications involving IoT devices and networks. Our scheme supports large universes, so it can efficiently support any strings as attributes, and does not require that, in the setup, public keys are generated for each attribute. The scheme is also unbounded[2], which implicitly ensures that it attains a better efficiency, even for large policies, compared to bounded schemes. In contrast, the efficiency of bounded schemes with constant-size ciphertexts [AC16, Att19] depends heavily on the choice of these bounds. Finally, because our scheme supports monotone span programs, it can enforce any fine-grained policies on encrypted data. Practitioners therefore do not need to restrict themselves to less expressive solutions in IoT settings anymore [KHA+19, ETS20].

Comparison with other schemes. Several schemes have been introduced over the years. Some can attain sufficiently short ciphertexts for some specific practical

[2] Note that our scheme is also unbounded in that it is "multi use" (Section 3.3.2).

Table 8.1. Comparison of ABE schemes with short ciphertexts. For each scheme, we list whether they are CP, the expressivity (Expr.), whether they are large-universe (LU), and whether they support unbounded (Unb.) policies or sets. For the schemes with short ciphertexts, we also give the asymptotic complexity of the storage costs of their master public keys (MPK), secret keys (SK) and ciphertexts (CT). We consider a scheme to have short ciphertexts if their asymptotic sizes are smaller than linear in the number of attributes, i.e., $\mathcal{O}(|\mathcal{S}|)$ or $\mathcal{O}(|\mathbb{A}|)$. Note that we have only listed schemes that are structurally different, i.e., that have a different PES.

Scheme	**CP**	**Expr.**	**LU**	**Unb.**		**Sizes**		
				$\lvert\mathbb{A}\rvert$	$\lvert\mathcal{S}\rvert$	MPK	SK	CT
[EMN+09]	✓	AND	✗	✓	✓	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
[HLR10]	✓	Threshold	✗	✓	✓	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(1)$
[CZF11]	✓	AND	✗	✓	✓	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(1)$
[ALdP11]	✗	(N)MSP	✓	✓	✗	$\mathcal{O}(N_k)$	$\mathcal{O}(N_k\lvert\mathbb{A}\rvert)$	$\mathcal{O}(1)$
[CCL+13]	✗	Threshold	✗	✓	✓	$\mathcal{O}(\lvert\mathcal{U}\rvert)$	$\mathcal{O}(\lvert\mathcal{U}\rvert\lvert\mathbb{A}\rvert)$	$\mathcal{O}(1)$
[Tak14]	✗	NMSP	✓	✗	✗	$\mathcal{O}(N_k)$	$\mathcal{O}(N_k\lvert\mathbb{A}\rvert)$	$\mathcal{O}(1)$
[AHY15]	✓	(N)MSP	✓	✗	✗	$\mathcal{O}((N_kN_1)^2\lambda)$	$\mathcal{O}((N_kN_1)^4\lambda^2)$	$\mathcal{O}(1)$
[AHM+16]	✗	MSP	✓	✓	✓	$\mathcal{O}(n_k)$	$\mathcal{O}(n_k\lvert\mathbb{A}\rvert)$	$\mathcal{O}(\frac{\lvert\mathcal{S}\rvert}{n_k})$
[AC16, AC17b]	✓	MSP	✓	✗	✗	$\mathcal{O}(N_1(N_2+N_k))$	$\mathcal{O}(\lvert\mathcal{S}\rvert N_1^2(N_2+N_k))$	$\mathcal{O}(1)$
[Att19]	✓	NMSP	✓	✗	✓	$\mathcal{O}(N_1N_2))$	$\mathcal{O}(\lvert\mathcal{S}\rvert N_1^2N_2)$	$\mathcal{O}(1)$
[AT20]	✓	(N)MSP	✓	✗	✓	$\mathcal{O}((N_2+N_k\lambda)^2)$	$\mathcal{O}((N_2+N_k\lambda)^4)$	$\mathcal{O}(1)$
[LL20b]	✗	MSP	✓	✓	✗	$\mathcal{O}(N_k)$	$\mathcal{O}(N_k\lvert\mathbb{A}\rvert)$	$\mathcal{O}(1)$
TinyABE	✓	MSP	✓	✓	✓	$\mathcal{O}(\hat{n}_1(\hat{n}_2+n_k))$	$\mathcal{O}(\hat{n}_1^2(\hat{n}_2+\hat{n}_k)\frac{\lvert\mathcal{S}\rvert}{n_k})$	$\mathcal{O}(\min(\frac{n_1}{\hat{n}_1},\frac{n_2}{\hat{n}_2}))$

Note: $\mathcal{U}$ = universe; $\mathbb{A}$ = access policy; $\mathcal{S}$ = set of attributes;
(N)MSP = (non-)monotone span program, n_1, n_2 = number of rows, columns of $\mathbb{A}$;
N_1, N_2, N_k = maximum bounds on $n_1, n_2, |\mathcal{S}|$;
$\hat{n}_1, \hat{n}_2, n_k$ = maximum partition sizes of $n_1, n_2, |\mathcal{S}|$

context. In particular, we consider schemes that can be configured to be small enough to fit, e.g., in memory of constrained devices or in Ethernet packets, even for large policies. As Table 8.1 shows, all of the CP-ABE schemes of this kind incur a trade-off: either they are not expressive, or they impose bounds (and by extension, they are inefficient). In contrast, TinyABE is the first CP-ABE scheme to overcome these limitations. Furthermore, compared to expressive schemes with ciphertext sizes that grow at least in the size of the policy or set (see e.g., Table 7.1), TinyABE can be configured to have a more efficient encryption. As such, it is feasible to implement ABE on IoT devices (see Section 8.5.3), which are mainly assumed to be required to encrypt and not decrypt.

8.3 Our construction: TinyABE

We present our construction. To this end, in Section 8.3.1, we give a step-by-step description on how these layering techniques can be applied, by first carefully reviewing the scheme. Roughly, we use the techniques of Attrapadung et al. [AHM+16, Att19] to remove the bounds on the attribute sets used in the key generation. Then, we apply the layering techniques to the ciphertext policy, by using the partitioning approach of

Attrapadung et al. (AHM+16) [AHM+16]. However, unlike in AHM+16, we need to partition in two “directions” due to the two-dimensional nature of access policies. In particular, for each policy, we split the set of rows into subsets of maximum size $\hat{n}_1$, and the set of columns into subsets of maximum size $\hat{n}_2$. Then, for each subset, we use a fresh “randomizer”. These randomizers are appropriately applied to the ciphertext component of AC16 that embeds the policy. To this end, we identify which parts of this ciphertext component correspond to the rows and which to the columns:

$$C'' = \left(\underbrace{\prod_{j\in[n_1],k\in[n_2]} g^{sA_{j,k}b_{j,k}}}_{\text{columns}} \right) \left(\underbrace{\prod_{i\in\overline{[n_k]},j\in[n_1]} g^{sx^i_{\rho(j)}b'_{i,j}}}_{\text{rows}} \right),$$

where $g^{b_{j,k}}$ and $g^{b'_{i,j}}$ denote public keys, and s is a random integer during encryption under access structure $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1\times n_2}$. For example, for each partitioned subset $\mathcal{S}'_l$ of $[n_1]$, we use a fresh randomizer s_l to compute the partial ciphertext $\prod_{i\in\overline{[n_k]},j\in\mathcal{S}'_l} g^{s_l x^i_{\rho(j)} b'_{i,j}}$. In the scheme, we use maps τ_1 and τ_2 to assign each row and column, respectively, to a partition. Furthermore, we define the maps $\hat{\tau}_1$ and $\hat{\tau}_2$ to map each row and column, respectively, that are in the same partition to a unique set of public keys.

8.3.1 Removing the bounds from AC16

We show how to make AC16 unbounded, by analyzing the scheme and showing, in steps, how the bounds can be removed by introducing more randomness.

The AC16 scheme. We briefly review the AC16 scheme [AC16]. The secret keys SK and ciphertexts CT are of the form

$$\begin{aligned} \text{SK} = (\{&K_{1,j} = g^{r_j}, K_{2,j,k} = g^{r_j b_{j,k} - v_k}, K_{3,j,j',k} = g^{r_j b_{j',k}}, \\ &K_{4,j,\text{att}} = g^{r_j \sum_{i\in\overline{[n_k]}} x^i_{\text{att}} b'_{i,j}}, K_{5,i,j,j'} = g^{r_j b'_{i,j'}}\}_{\substack{i\in\overline{[n_k]},j,j'\in[\hat{n}_1],j\neq j',\\ k\in[\hat{n}_2],\text{att}\in\mathcal{S}}}), \end{aligned}$$

$$\text{CT} = \left(C = M\cdot e(g,g)^{\alpha s}, C' = g^s, C'' = \prod_{j\in[n_1],k\in[n_2]} g^{sA_{j,k}b_{j,k}} \prod_{i\in\overline{[n_k]},j\in[n_1]} g^{sx^i_{\rho(j)}b'_{i,j}} \right)$$

where $g^{b_{j,k}}$ and $g^{b'_{i,j}}$ denote public keys, $v_1 = \alpha$ is the master-key, $r_j, v_k \in_R \mathbb{Z}_p$ are randomly chosen integers during the key generation for set $\mathcal{S}$ with $|\mathcal{S}| \leq n_k$, and s is a randomly chosen integer during encryption for access structure $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1\times n_2}$ such that $n_1 \leq N_1$ and $n_2 \leq N_2$ and $\rho\colon [n_1] \to \mathbb{Z}_p$, where $N_1, N_2 \in \mathbb{N}$

denote bounds on the policy size. Furthermore, x_{att} denotes the unique representation of an attribute att (represented as a string) in $\mathbb{Z}_p$, which can be generated with a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$.

Intuitively, decryption using a key SK for set $\mathcal{S}$ of a ciphertext CT with access policy $\mathbb{A}$ works by "singling out" each row $j \in \Upsilon = \{j' \in [n_1] \mid \rho(j') \in \mathcal{S}\}$, i.e., $e(g,g)^{r_j s(\sum_{k \in [\hat{n}_2]} A_{j,k} b_{j,k} + \sum_{i \in \overline{[n_k]}} x^i_{\rho(j)} b'_{i,j})}$ from C'' (and $K_{1,j}$). From this, $e(g,g)^{\sum_{k \in [\hat{n}_2]} A_{j,k} v_k}$ can be retrieved by using C', $K_{2,j,k}$ and $K_{4,j,\rho(j)}$. More concretely, this can be done because the secret keys are constructed in a specific way. That is, for all $j, j' \in [\hat{n}_1]$, it embeds the product $r_j b_{j',k}$, but only in the case that $j = j'$, it also embeds the secret v_k (where $v_1 = \alpha$). Similarly, for $j = j'$, only the secrets $r_j \sum_i x^i_{\text{att}} b'_{i,j}$ are given for those attributes att that are in the set $\mathcal{S}$. For $j \neq j'$, we can reconstruct $r_j \sum_i x^i_{\overline{\text{att}}} b'_{i,j'}$ for any attribute $\overline{\text{att}}$. To decrypt, we have to retrieve $e(g,g)^{\alpha s}$, for which we would need to pair $K_{2,j,k}$ with C' to obtain $e(g,g)^{r_j b_{j,k} s - v_k s}$. Then, the question is how we can cancel out $e(g,g)^{r_j b_{j,k} s}$. Roughly, we want to "single out" the j-th row of the access policy in the ciphertext component C''. Then, we pair $K_{1,j} = g^{r_j}$ with C'', and cancel out all resulting components $e(g,g)^{r_j s(A_{j',k} b_{j',k} + x^i_{\rho(j')} b'_{i,j'})}$ for $j \neq j'$ by using $K_{3,j,j',k}$ and $K_{5,i,j,j'}$ and pairing them with C'. Note that we just argued that we can reconstruct these components (regardless of whether $\rho(j') \notin \mathcal{S}$). This leaves us with components $e(g,g)^{r_j s(\sum_{k \in [\hat{n}_2]} A_{j,k} b_{j,k} + \sum_{i \in \overline{[n_k]}} x^i_{\rho(j)} b'_{i,j})}$. We can only cancel $\prod_{i \in [n_k]} e(g,g)^{r_j s x^i_{\rho(j)} b'_{i,j}}$ if $\rho(j) \in \mathcal{S}$ (by pairing $K_{4,j,\text{att}}$ with C'), which then yields $\prod_{k \in [\hat{n}_2]} e(g,g)^{r_j s A_{j,k} b_{j,k}}$. By combining this with $e(g,g)^{r_j b_{j,k} s - v_k s}$, we can obtain $e(g,g)^{\alpha s}$. For these last steps, we use the following LSSS property (Definition 2.5). If $\mathcal{S}$ satisfies $\mathbb{A}$, then there exist $\varepsilon_j \in \mathbb{Z}_p$ for all $j \in \Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ such that $\sum_{j \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1,0,...,0)$ for rows $\mathbf{A}_j$ of matrix $\mathbf{A}$. Thus, computing $\prod_{k \in [\hat{n}_2]} (e(g,g)^{r_j b_{j,k} s - v_k s})^{\varepsilon_j A_{j,k}}$ yields $\prod_{j \in \Upsilon, k \in [\hat{n}_2]} e(g,g)^{r_j \varepsilon_j A_{j,k} b_{j,k} s} e(g,g)^{-\alpha s}$. We finally obtain $e(g,g)^{\alpha s}$ by raising $\prod_{k \in [\hat{n}_2]} e(g,g)^{r_j s A_{j,k} b_{j,k}}$ to the power of ε_j for each $j \in \Upsilon$.

Removing the bound on set $\boldsymbol{\mathcal{S}}$. First, we remove the bound n_k on set $\mathcal{S}$, which is simpler than removing the bounds on the access policy. In fact, this has already been done by Attrapadung [Att19], so we only briefly review his version of the scheme. Note that this method also resembles the method used in [AHM+16]. The general idea is that the set $\mathcal{S}$ is partitioned in arbitrary sets of maximum size n_k. For each partition, we use the randomness provided by r_j and the public keys to embed the partition like in the original scheme. However, because we have $m = \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$ partitions, we need m fresh sets of randomness $\{r_j\}_{j \in [\hat{n}_1]}$ for each partition. Hence, the keys look like this:

$$\text{SK} = \{K_{1,j,l} = g^{r_{j,l}}, K_{2,j,k,l} = g^{r_{j,l} b_{j,k} - v_k}, K_{3,j,j',k,l} = g^{r_{j,l} b_{j',k}},$$

$$K_{4,j,\text{att}} = g^{r_{j,\iota(\text{att})} \sum_{i \in \overline{[n_k]}} x^i_{\text{att}} b'_{i,j}}, K_{5,i,j,j',l} = g^{r_{j,l} b_{i,j'}} \}_{\substack{j,j' \in [\hat{n}_1], j \neq j', k \in [\hat{n}_2], \\ i \in \overline{[n_k]}, l \in [m], \text{att} \in \mathcal{S}}},$$

where $\iota \colon \mathcal{S} \to [m]$ maps the attributes of $\mathcal{S}$ into partitions. To ensure that the partitions are small enough, we place a restriction on ι, i.e., $|\iota^{-1}(l)| \leq n_k$ for all $l \in [m]$.

Removing the bound from policy $\mathbb{A}$. It is considerably more difficult to remove the bounds on the access policy $\mathbb{A}$. Again, we need to introduce fresh randomness for each partition. However, we need to partition in two directions: the rows and the columns. Hence, we partition the access policy $\mathbb{A} = (\mathbf{A}, \rho)$ by splitting the rows into $m'_1 = \left\lceil \frac{n_1}{\hat{n}_1} \right\rceil$ partitions of maximum size $\hat{n}_1$, and the columns into $m'_2 = \left\lceil \frac{n_2}{\hat{n}_2} \right\rceil$ partitions of maximum size $\hat{n}_2$. In addition, we define the associated maps $\tau_\beta \colon [n_\beta] \to [m'_\beta]$ for $\beta \in \{1, 2\}$ that output the indices of the partitions in which the rows (for $\beta = 1$) and columns (for $\beta = 2$) are mapped. For each partition $l_\beta \in [m'_\beta]$, we introduce a randomizer s_{β, l_β}. In addition, we need to ensure that each row $j \in [n_1]$ in one partition uses a unique set of public parameters $\{g^{b_{j',k}}, g^{b'_{i,j'}}\}_{i \in \overline{[n_k]}, k \in [\hat{n}_2]}$ (with $j' \in [\hat{n}_1]$). Similarly, we need to ensure that each column $k \in [n_2]$ in one partition uses a unique set $\{b_{j,k'}\}_{j \in [n_1]}$ (with $k' \in [\hat{n}_2]$). We thus define the corresponding maps $\hat{\tau}_\beta \colon [n_\beta] \to [\hat{n}_\beta]$ such that $\hat{\tau}_\beta$ is injective on the subdomain $\tau_\beta^{-1}(l_\beta)$ for each $l_\beta \in [m'_\beta]$.

Then, we consider how we can apply any randomness in the ciphertext without causing incorrectness or insecurity. To this end, we analyze the AC16 ciphertext component C'', which is

$$\left(\underbrace{\prod_{j \in [n_1], k \in [n_2]} g^{s A_{j,k} b_{j,k}}}_{C''_{\mathbf{A}}} \right) \cdot \left(\underbrace{\prod_{i \in \overline{[n_k]}, j \in [n_1]} g^{s x^i_{\rho(j)} b'_{i,j}}}_{C''_{\rho}} \right).$$

For both parts $C''_{\mathbf{A}}$ and C''_{ρ}, we analyze with which randomness the randomness s needs to be replaced. As shown, the part associated with the access policy, i.e., $C''_{\mathbf{A}}$, is necessary to retrieve the secret $e(g,g)^{s A_{j,k} v_k}$, such that eventually $e(g,g)^{\alpha s}$ can be retrieved by computing $\prod_{j \in \Upsilon, k \in [n_2]} e(g,g)^{\varepsilon_j s A_{j,k} v_k}$. Note that, here, it is important that s is associated with $k = 1$ to ensure correctness of the scheme. However, for $k > 1$, we can use a different randomness. In short, for the $C''_{\mathbf{A}}$ part, we use the randomness associated with the m'_2 column partitions, which yields the transformation:

$$C''_{\mathbf{A}} \mapsto \prod_{j \in [n_1], k \in [n_2]} g^{s_{2, \tau_2(k)} A_{j,k} b_{\hat{\tau}_1(j), \hat{\tau}_2(k)}},$$

where we require that $s_{2,\tau_2(1)} = s$ to ensure correctness.

As shown, the part of C'' associated with the attribute map ρ, i.e., C''_ρ, ensures that the message is, albeit indirectly, sufficiently blinded. That is, in the "singling out" of row j, we could only obtain $\prod_k e(g,g)^{r_j s A_{j,k} b_{j,k}}$ (now: $\prod_k e(g,g)^{r_{j,\iota(\rho(j))} s_{2,\tau_2(k)} A_{j,k} b_{\hat{\tau}_1(j),\hat{\tau}_2(k)}}$) if we could cancel out $e(g,g)^{r_j s x^i_{\rho(j)} b'_{i,j}}$. This only worked if $\rho(j) \in \mathcal{S}$. In this case, using a fresh set $\{b'_{i,j}\}_{i \in \overline{[n_k]}}$ for each row j ensures that there is sufficient randomness for the entire partition. As such, it is straightforward that the C''_ρ part needs to be randomized for each partition of $\hat{n}_1$ rows, like in the removal of the bound on $\mathcal{S}$. For the row partitions, we had introduced the random integers s_{1,l_1} for each partition $l_1 \in [m'_1]$, and substituting s for these yields:

$$C''_\rho \mapsto \prod_{i \in \overline{[n_k]}, j \in [n_1]} g^{s_{1,\tau_1(j)} x^i_{\rho(j)} b'_{i,\hat{\tau}_1(j)}}.$$

Finally, we point out that the randomizers s_{1,l_1} and s_{2,l_2} are only used in combination with the public keys $b'_{i,j}$ and $b_{j,k}$, respectively. In our proofs, it becomes clear that we can therefore set $s_{2,l_2} = s_{1,l_2}$ for all $l_2 \in [m'_2]$.

8.3.2 The scheme

We give our scheme in the selective-security setting. A fully secure variant can be obtained by applying the AC17 [AC17b] transformation to our PES (Section 8.3.3).

Construction 8.1: TinyABE

TinyABE is defined as follows.

- Setup(λ): On input the security parameter λ, the algorithm generates three groups $\mathbb{G}, \mathbb{H}, \mathbb{G}_T$ of prime order p with generators $g \in \mathbb{G}$ and $h \in \mathbb{H}$, and chooses a pairing $e \colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$. It sets the universe of attributes $\mathcal{U} = \mathbb{Z}_p$, and chooses $\hat{n}_1 \in \mathbb{N}$ and $\hat{n}_2 \in \mathbb{N}$ as the maximum number of rows and columns that fit into one partition, respectively. It also chooses $n_k \in \mathbb{N}$, which is the maximum partition size of the keys. It then generates random $\alpha, b_{j,k}, b'_{i,j} \in_R \mathbb{Z}_p$ for all $i \in \overline{[n_k]}, j \in [\hat{n}_1], k \in [\hat{n}_2]$. It outputs MSK $= (\alpha, \{b_{j,k}, b'_{i,j}\}_{i \in \overline{[n_k]}, j \in [\hat{n}_1], k \in [\hat{n}_2]})$ as the master secret key and publishes the domain parameters $(p, \mathbb{G}, \mathbb{H}, \mathbb{G}_T, \hat{n}_1, \hat{n}_2, n_k)$ and the master public key as

$$\text{MPK} = (g, h, A = e(g,h)^\alpha, \{B_{j,k} = g^{b_{j,k}}, B'_{i,j} = g^{b'_{i,j}}\}_{i \in \overline{[n_k]}, j \in [\hat{n}_1], k \in [\hat{n}_2]}).$$

- KeyGen(MSK, $\mathcal{S}$): On input a set of attributes $\mathcal{S}$, the algorithm computes

$m = \left\lceil \frac{|\mathcal{S}|}{n_k} \right\rceil$, defines a partition map $\iota \colon \mathcal{S} \to [m]$ such that $|\iota^{-1}(l)| \leq n_k$ for each $l \in [m]$, and generates random elements $r_{j,l}, v_k \in_R \mathbb{Z}_p$ for each $j \in [\hat{n}_1], k \in [2, \hat{n}_2], l \in [m]$, setting $v_1 = \alpha$ and computes the secret key as

$$\mathrm{SK}_{\mathcal{S}} = (\{K_{1,j,l} = h^{r_{j,l}}, K_{2,j,k,l} = h^{r_{j,l} b_{j,k} - v_k}, K_{3,j,j',k,l} = h^{r_{j,l} b_{j',k}},$$

$$K_{4,j,\mathrm{att}} = h^{r_{j,\iota(\mathrm{att})} \sum_{i \in \overline{[n_k]}} x^i_{\mathrm{att}} b'_{i,j}}, K_{5,i,j,j',l} = h^{r_{j,l} b_{i,j'}}\}_{\substack{j,j' \in [\hat{n}_1], j \neq j', k \in [\hat{n}_2], \\ i \in \overline{[n_k]}, l \in [m], \mathrm{att} \in \mathcal{S}}}).$$

- Encrypt(MPK, $\mathbb{A}, M$): Message $M \in \mathbb{G}_T$ is encrypted under $\mathbb{A} = (\mathbf{A}, \rho)$ with $\mathbf{A} \in \mathbb{Z}_p^{n_1 \times n_2}$ and $\rho \colon [n_1] \to \mathcal{U}$ by computing $m'_1 = \left\lceil \frac{n_1}{\hat{n}_1} \right\rceil$ and $m'_2 = \left\lceil \frac{n_2}{\hat{n}_2} \right\rceil$, and defining partition maps for each $\beta \in [2]$: $\tau_\beta \colon [n_\beta] \to [m'_\beta]$ such that $|\tau_\beta^{-1}(l_\beta)| \leq \hat{n}_\beta$ for each $l_\beta \in [m'_\beta]$. For τ_2, we require that $\tau_2(1) = 1$. Define $\hat{\tau}_\beta \colon [n_\beta] \to [\hat{n}_\beta]$ such that $\hat{\tau}_\beta$ is injective on the subset $\tau_\beta^{-1}(l_\beta)$ for each $l_\beta \in [m'_\beta]$. Then, generate random integers $s, s_{l'} \in_R \mathbb{Z}_p$ for each $l' \in [2, \max(m'_1, m'_2)]$, set $s_1 = s$, and compute the ciphertext as

$$\mathrm{CT}_{\mathbb{A}} = \Bigg(C = M \cdot A^s, \{C_{l_1} = g^{s_{l_1}}\}_{l_1 \in [m'_1]}$$

$$C' = \left(\prod_{j \in [n_1], k \in [n_2]} B_{\hat{\tau}_1(j), \hat{\tau}_2(k)}^{s_{\tau_2(k)} A_{j,k}} \right) \left(\prod_{i \in \overline{[n_k]}, j \in [n_1]} (B'_{i, \hat{\tau}_1(j)})^{s_{\tau_1(j)} x^i_{\rho(j)}} \right) \Bigg).$$

- Decrypt($\mathrm{SK}_{\mathcal{S}}, \mathrm{CT}_{\mathbb{A}}$): Suppose that $\mathcal{S}$ satisfies $\mathbb{A}$, let $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$. Then, $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ exist with $\sum_{i \in \Upsilon} \varepsilon_j \mathbf{A}_j = (1, 0, ..., 0)$ (Definition 2.5). Message M is retrieved by computing $C \cdot C_2 \cdot C_3 \cdot C_4 \cdot C_5 / C_1$, where

$$C_1 = \prod_{j \in \Upsilon} e(C', K^{\varepsilon_j}_{1, \hat{\tau}_1(j), \iota(\rho(j))}), C_2 = \prod_{j \in \Upsilon, k \in [n_2]} e(C_{\tau_2(k)}, K^{\varepsilon_j A_{j,k}}_{2, \hat{\tau}_1(j), \hat{\tau}_2(k), \iota(\rho(j))}),$$

$$C_3 = \prod_{j \in \Upsilon, j' \in [n_1] \setminus \{j\}, k \in [n_2]} e(C_{\tau_2(k)}, K^{\varepsilon_j A_{j',k}}_{3, \hat{\tau}_1(j), \hat{\tau}_1(j'), \hat{\tau}_2(k), \iota(\rho(j))}),$$

$$C_4 = \prod_{j \in \Upsilon} e(C_{\tau_1(j)}, K^{\varepsilon_j}_{4, \hat{\tau}_1(j), \rho(j)}),$$

$$C_5 = \prod_{i \in \overline{[n_k]}, j \in \Upsilon, j' \in [n_1] \setminus \{j\}} e(C_{\tau_1(j')}, K^{\varepsilon_j x^i_{\rho(j')}}_{5, i, \hat{\tau}_1(j), \hat{\tau}_1(j'), \iota(\rho(j))}).$$

Correctness. The scheme is correct, i.e.,

$$
\begin{aligned}
C_2 \cdot C_3 &= e(g,h)^{\sum_{j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j s_{\tau_2(k)}(r_{\hat{\tau}_1(j),\mathtt{att}} A_{j',k} b_{j',k} - A_{j,k} v_{\hat{\tau}_2(k)})} \\
&= e(g,h)^{\sum_{j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s A_{j',k} b_{j',k}} \cdot e(g,h)^{-\alpha s}, \\
C_6 &= C \cdot C_2 \cdot C_3 \\
&= M \cdot e(g,h)^{\sum_{j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s_{\tau_2(k)} A_{j',k} b_{j',k}} \\
C_4 \cdot C_5 &= e(g,h)^{\sum_{i\in\overline{[n_k]},j\in\Upsilon,j'\in[n_1]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s_{\tau_1(j')} x^i_{\rho(j')} b'_{i,j'}} \\
C_1^{-1} &= e(g,h)^{-\sum_{i\in\overline{[n_k]},j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s_{\tau_2(k)} A_{j',k} b_{j',k}} \\
&\cdot e(g,h)^{-\sum_{i\in\overline{[n_k]},j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s_{\tau_1(j')} x^i_{\rho(j')} b'_{i,j'}}, \\
C_7 &= C_4 \cdot C_5 \cdot C_1^{-1} \\
&= e(g,h)^{-\sum_{j\in\Upsilon,j'\in[n_1],k\in[n_2]} \varepsilon_j r_{\hat{\tau}_1(j),\iota(\rho(j))} s_{\tau_2(k)} A_{j',k} b_{j',k}}, \\
C_6 \cdot C_7 &= C \cdot C_2 \cdot C_3 \cdot C_4 \cdot C_5 / C_1 = M.
\end{aligned}
$$

8.3.3 The associated pair encoding scheme

To prove security, we define the pair encoding of TinyABE, for which we use the variables $\hat{n}_1, \hat{n}_2, n_k, \mathcal{S}, \iota, \rho, \tau_1, \tau_2, \hat{\tau}_1, \hat{\tau}_2, n_1, n_2, \lambda_i, m, m'_1, m'_2$ from Construction 8.1.

Construction 8.2: PES for TinyABE

- Param(par): Let $\mathbf{b} = (\{b_{j,k}, b'_{i,j}\}_{i\in\overline{[n_k]},j\in[\hat{n}_1],k\in[\hat{n}_2]})$.

- EncK($\mathcal{S}$): Let

$$
\begin{aligned}
\mathbf{k}(\mathbf{r},\hat{\mathbf{r}},\mathbf{b}) = (\{&k_{2,j,k,l} = r_{j,l} b_{j,k} - v_k, k_{3,j,j',k,l} = r_{j,l} b_{j',k}, \\
&k_{4,j,\mathtt{att}} = r_{j,\iota(\mathtt{att})} \sum_{i\in\overline{[n_k]}} x^i_{\mathtt{att}} b'_{i,j}, k_{5,i,j,j',l} = r_{j,l} b_{i,j'}\}_{\substack{i\in\overline{[n_k]},j,j'\in[\hat{n}_1],j\neq j', \\ k\in[\hat{n}_2],l\in[m],\mathtt{att}\in\mathcal{S}}}),
\end{aligned}
$$

  where $\mathbf{r} = (\{r_{j,l}\}_{j\in[n_1],l\in[m]})$ are non-lone variables and $\hat{\mathbf{r}} = (\{v_k\}_{k\in[2,n_2]})$ are lone variables.

- EncC(($\mathbf{A}, \rho$)): Let $\mathbf{c}(\mathbf{s},\hat{\mathbf{s}},\mathbf{b}) = (c')$, where

$$
c' = \sum_{j\in[n_1],k\in[n_2]} s_{\tau_2(k)} A_{j,k} b_{\hat{\tau}_1(j),\hat{\tau}_2(k)} + \sum_{i\in\overline{[n_k]},j\in[n_1]} s_{\tau_1(j)} x^i_{\rho(j)} b'_{i,\hat{\tau}_1(j)}
$$

  and $\mathbf{s} = (\{s_{l'}\}_{l'\in[\max(m'_1,m'_2)]})$ are non-lone variables.

Theorem 8.1

The PES for TinyABE in Construction 8.2 satisfies Sym-Prop (Definition 4.2).

Corollary 8.1

Because Construction 8.2 satisfies Sym-Prop, Construction 8.1 is selectively secure (Theorem 4.1).

8.4 Security proof

We prove the symbolic property for TinyABE (Construction 8.2) similarly as for GLUE (Construction 7.2). In particular, we use the layering techniques by Rouselakis and Waters [RW13] to make the proof by Agrawal and Chase [AC17b, AC17c] unbounded. Roughly speaking, we use the layering techniques to embed all rows that are mapped to the same public-key components $b'_{i,j}$ and $b_{j,k}$ for $j \in [\hat{n}_1]$ in these public-key components. Similarly, we use the layering techniques to embed all columns that are mapped to the same public-key components $b_{j,k}$ for $k \in [\hat{n}_2]$.

8.4.1 The selective symbolic property

Our PES satisfies the selective security property. Let $\chi_{1,j} = \{j' \in [n_1] \mid \hat{\tau}_1(j') = \hat{\tau}_1(j)\}$ and $\chi_{2,k} = \{k' \in [n_2] \mid \hat{\tau}_2(k') = \hat{\tau}_2(k)\}$ for all $j \in [\hat{n}_1], k \in [\hat{n}_2]$. Let Υ as before and set $\overline{\Upsilon} = [n_1] \setminus \Upsilon$. Because $\mathcal{S}$ does not satisfy $\mathbb{A}$, there exists $\mathbf{w} = (1, w_2, ..., w_{n_2}) \in \mathbb{Z}_p^{n_2}$ such that $\mathbf{A}_j \mathbf{w}^\intercal = 0$ for all $j \in \Upsilon$ (Definition 2.5). Let $\hat{G}_{j,k}(x_{\text{att}}) = \sum_{i \in [n_k]} (x^i_{\text{att}} - x^i_{\rho(j)}) \mathbf{1}^{d_1 \times d_2}_{(i,j,k),\tau_1(j)}$ for all $j \in [n_1], k \in [n_2]$. Let $\Psi_l = \{\text{att} \in \mathcal{S} \mid \iota(\text{att}) = l\}$ be the l-th partition of $\mathcal{S}$ for all $l \in [m]$. Then, for each $l \in [m]$, we define $G_l(x_{\text{att}}) = \prod_{\text{att}' \in \Psi_l} (x_{\text{att}} - x_{\text{att}'}) = \sum_{i=0}^{n_k} u_{i,l} x^i_{\text{att}}$. We make the following substitutions:

$$b'_{0,j} \; : \; \sum_{j' \in \chi_{1,j}, k' \in [n_2]} A_{j',k'} \left(\mathbf{1}^{d_1 \times d_2}_{(1,j',k'),\tau_1(j')} - \sum_{i' \in [n_k]} x^{i'}_{\rho(j')} \mathbf{1}^{d_1 \times d_2}_{\tau_1(j'),(2,i',j',k')} \right)$$

$$b'_{i,j} \; : \; \sum_{j' \in \chi_{1,j}, k' \in [n_2]} A_{j',k'} \mathbf{1}^{d_1 \times d_2}_{\tau_1(j'),(2,i,j',k')}$$

$$b_{j,k} \; : \; - \sum_{k' \in \chi_{2,k}} \mathbf{1}^{d_1 \times d_2}_{\tau_2(k'),(1,j,k)}, \quad s_{l'} \; : \; \mathbf{1}^{d_1}_{l'}, \quad v_k \; : \; -w_k \left(\sum_{k' \in \chi_{2,k}} \mathbf{1}^{d_1}_{\tau_2(k')} \right)$$

$$r_{j,l} \ : \ \sum_{k' \in [n_2]} w_{k'} \left(\overline{\mathbf{1}}^{d_2}_{(1,j,k')} - \sum_{i' \in [n_k], j' \in \chi_{1,j} \cap \overline{\Upsilon}} \frac{u_{i',l}}{G_l(x_{\rho(j')})} \overline{\mathbf{1}}^{d_2}_{(2,i',j',k')} \right)$$

for all $i \in [n_k], j \in [\hat{n}_1], k \in [\hat{n}_2], l \in [m], l' \in [\max(m'_1, m'_2)]$, where the column indices $(1, j, k)$ and $(2, i, j, k)$ are mapped injectively in the interval $[d_2]$, where $d_1 = \max(m'_1, m'_2)$ and $d_2 = (n_k + 2)n_1 n_2$. It follows quickly that the polynomials evaluate to $\mathbf{0}$ (see the full version [VA22c]).

8.4.2 The co-selective symbolic property

For the co-selective property, we generalize the co-selective proof by Agrawal and Chase [AC17c]. In this proof, the coefficients of the polynomials $G_l(x_{\text{att}})$ are embedded in the variables $b'_{i,j}$. We make the following substitutions:

$$b'_{i,j} \ : \ \sum_{l \in [m]} u_{i,l} \mathbf{1}^{d_1 \times d_2}_{(1,j,l),(j,l)}, \quad b_{j,k} \ : \ \sum_{l \in [m]} \mathbf{1}^{d_1 \times d_2}_{(2,k),(j,l)}, \quad v_k \ : \ \mathbf{1}^{d_1}_{(2,k)},$$

$$r_{j,l} \ : \ \overline{\mathbf{1}}^{d_2}_{(j,l)}, \quad s_{l'} \ : \ \sum_{k \in \tau_2^{-1}(l')} w_k \mathbf{1}^{d_1}_{(2,\hat{\tau}_2(k))} - \sum_{j \in \tau_1^{-1}(l') \cap \overline{\Upsilon}, l \in [m]} \frac{\mathbf{A}_j \mathbf{w}^\intercal}{G_l(x_{\rho(j)})} \mathbf{1}^{d_1}_{(1,\hat{\tau}_1(j),l)}$$

for all $i \in [n_k], j \in [\hat{n}_1], k \in [\hat{n}_2], l \in [m], l' \in [\max(m'_1, m'_2)]$, where the row indices $(1, j, l)$ and $(2, k)$ in the interval $[d_1]$, and the column indices (j, l) are mapped injectively in the interval $[d_2]$, where $d_1 = \hat{n}_1 m + \hat{n}_2$ and $d_2 = \hat{n}_1 m$. It follows quickly that the polynomials go to $\mathbf{0}$ (see the full version [VA22c]).

8.5 Performance analysis

We analyze the performance of TinyABE for two configurations relevant to IoT settings. To illustrate the efficiency trade-offs and the advantages of TinyABE more clearly, we compare the efficiency of the two configurations with two large-universe CP-ABE schemes. In particular, we compare RW13 [RW13], which is an unbounded CP-ABE scheme with linear-size ciphertexts, and the version of AC16 [AC16] with unbounded attribute sets and constant-size ciphertexts: Att19 [Att19]. To effectively compare the efficiency of all relevant schemes, we use the same techniques as for GLUE (Section 7.6). In contrast, however, we use the pairing-friendly elliptic-curve group BLS12-381 [BLS02, Bow] for our analysis. For this curve, Scott has recently performed measurements on several IoT devices [Sco20], which we will use in our analysis in Section 8.5.3. We have run these benchmarks on a 1.6 GHz Intel i5-8250U processor[3] (see the full version [VA22c]).

[3] Our code is available as a Jupyter notebook at github.com/mtcvenema/tinyabe.

8.5.1 Computational costs of TinyABE

We list the computational costs of the key generation, encryption and decryption algorithms by listing the number of group operations required by these (see the full version [VA22c] for further details on our analysis):

- **Key generation:** $\hat{n}_1(m + \hat{n}_1\hat{n}_2 m + |\mathcal{S}| + n_k(\hat{n}_1 - 1)m)$ FBEs in $\mathbb{H}$;
- **Encryption:** one exponentiation in $\mathbb{G}_T$, m FBEs in $\mathbb{G}$, one $(n_k \min(\hat{n}_1, |\Upsilon|))$-MBEs in $\mathbb{G}$, the minimum of the following two costs:
  - one $(\hat{n}_1\hat{n}_2)$-MBE in $\mathbb{G}$;
  - one m'_2-MBE in $\mathbb{G}$ and $n_1 n_2$ multiplications in $\mathbb{G}$;

  and the minimum of the following two costs:
  - one $\hat{n}_1$-MBE in $\mathbb{G}$;
  - one m'_1-MBE and n_1 multiplications in $\mathbb{G}$.

  (Note that the ciphertext component C' can be computed in multiple ways, which we show in more detail in the full version [VA22c]. Specifically, the costs are upper bounded by a constant.)
- **Decryption:** one $(m'_1 + 1)$-multi-pairing operation, and $|\Upsilon|\hat{n}_1 n_2 + n_k n_1$ exponentiations and $(|\Upsilon| - 2)n_k n_1 + 2|\Upsilon|$ multiplications in $\mathbb{H}$.

Two configurations of TinyABE. To investigate the feasibility of TinyABE in IoT settings, we analyze the efficiency of TinyABE for two configurations, i.e.,

1. where encryption is optimal: $(n_k, \hat{n}_1, \hat{n}_2) = (1, 3, 3)$;
2. where ciphertexts are small: $(n_k, \hat{n}_1, \hat{n}_2) = (1, 13, 13)$.

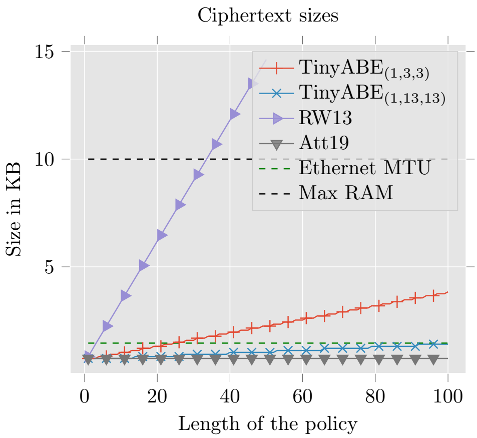
In particular, for the latter configuration, the ciphertexts are small enough such that they fit in one Ethernet packet for policy sizes $|\mathbb{A}| \leq 100$.

8.5.2 Comparison with RW13 and AC16/Att19

We compare the efficiency of TinyABE with RW13, a scheme with linear-size ciphertexts and Att19, the variant of AC16 with constant-size ciphertexts that is unbounded in sets $\mathcal{S}$. In particular, we focus on the ciphertext size and encryption efficiency, as ultimately, we want to optimize the scheme for low-quality networks and resource-constrained encryption devices. For all schemes, we require that they can support any $|\mathbb{A}|, |\mathcal{S}| \leq 100$, thus we set the upper bounds N_1, N_2 of Att19 on the number of rows and columns to $N_1 = N_2 = 100$. Because the key sizes and the key generation costs are linear and the differences between the schemes are large, we put those results in tables, and the rest in graphs.

Scheme	Key sizes (in KB)		
	Size of the set		
	1	10	100
TinyABE$_{(1,3,3)}$	7.39	73.89	738.87
TinyABE$_{(1,13,13)}$	450.71	4507.09	$45.1{\cdot}10^3$
RW13	0.76	4.17	38.27
Att19	$191.3{\cdot}10^3$	$1.9 \cdot 10^6$	$19.1 \cdot 10^6$

(a) Secret keys



(b) Ciphertexts

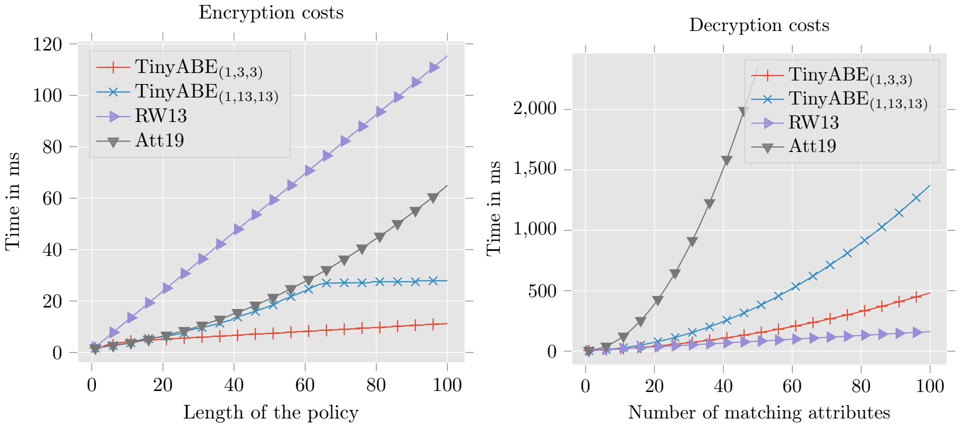
Figure 8.1. The key and ciphertext sizes of TinyABE$_{(n_k,\hat{n}_1,\hat{n}_2)}$, RW13 and Att19.

Storage costs. As Figure 8.1 shows, our secret keys are generally much larger than RW13. Nevertheless, our ciphertexts are much smaller. Our scheme's configurations never exceed the maximum RAM size, and TinyABE$_{(1,13,13)}$ never exceeds the maximum transmission unit (MTU), which we show to be beneficial in Section 8.5.3. Compared to Att19, our keys are much smaller, while our ciphertexts are marginally larger. Importantly, our master public key is, at most, 19.13 KB, in contrast to the 957 KB of Att19. In Section 8.5.3, we show that our scheme can thus be used in resource-constrained devices while Att19 cannot.

Computational costs. Similarly, Figure 8.2 shows that our key generation and decryption costs are higher than RW13, but our encryption is much more efficient. We show in Section 8.5.3 that this gain in encryption efficiency can make a difference between deployment or not, as it reduces the encryption timings on resource-constrained devices from minutes to mere seconds. (Also note that our key generation can be extended to an online/offline variant (see the full version [VA22c]). This allows the authority that generates keys to prepare many keys in advance (e.g., 0.7-44 MB per 100 attributes for $(n_k, \hat{n}_1, \hat{n}_2) \in \{(1,3,3),(1,13,13)\}$), which mitigates any potential issues caused by the large costs.) Moreover, all of our configurations outperform Att19, notably reducing the key generation costs to more feasible timings. While Att19 takes at least eight minutes to compute a key, our scheme never requires more than two minutes, even for large sets.

Scheme	Key generation costs (in ms)		
	Size of the set		
	1	10	100
TinyABE$_{(1,3,3)}$	20.3	202.8	$2 \cdot 10^3$
TinyABE$_{(1,13,13)}$	$1.2 \cdot 10^3$	$12.4 \cdot 10^3$	$123.7 \cdot 10^3$ ($\approx$ 2 minutes)
RW13	2.1	11.4	105.1
Att19	$525.3 \cdot 10^3$ ($>$ 8 minutes)	$5.3 \cdot 10^6$ ($>$ 1 hour)	$52.5 \cdot 10^6$ ($>$ 14 hours)

(a) Key generation



(b) Encryption

(c) Decryption

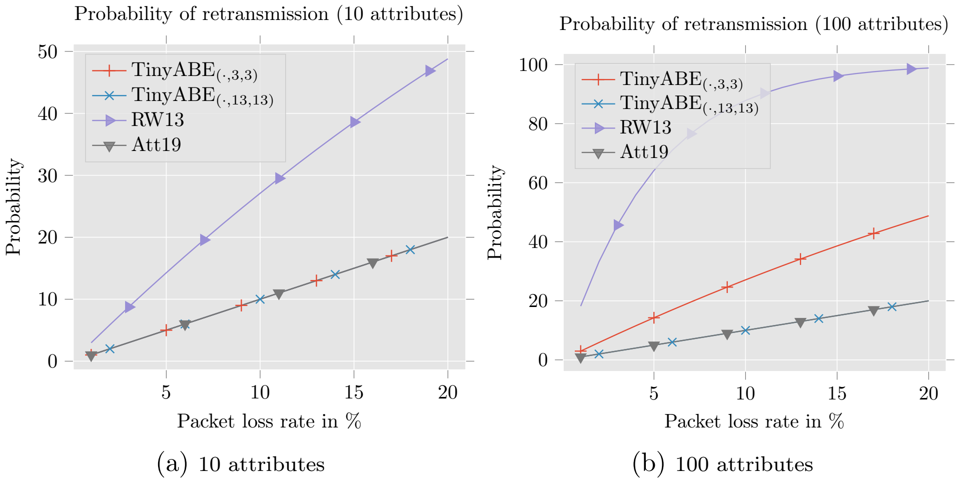
Figure 8.2. The computational costs (in milliseconds (ms)) of key generation, encryption and decryption with TinyABE$_{(n_k,\hat{n}_1,\hat{n}_2)}$, RW13 and Att19.

Comparison with other linear-sized schemes. The main reason why we compare our scheme with RW13 is because it is closely related to our scheme, and because it has linear-size ciphertexts and it is unbounded. In addition, to illustrate the advantages of TinyABE in IoT settings, we want to compare mainly its encryption (and the public key and ciphertext sizes) with a linear-sized scheme such as RW13. Because encryption with other popular large-universe schemes [BSW07, AC17a, ABGW17] is roughly as efficient as RW13 (see the full version [VA22c]), we expect that TinyABE compares similarly as favorably to those schemes as to RW13.

8.5.3 Advantages in IoT settings

We showcase some practical advantages of TinyABE in IoT settings.

Packet loss in low-quality networks. One of the main features of TinyABE is that the ciphertexts can be configured to be small, which is beneficial in low-quality



(a) 10 attributes

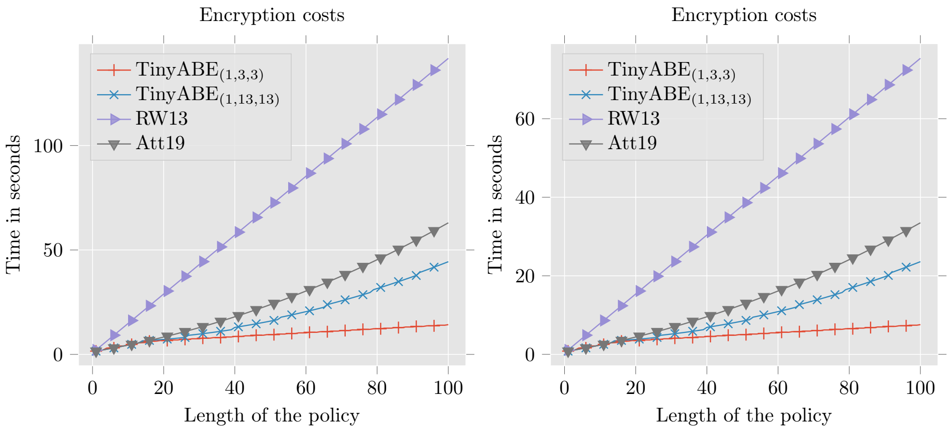
(b) 100 attributes

Figure 8.3. The probability that a partial ciphertext needs to be retransmitted for various packet loss rates.

networks. In general, large ciphertexts may increase the risk that at least one of the packets is dropped during transmission, in which case the dropped packets need to be retransmitted [PST20], delaying the message's time of arrival. This may be problematic for resource-constrained devices, such as IoT devices, because their communication channels are often characterized by high packet loss rates (see RFC8576[4]).

Through an example, we briefly illustrate that the increase in size of the ciphertext may increase the probability that one of the packets drops. For this example, we use a similar approach as in [PST20]. We consider the maximum transmission unit (MTU) of an Ethernet connection, i.e., 1500 bytes, and consider varying packet loss rates between 1% and 20%, with steps of 1%. In general, if we consider the shortest round-trip time, then the expected additional time incurred by re-transmitting a packet is 6.193 ms. In comparison, we can encrypt a message under a policy of length 20 with our least efficient configuration of our scheme in that time (Figure 8.2), so this additional overhead is relevant in practice. To illustrate the effect of packet loss on the efficiency of the scheme, we analyze the probability that at least a part of the ciphertext is dropped in Figure 8.3. As the figures show, TinyABE never exceeds 50%, even for large policies and high packet loss rates. In contrast, for small policies, RW13 exceeds 50% at high rates (i.e., $> 20\%$) and for large policies at small rates (i.e., $> 3.5\%$). For large policies and high rates, it is almost certain that a partial RW13 ciphertext drops. Therefore, our scheme clearly provides an advantage in low-quality networks compared to schemes with linear-size ciphertexts such as RW13.

[4] https://tools.ietf.org/html/rfc8576



(a) Arduino Nano 33 BLE Cortex-M4

(b) Fishino Piranha MIPS32 (@120MHz)

Figure 8.4. Conservative estimates of the encryption costs on two IoT devices.

Resource-constrained devices: memory. In contrast to other expressive schemes such as RW13 and Att19, the ciphertexts and master public key of TinyABE easily fit in memory, even of resource-constrained devices. In RFC7228[5], three classes of constrained devices are listed: with < 10, ≈ 10 and 50 KB of RAM, and with < 100, ≈ 100 and 250 KB of flash memory (ROM), respectively. In practice, the master public key can be stored in flash memory, such that only the components that are needed during encryption are loaded in RAM. While the RW13 public key (of 0.89 KB) fits easily in flash memory, our public keys are fairly large, e.g., 2.25-19.13 KB for $(n_k, \hat{n}_1, \hat{n}_2) \in \{(1,3,3),(1,13,13)\}$. Although it leaves slightly less space for the code and other applications than RW13 would, it easily fits in devices with at least 100 KB of flash memory. This is not the case for Att19, as the master public key of 957 KB is much larger than 100 KB for maximum policy length 30 or higher.

Additionally, while the ciphertexts of our scheme as well as the RW13 scheme would easily fit in 50 KB of RAM (even for large policies), it would be more problematic for devices with only 10 KB of RAM to fit RW13 ciphertexts. In fact, a ciphertext with policy length 33 already pushes the limit of 10 KB, and leaves no space for anything else, such as the payload. In contrast, TinyABE's ciphertexts easily fit in 10 KB of memory, even for large policies. For $\hat{n}_1 = 3$ and policy length $n_1 = 100$, the size of the ciphertext is 3.84 KB. For $\hat{n}_1 = 13$, the size of the ciphertext is 1.41 KB, which leaves an ample 8.59 KB of memory for, e.g., the payload and optimization through precomputation. In this way, we may be able to gain some significant speedup [FA17].

[5] https://tools.ietf.org/html/rfc7228

Resource-constrained devices: speed. As shown, for some parameter choices, our scheme provides fast encryption, even for large inputs. For instance, for $(n_k, \hat{n}_1) = (1, 3)$, our encryption algorithm is several factors faster than RW13, which may be desirable for resource-constrained devices. Recently, Scott [Sco20] tested the performance of some operations on the BLS12-381 curve on IoT devices. These results show that the devices with the slowest and fastest timings would approximately require 1.175 and 0.075 seconds, respectively, per exponentiation in $\mathbb{G}$. In Figure 8.4, we estimate the encryption costs for the two fastest devices measured in [Sco20]. For the fastest device, our most efficient configuration requires only 7.6 seconds to encrypt with a policy of length 100, while RW13 requires one minute and 15 seconds.

8.6 Future work

Our work paves the way for further improvements in the design and implementation of ABE in IoT settings. While we have theoretically analyzed the feasibility of implementing our schemes in practical settings such as low-quality networks and embedded devices, it would be useful to empirically test them. Notably, our conservative estimates for the IoT devices in Section 8.5.3 are based on the benchmarks in [Sco20]. By implementing the scheme—possibly using optimizations (e.g., through precomputation [FA17]) that have not been used in [Sco20] or using curves with more efficient arithmetic in $\mathbb{G}$ [CDS20]—it may perform even better than our estimates suggest. Finally, similarly as for GLUE (Section 7.8), we may want to achieve security under a non-parametrized assumption.

8.7 Conclusion

We proposed a new configurable unbounded large-universe CP-ABE scheme, mainly designed for settings with embedded devices or low-quality networks. We have proven the scheme secure in the AC17 framework, yielding efficient constructions that are provably fully secure. TinyABE can be configured such that encryption is very efficient, outperforming state-of-the-art CP-ABE schemes by several factors for large policies. Additionally, the ciphertexts are much shorter than those of schemes with linear-size ciphertexts, and are therefore more likely to fit in the constrained memories of embedded devices. Due to this shortness, ciphertexts are also much less likely to drop during transmission. While the ciphertexts are longer than those of schemes with constant-size ciphertexts, our public and secret keys are much shorter, and the computational costs are much lower. For these reasons, TinyABE is more practical for embedded devices and low-quality networks.

Chapter 9

Efficient and generic transformations for chosen-ciphertext secure PE

Although much progress has been made to generically achieve CPA-security efficiently, in practice, we also require CCA-security. Because achieving CCA-security on a case-by-case basis is a complicated task, several generic conversion methods have been proposed. However, these conversion methods may incur a significant efficiency trade-off. Notably, for CP-ABE, all generic conversion methods provide a significant overhead in the key generation, encryption or decryption algorithm. Additionally, many generic conversion techniques use one-time signatures to achieve authenticity, which are also known to significantly impact the efficiency.

In this chapter, we present a new approach to achieving CCA-security as generically and efficiently as possible, by splitting the CCA-conversion into two steps. The predicate of the scheme is first extended in a certain way, which is then used to achieve CCA-security generically e.g., by combining it with a hash function. To facilitate the first step efficiently, we also propose a novel predicate-extension transformation for a large class of pairing-based PE—covered by the pair and the predicate encodings frameworks—which incurs only a small constant overhead for all algorithms. In particular, this yields the most efficient generic CCA-conversion for CP-ABE.

9.1 Introduction

Over the years, many works have systematized and improved the techniques to achieve full security of pairing-based PE against chosen-plaintext attacks [Wee14, Att14a, CGW15, Att16, AC16, AC17b]. While these frameworks support CPA-security, in practice, it is often recommended or required that the scheme also provides security against chosen-ciphertext attacks (CCA) [NY90, Sho98]. To this end, many works have proposed CCA-secure PE schemes, e.g., [BF01, KG06, Gen06, KV08, TKN21].

Moreover, to achieve CCA-security generically, any of the proposed transformations can be used, e.g.,

- using non-interactive zero-knowledge (NIZK) proofs of well-formedness [BFM88];
- Fujisaki-Okamoto (FO) [FO99, HHK17];
- Canetti-Halevi-Katz (CHK) [CHK04];
- Boneh-Katz (BK) [BK05];
- Abe et al. (ACIK) [ACIK10];
- Yamada et al. (YA(SS)HK) [YAHK11, YAS$^+$12], which consists of two transformations: one for delegatable ABE and one for verifiable ABE;
- Blömer-Liske (BL) [BL16];
- Koppula-Waters (KW) [KW19a].

However, each of these generic transformations has a drawback. First, the transformation may be restricted to, e.g., hierarchical IBE (HIBE) [CHK04, BK05, ACIK10] or ABE [YAHK11, YAS$^+$12]. Second, the FO-transform, the NIZK approach, and the transformations for verifiable schemes [YAHK11, YAS$^+$12, BL16] incur an additional cost during decryption that is linear in the sizes of x and y, which is a significant cost for many ABE or inner-product encryption [KSW08] schemes. Alternatively, these additional costs may be linear in the security parameter[1], such as in KW [KW19a] and the transformations for delegatable CP-ABE [YAHK11]. Notably, for CP-ABE, no CCA-transformations yield a small and constant overhead.

In addition, most of these transformations—except for the NIZK, FO, BK and ACIK-transformations—use one-time signatures (OTS) to achieve authenticity of the ciphertexts. OTSs incur a considerable trade-off in storage and computational efficiency: either signing is efficient but the keys and signatures are large, or the keys and signatures are short but signing is inefficient. The BK-transformation improves on the CHK-transformation by replacing the OTS by a message authentication code (MAC) and a primitive called “encapsulation”, which can be constructed from a hash and yields no such efficiency trade-off [BK05]. Encapsulation allows the encrypting user to commit to a secret value, which is later used to compute a MAC on the ciphertext to attain ciphertext authenticity. Subsequently, the ACIK-transformation improves on the BK-transformation by applying a primitive called a “random-prefix collision-resistant” hash directly to the ciphertext. The encrypting user is therefore not required to commit to a secret value (which also needs to be encrypted), and thus, minimizes the storage overhead.

[1] Typically, the security parameter is fixed, e.g., equal to 128. Nevertheless, the additional costs are large.

Table 9.1. Comparison of the properties of the several CCA-transformations and our new transformations. For our CCA-transformations, we also consider alternative pathways based on the existing transformations to perform the two steps.

Variant	**Primitives used**	**Applicable to**	**Requirements**
FO [FO99, HHK17]	hash	All PE	-
CHK [CHK04]	OTS	(H)IBE	-
BK [BK05]	encapsulation, MAC, PRG	(H)IBE	-
ACIK [ACIK10, §7.2]	RPC	(H)IBE	partitioned KEM
YA(SS)HK [YAHK11, YAS$^+$12]	OTS	ABE	delegatable or verifiable ABE
BL [BL16]	hash	All PE	verifiable pair encodings
KW [KW19a]	PRG, OTS	All PE	-
Step 1 with CHK/BK	-	(H)IBE	-
Step 1 with YA(SS)HK	-	ABE	delegatable ABE
Step 2 with BK	encapsulation, MAC, PRG	All PE	-
Step 1 (new)	-	All PE	pair and predicate encodings
Step 2 (new)	RPC	All PE	decomposable PE

Note: PRG = pseudo-random generator,
RPC = random-prefix collision-resistant hash

9.1.1 Our contribution

In this chapter, we focus on generically achieving CCA-security for any PE as efficiently as possible. To this end, we propose a new high-level approach in the design of CCA-security transformations, by splitting any such transformations into two explicit steps. In the first step, the predicate of the scheme is extended. In the second step, the predicate-extended scheme is used to achieve CCA-security. Although several existing transformations take these steps implicitly, explicitly considering them as two steps may lead to more efficient (yet generic) constructions than previous methods allowed. To illustrate that, we propose two novel transformations that perform these two steps efficiently.

Warm-up: existing transformations. Apart from the NIZK, FO and KW transformations, all aforementioned generic transformations exploit the structure of the predicate to efficiently achieve CCA-security. Roughly, they all follow a similar approach: during encryption, the encrypting user commits to some value, which is then embedded in the predicate in addition to the original predicate. For example, for (H)IBE [CHK04, BK05, ACIK10], this value is embedded in the (additional "layer" of the) identity, and for delegatable CP-ABE, the bit-representation of the value is encoded as an AND-policy (and is taken in conjunction with the original policy). Because these transformations exploit the specific structures of the predicates, they are therefore only applicable to those predicates. Furthermore, depending on the technique, the value to which is commited is either generated independently of the ciphertext [CHK04, BK05, YAHK11] or by applying a hash with a specific property to the ciphertext [ACIK10, BL16]. Although the latter requires that the ciphertext is of a specific structure (which many pairing-based schemes satisfy), it relies on fewer primitives and yields less storage overhead in the ciphertext.

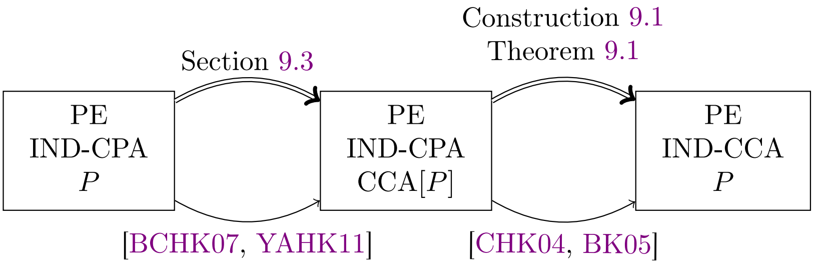


Figure 9.1. A high-level overview of the transformations and our associated definitions and theorems that prove security of the given transformations. The double-edged arrows indicate that we give a novel provably secure generic transformation in this chapter, while the normal-edged arrows provide transformations that have been given in other works.

Our transformations. On a high level, our approach consists of two steps with varying "levels of genericness" (of which an overview is shown in Figure 9.1). First, we transform a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P}$ for predicate P into a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P'. For this step, we propose novel generic constructions in the pair and predicate encodings frameworks. (We also show that our transformation for predicate encodings preserves the attribute-hiding property in [CGW15].) As a result, many pairing-based PE schemes can be transformed using this construction. Second, we transform any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P' into a CCA-secure PE $\Gamma_{\text{PE,IND-CCA},P}$ for the original predicate P. This step can be done by using similar approaches as CHK and BK. We also give a new transformation based on the ACIK-approach. This new transformation applies to any PE scheme for which the ciphertexts are "decomposable" (which is a similar notion to that of partitioned in [ACIK10]).

Although our transformation is less generic than fully generic transformations such as FO and KW, ours is more generic than most of the other transformations (Table 9.1). In fact, our approach can be seen as an efficient generalization of the transformations that exploit the specific structures of the predicate, i.e., CHK, BK, ACIK, and YA(SS)HK. However, by performing the transformation in two steps, we also allow for more efficient (yet generic) constructions. We show that this is especially beneficial for CP-ABE, for which existing such transformations always induce a linear computational overhead in at least one of the algorithms.

Step one: securely extending the predicate. We first extend the predicate P to some predicate $P' = \text{CCA}[P]$. The idea behind this is similar to the approach for hierarchical IBE [CHK04, BK05, BCHK07] and delegatable KP-ABE by Yamada et al. (YAHK) [YAHK11], and is later also applied using wildcards by Tomida et al. [TKN21]. Roughly, the secret key predicate y is extended to $y \wedge y'$, where y' is either an attribute or a wildcard $*$, and the ciphertext predicate x is extended to (x, x'), where x' is an attribute. The predicate is satisfied if $P(x, y)$ and either $y' = *$ or $y' = x'$ holds. We provide a new transformation in the pair and predicate encodings framework that extends the predicate in this way. The computational overhead incurred by our transformation is a low constant, and unlike YAHK, we do not require the PE scheme to be a delegatable KP-ABE for the transformation to work. Because we generically transform *any* PE into a scheme with this specific extended predicate, we can also efficiently support CCA-security in, e.g., CP-ABE. Roughly, we take an AND-composition over the original PE and an "all-or-one-identity" IBE, by using the ideas from Ambrona et al. [ABS17] and Attrapadung [Att19]. For the "all-or-one-identity" IBE, we use the first scheme of Kiltz and Vahlis [KV08] as inspiration, which is essentially implied by a composition of the Boneh-Boyen (BB) IBE [BB04] with a wildcard variant of the same scheme.

Step two: achieving CCA-security. We first consider on a high level what the CCA-transformation looks like. Let $\Gamma = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a predicate key-encapsulation scheme (possibly derived from a PE) for the extended predicate, such that that ciphertext is of the form

$$\mathrm{Encaps}(\mathrm{MPK}, (x, x')) = (\mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}),$$

where MPK is the master public key generated in the Setup, K is the encapsulated key to be used to symmetrically encrypt, CT_1 is some randomized part of the ciphertext that is independent of extended predicate (x, x'), and $\mathrm{CT}_{2,(x,x')}$ denotes the rest of the ciphertext. Following the approach by Kiltz and Vahlis [KV08] and Abe et al. (ACIK) [ACIK10], we first split the key-encapsulation algorithm into two parts, and then introduce an authenticated encryption scheme $\mathrm{SE} = (\mathrm{Enc}_\mathrm{K}, \mathrm{Dec}_\mathrm{K})$ and a random-prefix collision-resistant hash function RPC (which takes as input a random prefix k and another input to be hashed), i.e.,

$$\mathrm{Encaps}(\mathrm{MPK}, (x, x')) = (\underbrace{\mathrm{K}, \mathrm{CT}_1}_{\mathrm{Encaps}_1}, \underbrace{\mathrm{CT}_{2,(x,x')}}_{\mathrm{Encaps}_2}).$$

Then, we obtain the CCA-transformed encryption as follows

$$\mathrm{Encrypt}'(\mathrm{MPK}, \boxed{x}, M) = (\boxed{\mathrm{CT}_{\mathrm{sym}} = \mathrm{Enc}_\mathrm{K}(M \| \mathrm{CT}_{2,(x,x')})}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k),$$

where $(\mathrm{K}, \mathrm{CT}_1) \leftarrow \mathrm{Encaps}_1(\mathrm{MPK})$, $k \in_R \{0,1\}^\lambda$, $x' \leftarrow \mathrm{RPC}(k, \mathrm{CT}_1)$, and then $\mathrm{CT}_{2,(x,x')} \leftarrow \mathrm{Encaps}_2(\mathrm{MPK}, (x, x'))$.

Proving CCA-security. We prove CCA-security of the proposed generic construction similarly as other transformations [CHK04, BK05, KV08, ACIK10, YAHK11].

Specifically, the decryption queries are answered as follows. Suppose that $\mathrm{CT}_x = (\mathrm{CT}_{\mathrm{sym}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')})$ is some ciphertext and y is some predicate such that $P(x,y) = 1$, queried by the attacker. Then, the challenger can generate a secret key for $(y, y' = x')$, and decrypt the ciphertext. The challenger rejects a decryption query if it is similar to the challenge ciphertext $\mathrm{CT}^*_{x^*} = (\mathrm{CT}^*_{\mathrm{sym}}, \mathrm{CT}^*_1, \mathrm{CT}_{2,(x^*,x'^*)})$, i.e., if $\mathrm{CT}_1 \neq \mathrm{CT}^*_1$ and $x' = x'^*$, or if $\mathrm{K} = \mathrm{K}^*$ and $\mathrm{CT}_{\mathrm{sym}} \neq \mathrm{CT}^*_{\mathrm{sym}}$ or $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$. Intuitively, the probability that a valid ciphertext is rejected—i.e., the probability that a valid ciphertext satisfies any of these conditions—is negligible due to the random-prefix collision resistance of the hash RPC and the authenticity of the symmetric encryption scheme $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$.

Alternative pathways. As mentioned, we focus on achieving CCA-security as efficiently and as generically as possible. Although the two proposed transformations for the two steps are applicable to large classes of existing PE schemes, they do not apply to *all* PE schemes. For example, post-quantum schemes [AFV11, ABV$^+$12, Boy13, GVW13] are not covered by our predicate-extension transformation, and not all schemes may be decomposable and therefore qualify for our second-step transformation. To make our second step more generic, one could also use the BK-approach [BK05], which does not require the extended-predicate scheme to have ciphertexts with a certain structure[2]. However, it does provide more storage overhead and relies on more primitives (i.e., two independent hash functions, a MAC and a PRG). The latter may be undesirable in practice, e.g., because no suitable implementations are available of all primitives. In this regard, our second-step transformation could provide an effective solution, as it requires only one hash function. Importantly, because the second step can be done entirely generically, the effort of achieving CCA-security is reduced to finding an efficient predicate extension. Note that several such predicate-extension techniques have been described implicitly, e.g., the CHK- and YAHK-approaches extend the (H)IBE with another level in the hierarchy and extend the ABE ciphertext predicate with a conjunction or disjunction, respectively. Furthermore, for schemes for which there is no such predicate-extension technique available (that is sufficiently efficient), we only need to devise an efficient predicate-extension transformation, instead of performing a full-fledged CCA-security conversion.

9.1.2 Performance analysis and comparison

We compare the efficiency of our CCA-transformation with the others. From a theoretical standpoint, ours is the most efficient. It incurs only a small constant overhead in all algorithms and the key and ciphertext sizes in the first step, regardless of the size of the predicate. For all other transformations, this is not the case. Especially

[2] The security proof of the generalized variant of the BK-transformation is analogous to that of the BK-transformation itself.

for schemes with linear-sized predicates, such as ABE, this provides a significant efficiency improvement. In contrast, the other approaches applicable to ABE incur the following efficiency trade-offs:

- FO [FO99, HHK17]: in general, this approach incurs little to no overhead to most algorithms, except for the decryption algorithm, which requires an invocation of the encryption algorithm, whose costs are often linear;
- YAHK-del [YAHK11]: depending on the type of ABE, this transformation for delegatable schemes might either be very efficient or very costly. For KP-ABE, the transformation incurs only a small constant overhead in all algorithms and the key and ciphertext sizes. For CP-ABE, the transformation incurs an additional overhead that is linear in the security parameter in the encryption and decryption algorithms;
- YAHK-ver [YAHK11], BL [BL16]: these transformations for verifiable schemes incur little to no overhead in most of the algorithms and the key and ciphertext sizes, except for the decryption algorithm, which also verifies whether the ciphertexts are well-formed. The costs incurred by the verification step are similar to the decryption costs of the CPA-secure PE scheme, and therefore roughly double the decryption costs of the CCA-secure PE scheme (which are often linear in the predicate size);
- KW [KW19a]: this fully-generic transformation is very costly and incurs an overhead in all algorithms and sizes that is linear in the security parameter.

In Section 9.4, we analyze the performance of RW13 to show the advantage of our transformation compared to existing transformations. In particular, we analyze the costs of the CCA-secure variants of the fully secure version of RW13 [RW13] in the pair encodings framework, i.e., RWAC (Definition 4.4). We compare our CCA-variant with those that follow from applying FO and the transformations for delegatable and verifiable CP-ABE. Our analysis shows that our transformation has a much faster decryption than all existing transformations, while incurring a marginal overhead in the other algorithms compared to the fastest variants.

9.1.3 Organization

This chapter is structured as follows. We first give, in Section 9.2, the fully generic transformations from any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P' into a CCA-secure PE $\Gamma_{\text{PE,IND-CCA},P}$ for original predicate P, i.e., steps 2 and 3. After this, in Section 9.3, we propose novel generic transformations from any CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P}$ for predicate P to a CPA-secure PE $\Gamma_{\text{PE,IND-CPA},P'}$ for extended predicate P', i.e., step 1. We first give the more general steps of the transformation and then the less generic step, due to the "level of genericness". Finally, we compare the performance of our transformation in Section 9.4, and conclude the chapter in Sections 9.5 and 9.6 by discussing future directions.

9.2 Our generic CCA-transformation

We introduce our generic transformation for CCA-secure PE.

9.2.1 Step one: extending the predicate

Let $\Gamma_{\text{PE,IND-CPA},P} = (\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ be a predicate encryption scheme for the predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. In the first step of our approach, we transform it into a scheme $\Gamma_{\text{PE,IND-CPA},P'}$ for predicate $P' = \text{CCA}[P]$, where $\text{CCA}[P]$ denotes the predicate extension required by the CCA-transformation on predicate P, i.e., $P'\colon \mathcal{X}' \times \mathcal{Y}' \to \{0,1\}$, where

- $\mathcal{X}' = \mathcal{X} \times \mathcal{Z}$ and $\mathcal{Y}' = \mathcal{Y} \times (\mathcal{Z} \cup \{*\})$, where $|\mathcal{Z}| \geq 2^{2\lambda}$;
- $P'((x, x'), (y, y')) = 1$ if and only if
  - $P(x, y) = 1$ and $y' = *$;
  - or $P(x, y) = 1$ and $x' = y'$.

In Section 9.3, we give several predicate-extension transformations that generically transform a CPA-secure PE scheme for the predicate P in a CPA-secure PE scheme for predicate $\text{CCA}[P]$. Conceptually, we do this by making an AND-composition of the original PE scheme for predicate P with an “all-or-one-identity” IBE. In an “all-or-one-identity” IBE, a user is given either a key for one particular identity $y' \in \mathcal{Z}$ or all identities $y' = *$. These transformations are not fully generic, because they only apply to pairing-based ABE. In particular, they are given in the pair encodings framework (Chapter 4).

We note that a scheme with an extended predicate can also be obtained in other ways. For instance, the approaches used for (H)IBE [CHK04, BK05, BCHK07] also apply. Additionally, the generic transformations using delegation by Yamada et al. [YAHK11] yield suitable candidates as well, but only for KP-ABE and CP-ABE. Furthermore, the transformation by Tomida et al. [TKN21] using delegation is similar to our proposed constructions in Section 9.3, but these only work for their specific KP-ABE and CP-ABE schemes, and are not generic in the sense that they can be applied to any PE. Additionally, while our transformations in Section 9.3 are specific to pairing-based PE, they may also work for PE based on other cryptographic assumptions, for instance, by creating an “all-or-one-identity” IBE from a suitable IBE from post-quantum assumptions [GPV08], and taking an AND-composition with any post-quantum PE [AFV11, ABV$^+$12, Boy13, GVW13].

9.2.2 Step two: generic CCA-secure construction

Much like in [KV08] and [ACIK10], the predicate extension is generated from part of the ciphertext. To this end, we introduce the notion of "decomposable extended-predicate encryption (EPE)", which we use as input to the CCA-security transformation. In decomposable EPE, we decompose the ciphertext in three parts, such that one of the parts is used to generate the predicate extension with the hash.

Definition 9.1: Decomposable EPE

An EPE scheme with encryption algorithm Encrypt is called decomposable if the ciphertexts are decomposable. The ciphertexts are decomposable if $\mathrm{CT}_{(x,x')} \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M)$ can be decomposed as follows:

$$\mathrm{CT}_{(x,x')} = (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}), \text{ such that}$$

- only $\mathrm{CT}_{2,(x,x')}$ depends on (x, x');
- only CT_{M} contains the message;
- CT_{M} is uniquely determined by M, MPK and CT_1, and conversely, CT_1 is uniquely determined by M, MPK and CT_{M};
- $\mathrm{CT}_1 \in \mathcal{G}$ is generated independently of $\mathrm{CT}_{2,(x,x')}$;
- for any $(\hat{x}, \hat{x}') \in \mathcal{X}'$ with $\hat{x}' \neq x'$, we have that any $\mathrm{CT}_{2,(\hat{x},\hat{x}')}$ that is valid for CT_1 is such that $\mathrm{CT}_{2,(\hat{x},\hat{x}')} \neq \mathrm{CT}_{2,(x,x')}$;
- CT_1 is generated uniformly at random over $\mathcal{G}$, such that $\Pr[\mathrm{CT}_1 = \mathrm{CT}_1' \mid \mathrm{CT}_1' \in_R \mathcal{G}] \leq \mathrm{negl}(\lambda)$.

In this case, we also define two algorithms for encryption, i.e.,

- $\mathrm{Encrypt}_1(\mathrm{MPK}, M) \to (\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1)$;
- $\mathrm{Encrypt}_2(\mathrm{MPK}, (x, x')) \to \mathrm{CT}_{2,(x,x')}$,

such that

$$\mathrm{Encrypt}(\mathrm{MPK}, (x, x'), M) = (\mathrm{Encrypt}_1(\mathrm{MPK}, M), \mathrm{Encrypt}_2(\mathrm{MPK}, (x, x'))).$$

Decomposable EP-KEM. This definition also naturally extends to the key-encapsulation variants of EPE, i.e., by replacing CT_{M} by the encapsulated symmetric key. We can generically obtain a EP-KEM from a EPE by encrypting a randomly-generated symmetric key K. For PE schemes with a certain algebraic structure, we can also generically obtain a more efficient KEM.

More efficient transformation from PE to P-KEM. Let $\Gamma_{\mathrm{PE}} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encrypt}, \mathrm{Decrypt})$ be a decomposable predicate encryption scheme for the predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. Suppose that the operation on the group in which CT_{M} lives is multiplicative[3] and its operator is $\cdot$, and in particular, that $\mathrm{CT}_{\mathrm{M}} = M \cdot \mathrm{rand}$, where rand is some random element in the group in which CT_{M} lives. Let id denote the identity in this group. Then, we can generically define Encaps and Decaps from Encrypt and Decrypt as follows.

- $\mathrm{Encaps}(\mathrm{MPK}, x)$: Let $(\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x}) \leftarrow \mathrm{Encrypt}(\mathrm{MPK}, x, \mathrm{id})$. Then, this algorithm outputs $\mathrm{K} = \mathrm{CT}_{\mathrm{M}}$ as the symmetric key and $(\mathrm{CT}_1, \mathrm{CT}_{2,x})$ as the rest of the ciphertext.
- $\mathrm{Decaps}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x)$: This algorithm outputs the decapsulated symmetric key as $\mathrm{K}' \leftarrow \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1}$.

The correctness of the P-KEM follows from the correctness of the PE:

$$\begin{aligned}\mathrm{Decaps}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{CT}_x) &= \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1} \\ &= \mathrm{K} \cdot \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, (\mathrm{id} \cdot \mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,x}))^{-1} \\ &= \mathrm{K} \cdot \mathrm{Decrypt}(\mathrm{MPK}, \mathrm{SK}_y, \mathrm{Encrypt}(\mathrm{MPK}, x, \mathrm{id}))^{-1} = \mathrm{K} \cdot \mathrm{id}^{-1} = \mathrm{K}.\end{aligned}$$

The CPA-security of the P-KEM also follows readily from the PE. Let $\mathcal{A}_{\text{P-KEM}}$ be an attacker on the P-KEM, i.e., which can distinguish for a given $(\mathrm{K}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ whether K is a symmetric key or K is random. Then, it can be used to construct an attacker $\mathcal{A}_{\mathrm{PE}}$ for the PE scheme. Suppose $(\mathrm{CT}_{\mathrm{M}}, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ is the challenge ciphertext for M_0 or M_1. Then, pick $\beta \in_R \{0,1\}$ and send $(\mathrm{K} = \mathrm{CT}_{\mathrm{M}}/M_\beta, \mathrm{CT}_1, \mathrm{CT}_{2,x})$ to attacker $\mathcal{A}_{\text{P-KEM}}$. If it outputs that K is a symmetric key, then attacker $\mathcal{A}_{\mathrm{PE}}$ outputs β as the guess, and otherwise, it outputs $1-\beta$ as the guess. The advantage of $\mathcal{A}_{\text{P-KEM}}$ is equal to the advantage of $\mathcal{A}_{\mathrm{PE}}$.

9.2.3 Generic CCA-secure construction

We use a CPA-secure decomposable EP-KEM with an extended predicate to generically construct a CCA-secure hybrid PE for the original predicate.

Construction 9.1: Generic CCA-secure construction

Let $\Gamma_{\mathrm{PE}} = (\mathrm{Setup}, \mathrm{KeyGen}, \mathrm{Encaps}, \mathrm{Decaps})$ be a predicate key-encapsulation mechanism for the predicate $P\colon \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, and suppose $\Gamma_{\text{EP-KEM}} = (\mathrm{Setup}_{\text{EP-KEM}}, \mathrm{KeyGen}_{\text{EP-KEM}}, \mathrm{Encaps}_{\text{EP-KEM}}, \mathrm{Decaps}_{\text{EP-KEM}})$ is a decomposable extended-predicate KEM for predicate $P' = \mathrm{CCA}[P]$ (e.g., obtained with a predicate-extension transformation (Section 9.3)). Let $\mathrm{SE} = (\mathrm{Enc}_{\mathrm{K}}, \mathrm{Dec}_{\mathrm{K}})$ be an

[3] Something similar works for additive groups as well.

authenticated symmetric encryption scheme with key space $\mathcal{K}_\lambda$ equal to the space in which $\mathrm{CT_M}$ lives, and $\mathrm{RPC}\colon \{0,1\}^\lambda \times \mathcal{G} \to \mathcal{Z}$ be a random-prefix collision-resistant hash function. Then, we define $\Gamma'_{\mathrm{PE}} = (\mathrm{Setup}', \mathrm{KeyGen}', \mathrm{Encrypt}', \mathrm{Decrypt}')$ to be the CCA-secure hybrid encryption version of scheme Γ_{PE} for predicate P as

- $\mathrm{Setup}'_{\mathrm{PE}}(\lambda, \mathrm{par})$: On input λ and par, the setup generates $(\mathrm{MPK}, \mathrm{MSK}) \leftarrow \mathrm{Setup}_{\text{EP-KEM}}(\lambda, \mathrm{par})$, and sets $\mathrm{MPK}' = \mathrm{MPK}$ and $\mathrm{MSK}' = \mathrm{MSK}$.

- $\mathrm{KeyGen}'_{\mathrm{PE}}(\mathrm{MSK}', y)$: On input the master secret key MSK' and some $y \in \mathcal{Y}$, it returns $\mathrm{SK}'_y \leftarrow \mathrm{KeyGen}_{\text{EP-KEM}}(\mathrm{MSK}, (y, *))$.

- $\mathrm{Encrypt}'_{\mathrm{PE}}(\mathrm{MPK}', x, M)$: On input the master public key MPK', $x \in \mathcal{X}$ and message $M \in \{0,1\}^*$, the encrypting user computes $(\mathrm{K}, \mathrm{CT}_1) \leftarrow \mathrm{Encaps}_{1,\text{EP-KEM}}(\mathrm{MPK})$, picks $k \in_R \{0,1\}^\lambda$ and sets $x' = \mathrm{RPC}(k, \mathrm{CT}_1)$, then generates $\mathrm{CT}_{2,(x,x')} \leftarrow \mathrm{Encaps}_{2,\text{EP-KEM}}(\mathrm{MPK}, (x, x'))$, and computes[†] $\mathrm{CT}_{\mathrm{sym}} \leftarrow \mathrm{Enc}_{\mathrm{K}}(M \| \mathrm{CT}_{2,(x,x')})$, and returns

$$\mathrm{CT}'_x = (\mathrm{CT}_{\mathrm{sym}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k).$$

- $\mathrm{Decrypt}'_{\mathrm{PE}}(\mathrm{MPK}', \mathrm{SK}'_y, \mathrm{CT}'_x)$: On input the master public key MPK', the secret key SK'_y, and the ciphertext $\mathrm{CT}'_x = (\mathrm{CT}_{\mathrm{sym}}, \mathrm{CT}_1, \mathrm{CT}_{2,(x,x')}, k)$, if $P(x, y) = 1$, then the decrypting user computes $x' = \mathrm{RPC}(k, \mathrm{CT}_1)$ and

$$\mathrm{K}' \leftarrow \mathrm{Decaps}_{\text{EP-KEM}}(\mathrm{MPK}, \mathrm{SK}_{(y,*)}, (\mathrm{CT}_1, \mathrm{CT}_{2,(x,x')})).$$

  The user computes $(M' \| \mathrm{CT}'_{2,(x,x')}) \leftarrow \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}})$, and if $\mathrm{CT}'_{2,(x,x')} = \mathrm{CT}_{2,(x,x')}$, returns M'.

[†] If one uses an authenticated encryption scheme with associated data [Rog02], one can also treat $\mathrm{CT}_{2,(x,x')}$ as associated data, as it does not need to be secret.

Correctness. The scheme is correct, i.e., if $P(x, y) = 1$, then $M' = M$, because the correctness of the P-KEM ensures that $\mathrm{K} = \mathrm{K}'$, and thus, $(M' \| \mathrm{CT}'_{2,(x,x')}) = \mathrm{Dec}_{\mathrm{K}'}(\mathrm{CT}_{\mathrm{sym}}) = \mathrm{Dec}_{\mathrm{K}}(\mathrm{CT}_{\mathrm{sym}}) = \mathrm{Dec}_{\mathrm{K}}(\mathrm{Enc}_{\mathrm{K}}(M \| \mathrm{CT}_{2,(x,x')})) = (M \| \mathrm{CT}_{2,(x,x')})$.

Security. We prove the following theorem.

Theorem 9.1

In Definition 9.1, if $\Gamma_{\text{EP-KEM}}$ is a decomposable CPA-secure P-KEM for the extended predicate CCA$[P]$, RPC is a random-prefix collision-resistant hash function and SE $= (\text{Enc}_\text{K}, \text{Dec}_\text{K})$ is an authenticated encryption scheme, such that the RPC is independent of $\Gamma_{\text{EP-KEM}}$ and SE, then Γ'_{PE} is CCA-secure.

Proof. We prove this theorem in a series of games in which we start with the real CCA-security game: Game 0. Let $\text{CT}^*_{x^*} = (\text{CT}^*_{\text{sym}}, \text{CT}^*_1, \text{CT}^*_{2,(x^*,x'^*)}, k^*)$ denote the challenge ciphertext (with K^* being the symmetric key) for the challenge predicate x^* and message M_β. Let q be the number of decryption queries, and let X_i denote the event that attacker $\mathcal{A}_{\text{CCA}}$ is successful in Game i.

Game 1: In this game, everything is the same as in Game 0, except that, in the first query phase, all decryption queries with $\text{CT}_1 = \text{CT}^*_1$ are rejected. Additionally, in both query phases, the decryption queries with $(\text{CT}_1, k) \neq (\text{CT}^*_1, k^*)$ and $x' = x'^*$ are rejected. The probability that $\text{CT}_1 = \text{CT}^*_1$ holds for any honestly generated ciphertext is $\frac{1}{\mathcal{G}}$. Furthermore, the probability that any x' for $(\text{CT}_1, k) \neq (\text{CT}^*_1, k^*)$ is such that $\text{RPC}(k, \text{CT}_1) = x' = x'^* = \text{RPC}(k^*, \text{CT}^*_1)$ is equal to $\Pr[(\text{CT}_1, k) \neq (\text{CT}^*_1, k^*) \wedge \text{RPC}(k, \text{CT}_1) = \text{RPC}(k^*, \text{CT}^*_1)] = \mathsf{Adv}_{\text{RPC}}$. Hence, we have

$$|\Pr[X_0] - \Pr[X_1]| \leq \frac{q}{\mathcal{G}} + \mathsf{Adv}_{\text{RPC}}.$$

Game 2: In this game, everything is the same as in Game 1, except that, in the second query phase, all decryption queries are rejected where $\text{CT}_{\text{sym}} \neq \text{CT}^*_{\text{sym}}$ holds, and the key $\text{K} \leftarrow \text{Decaps}_{\text{EP-KEM}}(\text{MPK}, \text{SK}_{(y,*)}, \text{CT}_x)$ is such that $\text{K} = \text{K}^*$. Because this property can only hold if the ciphertext authenticity of the SE is broken, we have

$$|\Pr[X_1] - \Pr[X_2]| \leq \mathsf{Adv}_{\text{SE,CAUT}}.$$

Game 3: In this game, everything is the same as in Game 2, except that, in the second query phase, all valid decryption queries are rejected where $\text{CT}_{2,(x,x')} \neq \text{CT}^*_{2,(x^*,x'^*)}$ holds, and $\text{K} = \text{K}^*$ (and thus, $\text{CT}_1 = \text{CT}^*_1$). Note that this can happen only if the ciphertext authenticity of SE is broken, because the attacker has to generate a valid ciphertext for the same key K^* and another message. Hence, we have

$$|\Pr[X_2] - \Pr[X_3]| \leq \mathsf{Adv}_{\text{SE,CAUT}}.$$

Game 4: At this point, all ciphertexts that are queried in the second phase and that are not rejected are such that, for the keys, it holds that $\text{K} \neq \text{K}^*$. This follows from the fact that K is uniquely determined by MPK and CT_1 (and vice versa), and thus, if $\text{K} = \text{K}^*$, then $\text{CT}_1 = \text{CT}^*_1$. By extension, we have $(\text{CT}_{\text{sym}}, \text{CT}_{2,(x,x')}, k) \neq$

$(\mathrm{CT}^*_{\mathrm{sym}}, \mathrm{CT}^*_{2,(x^*,x'^*)}, k^*)$. (Note that, if $k \neq k^*$, we also have $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$, which follows from rejecting all ciphertexts with $x' = x'^*$ in Game 1. From the fact that the EP-KEM is decomposable, it follows that $x' \neq x'^*$ implies $\mathrm{CT}_{2,(x,x')} \neq \mathrm{CT}^*_{2,(x^*,x'^*)}$.) For these cases, we had rejected the decryption queries (in Games 2 and 3). Because this game is the same as Game 3, we have

$$|\Pr[X_3] - \Pr[X_4]| = 0.$$

Game 5: In this game, everything is the same as in Game 4, except that we generate the challenge ciphertext as follows. Let $\mathcal{O}_{\mathrm{RPC}}$ denote the oracle that finds $k \in \{0,1\}^\lambda$ such that $\mathrm{RPC}(k,g) = z$ for any given $(g,z) \in \mathcal{G} \times \mathcal{Z}$. Because RPC is independent of the P-KEM and symmetric encryption scheme, this does not give the attacker any advantage. Then, the challenger generates $(\mathrm{K}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}) \leftarrow \mathrm{Encaps}_{\text{EP-KEM}}(\mathrm{MPK}, (x^*, x'^*))$ for the challenge predicate x^* and randomly chosen x'^*, and queries the oracle $\mathcal{O}_{\mathrm{RPC}}$ with (CT_1, x'^*), which returns k^* if it exists. (Otherwise, it repeats the process of generating new ciphertexts until the oracle provides some output k^*. This likely succeeds because of the random-prefix collision resistance of the RPC. Intuitively, if many such inputs exist for which the oracle does not return a output, we can also find many g such that there exist at least two k, k' with $\mathrm{RPC}(k,g) = \mathrm{RPC}(k',g)$, which breaks the random-prefix collision resistance of the RPC.) The challenger then outputs the challenge ciphertext as $(\hat{\mathrm{K}}^*, \mathrm{CT}_1, \mathrm{CT}_{2,(x^*,x'^*)}, k^*)$, where $\hat{\mathrm{K}}^*$ is a randomly chosen key that replaces K^*. Because the attacker cannot make decryption queries for K^*, it can only distinguish this game from Game 4 by breaking the CPA-security of the EP-KEM. Therefore, we have

$$|\Pr[X_4] - \Pr[X_5]| \leq \mathsf{Adv}_{\text{EP-KEM,IND-CPA}}.$$

Game 6: In this game, everything is the same as in Game 5, except we replace the challenge message by a randomly generated message of the same length as M_β. By the ciphertext indistinguishability of the symmetric encryption scheme, no attacker can distinguish Game 5 from Game 6, i.e.,

$$|\Pr[X_5] - \Pr[X_6]| \leq \mathsf{Adv}_{\text{SE,CIND}}.$$

Summary: In this final game, because the ciphertext decrypts to a random message, the success probability of the attacker is $\frac{1}{2}$, i.e., $\Pr[X_6] = \frac{1}{2}$. This gives us the following upper bound on the advantage of the attacker in the real security game:

$$\begin{aligned}\mathsf{Adv}_{\text{PE,IND-CCA}} &= \left|\Pr[X_0] - \frac{1}{2}\right| \\ &\leq \frac{q}{\mathcal{G}} + \mathsf{Adv}_{\text{RPC}} + 2\mathsf{Adv}_{\text{SE,CAUT}} + \mathsf{Adv}_{\text{EP-KEM,IND-CPA}} + \mathsf{Adv}_{\text{SE,CIND}}.\end{aligned}$$

Since all advantages on the right-hand side are negligible in λ, it also holds that $\mathsf{Adv}_{\text{PE,IND-CCA}}$ is negligible in λ. □

Remark 9.1

Our proof techniques are similar to the ACIK techniques. In fact, by feeding $\mathrm{CT}_{2,(x,x')}$ through the authenticated symmetric encryption scheme, a part of the proof is more similar to the BK-approach. However, unlike BK, we use the same key K to encrypt and authenticate the message M, and to authenticate $\mathrm{CT}_{2,(x,x')}$. To ensure that this can be done securely, we require MPK, M, CT_{M} and CT_1 to be highly dependent on one another. This property is arguably easier to verify than ACIK's rejection property. Furthermore, we explicitly require CT_1 to be sufficiently random (which is a requirement that is inspired by the KV scheme [KV08]). Lastly, note that our property that, for any $(\hat{x}, \hat{x}') \in \mathcal{X}'$ with $\hat{x}' \neq x'$, we have that any $\mathrm{CT}_{2,(\hat{x},\hat{x}')}$ that is valid for CT_1 is such that $\mathrm{CT}_{2,(\hat{x},\hat{x}')} \neq \mathrm{CT}_{2,(x,x')}$, is similar to ACIK's unique-split property.

9.2.4 Variation with non-decomposable EP-KEM

To convert extended-predicate schemes that do not have decomposable ciphertexts, we can also base our second step of the transformation on a more generic conversion technique than ACIK [ACIK10], such as CHK [CHK04] or BK [BK05]. To apply those techniques, we can treat the extended predicate (x', y') similarly as the identity in those transformations. For example, as in the CHK-transformation, we can embed the verification key in the ciphertext's extended predicate x', and sign the resulting ciphertext with the associated signing key of the one-time signature scheme. Recall, however, that both these methods provide trade-offs in various practical aspects. That is, OTSs provide a significant efficiency trade-off, and the BK-approach induces a higher storage overhead and relies on more primitives.

9.3 New predicate-extension transformations

We give a high-level description of a concrete predicate-extension transformation (for which we provide a formal description in the appendix) for pairing-based ABE. Roughly, this transformation and its security proof follow a similar approach as Attrapadung [Att19]. In particular, we take as input a secure PE scheme (satisfying some properties) and perform a predicate transformation on it, i.e., an AND-composition (on the key) of the original scheme and an "all-or-one-identity" IBE scheme. To this end, we adapt the key-policy augmentation transformation of Attrapadung [Att19]. Our adaptation differs from the original in two ways. First, we ensure that, for the extended key predicate $(y, *)$, we can generate a key for all identities (y, y'). Second, we re-use the randomness used in the keys of the original scheme to randomize the partial "all-or-one-identity" key. In this way, we minimize the amount of randomness, and ultimately, the computational costs.

9.3.1 "All-or-one-identity" IBE

For the predicate extension, we use an "all-or-one-identity" IBE scheme. On a high level, we define the "all-or-one-identity" IBE scheme with identities $x', y' \in \mathbb{Z}_p = \mathcal{Z}$ as follows:

$$\begin{aligned}
\mathrm{MPK}' &= (g, h, e(g,h)^{\alpha s}, g^{b'_0}, g^{b'_1}),\\
\mathrm{SK}'_{y'} &= (h^{\alpha + r(b'_0 + y'b'_1)}, h^r), \mathrm{SK}'_* = (h^{\alpha + rb'_0}, h^{rb'_1}, h^r),\\
\mathrm{CT}'_{x'} &= (M \cdot e(g,h)^{\alpha s}, g^{s(b'_0 + x'b'_1)}, g^s).
\end{aligned}$$

With SK'_*, we can generate $\mathrm{SK}'_{y'}$ for any $y' \in \mathbb{Z}_p$, by computing:

$$h^{\alpha + rb'_0} \cdot \left(h^{rb'_1}\right)^{y'} = h^{\alpha + r(b'_0 + y'b'_1)}.$$

Note that this scheme is similar to the Boneh-Boyen IBE1 scheme [BB04], which is selectively secure, with the modification that it allows for the generation of a secret key that can be used for all identities.

9.3.2 AND-composition with a PE

The transformation of a PE for predicate P to the PE with extended predicate $\mathrm{CCA}[P]$ consists of an AND-composition with the "all-or-one-identity" IBE. For example, consider the following scheme:

$$\begin{aligned}
\mathrm{MPK} &= (g, h, e(g,h)^{\alpha s}, g^{\mathbf{b}}),\\
\mathrm{SK}_y &= (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha, \mathbf{r}, \mathbf{b}, y)}),\\
\mathrm{CT}_x &= (M \cdot e(g,h)^{\alpha s}, g^{\mathbf{s}}, g^{\mathbf{c}(\mathbf{s}, \mathbf{b}, x)}),
\end{aligned}$$

where $\mathbf{r}$, $\mathbf{k}$, $\mathbf{s}$, and $\mathbf{c}$ denote the vectors that describe the secret key and ciphertext, respectively. Then, the transformed scheme is of the form:

$$\begin{aligned}
\mathrm{MPK} &= (g, h, e(g,h)^{\alpha s}, g^{\mathbf{b}}, g^{b'_0}, g^{b'_1}),\\
\mathrm{SK}_{(y,y')} &= \begin{cases} (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1, \mathbf{r}, \mathbf{b}, y)}, h^{\alpha - \alpha_1 + rb'_0}, h^{rb'_1}) & \text{if } y' = *;\\ (h^{\mathbf{r}}, h^{\mathbf{k}(\alpha_1, \mathbf{r}, \mathbf{b}, y)}, h^{\alpha - \alpha_1 + r(b'_0 + y'b'_1)}) & \text{if } y' \in \mathbb{Z}_p, \end{cases}\\
\mathrm{CT}_{(x,x')} &= (M \cdot e(g,h)^{\alpha s}, g^{\mathbf{s}}, g^{\mathbf{c}(\mathbf{s}, \mathbf{b}, x)}, g^{s(b'_0 + x'b'_1)}),
\end{aligned}$$

where $\alpha_1 \in_R \mathbb{Z}_p$. We formulate this transformation in the pair encodings framework in Section 9.3.4. We prove security of the transformation in several ways. We show that the transformation for pair encodings preserves the symbolic property.

9.3.3 Decomposability of the ciphertexts

The resulting extended-predicate encryption scheme is decomposable (and even special decomposable):

$$\mathrm{CT}_{(x,x')} = (\underbrace{M \cdot e(g,h)^{\alpha s}}_{\mathrm{CT_M}}, \underbrace{g^{\mathbf{s}}}_{\mathrm{CT}_1}, \underbrace{g^{\mathbf{c}(\mathbf{s},\mathbf{b},x)}}_{\mathrm{CT}_{2,x}}, \underbrace{g^{s(b'_0+x'b'_1)}}_{\mathrm{CT}_{3,x'}}),$$

and can be easily transformed in a KEM by removing $\mathrm{CT_M}$ and setting $\mathrm{K} = e(g,h)^{\alpha s}$. For a fixed master public key MPK, the key K is then uniquely defined by CT_1 and vice versa, and CT_1 is generated uniformly at random. For $\hat{x}' \neq x'$, we have that $g^{s(b'_0+\hat{x}'b'_1)} \neq g^{s(b'_0+x'b'_1)}$. We can also define a different split, e.g., $\mathrm{CT}_1 = g^s$ and push the rest of $g^{\mathbf{s}}$ in $\mathrm{CT}_{2,x}$. Note that, if one chooses to encapsulate some randomly generated symmetric key $\mathrm{K} = M$, then $M \cdot e(g,h)^{\alpha s}$ should be included in CT_1 to ensure that K is uniquely defined by CT_1 and MPK.

9.3.4 Predicate-extension transformation for pair encodings

We define the predicate-extension transformation for pair encodings as follows.

Construction 9.2: PredEx-Trans **for PES**

Let Γ be a PES for predicate P. Then, we construct a PES for $\mathrm{CCA}[P]$ as follows:

- $\mathrm{Param}'(\mathrm{par}) = \mathrm{Param}(\mathrm{par}) + 2$: The common variables are $\mathbf{b}' = (\mathbf{b}, b'_0, b'_1)$, where $\mathbf{b}$ are the common variables of Γ.
- $\mathrm{EncKey}'((y,y'),p)$: Let $y \in \mathcal{Y}$ and $y' \in \mathbb{Z}_p \cup \{*\}$, and generate $\alpha_1 \in_R \mathbb{Z}_p$, and set $\alpha_2 = \alpha - \alpha_1$. Then, compute $\mathbf{k}^{(1)}(\alpha, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}, y) \leftarrow \mathrm{EncKey}(y,p)$, and replace each occurrence of α by α_1, yielding $\mathbf{k}^{(2)}(\alpha_1, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}, y)$. Additionally, compute

  $$\mathbf{k}^{(3)}(\alpha_2, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{b}', y') = \begin{cases} (\alpha_2 + r_1(b'_0 + y'b'_1)), & \text{if } y' \in \mathbb{Z}_p; \\ (\alpha_2 + r_1 b'_0, r_1 b'_1)), & \text{if } y' = *. \end{cases}$$

  Output $\mathbf{k}(\alpha, \mathbf{r}, \hat{\mathbf{r}}, \mathbf{b}', (y,y')) = (\mathbf{k}^{(2)}, \mathbf{k}^{(3)})$, where $\mathbf{r} = \mathbf{r}^{(1)}$, and $\hat{\mathbf{r}} = (\alpha_1, \hat{\mathbf{r}}^{(1)})$.
- $\mathrm{EncCt}'((x,x'),p)$: Let $x \in \mathcal{X}$ and $x' \in \mathbb{Z}_p$. Compute $\mathbf{c}' = \mathbf{c}'(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}, x) \leftarrow \mathrm{EncCt}(x,p)$. Output $\mathbf{c}(\mathbf{s}, \hat{\mathbf{s}}, \mathbf{b}', (x,x')) \leftarrow (\mathbf{c}', s(b'_0 + x'b'_1))$.

Pair/Correctness. Let $(x,x') \in \mathcal{X}'$ and $(y,y') \in \mathcal{Y}'$ be such that $P'((x,x'),(y,y')) = 1$. In particular, we have $P(x,y) = 1$ and either $y' = *$ or $x' = y'$. Let $(\mathbf{E}', \bar{\mathbf{E}}') \leftarrow$

$\mathsf{Pair}(x, y, p)$, such that $\mathbf{s}\mathbf{E}'(\mathbf{k}_1')^\intercal + \mathbf{c}'\bar{\mathbf{E}}'\mathbf{r}^\intercal = \alpha_1 s$. If $y' = *$, we recover

$$\alpha_2 s = \mathbf{s}\begin{pmatrix} 1 & x' \\ \mathbf{0}^\intercal & \mathbf{0}^\intercal \end{pmatrix}\mathbf{k}_2^\intercal + (s(b_0' + x'b_1'))\begin{pmatrix} 1 & \mathbf{0} \end{pmatrix}\mathbf{r}^\intercal.$$

If $y' \in \mathbb{Z}_p$, we recover

$$\alpha_2 s = \mathbf{s}\begin{pmatrix} 1 \\ \mathbf{0}^\intercal \end{pmatrix}\mathbf{k}_2^\intercal + (s(b_0' + x'b_1'))\begin{pmatrix} 1 & \mathbf{0} \end{pmatrix}\mathbf{r}^\intercal.$$

Finally, we recover $\alpha s = \alpha_1 s + \alpha_2 s$. Note that the output of $\mathsf{Pair}'((x, x'), (y, y'))$ is $(\mathbf{E}, \bar{\mathbf{E}})$, where

$$\mathbf{E} = \begin{cases} \left(\mathbf{E}', \begin{pmatrix} 1 & x' \\ \mathbf{0}^\intercal & \mathbf{0}^\intercal \end{pmatrix}\right) & \text{if } y' = *; \\ \left(\mathbf{E}', \begin{pmatrix} 1 \\ \mathbf{0}^\intercal \end{pmatrix}\right) & \text{if } y' \in \mathbb{Z}_p, \end{cases} \qquad \bar{\mathbf{E}} = \begin{pmatrix} \bar{\mathbf{E}}' \\ \begin{pmatrix} 1 & \mathbf{0} \end{pmatrix} \end{pmatrix}.$$

9.3.5 The PES-transformation preserves Sym-Prop

We show that the predicate-extension transformation preserves the symbolic property (Definition 4.2).

Theorem 9.2

Suppose that Γ satisfies (d_1, d_2)-Sym-Prop^+. Then, $\Gamma' = \mathrm{PredEx\text{-}Trans}(\Gamma)$ for $\mathrm{CCA}[P]$ satisfies $(2d_1, d_2 + 1)$-Sym-Prop^+.

Proof. We define the partial predicate $\bar{P}$ such that $\bar{P}(x', y') = 1$ if and only if $x' = y'$ or $y' = *$. Suppose that $(x, x') \in \mathcal{X}'$ and $(y, y') \in \mathcal{Y}'$ are such that $P'((x, x'), (y, y')) = 0$. This means that $P(x, y) = 0$ or $x' \neq y'$ (with $y' \in \mathbb{Z}_p$) holds (or both). In particular, let EncB, EncR, and EncS output matrix/vector substitutions for the variables α, $\mathbf{b}$, $\mathbf{r}, \hat{\mathbf{r}}$, $\mathbf{s} = (s, s_1, ...)$ and $\hat{\mathbf{s}}$, i.e., $\mathbf{a}, \mathbf{B}^{(1)}, \mathbf{r}^{(1)}, \hat{\mathbf{r}}^{(1)}, \mathbf{s}^{(1)}$, and $\hat{\mathbf{s}}^{(1)}$ (which are vectors of matrices/vectors). For these substitutions, it holds that, if $P(x, y) = 0$, then the polynomials in the encodings evaluate to $\mathbf{0}$.

- **The selective symbolic property:** First, we show that the selective symbolic property holds. We use the substitutions of Γ for the selective symbolic property to substitute the variables and polynomials of Γ' as follows:

$$b_i \;:\; \begin{pmatrix} \mathbf{B}_i^{(1)} & \mathbf{0} \end{pmatrix}, \qquad b_0' \;:\; -x'\mathbf{1}_{1,d_2+1}, \qquad b_1' \;:\; \mathbf{1}_{1,d_2+1},$$

$$r_1 \ : \ \begin{cases} \begin{pmatrix} \beta \mathbf{r}_1^{(1)} \\ \frac{\beta'(1-\beta)}{x'-y'} \end{pmatrix} & \text{if } y' \in \mathbb{Z}_p; \\ \begin{pmatrix} \beta \mathbf{r}_1^{(1)} \\ 0 \end{pmatrix} & \text{if } y = *, \end{cases} \qquad r_{i'} \ : \ \begin{pmatrix} \beta \mathbf{r}_{i'}^{(1)} \\ 0 \end{pmatrix}, \qquad \hat{r}_{i^{(2)}} \ : \ \beta \hat{\mathbf{r}}_{i^{(2)}}^{(1)},$$

$$\alpha \ : \ \mathbf{a}, \qquad \alpha_1 \ : \ \begin{pmatrix} \beta \\ \mathbf{0} \end{pmatrix}, \qquad s \ : \ \mathbf{s}_0^{(1)}, \qquad s_j \ : \ \mathbf{s}_j^{(1)}, \qquad \hat{s}_{j'} \ : \ (\mathbf{s}_{j'}^{(1)}, 0),$$

for all $i \in [n], i' \in [2, m_1], i^{(2)} \in [m_2], j \in [0, w_1], j' \in [w_2]$, where $\beta = 1 - P(x, y)$ and $\beta' = 1 - \bar{P}(x', y')$. Note that $\frac{\beta'(1-\beta)}{x'-y'}$ is well-defined, because if $y' = x'$, then $\beta' = 0$.

- **The co-selective property:** We also show that the co-selective property holds. We use the substitutions of Γ for the co-selective symbolic property to substitute the variables and polynomials of Γ' as follows:

$$b'_0 \ : \ \begin{cases} \mathbf{1}_{d_1+1,d_2+1}^{2d_1 \times (d_2+1)} + y' \mathbf{1}_{d_1+2,d_2+1}^{2d_1 \times (d_2+1)} - \mathbf{1}_{1,d_2+1}^{2d_1 \times (d_2+1)} - y' \mathbf{1}_{2,d_2+1}^{2d_1 \times (d_2+1)} & \text{if } y' \in \mathbb{Z}_p; \\ \mathbf{1}_{d_1+1,d_2+1}^{2d_1 \times (d_2+1)} - \mathbf{1}_{1,d_2+1}^{2d_1 \times (d_2+1)} & \text{if } y' = *, \end{cases}$$

$$b'_1 \ : \ \begin{cases} \mathbf{1}_{2,d_2+1}^{2d_1 \times (d_2+1)} - \mathbf{1}_{d_1+2,d_2+1}^{2d_1 \times (d_2+1)} & \text{if } y' \in \mathbb{Z}_p; \\ \mathbf{0}^{2d_1 \times (d_2+1)} & \text{if } y' = *, \end{cases}, \qquad b_i \ : \ \begin{pmatrix} \mathbf{0}^{d_1 \times d_2} & \mathbf{0}^{d_1 \times 1} \\ \mathbf{B}_i^{(1)} & \mathbf{0}^{d_1 \times 1} \end{pmatrix},$$

$$r_1 \ : \ \begin{pmatrix} \mathbf{r}_1^{(1)} \\ 1 \end{pmatrix}, \qquad r_{i'} \ : \ \begin{pmatrix} \mathbf{r}_{i'}^{(1)} \\ 0 \end{pmatrix}, \qquad \hat{r}_{i^{(2)}} \ : \ \left(\mathbf{0}^{d_1 \times 1} \hat{\mathbf{r}}_{i^{(2)}}^{(1)} \right), \qquad \alpha \ : \ \mathbf{a},$$

$$\alpha_1 \ : \ \mathbf{1}_{d_1+1}^{2d_1}, \qquad s \ : \ \beta \left(\mathbf{s}_0^{(1)} \quad \mathbf{s}_0^{(1)} \right) + \beta' \left(1 \quad \frac{1}{x'-y'} \quad \mathbf{0}^{2d_1-2} \right),$$

$$s_j \ : \ \left(\mathbf{s}_j^{(1)} \quad \mathbf{0}^{d_1} \right), \qquad \hat{s}_{j'} \ : \ \left(\mathbf{s}_{j'}^{(1)} \quad 0 \right),$$

for all $i \in [n], i' \in [2, m_1], i^{(2)} \in [m_2], j \in [0, w_1], j' \in [w_2]$, where $\beta = 1 - P(x, y)$, and $\beta' = 1 - \bar{P}(x', y')$. Here, we assume that $d_1 \geq 2$.

Thus, Sym-Prop$^+$ holds for Γ'. □

9.3.6 Example: predicate-extended version of RW13

As an example, we apply the transformation to RW13 (Construction 4.2).

Construction 9.3: The predicate-extended version of [RW13]

The predicate-extended version of RW13, obtained by applying Construction 9.2 to the PES of RW13 (Construction 4.2) is defined as follows:

- Param($\emptyset$): Set $n = 6$ and $\mathbf{b} = (b, b', b_0, b_1, b'_0, b'_1)$.

- EncKey$((\mathcal{S}, y'), p)$: On input set of attributes $\mathcal{S}$ and integer $y' \in \mathbb{Z}_p \cup \{*\}$, this algorithm generates $\alpha_1 \in_R \mathbb{Z}_p$, sets $\alpha_2 = \alpha - \alpha_1$, and outputs $\mathbf{k} = (k_0 = \alpha_1 + rb, \{k_{1,\text{att}} = rb' + r_{\text{att}}(b_0 + x_{\text{att}} b_1)\}_{\text{att} \in \mathcal{S}}, \mathbf{k}'_2)$, where $x_{\text{att}} \in \mathbb{Z}_p$ is the integer representation of att and
$$\mathbf{k}'_2 = \begin{cases} (\alpha_2 + r(b'_0 + y'b'_1)), & \text{if } y' \in \mathbb{Z}_p \\ (\alpha_2 + rb'_0, rb'_1)), & \text{if } y' = * \end{cases},$$
defined over non-lone variables $\mathbf{r} = (r, \{r_{\text{att}}\}_{\text{att} \in \mathcal{S}})$ and lone variable $\hat{\mathbf{r}} = (\alpha_1)$. Note that $m_1 = |\mathcal{S}| + 1$ and $m_2 = 1$.

- EncCt$((\mathbb{A}, x'), p)$: On input access policy $\mathbb{A} = (\mathbf{A}, \rho)$ and integer $x' \in \mathbb{Z}_p$, this algorithm outputs $\mathbf{c} = (\{c_{1,j} = \lambda_j + s_j b', c_{2,j} = s_j(b_0 + x_{\rho(j)} b_1)\}_{j \in [n_1]}, c_3 = s(b_0 + x'b_1))$, where $\lambda_j = A_{j,1} sb + \sum_{k \in [2,n_2]} A_{j,k} v_k$, defined over non-lone variables $\mathbf{s} = (s, \{s_j\}_{j \in [n_1]})$ and lone variables $\hat{\mathbf{s}} = (\{v_k\}_{k \in [2,n_2]})$. Note that $w_1 = n_1$ and $w_2 = n_2 - 1$.

- Pair$((\mathbb{A}, x'), (\mathcal{S}, y'), p)$: On input policy $\mathbb{A}$, integer $x' \in \mathbb{Z}_p$, set of attributes $\mathcal{S}$ and integer $y' \in \mathbb{Z}_p \cup \{*\}$, if $\mathbb{A} \models \mathcal{S}$ and $x' = y'$ or $y' = *$, then this algorithm determines $\Upsilon = \{j \in [n_1] \mid \rho(j) \in \mathcal{S}\}$ and $\{\varepsilon_j \in \mathbb{Z}_p\}_{j \in \Upsilon}$ such that $\sum_{j \in \Upsilon} \varepsilon_j \lambda_j = sb$ (Definition 2.5), and outputs two matrices $\mathbf{E} = \mathbf{1}^{(w_1+1) \times m_3}_{0,0} + \sum_{j \in \Upsilon} \varepsilon_j \mathbf{1}^{(w_1+1) \times m_3}_{j,\rho(j)} + \mathbf{E}''$ and $\overline{\mathbf{E}} = -\sum_{j \in \Upsilon} \varepsilon_j \left(\mathbf{1}^{w_3 \times m_1}_{(1,j),0} + \mathbf{1}^{w_3 \times m_1}_{(2,j),\rho(j)} \right) + \mathbf{1}^{w_3 \times m_1}_{w_3,0}$, where
$$\mathbf{E}'' = \begin{cases} \mathbf{1}^{(w_1+1) \times m_3}_{0,m_3-1} + x' \mathbf{1}^{(w_1+1) \times m_3}_{0,m_3}, & \text{if } y' \in \mathbb{Z}_p \\ \mathbf{1}^{(w_1+1) \times m_3}_{0,m_3}, & \text{if } y' = *. \end{cases}$$

It follows directly from Lemma 4.4 and Theorem 9.2 that the PES is secure.

Corollary 9.1

The PES in Construction 9.3 is symbolically secure. Thus, any scheme obtained with our generic compiler (Definition 4.10) is selectively secure, and any scheme obtained with the AC17 generic compiler (Section 4.4) is fully secure.

9.4 Performance analysis of concrete constructions

To illustrate the benefits of our transformation with respect to the efficiency compared to other generic transformation techniques, we analyze the storage and computational costs of the CCA-secure variants of RW13. Specifically, we benchmark the P-KEM part of these variants. (We leave out the KW [KW19a] transformation due to its evident blowup in costs as shown in Section 9.1.2.) We approximate the efficiency of the FO, delegation and verifiability-based transformations. For FO, we add the encryption costs to the decryption costs (for same-length inputs). Note that this also includes the public key storage cost in the secret key size as this is required for re-encryption. For verifiability-based transformations, we add one attribute in the ciphertext-policy input, and multiply the decryption costs by a factor 2. For delegation-based transformations, we assume that the length of the verification key of the used OTS is at least 128 bits (at the 128-bit security level), and thus, that the key set is extended with $2 \cdot 128 = 256$ attributes, and the ciphertext policy with 128 attributes. We have implemented the schemes in Rust[4] using the BLS12-381 crate provided by the zkCrypto group [ZkC20].

Table 9.2 summarizes the benchmarks obtained by running the code on an AMD Ryzen 7 3700X CPU, with a frequency of 4.1 GHz. We observe that our ciphertexts are generally smaller, while the keys are only a little larger. We also observe that our decryption is by far the most efficient, i.e., at least a factor 2 than all other variants. Furthermore, the key generation and encryption costs are only milliseconds slower than the most efficient variants. In conclusion, all transformations except for ours incur a significant trade-off: either attaining a large overhead in the key or ciphertext sizes, or incurring a very large overhead in at least one of the algorithms. In contrast, with respect to the decryption algorithm, our transformation outperforms all other transformations, with incredibly little sacrifice in key generation and encryption efficiency.

9.5 Future work

Throughout this chapter, we have mentioned several interesting directions for future work. Because the second step of the transformation can be done fully generically, it may be used to convert (decomposable) post-quantum PE as well, e.g., by making an AND-composition of a post-quantum PE and "all-or-one-identity" IBE. Finally, while this chapter is general for all PE, it might be applicable to an even larger class of encryption schemes, e.g., functional encryption [BSW11], which contains PE.

9.6 Conclusion

We have presented a new two-step approach to achieving CCA-security generically in PE schemes, which aims to convert PE schemes as efficiently as possible. Additionally, for each of these steps, we have proposed a new transformation. For the second-step transformation, we have generalized the ACIK-transform [ACIK10], which can now be applied to any PE scheme that is decomposable and for which the predicate can be securely extended. Compared to the more generic CHK- and BK-approaches, ACIK

[4] The code is available at `https://github.com/leonbotros/pe_cca`.

Table 9.2. Comparison of the storage and computational costs of the P-KEM part of several CCA-secure variants of the fully secure variant of RW13. The storage costs are expressed in bytes and the timings are expressed in milliseconds. The lowest costs are typeset in **bold**, and for 100 attributes, we also include the increase in costs compared to the lowest costs. We consider inputs of 1, 10 and 100 attributes. (Note that we use compressed point representation to minimize the storage costs.)

Variant	$\lvert\mathrm{MPK}\rvert$	$\lvert\mathrm{SK}_{\mathcal{S}}\rvert$				$\lvert\mathrm{CT}_{\mathbb{A}}\rvert$			
		1	10	100	Increase	1	10	100	Increase
CPA	768	768	4,224	38,784	-	384	2,976	28,896	-
FO	**768**	1,536	4,992	39,552	2%	672	3,264	29,184	1%
Del.	**768**	99,072	102,528	137,088	253%	37,248	39,840	65,760	128%
Ver.	**768**	**768**	**4,224**	**38,784**	0%	672	3,264	29,184	1%
Ours	960	1,152	4,608	39,168	1%	**480**	**3,072**	**29,008**	0.4%

(a) Storage costs

Variant	KeyGen				Encrypt				Decrypt			
	1	10	100	Increase	1	10	100	Increase	1	10	100	Increase
CPA	8.40	46.2	424	-	6.73	46.6	445	-	3.84	16.6	145	-
FO	8.40	46.2	424	0.4%	**6.73**	**46.6**	**445**	0%	10.6	63.1	590	307%
Del.	1082	1121	1499	255%	573	614	1013	127%	186	200	329	127%
Ver.	**8.37**	**46.0**	**422**	0%	11.1	51.0	449	0.9%	10.5	35.8	292	101%
Ours	12.5	50.1	427	1%	8.21	48.2	448	0.7%	**6.0**	**18.7**	**148**	2%

(b) Computational costs

provides less storage overhead and relies on fewer primitives. For the first-step transformation, we have proposed a new predicate-extension transformation that can be applied to any pairing-based schemes that can be captured in the pair and predicate encodings frameworks. Compared to existing (implicitly-described) predicate-extension techniques, ours is more efficient. Notably, for CP-ABE, existing such techniques are very inefficient. To show that our predicate-extension transformation indeed yields interesting improvements on existing ones, we have implemented RWAC. For all algorithms, our transformation incurs only a small constant overhead compared to the CPA-secure variant. In contrast, all other transformations incur a sizable overhead in at least one of the algorithms. In fact, our transformation is at least twice as fast in the decryption algorithm compared to all other transformations.

Chapter 10

Conclusions and outlook

10.1 Conclusions

Finally, we would like to conclude this thesis by reflecting on its goal: achieving all necessary core properties of ABE efficiently to maximize its advantages in practice. As mentioned in the introduction (Chapter 1), we aim to accomplish this by simplifying the design of practical schemes. In this way, it is possible to effectively construct schemes with all necessary properties. As argued in Chapter 4, the pair encodings framework proves to be a powerful tool in this endeavor, as it allows us to transform and compose existing schemes to build larger and more complex systems.

As our overview and systematization in Chapter 2 indicated, however, some challenges remained in the efficient and secure realization of all (necessary) core properties. In particular, previous techniques to simultaneously support non-monotonicity and large-universeness lead to schemes with an inefficient decryption. This may be undesirable in settings with computationally less powerful decryption devices, such as the WLAN use case. Similarly, most expressive large-universe schemes have large ciphertexts or an inefficient encryption. This may be undesirable in IoT settings, in which the encryption devices are resource constrained. In addition, none of the previous CCA-conversion techniques incurred only a small constant overhead in all algorithms for CP-ABE, impacting the efficiency undesirably in general. Our systematization also revealed that, although decentralized ABE provides interesting advantages in practice, very few schemes are decentralized. Moreover, as Chapter 5 shows, many schemes that aim to be decentralized are broken, suggesting that creating such schemes is significantly more difficult than creating single-authority schemes.

This thesis addresses these issues. First, to enable the fair benchmarking of multiple schemes, we have set up the ABE Squared framework, which we also use in the performance analyses of our new schemes. Then, to mitigate the aforementioned efficiency issues, we have proposed two new schemes: GLUE (Construction 7.1) and TinyABE (Construction 8.1). These provide flexible efficiency trade-offs, and can be fine-tuned to take the device constraints into account. Specifically, GLUE can be fine-tuned to have an efficient decryption, and TinyABE can be fine-tuned to have sufficiently small public keys and ciphertexts, as well as a fast encryption. In addition, we have devised new transformations for CCA-security that incur only a small constant overhead in all algorithms for CP-ABE. Furthermore, we have designed a new generic compiler that transforms any pair encoding that satisfies the symbolic property into a selectively secure scheme. This compiler also supports multi-authority extensions, and significantly simplifies the design of decentralized schemes. To illustrate this, we have proposed two new decentralized schemes that improve on the state of the art in the efficiency and core properties.

Importantly, because we have proven all of these constructions secure in the pair encodings framework (using the symbolic property), it is now possible to create schemes with any of the core properties and a flexible efficiency trade-off feature, generically. Specifically, GLUE and TinyABE can be considered "basic" schemes, which satisfy the CP-ABE, expressivity, large-universe and unbounded properties. (Furthermore, they can readily support an attribute-wise key generation, in a similar fashion as RW13 (Section 4.7.4).) By applying the transformations in [AC17b], the key-policy versions of these schemes can be obtained. By applying the transformations in [Att19, AT20, Amb21], any type of non-monotonicity can be achieved. To illustrate this, we have provided a variant of GLUE with OSW-type negations in Construction 7.4. By applying the transformations in [Att19, AT20], a variant supporting the more desirable OSWOT-negations (Sections 2.5.7 and 3.3.2) can be obtained. Furthermore, owing to the structure of the polynomial-based BB hash, all of these schemes can enjoy online/offline extensions that allow the key generation and encryption to be performed efficiently. To illustrate this, we have provided the online/offline version of GLUE (Construction 7.3). In addition, due to the genericness of our CCA-transformations, all these schemes can efficiently support CCA-security. Lastly, an additional desirable feature that the pair encodings framework provides is the ability to support several revocation measures generically [ABS17, YAE$^+$17], including those based on using negations [LSW10].

Regrettably, the only property that we considered and that cannot be achieved generically yet is the (decentralized) multi-authority property. Nevertheless, we have taken the first steps towards realizing this, by extending the pair encodings framework to simplify the efficient design of decentralized schemes. Additionally, the similarities between the selective symbolic property proofs of the PES for AC17 (Construction 4.3) and our decentralized construction from FDH (Construction 4.5) suggest that it may be possible to devise a generic transformation that puts any single-authority scheme in the multi-authority setting. If such a transformation can be devised, we can generically achieve any desirable (core) property.

10.2 Outlook: towards employing ABE in practice

The main reason why we focus on attaining all necessary properties, efficiently and generically, is to maximize the advantages of ABE in any practical employments. In this section, we reflect upon ETSI's use cases and to what extent ABE can achieve the properties required by ETSI. We also discuss two (ongoing) projects to which the author of this thesis has contributed. In the first project, Portunus, ABE is used to enforce access control in a large deployed real-world system. In the second project, PostGuard, predicate encryption is used to manage keys more easily than traditional public-key encryption allows.

Although we have made progress in attaining all necessary properties (also by considering the implementation and benchmarking of ABE schemes), much is still to be done in this endeavor. As the "Directions" and "Future work" sections of each chapter suggest, improvements can still be made for various theoretical and practical metrics. For example, the compilers using the symbolic property for its security rely on q-type assumptions even though static ones are preferred, and the implementations of GLUE and TinyABE may improve if other curves are used. Nevertheless, if one is willing to accept such q-type assumptions (and even the selective and static-security models) for the sake of practicality, much can already be achieved. In fact, our benchmarks already show that the current state-of-the-art libraries for pairing-based arithmetic are so efficient that, even for large policies and attribute sets, all algorithms of the most efficient scheme cost less than a second to execute (on a laptop, using BLS12-381).

10.2.1 ETSI's use cases

As mentioned in the introduction, ETSI has recently put efforts in standardizing ABE for several practical use cases that require the cryptographic enforcement of access control on data [ETS18a]. In this thesis, we have focused mainly on CP-ABE and its application as a cryptographic mechanism to enforce ABAC. The relevant use cases for this type of ABE are the WLAN, IoT[1] and cloud use cases. These use cases share some common goals: they require expressivity (Section 2.3.1), privacy of user-identifying attributes (Section 2.10.3) and revocation measures (Section 2.10.2). Furthermore, they need to support the addition of new attributes, which is why we introduced and discussed the notion of attribute resilience (Section 2.9). Interestingly, although considered useful, ETSI dismisses the support of negations in ABE due to their inefficiency. In this thesis, we have explored existing methods to support negations (Section 2.5.7); considered their effects on the attribute resilience (Section 2.9.2); and designed GLUE, which is a new ABE scheme that allows for a significant decrease in decryption costs in non-monotone schemes (Chapter 7).

In line with the main goal of this thesis, to achieve all necessary properties for any ETSI use case, a suitable "basic" scheme (e.g., AC17-LU, GLUE or TinyABE) can be chosen, and it can be extended with those additional properties that are needed. (Note, however, that these schemes can currently not hide the attributes, and thus

[1] Although ETSI only recommends KP-ABE schemes to be used for the IoT use case, the specified requirements imply the need for CP-ABE as well.

cannot preserve the privacy of the identifying attributes (Direction 2.12). Furthermore, as mentioned, the multi-authority property cannot be generically supported, which is required in, e.g., the WLAN use case.) Our recommendations on how to choose the most suitable basic scheme can be found below.

10.2.2 Cloudflare's Portunus

Even more recently, Cloudflare has presented Portunus, which is a large deployed system that uses CP-ABE to enforce access control in a distributed setting [LVV+23], and to which the author of this thesis has contributed. More specifically, Portunus is a cryptographic storage and access-control system that protects TLS keys. ABE ensures that customers of Cloudflare can choose, based on their location, which Cloudflare servers can relay TLS connections between the customer's websites and visitors of those websites. For example, the GDPR may require that TLS connections are established only with servers located in the European Union. To provide a fast connection between the website and website's visitor, it is paramount that only the closest server that satisfies the customer's demands relays the TLS connection. If access control is enforced in a more traditional way, the connection needs to be relayed through a centralized entity. Therefore, Cloudflare has chosen to use ABE, as it removes the need for such a centralized entity, and directly allows access control to be enforced on the ciphertext. To meet their customers' demands, Cloudflare requires that ABE satisfies all basic properties and additionally supports OT-type negations. To this end, they have chosen to use the TKN20 [TKN20] scheme, which is an extended version of the RW13 [RW13] scheme.

10.2.3 PostGuard

We want to briefly highlight the efforts around the development of the email-encryption service called PostGuard[2] [BSS+22], to which the author of this thesis has also contributed (see, e.g., Chapter 9). PostGuard uses identity-based and attribute-based primitives to ensure confidentiality and receiver authentication in email systems. By using attribute-based encryption, it can ensure those properties in a more fine-grained and user-friendly way than systems based on more traditional public-key encryption, e.g., OpenPGP[3]. In such traditional systems, the sender is responsible for locating and authenticating the recipient's public key. PostGuard moves this responsibility to authenticate away from the sender, and towards the recipient. Currently, the service uses anonymous identity-based encryption [BW06, CGW15] to achieve this, interpreting the policies specified by the sender as an identity string [GA07]. The main drawback of this approach is that, for each unique policy, the decrypting user needs a new secret key to decrypt. This approach therefore requires more interaction between the KGA and user than ABE would. The main reason, however, why ABE is not

[2] PostGuard is available at `https://postguard.nl/`

[3] `https://www.rfc-editor.org/rfc/rfc4880`

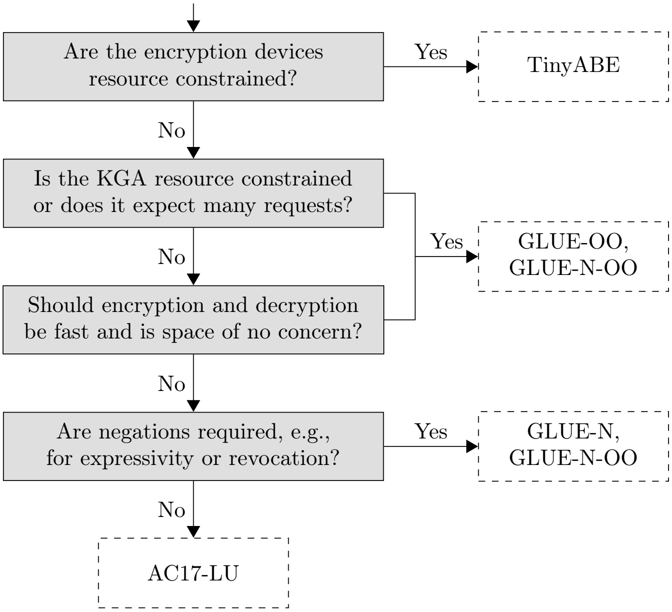

Are the encryption devices resource constrained?
Yes
TinyABE
No
Is the KGA resource constrained or does it expect many requests?
Yes
GLUE-OO, GLUE-N-OO
No
Should encryption and decryption be fast and is space of no concern?
No
Are negations required, e.g., for expressivity or revocation?
Yes
GLUE-N, GLUE-N-OO
No
AC17-LU

Figure 10.1. Decision graph for choosing a suitable scheme based on our recommendations.

used (yet) is that no attribute-hiding ABE schemes exist that support large universes and that are sufficiently expressive (Direction 2.12). In future work, the PostGuard team will conduct more research on devising such a scheme, and on how it can be integrated in the service.

Interestingly, to authenticate the recipient, the service uses digital identity wallets. In a broader context, the European Commission has recently prioritized the development of such wallets for all Europeans [Com]. Digital identity wallets allow individuals to authenticate themselves digitally by disclosing (some of) their identifying attributes, such as their name or date of birth. Because in many of the ABE use cases—e.g., email encryption [BSS+22], IoT and cloud [ETS18a] settings—attributes are used that may be identifying, such wallets are a natural fit for ABE.

10.2.4 Our recommendations for implementing ABE

To help future implementation endeavors, we make four recommendations.

Choosing suitable schemes. Depending on the setting, a different scheme may be the most suitable. In general, we identify three basic schemes satisfying the CP-ABE, expressivity, large-universe and unbounded properties, which could be desirable to employ: AC17-LU[4] (Construction 4.8), GLUE (which also covers RW13) and TinyABE. These schemes provide various trade-offs in the storage and computational efficiency and in the properties they can support. Notably, AC17-LU is not known to have extensions to the non-monotone or online/offline setting. To help practitioners choose a suitable scheme, we have devised a decision graph (Figure 10.1).

Support for multiple categories of computational devices. As already mentioned in Sections 2.7.1 and 7.7, to support various computational devices, multiple instantiations of GLUE and TinyABE (and even AC17-LU) can be employed simultaneously. By applying the parameter reuse transformation of Attrapadung [Att19], the size of the master public key only depends on the maximum size of the master public keys of each instantiation and the number of instantiations.

Property minimization. Because each property incurs some additional cost to the basic scheme, we recommend minimizing the supported properties to those that are strictly necessary. For example, as we mentioned in Section 2.3.1, it may not be feasible to support non-monotonicity for all attribute types. Hence, for such types, it may be better to use a monotone instantiation rather than a non-monotone. Owing to the flexible compositions of Attrapadung [Att19], such functionalities can be simultaneously supported in a single instantiation of a scheme.

Generality is key. To flexibly support multiple instantiations, we recommend implementing GLUE and TinyABE as generically as possible, such that the configurable nature of these schemes carries over to their implementations. Furthermore, on a lower level, we recommend implementing the underlying arithmetic as generically as possible, e.g., such that any pairing-friendly curve can be supported. (This does not mean, however, that optimized arithmetic such as fixed-base exponentations should not be used. On the contrary, we recommend using it.) In this way, it is easier to upgrade the implementation to a higher security level, should the used curve be compromised, or to switch to another curve, should a more efficient one be implemented.

[4] We consider AC17-LU and not Wat11-IV, because it completely outperforms Wat11-IV.

Bibliography

[ABGW17] 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, *CCS*, pages 647–664. ACM, 2017.

[ABS17] M. Ambrona, G. Barthe, and B. Schmidt. Generic transformations of predicate encodings: Constructions and applications. In J. Katz and H. Shacham, editors, *CRYPTO*, volume 10401 of *LNCS*, pages 36–66. Springer, 2017.

[ABV$^+$12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 280–297. Springer, 2012.

[AC16] S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC*, volume 9563 of *LNCS*, pages 259–288. Springer, 2016.

[AC17a] S. Agrawal and M. Chase. FAME: fast attribute-based message encryption. In B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *CCS*, pages 665–682. ACM, 2017.

[AC17b] S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT*, volume 10210 of *LNCS*, pages 627–656. Springer, 2017.

[AC17c] S. Agrawal and M. Chase. Simplifying design and analysis of complex predicate encryption schemes. Cryptology ePrint Archive, Report 2017/233, 2017.

[ACGU20] M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. In S. Moriai and H. Wang, editors, *ASIACRYPT*, volume 12493 of *LNCS*, pages 467–497. Springer, 2020.

[ACIK10] M. Abe, Y. Cui, H. Imai, and E. Kiltz. Efficient hybrid encryption from ID-based encryption. *Des. Codes Cryptogr.*, 54(3):205–240, 2010.

[AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In D. H. Lee and X. Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 21–40. Springer, 2011.

[AGH13] 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, *CCS*, pages 399–410. ACM, 2013.

[AGH15] 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, *CCS*, pages 1370–1381. ACM, 2015.

[AGM$^+$] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[AGM$^+$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. *J. Cryptogr. Eng.*, 3(2):111–128, 2013.

[AGOT14] 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*, volume 8616 of *LNCS*, pages 241–260. Springer, 2014.

[AHM$^+$16] N. Attrapadung, G. Hanaoka, T. Matsumoto, T. Teruya, and S. Yamada. Attribute based encryption with direct efficiency tradeoff. In M. Manulis, A.-R. Sadeghi, and S. A. Schneider, editors, *ACNS*, volume 9696 of *LNCS*, pages 249–266. Springer, 2016.

[AHO16a] M. Abe, F. Hoshino, and M. Ohkubo. Design in type-I, run in type-III: Fast and scalable bilinear-type conversion using integer programming. In M. Robshaw and J. Katz, editors, *CRYPTO*, volume 9816 of *LNCS*, pages 387–415. Springer, 2016.

[AHO$^+$16b] N. Attrapadung, G. Hanaoka, K. Ogawa, G. Ohtake, H. Watanabe, and S. Yamada. Attribute-based encryption for range attributes. In V. Zikas and R. De Prisco, editors, *SCN*, volume 9841 of *LNCS*, pages 42–61. Springer, 2016.

[AHY15] N. Attrapadung, G. Hanaoka, and S. Yamada. Conversions among several classes of predicate encryption and applications to ABE with various compactness tradeoffs. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT*, volume 9452 of *LNCS*, pages 575–601. Springer, 2015.

[AI09a] N. Attrapadung and H. Imai. Attribute-based encryption supporting direct/indirect revocation modes. In M. G. Parker, editor, *IMACC*, volume 5921 of *LNCS*, pages 278–300. Springer, 2009.

[AI09b] N. Attrapadung and H. Imai. Conjunctive broadcast and attribute-based encryption. In H. Shacham and B. Waters, editors, *Pairing*, volume 5671 of *LNCS*, pages 248–265. Springer, 2009.

[AL10] N. Attrapadung and B. Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In P. Q. Nguyen and D. Pointcheval, editors, *PKC*, volume 6056 of *LNCS*, pages 384–402. Springer, 2010.

[ALdP11] 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*, volume 6571 of *LNCS*, pages 90–108. Springer, 2011.

[Amb21] M. Ambrona. Generic negation of pair encodings. In J. A. Garay, editor, *PKC*, volume 12711 of *LNCS*, pages 120–146. Springer, 2021.

[APG$^+$11] 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*, pages 75–86. ACM, 2011.

[Ara17] D. Aranha. Pairings are not dead, just resting, 2017.

[ASLT13] J. Luis Fernández Alemán, I. Carrión Señor, P. Á. O. Lozoya, and A. Toval. Security and privacy in electronic health records: A systematic literature review. *J. Biomed. Informatics*, 46(3):541–562, 2013.

[AT20] N. Attrapadung and J. Tomida. Unbounded dynamic predicate compositions in ABE from standard assumptions. In *ASIACRYPT*, pages 405–436. Springer, 2020.

[Att14a] 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*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014.

[Att14b] N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. 2014.

[Att16] N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT*, volume 10032 of *LNCS*, pages 591–623. Springer, 2016.

[Att19] N. Attrapadung. Unbounded dynamic predicate compositions in attribute-based encryption. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT*, volume 11476 of *LNCS*, pages 34–67. Springer, 2019.

[AY15] N. Attrapadung and S. Yamada. Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In K. Nyberg, editor, *CT-RSA*, volume 9048 of *LNCS*, pages 87–105. Springer, 2015.

[Bar20] E. Barker. Recommendation for key management–part 1: General (revision 5). 2020.

[BB04] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[BBG05] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BBJ$^+$23] L. Botros, M. Brandon, B. Jacobs, D. Ostkamp, H. Schraffenberger, and M. Venema. Postguard: Towards easy and secure email communication. In *CHI EA*. ACM, 2023.

[BBS98] M. Blaze, G. Bleumer, and M Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, *EUROCRYPT*, volume 1403 of *LNCS*, pages 127–144. Springer, 1998.

[BCCT12] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In S. Goldwasser, editor, *ITCS*, pages 326–349. ACM, 2012.

[BCHK07] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BD19] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *J. Cryptol.*, 32(4):1298–1336, 2019.

[Bei96] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. Phd thesis, Ben Gurion University, 1996.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[BFM88] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 103–112. ACM, 1988.

[BK05] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 87–103. Springer, 2005.

[BKLS02] P. S. L. M. Barreto, H. Yong Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 354–368. Springer, 2002.

[BL16] J. Blömer and G. Liske. Construction of fully cca-secure predicate encryptions from pair encoding schemes. In K. Sako, editor, *CT-RSA*, volume 9610 of *LNCS*, pages 431–447. Springer, 2016.

[BLS02] P S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi, and G. Persiano, editors, *SCN*, volume 2576 of *LNCS*, pages 257–267. Springer, 2002.

[BN05] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *SAC*, volume 3897 of *LNCS*, pages 319–331. Springer, 2005.

[Bon98] D. Boneh. The decision Diffie-Hellman problem. In J. Buhler, editor, *ANTS-III*, volume 1423 of *LNCS*, pages 48–63. Springer, 1998.

[Bow] S. Bowe. BLS12-381: New zk-SNARK elliptic curve construction. https://blog.z.cash/new-snark-curve/.

[Boy08] X. Boyen. The uber-assumption family – a unified complexity framework for bilinear groups. In S. D. Galbraith and K. G. Paterson, editors, *Pairing*, volume 5209 of *LNCS*, pages 39–56. Springer, 2008.

[Boy13] X. Boyen. Attribute-based functional encryption on lattices. In A. Sahai, editor, *TCC*, volume 7785 of *LNCS*, pages 122–142. Springer, 2013.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *CCS*, pages 62–73. ACM, 1993.

[BSS$^+$22] M. Brandon, H. K. Schraffenberger, W. Sluis-Thiescheffer, T. van der Geest, D. Ostkamp, and B. Jacobs. Design principles for actual security. In *NordiCHI '22: Adjunct Proceedings of the Nordic Human-Computer Interaction Conference*, pages 41:1–41:6. ACM, 2022.

[BSW07] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *S&P*, pages 321–334. IEEE, 2007.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.

[BW06] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.

[BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC*, volume 4392 of *LNCS*, pages 535–554. Springer, 2007.

[CC09] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *CCS*, pages 121–130. ACM, 2009.

[CCL$^+$13] C. Chen, J. Chen, H. W. Lim, Z. Zhang, D. Feng, S. Ling, and H. Wang. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In *CT-RSA*, volume 7779 of *LNCS*, pages 50–67. Springer, 2013.

[CDLQ16] H. Cui, R. H. Deng, Y. Li, and B. Qin. Server-aided revocable attribute-based encryption. In I. G. Askoxylakis, S. Ioannidis, S. K. Katsikas, and C. A. Meadows, editors, *ESORICS*, volume 9879 of *LNCS*, pages 570–587. Springer, 2016.

[CDM15] P. Chaudhari, M. Lal Das, and A. Mathuria. On anonymous attribute based encryption. In S. Jajodia and C. Mazumdar, editors, *ICISS*, volume 9478 of *LNCS*, pages 378–392. Springer, 2015.

[CDS20] R. Clarisse, S. Duquesne, and O. Sanders. Curves with fast computations in the first pairing group. In S. Krenn, H. Shulman, and S. Vaudenay, editors, *CANS*, volume 12579 of *LNCS*, pages 280–298. Springer, 2020.

[CG17] J. Chen and J. Gong. ABE with tag made easy - concise framework and new instantiations in prime-order groups. In T. Takagi and T. Peyrin, editors, *ASIACRYPT*, volume 10625 of *LNCS*, pages 35–65. Springer, 2017.

[CGH04] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CGKW18] J. Chen, J. Gong, L. Kowalczyk, and H. Wee. Unbounded ABE via bilinear entropy expansion, revisited. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT*, volume 10820 of *LNCS*, pages 503–534. Springer, 2018.

[CGW15] 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*, volume 9057 of *LNCS*, pages 595–624. Springer, 2015.

[Cha07] M. Chase. Multi-authority attribute-based encryption. In S. P. Vadhan, editor, *TCC*, volume 4392 of *LNCS*, pages 515–534. Springer, 2007.

[Che06] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *LNCS*, pages 1–11. Springer, 2006.

[CHK04] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[Cho16] S. S. M. Chow. A framework of multi-authority attribute-based encryption with outsourcing and revocation. In X. Sean Wang, L. Bauer, and F. Kerschbaum, editors, *SACMAT*, pages 215–226. ACM, 2016.

[CKMS16] S. Chatterjee, N. Koblitz, A. Menezes, and P. Sarkar. Another look at tightness II: practical issues in cryptography. In R. C.-W. Phan and M. Yung, editors, *Mycrypt*, volume 10311 of *LNCS*, pages 21–55. Springer, 2016.

[CM14] J. Chen and H. Ma. Efficient decentralized attribute-based access control for cloud storage with user revocation. In *ICC*, pages 3782–3787. IEEE, 2014.

[CN07] L. Cheung and C. C. Newport. Provably secure ciphertext policy ABE. In P. Ning, S. De Capitani di Vimercati, and P. F. Syverson, editors, *CCS*, pages 456–465. ACM, 2007.

[Com] European Commission. European digital identity. https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/european-digital-identity_en.

[CS03] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[CS10] C. Costello and D. Stebila. Fixed argument pairings. In M. Abdalla and P. S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *LNCS*, pages 92–108. Springer, 2010.

[CW13] J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8043 of *LNCS*, pages 435–460. Springer, 2013.

[CW14a] J. Chen and H. Wee. Dual system groups and its applications — compact HIBE and more. Cryptology ePrint Archive, Report 2014/265, 2014.

[CW14b] J. Chen and H. Wee. Semi-adaptive attribute-based encryption and improved delegation for boolean formula. In M. Abdalla and R. De Prisco, editors, *SCN*, volume 8642 of *LNCS*, pages 277–297. Springer, 2014.

[CZF11] C. Chen, Z. Zhang, and D. Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In X. Boyen and X. Chen, editors, *ProvSec*, volume 6980 of *LNCS*, pages 84–101. Springer, 2011.

[Dam89] I. Damgård. A design principle for hash functions. In G. Brassard, editor, *CRYPTO*, volume 435 of *LNCS*, pages 416–427. Springer, 1989.

[Del07] C. Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 200–215. Springer, 2007.

[Den02] A. W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Y. Zheng, editor, *ASIACRYPT*, volume 2501 of *LNCS*, pages 100–109. Springer, 2002.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[DKW23] P. Datta, I. Komargodski, and B. Waters. Decentralized multi-authority ABE for NC^1 from computational-BDH. *J. Cryptol.*, 36(2):6, 2023.

[dLea10] R. de Lemos et al. Software engineering for self-adaptive systems: A second research roadmap. In R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, editors, *Software Engineering for Self-Adaptive Systems II*, volume 7475 of *LNCS*, pages 1–32. Springer, 2010.

[dlPVA] A. de la Piedra, M. Venema, and G. Alpár. ABE squared. https://github.com/abecryptools/abe_squared.

[dlPVA22] A. de la Piedra, M. Venema, and G. Alpár. ABE squared: Accurately benchmarking efficiency of attribute-based encryption. *TCHES*, 2022(2):192–239, 2022.

[EHK$^+$13] A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An algebraic framework for Diffie-Hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8043 of *LNCS*, pages 129–147. Springer, 2013.

[EMN$^+$09] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *ISPEC*, pages 13–23. Springer, 2009.

[ETS18a] ETSI. ETSI TS 103 458 (V1.1.1). Technical specification, European Telecommunications Standards Institute (ETSI), 2018.

[ETS18b] ETSI. ETSI TS 103 532 (V1.1.1). Technical specification, European Telecommunications Standards Institute (ETSI), 2018.

[ETS20] ETSI. Even more advanced cryptography – industry applications and use cases for advanced cryptography. Technical report, European Telecommunications Standards Institute (ETSI), 2020.

[FA17] H. Fujii and D. F. Aranha. Curve25519 for the Cortex-M4 and beyond. In T. Lange and O. Dunkelman, editors, *LATINCRYPT*, volume 11368 of *LNCS*, pages 109–127. Springer, 2017.

[FEN] The FENTEC project. `https://github.com/fentec-project`.

[FO99] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 537–554. Springer, 1999.

[Fre10] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.

[GA07] M. Green and G. Ateniese. Identity-based proxy re-encryption. In Jonathan Katz and Moti Yung, editors, *ACNS*, volume 4521 of *LNCS*, pages 288–306. Springer, 2007.

[Gal14] S. D. Galbraith. New discrete logarithm records, and the death of type 1 pairings. `https://ellipticnews.wordpress.com/`, 2014.

[Gam84] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *CRYPTO*, volume 196 of *LNCS*, pages 10–18. Springer, 1984.

[Gen06] C. Gentry. Practical identity-based encryption without random oracles. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *LNCS*, pages 445–464. Springer, 2006.

[GF16] L. Ghammam and E. Fouotsa. Adequate elliptic curves for computing the product of n pairings. In S. Duquesne and S. Petkova-Nikova, editors, *WAIFI*, volume 10064 of *LNCS*, pages 36–53, 2016.

[GGH$^+$13] S. Garg, C. Gentry, S. Halevi, A. Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8043 of *LNCS*, pages 479–499. Springer, 2013.

[GHM$^+$17] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, pages 51–68. ACM, 2017.

[GHW11] M. Green, S. Hohenberger, and B. Waters. Outsourcing the decryption of ABE ciphertexts. In *USENIX Security Symposium*. USENIX Association, 2011.

[GKSW10] S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *CCS*, pages 121–130. ACM, 2010.

[GMT20] A. Guillevic, S. Masson, and E. Thomé. Cocks-Pinch curves of embedding degrees five to eight and optimal ate pairing computation. *Des. Codes Cryptogr.*, 88(6):1047–1081, 2020.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008.

[GPSW06a] 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. De Capitani di Vimercati, editors, *CCS*. ACM, 2006.

[GPSW06b] 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.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In C. Dwork, editor, *STOC*, pages 197–206. ACM, 2008.

[GS06] R. Granger and N.P. Smart. On computing products of pairings. Cryptology ePrint Archive, Report 2006/172, 2006.

[GS19] A. Guillevic and S. Singh. On the alpha value of polynomials in the tower number field sieve algorithm. Cryptology ePrint Archive, Report 2019/885, 2019.

[Gui13] 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*, volume 7954 of *LNCS*, pages 357–372. Springer, 2013.

[Gui20a] A. Guillevic. Pairing-friendly curves. `https://members.loria.fr/AGuillevic/pairing-friendly-curves/`, 2020.

[Gui20b] A. Guillevic. A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC*, volume 12111 of *LNCS*, pages 535–564. Springer, 2020.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *STOC*, pages 545–554. ACM, 2013.

[GVW15] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. *J. ACM*, 62(6):45:1–45:33, 2015.

[GZC$^+$12] A. Ge, R. Zhang, C. Chen, C. Ma, and Z. Zhang. Threshold ciphertext policy attribute-based encryption with constant size ciphertexts. In W. Susilo, Y. Mu, and J. Seberry, editors, *ACISP*, volume 7372 of *LNCS*, pages 336–349. Springer, 2012.

[GZZ$^+$13] A. Ge, J. Zhang, R. Zhang, C. Ma, and Z. Zhang. Security analysis of a privacy-preserving decentralized key-policy attribute-based encryption scheme. *IEEE Trans. Parallel Distributed Syst.*, 24(11):2319–2321, 2013.

[Ham11] M. Hamburg. Spatial encryption. Cryptology ePrint Archive, Report 2011/389, 2011.

[HFK$^+$19] C. T. Hu, D. F. Ferraiolo, D. R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. Guide to attribute based access control (ABAC) definition and considerations, 2019.

[HHK17] D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Y. Kalai and L. Reyzin, editors, *TCC*, volume 10677 of *LNCS*, pages 341–371. Springer, 2017.

[HLR10] J. Herranz, F. Laguillaumie, and C. Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In P. Q. Nguyen and D. Pointcheval, editors, *PKC*, volume 6056 of *LNCS*, pages 19–34. Springer, 2010.

[HMCB11] J. Hiller, M. S. McMullen, W. M. Chumney, and D. L. Baumer. Privacy and security in the implementation of health information technology (electronic health records): U.S. and EU compared. *Boston University Journal of Science & Technology Law*, 17(1):1–39, 2011.

[HRS16] A. Hülsing, J. Rijneveld, and P. Schwabe. Armed SPHINCS - computing a 41 KB signature in 16 KB of RAM. In *PKC*, pages 446–470. Springer, 2016.

[HSM$^+$14] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. Ho Au. PPDCP-ABE: privacy-preserving decentralized ciphertext-policy attribute-based encryption. In M. Kutylowski and J. Vaidya, editors, *ESORICS*, volume 8713 of *LNCS*, pages 73–90. Springer, 2014.

[HSM$^+$15] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. Ho Allen Au. Improving privacy and security in decentralized ciphertext-policy attribute-based encryption. *IEEE Trans. Inf. Forensics Secur.*, 10(3):665–678, 2015.

[HSMY12] J. Han, W. Susilo, Y. Mu, and J. Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Trans. Parallel Distributed Syst.*, 23(11):2150–2162, 2012.

[HSN08] K. Häyrinen, K. Saranto, and P. Nykänen. Definition, structure, content, use and impacts of electronic health records: A review of the research literature. *Int. J. Medical Informatics*, 77(5):291–304, 2008.

[HW13] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In K. Kurosawa and G. Hanaoka, editors, *PKC*, volume 7778 of *LNCS*, pages 162–179. Springer, 2013.

[HW14] S. Hohenberger and B. Waters. Online/offline attribute-based encryption. In Hugo Krawczyk, editor, *PKC*, volume 8383 of *LNCS*, pages 293–310. Springer, 2014.

[HXL15] J. Hong, K. Xue, and W. Li. Comments on "DAC-MACS: Effective data access control for multiauthority cloud storage systems" / security analysis of attribute revocation in multiauthority data access control for cloud storage systems. *IEEE Trans. Inf. Forensics Secur.*, 10(6):1315–1317, 2015.

[IPN$^+$09] L. Ibraimi, M. Petkovic, S. Nikova, P. H. Hartel, and W. Jonker. Mediated ciphertext-policy attribute-based encryption and its application. In H. Youl Youm and M. Yung, editors, *WISA*, volume 5932 of *LNCS*, pages 309–323. Springer, 2009.

[JLWW13] T. Jung, X. Y. Li, Z. Wan, and M. Wan. Privacy preserving cloud data access with multi-authorities. In *INFOCOM*, pages 2625–2633. IEEE, 2013.

[JLWW15] T. Jung, X. Y. Li, Z. Wan, and M. Wan. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. *IEEE Trans. Inf. Forensics Secur.*, 10(1):190–199, 2015.

[JLWW16] T. Jung, X. Y. Li, Z. Wan, and M. Wan. Rebuttal to "Comments on 'Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption"'. *IEEE Trans. Inf. Forensics Secur.*, 11(4):868, 2016.

[KB16] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *CRYPTO*, volume 9814 of *LNCS*, pages 543–571. Springer, 2016.

[KG06] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In L. M. Batten and R. Safavi-Naini, editors, *ACISP*, volume 4058 of *LNCS*, pages 336–347. Springer, 2006.

[KHA$^+$19] S. Kumar, Y. Hu, M. P. Andersen, R. Ada Popa, and D. E. Culler. JEDI: many-to-many end-to-end encryption and key delegation for IoT. In *28th USENIX Security Symposium*, pages 1519–1536. USENIX Association, 2019.

[KL10] S. Kamara and K. E. Lauter. Cryptographic cloud storage. In R. Sion, R. Curtmola, S. Dietrich, A. Kiayias, J. M. Miret, K. Sako, and F. Sebé, editors, *RLCPS*, volume 6054 of *LNCS*, pages 136–149. Springer, 2010.

[KL15] L. Kowalczyk and A. Lewko. Bilinear entropy expansion from the decisional linear assumption. In R. Gennaro and M. Robshaw, editors, *CRYPTO*, volume 9216 of *LNCS*, pages 524–541. Springer, 2015.

[KM15] N. Koblitz and A. J. Menezes. The random oracle model: a twenty-year retrospective. *Des. Codes Cryptogr.*, 77(2-3):587–610, 2015.

[KSS08] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In S. D. Galbraith and K. G. Paterson, editors, *Pairing*, volume 5209 of *LNCS*, pages 126–135. Springer, 2008.

[KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008.

[KV08] E. Kiltz and Y. Vahlis. CCA2 secure IBE: standard model efficiency through authenticated symmetric encryption. In Tal Malkin, editor, *CT-RSA*, volume 4964 of *LNCS*, pages 221–238. Springer, 2008.

[KW19a] V. Koppula and B. Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In A. Boldyreva and D. Micciancio, editors, *CRYPTO*, volume 11693 of *LNCS*, pages 671–700. Springer, 2019.

[KW19b] L. Kowalczyk and H. Wee. Compact adaptively secure ABE for NC1 from k-Lin. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT*, volume 11476 of *LNCS*, pages 3–33. Springer, 2019.

[LCH$^+$11] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In V. Atluri and C. Díaz, editors, *ESORICS*, volume 6879 of *LNCS*, pages 278–297. Springer, 2011.

[LCL$^+$13] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou. Fine-grained access control system based on outsourced attribute-based encryption. In J. Crampton, S. Jajodia, and K. Mayes, editors, *ESORICS*, volume 8134 of *LNCS*, pages 592–609. Springer, 2013.

[LCLS08] H. Lin, Z. Cao, X. Liang, and J. Shao. Secure threshold multi authority attribute based encryption without a central authority. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *INDOCRYPT*, volume 5365 of *LNCS*, pages 426–436. Springer, 2008.

[LCW13] Z. Liu, Z. Cao, and D. S. Wong. Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on eBay. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *CCS*, pages 475–486. ACM, 2013.

[Lew12] 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*, volume 7237 of *LNCS*, pages 318–335. Springer, 2012.

[LHC$^+$11] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie. Multi-authority ciphertext-policy attribute-based encryption with accountability. In B. S. N. Cheung, L. Chi Kwong Hui, R. S. Sandhu, and D. S. Wong, editors, *ASIACCS*, pages 386–390. ACM, 2011.

[LL20a] H. Lin and J. Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT*, volume 12107 of *LNCS*, pages 247–277. Springer, 2020.

[LL20b] H. Lin and J. Luo. Succinct and adaptively secure ABE for ABP from k-Lin. In S. Moriai and H. Wang, editors, *ASIACRYPT*, volume 12493 of *LNCS*, pages 437–466. Springer, 2020.

[LN09] G. Leurent and P. Q. Nguyen. How risky is the random-oracle model? In S. Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 445–464. Springer, 2009.

[LOS$^+$10] 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*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.

[LRZW09] J. Li, K. Ren, B. Zhu, and Z. Wan. Privacy-aware attribute-based encryption with user accountability. In P. Samarati, M. Yung, F. Martinelli, and C. Agostino Ardagna, editors, *ISC*, volume 5735 of *LNCS*, pages 347–362. Springer, 2009.

[LSW10] A. Lewko, A. Sahai, and B. Waters. Revocation systems with very small private keys. In *IEEE S & P*, pages 273–285, 2010.

[LT18] J. Lai and Q. Tang. Making any attribute-based encryption accountable, efficiently. In J. López, J. Zhou, and M. Soriano, editors, *ESORICS*, volume 11099 of *LNCS*, pages 527–547. Springer, 2018.

[LVV$^+$23] W. Ladd, T. Verma, M. Venema, A. Faz-Hérnandez, B. McMillion, A. Wildani, and N. Sullivan. Portunus: Re-imagining access control in distributed systems. In *USENIX ATC*, pages 35–52, 2023.

[LW10a] A. Lewko and B. Waters. Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351, 2010.

[LW10b] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In D. Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.

[LW11a] A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588. Springer, 2011.

[LW11b] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *LNCS*, pages 547–567. Springer, 2011.

[LW12] A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198. Springer, 2012.

[LW14] A. Lewko and B. Waters. Why proving HIBE systems secure is difficult. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 58–76. Springer, 2014.

[LW15] Z. Liu and D. S. Wong. Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe. In T. Malkin, V. Kolesnikov, A. Lewko, and M. Polychronakis, editors, *ACNS*, volume 9092 of *LNCS*, pages 127–146. Springer, 2015.

[LXXH16] W. Li, K. Xue, Y. Xue, and J. Hong. TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage. *IEEE Trans. Parallel Distributed Syst.*, 27(5):1484–1496, 2016.

[Lyn13] B. Lynn. The Stanford pairing based crypto library. `http://crypto.stanford.edu/pbc`, 2013.

[LYZL18] J. K. Liu, T. H. Yuen, P Zhang, and K. Liang. Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In B. Preneel and F. Vercauteren, editors, *ACNS*, volume 10892 of *LNCS*, pages 516–534. Springer, 2018.

[MGZ19] C. Ma, A. Ge, and J. Zhang. Fully secure decentralized ciphertext-policy attribute-based encryption in standard model. In F. Guo, X. Huang, and M. Yung, editors, *Inscrypt*, volume 11449 of *LNCS*, pages 427–447. Springer, 2019.

[Mil04] V. S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptol.*, 17(4):235–261, 2004.

[MJ18] Y. Michalevsky and M. Joye. Decentralized policy-hiding ABE with receiver privacy. In J. López, J. Zhou, and M. Soriano, editors, *ESORICS*, volume 11099 of *LNCS*, pages 548–567. Springer, 2018.

[MKE08] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *LNCS*, pages 20–36. Springer, 2008.

[MMM05] S. Malek, M. Mikic-Rakic, and N. Medvidovic. A decentralized redeployment algorithm for improving the availability of distributed systems. In A. Dearle and S. Eisenbach, editors, *Component Deployment*, volume 3798 of *LNCS*, pages 99–114. Springer, 2005.

[Möl01] B. Möller. Algorithms for multi-exponentiation. In S. Vaudenay and A. M. Youssef, editors, *SAC*, volume 2259 of *LNCS*, pages 165–180. Springer, 2001.

[MST17] Q. M. Malluhi, A. Shikfa, and V. C. Trinh. A ciphertext-policy attribute-based encryption scheme with optimized ciphertext size and fast decryption. In R. Karri, O. Sinanoglu, A.-R. Sadeghi, and X. Yi, editors, *AsiaCCS*, pages 230–240. ACM, 2017.

[MTP$^+$21] M. La Manna, L. Treccozzi, P. Perazzo, S. Saponara, and G. Dini. Performance evaluation of attribute-based encryption in automotive embedded platform for secure software over-the-air update. *Sensors*, 21(2):515, 2021.

[MZY16] H. Ma, R. Zhang, and W. Yuan. Comments on "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption". *IEEE Trans. Inf. Forensics Secur.*, 11(4):866–867, 2016.

[NDCW15] J. Ning, X. Dong, Z. Cao, and L. Wei. Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS*, volume 9327 of *LNCS*, pages 270–289. Springer, 2015.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In H. Ortiz, editor, *STOC*, pages 427–437. ACM, 1990.

[NYO08] T. Nishide, K. Yoneyama, and K. Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, editors, *ACNS*, volume 5037 of *LNCS*, pages 111–129, 2008.

[OSW07] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *CCS*, pages 195–203. ACM, 2007.

[OT10] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010.

[OT12] T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT*, pages 349–366. Springer, 2012.

[OT13] T. Okamoto and K. Takashima. Decentralized attribute-based signatures. In K. Kurosawa and G. Hanaoka, editors, *PKC*, volume 7778 of *LNCS*, pages 125–142. Springer, 2013.

[PO17] H. S. G. Pussewalage and V. A. Oleshchuk. A distributed multi-authority attribute based encryption scheme for secure sharing of personal health records. In E. Bertino, R. S. Sandhu, and E. R. Weippl, editors, *SACMAT*, pages 255–262. ACM, 2017.

[PP03] K. G. Paterson and G. Price. A comparison between traditional public key infrastructures and identity-based cryptography. *Inf. Secur. Tech. Rep.*, 8(3):57–72, 2003.

[PRMV21] P. Perazzo, F. Righetti, M. La Manna, and C. Vallati. Performance evaluation of attribute-based encryption on constrained IoT devices. *Comput. Commun.*, 170:151–163, 2021.

[PST20] C. Paquin, D. Stebila, and G. Tamvada. Benchmarking post-quantum cryptography in TLS. In *PQCrypto*, pages 72–91. Springer, 2020.

[PTMW10] M. Pirretti, P. Traynor, P. D. McDaniel, and B. Waters. Secure attribute-based systems. *J. Comput. Secur.*, 18(5):799–837, 2010.

[QLZ13] H. Qian, J. Li, and Y. Zhang. Privacy-preserving decentralized ciphertext-policy attribute-based encryption with fully hidden access structure. In S. Qing, J. Zhou, and D. Liu, editors, *ICICS*, volume 8233 of *LNCS*, pages 363–372. Springer, 2013.

[RCS12] S. C. Ramanna, S. Chatterjee, and P. Sarkar. Variants of Waters' dual system primitives using asymmetric pairings - (extended abstract). In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 298–315. Springer, 2012.

[Rog02] P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *CCS*, pages 98–107. ACM, 2002.

[RS91] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO*, volume 576 of *LNCS*, pages 433–444. Springer, 1991.

[RS04] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In B. K. Roy and W. Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 371–388. Springer, 2004.

[RW13] 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, *CCS*, pages 463–474. ACM, 2013.

[RW15] Y. Rouselakis and B. Waters. Efficient statically-secure large-universe multi-authority attribute-based encryption. In R. Böhme and T. Okamoto, editors, *FC*, volume 8975 of *LNCS*, pages 315–332. Springer, 2015.

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[Sco03] M. Scott. MIRACL cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library. `https://github.com/miracl/MIRACL`, 2003.

[Sco11] M. Scott. On the efficient implementation of pairing-based protocols. In L. Chen, editor, *IMACC*, volume 7089 of *LNCS*, pages 296–308. Springer, 2011.

[Sco20] M. Scott. On the deployment of curve based cryptography for the internet of things. Cryptology ePrint Archive, Report 2020/514, 2020.

[Sha79] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.

[Sho98] V. Shoup. Why chosen ciphertext security matters. IBM Research Report RZ 3076, November 1998.

[SKSW20] Y. Sakemi, T. Kobayashi, T. Saito, and R. S. Wahby. Pairing-Friendly Curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-09, Internet Engineering Task Force, November 2020. Work in Progress.

[SR13] A. H. Sánchez and F. Rodríguez-Henríquez. NEON implementation of an attribute-based encryption scheme. In M. J. Jacobson Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS*, volume 7954 of *LNCS*, pages 322–338. Springer, 2013.

[SRGS12] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu. Policy-sealed data: A new abstraction for building trusted cloud services. In *USENIX Security Symposium*, pages 175–188. USENIX Association, 2012.

[SS94] R. S. Sandhu and P. Samarati. Access control: principles and practice. *IEEE Commun. Mag.*, 32(9):40–48, 1994.

[SSW12] A. Sahai, H. Seyalioglu, and B. Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO*, pages 199–217. Springer, 2012.

[SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[Tak14] K. Takashima. Expressive attribute-based encryption with constant-size ciphertexts from the decisional linear assumption. In *SCN*, pages 298–317. Springer, 2014.

[TKN20] J. Tomida, Y. Kawahara, and R. Nishimaki. Fast, compact, and expressive attribute-based encryption. In A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, editors, *PKC*, volume 12110 of *LNCS*, pages 3–33. Springer, 2020.

[TKN21] J. Tomida, Y. Kawahara, and R. Nishimaki. Fast, compact, and expressive attribute-based encryption. *Des. Codes Cryptogr.*, 89(11):2577–2626, 2021.

[VA20] M. Venema and G. Alpár. A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. Cryptology ePrint Archive, Report 2020/460, 2020.

[VA21] M. Venema and G. Alpár. A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. In K. G. Paterson, editor, *CT-RSA*, volume 12704 of *LNCS*, pages 100–125. Springer, 2021.

[VA22a] M. Venema and G. Alpár. TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. In L. Batina and J. Daemen, editors, *AFRICACRYPT*, volume 13503 of *LNCS*, pages 103–129. Springer, 2022.

[VA22b] M. Venema and G. Alpár. GLUE: Generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. Cryptology ePrint Archive, Paper 2022/613, 2022.

[VA22c] M. Venema and G. Alpár. TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. Cryptology ePrint Archive, Paper 2022/569, 2022.

[VA23] M. Venema and G. Alpár. GLUE: generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. In A. Boldyreva and V. Kolesnikov, editors, *PKC*, volume 13940 of *LNCS*, pages 652–682. Springer, 2023.

[VAH21] M. Venema, G. Alpár, and J.-H. Hoepman. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. Cryptology ePrint Archive, Report 2021/1172, 2021.

[VAH23] M. Venema, G. Alpár, and J.-H. Hoepman. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. *Des. Codes Cryptogr.*, 91(1):165–220, 2023.

[VB22] M. Venema and L. Botros. Efficient and generic transformations for chosen-ciphertext secure predicate encryption. Cryptology ePrint Archive, Paper 2022/1436, 2022.

[Ven23a] M. Venema. A practical compiler for attribute-based encryption: New decentralized constructions and more. In M. Rosulek, editor, *CT-RSA*, volume 13871 of *LNCS*, pages 132–159. Springer, 2023.

[Ven23b] M. Venema. A practical compiler for attribute-based encryption: New decentralized constructions and more. Cryptology ePrint Archive, Paper 2023/143, 2023.

[Wat08] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008.

[Wat09] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In S. Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.

[Wat11] 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*, volume 6571 of *LNCS*, pages 53–70. Springer, 2011.

[WB19] R. S. Wahby and D. Boneh. Fast and simple constant-time hashing to the BLS12-381 elliptic curve. *TCHES*, 2019(4):154–179, 2019.

[Wee14] H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014.

[Wee17] H. Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Y. Kalai and L. Reyzin, editors, *TCC*, volume 10677 of *LNCS*, pages 206–233. Springer, 2017.

[WJB17] X. Wu, R. Jiang, and B. Bhargava. On the security of data access control for multiauthority cloud storage systems. *IEEE Trans. Serv. Comput.*, 10(2):258–272, 2017.

[WZC15] M. Wang, Z. Zhang, and C. Chen. Security analysis of a privacy-preserving decentralized ciphertext-policy attribute-based encryption scheme. *Concurr. Comput. Pract. Exp.*, 28(4):1237–1245, 2015.

[WZSI14] X. Wang, J. Zhang, E. M. Schooler, and M. Ion. Performance evaluation of attribute-based encryption: Toward data privacy in the IoT. In *ICC*, pages 725–730. IEEE, 2014.

[XFZ$^+$14] F. Xhafa, J. Feng, Y. Zhang, X. Chen, and J. Li. Privacy-aware attribute-based PHR sharing with user accountability in cloud computing. *J. Supercomput.*, 71(5):1607–1619, 2014.

[YAE$^+$17] K. Yamada, N. Attrapadung, K. Emura, G. Hanaoka, and K. Tanaka. Generic constructions for fully secure revocable attribute-based encryption. In S. N. Foley, D. Gollmann, and E. Snekkenes, editors, *ESORICS*, volume 10493 of *LNCS*, pages 532–551. Springer, 2017.

[YAHK11] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. Generic constructions for chosen-ciphertext secure attribute based encryption. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC*, volume 6571 of *LNCS*, pages 71–89. Springer, 2011.

[YAHK14] S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In H. Krawczyk, editor, *PKC*, volume 8383 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2014.

[YAS$^+$12] S. Yamada, N. Attrapadung, B. Santoso, J. C. N. Schuldt, G. Hanaoka, and N. Kunihiro. Verifiable predicate encryption and applications to CCA security and anonymous predicate authentication. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 243–261. Springer, 2012.

[YJ12] K. Yang and X. Jia. Attributed-based access control for multi-authority systems in cloud storage. In *IEEE Distributed Computing Systems*, pages 536–545. IEEE Computer Society, 2012.

[YJ14] K. Yang and X. Jia. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Trans. Parallel Distributed Syst.*, 25(7):1735–1744, 2014.

[YJR$^+$13] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie. DAC-MACS: effective data access control for multiauthority cloud storage systems. *IEEE Trans. Inf. Forensics Secur.*, 8(11):1790–1801, 2013.

[YJRZ13] K. Yang, X. Jia, K. Ren, and B. Zhang. DAC-MACS: effective data access control for multi-authority cloud storage systems. In *INFOCOM*, pages 2895–2903. IEEE, 2013.

[YWRL10] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In D. Feng, D. A. Basin, and P. Liu, editors, *ASIACCS*, pages 261–270. ACM, 2010.

[ZCa21] ZCash. Company. `https://z.cash/`, 2021.

[ZCL+13] Y. Zhang, X. Chen, J. Li, D. Wong, and H. Li. Anonymous attribute-based encryption supporting efficient decryption test. In K. Chen, Q. Xie, W. Qiu, N. Li, and W.-G. Tzeng, editors, *AsiaCCS*, pages 511–516. ACM, 2013.

[Zeu20] Zeutro. The OpenABE library - open source cryptographic library with attribute-based encryption implementations in C/C++. `https://github.com/zeutro/openabe`, 2020.

[ZGT+16] K. Zhang, J. Gong, S. Tang, J. Chen, X. Li, H. Qian, and Z. Cao. Practical and efficient attribute-based encryption with constant-size ciphertexts in outsourced verifiable computation. In X. Chen, X. Wang, and X. Huang, editors, *ASIACCS*, pages 269–279. ACM, 2016.

[ZH10a] Z. Zhou and D. Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption. Cryptology ePrint Archive, Report 2010/395, 2010.

[ZH10b] Z. Zhou and D. Huang. On efficient ciphertext-policy attribute based encryption and broadcast encryption: extended abstract. In E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, editors, *CCS (poster)*, pages 753–755. ACM, 2010.

[ZHW15] Z. Zhou, D. Huang, and Z. Wang. Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption. *IEEE Trans. Computers*, 64(1):126–138, 2015.

[ZkC20] ZkCrypto. Zero-knowledge cryptography in Rust. `https://github.com/zkcrypto`, 2020.

[ZPM+15] E. Zavattoni, L. J. Dominguez Perez, S. Mitsunari, A. H. Sánchez-Ramírez, T. Teruya, and F. Rodríguez-Henríquez. Software implementation of an attribute-based encryption scheme. *IEEE Trans. Computers*, 64(5):1429–1441, 2015.

List of Abbreviations

ABE	**A**ttribute-**b**ased **e**ncryption
ABAC	**A**ttribute-**b**ased **a**ccess **c**ontrol
ADM	**A**uthority-**d**ependence **m**inimization
CCA	**C**hosen-**c**iphertext **a**ttack
CPA	**C**hosen-**p**laintext **a**ttack
CP-ABE	**C**iphertext-**p**olicy **a**ttribute-**b**ased **e**ncryption
CR	**C**ollision **r**esistant
CT	**C**ipher**t**ext
(d_1, d_2)-pDBDH	(d_1, d_2)-**p**arallel **d**ecisional **b**ilinear **D**iffie-**H**ellman
DBDH	**D**ecisional **b**ilinear **D**iffie-**H**ellman
DNF	**D**isjunctive **n**ormal **f**orm
DSG	**D**ual **s**ystem **g**roups
EHR	**E**lectronic **h**ealth **r**ecord
FBE	**F**ixed-**b**ase **e**xponentiations
FDH	**F**ull-**d**omain **h**ash
GLUE	**G**eneralized, **l**arge-universe, **u**nbounded, **e**xpressive
GGM	**G**eneric **g**roup **m**odel
GP	**G**lobal **p**arameters
IBE	**I**dentity-**b**ased **e**ncryption
ICT	**I**ntermediate **c**ipher**t**ext
IND	**Ind**istinguishability
ISK	**I**ntermediate **s**ecret **k**ey
IoT	**I**nternet of **T**hings
KGA	**K**ey **g**eneration **a**uthority
KP-ABE	**K**ey-**p**olicy **a**ttribute-**b**ased **e**ncryption
LSSS	**L**inear **s**ecret **s**haring **s**cheme
MA-ABE	**M**ulti-**a**uthority **a**ttribute-**b**ased **e**ncryption
MA-CP-ABE	**M**ulti-**a**uthority **c**iphertext-**p**olicy **a**ttribute-**b**ased **e**ncryption
MBE	**M**ultiple-**b**ase **e**xponentiations
MPK	**M**aster **p**ublic **k**ey
MSK	**M**aster **s**ecret **k**ey
(N)MSP	(**N**on-)**m**onotone **s**pan **p**rogram
NT-ADM	**N**on-**t**rivial **a**uthority-**d**ependence **m**inimization
OO	**O**nline/**o**ffline

PE	**P**redicate **e**ncryption
P-KEM	**P**redicate **k**ey-**e**ncapsulation **m**echanism
Radboudumc	**R**adboud **U**niversity **M**edical **C**enter
RBAC	**R**ole-**b**ased **a**ccess **c**ontrol
ROM	**R**andom **o**racle **m**odel
RPC	**R**andom-**p**refix **c**ollision resistant
SK	**S**ecret **k**ey
(S)XDH	(**S**ymmetric) **e**xternal **D**iffie-**H**ellman
VBE	**V**ariable-**b**ase **e**xponentiation

Index

About the author

Marloes Venema was born on 13 May 1993 in Winterswijk, the Netherlands.

March 2023 – present
Postdoctoral Researcher
Cryptography and IT Security Group
University of Wuppertal, Germany

September 2018 – September 2023
PhD Candidate
Digital Security Group
Radboud University, the Netherlands

September 2016 – August 2018
Master of Science (cum laude)
Computing Science
Radboud University, the Netherlands

September 2011 – August 2016
Bachelor of Science
Mathematics
Radboud University, the Netherlands

Remember that it's real: the need to create it, the very thing that fulfills you.

Drew Gasparini

List of publications

9. A. de la Piedra, M. Venema, and G. Alpár. ACABELLA: automated (crypt)analysis of attribute-based encryption leveraging linear algebra. *To appear at CCS*, 2023

8. W. Ladd, T. Verma, M. Venema, A. Faz-Hérnandez, B. McMillion, A. Wildani, and N. Sullivan. Portunus: Re-imagining access control in distributed systems. In *USENIX ATC*, pages 35–52, 2023

7. L. Botros, M. Brandon, B. Jacobs, D. Ostkamp, H. Schraffenberger, and M. Venema. Postguard: Towards easy and secure email communication. In *CHI EA*. ACM, 2023

6. M. Venema and G. Alpár. GLUE: generalizing unbounded attribute-based encryption for flexible efficiency trade-offs. In A. Boldyreva and V. Kolesnikov, editors, *PKC*, volume 13940 of *LNCS*, pages 652–682. Springer, 2023

5. M. Venema. A practical compiler for attribute-based encryption: New decentralized constructions and more. In M. Rosulek, editor, *CT-RSA*, volume 13871 of *LNCS*, pages 132–159. Springer, 2023

4. M. Venema, G. Alpár, and J.-H. Hoepman. Systematizing core properties of pairing-based attribute-based encryption to uncover remaining challenges in enforcing access control in practice. *Des. Codes Cryptogr.*, 91(1):165–220, 2023

3. M. Venema and G. Alpár. TinyABE: Unrestricted ciphertext-policy attribute-based encryption for embedded devices and low-quality networks. In L. Batina and J. Daemen, editors, *AFRICACRYPT*, volume 13503 of *LNCS*, pages 103–129. Springer, 2022

2. A. de la Piedra, M. Venema, and G. Alpár. ABE squared: Accurately benchmarking efficiency of attribute-based encryption. *TCHES*, 2022(2):192–239, 2022

1. M. Venema and G. Alpár. A bunch of broken schemes: A simple yet powerful linear approach to analyzing security of attribute-based encryption. In K. G. Paterson, editor, *CT-RSA*, volume 12704 of *LNCS*, pages 100–125. Springer, 2021