Malleable Cryptography: Advances and Applications to Privacy-enhancing Technologies

Soutenue par
Octavio Pérez Kempner
Le 26 Octobre 2022

École doctorale n°386
Science Mathématiques de Paris Centre

Spécialité
Informatique

Defence Version (Wednesday 26th October, 2022)

To my sister Lucía

Résumé

Cette thèse étudie la malléabilité dans le contexte du chiffrement à clé publique et des signatures numériques, en présentant les avancées et les applications des technologies améliorant la confidentialité.

La première partie aborde le problème de l'égalité générique des textes en clair et les preuves d'inégalité. Étant donné deux textes chiffrés générés par un schéma de chiffrement à clé publique, le problème de l'égalité des textes chiffrés consiste à déterminer si les textes chiffrés contiennent la même valeur. Parallèlement, le problème de l'inégalité du texte clair consiste à déterminer s'ils contiennent une valeur différente. Les travaux précédents se sont concentrés sur la construction de nouveaux schémas ou sur l'extension de schémas existants afin d'inclure le support de l'égalité/inégalité du texte en clair. Nous proposons des preuves génériques et simples à connaissance zéro pour les deux problèmes, qui peuvent être instanciées avec divers schémas de chiffrement. Pour ce faire, nous formalisons les notions liées à la malléabilité dans le contexte du chiffrement à clé publique et proposons un cadre de définition pour le chiffrement générique aléatoire, que nous utilisons pour construire nos protocoles.

La partie suivante est consacrée aux signatures préservant la structure sur les classes d'équivalences, le principal élément constitutif des parties suivantes. Initialement, nous proposons des constructions nouvelles et plus efficaces sous des hypothèses standard. Ensuite, nous construisons un schéma d'accréditation établi sur les attributs sous des hypothèses standard, qui étend les travaux précédents de plusieurs façons. Nous améliorons notamment l'expressivité, les compromis d'efficacité et proposons une notion de dissimulation de l'émetteur qui permet aux détenteurs de lettres de créance de cacher l'identité de l'émetteur pendant les utilisations.

La dernière partie est consacrée à la présentation de Protego, un nouveau schéma d'accréditation pour les blockchains à autorisation. Il s'appuie sur les contributions précédentes et bien qu'il soit discuté dans le contexte des blockchains à autorisation, il peut également être utilisé dans d'autres contextes. Pour démontrer l'aspect pratique de Protego, nous fournissons un prototype et des benchmarks montrant que Protego est plus de deux fois plus rapide que les approches de l'état de l'art basées sur Idemix, le schéma d'accréditation le plus largement utilisé pour les blockchains à autorisation.

Abstract

This thesis studies malleability in the context of public-key encryption and digital signatures, presenting advances and applications to privacy-enhancing technologies.

The first part addresses the problem of Generic Plaintext Equality and Inequality Proofs. Given two ciphertexts generated with a public-key encryption scheme, the problem of plaintext equality consists in determining whether the ciphertexts hold the same value. Similarly, the problem of plaintext inequality consists in deciding whether they hold different values. Previous work has focused on building new schemes or extending existing ones to include support for plaintext equality/inequality. We propose generic and simple zero-knowledge proofs for both problems, which can be instantiated with various encryption schemes. We do so by formalizing notions related to malleability in the context of public-key encryption and proposing a definitional framework for Generic Randomisable Encryption, which we use to build our protocols.

The next part turns to Structure-Preserving Signatures on Equivalence Classes, the main building block of subsequent parts. First, we propose new and more efficient constructions under standard assumptions. Then, we build an anonymous attribute-based credential (ABC) scheme under standard assumptions, which extends previous work in several ways. We improve expressiveness, provide efficiency trade-offs and propose an issuer-hiding notion that allows credential holders to hide the issuer's identity during showings.

The last part is devoted to presenting Protego, a new ABC scheme for permissioned blockchains. It builds upon the previous contributions, and although it is discussed in the context of permissioned blockchains, it can also be used in other settings. To show the practicality of Protego, we provide a prototype implementation and benchmarks showing that Protego is more than twice faster than state-of-the-art approaches based on Idemix, the most widely used ABC scheme for permissioned blockchains.

Acknowledgements

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.

— Dante Alighieri

Lost in the dark forest, Dante found his way with the help of three beloved and admired guides (Virgil, Beatrice and Bernard de Clairvaux). As portrayed by Dante, major challenges in life cannot be tackled without the help of beloved and admired people who can guide us to obtain the required strength and wisdom. Doing a PhD thesis is one of such challenges. It requires fighting multiple battles on multiple fronts. Furthermore, paraphrasing something I once read, *doing a PhD means that you were able to irrationally pursue an unscoped objective for years and years.* Not everyone does that. However, I am writing these lines four years after I decided to go down the rabbit hole.

Cuba? are you sure? Will they give me a visa? Do you think the summer school is worth it? Do you know any of the speakers?. Five questions that completely changed the course of my life back in 2017. By that time, I already liked mathematics much more than when I started my computer science degree. The word “cryptography” was also starting to sound more intriguing. However, I would not have travelled to Cuba if Tuba (yes, a single letter made all the difference) had not answered all those questions positively for me. He encouraged me to find my way and to discover what the crypto (meaning *cryptography*) community looked like. Most importantly, like Virgil for Dante, he guided me through those winding roads, clearing out my fears and believing in me when I did not have the strength to do so. Thank you, Alfredo “Tuba” Viola.

The concept of grace, theologically speaking, is usually understood as a divine favour or love given generously, freely, unexpectedly and in a totally undeserved way. Beatrice represents it in Dante’s poem. Just like Beatrice for Dante, my beloved parents and sister have always been there for me. These few lines do not do justice to how much I love and cherish them. At best, they are one of the ways I found to put down in words how grateful I will be to them forever.

Unlike Dante, who met Bernard de Clairvaux (one of the purest souls in Dante’s universe) at the end of his trip, I met mine at the beginning. Not even in my *dreams* did I see myself doing a PhD in one of the finest schools in europe. Nevertheless, here I am, advocating for the “standardisation” of *café gourmand* in restaurants and happy with my french pronunciation of *croissant.* And it is all linked to my Bernard de Clairvaux, better known by the given Irish name meaning *dream*, Aisling. Meeting you in Cuba was definitely comparable to what Dante experienced with Bernard de Clairvaux, and I will always count my blessings to have met you.

That's what it means to be human - learn, grow, see how far you can get.
— Aisling Connolly

Doing a PhD allowed me to pursue my dreams, learn, grow and push the limits on how far I could get. I strengthened friendships, met incredible people, places and learned by doing. The following lines are dedicated to thank all the people that have also been part of this journey over the past four years.

To the jury. Je tiens à exprimer mes sincères remerciements à Sébastien Canard et Jean-Sébastien Coron pour s'intéresser aux travaux que j'ai réunis dans ce document et accepté d'être rapporteurs de mon manuscrit de thèse. I would also like to express my sincere thanks to Olivier Blazy, Anna Lysyanskaya, Carla Ràfols and Daniel Slamanig for agreeing to be part of my jury. The jury's previous work has inspired most of this thesis's contributions. It is a great honour to have the opportunity to present and discuss the results of this thesis with a jury from which I learned a lot over the past years.

To David. Shichi-go-san (seven-five-three). Based on the evidence, it only takes seven seconds to learn about a sharp idea of yours whenever I come across some of the big names in cryptography and tell them that I am one of your students. I hope it will stay like that for many years to come and that we keep using no less than five languages to discuss. You once told me, "nothing is impossible for David Naccache", and it took you three seconds to make the impossible possible. I will be forever grateful for the many opportunities, encouragement, and freedom you gave me.

To Pascal. It took me a while to see that your way of dealing with your students was very much alike the way I would go with students. From that point onwards, I have been learning things from you with every talk, and I cannot stress how happy I am to have met you. You helped me to develop the tools to do research, had patience when it was very much needed and always respected my time. Thank you for being a mentor in this journey. Here's to more collaborations and basketball ;-)

To Natacha. You have been unconditionally there for me. I wish every PhD student could have a Natacha Laniado. A godmother that teaches you the important stuff of doing a PhD. Someone that supports you through all the years it takes you. Not to mention the invaluable help to fix all the grammar and spelling mistakes in a matter of seconds to reach the submission deadlines on time. You have also been a friend and I very much thank you for that.

To my co-authors. Aisling, Aurelie, David, Jérôme, Mirko, Olivier, Pascal and Xavier. I learned a lot and really enjoyed working with you!

To ENS. A beautiful and unique place that hosted me for the past four years as my academic home. In particular, I would like to thank Stéphane, Fouzia and Sophie for their patience and help when trying to make the French administration happy.

To Almerys and the Be-ys group. Merci de m'avoir accueilli pour que je puisse faire une thèse CIFRE chez Almerys. Pendant ces quatre années, j'ai pu bénéficier d'un point de vue industriel et toujours avec une totale liberté de recherche.

To InCo-FING-UdelaR. My alma mater. Gracias a todos aquellos colegas y funcionarios que me apoyaron para que pueda continuar mis estudios en Francia.

To Criptolatino. Long live to this beautiful project!. I would like to thank Abdelrahaman, Arantxa, Daniel, Fabio, Fernando, Gustavo, Jesús-Javier and Sofía for their effort in building up the Latin American community in cryptography.

To David Manset. We faced some pretty tough challenges these past years. Thanks for being there to discuss and define the best strategy to follow during the difficult times. Thanks for trusting me. Thanks for the all the talks, your mentorship, and teachings. It has been a pleasure to fight alongside you. Santé!

To Olivier Raynaud. Merci de tous les efforts que tu as fait pour me fournir un bon environnement de travail et les meilleures conditions pour mener à bien nos projets de recherche.

To Santiago. El segundo padrino que me dió la vida. Gracias por estar siempre, por la confianza, por escuchar, aconsejar y por enseñar lo fundamental de las cosas.

To Mirko. Horribles chistes pero por suerte gran tipo y amigo. Gracias totales por estar siempre al firme, acompañando, aconsejando, bancando y compartiendo comidas y cafés. Orgulloso de haberte sacado un poco de la oficina estos años!

To Rolando. Has sido parte de todo esto desde antes que supiéramos que iba a estudiar en París. Gracias por las constantes enseñanzas y la amistad de siempre.

To Mariana. Amiga de todas las horas. Saber que estás y que sepas que estoy a pesar de la distancia y del tiempo siempre me saca una sonrisa :)

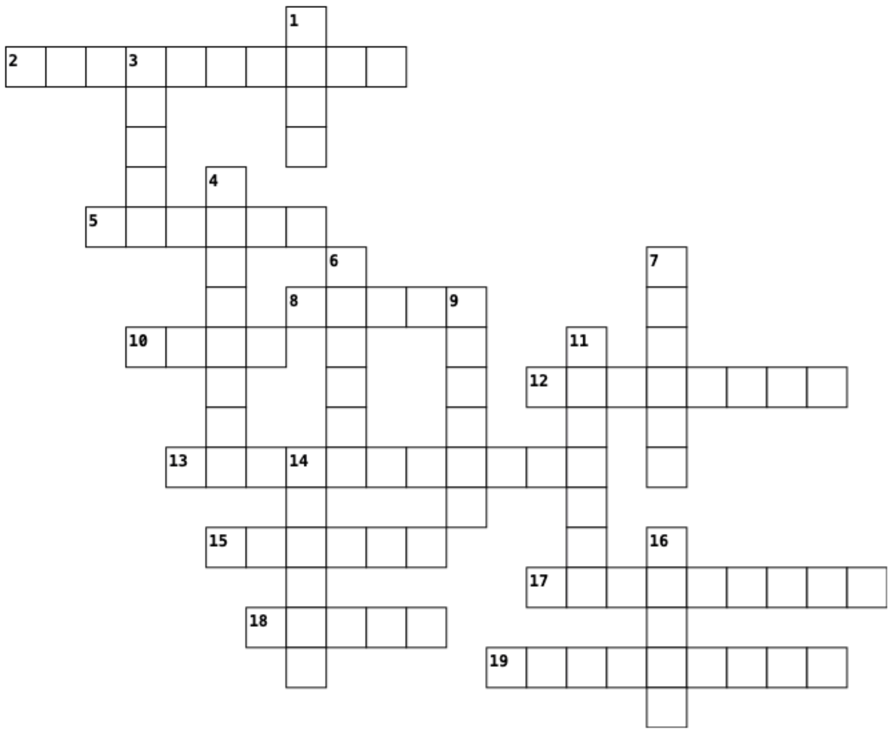
To Aure and Elena. Conseguirme un piso en frente a la residencia fue la primer buena idea que tuve en mi doctorado. Gracias por enseñarme la diferencia entre comida y almuerzo, a cocinar paella, tortilla y mil cosas más!

To Victoria. Gracias por todas las charlas, risas, consejos y mensajes de aliento. Lo difícil cuesta pero es más fácil lograr objetivos cuando uno tiene amigos que admira. Y si alcanzamos una cumbre, siempre hay otra más allá ...

To Nicolás. Un hermano que me dió la vida. Gracias por seguir al firme a pesar de la distancia. No tantas gracias por mantemerme al tanto de la realidad rioplantese.

Cultivo una rosa blanca,
En julio como en enero,
Para el amigo sincero
Que me da su mano franca.
Y para el cruel que me arranca
El corazón con que vivo,
Cardo ni oruga cultivo:
Cultivo la rosa blanca.

— José Martí



Across
2. Ni a Jagger le calza mejor el Rock 'n roll
5. Jackson, testigo de mucha noche
8. Patricia bien fría o una Quilmes?
10. Excesivo uso de la palabra "gurise"
12. Caminatas, bicicleateadas y tulipanes
13. Le dicen sanguijuela y es manya
15. Suele vestir de negro
17. Marinero y pescador de verdad
18. Pasan los años y la lapicera guardada
19. De Talisker sabe el hombre

Down
1. Un chocolate con ...
3. Siempre con un nombre gammer
4. Treinta y Tres
6. Viene en tamaño "pocket"
7. Compañero de viajes inolvidables
9. Seguirá de patineta por la rambla?
11. De Colonia al mundo sin escalas
14. Discutidor nato
16. Bigote

To Malos Pensamientos. Gracias a Jorge Díaz, Orlando Petinatti, Luigi Tempone y a la mejor audiencia del mundo. Hicieron que la pandemia fuera más llevadera.

List of Abbreviations

ABC	Anonymous Attribute-Based Credentials
CA	Certificate Authority
CRS	Common Reference String
DL	Discrete Logarithm
DDH	Decisional Diffie-Hellman
EEA	Extended Euclidean Algorithm
EUF-CMA	Existential Unforgeability under adaptive Chosen-Message Attacks
EUF-CoMA	Existential Unforgeability under Chosen open Message Attacks
extKerMDH	External Kernel Diffie-Hellman
GDPR	General Data Protection Regulation
GGM	Generic Group Model
GSDH	Generalised Strong Diffie-Hellman
HVZK	Honest Verifier Zero-Knowledge
IND-CPA	Indistinguishability of ciphertexts under Chosen-Plaintext Attacks
IND-CCA1	Indistinguishability of ciphertexts under non-adaptive Chosen-Ciphertext Attacks
IND-CCA2	Indistinguishability of ciphertexts under adaptive Chosen-Ciphertext Attacks
IND-RCCA	Indistinguishability of ciphertexts under Replayable Chosen-Ciphertext Attacks
IK-CPA	Indistinguishability of Keys under Chosen-Plaintext Attacks
IK-CCA	Indistinguishability of Keys under Chosen-Ciphertext Attacks

KerMDH	Kernel Diffie-Hellman
KeyRand	Key-Randomisable
MDDH	Matrix Diffie-Hellman
MAC	Message Authentication Code
MSP	Membership Service Provide
MsgRand	Message-Randomisable
NIZK	Non-Interactive Zero-Knowledge proof
PETs	Privacy-enhancing Technologies
PKE	Public-key Encryption
PoE	Proof of Exponentiation
QA-NIZK	Quasi-Adaptive Non-Interactive Zero-Knowledge proof
RCD	Random Coin Decryptable
Rand	Randomisable
SCDS	Set-Commietment scheme supporting Disjoint Sets
SDH	Strong Diffie-Hellman
SHVZK	Special Honest Verifier Zero-Knowledge
SPS	Structure-Preserving Signatures
SPS-EQ	Structure-Preserving Signatures on Equivalence Classes
SXDH	Symmetric eXternal Diffie-Hellman
ZKP	Zero-Knowledge Proof
ZKPoK	Zero-Knowledge Proof of Knowledge

List of Figures

List of Tables

Contents

1
Introduction

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel.

— Whitfield Diffie & Martin Hellman

1.1 Modern Cryptography

Cryptography (and, more generally, the field of cryptology) has gone from being considered an art to gaining recognition as a science and mathematical discipline throughout history. Commonly known as *the science of secret*, a term coined by Jacques Stern, the father of modern French cryptology, it was not until the late 1970s that it began to show its full potential.

The groundbreaking work of Diffie and Hellman from 1976, "New Directions in Cryptography" [DH76], presented it as "*the study of "mathematical" systems for solving two kinds of security problems: privacy and authentication*". However, the introduction of Public-key Encryption (PKE) in [DH76] was so revolutionary that the words "privacy" and "authentication", although still fundamental, fall short of describing it thereafter. Instead, the broader concept of *modern* cryptography began to be used to encompass all subsequent developments in the field.

The concept of modern cryptography is best put through the words of Katz and Lindell when they mention in their book that *"it involves the study of mathematical techniques for securing digital information, systems, and distributed computations against adversarial attacks"* [KL21].

Following the dot-com bubble and the massive adoption of the Internet, the entire digital universe surpassed 44 zettabytes of data in 2020. In other words, in 2020 there were 40 times more bytes in the digital world than stars in the observable universe [Des19]. This scenario highlighted the tensions of regulating the use of digital data to protect human rights as a rocky trail.

While regulations can help individuals to exercise their rights in case of abuse, they can also become a powerful tool for governments to restrict certain rights whenever they are ill-defined or purposely wrong. For this reason, a lot of recent efforts in modern cryptography have been done in the context of *Privacy-enhancing Technologies* (or simply PETs), which aim at easing the previous tensions in the following way:

Privacy-enhancing Technologies is a system of Information and Communication Technology (ICT) measures protecting informational privacy by eliminating or minimising personal data, thereby preventing unnecessary or unwanted processing of personal data without the loss of the functionality of the information system.
— [BVE+03]

In this thesis, we dive into modern cryptography through the optics of privacy-enhancing technologies. Therefore, our focus and contributions are designing practical and efficient techniques to protect informational privacy by preventing unnecessary or unwanted processing of personal (or private) data.

To achieve the goals mentioned above, we present techniques based on zero-knowledge proofs, digital signatures and, in general, malleable cryptography.

1.2 Malleable Cryptography

Historically speaking, the concept of malleability in cryptography was first introduced by the seminal work of Dolev, Dwork, and Naor titled "Non-malleable Cryptography" [DDN91]. Authors first observed that the notion of semantic security was not enough for some applications in the context of public-key encryption. Subsequently, they proposed an extension to it so that given a ciphertext, it is impossible to generate a different one such that the respective plaintexts are related. In other words, the key observation made by the authors was that while semantic security ensured *privacy*, it did not imply *independence*. Thus, their goal was to force that implication.

With the above in mind, the previous problem turned into ensuring that given a ciphertext c, it should not be possible to somehow change it (*malleate it*) to obtain another ciphertext c' such that the message under c' is related to that under c. As a result, the notion was called "non-malleability", and it was latter shown to be equivalent to the notion of ciphertext indistinguishability under adaptive chosen-ciphertext attacks (IND-CCA2) by Bellare and Sahai [BS99] (see Chapter 2 for a detailed discussion).

The contributions by Dolev, Dwork and Naor also included a formalization of the previous idea in the context of string commitments and zero-knowledge proofs of knowledge. This set the ground for further contributions in the field, which includes the works by [Sah99] on Non-malleable Non-Interactive Zero-Knowledge Proofs, by Dziembowski *et al.* on Non-malleable Codes [DPW10], and by Goyal and Kumar on Non-malleable Secret Sharing [GK18].

While the original motivation to build non-malleable primitives continues to be an active field of research (*e.g.,* [RS21, BGW19, FKPS21]), malleability can also be seen as a positive feature instead of a weakness or an attack. In this regard, one of the first references is due to Shoup [Sho01], who suggested the term *benign malleability* as a (positive) relaxation of IND-CCA2 (also further relaxed by Prabhakaran and Rosulek [PR07], as discussed in Chapter 2). Such relaxations characterised ciphertext's randomisations (without acting on the related plaintexts). The first attempt to tackle the problem of controlled malleability in PKE with respect to the plaintexts was made by Boneh, Segev and Waters [BSW12]. They introduced the concept of *targeted malleability* to allow the computation of a specific set of "allowable" functions.

Because the reader may find the terms "malleable" and "homomorphic" being used interchangeably in some works, we would like to make the following distinction. While (fully) homomorphic encryption [Gen09] realizes (general) computations over encrypted data, our interest is broader. In this regard, the term homomorphic can be confusing as it refers to a mathematical property. The same distinction would apply to other primitives like fully homomorphic signatures [GVW15] and (fully homomorphic) Non-Interactive Zero-Knowlede (NIZK) proofs [ADKL19]. Thus, we consider the term malleable to be better suited than homomorphic to reason about a particular functionality regardless of how it is actually implemented. Therefore, we opt to use the term malleable and not homomorphic, emphasizing that one would usually work with a controlled form of malleability, determined by some concrete functionality, implemented in certain way.

Besides the work on malleable PKE, this thesis also explores malleable proof systems [CKLM12] and malleable signatures [CKLM14]. The latter, in the form of Structure-Preserving Signatures on Equivalence Classes (SPS-EQ) [FHS19]. Perhaps surprisingly, the idea of equivalence classes will be both, our starting and ending point as it was a concept also discussed by Shoup in [Sho01].

1.3 Summary of Contributions

This thesis studies malleable cryptography in the context of PKE, proof systems and digital signatures. It builds upon the following works:

- *Generic Plaintext Equality and Inequality Proofs* [BBLPK21a], presented at Financial Cryptography 2021, and its full version [BBLPK21b].
- *Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes* [CLPK22], presented at Public Key Cryptography 2022, and its full version [CLPK21].
- *Protego: Efficient, Revocable and Auditable Anonymous Credentials with Applications to Hyperledger Fabric* [CDLPK22], accepted at Indocrypt 2022.

As a result, advances in recent constructions and their application to privacy-enhancing technologies are presented. Furthermore, implementations and benchmarks considering state-of-the-art alternatives are provided to show the practicality of our contributions. Below, we briefly present each of the above works.

1.3.1 Generic Plaintext Equality and Inequality Proofs

Given two ciphertexts generated with a PKE scheme, the problem of plaintext equality consists in determining whether the ciphertexts encrypt the same message. Similarly, the problem of plaintext inequality consists in deciding whether they encrypt a different message.

Previous work has focused on building new schemes or extending existing ones to include support for plaintext equality/inequality. Instead, we propose generic and simple zero-knowledge proofs for both problems, which can be instantiated with various encryption schemes. To support this claim, we list a number of them indicating the relation with our protocols.

First, we consider the context where a prover with access to the secret key wants to convince a verifier, who has access to the public key and ciphertexts, on the equality/inequality without revealing information about the plaintexts. We also consider the case where the prover knows the encryption's randomness instead of the secret key. Finally, we propose sigma protocols (*i.e.,* public coin, three-move protocols with honest verifier zero-knowledge) for plaintext equality that lead to NIZK proofs in the random oracle model via the Fiat-Shamir transform.

To prove our protocols security, we formalize notions related to malleability in the context of PKE proposing a definitional framework for *Generic Randomisable Encryption*, which we use to build our protocols.

Finally, we also see an added value in our contributions in terms of serving as a pedagogical tool to introduce zero-knowledge proofs (ZKP). Usual examples are graph 3-coloring or graph isomorphism. Although such protocols can be explained without requiring any advanced cryptographic knowledge, they are not used in real-world applications. On the contrary, our protocols are useful for real-world applications of ZKP. Furthermore, the protocols that we present are very intuitive and can easily be explained without requiring advanced cryptographic knowledge outside the concept of PKE (*i.e.,* it is not necessary to understand how an encryption scheme works). For these reasons, we think our protocols can serve as a convincing pedagogical example to explain ZKP to a larger audience with little mathematical background.

1.3.2 Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes

Anonymous attribute-based credentials (ABCs) are a powerful tool allowing users to authenticate while maintaining privacy. When instantiated from SPS-EQ, one can obtain a controlled form of malleability, leading to increased functionality and privacy for the user.

Existing constructions consider equivalence classes on the message space, allowing the joint randomisation of credentials and their corresponding signatures. In this work, we additionally consider equivalence classes on the *key space* [CL19]. As a result, we obtain an *issuer-hiding* notion, where the issuing organisation is not revealed when a user shows a credential. To realize it, we modify a recent SPS-EQ scheme [KSD19] to support a fully adaptive NIZK proof from [CH20], showing how to extend it to support equivalence classes on the key space.

Our work follows the ABC framework from Fuchsbauer, Hanser and Slamanig [FHS19], improving over prior work in the following ways:

1. We extend the set-commitment scheme from [FHS19] to build a more expressive credential system allowing the generation of witnesses for disjoint sets ([FHS19] only allows selective disclosure of attributes). This way, users

are able to prove that they do not hold a given set of attributes in their credentials more easily.

2. We incorporate a proof of exponentiation to outsource part of the computational cost from the verifier to the prover, which can be very useful in several settings.
3. As previously mentioned, users can hide the identity of the issuer during showings if the underlying SPS-EQ provides issuer-hiding features.

Relying on a Common Reference String, we obtain an efficient credential system with increased expressiveness and privacy, whose security can be proven under standard assumptions.

1.3.3 Protego: Efficient, Revocable and Auditable Anonymous Credentials with Applications to Hyperledger Fabric

Recent works to improve privacy and auditability in permissioned blockchains like Hyperledger Fabric relies on the Idemix credential system [Zur13] as it is the only anonymous credential system that has been integrated to date. However, the current Idemix implementation in Hyperledger Fabric (v2.4) only supports a fixed set of attributes. It does not support revocation features, nor does it support anonymous endorsement of transactions (in Fabric, transactions need to be endorsed by a subset of peers before they can go through the consensus process). Moreover, as Idemix uses a blind signature scheme with zero-knowledge proofs of signature possession, it involves linear computation and communication in the number of attributes. A prototype Idemix extension based on delegatable credentials was proposed to include revocation, auditability, and to gain privacy for users. This recent extension by Bogatov *et al.* [BDCET21] from 2021 makes extensive use of pseudonyms and zero-knowledge proofs.

This work explores how to gain efficiency, functionality, and further privacy for users by departing from recent work on anonymous credentials based on Structure-Preserving Signatures on Equivalence Classes. More in detail:

1. We extend recent works in ABCs [FHS19, CLPK22] based on SPS-EQ to support auditability features while also integrating the revocation features from [DHS15a]. This extension relies on the random oracle model (already present in the blockchain setting) to generate non-interactive showing proofs.
2. A showing involving k-of-n attributes (selective disclosure) has asymptotic complexity $O(n)$ for both the user and the verifier with Idemix. The use of SPS-EQ lowers them to $O(\max\{n-k, k\})$ and $O(1)$, respectively.
3. We present and discuss two alternatives to the use of delegatable credentials to hide the identity of credential issuers based on the SPS-EQ properties.

As a result, we propose Protego and Protego Duo, two alternatives for Idemix and its recent extensions. While Protego is based on the issuer-hiding ideas from [CLPK22], Protego Duo relies on the ones from [BEK$^+$21]. We discuss how they can be used in the permissioned blockchain setting and integrated to Hyperledger Fabric. We also provide a prototype implementation, showing that both alternatives are twice as fast as state-of-the-art approaches. Along the way, we review the privacy model in Fabric illustrating the various privacy requirements.

1.4 Other Contributions

We have contributed to two projects from the European Union's Horizon 2020 Research and Innovation Programme. These are MyHealthMyData (under grant agreement No 732907) and CUREX (under grant agreement No 826404). Both involved the design and development of blockchain-based solutions, serving as an inspiration for our work on zero-knowledge proofs and anonymous credentials. Contributions related to these projects are not described here because, even though they also target privacy-enhancing technologies, they are not associated with the concept of malleable cryptography, which is the main topic of this thesis. Therefore, the reader is referred to the related projects for a complete list of publications.

1.5 Organisation of this Thesis

This thesis is organised into seven chapters as follows:

Chapter 1 is the present introduction.

Chapter 2 contains the necessary preliminaries to assist the reader in subsequent chapters. It introduces the different formalisms, security notions, notation, cryptographic assumptions and primitives.

Chapter 3 presents our contributions in the context of PKE and is based on the work in [BBLPK21a]. This chapter can be read independently of the remainder of this thesis.

Chapter 4 serves as an introduction to Chapter 5 and 6. It is based on the work in [CLPK22] and [CDLPK22]. This chapter presents constructions of SPS-EQ under standard assumptions.

Chapter 5 presents our ABC scheme in [CLPK22], which depends on the previous chapter.

Chapter 6 extends the ABC scheme from the previous chapter, presenting new features and a more efficient instantiation. It discusses its integration with permissioned blockchains and is based on the work in [CDLPK22].

Chapter 7 concludes this thesis discussing open issues and future work to expand the work presented in this thesis.

2 Preliminaries

This is a field in which all the giants have been proven wrong multiple times. (The trick is to be proven wrong in interesting ways.)

— Matt Blaze

2.1 Provable Security

We presented how cryptography evolved from being considered as an art to gaining recognition as a science in the introduction. In the present section, we elaborate on how the notion of provable security played a key role in that transition.

As beautifully explained in the brief introduction to Provable Security by Michel Abdalla [Abd14], the most common approach to validate the security of a cryptographic scheme in the past, was to search for attacks and to declare the scheme secure if no attack was found to contradict its security. Unfortunately, such an approach cannot exclude the possibility that an attack exists, demeaning one's hope to achieve a heuristic notion of security at best. Furthermore, *ad hoc* attacks and defence mechanisms were the main resources used by both sides trying to outsmart the other when reasoning about the security of a concrete scheme.

Provable security builds upon the following three pillars to overcome the above limitations when arguing the security of a cryptographic scheme: *definitions*, *reductions* and *hard problems*. First, it allows us to abstract and define general notions for cryptographic primitives, which enable us to reason about their security [Con19]. Secondly, it allows us to relate these notions formally in terms of reductions as done in Complexity Theory to show that solving a problem A is at least as difficult as solving another problem B. Ultimately, it relies on hard problems to build meaningful reductions.

To define a general notion properly, definitions are given in the form of a syntax alongside correctness and security properties. The syntax specifies the language used to describe a particular primitive, the expected input and output of the related algorithms, parameters and domains. Correctness specifies the required interactions to consider the scheme to work correctly. Finally, the security properties specify the goal that the primitive should achieve when considering adversaries with some specified capabilities.

All in all, provable security provides a rigorous framework for cryptographers to define and analyze the security of cryptographic primitives, as long as the model captures reality in a meaningful way.

2.2 Notation

Integers, sets and probabilities. We denote by $\mathbb{Z}$ and $\mathbb{N}$ the sets of integers and positive integers. The set of integers $1, 2, ..., n$ is denoted $[n]$. If $p \in \mathbb{N}$, we call $\mathbb{Z}_p$ the ring of integers modulus p, which has the structure of a field when p is prime. For a set $\mathcal{S}$ and $r \in \mathcal{S}$, we use the notation $r \xleftarrow{\$} \mathcal{S}$ to indicate that r has been sampled uniformly at random from $\mathcal{S}$. For any finite set of elements $\mathcal{S}$, $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. A function $\varepsilon(\lambda) : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $\mathsf{poly}(\lambda) : \mathbb{N} \to \mathbb{R}$, there exists $\lambda_0 \in \mathbb{N}$ such that $\varepsilon(\lambda) \leq 1/\mathsf{poly}(\lambda)$ for all $\lambda \geq \lambda_0$. The notation $\mathsf{poly}(\lambda)$ is also used for polynomial functions in general. For a random variable X and possible outcome x, we denote $\Pr[X = x]$ the probability of the event $X = x$.

Algorithms. Throught this thesis, we work with probabilistic polynomial time algorithms (or p.p.t for short). This class of algorithms has access to a random tape, and their execution time is bounded by some polynomial poly in the size of some input $\lambda \in \mathbb{N}$, on all inputs and all random tapes. For this reason, λ is referred to as the *security parameter*. Since what usually matters is its length in bits, it is passed as input to all algorithms in the unary form 1^λ.

All p.p.t algorithms are assumed to use a random tape, and produce non-deterministic results. When a p.p.t algorithm does not use its random tape, we explicitly refer to it as deterministic. Sometimes, we refer to p.p.t algorithms as *efficient* algorithms.

For an algorithm $\mathcal{A}$ we denote the execution of $\mathcal{A}$ with input x and output y as $y \leftarrow \mathcal{A}(x)$ if the execution is deterministic and by $y \xleftarrow{\$} \mathcal{A}(x)$ if the execution is probabilistic. Sometimes, we overload the operator $\leftarrow$ to denote assignments. The notation $\mathcal{A}(x; r)$ is used when a random value r (usually computed internally by $\mathcal{A}$) is directly passed to $\mathcal{A}$ on input x. Whenever an algorithm fails, the output is set to $\perp$. Public parameters (denoted as pp) are often passed implicitly to all algorithms (although they are properly defined in the syntax of each cryptographic primitive). Besides the usual syntax to specify algorithms in terms of flow control sentences, we also use the sentence check to compute boolean values, assuming the algorithm aborts whenever a result within a check block is false.

Pairing-based Cryptography. Let BGGen be a p.p.t algorithm that on input 1^λ, returns a description $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$ of an asymmetric bilinear group where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p with $\lceil \log_2 p \rceil = \lambda$, P_1 and P_2

are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable (non-degenerate, *i.e.,* $e(P_1, P_2)$ generates $\mathbb{G}_T$) bilinear map. BG is said to be of Type-3 if no efficiently computable isomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$ are known. For all $a \in \mathbb{Z}_p$, we denote by $[a]_s = aP_s \in \mathbb{G}_s$ the implicit representation of a in $\mathbb{G}_s$ for $s \in \{1, 2\}$. For matrices (or vectors) $\mathbf{A}$, $\mathbf{B}$ we extend the pairing notation to $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T \in \mathbb{G}_T$.

Polynomials. For a set $\mathcal{X}$ with elements in $\mathbb{Z}_p$, we refer to $\mathsf{Ch}_{\mathcal{X}}(X) = \prod_{x \in \mathcal{X}}(X + x) = \sum_{i=0}^{i=n} c_i \cdot X^i$ (a monic polynomial of degree $n = |\mathcal{X}|$ and defined over $\mathbb{Z}_p[X]$) as its *characteristic polynomial.* For a group generator P, $\mathsf{Ch}_{\mathcal{X}}(s)P$ can be efficiently computed (*e.g.,* using the Fast Fourier Transform) when given $(s^i P)_{i=0}^{|\mathcal{X}|}$ but not s. This is because $\mathsf{Ch}_{\mathcal{X}}(s)P = \sum_{i=0}^{i=n}(c_i \cdot s^i)P$.

Other symbols and shorthands. Black-box algorithms used as oracles are denoted as $\mathcal{O}$ and accessed as $\mathcal{A}^{\mathcal{O}(x)}(y)$. Cryptographic hash functions are denoted by $\mathcal{H}(x)$. Whenever the symbol $:=$ is written, it is a shorthand for "*defined as*". The symbol $\approx$ is used to indicate that two probability ensembles are statistically close. Equivalence relations are denoted with the symbol $\sim_{\mathcal{R}}$, with $\mathcal{R}$ being the equivalence relation. Similarly, a member x of an equivalence class $\mathcal{R}$ is denoted within brackets as $[x]_{\mathcal{R}}$ (slightly overloading the same notation for pairings).

2.3 Cryptographic Background

In the following, we present the security models, algorithms and lemmas related to working with polynomials and the cryptographic assumptions used in this thesis.

2.3.1 Models

We use models in cryptography to define the adversary's capabilities (*i.e.,* the power it is assumed to have when attempting to break a cryptographic scheme). The strongest model is known as the *standard model*, and it only assumes that the adversary is limited by the amount of time and computational power available. Execution time is usually measured in terms of computation steps for some reasonable notion of step (*i.e.,* CPU cycles). In contrast, computational power is given by the computational model used (*i.e.,* assuming the adversary can be modeled as a Turing machine [Tur36]).

The most widely used assumptions in the standard model are the hardness of the discrete logarithm problem and integer factorization. Although it may sound unintuitive, *weaker* assumptions are preferred over *stronger* assumptions. This is because if an assumption A is stronger than B, it means that A implies B. Therefore, A represents an "easier" problem that can be reduced to a "harder" one since ¬B implies ¬A.

As cryptographic primitives evolve towards more complex constructions and dependant on different components, analyzing their security becomes harder. For this reason, more than often, it is useful to work with *idealised* models that capture some desired property or set of properties in a *black-box* manner. By black-box manner, we mean that access to some *idealised* object that realizes the desired properties is provided using a clear interface and abstracting everything else, which is assumed to work as exactly as specified. We work with two idealised models, the Random Oracle Model (ROM) [BR93] and the Generic Group Model (GGM) [Sho97, Mau05]. We also work with the Common Reference String Model (CRS) [Dam00], which assumes the existence of a trusted setup.

Random Oracle Model. This model provides a truly random function that returns the same value when queried with a given input twice. For this to work, the function would require a table to store the input queries alongside the corresponding output, leading to a description that would be exponentially large. It follows that such a function can only be an idealised object as an efficiently-computable function achieving that property cannot exist. In practice, real-world applications rely on cryptographic hash functions to instantiate random oracles.

If an adversary can break the security of a scheme in the ROM, the immediate conclusion is that the hash function used to instantiate the random oracle was not "good enough" [Kat02]. Therefore, using the ROM gives confidence relative to finding some weakness in the hash function used to instantiate it. That said, a scheme secure in the ROM could be broken without finding a weakness in the hash function used so this model is clearly a relaxation of the standard model. Moreover, there are artificial schemes proven secure in the ROM, but for which any instantiation with a hash function makes them insecure [CGH98, GR04]. Put differently, a proof in the ROM does not guarantee the same security as the standard model, but it is clear improvement over having no proof at all.

Generic Group Model. It is similar to the ROM, but abstracting computations in a group. The aim is to encapsulate the fact that group elements do not give any information about the underlying group structure that can be exploited to break the scheme in question. To that end, an oracle returns a random encoding of resulting group elements in a consistent way with respect to previous queries and any algorithm having access to such encodings can only perform two actions: compute the composition of two group elements and check for equality. Like the ROM, it suffers similar shortcomings since, in practice, the group structure under which a cryptographic primitive is implemented will usually be known and could potentially be exploited. That being said, the GGM is a widely used model when looking for efficient constructions. A more detailed presentation of the GGM and related models can be found in [Plo21].

Common Reference String Model. This model assumes the existence of a trusted third party responsible for generating a common reference string (or public parameters) correctly and honestly (*i.e.,* discarding any auxiliary information that could be used to abuse the scheme in question). Sometimes this model is referred to as the *auxiliary* string model or the *common random* string model (if the reference string happens to be a uniformly random string).

2.3.2 Polynomials

When working with polynomials, we use the Schwartz-Zippel lemma and the Extended Euclidean Algorithm (EEA) as presented in [GOP+16].

Lemma 1 (Schwartz-Zippel)**.** Let $q_1(x), q_2(x)$ be two d-degree polynomials from $\mathbb{Z}_p[X]$ with $q_1(x) \neq q_2(x)$, then for $s \xleftarrow{\$} \mathbb{Z}_p$, the probability that $q_1(s) = q_2(s)$ is at most d/p, and the equality can be tested in time $O(d)$.

2.3.3 Assumptions

In the following, we recall well-known Diffie-Hellman assumptions in the bilinear group setting and matrix assumptions introduced with the algebraic framework from [EHK+17] and [MRV16]. We include a generalisation of the Strong Diffie-Hellman assumption, called the q-co-Generalised-Strong-Diffie-Hellman assumption, first introduced in [FHS19]. Besides, we also include the extKerMDH assumption [CH20], a generalisation of the KerMDH assumption.

Let BGGen be a bilinear-group generator that outputs $\mathsf{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P_1, P_2, e)$. For $k \in \mathbb{N}$, we denote by $\mathcal{D}_k$ a matrix distribution that outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.

Diffie-Hellman Assumptions

DL. The *Discrete Logarithm* assumption holds in $\mathbb{G}_i$ for BGGen if for all probabilistic polynomial-time (p.p.t) adversaries $\mathcal{A}$, the following probability is negligible.

$$\Pr\left[\ \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); a \xleftarrow{\$} \mathbb{Z}_p; a' \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, aP_i)\ :\ a' = a\ \right]$$

q-co-DL. The *q-co-Discrete Logarithm* assumption holds in BGGen if for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\Pr\left[\ \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); a \xleftarrow{\$} \mathbb{Z}_p; a' \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, (a^jP_1, a^jP_2)_{j\in[q]})\ :\ a' = a\ \right]$$

DDH. The *Decisional Diffie-Hellman* assumption holds in $\mathbb{G}_i$ for BGGen, if for all p.p.t adversaries $\mathcal{A}$ the following probability is negligible,

$$\Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}, \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda), r, s, t \xleftarrow{\$} \mathbb{Z}_p \\ b^* \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, rP_i, sP_i, ((1-b)\cdot t + b \cdot rs)P_i) \end{array} \ :\ b^* = b\right] - \frac{1}{2}$$

SXDH. The *Symmetric eXternal Diffie-Hellman* assumption holds for BGGen if DDH holds in $\mathbb{G}_1$ and in $\mathbb{G}_2$.

q-co-GSDH. The *q-co-Generalised-Strong-Diffie-Hellman* assumption holds for BGGen, if for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\Pr\left[\begin{array}{l} \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda), s \xleftarrow{\$} \mathbb{Z}_p^* \\ (Q, f_1, f_2) \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, (s^iP_1, s^iP_2)_{0<i\leq q}) \end{array} \ :\ \begin{array}{l} Q \in \mathbb{G}_1\ \wedge f_1, f_2 \in \mathbb{Z}_p[X] \wedge \\ 0 \leq deg\ f_1 < deg\ f_2 \leq q \wedge \\ e(Q, f_2(s)P_2) = e(f_1(s)P_1, P_2) \end{array}\right]$$

$\mathcal{D}_k$-MDDH. The $\mathcal{D}_k$*-Matrix Diffie-Hellman* assumption holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_p^k$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ and all p.p.t adversaries $\mathcal{A}$, the following advantage is negligible,

$$\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) = |\Pr[\mathcal{A}(\mathsf{BG},[\mathbf{A}]_s,[\mathbf{Aw}]_s)=1] - \Pr[\mathcal{A}(\mathsf{BG},[\mathbf{A}]_s,[\mathbf{u}]_s)=1]|$$

$\mathcal{D}_k$-KerMDH. Let $s \in \{1,2\}$. The $\mathcal{D}_k$-*Kernel Diffie-Hellman* assumption holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$ and all p.p.t adversaries $\mathcal{A}$, the following advantage is negligible,

$$\mathbf{Adv}^{\mathsf{KerMDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) = \Pr\Big[[\mathbf{c}]_{3-s} \xleftarrow{\$} \mathcal{A}(\mathsf{BG},[\mathbf{A}]_s) : \mathbf{c}^\intercal\mathbf{A} = 0 \wedge \mathbf{c} \neq 0\Big]$$

The $\mathcal{D}_k$-KerMDH assumption is the computational analogue of the $\mathcal{D}_k$-MDDH assumption for any matrix $\mathcal{D}_k$. The following lemma relates both assumptions.

Lemma 2 (**$\mathcal{D}_k$-$\mathsf{MDDH} \Rightarrow \mathcal{D}_k$-$\mathsf{KerMDH}$** [MRV16])**.** Let $k \in \mathbb{N}$ and let $\mathcal{D}_k$ be a matrix distribution. For any p.p.t adversary $\mathcal{A}$, there exists a p.p.t adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\mathsf{KerMDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s}(\mathcal{B})$.

As stated in [GHKP18], for $Q \in \mathbb{N}$, $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times Q}$ and $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times Q}$, one can also consider the Q-fold $\mathcal{D}_k$-MDDH assumption, which states that distinguishing tuples of the form $([\mathbf{A}]_s,[\mathbf{AW}]_s)$ from $([\mathbf{A}]_i,[\mathbf{U}]_i)$ is hard. A challenge for the Q-fold $\mathcal{D}_k$-MDDH assumption consists of Q independent challenges of the $\mathcal{D}_k$-MDDH assumption (with the same $\mathbf{A}$ but different randomness $\mathbf{w}$). In [EHK$^+$13], it is shown that the two problems are equivalent, where the reduction loses at most a factor 1 (since we work with dimensions $(k+1)\times k$). The following lemma relates both problems.

Lemma 3 (**Random self-reducibility of $\mathcal{D}_k$-MDDH** [EHK$^+$13])**.** Let $k, Q \in \mathbb{N}$ with $Q > 1$ and $s \in \{1,2,T\}$. For any p.p.t adversary $\mathcal{A}$, there exists an adversary $\mathcal{B}$ with running time $T(\mathcal{B}) \approx T(\mathcal{A}) + Q\cdot \mathsf{poly}(\lambda)$, with $\mathsf{poly}(\lambda)$ independent of $T(\mathcal{A})$, and s.t.

$$\mathbf{Adv}^{Q-\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{B}}(\lambda) + \tfrac{1}{p-1}.$$

Here $\mathbf{Adv}^{Q-\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_s,\mathcal{A}}(\lambda) := |\Pr[\mathcal{A}(\mathsf{BG},[\mathbf{A}]_s,[\mathbf{AW}]_s)=1] - \Pr[\mathcal{A}(\mathsf{BG},[\mathbf{A}]_s,[\mathbf{U}]_s)=1]$, where the probability is taken over $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\lambda)$, $\mathbf{A} \leftarrow \mathcal{D}_k$, $\mathbf{W} \leftarrow \mathbb{Z}_p^{k\times Q}$ and $\mathbf{U} \leftarrow \mathbb{Z}_p^{(k+1)\times Q}$.

The $\mathcal{D}_k$-l-$\mathsf{extKerMDH}$ assumption allows an adversary to extend the given matrix but requires it to output multiple, linearly independent vectors in the kernel. An explanation on why it is a natural extension of the $\mathcal{D}_k$-KerMDH assumption can be found in [CH20] (Section 4.2). Furthermore, languages used in this thesis are *trapdoor-reducible*, which is a stronger notion of witness samplable (Section 4.1 from [CH20]). Hence, we present the falsifiable variant of this assumption.

$\mathcal{D}_k$-l-$\mathsf{extKerMDH}$ Assumption [CH20]

Let $\mathcal{D}_k$ be a matrix distribution, $l, k \in \mathbb{N}$, and $s \in \{1,2\}$. We say that the $\mathcal{D}_k$-l-$\mathsf{extKerMDH}$ *assumption* holds in $\mathbb{G}_s$ relative to BGGen, if for every $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$, $\mathbf{D} \xleftarrow{\$} \mathcal{D}_k$, and all p.p.t. adversaries $\mathcal{A}$ the following probability is negligible.

$$\Pr\left[\begin{array}{c|c} [\mathbf{C}]_{3-s} \in \mathbb{G}_3^{(l+1)\times(k+l+1)} \wedge [\mathbf{B}]_s \in \mathbb{G}_s^{l\times k} & \mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda); \mathbf{D} \xleftarrow{\$} \mathcal{D}_k \\ \wedge\ [\mathbf{C}]_{3-s}[\mathbf{D}']_s = 0 & ([\mathbf{C}]_{3-s}, [\mathbf{B}]_s) \xleftarrow{\$} \mathcal{A}(\mathsf{BG}, [\mathbf{D}]_s) \\ \wedge\ \mathsf{rank}(\mathbf{C}) \geq l+1 & [\mathbf{D}']_s := \left[\begin{smallmatrix}\mathbf{D}\\ \mathbf{B}\end{smallmatrix}\right]_s \end{array}\right]$$

2.4 Cryptographic Primitives

2.4.1 One-way Functions

One of the most fundamental problems in cryptography is finding *easy* tasks in a given context which are *hard* in others. One-way functions represent very well that idea and serve as the main building block for a variety of cryptographic primitives. Informally speaking, we say that a function is *one-way* if it is easy to compute and hard to invert (*i.e.,* finding any valid preimage of a random image). In other words, let f be a function s.t. $f : \{0,1\}^* \to \{0,1\}^*$. We say that f is a one-way function if the task of computing $f(x)$ is easy given x but the task of computing x is hard given $f(x)$ for an x picked randomly.

The problem with one-way functions is that there are no known constructions. In fact, the existence of a one-way function would prove that $\mathcal{P} \neq \mathcal{NP}$, which is the most fundamental open problem in computer science.

For the above reason, one-way functions are often seen as an assumption themselves or one works with *candidate* one-way functions provided some other assumption holds. For example, in the next section, we present the Pedersen commitment scheme [Ped92], whose one-wayness relies on the DL assumption.

2.4.2 Cryptographic Hash Functions

An *unkeyed* cryptographic hash function $\mathcal{H}_\kappa : \{0,1\}^* \to \{0,1\}^\kappa$ is a deterministic polynomial-time algorithm mapping an arbitrary-length message to a bitstring of fixed size κ, the *hash value* or *message digest*. They are assumed to be *collision resistant* [Dam88], which means that it should be computationally infeasible to find two values $x, y \in \{0,1\}^*$ such that $\mathcal{H}_\kappa(x) = \mathcal{H}_\kappa(y)$. In practice, cryptographic hash functions are *unkeyed* and thus they do not satisfy the theoretical definition of collision resistance [Rog06].

Two other security properties are also desirable for cryptographic hash functions. These are *preimage resistance*, which requires $\mathcal{H}_\kappa$ to behave as a one-way function, and *second-preimage resistance*, which is implied by collision resistance and requires that given x it should be infeasible to find x' such that $\mathcal{H}_\kappa(x) = \mathcal{H}_\kappa(x')$.

2.4.3 Commitment Schemes

Commitment schemes are a fundamental building block in modern cryptography. Intuitively they allow a party to commit to a value that stays secret until certain conditions are met. Below we introduce the formal definition and security properties of this primitive.

Definition 1: Commitment Scheme

A non-interactive commitment scheme $\Gamma = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ on a message space $\mathcal{M}$ is a tuple such that:

$\mathsf{Setup}(1^\lambda)$ is a p.p.t algorithm that, given the security parameter λ, outputs public parameters pp.

$\mathsf{Commit}(\mathsf{pp}, m)$ is a p.p.t algorithm that, given pp and a message m, outputs a commitment/opening pair (c, d) for m.

$\mathsf{Verify}(\mathsf{pp}, c, d, m)$ is a deterministic algorithm that, given pp, a commitment and opening pair (c, d) and a message m, outputs true iff m is a valid opening of c according to d.

The commitment scheme correctness requires that for every message m and every security parameter λ: $\Pr\left[\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) : \mathsf{Verify}(\mathsf{pp}, \mathsf{Commit}(\mathsf{pp}, m), m) = 1\right] = 1$. Two security properties are required for commitment schemes: binding and hiding. Binding states that it should be infeasible for any party to come up with an opening that would reveal a different value than the one initially committed. Hiding states that it should be infeasible for any party to reveal a committed message without the corresponding opening. If a scheme is perfectly binding, it can only be computationally hiding or the other way round.

Definition 2: Hiding and Binding

A commitment scheme Γ has the hiding security property if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{Hiding}}_{\Gamma,\mathcal{A}}(\lambda) := 2\cdot \Pr\left[\mathbf{Exp}^{\text{Hiding}}_{\Gamma,\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{Hiding}}_{\Gamma,\mathcal{A}}(\lambda)$ is the experiment shown at the bottom (left side).
A commitment scheme Γ has the binding security property if the advantage of any p.p.t algorithm $\mathcal{A}$ defined by

$$\mathbf{Adv}^{\text{Binding}}_{\Gamma,\mathcal{A}}(\lambda) := \Pr\left[\mathbf{Exp}^{\text{Binding}}_{\Gamma,\mathcal{A}}(\lambda) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{Binding}}_{\Gamma,\mathcal{A}}(\lambda)$ is the experiment shown at the bottom (right side).

```
$\mathbf{Exp}^{\text{Hiding}}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_1(\mathsf{pp})$
$b \xleftarrow{\$} \{0,1\}$
$(c, d) \leftarrow \mathsf{Commit}(\mathsf{pp}, m_b)$
$b' \leftarrow \mathcal{A}_2(c, \mathsf{st})$
return $b = b'$
```

```
$\mathbf{Exp}^{\text{Binding}}_{\Gamma,\mathcal{A}}(\lambda)$
$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$(c, d, m, d', m') \leftarrow \mathcal{A}_1(\mathsf{pp})$
if $m \neq m'$ then
    return $\mathsf{Verify}(\mathsf{pp}, c, d, m) \wedge \mathsf{Verify}(\mathsf{pp}, c, d', m')$
else return $0$
```

The Pedersen commitment [Ped92] is perfectly hiding and computationally binding under the DL assumption in $\mathbb{G}$. It consists of the following algorithms:

$\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, generates a group $\mathbb{G}$ of prime order p with $\lceil \log_2 p \rceil = \lambda$, picks a generator P_1 and $\tau \xleftarrow{\$} \mathbb{Z}_p^*$. It outputs $\mathsf{pp} = (\mathbb{G}, p, P_1, H = \tau P_1)$ and trapdoor information τ.

$\mathsf{Commit}(\mathsf{pp}, m)$: On input pp and $m \in \mathbb{Z}_p$ outputs $(c, d) \leftarrow (mP_1 + rH, r)$.

$\mathsf{Verify}(\mathsf{pp}, c, d, m)$: On input c, d and m outputs true iff $mP_1 + dH = c$.

We also use a slightly different formalization for commitment schemes. Instead of defining the algorithm Verify, one can define the algorithm Open that outputs a message m instead of a truth value as Verify does. For example, if we consider the Pedersen commitment, one can define the opening d as (r, m) so that the algorithm Open takes $c, (r, m)$ as input and outputs m if $mP_1 + dH = c$ or $\perp$ otherwise. Another difference with respect to the literature depending on the application, is the use of public parameters. One can also define a commitment scheme with evaluation and verification keys.

2.4.4 Public-Key Encryption

As mentioned in the introduction, PKE revolutionised the field of cryptography. To introduce this concept, we opt to quote the following extract from Whitfield Diffie's work "Cryptography, the Next Two Decades" [Dif81], presented during the first CRYPTO conference in 1981:

A public key cryptosystem is one in which the conversion of plaintext to ciphertext and the conversion of ciphertext to plaintext are done using different keys. Furthermore, given one of the keys, it is just as difficult to discover the other as it would be to discover the plaintext given only a sample of the ciphertext. This separation of the keys for encrypting and decrypting makes it possible to disclose one (the public key) while retaining the other (the secret key.)

— Whitfield Diffie

In this section, we present the formal definition and security notions for public-key encryption schemes that are used in this thesis. By convention, we denote the set of the plaintexts, public keys, random coins and ciphertexts by $\mathcal{M}$, $\mathcal{K}$, $\mathcal{R}$ and $\mathcal{C}$, respectively.

Definition 3: Public-key encryption scheme

A PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is a triple of (possibly randomised) algorithms such that:

$\mathsf{KGen}(1^k)$ is a p.p.t algorithm that, given the security parameter k, outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}(\mathsf{pk}, m; r)$ is a p.p.t algorithm that, given a message m, random coins r and pk, outputs a ciphertext c.

$\mathsf{Dec}(\mathsf{sk}, c)$ is a deterministic algorithm that, given a ciphertext c and sk, outputs a message m or $\perp$.

Correctness of public-key encryption requires that for every message m and every security parameter λ: $\Pr\left[(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^{\lambda}) : \mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m)) = m\right] = 1$.

Security is usually defined in terms of ciphertext indistinguishability. The most basic variant of ciphertext indistinguishability is known as *indistinguishability of ciphertexts under chosen-plaintext attacks*. This notion is equivalent to the original definition of *semantic security* [GM82], which is the computational analogue of perfect secrecy (*i.e.,* information-theoretic security) [Sha49]. The main idea is that an attacker with access to the ciphertext should not get a considerable advantage over someone without access to it when identifying the related plaintext.

Definition 4: IND-CPA

A PKE scheme $\Pi = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ has indistinguishable ciphertexts under chosen-plaintext attacks (IND-CPA), if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\left[\mathbf{Exp}^{\text{IND-CPA}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IND-CPA}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
$\mathbf{Exp}^{\text{IND-CPA}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(m_0,m_1,\mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk},m_b)$
$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{pk},c,\mathsf{st})$
**return** $b = b'$
```

The above definition rules out the "possibly randomised" option for the encryption algorithm when defining a PKE scheme. To be secure under a minimal notion such as IND-CPA, the scheme must have a randomised encryption algorithm.

Stronger notions allow the adversary to query a decryption oracle for arbitrary ciphertexts. We have two cases, non-adaptive and adaptive. The former case allows the adversary to send queries only before receiving the challenger's ciphertext. The latter allows it to send queries before and after (as long as it does not query the challenger's ciphertext). Below we present the two formalizations.

Definition 5: IND-CCA1

A PKE scheme $\Pi = (\mathsf{Gen},\mathsf{Enc},\mathsf{Dec})$ has indistinguishable ciphertexts under non-adaptive chosen-ciphertext attacks (IND-CCA1) if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1,\mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-CCA1}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\left[\mathbf{Exp}^{\text{IND-CCA1}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IND-CCA1}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
$\mathbf{Exp}^{\text{IND-CCA1}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{pk}, c, \mathsf{st})$
return $b = b'$
```

Definition 6: IND-CCA2

A PKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable ciphertexts under adaptive chosen-ciphertext attacks (IND-CCA2), if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\Big[\mathbf{Exp}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\Big] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
$\mathbf{Exp}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk}, c, \mathsf{st})$
return $b = b'$
```

One implication of giving access to a decryption oracle is that ciphertexts outputted by an IND-CCA2 secure scheme are *non-malleable* [BDPR98]. If this was the case, the adversary could call the decryption oracle on c' to obtain a message m' (related to m_0 or m_1) and trivially win the experiment.

Prabhakaran and Rosulek proposed a relaxation of the IND-CCA2 notion called *indistinguishability of ciphertexts under replayable chosen-ciphertext attacks* (IND-RCCA). Their idea was to capture encryption schemes that are IND-CCA2 secure "except that they allow anyone to generate new ciphertexts that decrypt to the same value as a given ciphertext" [PR07].

Definition 7: IND-RCCA [PR07]

A PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable ciphertexts under replayable chosen-ciphertext attacks (IND-RCCA), if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-RCCA}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\Big[\mathbf{Exp}^{\text{IND-RCCA}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\Big] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IND-RCCA}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
Experiment $\mathbf{Exp}^{\text{IND-RCCA}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk},\mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk},\cdot)}(\mathsf{pk})$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{GDec}(\mathsf{sk},\cdot)}(\mathsf{pk}, c, \mathsf{st})$
return $b = b'$

Oracle $\mathsf{GDec}(\mathsf{sk}, c)$
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
if $m \in \{m_0, m_1\}$
 then return replay
else return $m$
```

In the experiment $\mathbf{Exp}^{\text{IND-RCCA}}_{\Pi,\mathcal{A}}(k)$, the adversary can only access a *guarded decryption oracle* (GDec) once the challenge ciphertext has been received. When given any ciphertext that decrypts either to m_0 or m_1, this oracle returns a special message replay (meaning the ciphertext is a rerandomised ciphertext of one of the challenge messages).

Bellare *et al.* [BBDP01] studied another security requirement for public-key encryption schemes named *key-privacy.* As the authors explain, it asks whether or not an attacker in possession of a ciphertext is able to tell which specific key, out of a set of known public keys, is the one under which the ciphertext was created. Furthermore, the authors show that the notion of key-privacy is orthogonal to that of data-privacy (captured by the indistinguishability of ciphertexts). The following definitions from [BBDP01] formalize the notions of indistinguishability of keys under chosen-plaintext attack and under chosen-ciphertext attack.

Definition 8: IK-CPA

A PKE scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable keys under chosen-plaintext attacks (IK-CPA), if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\left[\mathbf{Exp}^{\text{IK-CPA}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IK-CPA}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
$\mathbf{Exp}^{\text{IK-CPA}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk}_0,\mathsf{sk}_0) \xleftarrow{\$} \mathsf{KGen}(1^k)$;$(\mathsf{pk}_1,\mathsf{sk}_1) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(m, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk}_0, \mathsf{pk}_1)$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_b, m)$
$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{pk}_0, \mathsf{pk}_1, c, \mathsf{st})$
return $b = b'$
```

Definition 9: IK-CCA

A public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has indistinguishable keys under chosen-ciphertext attacks (IK-CCA), if the advantage of any p.p.t algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ defined by

$$\mathbf{Adv}^{\text{IND-CCA2}}_{\Pi,\mathcal{A}}(k) := 2\cdot \Pr\left[\mathbf{Exp}^{\text{IK-CCA}}_{\Pi,\mathcal{A}}(k) \Rightarrow \mathsf{true}\right] - 1 \text{ is negligible,}$$

where $\mathbf{Exp}^{\text{IK-CCA}}_{\Pi,\mathcal{A}}(k)$ is the experiment shown below.

```
$\mathbf{Exp}^{\text{IK-CCA}}_{\Pi,\mathcal{A}}(k)$
$(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KGen}(1^k)$; $(\mathsf{pk}_1, \mathsf{sk}_1) \xleftarrow{\$} \mathsf{KGen}(1^k)$
$(m, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{Dec}(\mathsf{sk}_0,\cdot),\mathsf{Dec}(\mathsf{sk}_1,\cdot)}(\mathsf{pk}_0, \mathsf{pk}_1)$
$b \xleftarrow{\$} \{0,1\}$; $c \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_b, m)$
$b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{Dec}(\mathsf{sk}_0,\cdot),\mathsf{Dec}(\mathsf{sk}_1,\cdot)}(\mathsf{pk}_0, \mathsf{pk}_1, c, \mathsf{st})$
return $b = b'$
```

We now present ElGamal's encryption scheme [ElG84], which is IND-CPA secure. Consider a group $\mathbb{G}$ with generator P. The key generation algorithm picks $a \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $\mathsf{sk} = a$ and $\mathsf{pk} = aP$. Then, to encrypt a message m, the encryption algorithm picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $(rP, m + r\mathsf{pk})$. Finally, the decryption algorithm takes a ciphertext (C_0, C_1) and computes $C_0 - \mathsf{sk}C_1$.

ElGamal is also IK-CPA (although it may not generally be the case). Furthermore, ElGamal's ciphertexts can be decrypted with knowledge of the randomness used to generate the ciphertexts. For example, given a ciphertext $(C_0, C_1) = (rP, m + r\mathsf{pk})$, knowing r reveals m as it suffices to compute $C_1 - r\mathsf{pk}$ to obtain m. The following definition from [BL16] formalizes this idea.

Definition 10: Random Coin Decryptable PKE (RCD-PKE)

A probabilistic PKE scheme is Random Coin Decryptable if there exists a polynomial-time algorithm CDec such that for any public key $\mathsf{pk} \in \mathcal{K}$, any $m \in \mathcal{M}$, and any random coins r, the following equation holds: $\mathsf{CDec}(\mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, m; r), r) = m$.

2.4.5 Digital Signatures

Definition 11: Digital signature scheme

A digital signature scheme is a tuple $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ of algorithms where:

$\mathsf{KGen}(1^\lambda)$ is a p.p.t algorithm that, given a security parameter λ, outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$ is a p.p.t algorithm that, given a secret key sk and a message m, outputs a signature σ on the message m.

$\mathsf{Verify}(m, \sigma, \mathsf{pk})$ is a deterministic algorithm that takes as input a public key pk, a message m and a signature σ. If σ is a valid signature on m it outputs 1 and 0 otherwise.

Correctness of digital signatures requires that for every message m and every security parameter λ: $\Pr\left[(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda) : \mathsf{Verify}(m, \mathsf{Sign}(\mathsf{sk}, m), \mathsf{pk}) = 1\right] = 1$. Security is usually defined in terms of existential unforgeability. The latter concept captures the three fundamental properties: authentication, integrity and non-repudiation. It specifies that it would suffice for an adversary to come up with a single message-signature pair that verifies (for a message that has never been signed by the legitimate signer) to dismiss the scheme's security.

Below we give a definition for the most common variant which is *Existential UnForgeability under adaptive Chosen-Message Attacks* (EUF-CMA) [GMR88].

Definition 12: EUF-CMA

A digital signature scheme $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ is existentially unforgeable under adaptively chosen-message attacks if, for all p.p.t adversaries $\mathcal{A}$ with access to a signing oracle Sign, the following probability is negligible,

$$\Pr\left[\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}) \\ (m^*,\sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})\end{array} : m^* \notin Q \ \wedge \ \mathsf{Verify}(m^*,\sigma^*,\mathsf{pk}) = 1\right],$$

where Q is the set of queries that $\mathcal{A}$ has issued to the signing oracle.

2.4.6 Zero-Knowledge Proofs

For interactive machines $\mathcal{P}$ (the prover) and $\mathcal{V}$ (the verifier), we denote as in [Pas04] that $\langle \mathcal{P}(w), \mathcal{V}(z)\rangle(x)$ is the random variable representing $\mathcal{V}$'s output when interacting with $\mathcal{P}$ on common input x, when the random input to each machine is uniformly and independently chosen; with w and z being auxiliary inputs. Similarly, as in [FHS19], we denote formal languages defined by some binary polynomial-time (witness) relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ as $L_{\mathcal{R}} = \{x \mid \exists w : (x,w) \in \mathcal{R}\} \subseteq \{0,1\}^*$. For such types of relations, membership of $x \in L_{\mathcal{R}}$ can be decided in polynomial-time (in $|x|$) when given a witness w of length polynomial in $|x|$ certifying $(x,w) \in \mathcal{R}$. In other words, languages of the form $L_{\mathcal{R}}$ belong to the complexity class $\mathcal{NP}$.

With the above in mind, we introduce now the definition for *interactive proof systems* following the formalisms from [Pas04] and [GT20].

Definition 13: Interactive Proof System

Let ϵ_c, ϵ_s: $\mathbb{N} \to [0,1)$ such that both are computable in polynomial-time in $|\lambda|$ and $\epsilon_c(|\lambda|) + \epsilon_s(|\lambda|) < 1 - 1/\mathsf{poly}(|\lambda|)$. $(\mathcal{P},\mathcal{V})$ is called an *interactive proof system* for the language $L_{\mathcal{R}}$ with completeness and soundness errors ϵ_c and ϵ_s, respectively, if $\mathcal{V}$ is p.p.t and the following conditions hold:

- Completeness: For every $x \in L_{\mathcal{R}}$ there exists a (witness) string w such that for every auxiliary input $z \in \{0,1\}^*$:
$$\Pr[\langle \mathcal{P}(w), \mathcal{V}(z)\rangle(x) = 1] = 1 - \epsilon_c(|x|)$$
- Soundness: For every $x \notin L_{\mathcal{R}}$, every interactive machine $\mathcal{P}^*$, and every $w, z \in \{0,1\}^*$:
$$\Pr[\langle \mathcal{P}^*(w), \mathcal{V}(z)\rangle(x) = 1] \leq \epsilon_s(|x|)$$

If $\epsilon_c = 0$, we say the system has *perfect completeness*. If the soundness condition is required to hold only with respect to a computationally bounded prover $\mathcal{P}^*$, $(\mathcal{P},\mathcal{V})$ is called an interactive *argument* system.

In 1985, Goldwasser, Micali and Rackoff introduced the notion of zero-knowledge proofs in their seminal work "The Knowledge Complexity of Interactive Proof-Systems" [GMR85].

Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question.
— Goldwasser, Micali & Rackoff

The key observation elaborated by the authors was to equate the idea of "no additional knowledge" with that of specifying a way in which a (dishonest) verifier could interact with a party (who is not a real prover) in the same way as it would interact with an (honest) prover. Such a party is captured by an algorithm called *the simulator*. By "same way", we mean that from the verifier's point of view, a transcript produced by the simulator looks indistinguishable from the one that would result from a real interaction with the prover. Furthermore, the notion considered here is a *black-box* one in the sense that the simulator has black-box access to such a verifier. It is also important to stress that the simulator's goal is to generate a *complete* transcript but without access to the witness. Hence, the simulator is often given extra power, such as the ability to run in expected polynomial time rather than strict polynomial time or access to some auxiliary input compared to the prover.

The main conclusion from the above is the fact that if one succeeds in defining a simulator for an interactive proof system, the indistinguishability of two transcripts implies that whatever a malicious verifier could have learnt interacting with a real prover is no different to what it could have learnt without interacting with one. Since the latter interaction does not involve the witness at all, the interactive proof system is zero-knowledge. In the following, we formalize this notion.

Definition 14: Zero-Knowledge

An interactive proof system $(\mathcal{P}, \mathcal{V})$ is *zero-knowledge* if for every p.p.t interactive machine $\mathcal{V}^*$ there exists a probabilistic expected polynomial-time algorithm $\mathcal{S}$ (called simulator) such that the following two ensembles are computationally indistinguishable (when the distinguishing gap is a function in $|x|$): $\{\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)\}_{z\in\{0,1\}^*, x\in L_\mathcal{R}}$ for an arbitrary w s.t. $(x,w)\in\mathcal{R}$ and $\{\mathcal{S}(x,z)\}_{z\in\{0,1\}^*, x\in L_\mathcal{R}}$. That is, for every probabilistic algorithm $\mathcal{D}$ running in time polynomial in the length of its first input, all $(x,w)\in\mathcal{R}$ and all auxiliary inputs $z,z'\in\{0,1\}^*$ it holds that:

$$|\Pr[\mathcal{D}(x,z',\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)) = 1] - \Pr[D(x,z',\mathcal{S}(x,z)) = 1]| < 1/\mathsf{poly}(|x|)$$

Let us now recap on what the three properties of interactive proof systems intuitively mean. First, completeness states that things work as expected (*i.e.,* the proof system works for valid statements). Soundness protects honest verifiers against malicious provers (*i.e.,* the proof system does not work for invalid statements). Finally, zero-knowledge protects honest provers against malicious verifiers (*i.e.,* the proof system is good at keeping secrets).

Sometimes, we use the term "perfect zero-knowledge" to refer to proof systems where the two ensembles are *identically* distributed. We will also refer to a weaker variant called *Honest Verifier Zero-Knowledge* (HVZK). Only a single verifier

$\mathcal{V} = \mathcal{V}^*$ that always follows the protocol is considered in this variant.

We have already introduced the definitions for interactive proof systems and zero-knowledge proof systems but without taking much time to discuss *interaction*. Efficiency and adequacy of a proof system to model a particular problem depends on the interactions between the prover and the verifier. Interaction is measured in terms of communication rounds (number of messages exchanged) and communication bandwidth (size of messages exchanged). In the following, we introduce a particular type of zero-knowledge proof system that is called a *sigma protocol*.

Definition 15: Sigma protocol

An interactive proof system $(\mathcal{P}, \mathcal{V})$ is said to be a *sigma protocol* for a relation $\mathcal{R}$ when it uses the following pattern: $\mathcal{P}$ sends a commitment C, $\mathcal{V}$ sends a challenge b, $\mathcal{P}$ sends a response r, after which $\mathcal{V}$ accepts or rejects the proof, and the following requirement holds:

- Special soundness: There exists a polynomial-time algorithm $\mathcal{E}$ that given any x and any pair of accepting transcripts $(t, t') = ((C, b, r), (C, b', r'))$ for x such that $b \neq b'$:
$$\Pr[w \leftarrow \mathcal{E}(x, t, t') : (x, w) \in \mathcal{R}] \text{ is overwhelming.}$$

Sigma protocols usually use the notion of *Special Honest Verifier Zero-Knowledge* (SHVZK). This notion is a particular case of HVZK for which, given a valid statement x and fixed challenge b, the simulator needs to produce an accepting transcript (C, b, r) that has the same distribution as a real one.

The following lemmas about sigma protocols are used in Chapter 3.

Lemma 4 (Lemma 6.2.6 from [HL10])**.** The properties of sigma protocols are invariant under parallel repetition. That is, the ℓ-times parallel repetition of a sigma protocol for $\mathcal{R}$ with challenge length t yields a new sigma protocol for $\mathcal{R}$ with challenge length $\ell \cdot t$.

Lemma 5 (Lemma 6.2.7 from [HL10])**.** If there exists a sigma protocol Π for $\mathcal{R}$ with challenge length t, then there exists a sigma protocol Π' for $\mathcal{R}$ with challenge length t', for any $t' > t$.

So far, we have only considered zero-knowledge proofs with the idea that not even a malicious verifier could learn anything else besides the validity of a statement. While this is quite a strong notion already, one can go a step further and wonder about the relation between the prover and the witness. For some zero-knowledge proofs, it is possible to conclude that the prover *must* know the actual witness for the statement in question. This is a stronger soundness notion, often called *knowledge soundness*, because it not only gives assurance to the verifier about the validity of the statement but also about the relation between the prover and the witness (*i.e.,* the prover knows it). This latter notion is formalised by defining a machine called *the extractor* that, given access to a (potentially malicious) prover $\mathcal{P}^*$, is able to compute the witness w. Similarly to the simulator, the extractor is usually given some extra power like the ability to rewind the prover. Zero-knowldge proofs with this property are called *Zero Knowledge Proof of Knowledge* (ZKPoK). The following theorem relates sigma protocols with proofs of knowledge.

Theorem 6 (Theorem 6.3.2 from [HL10])**.** Let Π be a sigma protocol for a relation $\mathcal{R}$ with challenge length t. Then Π is a proof of knowledge with knowledge error (*i.e.,* the probability that the extractor fails) 2^{-t}.

In [Lin01], Lindell extends the definition of special soundness to proofs of knowledge that are not sigma protocols. We now recall it using the formalism introduced in [BCC+16], where t is a transcript of the protocol execution and s represents the state of $\mathcal{P}^*$, including its random tape.

Definition 16: Statistical Witness-Extended Emulation [BCC+16]

An interactive proof system $(\mathcal{P}, \mathcal{V})$ has *statistical witness-extended emulation* if, for all deterministic polynomial-time $\mathcal{P}^*$, there exists an expected polynomial-time emulator $\mathcal{E}$ such that for all interactive adversaries $\mathcal{A}$:

$$\Pr\left[\begin{array}{l}(x,s) \leftarrow \mathcal{A}(1^\lambda)\\ t \leftarrow \langle \mathcal{P}^*(x,s), \mathcal{V}(x)\rangle :\\ 1 \leftarrow \mathcal{A}(t)\end{array}\right] \approx \Pr\left[\begin{array}{l}(x,s) \leftarrow \mathcal{A}(1^\lambda)\\ (t,w) \leftarrow \mathcal{E}^{\langle \mathcal{P}^*(x,s), \mathcal{V}(x)\rangle}(x) :\\ 1 \leftarrow \mathcal{A}(t) \text{ and if t is accepting}\\ \text{then } (x,w) \in \mathcal{R}\end{array}\right]$$

where the oracle called by $\mathcal{E}$ permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards.

2.4.7 NIZK and Malleable Proof Systems

Non-Interactive Zero-Knowledge proofs (or NIZKs for short) are proof systems for which a single message from the prover to the verifier is exchanged. Upon receiving a message from the prover, the verifier either accepts or rejects the proof. In the following, we present the first type of NIZKs, used in Chapter 3 and 6. It is a generic and heuristic method to transform a sigma protocol into a NIZK, assuming the existence of a hash function that behaves like a random oracle. We follow the formalization from [BPW12], always using the *strong* variant of the transform.

Definition 17: Fiat-Shamir Transform [FS87]

Let $\Sigma = (\mathsf{Prove}_\Sigma, \mathsf{Verify}_\Sigma)$ be a sigma protocol and $\mathcal{H}$ a cryptographic hash function. The *strong Fiat-Shamir transform* of Σ is the proof system $\mathsf{sFS}_\mathcal{H}(\Sigma) = (\mathsf{Prove}, \mathsf{Verify})$ defined as follows:

$\mathsf{Prove}(w,x)$ runs $\mathsf{Prove}_\Sigma(w,x)$ to obtain a commitment C. Computes $b \leftarrow \mathcal{H}(x,C)$ and completes the run of $\mathsf{Prove}_\Sigma(w,x)$ with b as input to get a response r. It outputs the pair (b,r).

$\mathsf{Verify}(x,b,r)$ computes C from (x,b,r) and outputs $\mathsf{Verify}_\Sigma(x,C,b,r)$.

Remark 1. While sigma protocols are only HVZK, using the Fiat-Shamir transform results in a NIZK that has full zero-knowledge in the ROM.

The second type of NIZKs presented in this section requires a common reference string (crs) instead of a hash function. Below we introduce the corresponding syntax and security properties for this type of NIZKs considering fully adaptive NIZK arguments (*i.e.,* the crs does not depend on the language distribution or language parameters).

A fully adaptive NIZK Π for a family of language distribution $\{\mathcal{D}_{\mathsf{pp}}\}_{\mathsf{pp}}$ consists of four probabilistic algorithms:

$\mathsf{PGen}(1^\lambda)$: is a p.p.t algorithm that, given a security parameter λ, outputs public parameters pp and a common reference string crs. For simplicity, public parameters will be assumed to be implicitly included in the crs.

$\mathsf{PTGen}(1^\lambda)$: is like PGen but it also returns a trapdoor τ (if any).

$\mathsf{Prove}(\mathsf{crs}, \rho, x, w)$: is a p.p.t algorithm that, given crs, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$ and a statement x with witness w, outputs a proof π for $x \in \mathcal{L}_\rho$.

$\mathsf{Verify}(\mathsf{crs}, \rho, x, \pi)$: is a deterministic algorithm that, given crs, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$, a statement x and a proof π, accepts or rejects the proof.

$\mathsf{PSim}(\mathsf{crs}, \tau, \rho, x)$: is a p.p.t algorithm that, given crs, τ, a language description $\rho \in \mathcal{D}_{\mathsf{pp}}$ and a statement x, outputs a simulated proof for the statement $x \in \mathcal{L}_\rho$.

The following properties need to hold for NIZK arguments with respect to a family of language distributions $\mathcal{D}_{\mathsf{pp}}$.

Perfect Completeness:

$$\Pr\left[\ \mathsf{Verify}(\mathsf{crs}, \rho, x, \pi) = 1 \ \middle|\ \begin{array}{c} (\mathsf{pp}, \mathsf{crs}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda); \rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}}); \\ (x, w) \in R_\rho; \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, \rho, x, w) \end{array}\right] = 1$$

Computational Soundness: For every efficient adversary $\mathcal{A}$,

$$\Pr\left[\ \begin{array}{c} \mathsf{Verify}(\mathsf{crs}, \rho, x, \pi) = 1 \\ \wedge\ x \notin \mathcal{L}_\rho \end{array} \ \middle|\ \begin{array}{c} (\mathsf{pp}, \mathsf{crs}) \xleftarrow{\$} \mathsf{PGen}(1^\lambda); \\ \rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}}); (\pi, x) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \rho) \end{array}\right] \approx 0$$

where the probability is taken over PGen.

Perfect Zero-Knowledge: For all λ, all $(\mathsf{pp}, \mathsf{crs}, \tau) \in \mathsf{Supp}(\mathsf{PTGen}(1^\lambda))$, all $\rho \in \mathsf{Supp}(\mathcal{D}_{\mathsf{pp}})$ and all $(x, w) \in R_\rho$, the distributions $\mathsf{Prove}(\mathsf{crs}, \rho, x, w)$ and $\mathsf{PSim}(\mathsf{crs}, \tau, \rho, x)$ are identical.

We conclude the present section by introducing the notions of malleable proof systems given in [CKLM12] and [KSD19], respectively.

Let $\mathcal{R}_\mathcal{L}$ be the witness relation associated to a language $\mathcal{L}$, then a controlled malleable proof system is accompanied by a family of efficiently computable n-ary transformations $T = (T_x, T_w)$ such that for any n-tuple $\{(x_1, w_1), \ldots, (x_n, w_n)\} \in \mathcal{R}_\mathcal{L}^n$ it holds that $(T_x(x_1, \ldots, x_n), T_w(w_1, \ldots, w_n)) \in \mathcal{R}_\mathcal{L}$. Intuitively, such a proof system allows when given valid proofs $\{\Omega_i\}_{i\in[n]}$ for statements $\{x_i\}_{i\in[n]}$ with associated witnesses $\{w_i\}_{i\in[n]}$ to publicly compute a valid proof Ω for word $x = T_x(x_1, \ldots, x_n)$ corresponding to witness $w := T_w(w_1, \ldots, w_n)$ using an additional algorithm ZKEval which is defined as follows:

$\mathsf{ZKEval}(\mathsf{crs}, \mathcal{T}, (x_i, \Omega_i)_{i\in[n]})$: takes as input a common reference string crs, a transformation $T \in \mathcal{T}$, words $x_1, \ldots, x_n$ and their corresponding proofs $\Omega_1, \ldots, \Omega_n$, and outputs a new word $x' := T_x(x_1, \ldots, x_n)$ and proof Ω'.

Proofs computed by ZKEval should be indistinguishable from freshly computed proofs for the resulting word x' and corresponding witness w'. The following definition captures this notion.

Derivation Privacy: A NIZK proof system Π, malleable with respect to a set of transformations $\mathcal{T}$ defined on some relation $\mathcal{R}$ is *derivation private* if, for all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\Pr\left[\begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{PGen}(1^\lambda), b \xleftarrow{\$} \{0,1\} \\ (\mathsf{st}, ((x_i, w_i), \Omega_i)_{i \in [q]}, T) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}), \\ \textbf{if } (T \notin \mathcal{T} \vee (\exists\, i \in [q] : (\mathsf{Verify}(\mathsf{crs}, x_i, \Omega_i) = 0) \vee (x_i, w_i) \notin \mathcal{R}) \\ \textbf{return } \bot, \\ \textbf{else if } b = 0 : \Omega \leftarrow \mathsf{Prove}(\mathsf{crs}, T_x((x_i)_{i \in [q]}), T_w((w_i)_{i \in [q]})), \\ \textbf{else if } b = 1 : \Omega \leftarrow \mathsf{ZKEval}(\mathsf{crs}, T, (x_i, \pi_i)_{i \in [q]}), \\ b' \xleftarrow{\$} \mathcal{A}(\Omega, \mathsf{st}) \end{array} \quad : \quad b = b' \right]$$

3

Generic Plaintext Equality and Inequality Proofs

I know what I know, and I don't know anything else.

— Yehuda Lindell

Given two ciphertexts generated with a PKE scheme, the problem of plaintext equality consists in determining whether the ciphertexts encrypt the same message. Similarly, the problem of plaintext inequality consists in deciding whether they encrypt a different message. First, this chapter proposes several definitions to characterize randomisation in PKE. Then, based on those definitions, we aim to build protocols for plaintext equality and inequality. Therefore, we dwell on what it means to randomise plaintexts, ciphertexts, keys, and combinations thereof. Unlike prior approaches, our aim is not to focus on the particular properties of any concrete scheme. Consequently, we obtain generic definitions and protocols that can be instantiated with different PKE schemes.

This chapter relies on joint work with Olivier Blazy, Xavier Bultel and Pascal Lafourcade. The presented material is based on [BBLPK21a].

3.1 Introduction

There are scenarios in which deciding equality can easily be done. For instance, if both ciphertexts were generated using the same key and the encryption scheme is deterministic or if access to a trusted third party, who knows the private key, is provided. However, practical scenarios where a prover needs to convince a verifier of the equality or inequality of plaintexts require stronger guarantees (*i.e.*, the verifier must learn no additional information than the yes or no answer).

Well-known examples include the use of such proofs in voting protocols [RS06, Rya08, DJN10], reputation systems [HBBS13, DM14] and cloud-based applications [Rei18]. Additionally, protocols with broadcasting phases where one of the parties needs to broadcast encrypted messages under different keys to several parties can also benefit from these proofs. A less common example involves a client who needs to regularly store encrypted information in a backup server (or in a distributed database such as blockchain) while being able to convince any third party of minimal claims about it. In addition, non-interactive variants are also very useful when online interaction between the parties is undesirable or public verifiability is preferred.

Sometimes equality or inequality proofs are used as subroutines and must be integrated with other software. Therefore, flexible alternatives (*e.g.,* without relying on specific constructions requiring particular configurations or hardware) are essential to overcome possible conflicts when a use case changes. This is where generic protocols come in handy. As they can be instantiated with multiple schemes, integrating them into different settings is easier.

We focus on two-party protocols, where two ciphertexts and auxiliary inputs are given. The prover tries to convince a verifier about the ciphertexts' plaintext equality or inequality. The prover and the verifier share a common input consisting of a set of public keys and ciphertexts generated with those keys. The prover also knows the corresponding set of secret keys or the randomness used to encrypt the plaintexts. As previously mentioned, our aim is to design generic plaintext equality and inequality protocols in this setting.

3.2 Contributions

Using randomisation properties of PKE schemes, we build generic zero-knowledge protocols from standard techniques.

Our first contribution introduces different notions related to the concept of malleability in PKE. In particular, we make a clear distinction between how a ciphertext can be randomised (*e.g.* the ciphertext alone (Rand), the plaintext message (MsgRand), or the corresponding key (KeyRand)). As a result, we characterize PKE schemes in terms of generic randomisable encryption properties, which we use to build our protocols.

Our second contribution is the construction of two interactive zero-knowledge protocols, PEQ and PINEQ, for plaintext equality and inequality. These protocols are secure against malicious verifiers. However, for each of them, we first present a weaker variant (HPEQ and HPINEQ, respectively), which is only secure against honest verifiers. The protocol PEQ requires the PKE scheme to allow the randomisation of both, the ciphertext and the corresponding plaintext message. In contrast, the protocol PINEQ only requires the former one.

Our third contribution is plaintext equality protocols based on proofs of knowledge of the secret key (protocols MATCHPEQ and SIGPEQ) or of the randomness used for the encryption (protocol RSPEQ). Either case admits non-interactive versions applying the strong Fiat-Shamir transform (Definition 17 on page 23), but both require schemes with less common properties. For example, the schemes need to be key-randomisable or random coin decryptable (RCD, Definition 10 on page 19).

In Table 3.1, we list some well-known PKE schemes and their relationship

						Perfect ZK		ZKPoK		
Scheme	Security	RCD	Rand	MsgRand	KeyRand	PEQ	PINEQ	MATCHPEQ	SIGPEQ	RSPEQ
[ElG86]	IND-CPA	✓	✓	✓	✓	✓	✓	✓	✓	✓
[Pai99]	IND-CPA	✓	✓	✓		✓	✓	✓		✓
[GM82]	IND-CPA		✓	✓		✓	✓	✓		
[Dam92]	IND-CCA1	✓	✓	✓	✓	✓	✓	✓	✓	✓
[CS98]	IND-CCA1	✓	✓	✓		✓	✓			✓
[PR07]	RCCA	✓	✓				✓			

Table 3.1: PKE schemes and their properties.

with our definitions and protocols. We stress that although fully homomorphic schemes such as those based on lattices could also be used to instantiate our protocols, partial homomorphic properties presented in the scheme can be used to implement the different algorithms as well. Nonetheless, as shown with the scheme from [PR07], being partially homomorphic is not necessary.

3.3 Related Work

Jakobsson *et al.* [JJ00] introduced the concept of distributed plaintext equality test, which allows $n > 1$ parties to determine whether two ElGamal ciphertexts encrypt the same or different message without learning it, but it requires knowledge of the secret key and assumes at least one of the parties is honest. Very recently, McMurtry *et al.* [MPT20] showed that several follow-up works based on the plaintext equality test from [JJ00] are flawed (because they use it as if no trust assumptions where needed), and showed how to fix it.

Choi *et al.* [CEJ+07] proposed zero-knowledge equality/inequality proofs for boolean ElGamal ciphertexts with knowledge of the secret key. Their work requires the randomness used to produce the two ciphertexts. Parkes *et al.* [PRST06] proposed zero-knowledge equality/inequality proofs for Paillier ciphertexts given access to the randomness used to produce the ciphertexts or access to the plaintexts.

In [BCV15], Blazy *et al.* introduced a generic approach to prove a non-membership concerning some language in non-interactive zero-knowledge. For example, they showed how to prove the plaintext inequality of two ElGamal ciphertexts given that the prover knows the plaintext and the randomness used to produce one of the ciphertexts. More recently, Blazy et al. [BDSS16] introduced a generic technique for non-interactive zero-knowledge plaintext equality/inequality proofs in which the prover is given two ciphertexts and trapdoor information. In such a scenario, none of the parties has access to the secret key nor the randomness used to produce the ciphertexts. While being *generic*, those constructions [BCV15, BDSS16] require a specific kind of zero-knowledge proofs. More precisely, they need to build a zero-knowledge proof showing that a zero-knowledge proof was computed honestly. While this design works elegantly with pairing-based cryptography (as Groth-Sahai proofs [GS08] allows to prove in zero-knowledge a pairing-product equation while also being verifiable with a pairing product equation), this often fails (or requires *ad-hoc* constructions that are far from being efficient) in other settings. For example, when considering Schnorr [Sch91] proofs, the random oracle prevents any kind of chaining. Therefore, another design is required to allow such functionality.

Extensions for PKE schemes such that given a plaintext, a ciphertext and a public key, it is universally possible to check whether the ciphertext encrypts the plaintext under the key also exist. Such an extension has been proposed by Canard *et al.* [CFGL12] under the name of Plaintext Checkable Encryption.

There are also different works proposing schemes to support plaintext equality tests from user-specified authorization. For instance, in [Tan12], two users who have their keys can issue tokens to a proxy to authorize it to perform the plaintext equality test for their ciphertexts. Yang et al. [YTHW10] constructed a probabilistic scheme that allows anyone provided with two ciphertexts to check if they encrypt the same message, considering that the ciphertexts may not have been generated with the same key. They do this achieving a weak form of IND-CCA2.

Previous work rests on specific constructions, which do not allow the scheme to be separated from the protocol's requirements. Our approach is different because we first seek to define protocols independently of the scheme and then present the necessary conditions for a scheme to be compatible with them. As a result and unlike prior work, we present several protocols that can be integrated with existing pieces of software just as if they were templates allowing one to switch from one scheme to another more easily.

3.4 Generic Randomisable Encryption

This section proposes several definitions to characterize the kinds of randomisations that PKE schemes may support.

To begin with, we present a definition of *re-randomisability* [PR07], which has also been introduced under different names or variants ([CKN03, HS00, GJJS04, FFHR19]). However, unlike previous work, we include the notion of *derandomisability* and omit the distinction with universal re-randomisability from [PR07], which we consider implicit unless otherwise said.

Informally, a scheme that is randomisable and derandomisable supports the generation of fresh ciphertexts and the "rollback" process. Furthermore, we will say that a scheme achieves perfect randomisability when no adversary can distinguish between fresh encryptions of the original message and ciphertext randomisations.

Definition 18: Randomisable PKE scheme (Rand-PKE) **[PR07]**

A PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is *randomisable* if there exists a polynomial-time algorithm Rand such that:

1. Rand takes $c \in \mathcal{C}$, $r \in \mathcal{R}$ and returns $c' \in \mathcal{C}$.
2. $\forall\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, $r \in \mathcal{R}$ and $c \in \mathcal{C}$:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Rand}(c, r)) = \mathsf{Dec}(\mathsf{sk}, c)\right] = 1$$

Moreover, we say that Π is *derandomisable* if, for any $c \in \mathcal{C}$ and $r \in \mathcal{R}$, there exists an efficiently computable r^* such that:

$$\Pr\left[c = \mathsf{Rand}(\mathsf{Rand}(c, r), r^*)\right] = 1$$

Definition 19: Computational and perfect randomisability [PR07]

A $\mathsf{Rand\text{-}PKE}$ scheme is *computationally* randomisable if, for any λ, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and any polynomial-time distinguisher $\mathcal{D}$, there exists a negligible function $\epsilon(\cdot)$ such that:

$$\left| \Pr\left[\begin{array}{l} r' \xleftarrow{\$} \mathcal{R} \\ c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r') : b = 1 \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c') \end{array}\right] - \Pr\left[\begin{array}{l} r' \xleftarrow{\$} \mathcal{R} \\ c' \leftarrow \mathsf{Rand}(c, r') : b = 1 \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c') \end{array}\right] \right| \leq \epsilon(\lambda)$$

We say that the scheme is *perfectly* randomisable when $\epsilon(\lambda) = 0$.

We now introduce the following definitions that specify how the random coins used to produce fresh encryptions and randomisations can relate together. We will say that a PKE scheme is strongly randomisable when it also supports efficient algorithms to compute such relationships.

Definition 20: Strong Randomisable PKE scheme

A PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is *strongly randomisable* if it is a $\mathsf{Rand\text{-}PKE}$ and there exists a polynomial-time algorithm RandR such that:

1. RandR takes r and $r' \in \mathcal{R}$ and returns $r'' \in \mathcal{R}$.
2. $\forall$ $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$ and $r'' \leftarrow \mathsf{RandR}(r, r')$, the following equation holds:

$$\mathsf{Rand}(\mathsf{Enc}(\mathsf{pk}, m; r), r') = \mathsf{Enc}(\mathsf{pk}, m; r'')$$

Moreover, we say that Π is *random-extractable* if there exists a a polynomial-time algorithm $\mathsf{RandExt}$ such that for any $(r, r', r'') \in \mathcal{R}^3$:

$$\Pr\left[r = \mathsf{RandExt}(r', r'') : r'' \leftarrow \mathsf{RandR}(r, r')\right] = 1$$

Definition 21: Computational and perfect strong randomisability

A $\mathsf{Rand\text{-}PKE}$ scheme is *computationally* strongly randomisable if, for any λ, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and any polynomial-time distinguisher $\mathcal{D}$, there exists a negligible function $\epsilon(\cdot)$ such that:

$$\left| \Pr\left[\begin{array}{l} r' \xleftarrow{\$} \mathcal{R} \\ c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r') \\ b \leftarrow \mathcal{D}(\mathsf{pk}, r, c, r', c') \end{array} : b = 1\right] - \Pr\left[\begin{array}{l} r'' \xleftarrow{\$} \mathcal{R} \\ r' \leftarrow \mathsf{RandR}(r, r'') \\ c' \leftarrow \mathsf{Rand}(c, r'') \\ b \leftarrow \mathcal{D}(\mathsf{pk}, r, c, r', c') \end{array} : b = 1\right] \right| \leq \epsilon(\lambda)$$

We say that the scheme is *perfectly* strongly randomisable when $\epsilon(\lambda) = 0$.

We now define the notion of *message-randomisability* considering three different algorithms. The first one computes the plaintext's randomisation, the second computes it on the ciphertext, and the third one computes the randomness given two messages. $\mathcal{R}_{\mathsf{M}}$ denotes the set of random coins for message randomisations.

Definition 22: Message-randomisable PKE scheme ($\mathsf{MsgRand\text{-}PKE}$)

A PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is *message-randomisable* if there exists a set $\mathcal{R}_{\mathsf{M}}$ and two polynomial-time algorithms $\mathsf{MsgRandM}$ and $\mathsf{MsgRandC}$ such that:

1. $\mathsf{MsgRandM}$ takes $m \in \mathcal{M}$, $r \in \mathcal{R}_{\mathsf{M}}$ and returns $m' \in \mathcal{M}$. Moreover, the function $f_r : \mathcal{M} \Rightarrow \mathcal{M}$ defined by $f_r(m) = \mathsf{MsgRandM}(m, r)$, is bijective.
2. $\mathsf{MsgRandC}$ takes $c \in \mathcal{C}$, $r \in \mathcal{R}_{\mathsf{M}}$ and returns $c' \in \mathcal{C}$.
3. $\forall$ $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r' \in \mathcal{R}$, $r \in \mathcal{R}_{\mathsf{M}}$ and $c = \mathsf{Enc}(\mathsf{pk}, m; r')$:
$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{MsgRandC}(c, r)) = \mathsf{MsgRandM}(m, r)] = 1$$

Moreover, we say that Π is *message-derandomisable* if, for any $m \in \mathcal{M}$ and $r \in \mathcal{R}_{\mathsf{M}}$, there exists a unique efficiently computable r^* such that:
$$\Pr[m = \mathsf{MsgRandM}(\mathsf{MsgRandM}(m, r), r^*)] = 1$$
Finally, we say that Π is *message-random-extractable* if there exists a p.p.t algorithm $\mathsf{MsgRandExt}$ such that for any $m \in \mathcal{M}$ and $r \in \mathcal{R}_{\mathsf{M}}$:
$$\Pr[r = \mathsf{MsgRandExt}(m, \mathsf{MsgRandM}(m, r))] = 1$$

Note that we require $\mathsf{MsgRandM}$ to be bijective. This property is implicity required for the message-derandomisability. Indeed, if a randomised message can be obtained using different messages but the same randomness, then it could also be derandomised in several ways, which would contradict our definition.

Definition 23: Computational and perfect message-randomisability

A $\mathsf{MsgRand}$-PKE scheme is *computationally* message-randomisable if, for any λ, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and any polynomial-time distinguisher $\mathcal{D}$, there exists a negligible function $\epsilon(\cdot)$ such that:
$$\left| \Pr\left[\begin{array}{l} m' \xleftarrow{\$} \mathcal{M} \\ c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r) \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c') \end{array} : b = 1\right] - \Pr\left[\begin{array}{l} r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}} \\ c' \leftarrow \mathsf{MsgRandC}(c, r_{\mathsf{M}}) \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c, c') \end{array} : b = 1\right] \right| \leq \epsilon(\lambda)$$
We say that the scheme is *perfectly* message-randomisable when $\epsilon(\lambda) = 0$.

PKE schemes whose sets of random coins and messages are cyclic groups (with identity elements id_1 and id_2, respectively) and that are homomorphic (*i.e.,* $\mathsf{Enc}(\mathsf{pk}, m; r) * \mathsf{Enc}(\mathsf{pk}, m'; r') = \mathsf{Enc}(\mathsf{pk}, m * m'; r * r')$), are randomisable and message-randomisable. For example, to randomise a ciphertext $\mathsf{Enc}(\mathsf{pk}, m; r)$ with r' one can compute: $\mathsf{Enc}(\mathsf{pk}, m; r) * \mathsf{Enc}(\mathsf{pk}, \mathsf{id}_1; r') = \mathsf{Enc}(\mathsf{pk}, m; r * r')$. Similarly, to randomise the plaintext with m': $\mathsf{Enc}(\mathsf{pk}, m; r) * \mathsf{Enc}(\mathsf{pk}, m'; \mathsf{id}_2) = \mathsf{Enc}(\mathsf{pk}, m * m'; r)$.

The last notion to be defined is *key-randomisability.* We do so considering three algorithms as well. The first one randomises the public key, the second randomises the secret key, and the third one randomises a ciphertext given a randomised public key. For simplicity, we denote the set of random coins for key randomisations as $\mathcal{R}_{\mathsf{K}}$ although public and secret keys may not necessairly be defined over the same space.

Definition 24: Key-randomisable PKE scheme ($\mathsf{KeyRand}$-PKE)

A PKE scheme $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is *key-randomisable* if there exists a set $\mathcal{R}_{\mathsf{K}}$ and three polynomial-time algorithms such that:

1. $\mathsf{KeyRandPk}$ takes a public key pk, $r_{\mathsf{K}} \in \mathcal{R}_{\mathsf{K}}$ and returns pk'.

2. $\mathsf{KeyRandSk}$ takes a secret key sk, $r_{\mathsf{K}} \in \mathcal{R}_{\mathsf{K}}$ and returns sk'.
3. $\mathsf{KeyRandC}$ takes $c \in \mathcal{C}$, $r \in \mathcal{R}_{\mathsf{K}}$ and returns $c' \in \mathcal{C}$.
4. $\forall\ (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $r_{\mathsf{K}} \in \mathcal{R}_{\mathsf{K}}$ and $c = \mathsf{Enc}(\mathsf{pk}, m; r)$:

$$\Pr\left[\begin{array}{l}\mathsf{Dec}(\mathsf{sk}, c) = \mathsf{Dec}(\mathsf{KeyRandSk}(\mathsf{sk}, r_{\mathsf{K}}), \mathsf{KeyRandC}(c, r_{\mathsf{K}}))\\ \wedge\ \mathsf{Dec}(\mathsf{KeyRandSk}(\mathsf{sk}, r_{\mathsf{K}}), \mathsf{Enc}(\mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{K}}), m; r)) = m\end{array}\right] = 1$$

Moreover, we say that Π is *key-derandomisable* if for any secret key sk and $r_{\mathsf{K}} \in \mathcal{R}_{\mathsf{K}}$, there exists an efficiently computable r^*_{K} such that:

$$\Pr\left[\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{KeyRandSk}(\mathsf{sk}, r_{\mathsf{K}}), r^*_{\mathsf{K}})\right] = 1.$$

Definition 25: Computational and perfect key-randomisability

A $\mathsf{KeyRand\text{-}PKE}$ scheme is *computationally* key-randomisable if, for any λ, $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$, $m \in \mathcal{M}$, $r \in \mathcal{R}$, $c = \mathsf{Enc}(\mathsf{pk}, m; r)$ and any polynomial-time distinguisher $\mathcal{D}$, there exists a negligible function $\epsilon(\cdot)$ such that:

$$\left|\Pr\left[\begin{array}{l}(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KGen}(1^\lambda)\\ c' \leftarrow \mathsf{Enc}(\mathsf{pk}', m; r)\\ b \leftarrow \mathcal{D}(\mathsf{sk}, \mathsf{pk}, c', \mathsf{sk}', \mathsf{pk}')\end{array} : b = 1\right] - \Pr\left[\begin{array}{l}r_{\mathsf{K}} \xleftarrow{\$} \mathcal{R}_{\mathsf{K}}\\ \mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{K}})\\ \mathsf{sk}' = \mathsf{KeyRandPk}(\mathsf{sk}, r_{\mathsf{K}})\\ c' = \mathsf{KeyRandC}(c, r_{\mathsf{K}})\\ b \leftarrow \mathcal{D}(\mathsf{sk}, \mathsf{pk}, c', \mathsf{sk}', \mathsf{pk}')\end{array} : b = 1\right]\right| \leq \epsilon(\lambda)$$

We say that the scheme is *perfectly* key-randomisable when $\epsilon(\lambda) = 0$.

3.5 Interactive Protocols

This section presents protocols for proving plaintext equality and inequality where the common input consists of a public key and two ciphertexts generated with it. As private input, the prover has the corresponding private key. For plaintext inequality protocols we require the scheme to be randomisable, whereas, for plaintext equality, we also require it to be message-randomisable. In both cases, we first introduce an HVZK variant, which we then modify to achieve zero-knowledge in the presence of malicious verifiers.

3.5.1 Plaintext Inequality

Let us first introduce our protocol HPINEQ (Figure 3.1). It starts with the verifier choosing $r \xleftarrow{\$} \mathcal{R}$ and $b \xleftarrow{\$} \{0, 1\}$. Then it computes $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ and sends c'_b to the prover. At this stage, the prover receives a ciphertext that cannot link without decryption to c_0 or c_1. Since the verifier is honest, the prover either decrypts c'_b to m_0 or m_1 and can determine b and send it back to the verifier. The verifier accepts if and only if it receives b as expected.

The idea of this protocol can easily be explained to a large audience by replacing the ciphertexts with closed boxes using a padlock. Consider two identical closed boxes that contain a white card and a black card, respectively. The prover has a key that allows him to open both boxes and wants to prove to the verifier that the boxes contain different objects without revealing anything else. The verifier secretly chooses one of the two boxes and challenges the prover to guess which

```
Prover(sk, pk, $c_0, c_1$)                              Verifier(pk, $c_0, c_1$)
                                                        if $(\mathsf{pk}, c_0, c_1) \notin \mathcal{K} \times \mathcal{C}^2$ return $\bot$
                                                        $r \xleftarrow{\$} \mathcal{R}$; $b \xleftarrow{\$} \{0,1\}$
                                                        $c'_b \leftarrow \mathsf{Rand}(c_b, r)$
                                   $\xleftarrow{c'_b}$
if $\mathsf{Dec}(\mathsf{sk}, c'_b) = \mathsf{Dec}(\mathsf{sk}, c_0)$
then $z \leftarrow 0$ else $z \leftarrow 1$
                                   $\xrightarrow{z}$
                                                        return $z = b$
```

Figure 3.1: One execution of protocol HPINEQ (repeated λ times).

box he has chosen. The prover secretly opens the box and deduces which one it is by the card color. Upon telling the verifier which was the box and if the verifier agrees, they repeat the protocol λ times. If the two boxes contain the same card, the prover has no information about the recieved box and will fail to answer correctly with probability $1 - 1/2^\lambda$.

Theorem 7. Let Π be a PKE scheme, which is computationally randomisable (Definition 19 on page 31). If Π is used in HPINEQ, then HPINEQ is complete, computationally sound and perfect HVZK.

Proof. Completeness. When interacting with an honest verifier, the prover can compute $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c'_b)$, check whether it is equal to the message encrypted by c_0 or c_1 and send the value z so that the verifier always accepts.
Soundness. Let us define the following algorithm, which will be used to generate instances of the protocol HPINEQ:

```
GenInstance($1^\lambda, \mathcal{R}, \mathcal{M}$)
$(r_0, r_1) \xleftarrow{\$} \mathcal{R}^2; m \xleftarrow{\$} \mathcal{M}$
$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
$c_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r_0)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r_1)$
return $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
```

For a tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ returned by GenInstance, the prover and the verifier recieve common input $x = (\mathsf{pk}, c_0, c_1)$. The prover also receives auxiliary input $w = \mathsf{sk}$. We recall that $\forall\ x \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol.

It follows that $\forall\ x$ s.t. $x \notin L_\mathcal{R}$ and $x \notin \mathcal{K} \times \mathcal{C}^2$: $\Pr[\langle \mathcal{P}^*(w), \mathcal{V}(z)\rangle(x) = 1] = 0$ for any witness w and any bit-string z. Furthermore, for all instances x such that $x \notin L_\mathcal{R}$ and $x \in \mathcal{K} \times \mathcal{C}^2$: $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})\}$. This means that the soundness of the protocol HPINEQ can be proven by showing that for any witness w, any bit-string z and any instance $x = (\mathsf{pk}, c_0, c_1)$ generated from the output of GenInstance, $\Pr[\langle \mathcal{P}^*(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligible.

Let us now define an experiment $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}$ that takes a tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ generated by GenInstance as input, where an adversary $\mathcal{A}$ plays one round of the protocol HPINEQ as a (dishonest) prover against a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a pair of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2)$

```
Game $G_0$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$
return $z = b$
```

```
Game $G_1$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
$c'_1 \leftarrow \mathsf{Rand}(c_1, r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$
return $z = b$
```

```
Game $G_2$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$
return $z = b$
```

Figure 3.2: Sequence of games for HPINEQ.

where $\mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ instantiates the dishonest prover and returns a state st, and $\mathcal{A}_2(\mathsf{st}, c')$ corresponds to the interaction between the verifier and the prover (*i.e.*, it takes a challenge c' as input and returns a response z). The experiment runs as follows:

```
$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, \mathsf{pk}, c_0, c_1))$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0,1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c'_b)$
return $z = b$
```

Next, we prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to 1/2 for any tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Observe that for all w such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, it holds that $x \in \{(\mathsf{pk}, c_0, c_1) \mid (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})\}$. Since the protocol can be repeated λ times, it follows that for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, $\Pr[\langle \mathcal{P}^*(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligibly close to $1/2^\lambda$, which means that the soundness probability is also negligible (when run λ times).

We define a sequence of games (Figure 3.2) which are played between an adversary $\mathcal{A}$ and the challenger.

Game 0: The first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, \mathsf{pk}, c_0, c_1))$ for a fixed tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Considering the adversary's view, game G_0 represents a real execution of the protocol HPINEQ. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

Game 1: G_1 is defined as G_0 except that we replace the instruction $c'_0 \leftarrow \mathsf{Rand}(c_0, r)$ by $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$ and $c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$.

We claim and prove by reduction that:

$$|\Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1]| \leq \epsilon_{\mathsf{rand}}(\lambda),$$

where $\epsilon_{\mathsf{rand}}(\lambda)$ is the randomisability advantage of Π. Let c' be a ciphertext generated by one of these two methods:

1. $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r')$
2. $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0, r')$ (where $m = \mathsf{Dec}(\mathsf{sk}, c_0)$)

We build the distinguisher $\mathcal{D}(\mathsf{pk}, c_0, c')$ as follows: $\mathcal{D}$ simulates the protocol HPINEQ for the fixed statement $x = (\mathsf{pk}, c_0, c_1)$, except that if $b = 0$, it sets $c'_0 \leftarrow c'$. If the proof is accepted then $\mathcal{D}$ returns 1, else it returns 0.

- If $c' \leftarrow \mathsf{Rand}(c_0, r')$, $\mathcal{D}$ perfectly simulates G_0, so:
  $\Pr[\mathcal{A} \text{ wins } G_0] = \Pr\left[\ r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c, r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c');\ \ : b = 1\right]$
- If $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r')$, $\mathcal{D}$ perfectly simulates G_1, so:
  $\Pr[\mathcal{A} \text{ wins } G_1] = \Pr\left[\ r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c');\ \ : b = 1\right]$

which concludes the proof of the claim.

Game 2: G_2 is defined as G_1 except that we replace the instruction $c'_1 \leftarrow \mathsf{Rand}(c_1, r)$ by $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$ and $c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$.

We claim that $|\Pr[\mathcal{A} \text{ wins } G_1] - \Pr[\mathcal{A} \text{ wins } G_2]| \leq \epsilon_{\mathsf{rand}}(\lambda)$, which we prove as in the previous game. Finally, since $m = \mathsf{Dec}(\mathsf{sk}, c_0) = \mathsf{Dec}(\mathsf{sk}, c_1)$ and c'_b is always computed from $\mathsf{Enc}(\mathsf{pk}, m)$ in G_2, we deduce that c'_b does not depend on b, which implies that $\mathcal{A}$ receives no information that depends on b. Therefore, the best strategy of $\mathcal{A}$ in G_2 is to guess b at random, so:

$$\Pr[\mathcal{A} \text{ wins } G_2] = 1/2$$

Based on the indistinguishability of transitions from the given game sequence, we conclude that if we repeat the protocols λ times, the probability that $\mathcal{A}$ breaks the soundness is negligible and majored by:

$$\Pr[\mathcal{A} \text{ wins } G_0]^\lambda \leq (2 \cdot \epsilon_{\mathsf{rand}}(\lambda) + 1/2)^\lambda$$

Zero-Knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}, c_0, c_1)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$ and computes $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(\mathsf{pk}, c_b, r)$. Finally, it returns (c'_b, b). The simulator acts as in the real protocol, so it perfectly simulates the proof. □

$\mathsf{Prover}(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$		$\mathsf{Verifier}(\mathsf{pk}, c_0, c_1)$
		if $(\mathsf{pk}, c_0, c_1) \notin \mathcal{K} \times \mathcal{C}^2$ **return** $\perp$ $r \xleftarrow{\$} \mathcal{R}; b \xleftarrow{\$} \{0, 1\}; c'_b \leftarrow \mathsf{Rand}(c_b, r)$
	$\xleftarrow{c'_b}$	
$z \leftarrow \neg(\mathsf{Dec}(\mathsf{sk}, c_0) = \mathsf{Dec}(\mathsf{sk}, c'_b))$ $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(z)$		
	$\xrightarrow{\mathsf{comm}}$	
	$\xleftarrow{r,b}$	
if $c'_b \neq \mathsf{Rand}(c_b, r)$ **then** $\mathsf{op} \leftarrow \perp$	$\xrightarrow{\mathsf{op}}$	
		$z' \leftarrow \mathsf{Open}(\mathsf{comm}, \mathsf{op})$ **return** $z' = b$

Figure 3.3: One execution of protocol PINEQ (repeated λ times).

Protocol PINEQ (Figure 3.3), is an amendment of HPINEQ that uses a commitment scheme. Without it, a malicious verifier could send a ciphertext that is not a randomisation of c_0 or c_1 and check whether it encrypts the same value. The commitment scheme protects the prover from such verifiers. To this end, the verifier first randomises the ciphertext and then sends it to the prover, which computes z and commits to it. Then, the verifier reveals the randomness used at the first stage and the prover opens the commitment if and only if these values are consistent with the ciphertext obtained from the verifier.

Theorem 8. Let Π be a PKE scheme, which is derandomisable (Definition 18 on page 30) and perfectly strong randomisable (Definition 21 on page 31). Let Γ be a commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol PINEQ, then PINEQ is complete, computationally sound and perfect zero-knowledge.

Proof. Completeness. Since Π is randomisable, the prover can decrypt c'_b (or c'_{1-b}) and compare the plaintext with the decryptions of c_0 and c_1 to determine z. Hence, P commits to the right value, and V always accepts the proof.
Soundness. Let us define the following algorithm:

```
$\mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})$
$(r, r_0, r_1) \xleftarrow{\$} \mathcal{R}^3; m \xleftarrow{\$} \mathcal{M}$
$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r); c_0 \leftarrow \mathsf{Rand}(c, r_0); c_1 \leftarrow \mathsf{Rand}(c, r_1)$
$\mathbf{return}\ (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$
```

We recall that $\forall\ x \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. It follows that $\forall\ x$ such that $x \notin L_{\mathcal{R}}$ and $x \notin \mathcal{K} \times \mathcal{C}^2$: $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1] = 0$ for any witness w and any bit-string z. Since the scheme is perfectly strongly randomisable, the ciphertexts produced by the encryption algorithm follow the same distribution as those produced by the randomisation algorithm. Therefore, $\forall\ x$ such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, we have $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})\}$. This means that the soundness of the protocol PINEQ can be proven by showing that for any witness w, any auxiliary input z, and any instance $x = (\mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$, $\Pr[\langle \mathcal{B}(x), \mathcal{V}(z)\rangle(x) = 1]$ is negligible.

Let us define an experiment $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}$, which takes as input a tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$. In the experiment, the adversary $\mathcal{A}$ executes the protocol PINEQ as a (dishonest) prover against a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a triplet of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ where $\mathcal{A}_1(w, x)$ initiates the dishonest prover and returns st_1; $\mathcal{A}_2(\mathsf{st}_1, c'_b)$ corresponds to the first interaction of the protocol (*i.e.,* it takes a challenge c'_b as input and returns a state st_2 and the response comm); and $\mathcal{A}_3(\mathsf{st}_2, r, b)$ corresponds to the second interaction (*i.e.,* it takes a challenge r and b as input and returns a response op). The experiment runs as follows:

```
$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0, 1\}, r \xleftarrow{\$} \mathcal{R}; c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$
$\mathbf{return}\ \mathsf{Open}(\mathsf{comm}, \mathsf{op}) = b$
```

```
**Game** $G_0$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathcal{R}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$
**return** $\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = b$
```

```
**Game** $G_1$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}; r' \xleftarrow{\$} \mathcal{R}; r \leftarrow \mathsf{RandR}(r_b, r')$
$c'_b \leftarrow \mathsf{Rand}(c, r')$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, r, b)$
**return** $\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = b$
```

Figure 3.4: Sequence of games for PINEQ.

We prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to 1/2 for any tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$. $\forall\ x$ s.t. $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, we have that $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, r_0, r_1, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})\}$. Moreover, since the protocol is repeated λ times, one can deduce that for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, it holds that $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligibly close to $1/2^\lambda$. This means that the soundness probability is negligible (when executed λ times).

In Figure 3.4, we define a sequence of games between an adversary $\mathcal{A}$ and a challenger for a fixed tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$.

Game 0: The first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$. Considering the adversary's view, game G_0 is a real execution of PINEQ. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

Game 1: G_1 proceeds as G_0 except that we replace the instructions $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ by $r' \xleftarrow{\$} \mathcal{R}$, $c'_b \leftarrow \mathsf{Rand}(c, r')$ and $r \leftarrow \mathsf{RandR}(r_b, r')$. We note that since the encryption scheme is strongly randomisable, derandomisable, and $c_b \leftarrow \mathsf{Rand}(c, r_b)$: $c = \mathsf{Rand}(c_b, r_b)$ and $c'_b = \mathsf{Rand}(c_b, \mathsf{RandR}(r_b, r')) = \mathsf{Rand}(c_b, r)$. Moreover, an element r produced by the sequence of instructions $r' \xleftarrow{\$} \mathcal{R}; r \leftarrow \mathsf{RandR}(r_b, r')$ follows the same distribution as $r \xleftarrow{\$} \mathcal{R}$. Therefore:

$$\Pr[\mathcal{A} \text{ wins } G_0] = \Pr[\mathcal{A} \text{ wins } G_1]$$

Next, we claim and prove by reduction that $\left|\Pr[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| = \epsilon_{binding}(\lambda)/2$, where $\epsilon_{binding}(\lambda)$ is the adversary's advantage in the binding experiment of the commitment scheme (Definition 2 on page 14). In game G_1, the challenger can generate a random coin r for both $b = 1$ and $b = 0$ on the same challenge $c' = c'_0 = c'_1$ because it builds c'_b computing $r' \xleftarrow{\$} \mathcal{R}$, $c' \leftarrow \mathsf{Rand}(c, r')$, and r computing $r \leftarrow \mathsf{RandR}(r_b, r')$. To break the soundness of the protocol, $\mathcal{A}$ must be able to succeed in both cases ($b = 1$ and $b = 0$) with non negligible probability at each round. If this is not the case, the advantage of $\mathcal{A}$ is bounded by a value that is negligibly close to $1/2^\lambda$. We show that if such an adversary exists, we can build an algorithm that breaks the binding property of the commitment scheme. If the adversary is able to succeed for both cases ($b = 1$ and $b = 0$), then it is able to open its commitment comm for two different messages $z = 0$ and $z = 1$, which is equivalent to breaking the binding property.

We build an adversary $\mathcal{B}$ that has an advantage $\epsilon_{binding}(\lambda)$ in the binding experiment as follows: $\mathcal{B}$ receives the commitment scheme's public parameters pp and simulates the game G_1 for $b = 0$ to $\mathcal{A}$ using pp. More formally, $\mathcal{B}$ runs as follows:

```
$\mathcal{B}(\mathsf{pp}, \mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$r' \xleftarrow{\$} \mathcal{R}; \hat{r}_0 \leftarrow \mathsf{RandR}(r_0, r'); \hat{r}_1 \leftarrow \mathsf{RandR}(r_1, r')$
$c' \leftarrow \mathsf{Rand}(c, r')$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c')$
$\mathsf{op}_0 \leftarrow \mathcal{A}_3(\mathsf{st}_2, \hat{r}_0, b)$
$\mathsf{op}_1 \leftarrow \mathcal{A}_3(\mathsf{st}_2, \hat{r}_1, b)$
if $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = 0$ then $\mathsf{win}_0 \leftarrow 1$
if $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1) = 1$ then $\mathsf{win}_1 \leftarrow 1$
if $\mathsf{win}_0 = 1 \wedge \mathsf{win}_1 = 1$ return $(\mathsf{comm}, \mathsf{op}_0, \mathsf{op}_1)$
```

Note that if $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = 0 \neq 1 = \mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1)$, $\mathcal{B}$ wins the experiment. Therefore, we have that:

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins } G_1] &= \Pr[b=0] \cdot \Pr[\mathsf{win}_0 = 1] + \Pr[b=1] \cdot \Pr[\mathsf{win}_1 = 1] \\
&= \frac{1}{2}\Pr[\mathsf{win}_0 = 1] + \frac{1}{2}\Pr[\mathsf{win}_1 = 1]
\end{aligned}$$

Let us now set three events E_0, E_1 and E_2 as follows:

- $E_0 =$ "$\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = \perp$ or $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1) = \perp$".
- $E_1 =$ "$\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) \neq \mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1)$ and $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) \neq \perp$ and $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1) \neq \perp$". Note that $\Pr[E_1] = \epsilon_{binding}(\lambda)$.
- $E_2 =$ "$\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = \mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1)$ and $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) \neq \perp$ and $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1) \neq \perp$".

We have that:

$$\begin{aligned}
\Pr[E_0] + \Pr[E_1] + \Pr[E_2] &= 1 \\
\Leftrightarrow \Pr[E_2] &= 1 - \Pr[E_0] - \Pr[E_1] \\
\Rightarrow \Pr[E_2] &\leq 1 - \epsilon_{binding}(\lambda)
\end{aligned}$$

Moreover,

$$\begin{aligned}
\Pr[\mathsf{win}_1 = 1] &= \sum_{i=0}^{2} \Pr[E_i] \cdot \Pr[\mathsf{win}_1 = 1 | E_i] \\
&\leq \Pr[E_1] \cdot \Pr[\mathsf{win}_1 = 1 | E_1] + \Pr[E_2] \cdot \Pr[\mathsf{win}_1 = 1 | E_2] \\
&\leq \epsilon_{binding}(\lambda) \cdot \Pr[\mathsf{win}_1 = 1 | E_1] + (1 - \epsilon_{binding}(\lambda)) \cdot \Pr[\mathsf{win}_1 = 1 | E_2]
\end{aligned}$$

In the case E_2, if $\mathcal{A}$ wins the case $b = 0$, it loses the case $b = 1$. We have $\Pr[\mathsf{win}_1 = 1 | E_2] = 1 - \Pr[\mathsf{win}_0 = 1]$. On the other hand, in the case E_1, $\Pr[\mathsf{win}_1 = 1 | E_1] = \Pr[\mathsf{win}_0 = 1]$, which implies that:

$$\begin{aligned}
\Pr[\mathsf{win}_1 = 1] &\leq \epsilon_{binding}(\lambda) \cdot \Pr[\mathsf{win}_0 = 1] + (1 - \epsilon_{binding}(\lambda)) \cdot (1 - \Pr[\mathsf{win}_0 = 1]) \\
&\leq 1 - \Pr[\mathsf{win}_0 = 1] + \epsilon_{binding}(\lambda) \cdot (2 \cdot \Pr[\mathsf{win}_0 = 1] - 1)
\end{aligned}$$

This in turns implies that:

$$\begin{aligned}
\Pr[\mathsf{win}_1 = 1] + \Pr[\mathsf{win}_0 = 1] &\leq 1 + \epsilon_{binding}(\lambda) \cdot (2 \cdot \Pr[\mathsf{win}_0 = 1] - 1) \\
\Leftrightarrow \frac{1}{2} \cdot \Pr[\mathsf{win}_1 = 1] + \frac{1}{2} \cdot \Pr[\mathsf{win}_0 = 1] &\leq \frac{1}{2} + \epsilon_{binding}(\lambda) \cdot \left(\Pr[\mathsf{win}_0 = 1] - \frac{1}{2}\right) \\
\Leftrightarrow \left|\Pr[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| &\leq \frac{\epsilon_{binding}(\lambda)}{2}
\end{aligned}$$

Finally, if the protocol is repeated λ times, the soundness probability becomes:

$$\Pr[\mathcal{A} \text{ wins } G_0]^\lambda = \Pr[\mathcal{A} \text{ wins } G_1]^\lambda \leq \left(\tfrac{1+\epsilon_{binding}(\lambda)}{2}\right)^\lambda$$

Since the commitment scheme has the binding property by hypothesis, $\epsilon_{binding}(\lambda)$ is negligible and so is $\Pr[\mathcal{A} \text{ wins } G_0]^\lambda$, which concludes the proof of the soundness.
Zero-Knowledge. Let $\mathcal{V}^*$ be a dishonest verifier. We define a Simulator $\mathcal{S}_{\mathcal{V}^*}(x)$ where $x = (\mathsf{pk}, c_0, c_1)$ that perfectly simulates $\mathcal{V}^*$. The simulator $\mathcal{S}$ generates c'_b and r as the dishonest verifier $\mathcal{V}^*$.

- If $c'_b = \mathsf{Rand}(c_z, r_z)$ where $z \xleftarrow{\$} \{0,1\}$, then the simulator runs $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(z)$ and returns the transcript $(c'_b, \mathsf{comm}, (r, z), \mathsf{op})$.
- If $c'_b \neq \mathsf{Rand}(c_z, r_z)$ where $z \xleftarrow{\$} \{0,1\}$, then the simulator picks comm in the uniform distribution on the commitment set and returns the transcript $(c'_b, \mathsf{comm}, (r, z), \perp)$.

The simulator follows the same distribution as the real protocol. If the verifier sends a wrong r, then it simulates the prover's messages using a random commitment comm and the $\perp$ symbol. Indeed, since the commitment's open key remains unknown, the prover commitment is like a random commitment picked in the uniform distribution according to the perfect hiding property. □

3.5.2 Plaintext Equality

As before, we begin explaining our protocol HPEQ (Figure 3.5). First, the verifier chooses $r \xleftarrow{\$} \mathcal{R}$, $r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$ and $b \xleftarrow{\$} \{0,1\}$. Then it computes $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ and $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$ to send c''_b to the prover. At this stage, the prover receives a ciphertext that cannot be linked to c_0 nor to c_1. The prover decrypts c''_b obtaining

$\mathsf{Prover}(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$		$\mathsf{Verifier}(\mathsf{pk}, c_0, c_1)$
		if $(\mathsf{pk}, c_0, c_1) \notin \mathcal{K} \times \mathcal{C}^2$ **return** $\perp$
		$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}; b \xleftarrow{\$} \{0,1\}$
		$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
		$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$
	$\xleftarrow{c''_b}$	
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c''_b); m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$		
$z \leftarrow \mathsf{MsgRandExt}(m', m)$		
	$\xrightarrow{z}$	
		return $z = r_\mathsf{M}$

Figure 3.5: One execution of protocol HPEQ (repeated λ times).

a message m', which corresponds to a message-randomisation of either the message decrypted by c_0 or by c_1. The prover computes $z \leftarrow \mathsf{MsgRandExt}(m', m)$ and sends it to the verifier. The verifier accepts if and only if $z = r_\mathsf{M}$. Since both ciphertexts, c_0 and c_1, belong to $\mathsf{Enc}(\mathsf{pk}, m)$, the prover can always compute z correctly. If this is not the case, then a cheating prover can only correctly guess the bit b with probability at most $1/2$.

Theorem 9. Let Π be a PKE scheme, which is computationally randomisable (Definition 19 on page 31), computationally message-randomisable and message-random-extractable (Definition 22 on page 31). If Π is the scheme used in HPEQ, then HPEQ is complete, computationally sound and perfect HVZK.

Proof. Completeness. When interacting with an honest verifier, the prover computes $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c''_b)$. Then, given that $c_0, c_1 \in \mathsf{Enc}(\mathsf{pk}, m)$ and that the scheme is message-random-extractable, the prover can always compute $z \leftarrow \mathsf{MsgRandExt}(m', m)$, with $z = r_M$ so that the verifier always accepts.
Soundness. Let us define the following algorithm:

```
$\mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{M})$
--------------------------------
$(r_0, r_1) \xleftarrow{\$} \mathcal{R}^2,\ m_0 \xleftarrow{\$} \mathcal{M}$
$m_1 \xleftarrow{\$} \mathcal{M} \backslash \{m_0\}$
$(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(1^\lambda)$
$c_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r_0)$
$c_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r_1)$
$\mathbf{return}\ (\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
```

We recall that for all $x \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. Furthermore, for all instances x such that $x \notin L_\mathcal{R}$ and $x \notin \mathcal{K} \times \mathcal{C}^2$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1] = 0$ for any witness w and any bit-string z. On the other hand, for all instances x s.t. $x \notin L_\mathcal{R}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{M})\}$. This means that the soundness of the protocol HPEQ can be proven by showing that for any witness w, any bit-string z and any instance $x = (\mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligible.

In the following, we define an experiment that takes as input a tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$. In such experiment, the adversary $\mathcal{A}$ executes the protocol HPEQ as a (dishonest) prover with a *challenger* that plays the role of an honest verifier. The $\mathcal{A}$ is defined as a pair of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1(w, x)$ initiates the dishonest prover and returns a state st; and $\mathcal{A}_2(\mathsf{st}, c')$ corresponds to the interaction between the verifier and the prover (it takes a challenge c' as input and returns a response z). The experiment is presented below.

```
$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, \mathsf{pk}, c_0, c_1))$
--------------------------------
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$r \xleftarrow{\$} \mathcal{R};\ r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M};\ b \xleftarrow{\$} \{0, 1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$
$z \leftarrow \mathcal{A}(\mathsf{st}, c''_b)$
$\mathbf{return}\ z = r_\mathsf{M}$
```

In what follows, we prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to 1/2 for any tuple $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Furthermore, when the protocol is repeated λ times, for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligibly close to $1/2^\lambda$, meaning the soundness probability is negligible.

In Figure 3.6 we present a sequence of games played between the adversary $\mathcal{A}$ and the challenger.

Game 0: The first game G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, \mathsf{pk}, c_0, c_1))$ for a fixed $(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$. Considering the adversary's view, G_0 represents a real execution of one round of the protocol HPEQ. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

```
Game $G_0$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$
$b \xleftarrow{\$} \{0,1\}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
return $z = r_\mathsf{M}$
```

```
Game $G_1$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$
$b \xleftarrow{\$} \{0,1\}$
$m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r)$
$c'_1 \leftarrow \mathsf{Rand}(c_1, r)$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
return $z = r_\mathsf{M}$
```

```
Game $G_2$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$
$b \xleftarrow{\$} \{0,1\}$
$m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r)$
$m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r)$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_\mathsf{M})$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
return $z = r_\mathsf{M}$
```

```
Game $G_3$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}; b \xleftarrow{\$} \{0,1\}$
$m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r)$
$m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r)$
$m'_0 \xleftarrow{\$} \mathcal{M}; c''_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r)$
$c''_1 \leftarrow \mathsf{MsgRandC}(c'_1, r_\mathsf{M})$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
if $(b = 0)$ then
 return $z = \mathsf{MsgRandExt}(m_0, m'_0)$
else return $z = r_\mathsf{M}$
```

```
Game $G_4$
$\mathsf{st} \leftarrow \mathcal{A}_1(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}; b \xleftarrow{\$} \{0,1\}$
$m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$
$c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r)$
$m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$
$c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r)$
$m'_0 \xleftarrow{\$} \mathcal{M}; c''_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r)$
$m'_1 \xleftarrow{\$} \mathcal{M}; c''_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_1; r)$
$z \leftarrow \mathcal{A}_2(\mathsf{st}, c''_b)$
return $z = \mathsf{MsgRandExt}(m_b, m'_b)$
```

Figure 3.6: Sequence of games for HPEQ.

Game 1: G_1 is defined as G_0 except that we replace the instruction $c'_0 \leftarrow \mathsf{Rand}(c_0, r)$ by $m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$ and $c'_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r)$.

We claim and prove by reduction that $|\Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1]| \leq \epsilon_{\mathsf{rand}}(\lambda)$, where $\epsilon_{\mathsf{rand}}(\lambda)$ is the randomisability advantage of the encryption scheme. Let c' be a ciphertext generated by one of these two methods (where $m_0 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$):

1. $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r')$

2. $r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0, r')$

We build the distinguisher $\mathcal{D}(\mathsf{pk}, c_0, c')$ as follows: $\mathcal{D}$ simulates the protocol HPEQ, except that if $b = 0$, it sets $c'_0 =\leftarrow c'$. If the proof is accepted then $\mathcal{D}$ returns 1, else it returns 0.

- If $c' \leftarrow \mathsf{Rand}(c_0, r')$, $\mathcal{D}$ perfectly simulates G_0, so:

$$\Pr[\mathcal{A} \text{ wins } G_0] = \Pr\left[\ r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Rand}(c_0, r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c');\quad : b = 1\right]$$

- If $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r')$, $\mathcal{D}$ perfectly simulates G_1, so:

$$\Pr[\mathcal{A} \text{ wins } G_1] = \Pr\left[\ r' \xleftarrow{\$} \mathcal{R}; c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m_0; r'); b \leftarrow \mathcal{D}(\mathsf{pk}, c_0, c');\quad : b = 1\right]$$

which concludes the proof of the claim.

Game 2: G_2 is defined as G_1 except that we replace the instruction $c'_1 \leftarrow \mathsf{Rand}(c_1, r)$ by $m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, c_1)$ and $c'_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1; r)$. We claim and prove as before that

$$|\Pr[\mathcal{A} \text{ wins } G_1] - \Pr[\mathcal{A} \text{ wins } G_2]| \leq \epsilon_{\mathsf{rand}}(\lambda)$$

Game 3: G_3 is defined as G_2 except that we replace the instruction $c''_0 \xleftarrow{\$} \mathsf{MsgRand}(c'_0, r_{\mathsf{M}})$ by $m'_0 \xleftarrow{\$} \mathcal{M}$ and $c''_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r)$. We claim and prove by reduction that

$$|\Pr[\mathcal{A} \text{ wins } G_2] - \Pr[\mathcal{A} \text{ wins } G_3]| \leq \epsilon_{\mathsf{msgRand}}(\lambda)$$

where $\epsilon_{\mathsf{msgRand}}(\lambda)$ is the message-randomisability advantage of the encryption scheme.

Let c'' be a ciphertext generated by one of these two methods:

1. $m'_0 \xleftarrow{\$} \mathcal{M}; c'' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r'_0)$
2. $r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{M}})$

We build a distinguisher $\mathcal{D}(\mathsf{pk}, c'_0, c'')$ as follows: $\mathcal{D}$ simulates the protocol HPEQ, except that if $b = 0$, then it sets $c''_0 \leftarrow c''$. If the proof is accepted then $\mathcal{D}$ returns 1, else it returns 0.

- If $c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{M}})$, $\mathcal{D}$ perfectly simulates G_2, so:

$$\Pr[\mathcal{A} \text{ wins } G_2] = \Pr\begin{bmatrix} r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}; & \\ c'' \leftarrow \mathsf{MsgRandC}(c'_0, r_{\mathsf{M}}); & : b = 1 \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c'_0, c''); & \end{bmatrix}$$

- If $c'' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r'_0)$, $\mathcal{D}$ perfectly simulates G_3, so:

$$\Pr[\mathcal{A} \text{ wins } G_3] = \Pr\begin{bmatrix} m'_0 \xleftarrow{\$} \mathcal{M}; & \\ c'' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r'_0); & : b = 1 \\ b \leftarrow \mathcal{D}(\mathsf{pk}, c'_0, c''); & \end{bmatrix}$$

which concludes the proof of the claim.

Game 4: G_4 is defined as G_3 except that we replace the instruction $c''_1 \xleftarrow{\$} \mathsf{MsgRand}(c'_1, r_{\mathsf{M}})$ by $m'_1 \xleftarrow{\$} \mathcal{M}$ and $c''_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m'_0; r)$. As before, it follows that

$$|\Pr[\mathcal{A} \text{ wins } G_3] - \Pr[\mathcal{A} \text{ wins } G_4]| \leq \epsilon_{\mathsf{msgRand}}(\lambda)$$

Finally, in G_4, m'_0 and m'_1 are randomly picked and independent of b. Therefore, the adversary receives no information that depends on b from c''_b. Since m_0 and m_1 are different, it follows that the best strategy $\mathcal{A}$ has is to guess b and compute z as $z \leftarrow \mathsf{MsgRandExt}(m_b, \mathsf{Dec}(\mathsf{sk}, c''_b))$. We conclude that $\Pr[\mathcal{A} \text{ wins } G_4] = 1/2$.

Based on the indistinguishability of transitions from the given game sequence, we conclude that if we repeat the protocols λ times, the probability that $\mathcal{A}$ breaks the soundness is negligible and majored by:

$$\Pr[\mathcal{A} \text{ wins } G_0]^\lambda \leq \left(2 \cdot (\epsilon_{\mathsf{rand}}(\lambda) + \epsilon_{\mathsf{msgRand}}(\lambda)) + \tfrac{1}{2}\right)^\lambda .$$

Zero-Knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}, c_0, c_1)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0,1\}$, $r \xleftarrow{\$} \mathcal{R}, r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, and computes $c'_b \leftarrow \mathsf{Rand}(\mathsf{pk}, c_b, r)$ and $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{M}})$. Finally, it returns (c''_b, b). The simulator acts as in the real protocol so it perfectly simulates the proof. □

$\mathsf{Prover}(\mathsf{sk}, \mathsf{pk}, c_0, c_1)$		$\mathsf{Verifier}(\mathsf{pk}, c_0, c_1)$
		if $(\mathsf{pk}, c_0, c_1) \notin \mathcal{K} \times \mathcal{C}^2$ **return** $\perp$ $r \xleftarrow{\$} \mathcal{R}$; $r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$; $b \xleftarrow{\$} \{0,1\}$ $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{M}})$
	$\xleftarrow{c''_b}$	
$m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c''_b); m \leftarrow \mathsf{Dec}(\mathsf{sk}, c_0)$ $z \leftarrow \mathsf{MsgRandExt}(m', m)$ $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(z)$		
	$\xrightarrow{\mathsf{comm}}$	
	$\xleftarrow{(r, r_{\mathsf{M}}, b)}$	
if $\mathsf{MsgRandC}(\mathsf{Rand}(c_b, r), r_{\mathsf{M}}) \neq c''_b$ **then** $\mathsf{op} \leftarrow \perp$		
	$\xrightarrow{\mathsf{op}}$	
		$z' \leftarrow \mathsf{Open}(\mathsf{comm}, \mathsf{op})$ **return** $z' = r_{\mathsf{M}}$

Figure 3.7: One execution of protocol PEQ (repeated λ times).

Figure 3.7 presents a variant that has perfect (full) zero-knowledge thanks to the use of a commitment scheme. Without one, a malicious verifier could send a ciphertext c^* for which the corresponding message m^* is known. Once z is received from the prover, the verifier will gain information about the relation between m, m^* and z to compute m. In brief, the prover first commits to the value z, relying on the hiding property of the commitment scheme. Then, it checks whether the verifier has correctly randomised the messages or not to open the commitment.

Theorem 10. Let Π be a PKE scheme, which is perfectly strong randomisable and derandomisable, perfectly message-randomisable and message-derandomisable and message-random-extractable. Let Γ be the commitment scheme, which is computationally binding and perfectly hiding. If Π and Γ are used in the protocol PEQ, then PEQ is complete, computationally sound and perfect zero-knowledge.

Proof. Completeness. If $\mathsf{Dec}(\mathsf{sk}, c_0) = \mathsf{Dec}(\mathsf{sk}, c_1)$, a message-randomisation of both ciphertexts will decrypt to the same message m'. Hence, regardless of which ciphertext the prover receives from the verifier, if it is a message-randomisation of either c_0 or c_1 and the scheme is message-random-extractable, the prover can correctly compute $z \leftarrow \mathsf{MsgRandExt}(m', m)$ with $z = r_{\mathsf{M}}$. It follows that the prover always opens a commitment for r_{M} so that the verifier accepts.
Soundness. Once again, we define the following algorithm:

```
$\mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{R}_{\mathsf{M}}, \mathcal{M})$
$(r, r_0, r_1) \xleftarrow{\$} \mathcal{R}^3; r^0_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$
$r^1_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}} \backslash \{r^0_{\mathsf{M}}\}; m \xleftarrow{\$} \mathcal{M}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
$c'_0 \leftarrow \mathsf{MsgRandC}(c, r^0_{\mathsf{M}}); c'_1 \leftarrow \mathsf{MsgRandC}(c, r^1_{\mathsf{M}})$
$c_0 \leftarrow \mathsf{Rand}(c'_0, r_0); c_1 \leftarrow \mathsf{Rand}\ (c'_1, r_1)$
$\textbf{return}\ (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$
```

Note that since c'_0 and c'_1 are message-randomisations of c with two different coins $r_{\mathsf{M},0} \neq r_{\mathsf{M},1}$, it holds that $\mathsf{Dec}(\mathsf{sk}, c'_0) \neq \mathsf{Dec}(\mathsf{sk}, c'_1)$ for any $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ returned by $\mathsf{GenInstance}$. We recall that for all $x \notin \mathcal{K} \times \mathcal{C}^2$, the verifier aborts the protocol. For all instances x such that $x \notin L_{\mathcal{R}}$ and $x \notin \mathcal{K}\times\mathcal{C}^2$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1] = 0$ for any witness w and any bit-string z.

Since the encryption scheme is perfectly randomisable, message-randomisable, derandomisable and message-derandomisable, the ciphertexts produced by the encryption algorithms on random messages follow the same distribution as the ones produced by the randomisation and messages randomisation algorithms. Therefore, for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$: $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{R}_{\mathsf{M}}, \mathcal{M})\}$. This means that the soundness of the protocol PEQ can be proven by showing that for any witness w, any bit-string z and any instance $x = (\mathsf{pk}, c_0, c_1)$ generated from the output of $\mathsf{GenInstance}$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle\ (x) = 1]$ is negligible.

In the following, we define an experiment that takes on input a tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$ generated by $\mathsf{GenInstance}$. In such an experiment, the adversary $\mathcal{A}$ executes the protocol PEQ as a (dishonest) prover with a *challenger* that plays the role of an honest verifier. We define $\mathcal{A}$ as a triplet of p.p.t algorithms $(\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ where $\mathcal{A}_1(w, x)$ initiates the prover and returns a state st_1; $\mathcal{A}_2(\mathsf{st}_1, c''_b)$ corresponds to the first interaction between the verifier and the prover (it takes a challenge c''_b as input, returning a state st_2 and response comm); and $\mathcal{A}_3(\mathsf{st}_2, (r, r_{\mathsf{M}}, b))$ corresponds to the second interaction between the verifier and the prover (it takes a challenge (r, r_{M}, b) as input and returns a response op). The experiment is defined as follows:

```
$\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}; r \xleftarrow{\$} \mathcal{R}; r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r); c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{M}})$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c''_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, (r, r_{\mathsf{M}}, b))$
**return** $\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = r_{\mathsf{M}}$
```

Now we prove that for any adversary $\mathcal{A}$, the probability of winning this experiment is negligibly close to $1/2$ for any tuple $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$. First we observe that for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, $x \in \{(\mathsf{pk}, c_0, c_1) | (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1) \leftarrow \mathsf{GenInstance}(1^\lambda, \mathcal{R}, \mathcal{R}_{\mathsf{M}}, \mathcal{M})\}$. Then, if the protocol is repeated λ times, for all instances x such that $x \notin L_{\mathcal{R}}$ and $x \in \mathcal{K} \times \mathcal{C}^2$, $\Pr[\langle \mathcal{B}(w), \mathcal{V}(z)\rangle(x) = 1]$ is negligibly close to $1/2^\lambda$, which means that the soundness probability is negligible. We define a sequence of games (Figure 3.8) played between the adversary $\mathcal{A}$ and the challenger.

Game 0: G_0 is $\mathsf{Exp}^{\mathsf{Sound}}_{\mathcal{A}}(1^\lambda, (\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1))$ for a fixed $(\mathsf{sk}, c, r_0, r_1, \mathsf{pk}, c_0, c_1)$. Considering the adversary's view, G_0 represents a real execution of the protocol PEQ. We say that "$\mathcal{A}$ wins G_0" when the output is 1.

```
**Game** $G_0$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \mathcal{R}$
$c'_b \leftarrow \mathsf{Rand}(c_b, r)$
$r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{M}})$

$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c''_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, (r, r_{\mathsf{M}}, b))$
**if** $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = r_{\mathsf{M}})$ **return** $1$
**else return** $0$
```

```
**Game** $G_1$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$b \xleftarrow{\$} \{0,1\}$
$r' \xleftarrow{\$} \mathcal{R}$
$c'_b \leftarrow \mathsf{Rand}(c, r'); r \leftarrow \mathsf{RandR}(r_b, r')$
$r'_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$
$c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r'_{\mathsf{M}})$
$r_{\mathsf{M}} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}(\mathsf{sk}, c_b), \mathsf{Dec}(\mathsf{sk}, c''_b))$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c''_b)$
$\mathsf{op} \leftarrow \mathcal{A}_3(\mathsf{st}_2, (r, r_{\mathsf{M}}, b))$
**if** $(\mathsf{Open}(\mathsf{comm}, \mathsf{op}) = r_{\mathsf{M}})$ **return** $1$
**else return** $0$
```

Figure 3.8: Sequence of games for PEQ.

Game 1: G_1 proceeds as G_0 except that we replace some of the instructions as described below.

First, we replace $r \xleftarrow{\$} \mathcal{R}$ and $c'_b \leftarrow \mathsf{Rand}(c_b, r)$ by $r' \xleftarrow{\$} \mathcal{R}$, $c'_b \leftarrow \mathsf{Rand}(c, r')$ and $r \leftarrow \mathsf{RandR}(r_b, r')$. We note that since the encryption scheme is strongly randomisable, derandomisable, and $c_b = \mathsf{Rand}(c, r_b)$: $c = \mathsf{Rand}(c_b, r_b)$ and $c'_b = \mathsf{Rand}(c_b, \mathsf{RandR}(r_b, r')) = \mathsf{Rand}(c_b, r)$. On the other hand, since the encryption scheme is perfectly strongly randomisable, an element r produced by the sequence of instructions $r' \xleftarrow{\$} \mathcal{R}; r \leftarrow \mathsf{RandR}(r_b, r')$ follows the same distribution as $r \xleftarrow{\$} \mathcal{R}$.

Subsequently, we replace the instructions $r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$ and $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r_{\mathsf{M}})$ by $r'_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_{\mathsf{M}}$, $c''_b \leftarrow \mathsf{MsgRandC}(c'_b, r'_{\mathsf{M}})$ and $r_{\mathsf{M}} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}(\mathsf{sk}, c_b), \mathsf{Dec}(\mathsf{sk}, c''_b))$. We note that since the encryption scheme is message-randomisable and message-random-extractable, the message encrypted in c''_b is indistinguishable from a message chosen at random in $\mathcal{M}$. Furthermore, c''_b is computed as $c''_b \leftarrow \mathsf{MsgRandC}(\mathsf{Rand}(c_b, r), r_\mathsf{M})$ for both, G_0 and G_1. We deduce that $\Pr[\mathcal{A} \text{ wins } G_0] = \Pr[\mathcal{A} \text{ wins } G_1]$.

Next, we claim and prove by reduction that $\left|\Pr[\mathcal{A} \text{ wins } G_1] - \frac{1}{2}\right| = \epsilon_{binding}(\lambda)/2$, where $\epsilon_{binding}(\lambda)$ is the adversary's advantage in the binding experiment of the commitment scheme (Definition 2 on page 14).

We use the following strategy. In G_1, the challenger can generate a random coin r for both $b = 1$ and $b = 0$ for the same challenge $c'' = c''_0 = c''_1$ because it builds c'' by computing $r' \xleftarrow{\$} \mathcal{R}$, $c' \leftarrow \mathsf{Rand}(c, r')$, $r'_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$, $c'' \leftarrow \mathsf{MsgRandC}(c', r'_\mathsf{M})$, $r \leftarrow \mathsf{RandR}(r_b, r')$ and $r_\mathsf{M} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}(\mathsf{sk}, c_b), \mathsf{Dec}(\mathsf{sk}, c''_b))$. Therefore, to break the protocol's soundness, the adversary must succeed for both cases $b = 1$ and $b = 0$ with non negligible probability at each round. If this is not the case, the adversary's advantage is bounded by a value that is negligibly close to $1/2^\lambda$.

We show that if such an adversary exists, we can build an algorithm that breaks the commitment scheme's binding property. More in detail, if the adversary is able to succeed the proof for both cases $b = 1$ and $b = 0$, then he is able to open its commitment comm for two different messages $z = 0$ and $z = 1$, which is equivalent to breaking the binding property. Let us define an adversary $\mathcal{B}$, which has an advantage $\epsilon_{binding}(\lambda)$ on the binding experiment as follows.

```
$\mathcal{B}(\mathsf{pp}, \mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$\mathsf{st}_1 \leftarrow \mathcal{A}_1(\mathsf{sk}, (\mathsf{pk}, c_0, c_1))$
$r' \xleftarrow{\$} \mathcal{R}; \hat{r}_0 \leftarrow \mathsf{RandR}(r_0, r'); \hat{r}_1 \leftarrow \mathsf{RandR}(r_1, r')$
$c' \leftarrow \mathsf{Rand}(c, r'); r'_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}; c'' \leftarrow \mathsf{MsgRandC}(c', r'_\mathsf{M})$
$\hat{r}^0_\mathsf{M} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}(\mathsf{sk}, c_0), \mathsf{Dec}(\mathsf{sk}, c''))$
$\hat{r}^1_\mathsf{M} \leftarrow \mathsf{MsgRandExt}(\mathsf{Dec}(\mathsf{sk}, c_1), \mathsf{Dec}(\mathsf{sk}, c''))$
$(\mathsf{comm}, \mathsf{st}_2) \leftarrow \mathcal{A}_2(\mathsf{st}_1, c'')$
$\mathsf{op}_0 \leftarrow \mathcal{A}_3(\mathsf{st}_2, (\hat{r}_0, \hat{r}^0_\mathsf{M}, b)); \mathsf{op}_1 \leftarrow \mathcal{A}_3(\mathsf{st}_2, (\hat{r}_1, \hat{r}^1_\mathsf{M}), b)$
if $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = 0$ then $\mathsf{win}_0 \leftarrow 1$
if $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1) = 1$ then $\mathsf{win}_1 \leftarrow 1$
if $\mathsf{win}_0 = 1 \wedge \mathsf{win}_1 = 1$ return $(\mathsf{comm}, \mathsf{op}_0, \mathsf{op}_1)$
```

Note that if $\mathsf{win}_0 = 1$ and $\mathsf{win}_1 = 1$, $\mathcal{B}$ wins the binding experiment since $\mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_0) = 0 \neq 1 = \mathsf{Open}(\mathsf{pp}, \mathsf{comm}, \mathsf{op}_1)$.

Using the same argument as in the soundness proof of PINEQ, we can prove that:

$$\Pr[\mathcal{A} \text{ wins } G_0]^\lambda = \Pr[\mathcal{A} \text{ wins } G_1]^\lambda \leq \left(\frac{1+\epsilon_{binding}(\lambda)}{2}\right)^\lambda$$

Since the commitment scheme has the binding property by hypothesis, $\epsilon_{binding}(\lambda)$ is negligible, so $\Pr[\mathcal{A} \text{ wins } G_0]^\lambda$, which concludes the proof of the soundness.

Zero-Knowledge. Let $\mathcal{V}^*$ be a dishonest verifier. To prove that the protocol is zero-knowledge, we define a Simulator $\mathcal{S}_{\mathcal{V}^*}(x)$ where $x = (\mathsf{pk}, c_0, c_1)$ that perfectly simulates the interaction with $\mathcal{V}^*$. The simulator $\mathcal{S}_{\mathcal{V}^*}$ generates c''_b, r_b and r^b_M as the dishonest verifier $\mathcal{V}^*$.

- If $c''_b = \mathsf{MsgRandCRand}((c_b, r_b), r^b_\mathsf{M})$ where $b \xleftarrow{\$} \{0, 1\}$, then the simulator runs $(\mathsf{comm}, \mathsf{op}) \leftarrow \mathsf{Commit}(r^b_\mathsf{M})$ and returns the transcript $(c'_b, \mathsf{comm}, (r_b, r^b_\mathsf{M}, b), \mathsf{op})$.

- If $c''_b \neq \mathsf{MsgRandCRand}((c_b, r_b), r^b_\mathsf{M})$ where $b \xleftarrow{\$} \{0,1\}$, then the simulator picks comm in the uniform distribution on the commitment set and returns the transcript $(c'_b, \mathsf{comm}, (r_b, r^b_\mathsf{M}, b), \perp)$.

The simulator follows the same distribution as the real protocol. When the verifier sends a wrong pair (r, r_M), it simulates the prover messages using a random commitment comm and the $\perp$ symbol. Indeed, since the commitment's open key remains unknown, the prover commitment is like a random commitment picked in the uniform distribution according to the perfect hiding property. □

Remark 2. In Theorems 8 and 10, zero-knowledge can be *computational* if the randomisation conditions are computational instead of perfect or if the commitment scheme being used has only computational hiding.

3.6 Non-Interactive Protocols for Plaintext Equality

We switch our attention to protocols that are ZKPoK for plaintext equality. In section 3.6.1 we focus on ZKPoK of the secret key, whereas in section 3.6.2, we focus on ZKPoK of the randomness used to generate the ciphertexts. The application is not the same because if the prover knows the secret key, the use case to consider is when the prover acts as a receiver of those ciphertexts. On the other hand, if the prover knows the randomness used to generate the ciphertexts, the use case to consider is when the prover acts as a sender.

Since sigma protocols are invariant under parallel repetition (Lemma 4 on page 22), one can apply the strong Fiat-Shamir transformation (Definition 17 on page 23) to obtain a NIZK secure in the ROM. In other words, the prover should generate λ commitments $(r_1, ..., r_\lambda)$, calculate $c \in \{0,1\}^\lambda$ as $c \leftarrow \mathcal{H}(r_1 || r_2 ... r_\lambda || x)$ for a statement x, and finally, compute the responses z_i for all r_i using the i-th bit of c. This way, the soundness error of $1/2$ is amplified to $1/2^\lambda$.

3.6.1 Protocols Based on the Secret Key

Protocols in this section additionally require the scheme to be key-randomisable. We present a protocol called MATCHPEQ (Figure 3.9), which relies on a ZKP to prove that the decryption of a given ciphertext matches a given message. Such proofs are known for numerous encryption schemes (*e.g.*, [Dam92, ElG84, GM82, Pai99]). Then, we introduce an auxiliary protocol called MATCH (Figure 3.10), which meets the requirement of MATCHPEQ (its a proof system for the above mentioned). We also present here a third protocol called SIGPEQ (Figure 3.11), which merges the two previous ones. It requires a randomisable, message-randomisable and key-randomisable scheme. However, it does not require any other protocol as a subroutine, which makes it more efficient than MATCHPEQ instantiated with MATCH. An interesting additional property of MATCHPEQ and SIGPEQ is that both can also be used to prove plaintext equality of two ciphertexts encrypted under different keys.

In protocol MATCHPEQ, the prover sends two message-randomisations to the verifier, who then challenges it on these ciphertexts. If both ciphertexts encrypt message-randomisations of the same message, then the prover can either prove

$\mathsf{Prover}((\mathsf{sk}_0, \mathsf{sk}_1), (\mathsf{pk}_0, \mathsf{pk}_1), c_0, c_1)$		$\mathsf{Verifier}((\mathsf{pk}_0, \mathsf{pk}_1), c_0, c_1)$
$r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}; (r_0, r_1) \xleftarrow{\$} \mathcal{R}^2$		
$m \leftarrow \mathsf{Dec}(\mathsf{sk}_0, c_0)$		
$c'_0 \leftarrow \mathsf{MsgRandC}(c_0, r_\mathsf{M})$		
$c'_1 \leftarrow \mathsf{MsgRandC}(c_1, r_\mathsf{M})$		
$c''_0 \leftarrow \mathsf{Rand}(c'_0, r_0)$		
$c''_1 \leftarrow \mathsf{Rand}(c'_1, r_1)$		
$m'' \leftarrow \mathsf{MsgRandM}(m, r_\mathsf{M})$		
	$\xrightarrow{(c''_0, c''_1)}$	
		$b \xleftarrow{\$} \{0,1\}$
	$\xleftarrow{b}$	
....	**if** $(b = 0)$	....
	$\xrightarrow{m''}$	

The prover runs a protocol twice (*e.g.*, MATCH) to prove the verifier that:

$\mathsf{Dec}(\mathsf{sk}_0, c''_0) = m''$ and $\mathsf{Dec}(\mathsf{sk}_1, c''_1) = m''$

The verifier accepts iff both proofs are valid.

....	**else**	....
	$\xrightarrow{(r_\mathsf{M}, r_0, r_1)}$	
		$\tilde{c}'_0 \leftarrow \mathsf{MsgRandC}(c_0, r_\mathsf{M})$
		$\tilde{c}'_1 \leftarrow \mathsf{MsgRandC}(c_1, r_\mathsf{M})$
		$\tilde{c}''_0 \leftarrow \mathsf{Rand}(\tilde{c}'_0, r_0)$
		$\tilde{c}''_1 \leftarrow \mathsf{Rand}(\tilde{c}'_1, r_1)$
		return $(\tilde{c}''_0 = c''_0) \wedge (\tilde{c}''_1 = c''_1)$

Figure 3.9: One execution of protocol $\mathsf{MATCHPEQ}$ (repeated λ times).

that it correctly did the message-randomisations or that both ciphertexts encrypt the same message.

Theorem 11. Let Π be the PKE scheme used in $\mathsf{MATCHPEQ}$. If Π is perfectly randomisable and derandomisable, perfectly message-randomisable and message-derandomisable, and if the proof in step three is instantiated by a sigma protocol that is correct, special sound, and perfectly zero-knowledge, then $\mathsf{MATCHPEQ}$ is complete, has statistical witness-extended emulation, and perfect zero-knowledge.

Proof. Completeness. Note that if c_0 and c_1 are both encryptions of the same message m, then c''_0 and c''_1 are both encryptions of m''. This is because m'' is a message-randomisation of m and the same r_M is used to compute c''_0, c''_1 and m''. It follows that if the verifier's challenge is $b = 0$, the verifier accepts both proofs and outputs accept. Similarly, if the challenge is $b = 1$, the verifier will obtain the same ciphertexts (it uses the same randomness) and so it outputs accept. We conclude that the verifier always accepts the proof.

Statistical witness-extended emulation. As this proof system is not a sigma protocol, because the verifier sends several challenges to the prover, it cannot be special sound. Hence, we prove the more general notion of statistical witness-extended emulation instead of special soundness. To prove the statistical witness-extended emulation, we use the forking lemma given in [BCC+16]: let $\mathcal{T}$ be an accepted transcript tree for our protocol, and a statement y, *i.e.*, a tree where each node is labelled by a message transmitted during the protocol, each node labelled by a prover message has 2 children labelled by two different challenges, and any path from the root to any leaf is an accepted transcript. The forking lemma given in [BCC+16] ensures that if there exists an extractor $\mathcal{E}$ such that $\Pr[w \leftarrow \mathcal{E}(x,\mathcal{T}) : (w,x) \in \mathcal{R}]$ is overwhelming, then our protocol is statistical witness-extended emulation. This can be viewed as a generalisation of proofs for special soundness: each time the verifier sends a challenge, we fork the protocol for two possible challenges.

We parse $\mathcal{T}$ as $((c''_0, c''_1), (0, m'', \mathcal{T}'), (1, (r_\mathsf{M}, r_0, r_1)))$ where $\mathcal{T}'$ is a subtree that contains transcript trees of the two proofs $\mathsf{ZK}\{\mathsf{sk}_0 : \mathsf{Dec}(\mathsf{sk}_0, c''_0) = m''\}$ and $\mathsf{ZK}\{\mathsf{sk}_1 : \mathsf{Dec}(\mathsf{sk}_1, c''_1) = m''\}$. Note that we can extract two accepted transcripts for these two proofs from $\mathcal{T}'$ such that the two transcripts have the same commitment and two different challenges.

Since our protocol is instantiated with a special sound sigma protocol, there exists an extractor $\mathcal{E}'$ that takes theses transcripts as input and returns sk_0 and sk_1 such that $\mathsf{Dec}(\mathsf{sk}_0, c''_0) = m'' = \mathsf{Dec}(\mathsf{sk}_1, c''_1)$. Our simulator $\mathcal{E}$ runs $\mathcal{E}'$ with the appropriate transcripts, receives sk_0 and sk_1, and returns $\mathsf{sk}_0, \mathsf{sk}_1$. We set $c'_0 \leftarrow \mathsf{MsgRandC}(c_0, r_\mathsf{M})$ and $c'_1 \leftarrow \mathsf{MsgRandC}(c_1, r_\mathsf{M})$. Since each path of the tree corresponds to an accepted transcript, we have that $c''_0 = \mathsf{Rand}(c'_0, r_0)$ and $c''_1 = \mathsf{Rand}(c'_1, r_1)$. In the following, we prove that $\mathsf{Dec}(\mathsf{sk}_0, c_0) = \mathsf{Dec}(\mathsf{sk}_1, c_1)$. first, from message-randomisability, we have, for $i \in \{1, 2\}$: $c'_i = \mathsf{MsgRandC}(c_i, r_\mathsf{M}) \Rightarrow \mathsf{Dec}(\mathsf{sk}_i, c'_i) = \mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_i, c_i), r_\mathsf{M})$. Moreover, using randomisability of the encryption scheme, we have: $c''_i = \mathsf{Rand}(c'_i, r_i) \Rightarrow \mathsf{Dec}(\mathsf{sk}_i, c'_i) = \mathsf{Dec}(\mathsf{sk}_i, c''_i)$. We deduce that $\mathsf{Dec}(\mathsf{sk}_i, c''_i) = \mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_i, c_i), r_\mathsf{M})$. Finally, since $\mathsf{Dec}(\mathsf{sk}_0, c''_0) = \mathsf{Dec}(\mathsf{sk}_1, c''_1)$, we have that: $\mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_0, c_0), r_\mathsf{M}) = \mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_1, c_1), r_\mathsf{M}) \Rightarrow \mathsf{MsgRandM}(\mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_0, c_0), r_\mathsf{M}), r^*_\mathsf{M}) = \mathsf{MsgRandM}(\mathsf{MsgRandM}(\mathsf{Dec}(\mathsf{sk}_1, c_1), r_\mathsf{M}), r^*_\mathsf{M}) \Rightarrow \mathsf{Dec}(\mathsf{sk}_0, c_0) = \mathsf{Dec}(\mathsf{sk}_1, c_1)$. Which concludes the proof of the statistical witness-extended emulation.

Zero-knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1, m)$. Since the two proofs used in the case $b = 0$ are zero-knowledge by hypothesis, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ that perfectly simulate the transcripts of these two proof systems. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $m'' \xleftarrow{\$} \mathcal{M}$ and $(r''_0, r''_1) \xleftarrow{\$} \mathcal{R}^2$, generates $c''_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m''; r''_0)$ and $c''_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m''; r''_1)$, then it generates two transcripts t_1 and t_2 for $\mathsf{ZK}\{\mathsf{sk}_0 : \mathsf{Dec}(\mathsf{sk}_0, c''_0) = m''\}$ and $\mathsf{ZK}\{\mathsf{sk}_1 : \mathsf{Dec}(\mathsf{sk}_1, c''_1) = m''\}$ using the simulator $\mathcal{S}_1(pk_1, c''_0, m'')$ and $\mathcal{S}_2(pk_1, c''_0, m'')$.
- If $b = 1$, the simulator picks $r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_M$ and $(r_0, r_1) \xleftarrow{\$} \mathcal{R}^2$, then it computes $c'_0 \leftarrow \mathsf{MsgRandC}(c_0, r_\mathsf{M})$, $c'_1 \leftarrow \mathsf{MsgRandC}(c_1, r_\mathsf{M})$, $c''_0 \leftarrow \mathsf{Rand}(c'_0, r_0)$ and $c''_1 \leftarrow \mathsf{Rand}(c'_1, r_1)$.

Note that in the case $b = 1$ the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(w), \mathcal{V}^*(z) \rangle(x)\}$, we define a hybrid distribution $\mathcal{H}$ that runs the real protocol, except that it replaces the instructions $c''_0 \leftarrow \mathsf{Rand}(c'_0, r_0)$ and $c''_1 \leftarrow \mathsf{Rand}(c'_1, r_1)$ by $c''_0 \leftarrow \mathsf{Enc}(\mathsf{pk}, m''; r_0)$ and $c''_1 \leftarrow \mathsf{Enc}(\mathsf{pk}, m''; r_1)$ respectively. First, we have that the real protocol is indistinguishable from the distribution $\mathcal{H}$ according to the definition of randomisability. On the other hand, the distribution $\mathcal{H}$ is indistinguishable from the distribution of the simulator $\mathcal{S}(x)$. To show that, we argue that the only difference between the two distributions is that m'' is chosen at random in the simulator, and is generated by key-randomisation algorithms on m using a random coin r_M in $\mathcal{H}$. Hence, the two distributions are indistinguishable according to the message-randomisation definition. Finally, we deduce that the real protocol and the simulator produce indistinguishable distributions, which concludes the proof. $\square$

For MATCH, we consider a setting in which the verifier has access to pk, c, m and challenges the prover on c being an encryption of m under pk. This protocol's intuition is that if the scheme is randomisable and key-randomisable, the prover can generate a new ciphertext for the same massage but under different keys. The verifier is then allowed to check that 1) the prover can generate a new ciphertext c'' which decrypts to the same message and 2) by decrypting c'' to m, conclude that the original ciphertext c is also an encryption of m.

Theorem 12. Let Π be the PKE scheme used in MATCH. If Π is perfectly randomisable perfectly key-randomisable and key-derandomisable (Definition 24 on page 32), then MATCH is complete, special sound, and perfect zero-knowledge.

```
Prover(sk, pk, c, m)                                   Verifier(pk, c, m)
$r \xleftarrow{\$} \mathcal{R}; r_\mathsf{K} \xleftarrow{\$} \mathcal{R}_\mathsf{K}$
$\mathsf{pk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_\mathsf{K})$
$\mathsf{sk}' \leftarrow \mathsf{KeyRandSk}(\mathsf{sk}, r_\mathsf{K})$
$c' \leftarrow \mathsf{KeyRandC}(c, r_\mathsf{K})$
$c'' \leftarrow \mathsf{Rand}(c', r)$
                        $\xrightarrow{(\mathsf{pk}', c'')}$
                                                       $b \xleftarrow{\$} \{0,1\}$
                        $\xleftarrow{b}$
if $(b = 0)$ then $z \leftarrow \mathsf{sk}'$
else $z \leftarrow (r, r_\mathsf{K})$
                        $\xrightarrow{z}$
                                                       if $b = 1$ then
                                                         $\widetilde{\mathsf{pk}}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_\mathsf{K})$
                                                         $\tilde{c}' \leftarrow \mathsf{KeyRandC}(c, r_\mathsf{K})$
                                                         $\tilde{c}'' \leftarrow \mathsf{Rand}(\tilde{c}', r)$
                                                         return $(\tilde{c}'' = c'') \wedge (\widetilde{\mathsf{pk}}' = \mathsf{pk}')$
                                                       else return $(m = \mathsf{Dec}(sk', c''))$
```

Figure 3.10: One execution of protocol MATCH (repeated λ times).

Proof. Completeness. Since $c \in \mathsf{Enc}(\mathsf{pk}, m)$, when the prover randomises c into c'' after randomising the keys, it holds that $c'' \in \mathsf{Enc}(\mathsf{pk}, m)$. It follows that when the verifier's challenge is $b = 1$, the verifier always accepts the proof. Similarly, when the verifier's challenge is $b = 0$, upon receiving the randomness used by the prover, the verifier can check that indeed $\widetilde{c}'' = c''$ and that $\widetilde{\mathsf{pk}}' = \mathsf{pk}'$, so it always accepts the proof as well.
Special Soundness. Let the two following transcripts t_0 $=((\mathsf{pk}', c''), 0,\ (r, r_{\mathsf{K}}))$ and t_1 $=((\mathsf{pk}', c''), 1, \mathsf{sk}'))$ be for the statement $y = (\mathsf{pk}, c, m)$. We assume that t_0 and t_1 are transcripts of accepted proofs, *i.e.,* $\mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{K}})$, $c' = \mathsf{KeyRandC}(c, r_{\mathsf{K}})$, $c'' = \mathsf{Rand}(c', r)$ and $m = \mathsf{Dec}(\mathsf{sk}', c'')$. We define the knowledge extractor as follows: $\mathcal{E}(y, t, t')$ returns $\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{sk}', r^*_{\mathsf{K}})$. We first note that this algorithm is polynomial-time since computing r^*_{K} from r and running $\mathsf{KeyRandSk}$ are polynomial-time operations. In the following, we prove that $m = \mathsf{Dec}(\mathsf{sk}, c)$. First, we have: $c'' = \mathsf{Rand}(c', r) \Rightarrow \mathsf{Dec}(\mathsf{sk}', c') = \mathsf{Dec}(\mathsf{sk}', c'')$. On the other hand, we have $\mathsf{sk} = \mathsf{KeyRandSk}(\mathsf{sk}', r^*_{\mathsf{K}})$, so $\mathsf{sk}' = \mathsf{KeyRandSk}(\mathsf{sk}, r_{\mathsf{K}})$, we deduce: $\mathsf{pk}' = \mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{K}}) \ \wedge\ c' = \mathsf{KeyRandC}(c, r_{\mathsf{K}}) \Rightarrow \mathsf{Dec}(\mathsf{sk}', c') = \mathsf{Dec}(\mathsf{sk}, c)$. Finally, we have: $m = \mathsf{Dec}(\mathsf{sk}', c'') = \mathsf{Dec}(\mathsf{sk}', c') = \mathsf{Dec}(\mathsf{sk}, c)$, which concludes the special soundness proof.
Zero-knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}, c, m)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $r \xleftarrow{\$} \mathcal{R}$, generates $(\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KGen}(1^k)$ and runs $c'' \leftarrow \mathsf{Enc}(\mathsf{pk}', m; r)$, then it returns the transcript $t = ((\mathsf{pk}', c''), 0, \mathsf{sk}'))$.
- If $b = 1$, the simulator picks $r_{\mathsf{K}} \xleftarrow{\$} \mathcal{R}_{\mathsf{K}}$ and $r \xleftarrow{\$} \mathcal{R}$, generates $\mathsf{pk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}, r_{\mathsf{K}}); \mathsf{sk}' \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}, r_{\mathsf{K}}); c' \leftarrow \mathsf{KeyRandC}(c, r_{\mathsf{K}})$; and $c'' \leftarrow \mathsf{Rand}(c', r)$, then it returns $t = ((\mathsf{pk}', c''), 1, (r, r_{\mathsf{K}}))$.

Note that in the case $b = 1$, the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)\}$ when $b = 0$, we define a hybrid distribution $\mathcal{H}$ that runs the real protocol, except that it replaces the instruction $c'' = \mathsf{Rand}(c', r)$ by $c'' = \mathsf{Enc}(\mathsf{pk}', m)$. First, we have that the real protocol is indistinguishable from the distribution $\mathcal{H}$ according to the definition of randomisability. On the other hand, the distribution $\mathcal{H}$ is indistinguishable from the distribution of the simulator $\mathcal{S}(x)$. To show that, we argue that the only difference between the two distributions is that $(\mathsf{sk}', \mathsf{pk}')$ are generated from the key generation algorithm in the simulator, and by key-randomisation algorithms on $(\mathsf{sk}, \mathsf{pk})$ in $\mathcal{H}$. Hence, the two distributions are indistinguishable according to the key-randomisation definition. Finally, we deduce that the real protocol and the simulator produce indistinguishable distributions. □

To conclude this section, we present the protocol SIGPEQ, a sigma protocol for plaintext equality of two ciphertexts built upon the previous ones. In this protocol, the prover performs a message-randomisation on the ciphertexts and a key-randomisation to obtain new ciphertexts. These ciphertexts decrypt to the same message m' but under a different key. Once the prover sends the public keys and the new ciphertexts to the verifier, the verifier challenges the prover. The intuition behind the challenge is that if the two ciphertexts obtained by the verifier are message-randomisations of the same message, then the prover should be able to provide either the corresponding secret key to confirm it or the randomness used to verify the procedure. This protocol is more efficient because it requires exactly λ rounds while MATCHPEQ requires λ rounds times the number of rounds of MATCH.

```
Prover$(\mathsf{sk}_0, \mathsf{sk}_1, \mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$                Verifier$(\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$
$(r_\mathsf{K}^1, r_\mathsf{K}^2) \xleftarrow{\$} \mathcal{R}_\mathsf{K}^2$
$r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M};\ (r_0, r_1) \xleftarrow{\$} \mathcal{R}^2$
$c_0' \leftarrow \mathsf{Rand}(c_0, r_0)$
$c_1' \leftarrow \mathsf{Rand}(c_1, r_1)$
$\mathsf{pk}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_0, r_\mathsf{K}^1)$
$\mathsf{sk}_0' \leftarrow \mathsf{KeyRandSk}(\mathsf{sk}_0, r_\mathsf{K}^1)$
$\mathsf{pk}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_\mathsf{K}^2)$
$\mathsf{sk}_1' \leftarrow \mathsf{KeyRandSk}(\mathsf{sk}_1, r_\mathsf{K}^2)$
$c_0'' \leftarrow \mathsf{KeyRandC}(c_0', r_\mathsf{K}^1)$
$c_1'' \leftarrow \mathsf{KeyRandC}(c_1', r_\mathsf{K}^2)$
$c_0''' \leftarrow \mathsf{MsgRandC}(c_0'', r_\mathsf{M})$
$c_1''' \leftarrow \mathsf{MsgRandC}(c_1'', r_\mathsf{M})$
$C \leftarrow (\mathsf{pk}_0', \mathsf{pk}_1', c_0''', c_1''')$
                                   $\xrightarrow{C}$
                                                  $b \xleftarrow{\$} \{0, 1\}$
                                   $\xleftarrow{b}$
if $(b = 0)$ $z \leftarrow (\mathsf{sk}_0', \mathsf{sk}_1')$
else $z \leftarrow (r_0, r_1, r_\mathsf{K}^1, r_\mathsf{K}^2, r_\mathsf{M})$
                                   $\xrightarrow{z}$
                                                  if $b = 1$
                                                    $\tilde{c}_0' \leftarrow \mathsf{Rand}(c_0, r_0)$
                                                    $\tilde{c}_1' \leftarrow \mathsf{Rand}(c_1, r_1)$
                                                    $\widetilde{\mathsf{pk}}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_0, r_\mathsf{K}^1)$
                                                    $\widetilde{\mathsf{pk}}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_\mathsf{K}^2)$
                                                    $\tilde{c}_0'' \leftarrow \mathsf{KeyRandC}(\tilde{c}_0', r_\mathsf{K}^1)$
                                                    $\tilde{c}_1'' \leftarrow \mathsf{KeyRandC}(\tilde{c}_1', r_\mathsf{K}^2)$
                                                    $\tilde{c}_0''' \leftarrow \mathsf{MsgRandC}(\tilde{c}_0'', r_\mathsf{M})$
                                                    $\tilde{c}_1''' \leftarrow \mathsf{MsgRandC}(\tilde{c}_1'', r_\mathsf{M})$
                                                    return $((\tilde{c}_0''' = c_0''') \wedge (\tilde{c}_1''' = c_1''')$
                                                      $\wedge\ (\widetilde{\mathsf{pk}}_0' = \mathsf{pk}_0') \wedge (\widetilde{\mathsf{pk}}_1' = \mathsf{pk}_1'))$
                                                  return $\mathsf{Dec}(\mathsf{sk}_0', c_0''') = \mathsf{Dec}(\mathsf{sk}_1', c_1''')$
```

Figure 3.11: One execution of protocol SIGPEQ (repeated λ times).

Theorem 13. Let Π be the PKE scheme used in SIGPEQ. If Π is perfectly randomisable, perfectly message-randomisable and perfectly key-randomisable, then SIGPEQ is complete, special sound, and perfect zero-knowledge.

Proof. Completeness. Since the scheme is perfectly randomisable, perfectly message-randomisable and key-randomisable the ciphertexts c_0''' and c_1''' will both decrypt using the key sk_0 (resp. sk_1) to the same message m', which is a randomisation of m with r_M. We conclude that all the verifier needs to do is check the procedures following the same steps, and so it always accepts the proof.
Special soundness. Let the two following transcripts t_0 $=((\mathsf{pk}', c_0''', c_1'''), 0, (r_0, r_1, r_\mathsf{K}^1, r_\mathsf{K}^2, r_\mathsf{M}))$ and $t_1 = ((\mathsf{pk}', c_0''', c_1'''), 1, (\mathsf{sk}_0', \mathsf{sk}_1'))$ for the statement $y = (\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$. We assume that t_0 and t_1 are transcripts of accepted proofs, *i.e.,* computing:

- $\tilde{c}_0' \leftarrow \mathsf{Rand}(c_0, r_0)$ and $\tilde{c}_1' \leftarrow \mathsf{Rand}(c_1, r_1)$,
- $\widetilde{\mathsf{pk}}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_0, r_\mathsf{K}^1)$ and $\widetilde{\mathsf{pk}}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_\mathsf{K}^2)$,
- $\tilde{c}_0'' \leftarrow \mathsf{KeyRandC}(\tilde{c}_0', r_\mathsf{K}^1)$ and $\tilde{c}_1'' \leftarrow \mathsf{KeyRandC}(\tilde{c}_1', r_\mathsf{K}^2)$,
- $\tilde{c}_0''' \leftarrow \mathsf{MsgRandC}(\tilde{c}_0'', r_\mathsf{M})$ and $\tilde{c}_1'' \leftarrow \mathsf{MsgRandC}(\tilde{c}_1'', r_\mathsf{M})$

we have:

- $(\tilde{c}_0''' = c_0''')$ and $(\tilde{c}_1''' = c_1''')$,
- $(\widetilde{\mathsf{pk}}_0' = \mathsf{pk}_0')$ and $(\widetilde{\mathsf{pk}}_1' = \mathsf{pk}_1')$, and
- $\mathsf{Dec}(\mathsf{sk}_0', c_0''') = \mathsf{Dec}(\mathsf{sk}_1', c_1''')$

We define the knowledge extractor as follows: $\mathcal{E}(y, t, t')$ returns $(\mathsf{sk}_0^*, \mathsf{sk}_1^*)$ such that $\mathsf{sk}_0^* = \mathsf{KeyRandSk}(\mathsf{sk}_0', r_\mathsf{K}^{1^*})$ and $\mathsf{sk}_1^* = \mathsf{KeyRandSk}(\mathsf{sk}_1', r_\mathsf{K}^{2^*})$. We first note that this algorithm is polynomial-time since computing $r_\mathsf{K}^{1^*}$ and $r_\mathsf{K}^{2^*}$from r_K^1 and r_K^2, and running $\mathsf{KeyRandSk}$ are polynomial-time operations. In the following, we prove that $\mathsf{Dec}(\mathsf{sk}_0^*, c_0) = \mathsf{Dec}(\mathsf{sk}_1^*, c_1)$. We set $m''' = \mathsf{Dec}(\mathsf{sk}_0^*, c_0''')$, then we have $m''' = \mathsf{Dec}(\mathsf{sk}_1^*, c_1''')$. We set $m_0'' = \mathsf{Dec}(\mathsf{sk}_0^*, \tilde{c}_0'')$ and $m_1'' = \mathsf{Dec}(\mathsf{sk}_1^*, \tilde{c}_1'')$. For all $i \in \{0, 1\}$, we have:

$$\tilde{c}_i''' = \mathsf{MsgRandC}(\tilde{c}_i'', r_\mathsf{M}) \Rightarrow m''' = \mathsf{Dec}(\mathsf{sk}_i', c_i''') = \mathsf{MsgRandM}(m_i'', r_\mathsf{M})$$

We deduce that $\mathsf{MsgRandM}(m_0'', r_\mathsf{M}) = \mathsf{MsgRandM}(m_1'', r_\mathsf{M})$, and since $\mathsf{MsgRandM}(\cdot, r_\mathsf{M})$ is bijective, we have that $m_0'' = m_1''$. We set $m'' = m_0''$. Moreover, since $\mathsf{sk}_i^* = \mathsf{KeyRandSk}(\mathsf{sk}_i', r_\mathsf{K}^{i^*})$ we have $\mathsf{sk}_i' = \mathsf{KeyRandSk}(\mathsf{sk}_i^*, r_\mathsf{K}^i)$, so:

$$\tilde{c}_i'' = \mathsf{KeyRandC}(\tilde{c}_i', r_\mathsf{K}^i) \Rightarrow m'' = \mathsf{Dec}(\mathsf{sk}_i', \tilde{c}_i'') = \mathsf{Dec}(\mathsf{sk}_i^*, \tilde{c}_i')$$

We deduce that $\mathsf{Dec}(\mathsf{sk}_0^*, \tilde{c}_i') = \mathsf{Dec}(\mathsf{sk}_1^*, \tilde{c}_1')$. Finally, we have:

$$\tilde{c}_i' = \mathsf{Rand}(c_i, r_i) \Rightarrow m'' = \mathsf{Dec}(\mathsf{sk}_i^*, \tilde{c}_i') = \mathsf{Dec}(\mathsf{sk}_i^*, c_i)$$

We deduce that $\mathsf{Dec}(\mathsf{sk}_0^*, c_1) = \mathsf{Dec}(\mathsf{sk}_1^*, c_1)$, which conclude the proof of the special soundness.
Zero-knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1, m)$. The simulator $\mathcal{S}$ picks $b \overset{\$}{\leftarrow} \{0, 1\}$, then:

- If $b = 0$, the simulator picks $m'' \overset{\$}{\leftarrow} \mathcal{M}$, generates $(r_0, r_1) \overset{\$}{\leftarrow} \mathcal{R}$ the keys $(\mathsf{pk}_0', \mathsf{sk}_0') \overset{\$}{\leftarrow} \mathsf{KGen}(1^k)$ and $(\mathsf{pk}_1', \mathsf{sk}_1') \overset{\$}{\leftarrow} \mathsf{KGen}(1^k)$, and computes $c_0''' \leftarrow \mathsf{Enc}(\mathsf{pk}_0', m', r_0)$ and $c_1''' \leftarrow \mathsf{Enc}(\mathsf{pk}_1', m', r_1)$.
  It then returns $((\mathsf{pk}_0', \mathsf{pk}_1', c_0''', c_1'''), 0, (\mathsf{sk}_0', \mathsf{sk}_1'))$.
- If $b = 1$, the simulator picks $(r_\mathsf{K}^1, r_\mathsf{K}^1) \overset{\$}{\leftarrow} \mathcal{R}_K^2$, $r_\mathsf{M} \overset{\$}{\leftarrow} \mathcal{R}_M$, $(r_0, r_1) \overset{\$}{\leftarrow} \mathcal{R}^2$ and

generates $c_0' \leftarrow \mathsf{Rand}(c_0, r_0)$, $c_1' \leftarrow \mathsf{Rand}(c_1, r_1)$, $\mathsf{pk}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_0, r_{\mathsf{K}}^1)$, $\mathsf{pk}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_{\mathsf{K}}^2)$, $c_0'' \leftarrow \mathsf{KeyRandC}(c_0', r_{\mathsf{K}}^1)$, $c_1'' \leftarrow \mathsf{KeyRandC}(c_1', r_{\mathsf{K}}^2)$, $c_0''' \leftarrow \mathsf{MsgRandC}(c_0'', r_{\mathsf{M}})$, and $c_1''' \leftarrow \mathsf{MsgRandC}(c_1'', r_{\mathsf{M}})$.
It then returns $((\mathsf{pk}_0', \mathsf{pk}_1', c_0''', c_1'''), 1, (r_0, r_1, r_{\mathsf{K}}^1, r_{\mathsf{K}}^2, r_{\mathsf{M}}))$.

Note that in the case $b = 1$, the simulator follows the same steps as in the real protocol, so it is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)\}$, we define the following hybrid distributions:

- $\mathcal{H}_1$ that runs the real protocol, except that it replaces the instructions: $c_0' \leftarrow \mathsf{Rand}(c_0, r_0)$ and $c_1' \leftarrow \mathsf{Rand}(c_1, r_1)$ by: $c_0' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r_0)$ and $c_1' \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r_1)$. The distribution of $\mathcal{H}_1$ is indistinguishable from the distribution induced by the real protocol. To show that, we argue that the only difference between the two distributions is that:
  - **in the real protocol**, c_0' and c_1' are randomised.
  - **in** $\mathcal{H}_1$ c_0' and c_1' are fresh ciphertexts of the same message m as c_0 and c_1 in $\mathcal{H}_1$.

  Therefore, the two distributions are indistinguishable according to the (perfect) randomisability definition.
- $\mathcal{H}_2$ that runs the same protocol as $\mathcal{H}_1$, except that it replaces the instructions: $\mathsf{pk}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_0, r_{\mathsf{K}}^1)$, $\mathsf{sk}_0' \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}_0, r_{\mathsf{K}}^1)$, $\mathsf{pk}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{pk}_1, r_{\mathsf{K}}^2)$, $\mathsf{sk}_1' \leftarrow \mathsf{KeyRandPk}(\mathsf{sk}_1, r_{\mathsf{K}}^2)$, $c_0'' \leftarrow \mathsf{KeyRandC}(c_0', r_{\mathsf{K}}^1)$ and $c_1'' \leftarrow \mathsf{KeyRandC}(c_1', r_{\mathsf{K}}^2)$ by $(\mathsf{pk}_0', \mathsf{sk}_0') \xleftarrow{\$} \mathsf{KGen}(1^k)$, $(\mathsf{pk}_1', \mathsf{sk}_1') \xleftarrow{\$} \mathsf{KGen}(1^k)$, $c_0'' \leftarrow \mathsf{Enc}(\mathsf{pk}', m; r_0)$, and $c_1'' \leftarrow \mathsf{Enc}(\mathsf{pk}', m; r_1)$. The distribution of $\mathcal{H}_2$ is indistinguishable from the distribution of $\mathcal{H}_2$. To show that, we argue that the only difference between the two distributions is that:
  - **in** $\mathcal{H}_1$ pk_0', pk_1', sk_0', sk_1', c_0'' and c_1'' are obtained by randomisation of the key using the same coin r_{K} in $\mathcal{H}_1$.
  - **in** $\mathcal{H}_2$ (i) $(\mathsf{pk}_0', \mathsf{sk}_1')$ and $(\mathsf{pk}_0', \mathsf{sk}_1')$ are fresh keys and (ii) c_0'' and c_1'' are obtained using the same message m and the same coins (r_0, r_1) as c_0' and c_1' but using the fresh public keys pk_0' and pk_1'.

  Therefore, the two distributions are indistinguishable according to the (perfect) key-randomisation definition.
- $\mathcal{H}_3$ that runs the same protocol as $\mathcal{H}_2$, except that it replaces the instructions: $c_0''' \leftarrow \mathsf{MsgRandC}(c_0'', r_{\mathsf{M}})$ and $c_1''' \leftarrow \mathsf{MsgRandC}(c_1'', r_{\mathsf{M}})$ by: $m' \xleftarrow{\$} \mathcal{M}$, $c_0''' \leftarrow \mathsf{Enc}(\mathsf{pk}', m'; r_0)$, and $c_1''' \leftarrow \mathsf{Enc}(\mathsf{pk}', m'; r_1)$. Note that $\mathcal{H}_3$ and $\mathcal{S}(x)$ induce the same distribution. Moreover, the distribution of $\mathcal{H}_3$ is indistinguishable from the distribution of $\mathcal{H}_2$. To show that, we argue that the only difference between the two distributions is that:
  - **in** $\mathcal{H}_2$ m', c_0''' and c_1''' are obtained randomising the message in m, c_0'' and c_1'' using the same coin r_{M}.
  - **in** $\mathcal{H}_3$ m' is a fresh message encrypted in c_0''' and c_1''' using the same public-key pk' and the same random coins (r_0, r_1) as in c_0'' and c_1''.

  Therefore, the two distributions are indistinguishable according to the (perfect) message-randomisation definition.

□

```
$\mathsf{Prover}(r_0, r_1, \mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$                $\mathsf{Verifier}(\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$
$r_\mathsf{M} \xleftarrow{\$} \mathcal{R}_\mathsf{M}$; $(r'_0, r'_1) \xleftarrow{\$} \mathcal{R}^2$
$r''_0 \leftarrow \mathsf{RandR}(r_0, r'_0)$
$r''_1 \leftarrow \mathsf{RandR}(r_1, r'_1)$
$c'_0 \leftarrow \mathsf{Rand}(c_0, r'_0)$
$c'_1 \leftarrow \mathsf{Rand}(c_1, r'_1)$
$c''_0 \leftarrow \mathsf{MsgRandC}(c'_0, r_\mathsf{M})$
$c''_1 \leftarrow \mathsf{MsgRandC}(c'_1, r_\mathsf{M})$
                              $\xrightarrow{(c''_0, c''_1)}$
                                                 $b \xleftarrow{\$} \{0, 1\}$
                              $\xleftarrow{b}$
$\mathbf{if}\ (b = 0)\ z \leftarrow (r''_0, r''_1)$
$\mathbf{else}\ z \leftarrow (r'_0, r'_1, r_\mathsf{M})$
                              $\xrightarrow{z}$
                                                 $\mathbf{if}\ b = 0\ \mathbf{return}$
                                                   $\mathsf{CDec}(\mathsf{pk}_0, c''_0, r''_0) = \mathsf{CDec}(\mathsf{pk}_1, c''_1, r''_1)$
                                                 $\mathbf{else}$
                                                   $\tilde{c}'_0 \leftarrow \mathsf{Rand}(c_0, r'_0)$
                                                   $\tilde{c}'_1 \leftarrow \mathsf{Rand}(c_1, r'_1)$
                                                   $\tilde{c}''_0 \leftarrow \mathsf{MsgRandC}(\tilde{c}'_0, r_\mathsf{M})$
                                                   $\tilde{c}''_1 \leftarrow \mathsf{MsgRandC}(\tilde{c}'_1, r_\mathsf{M})$
                                                   $\mathbf{return}\ (\tilde{c}''_0 = c''_0) \wedge (\tilde{c}''_1 = c''_1)$
```

Figure 3.12: One execution of protocol RSPEQ (repeated λ times).

3.6.2 Protocols Based on the Encryption Randomness

Based on the previous ideas, we now present the protocol RSPEQ (Figure 3.12). It requires a random coin decryptable, strong randomisable and message-randomisable scheme. Similarly to SIGPEQ, the verifier challenges the prover to either provide the randomisers or to allow it to check the procedure.

Theorem 14. Let Π be the PKE scheme used in RSPEQ. If Π is perfectly strong randomisable, random-extractable, perfectly message-randomisable and random coin decryptable (Definition 10 on page 19), then RSPEQ is complete, special sound, and perfect zero-knowledge.

Proof. Completeness. Since the scheme is perfectly strong randomisable, perfectly message-randomisable and random coin decryptable, the ciphertexts c''_0 and c''_1 will both decrypt using the key r''_0 (resp. r''_1) to the same message m', which is a randomisation of m with r_M. We conclude that all the verifier needs to do is to check the procedure following the same steps, and so it always accepts the proof.
Special Soundness. Let $t_0 = ((c''_0, c''_1), 0, (r''_0, r''_1))$ and $t_1 = ((c''_0, c''_1), 1, (r'_0, r'_1, \mathsf{r_M}))$ be two transcripts of accepted proofs for the statement $y = (\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$. We define the knowledge extractor $\mathcal{E}(y, t, t')$ as follows: it parses t_0 and t_1 and returns $r^*_0 \leftarrow \mathsf{RandExt}(r'_0, r''_0)$ and $r^*_1 \leftarrow \mathsf{RandExt}(r'_1, r''_1)$. Since the scheme is random-extractable and a RCD-PKE, we have that $\mathsf{CDec}(\mathsf{pk}_0, c_0, r^*_0) = \mathsf{CDec}(\mathsf{pk}_1, c_1, r^*_1)$

which concludes the proof of special soundness.
Zero-knowledge. We define the simulator $\mathcal{S}(x)$ where $x = (\mathsf{pk}_0, \mathsf{pk}_1, c_0, c_1)$. The simulator $\mathcal{S}$ picks $b \xleftarrow{\$} \{0,1\}$, then:

- If $b = 0$, the simulator picks $m' \xleftarrow{\$} \mathcal{M}$, $(r_0, r_1) \xleftarrow{\$} \mathcal{R}$, computes $c_0'' \leftarrow \mathsf{Enc}(\mathsf{pk}_0, m', r_0)$ and $c_1'' \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m', r_1)$. It then returns $((c_0'', c_1''), 0, (r_0, r_1))$.
- If $b = 1$, the simulator picks $r_{\mathsf{M}} \xleftarrow{\$} \mathcal{R}_M$, $(r_0', r_1') \xleftarrow{\$} \mathcal{R}^2$, and generates $c_0' \leftarrow \mathsf{Rand}(c_0, r_0')$, $c_1' \leftarrow \mathsf{Rand}(c_1, r_1')$, $c_0'' \leftarrow \mathsf{MsgRandC}(c_0', r_{\mathsf{M}})$, and $c_1'' \leftarrow \mathsf{MsgRandC}(c_1', r_{\mathsf{M}})$. It then returns $((c_0'', c_1''), 1, (r_0', r_1', r_{\mathsf{M}}))$.

Note that in the case $b = 1$, the simulator follows the same steps as in the real protocol, so the protocol is perfectly simulated. To prove that the simulator follows the same distribution as the real protocol $\{\langle \mathcal{P}(w), \mathcal{V}^*(z)\rangle(x)\}$, we define the following hybrid distribution:

- $\mathcal{H}_1$ that runs the real protocol, except that it replaces the instructions: $r_0'' \leftarrow \mathsf{RandR}(r_0, r_0')$, $r_1'' \leftarrow \mathsf{RandR}(r_1, r_1')$, $c_0' \leftarrow \mathsf{Rand}(c_0, r_0')$ and $c_1' \leftarrow \mathsf{Rand}(c_1, r_1')$ by: $c_0' \leftarrow \mathsf{Enc}(\mathsf{pk}_0, m; r_0'')$, $c_1' \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m; r_1'')$. We argue that $\mathcal{H}_1$ is indistinguishable from the distribution induced by the real protocol. The only difference between both distributions is that **in the real protocol**, c_0' and c_1' are randomised with r_0'' and r_1'' computed by RandR, whereas **in** $\mathcal{H}_1$, c_0' and c_1' are fresh ciphertexts of the same message m. Such difference is indistinguishable according to the perfectly strong randomisable definition.
- $\mathcal{H}_2$ that runs the same protocol as $\mathcal{H}_1$, except that it replaces the instruction: $c_0'' \leftarrow \mathsf{MsgRandC}(c_0', r_{\mathsf{M}})$ and $c_1'' \leftarrow \mathsf{MsgRandC}(c_1', r_{\mathsf{M}})$ by: $m' \xleftarrow{\$} \mathcal{M}$, $c_0'' \leftarrow \mathsf{Enc}(\mathsf{pk}_0, m'; r_0'')$, and $c_1'' \leftarrow \mathsf{Enc}(\mathsf{pk}_1, m'; r_1'')$. Note that $\mathcal{H}_2$ and $\mathcal{S}(x)$ induce the same distribution. Moreover, the distribution of $\mathcal{H}_2$ is indistinguishable from the distribution of $\mathcal{H}_1$. To show that, we argue that the only difference between the two distributions is that:
  - **in** $\mathcal{H}_1$ m', c_0'' and c_1'' are obtained by randomising the message m in c_0'' and c_1'' using the same r_{M}.
  - **in** $\mathcal{H}_2$ m' is a fresh message encrypted in c_0'' and c_1'' using the random coins (r_0'', r_1'').

  It follows that the two distributions are indistinguishable according to the (perfect) message-randomisation definition.

□

3.7 Implementation and Evaluation

We implemented the protocols HPEQ, PEQ, HPINEQ, PINEQ, RSPEQ and SIGPEQ in Rust using the dalek library and ElGamal (see [BBLPK21a]). The implementation was done for academic purposes and simulating the interaction between a prover and a verifier. More in detail, we show in Table 3.2 the average running times using a Macbook Air (Chip M2 & 16GB RAM) with no extra optimizations and considering a security parameter $\lambda = 128$. We stress that the times shown consist of 128 *repetitions* for each protocol run so to achieve the desired soundness error. This information was gathered using the external crate *bencher*.

We now compare the efficiency of our protocols with the best (as far as we know) custom protocols for ElGamal that achieve the same security properties. Our

Protocol	HPEQ	PEQ	HPINEQ	PINEQ	RSPEQ	SIGPEQ
Avg. time	11.50	29.73	10.68	28.90	25.47	47.81
Deviation	0.02	0.42	0.02	0.03	0.54	1.74

Table 3.2: Running times in ms for different protocols using ElGamal.

	Equality proofs			Inequality proofs	
	[CP93]	PEQ	RSPEQ	[CS03]	PINEQ
Prover	**2**	6	4	**6**	**6**
Verifier	**2**	4	4	**4**	**4**
Rounds	**3**	4	3	**3**	4

Table 3.3: Number of exp. and rounds for plaintext equality/inequality proofs.

generic protocols PEQ, RSPEQ, and PINEQ are perfect zero-knowledge and do not rely on the ROM. Note that more efficient protocols exist under weaker hypothesis. For example, HVZK proofs can be done using Schnorr-like protocols [Sch90], non-interactive protocols can be done in the ROM replacing the challenge by the hash of the commitment, and non-interactive but computationally zero-knowlege proofs can be done using the Groth-Sahai construction from pairings [GS08].

Proving the equality of two ElGamal plaintexts (M_1, M_2) given two ciphertexts $c_0 = \mathsf{Enc}(\mathsf{pk}, M_1; r_0) = (r_0P, M_1{+}r_0\mathsf{pk})$ and $c_0 = \mathsf{Enc}(\mathsf{pk}, M_2; r_1) = (r_1P, M_2{+}r_1\mathsf{pk})$ is equivalent to proving that $(\alpha P, (r_0 - r_1)P, \alpha(r_0 - r_1)P)$ is a Diffie-Hellman tuple, which can be efficiently done with the Chaum-Pedersen protocol [CP93] (using either the secret key or the randomness as the witness). Similarly, proving the inequality of the two plaintexts is equivalent to prove that $(\alpha P, (r_0 - r_1)P, M_1 + \alpha(r_0 - r_1)P - M_2)$ is not a Diffie-Hellman tuple, which can be efficiently done with the Camenisch-Shoup protocol [CS03]. These protocols must be repeated λ times for a security parameter λ, like ours. Table 3.3 gives the number of exponentiations (the dominant operation in all the considered protocols) and rounds for a single run of each protocol. This comparison suggests that our generic protocols' cost is reasonable for perfect zero-knowledge protocols in the standard model.

3.8 Conclusions and Future Work

We characterised malleability in terms of randomisability, message-randomisability and key-randomisability for PKE. We defined and presented interactive and non-interactive ZKP based on those notions for plaintext equality and inequality. As a result, we obtained generic protocols that can be instantiated with different PKE schemes. Furthermore, we provided examples of PKE schemes with different properties and secure under different security models. Future work could explore the design of non-interactive protocols for plaintext inequality. Proposing protocols that do not require k rounds from a generic encryption scheme would contribute to the topic. Finally, a related idea that could enable more use cases in some settings would be to construct a generic "plaintext inequality test" to prove that a ciphertext's plaintext is smaller or greater than another plaintext.

4
Structure-Preserving Signatures on Equivalence Classes

...la science est l'œuvre de l'esprit humain, qui est plutôt destiné à étudier qu'à connaître, à chercher qu'à trouver la vérité.

— Évariste Galois

Structure-Preserving Signatures (SPS) [AFG+10] are digital signatures in which all messages, signatures, and public keys are elements in the source groups of a bilinear group. Moreover, the signature verification algorithm tests group membership and evaluates Pairing Product Equations (PPE). As described in [AFG+10, AGOT14, AFG+16], SPS can easily support signatures on vectors of group elements and present interesting randomisation properties. Alongside Groth-Sahai proofs [GS08], they provide a rich framework and have served to build efficient and modular constructions of cryptographic protocols since their introduction. Originally proposed by Hanser and Slamanig [HS14], Structure-Preserving Signatures on Equivalence Classes introduce a novel type of SPS. In chapters 5 and 6, these signatures are used to build anonymous credentials schemes. However, unlike Chapter 6, Chapter 5 takes a more theoretical approach, focusing on efficient constructions without relying on the GGM. In this regard, the main contribution of this chapter, based on joint work with Aisling Connolly and Pascal Lafourcade [CLPK21], is the proposal of new SPS-EQ constructions under standard assumptions, which are later used in Chapter 5.

4.1 Introduction

In [HS14], Hanser and Slamanig observed that if one considers a prime-order group $\mathbb{G}$ and defines the projective vector space $(\mathbb{G}^*)^\ell$, there is a partition into equivalence classes given by the following relation $\mathcal{R}$: $\boldsymbol{m} \in (\mathbb{G}^*)^\ell \sim_{\mathcal{R}} \boldsymbol{m}^* \in (\mathbb{G}^*)^\ell \iff \exists\ \mu \in \mathbb{Z}_p^* : \boldsymbol{m}^* = \mu \boldsymbol{m}$. Suppose the discrete logarithm problem is hard in $\mathbb{G}$, and one restricts the vector components to be non-zero. Given two vectors $\boldsymbol{m}$ and $\boldsymbol{m}^*$, it is difficult to distinguish whether they were randomly sampled or if they belong to the same equivalence class. Unless μ is known, the only way to check class membership is to pick a constant value and repeatedly apply the group operation with one of the vectors to check if the other one can be obtained. On the contrary, if the discrete logarithm of the components is known for both vectors, one can check the distance (component-pairwise) between them and conclude that they belong to the same equivalence class if it does not change (it is invariant in the class). Hence, Hanser and Slamanig defined SPS-EQ as SPS that produces signatures on an equivalence class instead of messages alone. Given a message and its corresponding signature, SPS-EQ provides a *controlled form of malleability* in which one can publicly (without requiring access to the secret key) adapt a signature to change the representative (message). The main challenge is guaranteeing that two message-signature pairs from the same equivalence class cannot be linked. The equivalence relation provides indistinguishability on the message space if the DDH assumption holds. If additionally, updated signatures are distributed like fresh signatures, message-signature pairs falling into the same class are unlinkable. For unlinkability to hold, signatures should also be randomised when adapting them to a new class representative. As described in [FHS19], given a representative and its corresponding signature, a random representative of the same class with an adapted signature are indistinguishable from a random message-signature pair.

SPS-EQ have been used to build several cryptographic protocols such as blind signatures [FHS15, FHKS16], ring and group signatures [BHKS18, DS18, BHSB19], sanitizable signatures [BLL$^+$19], access control encryption [FGKO17], and point collection systems [BEK$^+$20]. In particular, they have been used to build anonymous credentials [HS14, DHS15a, FHS19] and delegatable anonymous credentials. In the latter case, under the name of mercurial signatures [CL19, CL21], which are an extension of equivalence classes to the key space.

State-of-the-art constructions focused on building schemes under weaker assumptions and with tight security. The first attempts were made from Fuchsbauer *et al.* [FHKS16], and Fuchsbauer and Gay [FG18]. Subsequently, Khalili *et al.* [KSD19] proposed a new SPS-EQ, obtaining the first construction under standard assumptions and with a tight security reduction.

4.2 Contributions

First, we build a new SPS-EQ scheme whose security is proven under standard assumptions. Our departure point is a recent scheme by Khalili *et al.* [KSD19], which we modify to obtain a more efficient construction. The signature from [KSD19] relies on the SXDH assumption (Section 2.3.3 on page 11) and requires a CRS. Our construction also relies on a generalisation of the KerMDH, which is regarded as a standard assumption as well. As presented in the next section, our signature size is roughly 24% shorter while the Sign and ChgRep algorithms require 65% and 45% less exponentiations when compared to the construction from [KSD19], respectively. These improvements are of utmost importance from the practical point of view. For example, in the context of anonyous credentials, the algorithm ChgRep is used every time users present their credential.

Our other contribution pertrains mercurial signatures. Previous constructions

Scheme	$\lvert\sigma\rvert$	$\lvert\mathsf{pk}\rvert$	Sign	Verify	ChgRep	Assumption
[GHKP18]	$8\lvert\mathbb{G}_1\rvert + 6\lvert\mathbb{G}_2\rvert$	$2\lvert\mathbb{G}_1\rvert + (9+\ell)\lvert\mathbb{G}_2\rvert$	28E	**9P**	N/A	SXDH
[KSD19]	$8\lvert\mathbb{G}_1\rvert + 9\lvert\mathbb{G}_2\rvert$	$(\mathbf{2}+\ell)\lvert\mathbb{G}_2\rvert$	29E	11P	19P+38E	SXDH
Section 4.5	$\mathbf{9}\lvert\mathbb{G}_1\rvert + \mathbf{4}\lvert\mathbb{G}_2\rvert$	$(\mathbf{2}+\ell)\lvert\mathbb{G}_2\rvert$	**10E**	11P	**19P+21E**	extKerMDH, SXDH

Table 4.1: Signatures comparison including pairings (P) and exponentiations (E).

([CL19] and [CL21]) where proven secure in the GGM. In Section 4.5.3, we extend our previous construction to obtain the first mercurial signature scheme whose security is proven in the standard model, also assuming a CRS. Our extension only requires two extra multiplications in $\mathbb{Z}_p$, making it practically efficient as well.

4.3 Related Work

The construction of [FG18] is based on the family of matrix Diffie-Hellman assumptions [EHK+13]. They first modify an affine message authentication code (MAC) from [BKP14] to obtain a linear structure-preserving MAC, which is made publicly verifiable using a known technique in the context of SPS [KPW15]. This allows using a *tag* to randomise both the signature and message. Hence, the signing algorithm produces a signature along with a tag that can be used to randomise it. The resulting scheme is secure under a weaker notion of unforgeability called *existential unforgeability under chosen open message attack* (EUF-CoMA).

In [KSD19], the authors observe that using a structure-preserving MAC such as the one from [FG18] has an inherent problem in the security game. As messages and Matrix Decision Diffie-Hellman challenges belong to the same source group of the bilinear group, one cannot do better than EUF-CoMA security following this approach. Consequently, they proposed using an OR-Proof based on that in [GHKP18] to construct tightly secure structure-preserving MACs based on the key encapsulation mechanism of Gay *et al.* [GHK17]. This allowed the authors to circumvent the previous issue and obtain the first EUF-CMA secure SPS-EQ scheme with a tight security reduction under standard assumptions. In Table 4.1, we compare the recent signature constructions from [GHKP18] and [KSD19] with our construction from Section 4.5 to illustrate the efficiency gains.

4.4 Formal Definitions

In this section, we recall the syntax and security properties for SPS-EQ following the formalizations from [FHS19] and [KSD19]. First, we consider equivalence classes on the *message space*. Subsequently, we introduce definitions for SPS-EQ that act on both the message and key space.

Definition 26: Structure-Preserving Signature on Equivalence Classes

A Structure-Preserving Signature on Equivalence Classes (SPS-EQ) consists of the following algorithms:

$\mathsf{PGen}(1^\lambda)$ is a p.p.t algorithm that, given a security parameter λ, outputs public parameters pp.

$\mathsf{PTGen}(1^\lambda)$ is like PGen but it also returns a trapdoor τ (if any).

$\mathsf{KGen}(\mathsf{pp}, \ell)$ is a p.p.t algorithm that, given pp and a vector length $\ell > 1$, outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \boldsymbol{m})$ is a p.p.t algorithm that, given pp, a representative $\boldsymbol{m} \in (\mathbb{G}_i^*)^\ell$ for class $[\boldsymbol{m}]_\mathcal{R}$, a secret key sk, outputs a signature $\sigma' = (\sigma, \tau)$ (potentially including a tag τ) on the message $\boldsymbol{m}$.

$\mathsf{ChgRep}(\mathsf{pp}, \boldsymbol{m}, (\sigma, \tau), \mu, \mathsf{pk})$ is a p.p.t algorithm that takes as input pp, a representative message $\boldsymbol{m} \in (\mathbb{G}_i^*)^\ell$, a signature σ (and potentially a tag τ), a scalar μ, and a public key pk. It computes an updated signature σ' on new representative $\boldsymbol{m}^* = \mu\boldsymbol{m}$ and outputs $(\boldsymbol{m}^*, \sigma')$.

$\mathsf{Verify}(\mathsf{pp}, \boldsymbol{m}, (\sigma, \tau), \mathsf{pk})$ is a deterministic algorithm that takes as input pp, a representative message $\boldsymbol{m}$, a signature σ (potentially including a tag τ), and public key pk. If σ is a valid signature on $\boldsymbol{m}$ it outputs 1 and 0 otherwise.

Correctness of SPS-EQ schemes requires that for every message $\boldsymbol{m}$, security parameter λ, vector length ℓ, and $\mu \in \mathbb{Z}_p^*$:

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{PGen}(1^\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell) \end{array} : \begin{array}{l} \mathsf{Verify}(\mathsf{pp}, \boldsymbol{m}, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \boldsymbol{m}), \mathsf{pk}) \wedge \\ \mathsf{Verify}(\mathsf{pp}, \mathsf{ChgRep}(\mathsf{pp}, \boldsymbol{m}, \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \boldsymbol{m}), \\ \mu, \mathsf{pk}), \mathsf{pk}) \end{array}\right] = 1$$

Two security properties are required for SPS-EQ schemes: unforgeability and perfect adaption. The former is based on the usual EUF-CMA notion (Definition 12 on page 20). The latter requires ChgRep to output signatures identically distributed to new signatures on the respective representative. As previously mentioned, indistinguishability of the message space for equivalence classes and perfect adaption provides unlinkability between message-signature pairs in the same class. Definition 27 (given below) explicitly allows forgeries on the same equivalence class, which is the only allowed form of forgery (*i.e.,* controlled malleability).

Definition 27: EUF-CMA

An SPS-EQ scheme over $(\mathbb{G}_i^*)^\ell$ is existentially unforgeable under adaptively chosen-message attacks, if for all $\ell > 1$ and p.p.t adversaries $\mathcal{A}$ with access to a signing oracle Sign, the following probability is negligible,

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{PGen}(1^\lambda), \\ (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}, \ell), \\ ([\boldsymbol{m}^*]_i, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, \cdot)}(\mathsf{pk}) \end{array} : \begin{array}{c} [\boldsymbol{m}^*]_\mathcal{R} \neq [\boldsymbol{m}]_\mathcal{R} \ \forall\ [\boldsymbol{m}]_i \in Q \ \wedge \\ \mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}^*]_i, \sigma^*, \mathsf{pk}) = 1 \end{array}\right],$$

where Q is the set of queries that $\mathcal{A}$ has issued to the signing oracle Sign. Note that in the tag-based case this oracle returns a pair (σ, τ).

For perfect adaption we present a first definition in the CRS model (*i.e.,* honest parameters model) based on Definition 10 from [KSD19]. When this notion is defined considering adversaries who could maliciously generate keys, one obtains the strongest possible notion for perfect adaption. Unlike [KSD19], we opt to explicitly state that perfect adaption is defined with respect to the *message space*.

Definition 28: Perfect adaption of signatures (1)

An SPS-EQ over $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures (under malicious keys in the honest parameters model) with respect to the message space if for all tuples $(\mathsf{pp}, \mathsf{pk}, [\boldsymbol{m}]_i, \sigma, \mu)$ where $\mathsf{pp} \overset{\$}{\leftarrow} \mathsf{PGen}(1^\lambda)$, $[\boldsymbol{m}]_i \in (\mathbb{G}_i^*)^\ell$, $\mu \in \mathbb{Z}_p^*$, and $\mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}]_i, \sigma, \mathsf{pk}) = 1$, we have that the output of $\mathsf{ChgRep}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \mathsf{pk})$ is $([\mu\boldsymbol{m}]_i, \sigma^*)$, with σ^* being a uniformly random element in the space of signatures, conditioned on $\mathsf{Verify}(\mathsf{pp}, [\mu\boldsymbol{m}]_i, \sigma^*, \mathsf{pk}) = 1$.

We now introduce the notion of mercurial signatures from [CL19] as an extension to SPS-EQ. We include the algorithms ConvertPK, ConvertSK and ConvertSig as defined for mercurial signatures. ConvertSig is analogous to the ChgRep algorithm but restricted to acting on the equivalence class defined by the key space. ConvertPK and ConvertSK abstract the computation of new representative keys.

Definition 29: Mercurial signature

A Mercurial signature is a SPS-EQ that includes the following algorithms:

$\mathsf{ConvertPK}(\mathsf{pp}, \mathsf{pk}, \rho)$ is a p.p.t algorithm that, given pp, a key converter ρ, and pk, outputs a new representative public key $\mathsf{pk}' \in [\mathsf{pk}]_\mathcal{R}$.

$\mathsf{ConvertSK}(\mathsf{pp}, \mathsf{sk}, \rho)$ is a p.p.t algorithm that, given pp, a key converter ρ, and sk, outputs a new representative secret key $\mathsf{sk}' \in [\mathsf{sk}]_\mathcal{R}$.

$\mathsf{ConvertSig}(\mathsf{pp}, \boldsymbol{m}, (\sigma, \tau), \rho, \mathsf{pk})$ is a p.p.t algorithm that takes as input pp, $\boldsymbol{m}$, σ (potentially including a tag τ), a key converter ρ, and pk. It computes an updated signature σ' on $\boldsymbol{m}$ under a new representative public key $\mathsf{pk}' \in [\mathsf{pk}]_\mathcal{R}$ and outputs $(\boldsymbol{m}, \sigma')$.

A second definition for perfect adaption (introduced as *origin-hiding* in [CL19]) considers it with respect to the *key space*. For this reason, the algorithm ConvertSig is used instead of ChgRep.

Definition 30: Perfect adaption of signatures (2)

An SPS-EQ over $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures (under malicious keys in the honest parameters model) with respect to the key space $\mathcal{S}_{\mathsf{pk}}$ if for all tuples $(\mathsf{pp},[\mathsf{pk}]_j,[\boldsymbol{m}]_i,(\sigma,\tau),\rho)$ where $\mathsf{pp} \overset{\$}{\leftarrow} \mathsf{PGen}(1^\lambda)$, $[\mathsf{pk}]_j \in \mathcal{S}_{\mathsf{pk}}$, $[\boldsymbol{m}]_i \in (\mathbb{G}_i^*)^\ell$, $\mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), [\mathsf{pk}]_j) = 1$ and $\rho \in \mathbb{Z}_p^*$, we have that the output of $\mathsf{ConvertSig}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), \rho, [\mathsf{pk}]_j)$ is σ^*, with σ^* being a random element in the space of signatures, conditioned on $\mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}]_i, \sigma^*, \mathsf{ConvertPK}(\mathsf{pp}, [\mathsf{pk}]_j, \rho)) = 1$.

A third definition considers the joint executions of the algorithms ChgRep and ConvertSig. Therefore, we define a more general notion for perfect adaption where ChgRep takes μ and ρ as input, acting on both equivalence classes.

Definition 31: Perfect adaption of signatures (3)

An SPS-EQ over $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures (under malicious keys in the honest parameters model) if for all tuples $(\mathsf{pp}, [\mathsf{pk}]_j, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho)$ where $\mathsf{pp} \xleftarrow{\$} \mathsf{PGen}(1^\lambda)$, $[\mathsf{pk}]_j \in \mathcal{S}_{\mathsf{pk}}$, $[\boldsymbol{m}]_i \in (\mathbb{G}_i^*)^\ell$, $\mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), [\mathsf{pk}]_j) = 1$ and $\mu, \rho \in \mathbb{Z}_p^*$, we have that the output of $\mathsf{ChgRep}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{pk}]_j)$ is $([\mu\boldsymbol{m}]_i, \sigma^*)$, with σ^* being a random element in the space of signatures, conditioned on $\mathsf{Verify}(\mathsf{pp}, [\mu\boldsymbol{m}]_i, \sigma^*, \mathsf{ConvertPK}(\mathsf{pp}, [\mathsf{pk}]_j, \rho)){=}1$.

The mercurial signature construction presented in section 4.5.3 satisfies a weaker form of perfect adaption with respect to the key space. Instead of having perfect adaption under *malicious keys* in the honest parameters model, it has perfect adaption under *honestly generated keys* in the honest parameters model. For completeness, we provide below a fourth definition of perfect adaption, which only considers honestly generated keys.

Definition 32: Perfect adaption of signatures (4)

An SPS-EQ over $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures (under honest keys in the honest parameters model) if for all tuples $(\mathsf{pp}, [\mathsf{pk}]_j, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho)$ where $\mathsf{pp} \xleftarrow{\$} \mathsf{PGen}(1^\lambda)$, $[\mathsf{pk}]_j \in \mathsf{KGen}(\mathsf{pp}, \ell)$, $[\boldsymbol{m}]_i \in (\mathbb{G}_i^*)^\ell$, $\mathsf{Verify}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), [\mathsf{pk}]_j){=}1$ and $\mu, \rho \in \mathbb{Z}_p^*$, we have that the output of $\mathsf{ChgRep}(\mathsf{pp}, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{pk}]_j)$ is $([\mu\boldsymbol{m}]_i, \sigma^*)$, with σ^* being a random element in the space of signatures, conditioned on $\mathsf{Verify}(\mathsf{pp}, [\mu\boldsymbol{m}]_i, \sigma^*, \mathsf{ConvertPK}(\mathsf{pp}, [\mathsf{pk}]_j, \rho)){=}1$.

4.5 SPS-EQ from Standard Assumptions

We present an SPS-EQ scheme where the OR-based proof in [KSD19] is replaced by the one in [CH20] while adapting the related building blocks.

The starting point for the SPS-EQ construction in [KSD19] was the tightly secure SPS from [GHKP18], which builds on a structure-preserving MAC (based on the works from [GHK17] and [Hof17]) and a NIZK OR-Proof from [Ràf15]. To couple with equivalence classes, the authors proposed a way to adapt the OR-Proof so that it could be randomised and malleable. Unfortunately, as the CRS used in the OR-Proof from [Ràf15] was incompatible with the required randomisation properties, the authors were forced to build a quasi-adaptive NIZK (QA-NIZK) on top to overcome the limitation. In a QA-NIZK, the CRS is allowed to depend on the language. Contrary to that approach, in a fully adaptive NIZK the CRS is independent of the language (*i.e.,* the same proof system can be used for different languages). In the following, we explain how to circumvent the previous issue, obtaining a more efficient construction based on a fully adaptive NIZK.

```
$\underline{\mathsf{PGen}(1^\lambda):}$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $z \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs} \leftarrow (\mathsf{BG}, [z]_2)$; return $\mathsf{crs}$

$\underline{\mathsf{PTGen}(1^\lambda):}$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $z \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs} \leftarrow (\mathsf{BG}, [z]_2)$; return $(\mathsf{crs}, z)$

$\underline{\mathsf{PPro}(\mathsf{crs}, [\mathbf{x}_1]_1, \mathbf{w}_1, [\mathbf{x}_2]_1, \mathbf{w}_2):}$
// $[\mathbf{x}_j]_1 = \mathbf{A}_i\mathbf{w}_j$ with $\mathbf{A} \in \mathcal{M}^{2k\times k}$
$\mathbf{s}_j \xleftarrow{\$} \mathbb{Z}_p^k$; $z_{1-i} \xleftarrow{\$} \mathbb{Z}_p^*$; $\delta \xleftarrow{\$} \mathbb{Z}_p^*$
$[z_i]_2 \leftarrow \delta[z]_2 - [z_{1-i}]_2$
$[\mathbf{d}_i^j]_2 \leftarrow [z_i]_2\mathbf{w}_j + [\mathbf{s}_j]_2$
$[\mathbf{a}_i^j]_1 \leftarrow [\mathbf{A}_i]\mathbf{s}_j$
$\mathbf{d}_{1-i}^j \xleftarrow{\$} \mathbb{Z}_p^k$
$[\mathbf{a}_{1-i}^j]_1 \leftarrow \mathbf{A}_{1-i}\mathbf{d}_{1-i}^j - z_{1-i}\mathbf{x}_j$
return $([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, \delta P_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$

$\underline{\mathsf{PSim}(\mathsf{crs}, z, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1):}$
$z_0 \xleftarrow{\$} \mathbb{Z}_p$; $\delta \xleftarrow{\$} \mathbb{Z}_p^*$; $z_1 \leftarrow \delta z - z_0$
for all $i \in \{0,1\}$, $j \in \{1,2\}$ do
  $\mathbf{d}_i^j \xleftarrow{\$} \mathbb{Z}_p^k$
  $[\mathbf{a}_i^j]_1 \leftarrow \mathbf{A}_i\mathbf{d}_i^j - z_i\mathbf{x}_j$
return $([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, \delta P_1)_{i\in\{0,1\}}^{j\in\{1,2\}}$
```

```
$\underline{\mathsf{PRVer}(\mathsf{crs}, [\mathbf{x}]_1, \pi):}$
$([\mathbf{a}_i]_1, [\mathbf{d}_i]_2, [z_i]_2, Z_1)_{i\in\{0,1\}} \leftarrow \pi$
check $e(Z_1, [z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$
for all $i \in \{0,1\}$ check
  $e([\mathbf{A}_i]_1, [\mathbf{d}_i]_2) = e([\mathbf{x}]_1, [z_i]_2) + e([\mathbf{a}_i]_1, [1]_2)$
return 1

$\underline{\mathsf{PVer}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega):}$
$([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, Z_1)_{i\in\{0,1\}}^{j\in\{1,2\}} \leftarrow \Omega$
check $e(Z_1, [z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$
for all $i \in \{0,1\}$, $j \in \{1,2\}$ check
  $e([\mathbf{A}_i]_1, [\mathbf{d}_i^j]_2) = e([\mathbf{x}_j]_1, [z_i]_2) + e([\mathbf{a}_i^j]_1, [1]_2)$
return 1

$\underline{\mathsf{ZKEval}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega):}$
$([\mathbf{a}_i^j]_1, [\mathbf{d}_i^j]_2, [z_i]_2, Z_1)_{i\in\{0,1\}}^{j\in\{1,2\}} \leftarrow \Omega$
check $\mathsf{PVer}(\mathsf{crs}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1, \Omega)$
$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$; $Z_1' \leftarrow \alpha Z_1$
for all $i \in \{0,1\}$
  $[z_i']_2 \leftarrow \alpha[z_i]_2$
  $[\mathbf{a}_i']_1 \leftarrow \alpha[\mathbf{a}_i^1]_1 + \alpha\beta[\mathbf{a}_i^2]_1$
  $[\mathbf{d}_i']_2 \leftarrow \alpha[\mathbf{d}_i^1]_2 + \alpha\beta[\mathbf{d}_i^2]_2$
return $([\mathbf{a}_i']_1, [\mathbf{d}_i']_2, [z_i']_2, Z_1')$
```

Figure 4.1: Malleable NIZK argument for language $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$

4.5.1 Malleable NIZK Argument

Unlike the one from [KSD19], which is a one-time homomorphic QA-NIZK based on the OR-Proof from [Ràf15] and the QA-NIZK from [KW15], our malleable NIZK argument is based solely on the fully adaptive OR-Proof from [CH20]. This allows us to circumvent the randomisation problem in the OR-Proof from [Ràf15], avoiding the need to build a QA-NIZK atop.

As a result, we reduce the number of exponentiations required in the proving and ZKEval algorithms, which leads to a more efficient signature scheme. This comes at the cost of relying on the $\mathcal{L}_1$-1-extKerMDH assumption (Section 2.3.3 on page 11). We argue that the change is justified as the extKerMDH is a natural extension of the KerMDH assumption, and in this case, the assumption is also falsifiable.

Intuition. We look for a NIZK proof which can be randomisable and malleable so that randomised proofs look like fresh proofs while the malleability allows updating the proof statements. The goal is to obtain derivation privacy, which is crucial to perform the change of representative in the signature scheme.

The fully adaptive NIZK argument from [CH20] is based on a challenge $z = z_0 + z_1$, where z is in the CRS, and z_0 and z_1 are elements of the proof and chosen such that the equation holds. To randomise a proof, we need to randomise z_0 and z_1. Therefore, instead of checking the original equation with z, we will check for linear combinations of the equation $z = z_0 + z_1$. To do so, we modify the original proof to compute a random α such that $\alpha z - z_0 = z_1$ (for a fresh z_0), adding an extra element $Z_1 = \alpha P_1$ to the proof. Consequently, the verification algorithm will now check an extra pairing.

As observed in [KSD19], the malleability of the OR-NIZK proof can be achieved using a tag and a second NIZK for that tag with shared randomness. We follow the same approach, also providing a second verification algorithm (PRVer) to verify a single OR-proof, as used in the SPS-EQ construction. The resulting malleable NIZK argument for the OR-language defined below (for fixed $\mathbf{A}_0$ and $\mathbf{A}_1$) is given in Figure 4.1. As in [KSD19], PPro outputs two proofs with shared randomness.

$$\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1} = \{[\mathbf{x}]_1 \in \mathbb{G}_1^{2k} | \exists\ \mathbf{w} \in \mathbb{Z}_p^k : [\mathbf{x}]_1 = [\mathbf{A}_0]_1\mathbf{w} \vee [\mathbf{x}]_1 = [\mathbf{A}_1]_1\mathbf{w}\}$$

Theorem 15. The construction from Figure 4.1 is a fully adaptive NIZK argument for $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ if the falsifiable $\mathcal{L}_1$-1-$\mathsf{extKerMDH}$ assumption holds in $\mathbb{G}_2$.

Proof. We must show *completeness, perfect zero-knowledge, computational soundness* and *derivation privacy.* We do it in the same way as done for the original protocols from [CH20] (Theorem 19) and [KSD19] (Theorem 1).
Perfect Completeness. Let $[\mathbf{x}]_1 = [\mathbf{A}_0]_1\mathbf{w}$ be a valid statement for $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ with witness $\mathbf{w}_1$ (if the statement holds with respect to $\mathbf{A}_1$, the proof is analogous), and let π=(([$\mathbf{a}_i]_1$, $[\mathbf{d}_i]_2$, $[z_i]_2)_{i\in\{0,1\}}$, Z_1) be a valid proof for $[\mathbf{x}]_1 \in \mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$. In the following, we show that $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{x}]_1, \pi) = 1$. First, we have:

$$\begin{aligned}
& && e([\mathbf{A}_0]_1, [\mathbf{d}_0]_2) = e([\mathbf{x}]_1, [z_0]_2) + e([\mathbf{a}_0]_1, [1]_2) \\
& \iff && \\
& && [\mathbf{A}_0\mathbf{d}_0]_T = [\mathbf{x}z_0 + \mathbf{a}_0]_T \\
([\mathbf{a}_0]_1 = [\mathbf{A}_0]_1\mathbf{s}_1) & \iff && \\
& && [\mathbf{A}_0\mathbf{d}_0]_T = [\mathbf{x}z_0 + \mathbf{A}_0\mathbf{s}_1]_T \\
([\mathbf{d}_0]_1 = z_0\mathbf{w} + \mathbf{s}_1) & \iff && \\
& && [\mathbf{A}_0(z_0\mathbf{w} + \mathbf{s}_1)]_T = [\mathbf{x}z_0 + \mathbf{A}_0\mathbf{s}_1]_T \\
(\mathbf{x} = \mathbf{A}_0\mathbf{w}) & \iff && \\
& && [\mathbf{A}_0(z_0\mathbf{w} + \mathbf{s}_1)]_T = [\mathbf{A}_0(z_0\mathbf{w} + \mathbf{s}_1)]_T
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
& && e([\mathbf{A}_1]_1, [\mathbf{d}_1]_2) = e([\mathbf{x}]_1, [z_1]_2) + e([\mathbf{a}_1]_1, [1]_2) \\
& \iff && \\
& && [\mathbf{A}_1\mathbf{d}_1]_T = [\mathbf{x}z_1 + \mathbf{a}_1]_T \\
([\mathbf{a}_1]_1 = \mathbf{A}_1\mathbf{d}_1 - \mathbf{x}z_1) & \iff &&
\end{aligned}$$

$$[\mathbf{A}_1\mathbf{d}_1]_2 = [\mathbf{x}z_1 + \mathbf{A}_1\mathbf{d}_1 - \mathbf{x}z_1]_T$$

Finally, we have that $e(Z_1, [z]_2) = [\alpha z]_T = e([1]_1, [\alpha z]_2) = e([1]_1, [z_0]_2 + [z_1]_2)$.

Perfect Zero-Knowledge. We have to show that the distributions PSim and PPro are identical. As in [CH20] (Theorem 19), the simulator can generate an identically distributed proof when given the trapdoor while hiding the value α used to randomise the challenges with Z_1 as done by the real prover.

Computational Soundness. Again, the only difference from [CH20] (Theorem 19) is that we now use a linear relation to check the challenge, which is equivalent to verifying the original equation $z = z_0 + z_1$. Therefore, soundness can be proven following the proof from [CH20] almost in verbatim.

Derivation Privacy. Let $[\mathbf{x}_1]_1 = [\mathbf{A}_0]_1\mathbf{w}_1$ and $[\mathbf{x}_2]_1 = [\mathbf{A}_0]_1\mathbf{w}_2$ be valid statements for $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ (the proof is analogous if the statements hold with respect to $\mathbf{A}_1$ instead). Let $\pi^1 = (([\mathbf{a}_i]^1_1, [\mathbf{d}_i]^1_2, [z_i]_2)_{i\in\{0,1\}}, Z_1)$ and $\pi^2 = (([\mathbf{a}_i]^2_1, [\mathbf{d}_i]^2_2, [z_i]_2)_{i\in\{0,1\}}, Z_1)$ be valid proofs for $[\mathbf{x}_1]_1, [\mathbf{x}_2]_1 \in \mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$. Let $\widehat{\pi} = (([\widehat{\mathbf{a}}_i]_1, [\widehat{\mathbf{d}}_i]_2, [\widehat{z}_i]_2)_{i\in\{0,1\}}, \widehat{Z}_1) = (\alpha[\mathbf{a}_0]^1_1 + \alpha\beta[\mathbf{a}_0]^2_1, \alpha[\mathbf{a}_1]^1_1 + \alpha\beta[\mathbf{a}_1]^2_1, \alpha[\mathbf{d}_0]^1_2 + \alpha\beta[\mathbf{d}_0]^2_2, \alpha[\mathbf{d}_1]^1_2 + \alpha\beta[\mathbf{d}_1]^2_2, \alpha[z_0]_2, \alpha[z_1]_2, \alpha Z_1)$ be the output from $\mathsf{ZKEval}(\mathsf{crs}, [\mathbf{x}_1]_1, \mathbf{w}_1, [\mathbf{x}_2]_1, \mathbf{w}_2)$, where the corresponding witness is $\widehat{\mathbf{w}} = \mathbf{w}_1 + \beta\mathbf{w}_2$. ZKEval outputs a proof with new independent randomness α and β, which has identical distribution with respect to PProv when computing a single proof. Thus, we achieve perfect derivation privacy. $\square$

4.5.2 Construction

Our construction is shown in Figure 4.2. As previously mentioned, it relies on a tag to randomise and adapt the signature. More in detail, as explained in [KSD19], the departure point for the signature construction is an observation (core lemma from [GHKP18]) that for all $[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r$, with $r \xleftarrow{\$} \mathbb{Z}_p$, fixed matrices $[\mathbf{A}_0]_1, [\mathbf{A}_1]_1 \xleftarrow{\$} \mathcal{D}_1$, and a NIZK proof for $[\mathbf{t}]_1 \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$, the values $\mathbf{K}_0^\top[\mathbf{t}]_1$ and $(\mathbf{K}^\top + \mathbf{v}^\top)[\mathbf{t}]_1$ are indistinguishable under the MDDH assumption (Section 2.3.3) when $\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^2$ is a key and $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^2$. Furthermore, for all $[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1$ and $[\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$ with $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$, and a NIZK proof for $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \mathrm{span}(\mathbf{A}_0) \cup \mathrm{span}(\mathbf{A}_1)$, the tuples $(\mathbf{K}_0^\top[\mathbf{t}]_1, \mathbf{K}_0^\top[\mathbf{w}]_1)$ and $((\mathbf{K}_0^\top + \mathbf{v}^\top)[\mathbf{t}]_1, \mathbf{K}_0^\top[\mathbf{w}]_1)$ are also indistinguishable.

With the above in mind, a signature σ on a message $[\boldsymbol{m}]_1 \in (\mathbb{G}_1^*)^\ell$ will have the form $\sigma = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\boldsymbol{m}]_1$, with a tag $\tau = \mathbf{K}_0^\top[\mathbf{w}]_1$. The secret key sk will be $(\mathbf{K}_0, \mathbf{K})$ for $\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^2$ and $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^\ell$. Using the malleable NIZK argument from Section 4.5.1, two OR-Proofs sharing the same randomness are computed for $[\mathbf{t}]_1$ and $[\mathbf{w}]_1$, allowing anyone who knows the tag to randomise and adapt the signature to a new representative.

The security of our construction relies on a (new) core lemma. Next, we prove the lemma, unforgeability and perfect adaption with respect to the message space.

```
$\underline{\mathsf{SPS\text{-}EQ.PGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$
$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$\mathsf{crs} \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$
$\mathbf{return}\ \mathsf{pp}$

$\underline{\mathsf{SPS\text{-}EQ.PTGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$
$\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \tau) \xleftarrow{\$} \mathsf{PTGen}(1^\lambda; \mathsf{BG})$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$
$\mathbf{return}\ (\mathsf{pp}, \tau)$

$\underline{\mathsf{SPS\text{-}EQ.KGen}(\mathsf{pp}, \ell)}$:
$\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}; \mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times 2}$
$[\mathbf{B}]_2 \leftarrow [\mathbf{K}_0]_2[\mathbf{A}]_2$
$[\mathbf{C}]_2 \leftarrow [\mathbf{K}]_2[\mathbf{A}]_2$
$\mathsf{sk} \leftarrow (\mathbf{K}_0, \mathbf{K});\ \mathsf{pk} \leftarrow ([\mathbf{B}]_2, [\mathbf{C}]_2)$
$\mathbf{return}\ (\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{SPS\text{-}EQ.Sign}(\mathsf{pp}, \mathsf{sk}, [\boldsymbol{m}]_1)}$:
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1;\ [\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
$\Omega \leftarrow \mathsf{PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
$(\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1) \leftarrow \Omega$
$\mathbf{u}_1 \leftarrow \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top [\boldsymbol{m}]_1$
$\mathbf{u}_2 \leftarrow \mathbf{K}_0^\top [\mathbf{w}]_1$
$\sigma \leftarrow ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
$\tau \leftarrow ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$
$\mathbf{return}\ (\sigma, \tau)$

$\underline{\mathsf{SPS\text{-}EQ.Verify}(\mathsf{pp}, [\boldsymbol{m}]_1, (\sigma, \tau), \mathsf{pk})}$:
$([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1) \leftarrow \sigma$
$\mathbf{check}$
    $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
    $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) = e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\boldsymbol{m}]_1^\top, [\mathbf{C}]_2)$
$\mathbf{if}\ \tau \neq \perp\ \mathbf{check}$
    $([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \leftarrow \tau$
    $\mathsf{PRVer}(\mathsf{crs}, [\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
    $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) = e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
$\mathbf{return}\ 1$

$\underline{\mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{pp}, [\boldsymbol{m}]_1, \sigma, \tau, \mu, \mathsf{pk})}$:
$([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1) \leftarrow \sigma$
$([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \leftarrow \tau$
$\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
$\mathbf{check}$
    $\mathsf{PVer}(\mathsf{crs}, [\mathbf{t}]_1, [\mathbf{w}]_1, \Omega)$
    $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
    $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\boldsymbol{m}]_1^\top, [\mathbf{C}]_2)$
$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$
$[\mathbf{u}_1']_1 \leftarrow \mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1$
$[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$
$\mathbf{for\ all}\ i \in \{0, 1\}$
    $[z_i']_2 \leftarrow \alpha[z_i]_2$
    $[\mathbf{a}_i']_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$
    $[d_i']_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$
$\Omega' \leftarrow (([\mathbf{a}_i']_1, [d_i']_2, [z_i']_2)_{i\in\{0,1\}}, \alpha Z_1)$
$\sigma' \leftarrow ([\mathbf{u}_1']_1, [\mathbf{t}']_1, \Omega')$
$\mathbf{return}\ (\mu[\boldsymbol{m}]_1, \sigma')$
```

Figure 4.2: Our SPS-EQ scheme.

Lemma 16 (Core Lemma)**.** If the $\mathcal{D}_1$-MDDH (DDH) assumption holds in $\mathbb{G}_1$ and the tuple of algorithms $\Pi = (\mathsf{PGen}, \mathsf{PPro}, \mathsf{PSim}, \mathsf{PRVer})$ is a non-interactive zero-knowledge argument for $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$, then going from experiment $\mathsf{Exp}^{\mathsf{core}}_0$ to $\mathsf{Exp}^{\mathsf{core}}_1$ (Figure 4.3) can (up to negligible terms) only increase the winning chance of an adversary. More precisely, for every adversary $\mathcal{A}$, there exist adversaries $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{B}_2$ s.t

$$\mathbf{Adv}^{\mathsf{core}}_0(\mathcal{A}) - \mathbf{Adv}^{\mathsf{core}}_1(\mathcal{A}) \leq \Delta^{\mathsf{core}}_{\mathcal{A}}, \text{ where}$$

$$\Delta^{\mathsf{core}}_{\mathcal{A}} = (2 + 2\lceil \log Q \rceil \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}) + (8\lceil \log Q \rceil + 4)\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_1)$$
$$2\lceil \log Q \rceil \mathbf{Adv}^{\mathsf{snd}}_{\Pi}(\mathcal{B}_2) + \lceil \log Q \rceil \Delta_{\mathcal{D}_1} + \frac{(8\lceil \log Q \rceil + 4)}{p-1} + \frac{(\lceil \log Q \rceil)Q}{p}$$

and the term $\Delta_{\mathcal{D}_1}$ is statistically small.

```
$\underline{\mathsf{Exp}^{\mathsf{core}}_{\beta}(\lambda), \beta \in \{0,1\}:}$
$\mathsf{ctr} \leftarrow 0$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$
$\mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
$(\mathsf{crs}, \tau) \xleftarrow{\$} \mathsf{PGen}(1^\lambda; \mathsf{BG})$
$\mathsf{pp} \leftarrow (\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, \mathsf{crs})$
$\mathbf{k}_0, \mathbf{k}_1 \xleftarrow{\$} \mathbb{Z}_p^2$
$\mathsf{tag} \leftarrow \mathcal{A}^{\mathsf{TAGO}()}(\mathsf{pp})$
$\mathbf{return}\ \mathsf{VERO}(\mathsf{tag})$

$\underline{\mathsf{TAGO}():}$
$\mathsf{ctr} \leftarrow \mathsf{ctr} + 1;\ r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1,\ [\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
$\Omega \leftarrow \mathsf{PPro}(\mathsf{crs}, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
$[\mathbf{u}']_1 \leftarrow (\mathbf{k}_0 + \beta \cdot \mathbb{F}(\mathsf{ctr}))^\top [\mathbf{t}]_1$
$[\mathbf{u}'']_1 \leftarrow (\mathbf{k}_0 + \beta \cdot \mathbf{k}_1)^\top [\mathbf{w}]_1$
$\mathsf{tag} \leftarrow ([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$
$\mathbf{return}\ \mathsf{tag}$

$\underline{\mathsf{VERO}(\mathsf{tag}):}$
$([\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1, [\mathbf{u}']_1) \leftarrow \mathsf{tag}$
$\mathbf{if}\ \mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, Z_1)) \wedge$
$\exists\ \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}']_1 = (\mathbf{k}_0 + \beta \cdot \mathbb{F}(\mathsf{ctr}'))^\top [\mathbf{t}]_1$
$\mathbf{return}\ 1\ \mathbf{else\ return}\ 0$
```

Figure 4.3: Core lemma for our $\mathsf{SPS\text{-}EQ}$ scheme.

Proof. The proof of this lemma is very similar (in parts verbatim) to the one given in [KSD19], which in turn extends the original core lemma from [GHKP18]. The main difference is that we use the standard definition for zero-knowledge in our NIZK argument system instead of the composable one. However, as pointed out in [AJO+19], the standard notion of zero-knowledge suffices in this context. For completeness, we present the full proof below.

Game 0: We define **Game 0** = $\mathsf{Exp}^{\mathsf{core}}_0$ and thus by definition:

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathsf{core}}_0(\mathcal{A})$$

Game 1: In this game, we replace PPro with PSim in **Game 0** to compute the proof. An adversary $\mathcal{B}$ for **Game 1** is such that $T(\mathcal{B}) \approx T(\mathbf{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$\mathbf{Adv}_0 - \mathbf{Adv}_1 \leq \mathbf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{B}), \text{ where}$$

$\mathbf{Adv}_{\Pi}^{\mathsf{zk}}(\mathcal{B})$ is the advantage of $\mathcal{B}$ to break the zero-knowledge property from Π.
Game 2: In this game we pick $[\mathbf{t}]_1, [\mathbf{w}]_1 \xleftarrow{\$} \mathbb{G}_1^2$ instead of computing them as in the previous game. We can switch $[\mathbf{t}]_1$ and $[\mathbf{w}]_1$ to random over $\mathbb{G}_1^2$ by applying the $\mathcal{D}_1$-MDDH assumption. More precisely, let $\mathcal{A}$ be an adversary distinguishing between **Game 1** and **Game 2** and let $\mathcal{B}_1$ be an adversary given two Q-fold $\mathcal{D}_1$-MDDH challenges $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}_1]_1, ..., [\mathbf{q}_Q]_1)$ and $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}'_1]_1, ..., [\mathbf{q}'_Q]_1)$ as input. Now $\mathcal{B}_1$ sets up the game for $\mathcal{A}$ similar to **Game 1**, but instead choosing $\mathbf{A}_0 \xleftarrow{\$} \mathcal{D}_1$, it uses its challenge matrix $[\mathbf{A}_0]_1$ as part of the public parameters pp. Further, to answer tag queries $\mathcal{B}_1$ sets $[\mathbf{t}_i]_1 \leftarrow [\mathbf{q}_i]_1$, and $[\mathbf{w}_i]_1 \leftarrow [\mathbf{q}_i]_1$ and computes the rest accordingly. This is possible as the proof Ω is simulated from **Game 1** on. In case $\mathcal{B}_1$ was given a real $\mathcal{D}_1$-MDDH challenge, it simulates **Game 1** and otherwise **Game 2**. Thus, by Lemma 3, we have an adversary $\mathcal{B}_1$ with $T(\mathcal{B}_1) \approx T(\mathcal{A}) + Q \cdot \mathsf{poly}(\lambda)$ and

$$\mathbf{Adv}_1 - \mathbf{Adv}_2 \leq 2\mathbf{Adv}_{\mathcal{D}_1,\mathbb{G}_s}^{\mathsf{MDDH}}(\mathcal{B}_1) + \tfrac{2}{p-1}$$

Game 3.0: Let us denote a sequence of games with $3.i$, where $\mathbb{F}_i$ is the random function $\mathbb{F}$ on i-bit prefixes, and the i-bit prefix of ctr is $\mathsf{ctr}_{|i}$. In this game, we compute $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbb{F}_i(\mathsf{ctr}_{|i})[\mathbf{t}]_1$, and $[\mathbf{u}'']_1 = (\mathbf{k}_0 + \mathbf{k}'_0)[\mathbf{w}]_1$ (where $\mathbf{k}'_0 = \mathbb{F}_0(\mathsf{ctr}_{|0})$). In the verification algorithm also, we verify $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbb{F}_i(\mathsf{ctr}|i')[\mathbf{t}]_1$ for $\mathsf{ctr}' \leq \mathsf{ctr}$, and $[\mathbf{u}'']_1 = (\mathbf{k}_0 + \mathbf{k}'_0)[\mathbf{w}]_1$. As for all $\mathsf{ctr} \in \mathbb{N}$ we have $\mathbb{F}_0(\mathsf{ctr}_{|0}) = \mathbb{F}_0(\epsilon)$ and $\mathbf{k}_0$ is identically distributed to $\mathbf{k}_0 + \mathbb{F}_0(\epsilon)$ for $\mathbf{k}_0 \xleftarrow{\$} \mathbb{Z}_p$, we have

$$\mathbf{Adv}_{3.0} = \mathbf{Adv}_2$$

Game 3.i→Game 3.(i+1) We proceed via a series of hybrid games $H_{i.j}$ for $i \in [0,\log(Q) - 1]$ and $j \in [1, 8]$, marking the adversary's advantage on each game with $\mathbf{Adv}'$.
Game 3.i→$H_{i.1}$: In this game, we compute $[\mathbf{t}]_1 = [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_{1.i}$ and $[\mathbf{w}]_1 = [\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_{2.i}$, instead of picking them randomly. Here, ctr_{i+1} is the $i+1$'st bit of the binary representation of ctr. More precisely, we introduce an intermediary game $H_{i.0}$, where we choose $[\mathbf{t}_i]_1$ and $[\mathbf{w}_i]_1$ as

$$[\mathbf{t}_i]_1 = \begin{cases} [\mathbf{A}_{\mathsf{ctr}_{i+1}}] r_{1.i} \text{ for } r_{1.i} \xleftarrow{\$} \mathbb{Z}_p & \text{if } \mathsf{ctr}_{i+1} = 0 \\ [\mathbf{u}_i]_1 \text{ for } \mathbf{u}_i \xleftarrow{\$} \mathbb{Z}_p^2 & \text{otherwise} \end{cases}$$

$$[\mathbf{w}_i]_1 = \begin{cases} [\mathbf{A}_{\mathsf{ctr}_{i+1}}] r_{2.i} \text{ for } r_{2.i} \xleftarrow{\$} \mathbb{Z}_p & \text{if } \mathsf{ctr}_{i+1} = 0 \\ [\mathbf{u}'_i]_1 \text{ for } \mathbf{u}'_i \xleftarrow{\$} \mathbb{Z}_p^2 & \text{otherwise} \end{cases}$$

Let $\mathcal{A}$ be an adversary distinguishing between **Game 3.i** and $H_{i.0}$ and let $\mathcal{B}_1$ be an adversary given two Q-fold $\mathcal{D}_1$-MDDH challenges $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}_1]_1, ..., [\mathbf{q}_Q]_1)$ and $(\mathsf{BG}, [\mathbf{A}_0]_1, [\mathbf{q}'_1]_1, ..., [\mathbf{q}'_Q]_1)$. Then $\mathcal{B}_1$ sets up the game for $\mathcal{A}$ similar to **Game 3.i**, where it embeds $[\mathbf{A}_0]_1$ into the public parameters pp. Further, whenever obtaining a simulation query ctr with $\mathsf{ctr}_{i+1} = 0$, $\mathcal{B}_1$ sets $[\mathbf{t}_i]_1 \leftarrow [\mathbf{q}_i]_1$ and $[\mathbf{w}_i]_1 \leftarrow [\mathbf{q}'_i]_1$ and otherwise follows **Game 3.i**. Similarly, we can reduce the transition from game $H_{i.0}$ to $H_{i.1}$ to the MDDH assumption. We have

$$|\mathbf{Adv}_{3.i} - \mathbf{Adv}'_{i.1}| \leq 4\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_1) + \tfrac{4}{p-1}$$

$H_{i.1} \to H_{i.2}$: In this step, we reverse the transition from **Game 0** to **Game 1** and thus replace PSim with PPro from game $H_{i.1}$ on. We choose all $[\mathbf{t}]_1$, $[\mathbf{w}]_1$ in tag queries from $\mathcal{L}^{\vee}_{\mathbf{A}_0,\mathbf{A}_1}$ with corresponding witness and can thus honestly generate proofs. Therefore,

$$|\mathbf{Adv}'_{i.2} - \mathbf{Adv}'_{i.1}| \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.2} \to H_{i.3}$: From game $H_{i.3}$ on, we introduce an additionally check in the verification oracle. Namely, VERO checks that $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \mathrm{span}([\mathbf{A}_0]_1) \vee \mathrm{span}([\mathbf{A}_1]_1)$. We can employ the soundness of Π to obtain

$$|\mathbf{Adv}'_{i.3} - \mathbf{Adv}'_{i.2}| \leq \mathbf{Adv}^{\mathsf{snd}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.3} \to H_{i.4}$: Let $\mathbf{A}_0^{\perp} \in \mathrm{orth}(\mathbf{A}_0)$ and $\mathbf{A}_1^{\perp} \in \mathrm{orth}(\mathbf{A}_1)$. We introduce an intermediary game $H_{i.3.1}$, where we replace the random function $\mathbb{F}_i : \{0,1\}^i \to \mathbb{Z}_p^2$ by

$$\mathbb{F}'_i : \{0,1\}^i \to \mathbb{Z}_p^2,\ \mathbb{F}'_i(v) := (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma_i(v)\ \Upsilon_i(v))^{\top}$$

where $v \leftarrow \{0,1\}^i$ is an i-bit string and $\Gamma_i, \Upsilon_i : \{0,1\}^i \to \mathbb{Z}_p$ are two independent random functions. With probability $1 - \Delta_{\mathcal{D}_1}$ the matrix $(\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})$ has full rank. In this case, going from game $H_{i.3}$ to game $H_{i.3.1}$ consists merely in a change of basis, thus, these two games are perfectly indistinguishable. We obtain

$$|\mathbf{Adv}'_{i.3.1} - \mathbf{Adv}'_{i.3}| \leq \Delta_{\mathcal{D}_1}$$

We now define $\mathbb{F}_{i+1} : \{0,1\}^{i+1} \to \mathbb{Z}_p^2$ s.t,

$$\mathbb{F}_{i+1}(v) := \begin{cases} (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma'_i(v_{|i})\ \Upsilon_i(v_{|i}))^{\top} & \text{if } v_{|i+1} = 0 \\ (\mathbf{A}_0^{\perp}|\mathbf{A}_1^{\perp})(\Gamma_i(v_{|i})\ \Upsilon'_i(v_{|i}))^{\top} & \text{otherwise} \end{cases}$$

where $\Gamma'_i, \Upsilon'_i : \{0,1\}^i \to \mathbb{Z}_p$ are fresh independent random functions. Now $\mathbb{F}_{i+1}$ constitutes a random function $\{0,1\}^{i+1} \to \mathbb{Z}_p^2$. Replacing $\mathbb{F}'_{i+1}(\mathsf{ctr}_{|i})$ by $\mathbb{F}_{i+1}(\mathsf{ctr}_{|i+1})$ does not show up in any of the tag queries, as we have

$$\mathbb{F}_{i+1}(\mathsf{ctr}_{|i+1})^{\top}[\mathbf{t}]_1 = \mathbb{F}_{i+1}(\mathsf{ctr}_{|i+1})^{\top}[\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_1 = \ldots = \mathbb{F}'_i(\mathsf{ctr}_{|i})^{\top}[\mathbf{A}_{\mathsf{ctr}_{i+1}}]_1 r_1$$

In the verification oracle, we check $[\mathbf{t}]_1$, $[\mathbf{w}]_1 \in \mathrm{span}([\mathbf{A}_0]_1) \vee \mathrm{span}([\mathbf{A}_1]_1)$. Let us define $d_{[\mathbf{t}]} = 0$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_0)$ and $d_{[\mathbf{t}]} = 1$ if $\mathbf{t} \in \mathrm{span}(\mathbf{A}_1)$, and replace $\mathbb{F}_i(\mathsf{ctr}_{|i})$ by $\mathbb{F}_{i+1}(\mathsf{ctr}_{|i}|d_{[\mathbf{t}]})$. Thus, by similar reasoning as for tag queries, the change does not show up in the final verification query either. We obtain

$$|\mathbf{Adv}'_{i.4} - \mathbf{Adv}'_{i.3}| \leq \Delta_{\mathcal{D}_1}$$

$H_{i.4} \to H_{i.5}$: From game $H_{i.5}$ on, we extend the set $\mathcal{S}$ in the verification oracle from $\mathcal{S}_{i.4} := \mathbb{F}_{i+1}(\mathsf{ctr}'_{|i}|d_{[\mathbf{t}]}) : \mathsf{ctr}' \leq \mathsf{ctr}$ to $\mathcal{S}_{i.5} := \mathbb{F}_{i+1}(\mathsf{ctr}'_{|i}|b) : \mathsf{ctr}' \leq \mathsf{ctr}, b \in \{0,1\}$. That is, we regard a verification query $([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$ as valid, if there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbb{F}_{i+1}(\mathsf{ctr}'_{|i}|b)\top[\mathbf{t}]_1$ for $b \in \{0,1\}$ arbitrary, instead of requiring $b = d_{[\mathbf{t}]}$. As changing the verification oracle does not change the view of the adversary before providing its output and as we have $\mathcal{S}_{i.4} \subseteq \mathcal{S}_{i.5}$, the transition from game $H_{i.4}$ to game $H_{i.5}$ can only increase the chance of the adversary. We thus have

$$\mathbf{Adv}'_{i.4} \leq \mathbf{Adv}'_{i.5}$$

$H_{i.5} \to H_{i.6}$: The difference between game $H_{i.5}$ and game $H_{i.6}$ is that in the latter, we only regard a verification query $([\mathbf{t}]_1, [\mathbf{w}]_1, \Omega, [\mathbf{u}']_1, [\mathbf{u}'']_1)$ as valid, if there exists a $\mathsf{ctr}' \leq \mathsf{ctr}$ such that $[\mathbf{u}']_1 = (\mathbf{k}_0 + \mathbb{F}_{i+1}(\mathsf{ctr}'_{|i}|\mathsf{ctr}'_{|i+1})^\top [\mathbf{t}]_1$ (instead of allowing the last bit to be arbitrary). As the only way an adversary can learn the image of $\mathbb{F}_{i+1}$ on a value is via tag queries and $\mathbb{F}_{i+1}$ is a random function, a union bound over the elements in Q_{tag} yields

$$|\mathbf{Adv}'_{i.5} - \mathbf{Adv}'_{i.6}| \leq \tfrac{Q}{p}$$

$H_{i.6} \to H_{i.7}$: The oracle VERO does not perform the additional check $[\mathbf{t}]_1, [\mathbf{w}]_1 \in \mathrm{span}([\mathbf{A}_0]_1 \vee \mathrm{span}([\mathbf{A}_1]_1)$ anymore from game $H_{i.7}$ on. This is justified by the soundness of Π. As in transition $H_{i.2} \to H_{i.3}$, we obtain

$$|\mathbf{Adv}'_{i.6} \leq \mathbf{Adv}'_{i.7}| \leq \mathbf{Adv}^{\mathsf{snd}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.7} \to H_{i.8}$: This transition is similar to transition **Game 0** to **Game 1**. For an adversary $\mathcal{B}_2$ we obtain

$$\mathbf{Adv}'_{i.7} - \mathbf{Adv}'_{i.8} \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

$H_{i.8} \to$**Game 3.(i+1)**: We switch $[\mathbf{t}]_1, [\mathbf{w}]_1$ generated by TAGO to uniformly random over $\mathbb{G}_1^2$, using the MDDH assumption first on $[\mathbf{A}_0]_1$, then on $[\mathbf{A}_1]_1$. Similarly than for the transition **Game 3.i** $\to H_{i.1}$, we obtain

$$|\mathbf{Adv}_{3.(i+1)} - \mathbf{Adv}'_{i.8}| \leq 4\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_2) + \tfrac{4}{p-1}$$

Game 3.(log(Q))$\to \mathsf{Exp}^{\mathsf{core}}_{1,\mathcal{A}}$: It is left to reverse the changes introduced in the transitions from **Game 0** to **Game 2** to end up at the experiment $\mathsf{Exp}^{\mathsf{core}}_{1,\mathcal{A}}$. In order to do so we introduce an intermediary **Game 4**, where we set $[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1$ and $[\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$ for $r_1, r_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$. This corresponds to reversing transition **Game 1** to **Game 2**. By the same reasoning for every adversary $\mathcal{A}$ we thus obtain

$$|\mathbf{Adv}_{3.(\log Q)} - \mathbf{Adv}_4| \leq 2\mathbf{Adv}^{\mathsf{MDDH}}_{\mathcal{D}_1,\mathbb{G}_s}(\mathcal{B}_1) + \tfrac{2}{p-1}$$

As $[\mathbf{t}]_1$, $[\mathbf{w}]_1$ are now chosen from $\mathrm{span}([\mathbf{A}_0]_1)$ again, we have

$$\mathbf{Adv}_4 - \mathbf{Adv}^{\mathsf{core}}_1 \leq \mathbf{Adv}^{\mathsf{zk}}_{\Pi}(\mathcal{B}_2)$$

□

Theorem 17. If the KerMDH and MDDH assumptions hold (Section 2.3.3 on page 11), the SPS-EQ in Figure 4.2 is EUF-CMA (Definition 27 on page 62).

Proof. We prove the claim using a sequence of Games, and we denote the advantage of the adversary in the j-th game as $\mathbf{Adv}_j$.
Game 0: This game is the original game, and we have:

$$\mathbf{Adv}_0 = \mathbf{Adv}^{\mathsf{EUF-CMA}}_{\mathsf{SPS-EQ}}(\mathcal{A})$$

Game 1: In this game, in Verify, we replace the first pairing verification with the following equation:

$$[\mathbf{u}_1^*]_1 = \mathbf{K}_0^\top[\mathbf{t}^*]_1 + \mathbf{K}^\top[\boldsymbol{m}^*]_1$$

For any signature $\sigma = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ that passes the original verification but not verification of **Game 1**, the value $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\boldsymbol{m}^*]_1$ is a non-zero vector in the kernel of $\mathbf{A}$. Thus, if $\mathcal{A}$ outputs such a signature, we can construct an adversary $\mathcal{B}$ that breaks the $\mathcal{D}_1$-KerMDH assumption in $\mathbb{G}_2$. First, the adversary $\mathcal{B}$ receives $(\mathsf{BG}, [\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [z]_2)$, samples all other parameters and simulates **Game 1** for $\mathcal{A}$. When $\mathcal{B}$ receives the forgery from $\mathcal{A}$ as $\sigma = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ for $[\boldsymbol{m}^*]_1$, he passes the following values to its own challenger: $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\boldsymbol{m}^*]_1$. We have:

$$|\mathbf{Adv}_1 - \mathbf{Adv}_0| \leq \mathbf{Adv}_{\mathcal{D}_1,\mathbb{G}_2}^{\mathsf{KerMDH}}(\mathcal{A})$$

Game 2: In this game, we set $\mathbf{K}_0 = \mathbf{K}_0 + \mathbf{k}_0(\mathbf{a}^\perp)^\top$ (in the key generation we can pick $\mathbf{k}_0 \in \mathbb{Z}_p^2$ and $\mathbf{K}_0 \in \mathbb{Z}_p^{2\times 2}$ and set $\mathbf{K}_0$; we have $\mathbf{a}^\perp\mathbf{A} = 0$). We compute $[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\boldsymbol{m}]_1 + \mathbf{a}^\perp(\mathbf{k}_0)^\top[\mathbf{t}]_1$ and $[\mathbf{u}_2]_1 = \mathbf{K}_0^\top[\mathbf{w}]_1 + \mathbf{a}^\perp(\mathbf{k}_0)^\top[\mathbf{w}]_1$. There is no difference to the previous game since both are distributed identically. So, we have:

$$\mathbf{Adv}_2 = \mathbf{Adv}_1$$

Game 3: In this game, we add the part of $\mathbb{F}(\mathsf{ctr})$ for $\mathsf{ctr} = \mathsf{ctr} + 1$, where $\mathbb{F}$ is a random function, and obtain $[\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{K}^\top[\boldsymbol{m}]_1 + \mathbf{a}\perp(\mathbf{k}_0 + \mathbb{F}(\mathsf{ctr}))^\top[\mathbf{t}] - 1$ and $[\mathbf{u}_2]_1 = \mathbf{K}\top_0[\mathbf{w}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbf{k}')^\top[\mathbf{w}]_1$. In the verification we have:

$$1 \leftarrow \mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, \pi)) \text{ and}$$
$$\exists\ \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}_1]_1 = \mathbf{K}_0^\top[\mathbf{t}]_1 + \mathbf{a}^\perp(\mathbf{k}_0 + \mathbb{F}(\mathsf{ctr}'))^\top + \mathbf{K}^\top[\boldsymbol{m}]_1$$

Let $\mathcal{A}$ be an adversary that distinguishes between **Game 3** and **Game 2**. We can construct an adversary $\mathcal{B}_1$ that breaks the core lemma. $\mathcal{B}_1$ receives $\mathsf{pp} = (BG, [\mathbf{A}_0]_1, \mathsf{crs})$ from $\mathsf{Exp}_{\beta,\mathcal{B}_1}^{\mathsf{core}}$. $\mathcal{B}_1$ picks $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$, $\mathbf{a}^\perp \in \mathrm{orth}(\mathbf{A})$, $\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$, $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{2\times \ell}$, and sends public key $\mathsf{pk} = ([\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$ to $\mathcal{A}$. $\mathcal{B}_1$ uses the oracle $\mathsf{TAG}()$ to construct the signing algorithm. This oracle takes no input and returns $\mathsf{tag} = (([\mathbf{t}]_1, [\mathbf{w}]_1, (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1), [\mathbf{u}']_1, [\mathbf{u}'']_1)$. Then $\mathcal{B}_1$ computes $[\mathbf{u}_1]_1 = \mathbf{K}_0[\mathbf{t}]_1 + \mathbf{a}^\perp[\mathbf{u}']_1 + \mathbf{K}^\top[\boldsymbol{m}]_1$, $[\mathbf{u}_2]_1 = \mathbf{K}_0^\top[\mathbf{w}]_1 + \mathbf{a}^\perp[\mathbf{u}'']_1$, and sends the signature $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ and tag $\tau = ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$ to $\mathcal{A}$. When the adversary $\mathcal{A}$ sends the forgery $([\boldsymbol{m}^*]_1, \sigma^*) = ([\mathbf{u}_1^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$, $\mathcal{B}$ returns 0 if $[\mathbf{u}_1]_1 = 0$; otherwise it checks whether there exists $[\mathbf{u}'^*]_1$ such that $[\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\boldsymbol{m}^*]_1 = \mathbf{a}^\perp[\mathbf{u}'^*]_1$. If it does not hold, then it returns 0 to $\mathcal{A}$, otherwise $\mathcal{B}_1$ computes $[\mathbf{u}'^*]_1$, and calls the verification oracle $\mathsf{VERO}()$ on the tag $\mathsf{tag}^* = ([\mathbf{u}'^*]_1, [\mathbf{t}^*]_1, \Omega_1^*, [z_0^*]_2, [z_1^*]_2, Z_1^*)$ and returns the answer to $\mathcal{A}$. Using the core lemma, we have:

$$\mathbf{Adv}_2 - \mathbf{Adv}_3 \leq \mathbf{Adv}_{\mathsf{BG}}^{\mathsf{core}}(\mathcal{B}_1)$$

Game 4: In this game, we pick r_1, r_2 from $\mathbb{Z}_p^*$ instead of $\mathbb{Z}_p$. The difference of advantage between **Game 3** and **Game 4** is bounded by the statistical distance between the two distributions of r_1, r_2. So, under Q adversarial queries, we have:

$$\mathbf{Adv}_4 - \mathbf{Adv}_3 \leq \frac{Q}{p}$$

Game 5: In this game, we pick $\tilde{\mathsf{ctr}} \xleftarrow{\$} [1, Q]$, and we add a condition $\mathsf{ctr}' = \tilde{\mathsf{ctr}}$ to verification. Actually, now we have this conditions:

$$1 \leftarrow \mathsf{PRVer}(\mathsf{crs}, [\mathbf{t}]_1, (\Omega_1, [z_0]_2, [z_1]_2, Z_1)) \text{ and}$$
$$\exists \mathsf{ctr}' \leq \mathsf{ctr} : [\mathbf{u}_1]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbb{F}(\mathsf{ctr}'))^\top + \mathbf{K}^\top [\boldsymbol{m}]_1$$

Since the view of the adversary is independent of ctr, we have

$$\mathbf{Adv}_5 = \frac{\mathbf{Adv}_4}{Q}$$

Game 6: In this game, we can replace $\mathbf{K}$ by $\mathbf{K} + \mathbf{v}(\mathbf{a}^\perp)^\top$ for $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_p^\ell$. Also, we replace $\{\mathbb{F}(i) : i \in [1, Q], i \neq \mathsf{ctr}\}$ by $\{\mathbb{F}(i) + \mathbf{w}_i : i \in [1, Q], i \neq \tilde{\mathsf{ctr}}\}$, for $\mathbf{w}_i \xleftarrow{\$} \mathbb{Z}_p^{2k}$ and $i \neq \mathsf{ctr}$. So, in each i-th query, where $i \neq \mathsf{ctr}$, we compute

$$[\mathbf{u}_1]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top)[\boldsymbol{m}_i]_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbb{F}(i) + \mathbf{w}_i)^\top [\mathbf{t}]_1$$

Also, for $\tilde{\mathsf{ctr}}$-th query for the message $[\boldsymbol{m}_{\tilde{\mathsf{ctr}}}]_1$, we compute

$$[\mathbf{u}_1]_1 = \mathbf{K}_0^\top [\mathbf{t}]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top)[\boldsymbol{m}_{\tilde{\mathsf{ctr}}}]_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbb{F}(\tilde{\mathsf{ctr}}) + \mathbf{w}_i)^\top [\mathbf{t}]_1$$

So, $\mathcal{A}$ must compute the following:

$$[\mathbf{u}_1^*]_1 = \mathbf{K}_0^\top [\mathbf{t}^*]_1 + (\mathbf{K}^\top + \mathbf{a}^\perp \mathbf{v}^\top)[\boldsymbol{m}^*]_1 + \mathbf{a}^\perp (\mathbf{k}_0 + \mathbb{F}(\tilde{\mathsf{ctr}}) + \mathbf{w}_i)^\top [\mathbf{t}^*]_1$$

Since $\mathbf{m}^* \neq [\mathbf{m}_{\tilde{\mathsf{ctr}}}]_\mathcal{R}$ (in different classes) by definition of the security game, we can argue $\mathbf{v}^\top \mathbf{m}^*$ and $\mathbf{v}^\top \mathbf{m}_{\tilde{\mathsf{ctr}}}$ are two independent values, uniformly random over $\mathbb{G}_1$. So, $\mathcal{A}$ only can guess it with probability of $\frac{1}{p}$. So, we have

$$\mathbf{Adv}_{\mathsf{SPS-EQ}}^{\mathsf{EUF-CMA}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{BG}}^{\mathsf{KerMDH}}(\mathcal{B}) + \mathbf{Adv}_{\mathsf{BG}}^{\mathsf{core}}(\mathcal{B}_1) + \frac{2Q}{p}$$

□

Theorem 18. The SPS-EQ in Figure 4.2 perfectly adapts signatures with respect to the message space (Definition 28 on page 63).

To prove Theorem 18, we follow almost verbatim the original proof from [KSD19].

Proof. For all $[\boldsymbol{m}]_1$ and $\mathsf{pk} = ([\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$, a signature σ=($[\mathbf{u}_1]_1$, $[\mathbf{t}]_1$, Ω_1 ,$[z_0]_2$, $[z_1]_2$, Z_1) generated according to the CRS ($[\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [z]_2$) satisfying the verification algorithm must be of the form: σ=($\mathbf{K}_0^\top [\mathbf{A}_0]_1 r_1 + \mathbf{K}^\top [\boldsymbol{m}]_1$, $[\mathbf{A}_0]_1 r_1$, $[\mathbf{A}_0] s_1$, $[\mathbf{A}_1] d_1^1 - z_1 [\mathbf{A}_0]_1 r_1$, $[z_0]_2 r_1 + [s_1]_2$, $[d_1^1]_2$, $[z_0]_2$, $[z_1]_2$, Z_1). A signature output by ChgRep has the form σ=($\mathbf{K}_0^\top [\mathbf{A}_0]_1 (\mu r_1 + \beta r_2) + \mathbf{K}^\top [\mu \boldsymbol{m}]_1$, $[\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$, $[\mathbf{A}_0]\alpha(\mu s_1 + \beta s_2)$, $[\mathbf{A}_1]\alpha(\mu d_1^1 + \beta d_1^2) - z_1[\mathbf{A}_0]_1 \alpha(\mu r_1 + \beta r_2)$, $\alpha([z_0]_2(\mu r_1 + \beta r_2) + \mu [s_1]_2 + \beta [s_2]_2)$, $\alpha(\mu [d_1^1]_2 + \beta [d_1^2]_2)$, $\alpha [z_0]_2$, $\alpha [z_1]_2$, αZ_1), for new independent randomness α, β and μ so is a random element in the space of all signatures. Furthermore, the signature output by ChgRep is distributed identically to a fresh signature on message $[\mu \boldsymbol{m}]_1$ output by Sign. □

4.5.3 How to Obtain a Mercurial Signature

In the following, we present the required changes to the previous construction to obtain a mercurial signature. It suffices to define the algorithms ConvertPK, ConvertSK and ChgRep as shown in Figure 4.4. All of the algorithms have been listed for ease of exposition, and the changes are highlighted in grey.

Regarding the security properties, we also need to consider perfect adaption with respect to the key space. Unfortunately, as observed by Colin Putman after the publication of [CLPK22], the proposed extension suffers the same issue as previous constructions [CL19, CL21], in which only a weak form of perfect adaption is supported. Instead of having perfect adaption under *malicious* keys in the honest parameters model, we have perfect adaption under *honestly* generated keys in the honest parameters model. We thank Colin Putman for the previous observation and dedicate the following lines to explain why perfect adaption only holds with respect to honestly generated keys.

Without loss of generality, according to the matrix distributions, any given CRS will be of the form: $[\mathbf{A}]_2 = \begin{pmatrix}[a]_2\\ [1]_2\end{pmatrix}$, $[\mathbf{A}_0]_1 = \begin{pmatrix}[a_0]_1\\ [1]_1\end{pmatrix}$, $[\mathbf{A}_1]_1 = \begin{pmatrix}[a_1]_1\\ [1]_1\end{pmatrix}$. Therefore, given

$$\mathsf{sk} = \left(\begin{pmatrix} b_{1,1} & b_{1,2}\\ b_{2,1} & b_{2,2}\end{pmatrix}, \begin{pmatrix} c_{1,1} & c_{1,2}\\ \vdots & \vdots\\ c_{\ell,1} & c_{\ell,2}\end{pmatrix}\right) \text{ and } \mathsf{pk} = \left(\begin{pmatrix}[ab_{1,1}+b_{1,2}]_2\\ [ab_{2,1}+b_{2,2}]_2\end{pmatrix}, \begin{pmatrix}[ac_{1,1}+c_{1,2}]_2\\ \vdots\\ [ac_{\ell,1}+c_{\ell,2}]_2\end{pmatrix}\right)$$

a randomisation of pk, $\mathsf{pk}' = \rho\mathsf{pk}$ will have the form:

$$\mathsf{pk}' = \left(\begin{pmatrix}[\rho ab_{1,1}+\rho b_{1,2}]_2\\ [\rho ab_{2,1}+\rho b_{2,2}]_2)\end{pmatrix}, \begin{pmatrix}([\rho ac_{1,1}+\rho c_{1,2}]_2)\\ \vdots\\ ([\rho ac_{\ell,1}+\rho c_{\ell,2}]_2)\end{pmatrix}\right)$$

With the above in mind, we observe that if the secret key is maliciously generated, for example, $c_{1,2} = \gamma c_{1,1}$ and $c_{2,2} = \gamma c_{2,1}$ for a randomly chosen γ, then the owner of the secret key will be able to identify randomisations of the corresponding public key. To do so, it would suffice to multiply the first row of pk' by γ and verify that it's equal to the second row. This can be done without knowing ρ or a. Hence, the construction can only provide perfect adaption under honestly generated keys (*i.e.,* assuming all elements from the secret key are randomly chosen).

Theorem 19. The mercurial signature construction from Figure 4.4 has perfect adaption of signatures under honestly generated keys in the honest parameters model (Definition 32 on page 64).

Proof. For all $[\boldsymbol{m}]_1$ and $\mathsf{pk} = ([\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$, a signature $\sigma = ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$ generated according to the CRS $([\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, [z]_2)$ satisfying the verification algorithm must be of the form: $\sigma = (\mathbf{K}_0^\top[\mathbf{A}_0]_1 r_1 + \mathbf{K}^\top[\boldsymbol{m}]_1,\ [\mathbf{A}_0]_1 r_1,\ [\mathbf{A}_0]s_1,\ [\mathbf{A}_1]d_1^1 - z_1[\mathbf{A}_0]_1 r_1,\ [z_0]_2 r_1 + [s_1]_2,\ [d_1^1]_2,\ [z_0]_2,\ [z_1]_2,\ Z_1)$. A signature output by ChgRep has the form $\sigma = (\rho\mathbf{K}_0^\top[\mathbf{A}_0]_1(\mu r_1 + \beta r_2) + \rho\mathbf{K}^\top[\mu\boldsymbol{m}]_1,\ [\mathbf{A}_0]_1(\mu r_1 + \beta r_2),\ [\mathbf{A}_0]\alpha(\mu s_1 + \beta s_2),\ [\mathbf{A}_1]\alpha(\mu d_1^1 + \beta d_1^2) - z_1[\mathbf{A}_0]_1\alpha(\mu r_1 + \beta r_2),\ \alpha([z_0]_2(\mu r_1 + \beta r_2) + \mu[s_1]_2 + \beta[s_2]_2),\ \alpha(\mu[d_1^1]_2 + \beta[d_1^2]_2),\ \alpha[z_0]_2,\ \alpha[z_1]_2,\ \alpha Z_1)$, for new independent randomness α, β, μ and ρ so is a random element in the space of all signatures.

```
SPS-EQ.PGen(1^λ):
  BG ←$ BGGen(1^λ); $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
  crs ←$ PGen(1^λ; BG)
  pp ← (BG, $[\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1$, crs)
  return pp

SPS-EQ.PTGen(1^λ):
  BG ←$ BGGen(1^λ); $\mathbf{A}, \mathbf{A}_0, \mathbf{A}_1 \xleftarrow{\$} \mathcal{D}_1$
  (crs, $\tau$) ←$ PTGen(1^λ; BG)
  pp ← (BG, $[\mathbf{A}]_2, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1$, crs)
  return (pp, $\tau$)

SPS-EQ.KGen(pp, 1^λ):
  $\mathbf{K}_0 \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$; $\mathbf{K} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times 2}$
  $[\mathbf{B}]_2 \leftarrow [\mathbf{K}_0]_2[\mathbf{A}]_2$
  $[\mathbf{C}]_2 \leftarrow [\mathbf{K}]_2[\mathbf{A}]_2$
  sk ← $(\mathbf{K}_0, \mathbf{K})$
  pk ← $([\mathbf{B}]_2, [\mathbf{C}]_2)$
  return (sk, pk)

SPS-EQ.Sign(pp, sk, $[\boldsymbol{m}]_1$):
  $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
  $[\mathbf{t}]_1 \leftarrow [\mathbf{A}_0]_1 r_1$; $[\mathbf{w}]_1 \leftarrow [\mathbf{A}_0]_1 r_2$
  $\Omega \leftarrow$ PPro(crs, $[\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2$)
  $(\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1) \leftarrow \Omega$
  $\mathbf{u}_1 \leftarrow \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top [\boldsymbol{m}]_1$
  $\mathbf{u}_2 \leftarrow \mathbf{K}_0^\top [\mathbf{w}]_1$
  $\sigma \leftarrow ([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1)$
  $\tau \leftarrow ([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2)$
  return $(\sigma, \tau)$

SPS-EQ.ConvertSK(sk, $\rho$):
  $(\mathbf{K}_0, \mathbf{K}) \leftarrow$ sk; return $(\rho\mathbf{K}_0, \rho\mathbf{K})$

SPS-EQ.Verify(pp, $[\boldsymbol{m}]_1, (\sigma, \tau)$, pk):
  $([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1) \leftarrow \sigma$
  check
    PRVer(crs, $[\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1$)
    $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) = e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\boldsymbol{m}]_1^\top, [\mathbf{C}]_2)$
  if $\tau \neq \perp$ check
    $([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \leftarrow \tau$
    PRVer(crs, $[\mathbf{w}]_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1$)
    $e([\mathbf{u}_2]_1^\top, [\mathbf{A}_2]) = e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
  return 1

SPS-EQ.ChgRep(pp, $[\boldsymbol{m}]_1, \sigma, \tau, \mu, \rho$, pk):
  $([\mathbf{u}_1]_1, [\mathbf{t}]_1, \Omega_1, [z_0]_2, [z_1]_2, Z_1) \leftarrow \sigma$
  $([\mathbf{u}_2]_1, [\mathbf{w}]_1, \Omega_2) \leftarrow \tau$
  $\Omega \leftarrow (\Omega_1, \Omega_2, [z_0]_2, [z_1]_2, Z_1)$
  check
    PVer(crs, $[\mathbf{t}]_1, [\mathbf{w}]_1, \Omega$)
    $e([\mathbf{u}_2]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{w}]_1^\top, [\mathbf{B}]_2)$
    $e([\mathbf{u}_1]_1^\top, [\mathbf{A}]_2) \neq e([\mathbf{t}]_1^\top, [\mathbf{B}]_2) + e([\boldsymbol{m}]_1^\top, [\mathbf{C}]_2)$
  $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_p^*$
  $[\mathbf{u}'_1]_1 \leftarrow \rho(\mu[\mathbf{u}_1]_1 + \beta[\mathbf{u}_2]_1)$
  $[\mathbf{t}']_1 \leftarrow \mu[\mathbf{t}]_1 + \beta[\mathbf{w}]_1 = [\mathbf{A}_0]_1(\mu r_1 + \beta r_2)$
  for all $i \in \{0, 1\}$
    $[z'_i]_2 \leftarrow \alpha[z_i]_2$
    $[\mathbf{a}'_i]_2 \leftarrow \alpha\mu[\mathbf{a}_i^1]_2 + \alpha\beta[\mathbf{a}_i^2]_2$
    $[d'_i]_1 \leftarrow \alpha\mu[d_i^1]_1 + \alpha\beta[d_i^2]_1$
  $\Omega' \leftarrow (([\mathbf{a}'_i]_1, [d'_i]_2, [z'_i]_2)_{i\in\{0,1\}}, \alpha Z_1)$
  $\sigma' \leftarrow ([\mathbf{u}'_1]_1, [\mathbf{t}']_1, \Omega')$
  return $(\mu[\boldsymbol{m}]_1, \sigma')$

SPS-EQ.ConvertPK(pk, $\rho$):
  $([\mathbf{B}]_2, [\mathbf{C}]_2) \leftarrow$ pk
  return $(\rho[\mathbf{B}]_2, \rho[\mathbf{C}]_2)$
```

Figure 4.4: A mercurial signature construction from standard assumptions.

Furthermore, the signature output by ChgRep is distributed identically to a fresh signature on message $[\mu\boldsymbol{m}]_1$ output by Sign(pp, ConvertSK(sk, ρ), $[\mu\boldsymbol{m}]_1$). □

Unforgeability for mercurial signatures gives the adversary freedom to produce forgeries under a different public key, as long as it belongs to the same equivalence class as the original public key. In this regard, unforgeability of our construction follows almost verbatim from Theorem 17.

Remark 3. Since our construction requires the use of a tag, to implement the delegatable credentials from [CL19] with our construction, users would be required to store the tags and to randomise them when delegating to another user (*i.e.,* tags need to be randomised and passed along for each delegation level with the corresponding signature). Therefore, users should verify the tag's correctness when obtaining a signature, although that's not required for verification during credential presentations.

4.6 Conclusions and Future Work

In this chapter, we recalled the concept of Structure-Preserving Signatures on Equivalence Classes, the main building block for subsequent chapters. Furthermore, we proposed a new SPS-EQ scheme based on the construction from [KSD19], where we adapt the SPS-EQ scheme by alleviating the need to build a QA-NIZK incorporating results from the recent NIZK framework of [CH20]. We also showed how to extend our construction to obtain the first mercurial signature in the standard model. However, our construction requires a CRS and suffers the same limitation as the previously known ones: only a weak form of anonymity is achieved with respect to the key space. For this reason, obtaining a construction in the standard model without a CRS or a stronger notion for perfect adaption with respect to the key space is an open problem.

5
Anonymous Credentials

Arguing that you don't care about privacy because you have nothing to hide is no different than saying you don't care about free speech because you have nothing to say.

— Edward Snowden

Recent regulations, such as the General Data Protection Regulation (GDPR) [PoEU16], state that collected and processed data should not be held or further used for other purposes than the ones for which it was originally intended to be used. In the digital world, after some information has been exchanged between a user and a service provider, it becomes difficult to guarantee the user that service providers behave according to regulations like GDPR. Therefore, designing protocols to manage at a fine-grained level the information that users can be requested to provide in order to access a given service is of utmost importance. This chapter proposes such protocols in the form of anonymous credentials. It relies on joint work with Aisling Connolly and Pascal Lafourcade, and it is based on [CLPK21].

5.1 Introduction

Considering access to online services, designing protocols to manage the information users can be requested to present is of utmost importance to protect the user. A first step in the literature developed the concept of *attribute-based credentials* (ABC) to model how users could show a credential containing a set of attributes to access different services.

Subsequently, the development of *anonymous* attribute-based credentials made it possible to protect the holders' identity when showing a credential. Users could present a credential disclosing no information other than that revealed by the attributes they choose to show (*anonymity*) while also ensuring that the provided information is authentic (*unforgeability*). Proposed alternatives consider a third property *unlinkability* which ensures that multiple showings of the same credential cannot be linked. Credential systems that support an arbitrary number of unlinkable showings are said to be *multi-show*. In contrast, those that only allow a single use of an issued credential in an unlinkable fashion are called *one-show*.

Initial progress was made with respect to one-show constructions. Here, *blind signatures* are issued on commitments to attributes so that users can later show the signature and disclose some of the attributes while proving knowledge of those left unrevealed. Examples include [Bra00, BL13, PZ13], and [FHKS16].

In the multi-show setting, pioneering constructions (based on Camenisch and Lysyanskaya's (CL) signatures [CL03, CL04]) such as the one underlying the Idemix credential system [Zur13] rely on randomising the signature to then prove in zero-knowledge the correspondence between the set of attributes (disclosed and undisclosed), and the signature. A major drawback of such an approach is that the zero-knowledge proof used during showings is of variable-length and may require multiple sub-proofs. On the other hand, more recent constructions (*e.g.,* [CL11, CDHK15, San20, HP20, TG20, DHS15a, FHS19]) apply other techniques based on different lines of work to adapt the signature and the message without using ZKPoK, providing constant-size showings.

The concept of ABC has been recently extended to consider multi-authority credentials (*e.g.,* [HP20, SAB$^+$19]), where users obtain a single credential for a set of attributes not necessarily issued by a single authority. This work considers the *classical* setting (each credential is issued by a single authority).

5.2 Contributions

We take the ABC framework from Fuchsbauer, Hanser and Slamanig [FHS19] as our starting point and propose a number of improvements, which we discuss next.

First, we extend the set-commitment scheme from [FHS19] to build a more expressive credential system. Our extension receives the name of Set-Commitment scheme supporting Disjoint Sets (SCDS), allowing a credential holder to prove that a given set of attributes is not encoded in the credential. We also present another extension that allows a user to outsource some of the computational cost to a verifier when presenting a credential. This second extension is a Proof of Exponentiation (PoE) compatible with our SCDS. Additionally, we develop a notion called *issuer-hiding*. This notion allows users to hide the information that relates them to an issuing organisation when presenting a credential. Finally, based on the contributions from the previous chapter, we present an ABC construction whose security is proven under standard assumptions (assuming a CRS).

5.3 Related Work

Constructions in the classical setting differ regarding their expressiveness, efficiency, security model, how they handle revocation, and whether or not they provide non-interactive features. Unfortunately, achieving all these properties simultaneously has been challenging and tends to rely on complex or non-standard assumptions.

Scheme	[CL04]	[CL11]	[CL13]	[CDHK15] & [FHS19]	[TG20]	[San20]	[HP20]	Our work
	Issuing n-attr. credential							
Comm.	$O(n)$	$O(n)$	$O(n)$	$\boldsymbol{O(1)}$	$O(n)$	$\boldsymbol{O(1)}$	$O(n)$	$\boldsymbol{O(1)}$
User	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Issuer	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
	Showing k-of-n attributes (selective disclosure)							
$\|$ek$\|$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$	$O(n)$	$O(n)$
Comm.	$O(n)$	$O(1)$	$O(k)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
User	$O(n)$	$O(n)$	$O(k)$	$O(n-k)$	$O(n-k)$	$O(n-k)$	$\boldsymbol{O(1)}$	$O(\max\{n-k,k\})$
Verifier	$O(n)$	$O(n)$	$O(k)$	$O(k)$	$O(k)$	$O(k)$	$O(n)$	$\boldsymbol{O(1)}$

Table 5.1: Asymptotic complexities of ABC systems where n is the number of attributes in the credential and k is the number of disclosed ones during a showing.

When considering state-of-the-art credential systems, there are five lines of work with respect to the underlying signature scheme that is used to build them:

- CL signatures [CL04]: Idemix [Zur13] and [TG20].
- Aggregatable signatures: [CL11] and [HP20].
- Sanitizable signatures: [CL13].
- Redactable signatures: [CDHK15] and [San20].
- Structure-Preserving Signatures on Equivalence Classes (SPS-EQ): [FHS19].

The previous approaches present some limitations, as explained below.

Concrete efficiency. Most alternatives provide similar efficiency at the asymptotic level. We recall in Table 5.1 the asymptotic complexities for issuing and showing protocols, considering recent credential systems from each of the previous lines of work, including our work. For showing protocols, we consider the selective disclosure of attributes (*i.e.,* the ability to show multiple attributes while hiding others during a showing). While [HP20] is the only one with $O(1)$ complexity for the user during a showing, this comes at the cost of a more expensive verifier. Our work achieves $O(1)$ complexity for the verifier but keeping better asymptotics for the user. A detailed comparison on the concrete efficiency of ABC's (including an implementation benchmark) was provided in [TG20]. Since the recent works from [San20] and [HP20] where not available at that time, we provide in Section 5.7 an updated comparison for the most efficient constructions from Table 5.1.

Proof settings. Most of the previous work relies on security proofs in the GGM. In Table 5.2, we classify the ABC from Table 5.1 in terms of their assumed security models. Some can be instantiated with or without the ROM and a CRS. Ideally, one would like to have practically efficient (*e.g.,* not only on the asymptotic level) and expressive constructions in the standard model without requiring a CRS or the ROM. This chapter provides an alternative to [TG20], building on [FHS19] without relying on the GGM.

Issuer-hiding. Showing protocols of previous constructions (including [TG20]), verify signatures with a key that belongs to the authority that issued the credential. This restricts the use of ABC in scenarios where one would like to verify a valid credential without linking it to a particular authority. Our issuer-hiding proposal overcomes this limitation, which was not considered in previous works.

Scheme	[CL04]	[CL11]	[CL13]	[CDHK15]	[FHS19]	[TG20]	[San20]	[HP20]	Our work
Standard model	✓	✗	✓	✗	✗	✓	✗	✗	✓
Without CRS	✓	✓	✓	✗	✓	✓	✓	✗	✗
Without RO	✓	✗	✗	✓	✓	✓	✓	✗	✓

Table 5.2: Classification of ABC schemes, indicating whether security proofs are in the standard model, if a CRS is required and if the ROM is used.

5.4 Accumulators and Set-Commitments

In [DHS15b], Derler, Hanser and Slamanig revisited the notion of cryptographic accumulators and proposed a unified formal model which included the notions of undeniability and indistinguishability for accumulators, complementing the classical ones of correctness and collision-freeness. Moreover, they showed how to construct a commitment scheme using an indistinguishable accumulator in a black-box manner. The relation stems from the fact that the indistinguishability and collision-freeness notions for accumulators resemble those of hiding and binding for commitments.

In another work [HS14], Hanser and Slamanig built an ABC with constant-size credentials and constant-size showings (for selective disclosure of attributes) based on a polynomial commitment scheme with factor openings. They departed from the work of Kate *et al.* on constant-size polynomial commitments [KZG10] with the following observations; (1) If a credential is seen as a set of attributes mapped to the roots of a monic polynomial, then one can generate a polynomial commitment of constant-size to represent the credential using the approach from [KZG10]. (2) Instead of evaluating the polynomial at certain points, what is important to prove possession of an attribute is to open factors of the polynomial instead. (3) If one can open multiple factors in constant-size, a showing involving a selective disclosure of attributes can also be done in constant-size.
As a result, they proposed an indistinguishable bilinear accumulator ([Ngu05]) with batch membership proofs (*i.e.,* factor opening), which was subsequently re-stated as a *set-commitment* scheme in a follow-up work [FHS19].

A drawback of the ABC from [FHS19] is that the achieved level of *expressiveness* is limited. For example, it only allows show proofs for the conjunction of attributes in arbitrary subsets of attributes encoded in the credential (selective disclosure). Furthermore, verification involves a number of exponentiations that are linear in the size of the subset to be verified. This is undesirable if verification must be fast.

Thakur [Tha19] proposed a series of protocols for batch membership and non-membership proofs for bilinear accumulators using *proofs of exponentiation* (an idea previously introduced for accumulators in groups of unknown order by Boneh *et al.* [BBF19] and by Wesolowski [Wes19]) to shift the computational cost from the verifier to the prover. The main idea is to replace some of the exponentiations by a single polynomial division, using a non-interactive proof obtained via the Fiat-Shamir transform.

Batch proofs in the bilinear accumulator setting can be traced back to Papamanthou *et al.* [PTT11] and Ghosh *et al.* [GOP$^+$16]. The latter presents the same underlying ideas of the (non)membership proofs provided by Thakur, and a *Zero-Knowledge Dynamic Universal Accumulator*, which strengthens the notion of indistinguishability using the randomisation ideas from [DHS15b] (see Figure 3 and algorithms 6 and 7 from [GOP$^+$16], respectively). Moreover, Ghosh *et al.* proved that the notion of indistinguishability from [DHS15b] is equivalent to a stronger notion which they call zero-knowledge for accumulators, if and only if the accumulator is *static*. On the other hand, if the accumulator is *dynamic*, they prove that their notion of zero-knowledge is strictly stronger than the one of indistinguishability in the vein of [DHS15b]. In this sense, we point out that the vector commitment scheme from [Tha19] can also be hiding if the accumulator being used is adapted following the construction from [GOP$^+$16].

More recently, a new set-commitment scheme, including set intersection and set difference operations, was proposed in [TG20]. It provides more expressiveness than the one from [FHS19] but under a weaker hiding notion.

In the following, we present a new set-commitment scheme that is more expressive than the one in [FHS19] and almost as expressive as [TG20], but supporting a more efficient verification and using a stronger hiding notion.

5.4.1 A Set-Commitment Scheme Supporting Disjoint Sets

We extend the set-commitment scheme in [FHS19] to support *non-membership proofs* for disjoint sets while also including an optional *proof of exponentiation*, where most of the exponentiations are outsourced to the prover rather than being performed by the verifier. To do so, we borrow the previously mentioned ideas in [DHS15b], [GOP$^+$16] and [Tha19] (indistinguishable accumulators, non-membership batch proofs, and proofs of exponentiations, respectively), adapting them to the Type-3 setting. Our set-commitment scheme supporting disjoint sets is described next.

SCDS Syntax A *set-commitment scheme supporting disjoint sets* (SCDS) consists of the following p.p.t algorithms:

$\mathsf{Setup}(1^\lambda, 1^q)$ is a probabilistic algorithm which takes as input a security parameter λ and an upper bound q for the cardinality of committed sets, both in unary form. It outputs public parameters pp (including an evaluation key ek), and discards the trapdoor key s used to generate them. $\mathbb{Z}_p^* \setminus \{s\}$ defines the domain of set elements for sets of maximum cardinality q.

$\mathsf{TSetup}(1^\lambda, 1^q)$ is equivalent to Setup but also returns the trapdoor key s.

$\mathsf{Commit}(\mathsf{pp}, \mathcal{X})$ is a probabilistic algorithm which takes as input pp and a set $\mathcal{X}$ with $1 \leq |\mathcal{X}| \leq q$. It outputs a commitment C on set $\mathcal{X}$ and opening information O.

$\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O)$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, and opening information O. It outputs 1 if and only if O is a valid opening of C on $\mathcal{X}$.

$\mathsf{OpenSS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{S})$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, opening information O, and a non-empty set $\mathcal{S}$. If $\mathcal{S}$ is a subset of $\mathcal{X}$ committed to in C, OpenSS outputs a witness wss that attests to it. Otherwise, outputs $\perp$.

$\mathsf{OpenDS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})$ is a deterministic algorithm which takes as input pp, a commitment C, a set $\mathcal{X}$, opening information O, and a non-empty set $\mathcal{D}$. If $\mathcal{D}$ is disjoint from $\mathcal{X}$ committed to in C, OpenDS outputs a witness wds that attests to it. Otherwise, outputs $\perp$.

$\mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wss})$ is a deterministic algorithm which takes as input pp, a commitment C, a non-empty set $\mathcal{S}$, and a witness wss. If wss is a valid witness for $\mathcal{S}$ a subset of the set committed to in C, it outputs 1 and otherwise $\perp$.

$\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \mathsf{wds})$ takes as input pp, a commitment C, a non-empty set $\mathcal{D}$, and a witness wss. If wds is a valid witness for $\mathcal{D}$ being disjoint from the set committed to in C, it outputs 1 and otherwise $\perp$.

$\mathsf{PoE}(\mathsf{pp}, \mathcal{X}, \alpha)$ takes as input pp, a non-empty set $\mathcal{X}$, and a randomly-chosen value α. It computes a proof of exponentiation for the characteristic polynomial of $\mathcal{X}$ and outputs a proof π_Q and a witness Q.

A SCDS scheme is *secure* if it satisfies the properties of correctness, binding, hiding, and soundness. These notions are defined next, modified to suit the scheme, but following the usual convention for set-commitment schemes.

Correctness of SCDS schemes requires that for all $q > 0$, all $\lambda > 0$, all $\mathsf{pp} \in [\mathsf{Setup}(1^\lambda, 1^q)]$, all non-empty $\mathcal{S} \subseteq \mathcal{X}$ and all non-empty $\mathcal{D} : \mathcal{D} \cap \mathcal{X} = \emptyset$:

1. $\Pr\left[\ (C,O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp},\mathcal{X}) \ : \ \mathsf{Open}(\mathsf{pp},C,\mathcal{X},O) = 1\ \right] = 1$
2. $\Pr\left[\begin{array}{l}(C,O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp},\mathcal{X});\\ \mathsf{wss} \leftarrow \mathsf{OpenSS}(\mathsf{pp},C,\mathcal{X},O,\mathcal{S})\end{array} : \ \mathsf{VerifySS}(\mathsf{pp},C,\mathcal{S},\mathsf{wss}) = 1\right] = 1$
3. $\Pr\left[\begin{array}{l}(C,O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp},\mathcal{X});\\ \mathsf{wds} \leftarrow \mathsf{OpenDS}(\mathsf{pp},C,\mathcal{X},O,\mathcal{D})\end{array} : \ \mathsf{VerifyDS}(\mathsf{pp},C,\mathcal{D},\mathsf{wds}) = 1\right] = 1$

Definition 33: Binding

An SCDS scheme is *binding* if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$, the following probability is negligible,

$$\Pr\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q),\\ (C,\mathcal{X},O,\mathcal{X}',O') \xleftarrow{\$} \mathcal{A}(\mathsf{pp})\end{array} : \begin{array}{r}\mathsf{Open}(\mathsf{pp},C,\mathcal{X},O) = 1 \ \wedge\\ \mathsf{Open}(\mathsf{pp},C,\mathcal{X}',O') = 1 \ \wedge \ \mathcal{X} \neq \mathcal{X}'\end{array}\right]$$

Similar to [FHS19], we also strengthen the standard notion of hiding by allowing the adversary to have oracle access to some algorithms in the scheme, following the definition for indistinguishable accumulators from [DHS15b].

Definition 34: Hiding

An SCDS scheme is *hiding* if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$ with access to $\mathcal{O}_{\mathsf{SS}}$, an opening oracle which allows queries for sets $\mathcal{X}' \subseteq \mathcal{X}_0 \cap \mathcal{X}_1$, and to $\mathcal{O}_{\mathsf{DS}}$, for sets $\mathcal{X}'$ s.t. $\mathcal{X}' \cap \{\mathcal{X}_0 \cup \mathcal{X}_1\} = \emptyset$, there is a negligible function

$\epsilon(\cdot)$ such that:

$$\left| \Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}; \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathcal{X}_0, \mathcal{X}_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}); \\ (C, O) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, \mathcal{X}_b); \\ b^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{SS}}(\mathsf{pp}, C, \mathcal{X}_b, O, \cdot), \mathcal{O}_{\mathsf{DS}}(\mathsf{pp}, C, \mathcal{X}_b, O, \cdot)}(\mathsf{st}, C) \end{array} : b^* = b\right] - \frac{1}{2} \right| \leq \epsilon(\lambda)$$

where $\mathcal{X}_0$ and $\mathcal{X}_1$ are two distinct sets s.t. $1 \leq |\mathcal{X}_b| \leq q$.
If the above holds for $\epsilon = 0$, the scheme is said to be perfectly hiding.

While the binding notion says it should be infeasible to produce two distinct valid openings for the same commitment, the following notion states that it should be infeasible to produce valid witnesses for invalid sets.

Definition 35: Soundness

An SCDS scheme is sound if for all $q > 0$ and all p.p.t adversaries $\mathcal{A}$, the following probabilities are negligible,

1. $$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (C, \mathcal{X}, O, \mathcal{S}, \mathsf{wss}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}) \end{array} : \begin{array}{r} \mathcal{S} \not\subseteq \mathcal{X} \ \wedge \ \mathsf{OpenSS}(\mathsf{pp}, C, \mathcal{X}, O) = 1 \\ \wedge \ \mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wss}) = 1 \end{array}\right]$$

2. $$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (C, \mathcal{X}, O, \mathcal{D}, \mathsf{wds}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}) \end{array} : \begin{array}{r} \mathcal{D} \cap \mathcal{X} \neq \emptyset \ \wedge \ \mathsf{OpenDS}(\mathsf{pp}, C, \mathcal{X}, O) \\ = 1 \ \wedge \ \mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \mathsf{wds}) = 1 \end{array}\right]$$

Our construction is presented in Figure 5.1. As in [FHS19], we use a special opening for the case in which the committed set contains the trapdoor to achieve perfect correctness and perfect hiding. To prove that a set is disjoint with respect to the committed set, the EEA is computed to obtain the Bézout coefficients q_1 and q_2. This way, equality is checked by randomising q_1 and q_2, using a single PPE. Finally, the PoE computes a polynomial division, producing the corresponding proof.

Theorem 20. The SCDS construction from Figure 5.1 is correct and perfectly hiding. Furthermore, if the q-co-DL and q-co-GSDH assumptions hold (Section 2.3.3 on page 11), it is computationally binding and sound, respectively.

Proof. The proof strategy follows closely that of [FHS19], which we adapt to consider disjoint sets.

Correctness. The first two cases are the same as in the original proof from [FHS19] (Theorem 3). For the third one, let $(\mathsf{pp}, s, C, \mathcal{X}, O, \mathcal{D})$ be a tuple representing a valid instance. $\mathsf{OpenDS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})$ returns $\mathsf{wss} = \perp$ if $s \in \mathcal{X}$ and $\mathsf{wss} = (w_0, w_1) = ((r^{-1}q_1'(s))P_2, q_2'(s)P_1)$ otherwise. If $s \in \mathcal{X}$, $\mathsf{VerifyDS}$ returns 1 as $\mathsf{wss} = \perp$. If $s \notin \mathcal{X}$, $\mathsf{VerifyDS}$ returns 1 if $e(C, w_0) + e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) = e(P_1, P_2)$. As expected,

$$\begin{aligned} e(C, w_0) + e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) &= e(r\mathsf{Ch}_{\mathcal{X}}(s)P_1, (r^{-1}q_1'(s))P_2) + e(q_2'(s)P_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(P_1, P_2)^{\mathsf{Ch}_{\mathcal{X}}(s)q_1'(s) + q_2'(s)\mathsf{Ch}_{\mathcal{D}}(s)} \\ &= e(P_1, P_2)^{\mathsf{Ch}_{\mathcal{X}}(s)q_1(s) + \mathsf{Ch}_{\mathcal{D}}(s)q_2(s)} \\ &= e(P_1, P_2) \end{aligned}$$

```
$\underline{\mathsf{SCDS.Setup}(1^\lambda, 1^q)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{pp} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i\in[q]})$
**return** $\mathsf{pp}$

$\underline{\mathsf{SCDS.TSetup}(1^\lambda, 1^q)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathsf{pp} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i\in[q]})$
**return** $(\mathsf{pp}, s)$

$\underline{\mathsf{SCDS.PoE}(\mathsf{pp}, \mathcal{X}, \alpha)}$:
$Q \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)P_2$; Let $h(X)$ and $\beta$ s.t.
$\mathsf{Ch}_{\mathcal{X}}(X)=(X+\alpha)\cdot h(X)+\beta$
$\pi_Q \leftarrow h(s)P_2$
**return** $(\pi_Q, Q)$

$\underline{\mathsf{SCDS.Commit}(\mathsf{pp}, \mathcal{X})}$:
**if** $|\mathcal{X}| > q$ **return** $\perp$; $r \xleftarrow{\$} \mathbb{Z}_p^*$
**if** $\exists\ s' \in \mathcal{X} : s'P_1 = sP_1$
    $C \leftarrow rP_1$; $O \leftarrow (1,(r,s'))$
**else** $C \leftarrow r\cdot\mathsf{Ch}_{\mathcal{X}}(s)P_1$; $O \leftarrow (0,r)$
**return** $(C, O)$

$\underline{\mathsf{SCDS.Open}(\mathsf{pp}, C, \mathcal{X}, O)}$:
**if** $O = (1,(r,s')) \wedge\ s'P_1 = sP_1$
    **if** $C = rP_1$ **return** 1 **else** 0
**if** $O = (0,r)$
    **if** $C = r\cdot\mathsf{Ch}_{\mathcal{X}}(s)P_1$ **return** 1 **else** 0

$\underline{\mathsf{SCDS.OpenSS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{S})}$:
**if** $\mathsf{SCDS.Open}(C, \mathcal{X}, O) = 0\ \vee$
    $\mathcal{S} \not\subseteq \mathcal{X} \vee \mathcal{S} = \emptyset$ **return** $\perp$
**if** $O = (1,(r,s'))$
    **if** $s' \notin \mathcal{S}$ **return** $\mathsf{Ch}_{\mathcal{S}}(s')^{-1}C$
**if** $O = (0,r)$ **return** $r\cdot\mathsf{Ch}_{\mathcal{X}\setminus\mathcal{S}}(s)P_1$
**else return** $\perp$

$\underline{\mathsf{SCDS.VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wss}, [\mathsf{PoE}])}$:
**if** $(\mathcal{S} = \emptyset \wedge \mathsf{wss} = \perp)$ **return** 1
**if** $\exists\ s' \in \mathcal{S} : s'P_1 = sP_1$
    **if** $\mathsf{wss} = \perp$ **return** 1 **else** 0
**if** $\mathsf{PoE} = \perp$
    **return** $e(\mathsf{wss}, \mathsf{Ch}_{\mathcal{S}}(s)P_2) = e(C, P_2)$
**else**
    $(\alpha, \pi_Q, Q) \leftarrow \mathsf{PoE}$
    $\beta \leftarrow \mathsf{Ch}_{\mathcal{S}}(X)(\mathrm{mod}\ (X+\alpha))$
    **return** $e(sP_1{+}\alpha P_1, \pi_Q){+}e(\beta P_1, P_2)$
    $= e(P_1, Q)\ \wedge\ e(\mathsf{wss}, Q) = e(C, P_2)$

$\underline{\mathsf{SCDS.OpenDS}(\mathsf{pp}, C, \mathcal{X}, O, \mathcal{D})}$:
**if** $(t = 0 \vee |\mathcal{D} \cap \mathcal{X}| > 0)$ **return** $\perp$
**if** $O = (1,(r,s'))$
    **if** $s' \in \mathcal{D}$ **return** $\perp$ **else**
    $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$; $(w_0, w_1) \leftarrow (\gamma P_2, \frac{1-\gamma\cdot r}{\mathsf{Ch}_{\mathcal{D}}(s)}P_1)$
**if** $O = (0,r)$
    $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$; Let $q_1(X)$ and $q_2(X)$ s.t.
    $\mathsf{Ch}_{\mathcal{X}}(X)\cdot q_1(X) + \mathsf{Ch}_{\mathcal{D}}(X)\cdot q_2(X) = 1$
    $q_1'(s) \leftarrow q_1(s) + \gamma\cdot\mathsf{Ch}_{\mathcal{D}}(s)$
    $q_2'(s) \leftarrow q_2(s) - \gamma\cdot\mathsf{Ch}_{\mathcal{X}}(s)$
    $(w_0, w_1) \leftarrow ((r^{-1}\cdot q_1'(s))P_2, q_2'(s)P_1)$
**return** $(w_0, w_1)$

$\underline{\mathsf{SCDS.VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \mathsf{wds}, [\mathsf{PoE}])}$:
**if** $(\mathcal{D} = \emptyset \wedge \mathsf{wds} = \perp)$ **return** 1
**if** $\exists\ s' \in \mathcal{D} : s'P_1 = sP_1$
    **if** $\mathsf{wds} = \perp$ **return** 1 **else** 0
$(w_0, w_1) \leftarrow \mathsf{wds}$
**if** $\mathsf{PoE} = \perp$ **return**
    $e(C, w_0){+}e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2){=}e(P_1, P_2)$
**else**
    $(\alpha, \pi_Q, Q) \leftarrow \mathsf{PoE}$
    $\beta \leftarrow \mathsf{Ch}_{\mathcal{D}}(X)(\mathrm{mod}\ (X+\alpha))$
    **return** $e(sP_1{+}\alpha P_1, \pi_Q){+}e(\beta P_1, P_2)$
    $=e(P_1, Q)\ \wedge\ e(C, w_0){+}e(w_1, Q){=}e(P_1, P_2)$
```

Figure 5.1: Our SCDS construction

Binding. We refer the reader to [FHS19] (Theorem 4) since it's the same proof (the modifications introduced in our construction did not change the algorithms Setup and Open).

Hiding. Following the approach from [FHS19], we consider the view of an unbounded adversary $\mathcal{A}$ in the hiding experiment and assume w.l.o.g that every query $\mathcal{S}$ to the $\mathcal{O}_{\mathsf{SS}}$ oracle satisfies $\mathcal{S} \subset \mathbb{Z}_p$ and $\emptyset \neq \mathcal{S} \subseteq (\mathcal{X}_0 \cap \mathcal{X}_1)$. Similarly, every query $\mathcal{D}$ to the $\mathcal{O}_{\mathsf{DS}}$ oracle satisfies $\mathcal{D} \subset \mathbb{Z}_p$ and $\emptyset \neq \mathcal{D} \cap \{\mathcal{X}_0 \cup \mathcal{X}_1\} = \emptyset$. To prove perfect hiding, the results from the adversary queries should be independent of b. In the following, we will prove that this is the case for the queries made to the oracle $\mathcal{O}_{\mathsf{DS}}$. We omit to prove here the case for queries made to $\mathcal{O}_{\mathsf{SS}}$ as the corresponding proof can be found almost verbatim in [FHS19] (Theorem 6).

(1) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ with $s \in \mathcal{X}_0 \cap \mathcal{X}_1$: Note that for all queries $\mathcal{D}_j$, we have $s \notin \mathcal{D}_j$, and for both $b \in \{0,1\}$, $C_b = r_b P_1$ is uniformly random in $\mathbb{G}_1^*$ for some $r_b \in \mathbb{Z}_p^*$. Furthermore, jth query $\mathcal{D}_j$ to $\mathcal{O}_{\mathsf{DS}}$ is answered with $\mathsf{wds}_{j,b} = (\gamma P_2, \frac{1-\gamma \cdot r_b}{\mathsf{Ch}_{\mathcal{D}_j}(s)} P_1, \pi_Q, Q)$ for a uniformly random $\gamma \in \mathbb{Z}_p^*$, so it does not depend on the bit b. Thus it is information-theoretically hidden.

(2) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ s.t. s is contained in one of the sets; say $s \in \mathcal{X}_0$: As in the previous case, for all queries $\mathcal{D}_j$, we have $s \notin \mathcal{D}_j$. If $b = 0$ then $\mathcal{A}$ receives a uniformly random $C_0 = r_0 P_1$ in $\mathbb{G}_1^*$ for some $r_0 \in \mathbb{Z}_p^*$, and when it queries $\mathcal{D}_j$ to the $\mathcal{O}_{\mathsf{DS}}$ oracle, it receives $\mathsf{wds}_{j,0} = (\gamma P_2, \frac{1-\gamma \cdot r_0}{\mathsf{Ch}_{\mathcal{D}_j}(s)} P_1, \pi_Q, Q)$ for a uniformly random $\gamma \in \mathbb{Z}_p^*$. If $b = 1$ then $\mathcal{A}$ receives $C_1 = \mathsf{Ch}_{\mathcal{X}_1}(s) \cdot r_1 P_1$ for a random $r_1 \in \mathbb{Z}_p^*$ and the jth query $\mathcal{D}_j$ to $\mathcal{O}_{\mathsf{DS}}$ is answered with $\mathsf{wds}_{j,1} = (q_1'(s).\frac{1}{r_1} P_2, q_2'(s) P_1, \pi_Q, Q)$. In this case, $q_1'(s) = q_1(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{X}_1}(s)$, and $q_2'(s) = q_2(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{D}_j}(s)$ for a random $\gamma \in \mathbb{Z}_p^*$ so both witnessess $\mathsf{wds}_{j,0}$ and $\mathsf{wds}_{j,1}$ are indistinguishable and do not depend on the bit b. Therefore, b is information-theoretically hidden from $\mathcal{A}$.

(3) $\mathcal{A}$ chooses $\mathcal{X}_0, \mathcal{X}_1$ with $s \notin \mathcal{X}_0 \cup \mathcal{X}_1$: For both $b \in \{0,1\}$: $C_b = \mathsf{Ch}_{\mathcal{X}_b}(s) r_b P_1$ for a random $r_b \in \mathbb{Z}_p^*$. If $s \notin \mathcal{D}_j'$ the jth query is answered with $\mathsf{wds}_{j,b} = (q_1'(s)\frac{1}{r_b} P_2, q_2'(s) P_1, \pi_Q, Q)$. If $s \in \mathcal{D}_j'$: the jth query is answered with $\mathsf{wds}_{j,b} = (q_1'(s)\frac{1}{r_b} P_2, q_2'(s) P_1, \pi_Q, Q)$. Observe that in both cases the first witness component $q_1'(s).\frac{1}{r_b} P_2$ perfectly hides b because $q_1'(s) = q_1(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{X}_b}(s)$ is uniformly random for $\gamma \xleftarrow{\$} \mathbb{Z}_p^*$. Similarly, the second component $q_2'(s) = q_2(s) + \gamma \cdot \mathsf{Ch}_{\mathcal{D}_j}(s)$ is also uniformly random. We conclude that b is information theoretically hidden from $\mathcal{A}$.

Soundness. We prove both equations in the soundness definition by reduction to the q-co-GSDH assumption. To do so, we consider an adversary $\mathcal{B}$, which on input an instance $I = (\mathsf{BG}, s)$, sets $\mathsf{pp} \leftarrow \mathsf{BG}$ and runs $\mathcal{A}(\mathsf{pp})$ in the soundness game.

(1) We assume that $\mathcal{A}$ is able to output $(C, O, \mathcal{X}, \mathcal{S}, \mathsf{wss})$ s.t. $\mathcal{X} \nsubseteq \mathcal{S}$, $\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1$ and $\mathsf{VerifySS}(\mathsf{pp}, C, \mathcal{S}, \mathsf{wss}) = 1$.

Following the approach from [FHS19], we prove here the second inequality in the soundness definition considering an adversary $\mathcal{B}$ which on input an instance $I = (\mathsf{BG}, s)$, sets $\mathsf{pp} \leftarrow \mathsf{BG}$ and runs $\mathcal{A}(\mathsf{pp})$ in the soundness game. We omit to prove here the first inequality as the corresponding proof can be found almost verbatim in [FHS19] (Theorem 5).

(2) We now assume $\mathcal{A}$ is able to output $(C, O, \mathcal{X}, \mathcal{D}, \mathsf{wds})$ s.t. $\mathcal{X} \cap \mathcal{D} \neq \emptyset$, $\mathsf{Open}(\mathsf{pp}, C, \mathcal{X}, O) = 1$ and $\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \mathsf{wds}) = 1$.

(2.1) $s \notin \{\mathcal{X} \cup \mathcal{D}\}$: In this case, observe that $\exists\ c \in \mathbb{Z}_p^*$ s.t $s \neq c \in \mathcal{X} \cap \mathcal{D}$. Hence, the adversary can compute polynomials $q_1(s)$ and $q_2(s)$ s.t $\mathsf{Ch}_{\mathcal{X}}(s) = (c + X)q_1(s)$ and $\mathsf{Ch}_{\mathcal{D}}(s) = (c + X)q_2(s)$. Since $\mathsf{VerifyDS}(\mathsf{pp}, C, \mathcal{D}, \mathsf{wds}) = 1$, we have that:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(\mathsf{Ch}_{\mathcal{X}}(s) \cdot rP_1, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e((c+s)q_1(s) \cdot rP_1, w_0) \cdot e(w_1, (c+s)q_2(s)P_2) \\ &= e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2)^{(c+s)} \end{aligned}$$

Hence we have $e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2) = e(P_1, P_2)^{\frac{1}{(c+s)}}$.

$\mathcal{A}$ is able to efficiently compute $q_1(s)$ and $q_2(s)$, and so the left side of the last equation can also be efficiently computed by $\mathcal{A}$. It follows that $\mathcal{B}$ can output the pair $(c, e(q_1(s) \cdot rP_1, w_0) \cdot e(w_1, q_2(s)P_2))$ to break the q-co-GSDH assumption.

(2.2) $s \in \{\mathcal{X} \cap \mathcal{D}\}$: We have that $C = \gamma P_1$ for a random $\gamma \in \mathbb{Z}_p^*$ and that s is a root of $\mathsf{Ch}_{\mathcal{D}}(s)$. Therefore, the verification equation can be written as follows:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(C, w_0) \cdot e(w_1, [\mathsf{Id}]_2) \\ &= e(C \cdot w_1, w_0) \end{aligned}$$

Since $\mathcal{B}$ can efficiently compute the right side of the previous equation, $\mathcal{A}$ can also output a solution $(c, e(C \cdot w_1, w_0)^{\frac{1}{c+s}})$ to the q-co-GSDH problem.

(2.3) $s \in \mathcal{X}\ \wedge\ s \notin \mathcal{D}$: As before, $C = \gamma P_1$ for a random $\gamma \in \mathbb{Z}_p^*$, but we also have that $\exists\ c \in \mathbb{Z}_p^*$ s.t $s \neq c \in \mathcal{X} \cap \mathcal{D}$, and we can write $\mathsf{Ch}_{\mathcal{D}}(s)$=$(c + X)q_1(s)$. The verification equation can then be re-stated as:

$$\begin{aligned} e(P_1, P_2) &= e(C, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e(\gamma P_1, w_0) \cdot e(w_1, (c+s)q_1(s)P_2) \\ &= e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2)^{(c+s)} \end{aligned}$$

Hence, we have $e(P_1, P_2)^{\frac{1}{(c+s)}} = e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2)$, where the right side can be efficiently computed by $\mathcal{A}$. Therefore, $\mathcal{B}$ can output a solution $(c, e(w_1 \cdot \gamma P_1, w_0 \cdot q_2(s)P_2))$ to the q-co-GSDH problem.

(2.4) $s \notin \mathcal{X}\ \wedge\ s \in \mathcal{D}$: In this case we have that $C = \mathsf{Ch}_{\mathcal{D}}(s) \cdot rP_1$ with $\mathsf{Ch}_{\mathcal{D}}(s) = (c + X)q_1(s)$, for some $s \neq c \in \{\mathcal{X} \cap \mathcal{D}\}$. Again, c and q_1 can be efficiently computed and the verification equation can then re-stated as:

$$\begin{aligned} e(P_1, P_2) &= e(\mathsf{Ch}_{\mathcal{X}}(s) \cdot rP_1, w_0) \cdot e(w_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ &= e((c+s)q_1(s) \cdot rP_1, w_0) \cdot e(w_1, P_2^0) \\ &= e(q_1(s) \cdot rP_1 \cdot w_1, w_0)^{(c+s)} \end{aligned}$$

Hence, we have $e(P_1, P_2)^{\frac{1}{(c+s)}} = e(q_1(s) \cdot rP_1 \cdot w_1, w_0)$, where the right side can be efficiently computed by $\mathcal{A}$. Therefore, $\mathcal{B}$ can output a solution $(c, e(q_1(s) \cdot rP_1 \cdot w_1, w_0))$ to the q-co-GSDH problem. □

Security of the Proof of Exponentiation. We now discuss the integration and security of the PoE, considering it as an optional functionality, with our SCDS construction. We do this since it differs from the security analysis of the PoE given in [Tha19], which does not consider its use in the same context.

First, we observe that the protocols and propositions from [Tha19] consider a general pairing e and a general polynomial $f(X) \in \mathbb{F}_p[X]$. However, the protocol PoE used in this work relies on other restrictions; therefore, its security is not based on the same propositions. In our case, the pairing function e being used is of Type-III and α is chosen by the verifier, meaning that β is also determined by α. Below we argue the security of our PoE protocol considering the use of the PoE in the VerifySS algorithm of our scheme (a similar reasoning also applies to VerifyDS).

The adversary $\mathcal{A}$ is given α (chosen by the verifier at random) and has to produce witnesses w_1, w_2 and w_3 for a set $\mathcal{S}$, with $\mathsf{Ch}_{\mathcal{S}}(X) = (X + \alpha)q_{\mathcal{S}}(X) + \beta$ and s.t the following holds:

$$(e((s+\alpha)P_1, w_1) + e(\beta P_1, P_2) = e(P_1, w_2)) \wedge (e(w_3, w_2) = e(C, P_2))$$

Hence, the w_2 used to verify the first pairing equation is also used to verify the second pairing equation, adding a restriction on the witness required in the corresponding proposition from [Tha19] (in which the knowledge of exponent assumption [BP04] is required).

We can assume w.l.o.g that w_2 is of the form $\mathsf{Ch}_{\mathcal{D}}(s)P_2$ for some set $\mathcal{D} \subseteq \mathcal{X}$, where $\mathcal{X}$ is the accumulated set by C. Otherwise, even if the first pairing equation is verified, the second will fail if the q-co-GSDH assumption holds.

With the above in mind, we study the existence of $\mathsf{Ch}_{\mathcal{D}}(X)$ that verifies the first pairing equation, given that $(X + \alpha)$ and β are fixed for the adversary.

We have three cases: (1) $\mathcal{S} \subseteq \mathcal{D}$, (2) $\mathcal{D} \subset \mathcal{S}$ and (3) $\mathcal{D} \cap \mathcal{S} = \emptyset$.

(1) If $\mathcal{S} \subseteq \mathcal{D}$ and the adversary succeeds in producing the witnesses that pass both verifications, a proof for $\mathcal{D} \subseteq \mathcal{X}$ would also work as a proof for $\mathcal{S} \subseteq \mathcal{X}$ (the adversary is doing extra computations that are not necessary, and such case is not considered as an attack).

(2) $\mathcal{D} \subset \mathcal{S}$: We assume that α and $\alpha + 1$ do not belong to $\mathcal{S}$. We have that:

$$\mathsf{Ch}_{\mathcal{S}}(X) = (X+\alpha)q_{\mathsf{Ch}_{\mathcal{S}}}(X) + \beta \quad (5.1)$$

$$\mathsf{Ch}_{\mathcal{D}}(X) = (X+\alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X) + \beta \quad (5.2)$$

We deduce that $\beta = \mathsf{Ch}_{\mathcal{D}}(X) - (X+\alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X)$ then we obtain

$$\mathsf{Ch}_{\mathcal{S}}(X) = (X+\alpha)q_{\mathsf{Ch}_{\mathcal{S}}}(X) + \mathsf{Ch}_{\mathcal{D}}(X) - (X+\alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X)$$

Moreover, we have $\mathsf{Ch}_{\mathcal{S}}(X) = \mathsf{Ch}_{\mathcal{D}}(X)Q(X)$ and so we get that

$$\mathsf{Ch}_{\mathcal{D}}(X)(Q(X) - 1) = (X + \alpha)(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X))$$

Since $\alpha \notin \mathsf{Ch}_{\mathcal{D}}(X)$, the terms $(X+\alpha)$ and $(q_{\mathsf{Ch}_{\mathcal{S}}}(X)-q_{\mathsf{Ch}_{\mathcal{D}}}(X))$ have to divide $Q(X) - 1$ and $\mathsf{Ch}_{\mathcal{D}}(X)$ respectively. Therefore, we have:

$$\begin{aligned}(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X)) &= \mathsf{Ch}_{\mathcal{D}}(X)B \\ Q(X) - 1 &= (X + \alpha)B\end{aligned}$$

From the first equation we get that $deg(q_{\mathsf{Ch}_{\mathcal{S}}}(X) - q_{\mathsf{Ch}_{\mathcal{D}}}(X)) \leq deg(q_{\mathsf{Ch}_{\mathcal{S}}}(X)) = deg(\mathsf{Ch}_{\mathcal{S}}(X)) - 1$ (which follows from (1)). This means that $deg(\mathsf{Ch}_{\mathcal{D}}(X)) \leq deg(\mathsf{Ch}_{\mathcal{S}}(X)) - 1$. Looking at the second equation, we recall that $\mathsf{Ch}_{\mathcal{S}}(X) = \mathsf{Ch}_{\mathcal{D}}(X)Q(X)$. Since $\mathsf{Ch}_{\mathcal{S}}(X)$ and $\mathsf{Ch}_{\mathcal{D}}(X)$ are irreducible polynomials we can deduce that $B = 1$. Hence $Q(X) = (X + (\alpha + 1))$ and $(X + (\alpha + 1))$ is a factor of $\mathsf{Ch}_{\mathcal{S}}(X)$, which contradicts the assumption that $\alpha + 1 \notin \mathcal{S}$. We conclude that no such set $\mathcal{D}$ exists.

(3) $\mathcal{D} \cap \mathcal{S} = \emptyset$: In this case the adversary needs to produce a subset $\mathcal{D} \subseteq \mathcal{X}$ for which the following holds: $\mathsf{Ch}_{\mathcal{D}}(X) = (X + \alpha)q_{\mathsf{Ch}_{\mathcal{D}}}(X) + \beta$. In such case, looking at the first pairing equation we have that:

$$\begin{aligned}e((s + \alpha)P_1, w_1) + e(\beta P_1, P_2) &= e(P_1, \mathsf{Ch}_{\mathcal{D}}(s)P_2) \\ e(P_1, (s + \alpha)w_1) &= e(P_1, (\mathsf{Ch}_{\mathcal{D}}(s) - \beta)P_2)\end{aligned}$$

which means that $(s + \alpha)w_1 = (\mathsf{Ch}_{\mathcal{D}}(s) - \beta)P_2$. We show that if such a w_1 exists then we can build and adversary that breaks the q-co-SDH assumption. Therefore, we assume that $f(X) = \mathsf{Ch}_{\mathcal{D}}(X) - \beta$ does not divide $(X + \alpha)$. Since $f(X)$ and $(X + \alpha)$ are relatively prime we can compute polynomials $h_1(X), h_2(X)$ such that

$$f(X)h_1(X) + (X + \alpha)h_2(X) = 1$$

Set $w_1^* := h_1(s)w_1 + h_2(s)P_2$. Then $(s+\alpha)w_1^* = P_2$ and we have a pair (α, w_1^*) which breaks the q-co-SDH in $\mathbb{G}_2$.

Expressiveness of our Set-commitment Scheme. Before concluding this section, we elaborate on the expressiveness gained by adding a set-commitment supporting proofs of disjoint sets (*i.e.,* NAND showings).

First of all, scenarios considering access control policies can benefit of NAND showings as they allow users, for instance, to prove that they do not belong to a particular business unit within a company. Another example includes applications providing discounts to tourists of a particular region. Suppose an airline is offering discounts on flight tickets for regional flights inside Spain but only for citizens from outside the Schengen area. A user could use a NAND showing to prove that her country of origin does not belong to the Schengen area without disclosing the country. Similarly, users can prove that they are not residents of a particular country (*e.g.,* considering Spain, computing a NAND proof for the attribute {"residence, Spain"}). None of the previous statements could easily be done with the ABC from [FHS19] unless their negation was harcoded in the credential.

The following example from [TG20] (Section 6.2) illustrates how NAND showings can also be used to perform interval proofs. Suppose that a users want to prove that they are at least 18 year old. Assuming the current date is 2 January 2020 and the user's birthday is on 1 January 2002, we can have two redundant attributes {"byear = 2002"}, {"bmth = Jan2002"} for {"bday = 01Jan2002"} in the user's credential so that the verifier can ask for a NAND showing on the attribute set

$$\mathcal{D}=\{\{\text{“byear = 2020”}\}, \ldots, \{\text{“byear = 2003”}\}, \{\text{“bmth = Feb2002”}\}, \ldots, \{\text{“bmth = Dec2002”}\}, \{\text{“bday = 02Jan2002”}\}, \ldots, \{\text{“bday = 31Jan2002”}\}\}$$

Considering the work in [FHS19], such a proof could only be done encoding predefined statements like {"adult, >18"}, making things more complex to handle.

5.5 Extending the ABC framework of [FHS19]

To build an ABC scheme, [FHS19] relies on a set-commitment scheme and a SPS-EQ. A set-commitment represents a set of attributes, while a SPS-EQ signs a vector of group elements (among which the set-commitment is included). This way, a credential is formed by the message-signature part and auxiliary information to use it during showings. To present a credential, users randomise the message and signature pair to obtain a different representative for the same equivalence class. Subsequently, the opening witness for the set-commitment scheme is randomised consistently and presented alongside the credential. Finally, verifiers individually check the signature and witness for the set of attributes in question to accept or reject a user credential.

In terms of security, [FHS19] introduces a game-based security model for ABCs in the vein of Bellare, Shi and Zhang's model for group signatures [BSZ05]. Moreover, their model considers replay attacks and provides a strong form of anonymity against organisations that may maliciously generate keys. For a scheme to be secure in their model, the authors require it to be correct, unforgeable and anonymous. Correctness follows the usual notions. Unforgeability safeguards the verifier against potentially malicious users that may try to present a credential for attributes not issued by the corresponding authority. Finally, anonymity safeguards the user against potentially malicious organisations and verifiers that may collude to learn more information than the one intended to be presented by a user.

The purpose of this section is to present a new ABC extending that from [FHS19] to consider NAND showing proofs, the use of a CRS, and issuer-hiding features. On the one hand, a NAND showing proof allows users to demonstrate that a given set of attributes is not present in their credentials. On the other hand, using a CRS allows us to instantiate the framework with our mercurial signature scheme from

Chapter 4, obtaining a construction under standard assumptions with issuer-hiding features. The core differences in this extended ABC model follow naturally from (1) the addition of disjoint sets in the SCDS scheme in section 5.4.1, and (2) the removal of the key verification algorithm (as we work with a CRS).

5.5.1 ABC Syntax

An ABC scheme consists of the following p.p.t algorithms:

$\mathsf{Setup}(1^\lambda, 1^q)$ takes a security parameter λ, an upper bound q for the size of attribute sets, and outputs public parameters pp discarding any trapdoor.

$\mathsf{TSetup}(1^\lambda, 1^q)$ is like Setup but it also returns a trapdoor τ (if any).

$\mathsf{OrgKeyGen}(\mathsf{pp})$ takes pp as input and outputs an organisation key pair $(\mathsf{osk}, \mathsf{opk})$.

$\mathsf{UserKeyGen}(\mathsf{pp})$ takes pp as input and outputs a user key pair $(\mathsf{usk}, \mathsf{upk})$.

$\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X})$ and $\mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})$ are run by a user and the organisation respectively, who interact during execution. Obtain takes as input pp, the user's secret key usk, an organisations public key opk, and an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < q$. Issue takes as input pp, a user public key upk, the organisation's secret key osk, and an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < q$. At the end of this protocol, Obtain outputs a credential cred on $\mathcal{X}$ for the user or $\perp$ if the execution failed.

$\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathcal{X}, \mathcal{S}, \mathcal{D}, \mathsf{cred})$ and $\mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})$ are run by a user and a verifier respectively, who interact during execution. Show takes as input pp, an organisation public key opk, a credential cred for the attribute set $\mathcal{X}$, potentially non-empty sets $\mathcal{S} \subseteq \mathcal{X}$, $\mathcal{D} \nsubseteq \mathcal{X}$ representing attributes sets being a subset ($\mathcal{S}$) or disjoint ($\mathcal{D}$) to the attribute set ($\mathcal{X}$) committed in the credential. Verify takes as input pp, an organisation public key opk, the sets $\mathcal{S}$ and $\mathcal{D}$. At the end, Verify outputs 1 or 0 indicating whether or not the credential showing was accepted.

5.5.2 Security Properties

The following notions are based on the security model from [FHS19] (Section 5.1), which we adapt to consider the use of a crs (pp) and NAND showing proofs. Informally, an ABC scheme is secure if it has the following properties:

Correctness. A showing of a credential with respect to a non-empty sets $\mathcal{S}$ and $\mathcal{D}$ of attributes always verify if the credential was issued honestly on some attribute set $\mathcal{X}$ with $\mathcal{S} \subset \mathcal{X}$ and $\mathcal{D} \nsubseteq \mathcal{X}$.

Unforgeablility. Given at least one non-empty set $\mathcal{S} \subset \mathcal{X}$ or $\mathcal{D} \nsubseteq \mathcal{X}$, a user in possession of a credential for the attribute set $\mathcal{X}$ cannot perform a valid showing for $\mathcal{D} \subset \mathcal{X}$ nor for $\mathcal{S} \nsubseteq \mathcal{X}$. Moreover, no coalition of malicious users can combine their credentials and prove possession of a set of attributes which no single member has. This holds even after seeing showings of arbitrary credentials by honest users (thus, covering replay attacks).

Anonymity. During a showing, no verifier and no (malicious) organisation (even if they collude) is able to identify the user or learn anything about the user, except that she owns a valid credential for the shown attributes. Furthermore, different showings of the same credential are unlinkable.

In addition, we will also consider an issuer-hiding notion as introduced in [CLPK22] (under the name of signer-hiding). Informally speaking, it allows users to hide the identity of their issuer within a set of issuers. The following global variables and oracles are listed to introduce the corresponding formal definitions.

Global variables. At the beginning of each experiment, either the experiment computes an organisation key pair $(\mathsf{osk}, \mathsf{opk})$ or the adversary outputs opk. In the anonymity game there is a bit b, which the adversary must guess.

In order to keep track of all honest and corrupt users, we introduce the sets $\mathtt{HU}$, and $\mathtt{CU}$, respectively. We use the lists $\mathtt{UPK}$, $\mathtt{USK}$, $\mathtt{CRED}$, $\mathtt{ATTR}$ and $\mathtt{OWNR}$ to track user public and secret keys, issued credentials and corresponding attributes and to which user they were issued. Furthermore, we use the sets J_{LoR} and I_{LoR} to store which issuance indices and corresponding users have been set during the first call to the left-or-right oracle in the anonymity game.

Oracles. Considering an adversary $\mathcal{A}$ the oracles are as follows:

$\mathcal{O}_{\mathtt{HU}}(i)$ takes as input a user identity i. If $i \in \mathtt{HU} \cup \mathtt{CU}$, it returns $\perp$. Otherwise, it creates a new honest user i by running $(\mathtt{USK}[i], \mathtt{UPK}[i]) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{opk})$, adding i to the honest user list $\mathtt{HU}$ and returning $\mathtt{UPK}[i]$.

$\mathcal{O}_{\mathtt{CU}}(i, \mathsf{upk})$ takes as input a user identity i and (optionally) a user public key upk; if user i does not exist, a new corrupt user with public key upk is registered, while if i is honest, its secret key and all credentials are leaked. In particular, if $i \in \mathtt{CU}$ or if $i \in I_{\mathsf{LoR}}$ (that is, i is a challenge user in the anonymity game) then the oracle returns $\perp$. If $i \in \mathtt{HU}$, then the oracle removes i from $\mathtt{HU}$ and adds it to $\mathtt{CU}$; it returns $\mathtt{USK}[i]$ and $\mathtt{CRED}[j]$ for all j with $\mathtt{OWNR}[j] = i$. Otherwise (*i.e.*, $i \notin \mathtt{HU} \cup \mathtt{CU}$), it adds i to $\mathtt{CU}$ and sets $\mathtt{UPK}[i] \leftarrow \mathsf{upk}$.

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{X})$ takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \mathtt{HU}$, it returns $\perp$. Otherwise, it issues a credential to i by running

$$(\mathsf{cred}, \top) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \mathtt{USK}[i], \mathsf{opk}, \mathcal{X}), \mathsf{Issue}(\mathsf{pp}, \mathtt{UPK}[i], \mathsf{osk}, \mathcal{X}).$$

If $\mathsf{cred} = \perp$, it returns $\perp$. Else, it appends $(i, \mathsf{cred}, \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious organisation, issue a credential to an honest user. It takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \mathtt{HU}$, it returns $\perp$. Otherwise, it runs

$$(\mathsf{cred}, \cdot) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \mathtt{USK}[i], \mathsf{opk}, \mathcal{X}), \cdot),$$

where the Issue part is executed by $\mathcal{A}$. If $\mathsf{cred} = \perp$, it returns $\perp$. Else, it appends $(i, \mathsf{cred}, \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious user, obtain a credential from an honest organisation. It takes as input a user identity i and a set of attributes $\mathcal{X}$. If $i \notin \mathtt{CU}$, it returns $\perp$. Otherwise, it runs

$$(\cdot, I) \xleftarrow{\$} (\cdot, \mathsf{Issue}(\mathsf{pp}, \mathtt{UPK}[i], \mathsf{osk}, \mathcal{X})),$$

where the Obtain part is executed by $\mathcal{A}$. If I $= \perp$, it returns $\perp$. Else, it appends $(i, \perp, \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Show}}(j, \mathcal{S}, \mathcal{D})$ lets the adversary $\mathcal{A}$ play a dishonest verifier during a showing by an honest user. It takes as input an index of an issuance j and attributes sets $\mathcal{S}$ and $\mathcal{D}$. Let $i \xleftarrow{\$} \mathtt{OWNR}[j]$. If $i \notin \mathtt{HU}$, it returns $\perp$. Otherwise, it runs

$$(S, \cdot) \xleftarrow{\$} \mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathtt{ATTR}[j], \mathcal{S}, \mathcal{D}, \mathtt{CRED}[j]), \cdot)$$

where the Verify part is executed by $\mathcal{A}$.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S}, \mathcal{D})$ is the challenge oracle in the anonymity game where $\mathcal{A}$ must distinguish (multiple) showings of two credentials $\mathtt{CRED}[j_0]$ and $\mathtt{CRED}[j_1]$. The oracle takes two issuance indices j_0 and j_1 and attribute sets $\mathcal{S}$ and $\mathcal{D}$. If $J_{\mathsf{LoR}} \neq \emptyset$ and $J_{\mathsf{LoR}} \neq \{j_0, j_1\}$, it returns $\perp$. Let $i_0 \xleftarrow{\$} \mathtt{OWNR}[j_0]$ and $i_1 \xleftarrow{\$} \mathtt{OWNR}[j_1]$. If $J_{\mathsf{LoR}} \neq \emptyset$ then it sets $J_{\mathsf{LoR}} \xleftarrow{\$} \{j_0, j_1\}$ and $I_{\mathsf{LoR}} \xleftarrow{\$} \{i_0, i_1\}$. If $i_0, i_1 \neq \mathtt{HU} \vee \mathcal{S} \nsubseteq \mathtt{ATTR}[j_0] \cap \mathtt{ATTR}[j_1] \vee \mathcal{D} \cap \{\mathtt{ATTR}[j_0] \cup \mathtt{ATTR}[j_1]\} \neq \emptyset$, it returns $\perp$. Else, it runs

$$(S, \cdot) \xleftarrow{\$} (\mathsf{Show}(\mathsf{opk}, \mathtt{ATTR}[j_b], \mathcal{S}, \mathcal{D}, \mathtt{CRED}[j_b]), \cdot),$$

(with b set by the experiment) where the Verify part is executed by $\mathcal{A}$.

Correctness of ABC schemes requires that for all $\lambda > 0$, all $t > 0$, all $\mathcal{X}$ with $0 < |\mathcal{X}| \leq t$ and all $\emptyset \neq \mathcal{S} \subset \mathcal{X}$ and $\emptyset \neq \mathcal{D} \nsubseteq \mathcal{X}$ with $0 < |\mathcal{D}| \leq t$:

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{pp}); \\ (\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X}), \\ \mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})) \end{array} : \begin{array}{l} (\top, 1) \xleftarrow{\$} (\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathcal{X}, \mathcal{S}, \\ \mathcal{D}, \mathsf{cred}), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})) \end{array}\right] = 1$$

Definition 36: Unforgeability

An ABC scheme is unforgeable, if for all $\lambda > 0$, all $q > 0$ and p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathtt{CU}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}\}$ the following probability is negligible.

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp}); \\ (\mathcal{S}, \mathcal{D}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{opk}); \\ (\cdot, b^*) \xleftarrow{\$} (\mathcal{A}(\mathsf{st}), \mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{S}, \mathcal{D})) \end{array} : \begin{array}{l} b^* = 1 \wedge \\ \forall\, j : \mathtt{OWNR}[j] \in \mathtt{CU} \implies \\ \mathcal{S} \notin \mathtt{ATTR}[j] \vee \mathcal{D} \cap \mathtt{ATTR}[j] \neq \emptyset \end{array}\right]$$

Definition 37: Anonymity

An ABC scheme is anonymous, if for all $\lambda > 0$, all $q > 0$ and all p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathtt{CU}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{LoR}}\}$ the following probability is negligible.

$$\Pr\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); b \xleftarrow{\$} \{0,1\}; (\mathsf{opk}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}); \\ b^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{st})\end{array} : b^* = b\right] - \tfrac{1}{2}$$

Definition 38: Issuer-Hiding

An ABC scheme supports issuer-hiding if for all $\lambda > 0$, all $q > 0$, all $n > 0$, all $t > 0$, all $\mathcal{X}$ with $0 < |\mathcal{X}| \leq t$, all $\emptyset \neq \mathcal{S} \subset \mathcal{X}$ and $\emptyset \neq \mathcal{D} \not\subseteq \mathcal{X}$ with $0 < |\mathcal{D}| \leq t$, and p.p.t adversaries $\mathcal{A}$, the following holds.

$$\Pr\left[\begin{array}{l}\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, 1^q); \\ \forall\, i \in [n] : (\mathsf{osk}_i, \mathsf{opk}_i) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{pp}); j \xleftarrow{\$} [n]; \\ (\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{usk}, \mathsf{opk}_j, \mathcal{X}), \mathsf{Issue}(\mathsf{upk}, \mathsf{osk}_j, \mathcal{X})); \\ j^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Show}}}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, (\mathsf{opk}_i)_{i\in[n]})\end{array} : j^* = j\right] \leq \frac{1}{n}$$

5.6 ABC From Standard Assumptions

As previously explained, our ABC scheme is based on the one from [FHS19]. The main changes are the following:

1. Instead of instantiating the ABC with the signature from [FHS19] (secure in the GGM), we use our mercurial signature construction from Chapter 4.
2. As we use a signature scheme that relies on a CRS, we move the parameters of the set-commitment scheme from the organisation's key pair to the public parameters pp that include the previous CRS. Furthermore, we instantiate the ZKPoK's using Pedersen commitments and the construction from [Dam00], as suggested in [FHS19] (Remark 1).
3. Our showing protocol can be instantiated with two sets, $\mathcal{S}$ and $\mathcal{D}$, one to compute AND proofs (selective disclosure) and one to compute NAND proofs.
4. We integrate the proof of exponentiation into the showing protocol.
5. We build a NIZK argument that, alongside our signature scheme, allows us to achieve the issuer-hiding property.

Intuition. We begin explaining the difference to [FHS19] with respect to malicious organisations as it clarifies the changes introduced in the issuing protocol. We recall that in this context, the term *malicious organisations* refer to organisations whose key-pairs are generated in a way that trapdoor information is included. Such trapdoor information could later be used by an organisation to break anonymity, provided that extra information (a transcript of a given showing protocol containing a credential issued by the organisation) is available. The ABC scheme from [FHS19] defines a ZKPoK in the issuing protocol ($\Pi^{\mathcal{R}_O}$) for which the organisation needs to prove knowledge of the corresponding secret key to avoid the previous scenario. Since the keys in our SPS-EQ (Figure 4.4 from Chapter 4) need to be generated using the CRS (which includes the matrix $\mathbf{A}$), we do not need to request a ZKPoK from the organisation in the issuing protocol as the signature's verification

```
$\underline{\mathsf{Setup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds_{pp}}) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$
$(\mathsf{sps_{pp}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PGen}(1^\lambda; \mathsf{BG})$
$r \xleftarrow{\$} \mathbb{Z}_p^*; \mathsf{ck} \leftarrow (P_1, rP_1)$
$\mathbf{return}\ (\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{sps_{pp}}, \mathsf{ck})$

$\underline{\mathsf{OKGen}(\mathsf{pp})}$:
$\mathbf{return}\ \mathsf{SPS\text{-}EQ.KGen}(\mathsf{BG}, \mathsf{sps_{pp}}, 3)$
```

```
$\underline{\mathsf{TSetup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{scds}_\tau) \xleftarrow{\$} \mathsf{SCDS.TSetup}(1^\lambda, q)$
$(\mathsf{sps_{pp}}, \mathsf{sps}_\tau) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PTGen}(1^\lambda; \mathsf{BG})$
$r \xleftarrow{\$} \mathbb{Z}_p^*; \mathsf{ck} \leftarrow (P_1, rP_1)$
$\mathbf{return}\ (\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{sps_{pp}}, \mathsf{ck}, \mathsf{scds}_\tau, \mathsf{sps}_\tau, r)$

$\underline{\mathsf{UKGen}(\mathsf{pp})}$:
$\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*; \mathsf{upk} \leftarrow \mathsf{usk}P_1; \mathbf{return}\ (\mathsf{usk}, \mathsf{upk})$
```

Figure 5.2: BABC: Setup and key generation algorithms.

algorithm a pairing involving the matrix $\mathbf{A}$ and the organisation's public key $\mathsf{opk} = (\mathbf{B}, \mathbf{C})$ is used to check the signature. Hence, a signature that verifies rules out that 1) someone impersonated the issuer signing with a different secret key, and 2) that the public key was maliciously generated. Regarding the showing protocol, the only changes are the addition of NAND and exponentiation proofs. For the latter, we require the verifier to randomly pick the challenge and send it to the user.

```
$\underline{\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathcal{X})}$                              $\underline{\mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathcal{X})}$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*; a \leftarrow r_1 P_1$
$c \leftarrow \mathsf{Commit}(a, r_2)$
                                        $\xrightarrow{c}$
                                                    $e \xleftarrow{\$} \mathbb{Z}_p^*$
                                        $\xleftarrow{e}$
$z \leftarrow r_1 + e \cdot \mathsf{usk}$
$(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathcal{X}; \mathsf{usk})$
$r_3 \xleftarrow{\$} \mathbb{Z}_p^*; R \leftarrow r_3 C$
                                        $\xrightarrow{C, R, z, a, r_2}$
                                                    $\mathbf{if}\ (zP_1 \neq a + e \cdot \mathsf{upk} \vee$
                                                        $c \neq \mathsf{Commit}(a, r_2))\ \mathbf{return}\ \perp$
                                                    $\mathbf{if}\ (e(C, P_2) \neq e(\mathsf{upk}, \mathsf{Ch}_\mathcal{X}(s)P_2)$
                                                        $\wedge\ \forall\ x \in \mathcal{X} : xP_1 \neq \mathsf{ek}_1^0)\ \mathbf{return}\ \perp$
                                                    $(\sigma, \tau) \leftarrow \mathsf{SPS\text{-}EQ.Sign}((C, R, P_1), \mathsf{osk})$
                                        $\xleftarrow{(\sigma, \tau)}$
$\mathbf{check}\ \mathsf{SPS\text{-}EQ.Verify}((C, R, P_1), (\sigma, \tau), \mathsf{opk})$
$\mathbf{return}\ \mathsf{cred} = (C, (\sigma, \tau), r_3, O)$
```

Figure 5.3: BABC: Obtain and issue algorithms.

For ease of exposition, we first present a construction whose Show and Verify algorithms only consider selective disclosures of attributes, including the proof of exponentiation (PoE). For this construction, which we call BABC (as it is our basic ABC), we highlight the changes with respect to the original ABC scheme from [FHS19]. Setup and key generation algorithms are given in Figure 5.2, Obtain

```
Show(pp, usk, opk, $\mathcal{S}$, cred)                              Verify(pp, opk, $\mathcal{S}$)
$(C, (\sigma, \tau), r, O) \leftarrow$ cred; $\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$
if $O = (1, (o_1, o_2))$ then   $O' = (1, (\mu \cdot o_1, o_2))$
else $O' = \mu \cdot O$
$\sigma' \xleftarrow{\$}$ SPS-EQ.ChgRep$((C, rC, P_1), \sigma, \tau, \mu, \rho, \mathsf{opk})$
$(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$
cred$' \leftarrow (C_1, C_2, C_3, \sigma')$
wss $\leftarrow$ SCDS.OpenSS$(\mu C, \mathcal{S}, O')$
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$
$c_1 \leftarrow$ Commit$(a_1, r_2)$
$c_2 \leftarrow$ Commit$(a_2, r_4)$
$\Sigma_1 = ($cred$'$, wss, $c_1, c_2)$
                                  $\xrightarrow{\Sigma_1}$
                                                                     $($cred$'$, wss, $c_1, c_2) \leftarrow \Sigma_1$
                                                                     $e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
                                  $\xleftarrow{e, \tilde{e}}$
$\pi_1 \leftarrow$ SCDS.PoE$(\mathcal{S}, \tilde{e})$                                 $(C_1, C_2, C_3, \sigma) \leftarrow$ cred$'$
$z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$
$\Sigma_2 = ((z_i, a_i)_{i \in \{1,2\}}, \pi_1, r_2, r_4)$
                                  $\xrightarrow{\Sigma_2}$
                                                                     $((z_i, a_i)_{i \in \{1,2\}}, \pi_1, r_2, r_4) \leftarrow \Sigma_2$
                                                                     check
                                                                       $z_1 C_1 = a_1 + e C_2$
                                                                       $z_2 P_1 = a_2 + e C_3$
                                                                       $c_1 =$ Commit$(a_1, r_2)$
                                                                       $c_2 =$ Commit$(a_2, r_4)$
                                                                       SPS-EQ.Verify(cred$'$, opk)
                                                                       SCDS.VerifySS$(C_1, \mathcal{S}$, wss; $\pi_1, \tilde{e})$
```

Figure 5.4: BABC: Show and verify.

and Issue in Figure 5.3, and Show and Verify in Figure 5.4.

Subsequently, we present a second scheme, which is exactly as BABC but with Show and Verify algorithms supporting NAND proofs. For this reason, we call it NABC. In Figure 5.5, we give the Show and Verify algorithms for NABC, highlighting the differences with BABC. We observe that if a NAND proof is used, it increases bandwidth by 4 elements (two from $\mathbb{G}_1$ and two from $\mathbb{G}_2$), as the PoE can reuse the same challenge.

5.6.1 Issuer-Hiding Strategies

In the following, we present the issuer-hiding strategy introduced in [CLPK22]. In addition, independent and concurrent work by Bobolz *et al.* [BEK+21] also addressed the problem of hiding the identity of a credential issuer/signer (coining the term *issuer-hiding*). There, the authors propose a slightly different setting

```
Show(pp, usk, opk, $\mathcal{S}$, $\mathcal{D}$, cred)                    Verify(pp, $\mathcal{S}$, $\mathcal{D}$)
$(C, (\sigma, \tau), r, O) \leftarrow$ cred; $\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$
if $O = (1, (o_1, o_2))$ then   $O' = (1, (\mu \cdot o_1, o_2))$
else $O' = \mu \cdot O$
$\sigma' \xleftarrow{\$}$ SPS-EQ.ChgRep$((C, rC, P_1), \sigma, \tau, \mu, \rho,$ opk$)$
$(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$
cred$' \leftarrow (C_1, C_2, C_3, \sigma')$
wss $\leftarrow$ SCDS.OpenSS$(\mu C, \mathcal{S}, O')$
wds $\leftarrow$ SCDS.OpenDS$(\mu C, \mathcal{D}, O')$
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$
$c_1 \leftarrow$ Commit$(a_1, r_2)$
$c_2 \leftarrow$ Commit$(a_2, r_4)$
$\Sigma_1 = ($cred$'$, wss, wds, $c_1, c_2)$
                                        $\xrightarrow{\Sigma_1}$
                                                  $($cred$'$, wss, wds, $c_1, c_2) \leftarrow \Sigma_1$
                                                  $e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
                                        $\xleftarrow{e, \tilde{e}}$
$\pi_1 \leftarrow$ SCDS.PoE$(\mathcal{S}, \tilde{e})$                        $(C_1, C_2, C_3, \sigma) \leftarrow$ cred$'$
$\pi_2 \leftarrow$ SCDS.PoE$(\mathcal{D}, \tilde{e})$
$z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$
$\Sigma_2 = ((z_i, a_i, \pi_i)_{i \in \{1,2\}}, r_2, r_4)$
                                        $\xrightarrow{\Sigma_2}$
                                                  $((z_i, a_i, \pi_i)_{i \in \{1,2\}}, r_2, r_4) \leftarrow \Sigma_2$
                                                  check
                                                    $z_1 C_1 = a_1 + e C_2$
                                                    $z_2 P_1 = a_2 + e C_3$
                                                    $c_1 =$ Commit$(a_1, r_2)$
                                                    $c_2 =$ Commit$(a_2, r_4)$
                                                    SPS-EQ.Verify(cred$'$, opk$'$)
                                                    SCDS.VerifySS$(C_1, \mathcal{S}$, wss; $\pi_1, \tilde{e})$
                                                    SCDS.VerifyDS$(C_1, \mathcal{S}$, wds; $\pi_2, \tilde{e})$
```

Figure 5.5: NABC: Show and verify.

than the one from [CLPK22] to avoid using an OR-like proof as done in this work. In brief, the authors consider access policies of the form $\{\sigma_i, \mathsf{pk}_i\}_{i \in [n]}$, where σ_i is a signature on a given authority's public key pk_i produced by the verifier. As a result, users can prove the correspondence between a public key (defined in the policy) and the credential verification under that public key in zero knowledge, using a NIZK independent of the number of public keys defined in the policy. Finally, we note that our work is compatible with their formalization, deferring that discussion to Chapter 6.

When using the mercurial signature construction from Chapter 4, users can consistently randomise the signature on their credential and the issuer's public key. Therefore, a fully adaptive NIZK argument can be used to prove that a randomised issuer key belongs to the equivalence class of one of the keys contained in a list of issuers' keys. This way, the randomised issuer key can be used to verify the credential while hiding the issuer's identity (like in a ring signature). More in detail, as in the construction form Chapter 4, we have $k = 1$ and $\ell = 3$, the public keys consist of two vectors $[\mathbf{B}]_2 \in (\mathbb{G}_2^*)^2$ and $[\mathbf{C}]_2 \in (\mathbb{G}_2^*)^3$, where the secret keys have the form $\mathsf{sk} = (\mathbf{K}_0, \mathbf{K})$ with $\mathbf{K}_0 \xleftarrow{\$} (\mathbb{Z}_p^*)^{2\times 2}$ and $\mathbf{K} \xleftarrow{\$} (\mathbb{Z}_p^*)^{3\times 2}$. With this in mind, we can naturally define equivalence relations on the key spaces $\mathcal{S}_{\mathsf{sk}} = \{(\mathbb{Z}_p^*)^{2\times 2} \times (\mathbb{Z}_p^*)^{3\times 2}\}$ and $\mathcal{S}_{\mathsf{pk}} = \{(\mathbb{G}_2^*)^2 \times (\mathbb{G}_2^*)^3\}$ as follows:

$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}, \tilde{\mathsf{sk}}) \in \mathcal{S}_{\mathsf{sk}} \times \mathcal{S}_{\mathsf{sk}} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t } \tilde{\mathsf{sk}} = \rho \cdot \mathsf{sk}\}$$
$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \tilde{\mathsf{pk}}) \in \mathcal{S}_{\mathsf{pk}} \times \mathcal{S}_{\mathsf{pk}} \mid \exists\, \rho \in \mathbb{Z}_p^* \text{ s.t } \tilde{\mathsf{pk}} = \rho \cdot \mathsf{pk}\}$$

If we have a list of public keys $(\mathbf{B}_1, \mathbf{C}_1), ..., (\mathbf{B}_n, \mathbf{C}_n)$ and define the equivalence class of each public key as before $((\mathbf{B}_i', \mathbf{C}_i') = (\mathbf{B}_i, \mathbf{C}_i) \cdot \rho)$, we can efficiently prove that a given public key $(\mathbf{B}_i', \mathbf{C}_i')$ belongs to the equivalence class of one of the public keys $(\mathbf{B}_1, \mathbf{C}_1), ..., (\mathbf{B}_n, \mathbf{C}_n)$ for some $(\mathbf{B}_i, \mathbf{C}_i)$. The idea is to use a generalised version of the OR-Proof from [CH20] and build a generalised NIZK OR-Proof for the AND statements of the two components. The new language is defined as follows (in the credential construction we use $\ell = 3$):

$$\mathcal{L}_{\bigvee(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}} \quad = \quad \{(\mathbf{B}_i', \mathbf{C}_i') \in \mathbb{G}_2^{2\times\ell} \mid \exists\, \rho \in \mathbb{Z}_p^* : \vee\ (\mathbf{B}_i' = \mathbf{B}_i \cdot \rho \wedge \mathbf{C}_i' = \mathbf{C}_i \cdot \rho)_{i\in[n]}\}$$

The resulting NIZK argument is given in Figure 5.6. Next, we explain how the previous NIZK argument can be used to hide the identity of an issuer.

First, we need to consider a scenario in which n-authorities can issue credentials to different sets of users. As we are in the classical setting, we also assume that every user gets a credential from one of the n-authorities and that the organisation keys are certified and publicly available.

The verifier must check the signature using the corresponding public key when showing a credential. The idea is to use the NIZK argument so a user can randomise the public key and present this randomised key to the verifier. It will then check the NIZK to verify that the public key is valid (*i.e.,* it belongs to the equivalence class of one of the n-authorities). Therefore, signatures need to be adapted by users so that they can be verified with the randomised public key.

Theorem 21. The proof system given in Figure 5.6 is a fully adaptive NIZK argument for the language $\mathcal{L}_{\bigvee(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}}$.

Proof. The proof follows from Theorem 19 in [CH20]. The only difference is that we rely on the AND composition for sigma protocols to compile the sigma protocol from [CH20] using the same challenge for both proofs. □

Integration. As the NIZK argument is fully adaptive, users can choose the size of the anonymity set (*i.e.,* the set of public keys in the OR-Proof). We find this approach simpler than using delegatable credentials to achieve a similar result. Users do not need to interact with the organisations to compute the NIZK proof nor to adapt the signature. Moreover, there is no need to use pseudonyms for each

```
$\underline{\mathsf{SH.PGen}(1^\lambda)}$:
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $z \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{crs} \leftarrow (\mathsf{BG}, [z]_1)$; $\tau \leftarrow z$
return $(\mathsf{crs}, \tau)$
```

```
$\underline{\mathsf{SH.PPro}(\mathsf{crs}, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i), \rho)}$:
$// \ \mathbf{B}'_i = \mathbf{B}_i \cdot \rho \wedge \mathbf{C}'_i = \mathbf{C}_i \cdot \rho$
$s_1, s_2, z_1, \ldots, z_{n-1} \xleftarrow{\$} \mathbb{Z}_p$
$[z_n]_1 \leftarrow [z]_1 - \sum_{j=1}^{j=n-1}[z_j]_1$
$[\mathbf{a}_i^1]_2 \leftarrow s_1\mathbf{B}_i$; $[\mathbf{a}_i^2]_2 \leftarrow s_2\mathbf{C}_i$
$[d_i^1]_1 \leftarrow \rho[z_i]_1 + [s_1]_1$; $[d_i^2]_1 \leftarrow \rho[z_i]_1 + [s_2]_1$
for all $j \neq i \in [n]$ do
    $d_j^1, d_j^2 \xleftarrow{\$} \mathbb{Z}_p$
    $[\mathbf{a}_j^1]_2 \leftarrow d_j^1\mathbf{B}_j\text{-}z_j\mathbf{B}'_i$
    $[\mathbf{a}_j^2]_2 \leftarrow d_j^2\mathbf{C}_j\text{-}z_j\mathbf{C}'_i$
return $(([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]})$
```

```
$\underline{\mathsf{SH.PSim}(\mathsf{crs}, \tau, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i))}$:
$z_1, \ldots, z_{n-1} \xleftarrow{\$} \mathbb{Z}_p$
$[z_n]_1 \leftarrow [\tau]_1 - \sum_{j=1}^{j=n-1}[z_j]_1$
for all i $\in [n]$ do
    $d_i^1, d_i^2 \xleftarrow{\$} \mathbb{Z}_p$
    $[\mathbf{a}_i^1]_2 \leftarrow d_i^1\mathbf{B}_i\text{-}z_i\mathbf{B}'_i$
    $[\mathbf{a}_i^2]_2 \leftarrow d_i^2\mathbf{C}_i\text{-}z_i\mathbf{C}'_i$
return $(([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]})$
```

```
$\underline{\mathsf{SH.PVer}(\mathsf{crs}, (\mathbf{B}_i, \mathbf{C}_i)_{i\in[n]}, (\mathbf{B}'_i, \mathbf{C}'_i), \pi)}$:
$(([\mathbf{a}_n^k]_2, [d_n^k]_1)_{n\in[n]}^{k\in[2]}, ([z_j]_1)_{j\in[n-1]}) \leftarrow \pi$
$[z_n]_1 = [z]_1 - \sum_{j=1}^{j=n-1}[z_j]_1$
for all i $\in [n]$ check
    $e([d_i^1]_1, \mathbf{B}_i) = e([z_i]_1, \mathbf{B}'_i) + e([1]_1, [\mathbf{a}_i^1]_2)$
    $e([d_i^2]_1, \mathbf{C}_i) = e([z_i]_1, \mathbf{C}'_i) + e([1]_1, [\mathbf{a}_i^2]_2)$
return 1
```

Figure 5.6: Fully adaptive NIZK argument for $\mathcal{L}^{\vee}_{(\mathbf{B}_i \wedge \mathbf{C}_i)_{i\in[n]}}$

```
$\underline{\mathsf{Setup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds_{pp}}) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$
$(\mathsf{sps_{pp}}) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PGen}(1^\lambda; \mathsf{BG})$
$(\mathsf{sh_{pp}}, \mathsf{sh}_\tau) \leftarrow \mathsf{SH.PGen}(1^\lambda; \mathsf{BG})$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$
return $(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{sps_{pp}}, \mathsf{sh_{pp}}, \mathsf{ck})$
```

```
$\underline{\mathsf{TSetup}(1^\lambda, 1^q)}$:
$(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{scds}_\tau) \xleftarrow{\$} \mathsf{SCDS.TSetup}(1^\lambda, q)$
$(\mathsf{sps_{pp}}, \mathsf{sps}_\tau) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PTGen}(1^\lambda; \mathsf{BG})$
$(\mathsf{sh_{pp}}, \mathsf{sh}_\tau) \leftarrow \mathsf{SH.PGen}(1^\lambda; \mathsf{BG})$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $\mathsf{ck} \leftarrow (P_1, rP_1)$
return $(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{sps_{pp}}, \mathsf{ck}, \mathsf{scds}_\tau, \mathsf{sps}_\tau, \mathsf{sh}_\tau, r)$
```

Figure 5.7: LABC: Setup.

key. We essentially compute public key's pseudonyms "on the fly" guaranteeing that the signature adaption is done with respect to a valid public key. In other words, our NIZK argument is a proof of correct randomisation, where the same randomiser is used to adapt the signature and generate a pseudonymous public key.

Efficiency analysis. As the proof size is $9n - 1$ for an anonymity set of n-authorities, communication bandwidth will no longer be constant. Nevertheless, given the previously mentioned advantages we believe this is a fair trade-off for the added functionality. Furthermore, the computational cost is also substantially more efficient than similar variants (see, for instance, Table 2 from [CH20]).

In figures 5.7 (Setup) and 5.8 (Show and Verify), we present our last ABC construction, called LABC, based on the previous issuer-hiding approach. For the setup, we need to include the public parameters of the proof system. As before, we highlight the differences with the NABC construction. Key generation algorithms, Obtain and Issue remain unchanged with respect to the BABC construction.

```
$\mathsf{Show}(\mathsf{pp}, \mathsf{usk}, (\mathsf{opk}_i)_{i\in[n]}, \mathsf{opk}, \mathcal{S}, \mathcal{D}, \mathsf{cred})$          $\mathsf{Verify}(\mathsf{pp}, (\mathsf{opk}_i)_{i\in[n]}, \mathcal{S}, \mathcal{D})$
$(C, (\sigma, \tau), r, O) \leftarrow \mathsf{cred}$; $\mu, \rho \xleftarrow{\$} \mathbb{Z}_p^*$
**if** $O = (1, (o_1, o_2))$ **then** $O' = (1, (\mu \cdot o_1, o_2))$
**else** $O' = \mu \cdot O$
$\sigma' \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}((C, rC, P_1), \sigma, \tau, \mu, \rho, \mathsf{opk})$
$(C_1, C_2, C_3) \leftarrow \mu \cdot (C, rC, P_1)$
$\mathsf{cred}' \leftarrow (C_1, C_2, C_3, \sigma')$
$\mathsf{opk}' \leftarrow \mathsf{ConvertPK}(\mathsf{opk}, \rho)$
$\Pi \leftarrow \mathsf{SH.PPro}((\mathsf{opk}_i)_{i\in[n]}, \mathsf{opk}', \rho)$
$\mathsf{wss} \leftarrow \mathsf{SCDS.OpenSS}(\mu C, \mathcal{S}, O')$
$\mathsf{wds} \leftarrow \mathsf{SCDS.OpenDS}(\mu C, \mathcal{D}, O')$
$r_1, r_2, r_3, r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 C_1$; $a_2 \leftarrow r_3 P_1$
$c_1 \leftarrow \mathsf{Commit}(a_1, r_2)$
$c_2 \leftarrow \mathsf{Commit}(a_2, r_4)$
$\Sigma_1 = (\mathsf{cred}', \mathsf{wss}, \mathsf{wds}, c_1, c_2, \mathsf{opk}', \Pi)$
                                        $\xrightarrow{\Sigma_1}$
                                                  $(\mathsf{cred}', \mathsf{wss}, \mathsf{wds}, c_1, c_2, \mathsf{opk}', \Pi) \leftarrow \Sigma_1$
                                                  $e, \tilde{e} \xleftarrow{\$} \mathbb{Z}_p^*$
                                        $\xleftarrow{e, \tilde{e}}$
$\pi_1 \leftarrow \mathsf{SCDS.PoE}(\mathcal{S}, \tilde{e})$                 $(C_1, C_2, C_3, \sigma) \leftarrow \mathsf{cred}'$
$\pi_2 \leftarrow \mathsf{SCDS.PoE}(\mathcal{D}, \tilde{e})$
$z_1 \leftarrow r_1 + e \cdot (r \cdot \mu)$; $z_2 \leftarrow r_3 + e \cdot \mu$
$\Sigma_2 = ((z_i, a_i, \pi_i)_{i\in\{1,2\}}, r_2, r_4)$
                                        $\xrightarrow{\Sigma_2}$
                                                  $((z_i, a_i, \pi_i)_{i\in\{1,2\}}, r_2, r_4) \leftarrow \Sigma_2$
                                                  **check**
                                                    $z_1 C_1 = a_1 + e C_2$
                                                    $z_2 P_1 = a_2 + e C_3$
                                                    $c_1 = \mathsf{Commit}(a_1, r_2)$
                                                    $c_2 = \mathsf{Commit}(a_2, r_4)$
                                                    $\mathsf{SH.PVer}((\mathsf{opk}_i)_{i\in[n]}, \mathsf{opk}', \Pi_1)$
                                                    $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{cred}', \mathsf{opk}')$
                                                    $\mathsf{SCDS.VerifySS}(C_1, \mathcal{S}, \mathsf{wss}; \pi_1, \tilde{e})$
                                                    $\mathsf{SCDS.VerifyDS}(C_1, \mathcal{S}, \mathsf{wds}; \pi_2, \tilde{e})$
```

Figure 5.8: LABC: Show and verify.

5.6.2 Revocation Strategies

The natural approach to revocation would be to follow that described in [DHS15a], where they use the fact that randomisation of a credential is compatible with the randomisation of the accumulator and its corresponding witness. This approach requires the revocation authority to compute and maintain the witness list. As it uses the accumulator from [ATSM09], the cost of non-membership proofs is linear in the size of the accumulator (*i.e.*, revoked users), and this should be done at least once by the manager for every user. If the dynamic variant is used (as discussed in [DHS15a]), then users could be given their non-membership witness once and subsequently update it with a single constant size operation.

Another alternative to manage revocation would be to leverage NAND proofs with the following idea. When a credential is issued, users include a pseudonym given by the authority in their credential. Then, given a public list of revoked users, they compute a NAND proof for the list concerning their credential to prove they are not revoked. However, this requires the revocation list to be kept below the size limit of the set-commitment scheme.

5.6.3 Security Proofs

We omit the proof of correctness, as it follows from inspection. Again, for ease of exposition, we discuss unforgeability and anonymity of the NABC construction. In the proof of unforgeability, we distinguish whether the adversary wins the game by forging a signature, breaking the opening soundness of the commitment scheme or computing a discrete logarithm. The proof of unforgeability follows almost verbatim the strategy in [FHS19], with modifications to take care of disjoint sets.

Theorem 22. If the q-co-DL assumption holds (Section 2.3.3 on page 11), the ZKPoK's have perfect ZK, SCDS is sound, and $\mathsf{SPS\text{-}EQ}$ is EUF-CMA (Definition 27 on page 62), then NABC is unforgeable (Definition 36 on page 94).

Proof. We first introduce the following syntactic changes to the experiment, which allows us to distinguish forgeries: (1) We include the value R in the credential cred output by Obtain. (2) When the adversary makes a valid call to $\mathcal{O}_{\mathsf{Issue}}$ the experiment receives the values C, R and produces a signature σ; instead of appending $\perp$ to the list $\mathtt{CRED}$, the oracle now appends $((C, R), \sigma, \perp, \perp)$. Note the adversary's view in the experiment remains unchanged.

Assume that an efficient adversary $\mathcal{A}$ wins the unforgeability game with non-negligible probability and let $((C_1^*, C_2^*, C_3^*), \sigma^*)$ be the message-signature pair it uses and wss^* be the witness for an attribute set $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j]$, or wds^* be the witness for an attribute set $\mathcal{D}^* \subseteq \mathtt{ATTR}[j]$ for all j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. We distinguish the following cases:

Type 1: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} \neq [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, *, *) = \mathtt{CRED}[j]$ for all issuance indices j (*i.e.*, $\mathtt{OWNR}[j] \in \mathtt{HU} \cup \mathtt{CU}$). The pair $((C_1^*, C_2^*, C_3^*), \sigma^*)$ is a signature forgery and using $\mathcal{A}$ we construct and adversary $\mathcal{B}$ that breaks the EUF-CMA security of the SPS-EQ scheme.

Type 2: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} = [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, *, *) = \mathtt{CRED}[j]$ for some index j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. Since $\mathcal{A}$ wins if (1) $\mathcal{S} \nsubseteq \mathtt{ATTR}[j]$ or (2) $\mathcal{D} \subseteq \mathtt{ATTR}[j]$, it must have broken the soundness of the set-commitment scheme SCDS.

Type 3: $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}} = [(C, R, P)]_{\mathcal{R}}$ for $((C, R), \sigma, r, O) = \mathtt{CRED}[j]$ for some index j with $\mathtt{OWNR}[j] \in \mathtt{HU}$. Then we use $\mathcal{A}$ to break q-co-DL.

Type 1. In this case $\mathcal{B}$ interacts with a challenger $\mathcal{C}$ in the EUF-CMA game of $\mathsf{SPS\text{-}EQ}$ and simulates the ABC-unforgeability game for $\mathcal{A}$. The challenger $\mathcal{C}$ runs $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{crs})$ and gives opk to $\mathcal{B}$. Then $\mathcal{B}$ selects $a \xleftarrow{\$} \mathbb{Z}_p$, defines $\mathsf{scds}_{\mathsf{pp}}$

and sets $(\mathsf{osk}, \mathsf{opk}) \leftarrow (a, \mathsf{pk})$. Then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{opk})$ and simulates the environment and the oracles. All oracles are executed as in the real game, except the following which use the signing oracle instead of the signing key osk.

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{X})$: $\mathcal{B}$ computes $(C, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{ek}, \mathcal{X}; \mathsf{usk})$, picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and then queries its oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$ on $(C, r \cdot C, P)$ to obtain σ. $\mathcal{B}$ appends $(i, ((C, r \cdot C), \sigma, r, O), \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: Instead of signing (C, R, P), $\mathcal{B}$ obtains the signature σ from $\mathcal{C}$'s signing oracle. If successful, $\mathcal{B}$ appends $(i, ((C, R), \sigma, \bot, \bot), \mathcal{X})$ to $(\mathtt{OWNR}, \mathtt{CRED}, \mathtt{ATTR})$ and returns $\top$.

When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$, then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. If $\mathcal{A}$ produces a valid showing using a credential $((C_1^*, C_2^*, C_3^*), \sigma^*)$, then $\mathcal{B}$ rewinds $\mathcal{A}$ to the step after sending the commitments and restarts $\mathcal{A}$ with a new challenge $e' \neq e$. Then $\mathcal{B}$ performs a Schnorr-like knowledge extraction to obtain μ. If there is a credential $\bot \neq ((C', R'), \sigma', *, *) \in \mathtt{CRED}$ such that $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$ then $\mathcal{B}$ aborts (as the forgery is not of type 1). Otherwise, $\mathcal{B}$ has never queried a signature for class $[(C_1^*, C_2^*, C_3^*)]_{\mathcal{R}}$ and outputs $((C_1^*, C_2^*, C_3^*), \sigma*)$ as a forgery. $\mathcal{B}$, thus, breaks the EUF-CMA security of $\mathsf{SPS\text{-}EQ}$.

Type 2. Adversary $\mathcal{B}$ interacts with the challenger $\mathcal{C}$ in the soundness game for SCDS for some $q \geq 0$. First, $\mathcal{C}$ generates set-commitment parameters $\mathsf{scds_{pp}} \leftarrow (\mathsf{BG}, (s^i P_1, s^i P_2)_{i \in [q]})$ with $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$ and sends $\mathsf{scds_{pp}}$ to $\mathcal{B}$. $\mathcal{B}$ generates a key pair $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{crs})$ and runs $\mathcal{A}(\mathsf{opk})$, simulating the oracles. All oracles are as in the real game, except $\mathcal{O}_{\mathsf{Obtain}}$ in which $\mathcal{O}_{\mathsf{Issue}}$ is simulated as:
$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: $\mathcal{B}$ runs $\mathcal{A}$ twice to extract usk and sets $\mathtt{USK}[i] \leftarrow \mathsf{usk}$.
When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. Assume $\mathcal{A}$ produces a valid showing using $((C_1^*, C_2^*, C_3^*), \sigma^*)$ and a witness wss^* for the attribute set $\mathcal{S}^*$, or a valid witness wds^* for the attribute set $\mathcal{D}^*$ such that $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j]$ or $\mathcal{D}^* \subseteq \mathtt{ATTR}[j]$ for all j with $\mathtt{OWNR}[j] \in \mathtt{CU}$. Then $\mathcal{B}$ rewinds $\mathcal{A}$ to the step after sending the commitments and restarts $\mathcal{A}$ with a new challenge $e_1' \neq e_1$. $\mathcal{B}$ can then perform a knowledge extraction to obtain μ such that $C_3^* = \mu P$. Let $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$: if there is no credential $\bot = ((C', R'), *, *, *) \in \mathtt{CRED}$ then $\mathcal{B}$ aborts as the forgery was of type 1. Otherwise, let j^* be such that $((C', R'), *, *, *) = \mathtt{CRED}[j^*]$. If $\mathtt{OWNR}[j^*] \in \mathtt{HU}$ then $\mathcal{B}$ aborts as the forgery is of Type 3. Else we have $\mathtt{OWNR}[j^*] \in \mathtt{CU}$ and $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j^*]$ or $\mathcal{D}^* \subseteq \mathtt{ATTR}[j^*]$. If for some $a' \in \mathtt{ATTR}[j^*] : a'P = aP$ then $\mathcal{B}$ sets $O^* \leftarrow (1, a')$. Else, $\mathcal{B}$ sets $O^* \leftarrow (0, \mu \cdot \mathtt{USK}[\mathtt{OWNR}[j^*]])$. $\mathcal{B}$ outputs $(C_1^*, \mathtt{ATTR}[j^*], O^*, \mathcal{S}^*, \mathsf{wss}^*)$ which satisfies $\mathcal{S}^* \nsubseteq \mathtt{ATTR}[j^*] \neq \bot$ and $\mathsf{VerifySS}(\mathsf{pp}, C_1^*, \mathcal{S}^*, \mathsf{wss}^*) = 1$ or $\mathcal{B}$ outputs $(C_1^*, \mathtt{ATTR}[j^*], O^*, \mathcal{D}^*, \mathsf{wds}^*)$ which satisfies $\mathcal{D}^* \subseteq \mathtt{ATTR}[j^*] \neq \bot$ and $\mathsf{VerifyDS}(\mathsf{pp}, C_1^*, \mathcal{D}^*, \mathsf{wds}^*) = 1$.

$\mathcal{B}$'s output thus breaks the subset- or disjoint-set soundness of SCDS.

Type 3. In this case, we assume $\mathcal{A}$ can produce a forgery by computing a discrete log. We proceed via a sequence of games which are indistinguishable under q-co-DL. We denote an adversary succeeding to win **Game** i by S_i.
Game 0: The original game, which only outputs 1 if the forgery is of Type 3. **Game 1**: As **Game 0**, except for the following oracles:

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{S})$: As in **Game 0**, except that the experiment aborts if the set-commitment trapdoor is contained in $\mathcal{S}$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{S})$: As in **Game 0**, except that the experiment aborts if the set-commitment trapdoor is contained in $\mathcal{S}$.

Game 0 → Game 1: If $\mathcal{A}$ queries either set $\mathcal{S}, \mathcal{D}$ with $s \in \mathcal{S}$ or $d \in \mathcal{D}$ to one of the two oracles, then this breaks the q-co-DL assumption for $q = s$ and $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. Denoting the advantage of solving the q-co-DL by $\epsilon_{qDL}(\lambda)$, we have

$$|\Pr[S_0] - \Pr[S_1]| \leq \epsilon_{qDL}(\lambda).$$

Game 2: As **Game 1**, with the difference that the oracle $\mathcal{O}_{\mathsf{Show}}$ is run as follows

$\mathcal{O}_{\mathsf{Show}}(j, \mathcal{S})$: As in **Game 0**, but the freshness is simulated by leveraging the fact that the environment has access to the TSetup algorithm. The proof of knowledge can be run as normal, but given access to τ the elements of the Pedersen commitment can be changed.

Game 1 → Game 2: By the perfect zero-knowledge property we have

$$\Pr[S_1] = \Pr[S_2].$$

Game 3: As **Game 2**, except that the oracle $\mathcal{O}_{\mathtt{HU}}$ is run as follows:

$\mathcal{O}_{\mathtt{HU}}$: As in **Game 1**, but when executing $\mathsf{UKGen}(\mathsf{opk})$, the experiment draws $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p$ instead of $\mathsf{usk} \xleftarrow{\$} \mathbb{Z}_p^*$ and aborts if $\mathsf{usk} = 0$.

Game 2 → Game 3: Denoting by q_u the number of queries to $\mathcal{O}_{\mathtt{HU}}$, we have

$$|\Pr[S_2] - \Pr[S_3]| \leq \frac{q_u}{p}.$$

Game 4: As **Game 3**, except that when $\mathcal{A}$ eventually delivers a valid showing, the experiment rewinds $\mathcal{A}$ to the point before the commitments are sent, issues a new challenge and extracts a witness (r, σ). If the extractor fails, we abort.

Game 3 → Game 4: The success probability in **Game 4** is the same as in **Game 3**, unless the extraction fails, *i.e.,* using knowledge soundness, we have

$$|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{ks}(\lambda).$$

Game 5: As **Game 4**, except that we pick and index $k \xleftarrow{\$} [q_o]$, where q_o is the number of queries to $\mathcal{O}_{\mathsf{ObtIss}}$. Intuitively, this is the environment guessing that the adversary will use the kth issued credential in its Type 3 forgery.

The extracted witness is such that $w = (r, \mu) \in (\mathbb{Z}_p^*)^2$, and $C_2^* = rC_1^*$ and $C_3 = \mu P$. If the credential $((C', R'), \sigma', r', O') \leftarrow \mathtt{CRED}[k]$ is such that $(C', R', P') \neq \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$ then the experiment aborts. We further abort if the adversary wants to corrupt the owner of the kth credential and adapt $\mathcal{O}_{\mathtt{CU}}$ as follows:

$\mathcal{O}_{\mathtt{CU}}(i)$: As in **Game 0**, except that the experiment aborts when $i = \mathtt{OWNR}[k]$.

Game 4 → Game 5: When the forgery is of Type 3 then there exists some j s.t. for $\mathtt{CRED}[j] = ((C', R'), \sigma', r', O')$ we have $(C', R', P) = \mu^{-1} \cdot (C_1^*, C_2^*, C_3^*)$; moreover, $\mathtt{OWNR}[j] \in \mathtt{HU}$. With probability $\frac{1}{q_o}$, we have $k = j$, in which case the experiment does not abort, *i.e.,* we have

$$\Pr[S_5] \geq \frac{1}{q_o} \Pr[S_4]$$

We will now show that $\Pr[S_5] \leq \epsilon_{DL}(\lambda)$, where $\epsilon_{DL}(\lambda)$ is the advantage of solving the DLP. $\mathcal{B}$ plays the role of the challenger for $\mathcal{A}$ in **Game 5** and obtains a $\mathbb{G}_1$-DLP instance (BG, xP). $\mathcal{B}$ generates a key pair $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{crs})$. Then, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{opk})$ and simulates the oracles as in **Game 5**, except for $\mathcal{O}_{\mathsf{Obtlss}}$, whose simulation is as follows:

$\mathcal{O}_{\mathsf{Obtlss}}(i, \mathcal{S})$: Let this be the jth query. $\mathcal{B}$ first computes $C \leftarrow \mathtt{USK}[i]\cdot\mathsf{Ch}_{\mathcal{X}}(a)\cdot xP(= x\cdot C), O = (O, \mathtt{USK}[i])$ and appends $\mathsf{cred} = ((C, R)\sigma, \perp, O)$ to $\mathtt{CRED}$. Otherwise, **B** proceeds as in **Game 5**.

Note that since **Game 2**, the third component r of the credential is not required to simulate $\mathcal{O}_{\mathsf{Show}}$ queries. When $\mathcal{A}$ outputs $(\mathcal{S}^*, \mathcal{D}^*, \mathsf{st})$, then $\mathcal{B}$ runs $\mathcal{A}(\mathsf{st})$ and interacts with $\mathcal{A}$ as the verifier in the showing protocol. If $\mathcal{A}$ wins **Game 5** using (C_1^*, C_2^*, C_3^*) and conducting the proof of knowledge on the freshness, then $\mathcal{B}$ can rewind $\mathcal{A}$ and extract a witness $w = (r', \mu) \in (\mathbb{Z}_p^*)^3$ such that $C_2^* = r'C_1^*$ and $C_3^* = \mu P$. Further, we have that $((C', R'), \sigma', \perp, O') = \mathtt{CRED}[k]$. In the end, $\mathcal{B}$ outputs r' as a solution to the DLP in $\mathbb{G}_1$. We thus have

$$\Pr[S_5] \leq \epsilon_{DL}(\lambda)$$

Collecting the success probabilities, we have $\Pr[S_0] \leq q_o \cdot \epsilon_{DL}(\lambda) + \epsilon_{ks}(\lambda) + \frac{q_u}{p} + \epsilon_{qDL}(\lambda)$ where $q = t$ and q_o and q_u and the number of queries to $\mathcal{O}_{\mathsf{Obtlss}}$ and $\mathcal{O}_{\mathtt{HU}}$ respectively. □

Theorem 23. If the DDH assumption holds, the ZKPoK's have perfect ZK, and the SPS-EQ perfectly adapts signatures (Definition 28 on page 63); then, NABC is anonymous (Definition 37 on page 94).

Proof. The following proof is an adaptation (most of it verbatim) of the one given in [FHS19]. The only difference is that since we use a CRS and manage a slightly different definition for perfect adaption, we need to adjust the previous proof for this new setting. For ease of exposition, we only consider selective disclosures showings in the proof, but the adaptation for NAND showings follows directly. As in [FHS19], the proof proceeds by defining a sequence of indistinguishable games in the last of which the answers of $\mathcal{O}_{LoR}$ are independent of the bit b.

We assume that the adversary $\mathcal{A}$ will call $\mathcal{O}_{LoR}$ for some $(j_0, j_1, \mathcal{S})$ with both $\mathtt{OWNR}[j_0], \mathtt{OWNR}[j_1] \in \mathtt{HU}$. This is w.l.o.g. as otherwise the bit b is perfectly hidden from $\mathcal{A}$. Henceforth, we denote the event that the adversary wins Game i by S_i.
Game 0: The original anonymity game (Definition 37).

Game 1: As **Game 0**, except we replace Setup with TSetup.

Game 0 → Game 1: The adversary's view does not change so we have

$$\Pr[S_0] = \Pr[S_1].$$

Game 2: As **Game 1**, except that the experiment runs $\mathcal{O}_{\mathsf{LoR}}$ as follows:
$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 1**, but the ZKPoK for (C_1^*, C_2^*, C_3^*) is simulated.
Game 1 → Game 2: By perfect zero-knowledge of Π_2, we have

$$\Pr[S_1] = \Pr[S_2].$$

Game 3: As **Game 2**, except for the following changes. Let q_u be (an upper bound on) the number of queries made to $\mathcal{O}_{\mathtt{HU}}$. At the beginning of **Game 2** picks $k \overset{\$}{\leftarrow} [q_u]$ (it guesses that the user that owns the j_bth credential is registered at the kth call to $\mathcal{O}_{\mathtt{HU}}$) and runs $\mathcal{O}_{\mathtt{HU}}$, $\mathcal{O}_{\mathtt{CU}}$ and $\mathcal{O}_{\mathsf{LoR}}$ as follows:
$\mathcal{O}_{\mathtt{HU}}(i)$: As in **Game 2**, except if this is the kth call to $\mathcal{O}_{\mathtt{HU}}$ then it additionally defines $i^* \leftarrow i$.
$\mathcal{O}_{\mathtt{CU}}(i, \mathsf{upk})$: If $i \in \mathtt{CU}$ or $i \in I_{\mathsf{LoR}}$, it returns $\perp$ (as in the previous games). If $i = i^*$ then the experiment stops and outputs a random bit $b' \overset{\$}{\leftarrow} \{0, 1\}$. Otherwise, if $i \in \mathtt{HU}$ it returns user i's usk and credentials and moves i from $\mathtt{HU}$ to $\mathtt{CU}$; and if $i \notin \mathtt{HU} \cup \mathtt{CU}$, it adds i to $\mathtt{CU}$ and sets $\mathtt{UPK}[i] \leftarrow \mathsf{upk}$.
$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 2**, except that if $i^* \neq \mathtt{OWNR}[j_b]$, the experiment stops outputting $b' \overset{\$}{\leftarrow} \{0, 1\}$.
Game 2 → Game 3: By assumption, $\mathcal{O}_{\mathsf{LoR}}$ is called at least once with some input $(j_0, j_1, \mathcal{S})$ with $\mathtt{OWNR}[j_0]$, $\mathtt{OWNR}[j_1] \in \mathtt{HU}$. If $i^* = \mathtt{OWNR}[j_b]$ then $\mathcal{O}_{\mathsf{LoR}}$ does not abort and neither does $\mathcal{O}_{\mathtt{CU}}$ (it cannot have been called on $\mathtt{OWNR}[j_b]$ before that call to $\mathcal{O}_{\mathsf{LoR}}$ (otherwise $\mathtt{OWNR}[j_b] \notin \mathtt{HU}$); if called afterwards, it returns $\perp$, since $i^* \in I_{\mathsf{LoR}}$). Since $i^* = \mathtt{OWNR}[j_b]$ with probability $\frac{1}{q_u}$, the probability that the experiment does not abort is at least $\frac{1}{q_u}$, and thus

$$\Pr[S_3] \geq (1 - \tfrac{1}{q_u})\tfrac{1}{2} + \tfrac{1}{q_u} \cdot \Pr[S_2].$$

Game 4: Same as **Game 3**.

Game 3 → Game 4: Let $(\mathsf{BG}, xP_1, yP_1, zP_1)$ be a DDH instance for $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. After initialising the environment, the simulation initialises a list $L \leftarrow \emptyset$. The oracles are simulated as in **Game 3**, except for the subsequent oracles, which are simulated as follows:
$\mathcal{O}_{\mathtt{HU}}(i)$: As in **Game 3**, but if this is the kth call then, besides setting $i^* \leftarrow i$, it sets $\mathtt{USK}[i]\perp$ and $\mathtt{UPK}[i] \leftarrow xP_1$ (which implicitly sets $\mathsf{usk} \leftarrow x$)
$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$: As in **Game 3**, except for the computation of the following values if $i = i^*$. Let this be the jth call to this oracle. If $s \notin \mathcal{X}$, it computes C as $C \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot xP_1$ and sets $L[j] \leftarrow \perp$. If $s \in \mathcal{X}$ it picks $\rho \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, computes C as $C \leftarrow \rho \cdot xP_1$, sets $L[j] \leftarrow \rho$ and simulates the ZKPoK for upk (by the perfect ZK property of the simulation is perfect). (In both cases C is thus distributed as in the original game.)
$\mathcal{O}_{\mathsf{Show}}(i, \mathcal{S})$: As in **Game 3**, with the difference that if $\mathtt{OWNR}[j] = i^*$ and $s \notin \mathcal{S}$ it computes the witness $\mathsf{wss} \leftarrow \mu \mathsf{Ch}_{\mathcal{X} \setminus \mathcal{S}}(s) \cdot xP_1$. ($\mathsf{wss}$ is thus distributed as in the original game.)
$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 3**, with the following difference. Using self-reducibility of DDH, it picks $s, t \overset{\$}{\leftarrow} \mathbb{Z}_p$ and computes $Y' \leftarrow t \cdot yP_1 + sP_1 = y'P_1$ with $y' \leftarrow ty + s$, and $Z' \leftarrow t \cdot zP_1 + s \cdot xP_1 = (t(z - xy) + xy')P_1$.(If $z \neq xy$ then

Y' and Z' are independently random; otherwise $Z' = y'X$.) It performs the showing using the following values (implicitly setting $\mu \leftarrow y'$):

- If $s \notin \mathtt{ATTR}[j_b]$: $C_1 \leftarrow \mathsf{Ch}_{\mathcal{X}}(s)Z'$ and $\mathsf{wss} \leftarrow \mathsf{Ch}_{\mathcal{S}}(s)^{-1}C_1$
- If $s \in \mathtt{ATTR}[j_b]$ and $s \notin \mathcal{S}$: $C_1 \leftarrow \rho Z'$ with $\rho \leftarrow L[j_b]$ and $\mathsf{wss} \leftarrow \mathsf{Ch}_{\mathcal{S}}(s)^{-1}C_1$;
- If $s \in \mathcal{S}$: $c_1 \leftarrow \rho Z'$ with $\rho \leftarrow L[j_b]$ and $\mathsf{wss} \leftarrow \perp$;

Apart from an error event happening with negligible probability, we have simulated **Game 3** if the DDH instance was "real" and **Game 4** otherwise. If $xP_1 = 0_{\mathbb{G}_1}$, or if during the simulation of $\mathcal{O}_{\mathsf{LoR}}$ it occurs that $Y' = 0_{\mathbb{G}_1}$ or $Z' = 0_{\mathbb{G}_1}$, then the distribution of values is not as in one of the two games. Otherwise, we have implicitly set $\mathsf{usk} \leftarrow x$ and $\mu \leftarrow y'$ (for a fresh value y' at every call of $\mathcal{O}_{\mathsf{LoR}}$). In case of a DDH instance, we have (depending on the case) $C1 \leftarrow \mathsf{usk}\mu\mathsf{Ch}_{\mathcal{X}}(s) \cdot P_1$ (or $C_1 = \rho \cdot x\mu \cdot P_1 = \mu \cdot C$). Letting $\epsilon_{DDH}(\lambda)$ denote the advantage of solving the DDH problem and q_l the number of queries to the $\mathcal{O}_{\mathsf{LoR}}$, we have

$$|\Pr[S_3] - \Pr[S_4]| \leq \epsilon_{DDH}(\lambda) + (1 + 2q_l)\tfrac{1}{p}.$$

Game 5: Same as **Game 4**.

Game 4 → Game 5: Let $(\mathsf{BG}, xP_1, yP_1, zP_1)$ be a DDH instance for $\mathsf{BG} = \mathsf{BGGen}(1^\lambda)$. After initialising the environment, the simulation initialises a list $L \leftarrow \emptyset$. The oracles are simulated as in **Game 4**, except for the subsequent oracles, which are simulated as follows:

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$: As in **Game 4**, except for the computation of the following values if $i = i^*$.Let this be the jth call to this oracle. It first picks $u \xleftarrow{\$} \mathbb{Z}_p$ and sets $X' \leftarrow xP_1 + u \cdot P_1$ and $L[j] \leftarrow u$. If $s \notin \mathcal{X}$, it computes $C \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot \mathtt{USK}[i]P_1$ and $R \leftarrow \mathsf{Ch}_{\mathcal{X}}(s) \cdot \mathtt{USK}[i]X'$. If $s \in \mathcal{X}$, it picks $\rho \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $C \leftarrow \rho P_1$ and $R \leftarrow \rho X'$. In both cases it sets $r \leftarrow \perp$ (r is implicitly set to $r \leftarrow x' := x + u$ and C and $R = rC$ are distributed as in the original game; unless $X' = 0_{\mathbb{G}_1}$). Note that, since the ZKPoK in $\mathcal{O}_{\mathsf{Show}}$ is simulated, r is not used anywhere in the game.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S})$: As in **Game 4**, with the difference that it fetches $u \leftarrow L[j_b]$, picks $s, t \xleftarrow{\$} \mathbb{Z}_p$ and recomputes $Y' \leftarrow t \cdot yP_1 + sP_1 = y'P_1$ with $y' \leftarrow ty + s$,and $Z' \leftarrow t \cdot zP_1 + s \cdot xP_1 + ut \cdot yP_1 + us \cdot P_1 = (t(z - xy) + x'y')P_1$. It performs the showing as in the previous simulation (using the new Y', Z' and $\mu \leftarrow y'$).

Apart from an error event happening with negligible probability, we have simulated **Game 4** if the DDH instance was valid and **Game 5** otherwise. If $X' = 0_{\mathbb{G}_1}$ during the simulation of $\mathcal{O}_{\mathsf{Obtain}}$, or if during the simulation of $\mathcal{O}_{\mathsf{LoR}}$ it occurs that $Y' = 0_{\mathbb{G}_1}$ or $Z' = 0_{\mathbb{G}_1}$ then the distribution of values is not as in one of the two games. Otherwise, we have implicitly set $r \leftarrow x'$ (for a fresh value x' at every call of $\mathcal{O}_{\mathsf{Obtain}}$). Letting $\epsilon_{DDH}(\lambda)$ denote the advantage of solving the DDH problem, and q_o and q_l be the number of queries to $\mathcal{O}_{\mathsf{Obtain}}$ and $\mathcal{O}_{\mathsf{LoR}}$, respectively, we get

$$|\Pr[S_4] - \Pr[S_5]| \leq \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\tfrac{1}{p}.$$

In **Game 5**, by definition of perfect adaption the oracle $\mathcal{O}_{\mathsf{LoR}}$ returns a signature that is a random element in the space of signatures conditioned to verify with the shown credential (each generated with fresh independent randomness $\mu \leftarrow y'$, when

5

calling the oracle $\mathcal{O}_{\mathsf{LoR}}$), and with respect to a simulated proof. Hence, the bit b is information-theoretically hidden from $\mathcal{A}$, and we have $\Pr[S_5] = \frac{1}{2}$. Therefore, we have that:

$$\begin{array}{rcl}
\Pr[S_4] & \leq & \Pr[S_5] + \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\dfrac{1}{p} = \dfrac{1}{2} + \epsilon_{DDH}(\lambda) + (q_o + 2q_l)\dfrac{1}{p}, \\
\Pr[S_3] & \leq & \Pr[S_4] + \epsilon_{DDH}(\lambda) + (1 + 2q_l)\dfrac{1}{p} \leq \dfrac{1}{2} + 2 \cdot \epsilon_{DDH}(\lambda) + (1 + q_o + 4q_l)\dfrac{1}{p}, \\
\Pr[S_2] & \leq & \dfrac{1}{2} + q_u \cdot \Pr[S_3] - \dfrac{1}{2} \cdot q_u \leq \dfrac{1}{2} + q_u \cdot (2 \cdot \epsilon_{DDH}(\lambda) + (1 + q_o + 4q_l)\dfrac{1}{p}),
\end{array}$$

where $\Pr[S_2] = \Pr[S_1] = \Pr[S_0]$; q_u, q_o and q_l are the number of queries to $\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathsf{Obtain}}$ and $\mathcal{O}_{\mathsf{LoR}}$, respectively. Assuming security of the ZKPoKs and DDH, the adversary's advantage is thus negligible. □

Theorem 24. If the underlying signature scheme is a SPS-EQ which perfectly adapts signatures (Definition 32 on page 64), LABC supports issuer-hiding (Definition 38 on page 95).

Proof. Let us first observe that the adversary can guess the bit j^* with probability $1/n$. By definition of perfect adaption, for all tuples $(\mathsf{pp}, [\mathsf{opk}]_j, [\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho)$ s.t $(\sigma, \tau) \xleftarrow{\$} \mathsf{Sign}(\mathsf{pp}, \mathsf{osk}_j, [\boldsymbol{m}]_i)$, we have that $[\mu \cdot \boldsymbol{m}]_i$ and $[\rho \cdot \mathsf{opk}]_j$ are identically distributed in the message and key spaces, where $([\mu \cdot \boldsymbol{m}]_i, \sigma^*) \leftarrow \mathsf{ChgRep}([\boldsymbol{m}]_i, (\sigma, \tau), \mu, \rho, [\mathsf{opk}]_j)$ and $[\rho \cdot \mathsf{opk}]_j \leftarrow \mathsf{ConvertPK}(\mathsf{opk}_j, \rho)$. Furthermore, we also have that σ^* is a random element in the space of signatures conditioned on $\mathsf{Verify}([\mu \cdot \boldsymbol{m}]_i, \sigma^*, [\rho \cdot \mathsf{opk}]_j) = 1$. Therefore, an adversary with access to $[\mu \cdot \boldsymbol{m}]_i$, σ^* and $[\rho \cdot \mathsf{opk}]_j$ can only guess the bit j^* with probability at most $1/n$. □

5.7 Evaluation

We compare the efficiency of state-of-the-art ABC and ours (Section 5.6) in Table 5.3. For ease of exposition, we list the work in [FHS19] next to ours and consider an instantiation of it in the CRS model, using the same ZKPoK's as the ones used in Section 5.6.

When looking at a whole, the work in [San20] presents very good results while also allowing showings to prove relationships between attributes and to consider malicious keys. Nevertheless, security of the related construction is proven in the GGM model and, thus, falls short in that aspect. The same applies to the works from [HP20] and [FHS19].

Although we only considered the classical setting (credentials are issued by a single authority), it is worth mentioning that [HP20] does consider multi-authorities. As the authors point out, they base their construction on aggregatable signatures to allow multi-authorities and obtain the most efficient showing for the users. Their security model follows the game-based approach from [FHS19], but because of the multi-authority setting, they also consider malicious credential issuers with adaptive corruptions and collusions with malicious users. Unfortunately, this is done assuming that the keys are *honestly* generated.

[TG20] uses a set-commitment scheme which, alongside an SDH-based signature, leads to a credential system that supports a variety of show proofs for complex statements among which AND and NAND are included. For this reason, we also compare our work with the one from [TG20], considering NAND showings. In terms of security models, the authors provide a formalization for impersonation attacks and prove their scheme secure against impersonation under active and concurrent attacks. The security of their ABC scheme is proven in the standard model and providing a tight reduction.

Considering the different trade-offs, our ABC provides very similar performance when compared to [FHS19] and it is not too distant from the most efficient ones either. Unlike the rest, it can be adapted to different scenarios in case that reducing the verification cost is not needed. It can also be efficiently adapted to provide revocation features. Furthermore, as for many practical applications, the ability to perform AND and NAND showings suffices; we also achieve a good level of expressiveness. Finally, the issuer-hiding feature makes it suitable for scenarios in which the rest of the alternatives struggle.

5.8 Conclusions and Future Work

In this chapter, we provided improved constructions for different primitives in order to obtain an efficient and versatile credential system whose security is proven in the standard model. Our results explore multiple paths to extend the ABC framework of [FHS19] to cover more applications and scenarios where it can be used.

To improve the expressiveness of the set-commitment scheme in [FHS19], we allow openings on sets of attributes disjoint from those possessed by a user. We also enhance efficiency by employing the trick of allowing the prover to compute a proof of exponentiation, leaving the verifier only to compute a polynomial division.

We develop an issuer-hiding notion to allow a credential-bearing user to hide their issuing organisation upon presentation of the credential. As we increasingly see cases of (algorithmic) bias against users, notions such as this are of growing importance. Moreover, we also discussed directions to integrate revocation features.

Considering future work, we worked in the classical setting, where each credential is issued by a single authority. Therefore, it would be interesting to follow the related work on aggregatable signatures to see if we could lift SPS-EQ to the multi-authority setting. Furthermore, while our set-commitment scheme is more expressive than [FHS19], it is still less expressive than [TG20]. Hence, it would also be interesting to see if the set-commitment scheme introduced there would yield greater expressiveness to the ABC presented here. Likewise, to verify if the stronger security notions presented here could enhance the construction from [TG20].

ABC	[San20]	[HP20]	[TG20]	[FHS19]	Our work
			Parameters size (n-attributes)		
ek	$(\frac{n^2+n+2}{2})_{\mathbb{G}_1}+n_{\mathbb{G}_2}$	$(2n+2)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+(n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+(n+1)_{\mathbb{G}_2}$	$(n+1)_{\mathbb{G}_1}+(n+1)_{\mathbb{G}_2}$
Cred	$2_{\mathbb{G}_2}$	$4_{\mathbb{G}_1}$	$1_{\mathbb{G}_1}+6_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+1_{\mathbb{G}_2}+2_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1}+6_{\mathbb{G}_2}+3_{\mathbb{Z}_p}$
			Bandwidth		
Issue	$4_{\mathbb{G}_2}+2_{\mathbb{Z}_p}$	$n_{\mathbb{G}_1}$	$3_{\mathbb{G}_1}+(n+3)_{\mathbb{Z}_p}$	$12_{\mathbb{G}_1}+1_{\mathbb{G}_2}+8_{\mathbb{Z}_p}$	$14_{\mathbb{G}_1}+11_{\mathbb{G}_2}+7_{\mathbb{Z}_p}$
Show	$2_{\mathbb{G}_1}+2_{\mathbb{G}_2}+1_{\mathbb{G}_T}+2_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+1_{\mathbb{Z}_p}$	$3_{\mathbb{G}_1}+5_{\mathbb{Z}_p}$	$10_{\mathbb{G}_1}+1_{\mathbb{G}_2}+8_{\mathbb{Z}_p}$	$18_{\mathbb{G}_1}+14_{\mathbb{G}_2}+4_{\mathbb{Z}_p}$
			k-of-n attributes (AND)		
Usr	$(2(n\text{-}k)+2)_{\mathbb{G}_1}$,$2_{\mathbb{G}_2}$,**1**	$6_{\mathbb{G}_1}$	$(6+n\text{-}k)_{\mathbb{G}_1}$	$(11+n\text{-}k)_{\mathbb{G}_1}$,$1_{\mathbb{G}_2}$,**8**	$(20+n\text{-}k)_{\mathbb{G}_1}$,$(k\text{-}1)_{\mathbb{G}_2}$,**19**
Ver	$(k+1)_{\mathbb{G}_1}$,$1_{\mathbb{G}_T}$,**5**	$4_{\mathbb{G}_1}$,$2n_{\mathbb{G}_2}$,**3**	$5_{\mathbb{G}_1}$,$(k+1)_{\mathbb{G}_2}$,**3**	$4_{\mathbb{G}_1}$,$(k+1)_{\mathbb{G}_2}$,**10**	$10_{\mathbb{G}_1}$,**16**
			k-of-n attributes (NAND)		
Usr	N/A	N/A	$(6+n)_{\mathbb{G}_1}$	N/A	$(31+n)_{\mathbb{G}_1}$,$(9+2k)_{\mathbb{G}_2}$,**19**
Ver	N/A	N/A	$(2k+5)_{\mathbb{G}_1}$,$(k+3)_{\mathbb{G}_2}$,**3**	N/A	$10_{\mathbb{G}_1}$,**17**

Table 5.3: Efficiency of ABCs considering issuing and showing interactions (the number of pairings is marked in bold).

6
Protego: ABC for Permissioned Blockchains

Computers and networks have gotten 12 million times faster, people still complain cryptographic pairings are too slow. Including the Facebook like button on your page adds more latency than a pairing.

— Ian Miers

This chapter relies on joint work with Aisling Connolly, Jérôme Deschamps and Pascal Lafourcade. The presented material is based on [CDLPK22].

6.1 Introduction

When first introduced, the core use of blockchains was to facilitate *permissionless* participation; anyone could join and participate in any manner. On the other hand, the prospect of a *distributed ledger* is also of great use within, for example, a consortium, where several authorised organisations wish to share information among the group, but not necessarily to the public as a whole. This need gave rise to *permissioned* blockchains whereby authorities are established to define a set of participants. When a single authority controls all of the blockchain, we refer to it as *private permissioned*. When a federation of authorities (*consortium*), each in control of a subset of the allowed participants, shares it, we use the term *public permissioned* (or simply *federated*). While private permissioned blockchains are mostly used as intra-enterprise solutions (or when offered as a service), federated blockchains are preferred for inter-enterprise solutions.

The use of federated blockchains addresses the need to run a common business logic within a closed environment. For example, one can consider pharmaceutical companies that would like to trade sensitive information about product developments and agree on supplies or prices in a consortium with partial trust. Protecting privacy while being compliant with regulations and *Know Your Customer* practices is a recurrent problem in such scenarios. Agreeing with other entities to run a shared business logic should not imply that everything needs be public within the consortium. Privacy still needs to be provided without affecting existing regulations, *e.g.,* when considering bilateral agreements.

The most developed permissioned platform is Hyperledger Fabric (or simply Fabric). By default, it provides no privacy features as everything (users and transactions) within a given federation is public. However, motivated by the need to protect business interests and to meet regulatory requirements, some privacy features were integrated, notably, the Identity Mixer [Zur13, CV02] (or Idemix for short). This anonymous attribute-based credential scheme gave the first glimpse of privacy for users within a consortium. However, the current Idemix integration with Fabric is still quite limited in efficiency, functionality, and privacy levels. Recently, a prototype that extends Idemix [BDCET21], including revocation, auditing capabilities, and more privacy-preserving features, has emerged. This extension is based on *delegatable credentials* but still has inherent limitations. For example, it puts all trust on a root certificate authority and requires the generation of many zero-knowledge proofs to sign a transaction.

Recent results introduced newer models to build ABC's, providing a host of extra functionalities and more efficient constructions. The main goal of this work is to leverage such results, to position them in the blockchain scenario and provide an alternative to Idemix (and its extension) in a bid to overcome existing privacy and functional limitations while also improving efficiency.

6.2 Contributions

We explore alternative mechanisms to build a practical ABC. First, we extend recent works based on SPS-EQ [FHS19, CLPK22] to support auditability features while also integrating the revocation ideas from [DHS15a]. Such extension relies on the ROM (already present in the blockchain setting) to generate non-interactive showing proofs. We also present and discuss two alternatives to the use of delegatable credentials to hide the identity of credential issuers. To do so, we build upon the issuer-hiding strategies presented in the previous chapter. Compared to the ABC construction from Chapter 5, the modifications are as follows:

1. We adapt the model to non-interactive showings.
2. We keep the SCDS scheme as it is but replace the signature scheme with the one given in [CL19] to improve efficiency at the cost of working in the GGM.
3. We define a revocation authority as in [DHS15a], and an auditing authority in the model (not considered in the previous works).
4. We build a *malleable* NIZK argument that can be pre-computed to obtain a more efficient issuer-hiding feature.

As a result, we build Protego and Protego Duo, two new ABC's for permissioned blockchains that differ on the issuer-hiding approach. Both support revocation and auditing features, which are essential to enable a wider variety of use cases for permissioned blockchains. We discuss how to integrate our work with Fabric, compare it with Idemix and its recent extensions, and provide a prototype implementation showing that Protego and Protego Duo are more than two times faster than the most recent Idemix extension. Furthermore, a showing proof in Protego Duo is constant-size (8.3 kB), surpassing [BDCET21] in which the proof size grows linearly with the number of *attributes and delegation levels*.

6.3 Related Work

We describe the related work following two main streams; the results addressing privacy concerns in Fabric and parallel research developments.

Privacy concerns in Fabric. The most closely related work appears with the introduction of Idemix [Zur13] and its extension to include revocation and auditability [BDCET21]. Adding auditability is crucial for permissioned blockchains as they are often used in heavily regulated industries. Privacy-preserving auditing for distributed ledgers was introduced in [NVV18] under the guise of zkLedger. This general solution offered great functionality in that it provided confidentiality of transactions, and privacy of the users within the transaction. However, it assumed low transaction volume between few participants and, as such, is quite limited in scalability. Fabric-friendly auditable token payments were introduced in [ACC+20] and were based on threshold blind signatures. The core idea to achieve auditability was to encrypt the user's public key under the public key of an auditor. The same approach is used in [BDCET21], which we also use in this work. Although the auditing ideas are similar, the construction pertains solely to transaction privacy and offers no identity privacy for a user. Following the approach of gaining auditability of transactions, auditable smart contracts were captured by FabZK [KDJL+19], which is based on Pedersen commitments and zero-knowledge proofs. To achieve auditability, the structure of the ledger is modified, and as such, existing permissioned blockchain platforms would need to undergo significant changes to achieve auditability.

In Fabric, the validity of a transaction is established by obtaining *endorsements* from peers in the network. One of the limitations in Idemix and its extension is the lack of privacy or anonymity for endorsing peers. As a possible solution, [MR19] proposes an endorsement policy based on ring signatures, which do not reveal the endorsement set. Another approach to obtaining privacy-preserving endorsements was described in [ADCNS19], leveraging Idemix credentials to gain endorser privacy and, as such, inheriting the limitations (notably leaking the endorser's organisation) that come with Idemix.

Attribute-based credentials. As seen in the previous chapter, early anonymous credential schemes were built from blind signatures, whereby a user obtained a blind signature from an issuer on the user's commitment to its attributes. When the user later authenticates, they provide the signature, the shown attributes, and a proof of knowledge of all unshown attributes. These schemes are limited as they can only be shown once. Subsequent work like the one underlying Idemix [CL04] allowed for an arbitrary number of unlinkable showings. A user obtains a signature on a commitment on attributes, randomises the signature, and proves in zero-knowledge that the randomised signature corresponds to the shown and unshown attributes. The results presented in Chapter 5 showed how to circumvent multiple inefficiencies and drawbacks from previous constructions using set-commitment schemes and SPS-EQ. The work presented in this chapter builds on top of the contributions from Chapter 5 but relies on the GGM as we switch our attention to the most efficient alternatives. Therefore, we use the mercurial signature scheme from [CL19] instead of the one given in Chapter 4.

6.4 Privacy Notions in Hyperledger Fabric

Hyperledger Fabric is a permissioned blockchain framework focused on modularity and performance. Fabric's approach to modularity rests on decoupling services so to ease the configuration of blockchain networks within federated consortia. The *Membership Service Provider* (MSP) is responsible for the participant's enrollment, administration and management of related certificates (*e.g.,* issuance, distribution and validation). The concept of MSP resembles that of a Certificate Authority (CA). An *Ordering Service* orchestrates the consensus mechanism. As of version 2.4, Fabric only supports the RAFT [OO14] consensus protocol, which is crash fault tolerant. A *Chaincode Service* provides the required interfaces and functionalities to execute smart-contracts (called *chaincode* in Fabric). A *P2P Gossip Service* is responsible for disseminating blocks output by the ordering service to other peers. Additionally, Fabric also supports different policy configurations for the execution or modification of chaincodes.

Instead of performing an order-execute flow to process transactions (like Ethereum), Fabric has an *execute-order*-validate flow. In the former approach, transactions are first ordered to produce a block, and once the block is accepted, every node in the network will execute the transactions sequentially. In contrast, transactions in Fabric are executed *speculatively.*

First, a transaction proposal is created, and a subset of peers (*endorsers*) execute them (potentially in parallel) attesting to the result (*i.e.,* determining the corresponding write set given the read set). Every endorser executes each transaction proposal according to their version of the ledger.

Subsequently, provided that a transaction proposal obtains sufficient endorsements (as required by the endorsement policy, later explained), it is sent to the ordering service. At this point, the ordering service groups different transactions (without re-executing them) and forms a new block, delivering it to the peers.

As every transaction execution was done speculatively (without knowing whether or not such execution would be valid concerning the latest ledger state), they need to be validated after they were confirmed in a block by the ordering service. Every peer does this locally and sequentially upon receiving a new block. Again, without re-executing the transactions.

Every peer consistently applies the read/write sets for each transaction to their latest ledger state during the validation stage, updating it accordingly. For this reason, it could be the case that more than one transaction contained in a block becomes *invalid* after the validation phase (*e.g.,* due to an outdated read set following a change by a previous transaction in the block).

6.4.1 Overview of Hyperledger Fabric's Transaction Flow

Proposal. To execute a chaincode, one has to form a transaction proposal that specifies the function and arguments to be called. For every chaincode, an endorsement policy mandates which organisations execute and validate the transaction output. An example of endorsement policy can be "Admin.Org1 AND Member.Org2", meaning that the chaincode execution will be considered valid iff it was executed by a peer from Org1 with an Admin role and a peer from Org2. A proposal tx has the following fields: $\mathsf{tx} = <\mathsf{clientID}, \mathsf{chaincodeID}, \mathsf{txPayload}, \mathsf{timestamp}, \mathsf{clientSig}>$, where $\mathsf{txPayload}$ contains the chaincode's source code, metadata, and the corresponding endorsement policy id and parameters.

Execution. Clients are responsible for obtaining the endorsements for every proposal before submitting it to the ordering service. During this phase, a set of endorsers receive a proposal and simulate its execution against their local copy of the ledger. The result of this simulation is a read/write set for the proposal signed by the endorsing peer. Endorsers can also check access control policies against the client during this stage to verify that they are authorised to perform the proposed operation. An endorsement response has the following fields (where tid is a hash of tx): $<\mathsf{EndorserID}, \mathsf{tid}, \mathsf{chaincodeID}, \mathsf{txPayload}, \mathsf{readset}, \mathsf{writeset}, \mathsf{EndorserSig}>$.

Assembly. The submitting client collects the endorsers' responses, generates a transaction containing the proposal and the endorsements, and broadcasts it to the ordering service. During this phase, clients should check that the different endorsements are consistent concerning the read/write sets. They only need to obtain the required replies to verify that the endorsement policy is satisfied.

Consensus. Nodes from the ordering service receive transactions from the previous step, produce a new block and broadcast it to the network.

Validation. Peers receive new blocks, validate the transactions sequentially and commit them to the ledger. Those failing to verify the endorsement policy or with inconsistent read/write sets are marked as invalid but still committed.

6.4.2 Privacy Concerns

By default, Fabric does not provide any privacy-preserving feature; reading the blockchain anyone can learn:

- Who triggered a chaincode function using which arguments (proposals are signed by the clients).

- Who vouched for its execution (endorsers also sign their responses) concerning reading and writing sets.
- Why a given transaction was marked as invalid (either because of invalid read/write sets or because the endorsement policy check failed).

Furthermore, checking access control and endorsement policies links different organisations, users and their attributes to concrete actions on the system.

Such limitations severely restrict the use of Fabric. From the user's perspective, this is quite clear and impacts the enforcement of different regulations, such as the GDPR. For organisations, the case is similar. Consider a consortium of pharmaceutical organisations that run a common business logic to exchange information on medical research. If the entity behind a request is known, other organisations can infer (based on the request) which drug the entity in question is trying to develop. Moreover, regardless if the endorsement policy is public or not, if the endorsers are known, information about who executes what can disclose business relations between pharmaceuticals.

6.4.3 Idemix, its Limitations, and Extensions

Several proposals addressed the limitations above, but only Idemix has been integrated into Fabric. Idemix allows an MSP to issue attribute-based credentials to enable a user to sign a transaction anonymously. In brief, users generate a zero-knowledge proof attesting that the MSP issued them a credential on its attributes to sign a transaction. Fabric's support for Idemix was added in v1.3, providing the first solution to tackle the problem of *participant* privacy. Unfortunately, as for v2.4, the Idemix implementation still suffers severe limitations:

1. It supports a fixed set of only four attributes.
2. It does not support revocation features.
3. Credentials leak the MSP ID, meaning that anonymity is local to users within an organisation. For this reason, current deployments can only use a single MSP for the whole network, introducing a single point of failure.
4. It does not support the issuance of Idemix credentials for the endorsing peers, meaning that the identity of endorsers is always leaked.

As Fabric is the most developed and maintained open-source permissioned blockchain platform, there is an urgent need to overcome these limitations. The most promising effort to extend the functionality of Idemix appeared in [BDCET21]. They aimed to extend the original credential system to support delegatable credentials while integrating revocation and auditability features (solving three of the four limitations). Below we outline the main ideas introduced in [BDCET21].

Delegatable Credentials. In a bid to overcome the issue of Idemix credentials leaking the MSP ID, and thus the affiliation of the user, a trusted root authority provides credentials to intermediate authorities. This way, users can obtain credentials from intermediate authorities. To sign a transaction, the user must generate a zero-knowledge proof attesting that (1) the signer owns the credential; (2) the signature is valid; (3) all adjacent delegation levels are legitimate; and (4) that the top-level public key belongs to the root authority.

Revocation and Auditability. To generate efficient proofs of non-revocation, the system timeline is divided into *epochs*. Issued credentials are only valid for a given epoch and must be reissued as the timeline advances. For each epoch, a user requests a revocation handle that binds their public key to the epoch. When presenting a credential, the user also provides a proof of non-revocation. Users verifiably encrypt their public key under an authorised auditor's public key to enable transaction auditing.

6.4.4 Where Hyperledger Fabric's Privacy Stands

Although there have been improvements to the Idemix system over the years, some functionalities remain limited. For example, (1) there is still no notion of privacy for endorsers. (2) Delegatable credentials require proving knowledge of a list of keys. (3) The root authority is still a single point of failure. (4) Selective disclosure of attributes requires computation linear in the size of all the attributes encoded in the credential. (5) Many zero-knowledge proofs need to be generated for each transaction. (6) Many pairings need to be computed for verification. A number of these shortcomings arise because the underlying credential scheme is based on delegatable credentials.

6.5 Protego

We argue that changing some of the underlying building blocks is necessary to build an ABC scheme that overcomes the inherent limitations of Idemix and its extension. For this reason, we take the framework from Chapter 5, including the revocation extension originally proposed in [DHS15a], as our starting point. Below, we walk through the different building blocks and build an argument for how and why these components yield greater functionality and efficiency for a credential system in the permissioned blockchain setting.

SCDS. Using commitment schemes that allow to commit to *sets* of attributes enables constant-size openings of subsets (selective disclosure) of the committed sets. These schemes support commitment randomisation without the need to rely on zero-knowledge proofs of correct randomisation, as the corresponding witness for openings can be adapted accordingly with respect to the randomisation of the committed set. The set-commitment scheme presented in Chapter 5 is particularly useful in the permissioned blockchain setting, *e.g.,* to model access control policies. Furthermore, in the case of Fabric, the use of proof of exponentiations to outsource some of the computational cost from the verifier to the prover is an interesting feature considering endorsements. It makes the endorser's verification faster when validating a transaction proposal.

Mercurial Signatures. The introduction of SPS-EQ in [FHS19] allowed to adapt a signature on a representative message to a signature on a different representative (in a given equivalence class) without knowledge of the secret key. If the adapted signature is indistinguishable from a fresh signature on a random message, the scheme satisfies the notion of *perfect adaption*. This, together with the randomisability of the set-commitment scheme, allows to consistently and efficiently update the signature of a credential, bypassing the need to generate and keep account of pseudonyms and NIZK proofs that are required in all previous works based on Idemix. Our approach here is to use mercurial signatures to also randomise the correspondng public keys while consistently adapting the signatures.

ABC model. We can rely on the ROM and apply the Fiat-Shamir transform to the ABC construction from Chapter 5 (the showing protocol is a three move public coin one). However, in the previous ABC, interaction is required in the showing protocol to provide freshness (*i.e.,* to avoid replay attacks). We require the user to send the transaction proposal during the first move to overcome this issue. Thus, applying the Fiat-Shamir transform to the first move bounds the credential showing to that particular transaction so that it cannot be replayed.

In the following, we present our ABC scheme *Protego* introducing the syntax first. Subsequently, we elaborate on the revocation, auditing and issuer-hiding approaches. Finally, we discuss our construction and the integration with Fabric.

6.5.1 Syntax

ABC Syntax. An ABC consists of the following p.p.t algorithms:

$\mathsf{Setup}(1^\lambda, \mathsf{aux})$ takes a security parameter λ and some optional auxiliary information aux (which may fix a universe of attributes, attribute values and other parameters) and outputs public parameters pp, discarding any trapdoor.

$\mathsf{TSetup}(1^\lambda, \mathsf{aux})$ is like Setup but it also returns a trapdoor τ (if any).

$\mathsf{OKGen}(\mathsf{pp})$ takes pp and outputs an organisation key pair $(\mathsf{osk}, \mathsf{opk})$.

$\mathsf{UKGen}(\mathsf{pp})$ takes pp and outputs a user key pair $(\mathsf{usk}, \mathsf{upk})$.

$\mathsf{AAKGen}(\mathsf{pp})$ takes pp and outputs an auditor key pair $(\mathsf{ask}, \mathsf{apk})$.

$\mathsf{RAKGen}(\mathsf{pp})$ takes pp and outputs a revocation key pair $(\mathsf{rsk}, \mathsf{rpk})$.

$\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym})$ and $\mathsf{Issue}(\mathsf{pp},\mathsf{upk},\mathsf{osk},\mathsf{apk},\mathcal{X},\mathsf{nym})$ are run by a user and the organisation respectively, who interact during execution. Obtain takes pp, the user's secret key usk, an organisation's public key opk, an auditor's public key apk, an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < q$, and a pseudonym nym used for revocation. Issue takes pp, a public key upk, a secret key osk, an auditor's public key apk, an attribute set $\mathcal{X}$ of size $|\mathcal{X}| < q$, and a pseudonym nym. At the end of this protocol, Obtain outputs a credential cred on $\mathcal{X}$ for the user or $\perp$ if the execution failed.

$\mathsf{Show}(\mathsf{pp}, \mathsf{opk}, \mathsf{upk}, \mathsf{usk}, \mathsf{cred}, \mathcal{X}, \mathcal{S}, \mathcal{D}, \mathsf{aux})$ takes pp, a public key opk, a key pair $(\mathsf{usk}, \mathsf{upk})$, a credential cred for the attribute set $\mathcal{X}$, potentially non-empty sets $\mathcal{S} \subseteq \mathcal{X}$, $\mathcal{D} \nsubseteq \mathcal{X}$ representing attributes sets being a subset ($\mathcal{S}$) or disjoint ($\mathcal{D}$) to the attribute set ($\mathcal{X}$) committed in the credential, and auxiliary information aux. It outputs a proof π.

$\mathsf{Verify}(\mathsf{pp}, \mathsf{opk}, \mathcal{X}, \mathcal{S}, \mathcal{D}, \pi, \mathsf{aux})$ takes pp, the (potentially empty) sets $\mathcal{S}$ and $\mathcal{D}$, a proof π and auxiliary information aux. It outputs 1 or 0 indicating whether the credential showing proof π was accepted or not.

$\mathsf{RSetup}(\mathsf{pp}, (\mathsf{rsk}, \mathsf{rpk}), \mathsf{NYM}, \mathsf{RNYM})$ takes pp, a revocation key pair $(\mathsf{rsk}, \mathsf{rpk})$ and two disjoint lists NYM and RNYM (holding valid and revoked pseudonyms). It outputs auxiliary information $\mathsf{aux}_{\mathsf{rev}}$ for the revocation authority and revocation information $\mathbb{R} = (\mathbb{R}_V, \mathbb{R}_S)$. $\mathbb{R}_V$ is needed for verifying the revocation status and $\mathbb{R}_S$ is a list holding the revocation information per nym.

$\mathsf{Revoke}(\mathsf{pp}, (\mathsf{rsk}, \mathsf{rpk}), \mathsf{aux}_{\mathsf{rev}}, \mathbb{R}, b)$ takes pp, $(\mathsf{rsk}, \mathsf{rpk})$, $\mathsf{aux}_{\mathsf{rev}}$, $\mathbb{R}$ and a bit b indicating revoked/unrevoked. It outputs information $\mathbb{R}'$ and $\mathsf{aux}'_{\mathsf{rev}}$.

$\mathsf{AuditEnc}(\mathsf{upk}, \mathsf{apk})$ takes upk and apk. It outputs an encryption enc of upk under apk and auxiliary information α.

$\mathsf{AuditDec}(\mathsf{enc}, \mathsf{ask})$ takes enc and ask. It outputs a decryption of enc using ask.

$\mathsf{AuditPrv}(\mathsf{enc}, \alpha, \mathsf{usk}, \mathsf{apk})$ takes enc, α, usk, and apk. It generates a proof for enc being the encryption of upk under apk and outputs a proof π.

$\mathsf{AuditVerify}(\mathsf{apk}, \pi)$ takes apk and a proof π for the correct encryption of a user's public key under apk and outputs 1 if and only if the proof verifies.

6.5.2 Security Properties

In the following, we adapt the ABC security models from Chapter 5 and [DHS15a] to consider non-interactive showings as well as auditability. We denote by Tx the universe of transactions tx represented as bitstrings. Since transactions are passed to the verification algorithm we do not consider replay attacks as in the previous models. In other words, given that a showing corresponds to a specific transaction, we do not consider replay attacks for the same transaction as such attacks are trivially detected.

We consider a single revocation, issuing and auditing authority. Extension to the multi-issuing and multi-auditing authorities is straightforward as the corresponding keys can be generated independently. For revocation authorities, one needs to take into account the existence of multiple revocation accumulators and thus adapt the scheme accordingly. Like the issuer-hiding property, auditability is considered independently as it can be seen as an extension to the underlying scheme.

Before presenting the oracles and formal definitions, we introduce the following auxiliary lists, sets and global variables. For example, N represents the set of all pseudonyms nym while the sets NYM and RNYM represent the subsets of unrevoked and revoked pseudonyms respectively. Therefore, we have that $\mathsf{NYM} \cap \mathsf{RNYM} = \emptyset \ \wedge \ \mathsf{NYM} \cup \mathsf{RNYM} = \mathsf{N}$. $\mathtt{NYM}$, $\mathtt{HU}$ and $\mathtt{CU}$ are lists that keep track of which nym is assigned to which user. The global variables RI and $\mathsf{NYM}_{\mathsf{LoR}}$ (initially set to $\perp$) store the revocation information $(\mathbb{R}_S, \mathbb{R}_V)$ and the pseudonyms used in $\mathcal{O}_{\mathsf{LoR}}$ respectively. The oracles are defined as follows:

$\mathcal{O}_{\mathtt{HU}}(i)$ takes as input a user identity i. If $i \in \mathtt{HU} \cup \mathtt{CU}$, it returns $\perp$. Otherwise, it creates a new honest user i by running $(\mathtt{USK}[i], \mathtt{UPK}[i]) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{opk})$, adding i to the honest user list $\mathtt{HU}$ and returning $\mathtt{UPK}[i]$.

$\mathcal{O}_{\texttt{CU}}(i, \mathsf{upk})$ takes as input a user identity i and (optionally) a user public key upk; if user i does not exist, a new corrupt user with public key upk is registered, while if i is honest, its secret key and all credentials are leaked. In particular, if $i \in \texttt{CU}$, $i \in I_{\mathsf{LoR}}$ (that is, i is a challenge user in the anonymity game) or if $\mathsf{NYM}_{\mathsf{LoR}} \cap \mathsf{N}[i] \neq \emptyset$ then the oracle returns $\perp$. If $i \in \texttt{HU}$ then the oracle removes i from $\texttt{HU}$ and adds it to $\texttt{CU}$; it returns $\texttt{USK}[i]$ and $\texttt{CRED}[j]$ for all j with $\texttt{OWNR}[j] = i$. Otherwise (*i.e.,* $i \notin \texttt{HU} \cup \texttt{CU}$), it adds i to $\texttt{CU}$ and sets $\texttt{UPK}[i] \leftarrow \mathsf{upk}$.

$\mathcal{O}_{\texttt{RN}}(\mathsf{rsk}, \mathsf{rpk}, \mathsf{REV})$ takes as input the revocation secret key rsk, the revocation public key rpk and a list REV of pseudonyms to be revoked. If $\mathsf{REV} \cap \mathsf{RNYM} \neq \emptyset$ or $\mathsf{REV} \not\subseteq \mathsf{N}$ return $\perp$. Otherwise, set $\mathsf{RNYM} \leftarrow \mathsf{RNYM} \cup \mathsf{REV}$ and $\mathsf{RI} \leftarrow \mathsf{Revoke}(\mathsf{pp}, (\mathsf{rsk}, \mathsf{rpk}), \mathsf{RNYM}, \mathsf{RI}, 1)$.

$\mathcal{O}_{\mathsf{ObtIss}}(i, \mathcal{X})$ takes as input a user identity i, a pseudonym nym and a set of attributes $\mathcal{X}$. If $i \notin \texttt{HU}$ or $\exists\ j : \texttt{NYM}[j] = \mathsf{nym}$, it returns $\perp$. Otherwise, it issues a credential to i by running

$$(\mathsf{cred}, \top) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \texttt{USK}[i], \mathsf{opk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}),$$
$$\mathsf{Issue}(\mathsf{pp}, \texttt{UPK}[i], \mathsf{osk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}).$$

If $\mathsf{cred} = \perp$, it returns $\perp$. Else, it appends $(i, \mathsf{cred}, \mathcal{X}, \mathsf{nym})$ to $(\texttt{OWNR}, \texttt{CRED}, \texttt{ATTR}, \texttt{NYM})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Obtain}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious organisation, issue a credential to an honest user. It takes as input a user identity i, a pseudonym nym and a set of attributes $\mathcal{X}$. If $i \notin \texttt{HU}$, it returns $\perp$. Otherwise, it runs $(\mathsf{cred}, \cdot) \xleftarrow{\$} \mathsf{Obtain}(\mathsf{pp}, \texttt{USK}[i], \mathsf{opk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}), \cdot)$, where the Issue part is executed by $\mathcal{A}$. If $\mathsf{cred} = \perp$, it returns $\perp$. Else, it appends $(i, \mathsf{cred}, \mathcal{X}, \mathsf{nym})$ to $(\texttt{OWNR}, \texttt{CRED}, \texttt{ATTR}, \texttt{NYM})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Issue}}(i, \mathcal{X})$ lets the adversary $\mathcal{A}$, who impersonates a malicious user, obtain a credential from an honest organisation. It takes as input a user identity i, a pseudonym nym and a set of attributes $\mathcal{X}$. If $i \notin \texttt{CU}$, it returns $\perp$. Otherwise, it runs $(\cdot, I) \xleftarrow{\$} (\cdot, \mathsf{Issue}(\mathsf{pp}, \texttt{UPK}[i], \mathsf{osk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}))$, where the Obtain part is executed by $\mathcal{A}$. If $\mathrm{I} = \perp$, it returns $\perp$. Else, it appends $(i, \perp, \mathcal{X}, \mathsf{nym})$ to $(\texttt{OWNR}, \texttt{CRED}, \texttt{ATTR}, \texttt{NYM})$ and returns $\top$.

$\mathcal{O}_{\mathsf{Show}}(j, \mathcal{S}, \mathcal{D})$ lets the adversary $\mathcal{A}$ play a dishonest verifier during a showing by an honest user. It takes as input an index of an issuance j and attributes sets $\mathcal{S}$ and $\mathcal{D}$. Let $i \xleftarrow{\$} \texttt{OWNR}[j]$. If $i \notin \texttt{HU}$, it returns $\perp$. Otherwise, it runs

$$(S, \cdot) \xleftarrow{\$} \mathsf{Show}(\mathsf{pp}, \texttt{USK}[i], \texttt{UPK}[i], \mathsf{opk}, \texttt{ATTR}[j],$$
$$\mathcal{S}, \mathcal{D}, \texttt{CRED}[j], \mathsf{RI}, \mathsf{apk}, \mathsf{tx}), \cdot)$$

where the Verify part is executed by $\mathcal{A}$.

$\mathcal{O}_{\mathsf{LoR}}(j_0, j_1, \mathcal{S}, \mathcal{D})$ is the challenge oracle in the anonymity game where $\mathcal{A}$ must distinguish (multiple) showings of two credentials $\texttt{CRED}[j_0]$ and $\texttt{CRED}[j_1]$. The oracle takes two issuance indices j_0 and j_1 and attribute sets $\mathcal{S}$ and $\mathcal{D}$. If $J_{\mathsf{LoR}} \neq \emptyset$ and $J_{\mathsf{LoR}} \neq \{j_0, j_1\}$, it returns $\perp$. Let $i_0 \xleftarrow{\$} \texttt{OWNR}[j_0]$ and $i_1 \xleftarrow{\$} \texttt{OWNR}[j_1]$. If $J_{\mathsf{LoR}} \neq \emptyset$ then it sets $J_{\mathsf{LoR}} \xleftarrow{\$} \{j_0, j_1\}$ and $I_{\mathsf{LoR}} \xleftarrow{\$} \{i_0, i_1\}$. If $i_0, i_1 \neq \texttt{HU} \vee \mathsf{N}[i_0] = \perp \vee \mathsf{N}[i_1] = \perp \vee \mathsf{N}[i_0] \in \mathsf{RNYM} \vee \mathsf{N}[i_1] \in \mathsf{RNYM} \vee$

$\mathcal{S} \not\subseteq \mathtt{ATTR}[j_0] \cap \mathtt{ATTR}[j_1] \vee \mathcal{D} \cap \{\mathtt{ATTR}[j_0] \cup \mathtt{ATTR}[j_1]\} \neq \emptyset$, it returns $\perp$. Else, it adds $\mathsf{N}[i_b]$ to $\mathsf{NYM}_{\mathsf{LoR}}$ and runs

$$(S, \cdot) \xleftarrow{\$} (\mathsf{Show}(\mathsf{pp}, \mathtt{USK}[j_b], \mathtt{UPK}[j_b], \mathsf{opk}, \mathtt{ATTR}[j_b], \mathcal{S}, \mathcal{D}, \mathtt{CRED}[j_b], \mathsf{RI}, \mathsf{apk}, \mathsf{tx}), \cdot)$$

(with b set by the experiment) where the Verify part is executed by $\mathcal{A}$.

Intuitively, *correctness* requires that a credential showing with respect to a non-empty sets $\mathcal{S}$ and $\mathcal{D}$ of attributes always verifies if it was issued honestly on some attribute set $\mathcal{X}$ with $\mathcal{S} \subset \mathcal{X}$ and $\mathcal{D} \cap \mathcal{X} \neq \emptyset$.

Correctness. An ABC system is correct if $\forall\ \lambda > 0$, $\forall\ q, q' > 0$, $\forall\ \mathcal{X} : 0 < |\mathcal{X}| \leq q$, $\forall\ \emptyset \neq \mathcal{S} \subset \mathcal{X}$, $\forall\ \emptyset \neq \mathcal{D} \not\subseteq \mathcal{X} : 0 < |\mathcal{D}| \leq q$, $\forall\ \mathsf{NYM}, \mathsf{RNYM} \subseteq \mathsf{N} : 0 < |\mathsf{N}| \leq q' \wedge \mathsf{NYM} \cap \mathsf{RNYM} = \emptyset$, $\forall\ \mathsf{nym} \in \mathsf{NYM}$, $\forall\ \mathsf{nym}' \in \mathsf{RNYM}$ it holds that:

$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, (1^q, 1^{q'}))$; $(\mathsf{rsk}, \mathsf{rpk}) \xleftarrow{\$} \mathsf{RAKGen}(\mathsf{pp})$; $(\mathsf{ask}, \mathsf{apk}) \xleftarrow{\$} \mathsf{AAKGen}(\mathsf{pp})$; $(\mathbb{R}, \mathsf{aux}_{\mathsf{rev}}) \leftarrow \mathsf{RSetup}(\mathsf{pp}, \mathsf{rpk}, \mathsf{NYM}, \mathsf{RNYM})$; $(\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp})$; $(\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{pp})$; $(\mathsf{cred}, \top) \xleftarrow{\$} (\mathsf{Obtain}(\mathsf{pp}, \mathsf{usk}, \mathsf{opk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}), \mathsf{Issue}(\mathsf{pp}, \mathsf{upk}, \mathsf{osk}, \mathsf{apk}, \mathcal{X}, \mathsf{nym}))$; $(\mathbb{R}_S, \mathbb{R}_V) \leftarrow \mathsf{Revoke}(\mathsf{pp}, \mathbb{R}, \mathsf{aux}_{\mathsf{rev}}, \mathsf{nym}', 1)$; $\Omega \leftarrow \mathsf{Show}(\mathsf{pp}, \mathsf{usk}, \mathsf{upk}, \mathsf{opk}, \mathsf{cred}, \mathcal{S}, \mathcal{D}, \mathbb{R}, \mathsf{apk}, \mathsf{tx})$; $1 \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \mathsf{opk}, \mathbb{R}_V, \mathsf{rpk}, \mathsf{apk}, \mathsf{tx}, \Omega)$

Unforgeability and anonymity are defined as in the previous constructions based on SPS-EQ, but considering the revocation and auditing authorities.

Definition 39: Unforgeability

An ABC scheme is unforgeable, if $\forall\ \lambda, q, q' > 0$ and p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathtt{CU}}, \mathcal{O}_{\mathtt{RN}}, \mathcal{O}_{\mathsf{ObtIss}}, \mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Show}}\}$ the following probability is negligible.

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, (1^q, 1^{q'})); (\mathsf{rsk}, \mathsf{rpk}) \xleftarrow{\$} \mathsf{RAKGen}(\mathsf{pp}); \\ (\mathsf{ask}, \mathsf{apk}) \xleftarrow{\$} \mathsf{AAKGen}(\mathsf{pp}); (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp}); \\ (\mathcal{S}, \mathcal{D}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{opk}, \mathsf{rpk}, \mathsf{apk}); \\ (\cdot, b^*) \xleftarrow{\$} (\mathcal{A}(\mathsf{st}), \mathsf{Verify}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \mathsf{opk}, \mathsf{rpk}, \mathsf{apk}, \mathsf{RI}, \mathsf{tx}, \Omega)) : \\ b^* = 1 \wedge \forall\ j : \mathtt{OWNR}[j] \in \mathtt{CU} \implies \\ (\mathsf{N}[j] = \perp\ \vee\ (\mathsf{N}[j] \neq \perp\ \wedge (\mathcal{S} \not\subseteq \mathtt{ATTR}[j]\ \vee \\ \mathcal{D} \cap \mathtt{ATTR}[j] \neq \emptyset \vee \mathsf{N}[j] \in \mathsf{RNYM})) \end{array}\right]$$

Definition 40: Anonymity

An ABC scheme is anonymous, if $\forall\ \lambda, q, q' > 0$ and all p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O} := \{\mathcal{O}_{\mathtt{HU}}, \mathcal{O}_{\mathtt{CU}}, \mathcal{O}_{\mathtt{RN}}, \mathcal{O}_{\mathsf{Obtain}}, \mathcal{O}_{\mathsf{Show}}, \mathcal{O}_{\mathsf{LoR}}\}$ the following probability is negligible.

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, (1^q, 1^{q'})); (\mathsf{ask}, \mathsf{apk}) \xleftarrow{\$} \mathsf{AAKGen}(\mathsf{pp}); \\ b \xleftarrow{\$} \{0, 1\}; (\mathsf{opk}, \mathsf{rpk}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}); b^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{st}) \end{array} : b^* = b\right] - \frac{1}{2}$$

Finally, auditability requires that the auditor can recover the user's public key from an accepted showing proof.

Definition 41: Auditability

An ABC scheme is auditable, if $\forall\ \lambda, q, q' > 0, \mathsf{tx}, \mathsf{nym} \in \mathsf{NYM}, \mathbb{R}, \mathbb{R}_V$ and all p.p.t adversaries $\mathcal{A}$ having oracle access to $\mathcal{O}_{\mathsf{Issue}}$, the following probability is negligible.

$$\Pr\left[\begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda, (1^q, 1^{q'})); (\mathsf{rsk}, \mathsf{rpk}) \xleftarrow{\$} \mathsf{RAKGen}(\mathsf{pp}); \\ (\mathsf{ask}, \mathsf{apk}) \xleftarrow{\$} \mathsf{AAKGen}(\mathsf{pp}); (\mathsf{osk}, \mathsf{opk}) \xleftarrow{\$} \mathsf{OKGen}(\mathsf{pp}); \\ (\mathsf{usk}, \mathsf{upk}) \xleftarrow{\$} \mathsf{UKGen}(\mathsf{pp}); \\ (\mathcal{S}, \mathcal{D}, \mathsf{enc}, \Omega, \mathsf{st}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{opk}, \mathsf{rpk}, \mathsf{apk}, \mathsf{usk}, \mathsf{upk}, \mathsf{nym}); \\ (\cdot, b^*) \xleftarrow{\$} (\mathcal{A}(\mathsf{st}), \mathsf{Verify}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \mathsf{opk}, \mathbb{R}_V, \mathsf{rpk}, \mathsf{apk}, \mathsf{RI}, \mathsf{tx}, \Omega)) : \\ b^* = 1 \land \mathsf{upk} \neq \mathsf{AuditDec}(\mathsf{enc}, \mathsf{ask}) \end{array}\right]$$

6.5.3 Construction

We present our construction concerning the main building blocks.

Revocation. The revocation system from [DHS15a] defines a revocation authority responsible for managing an allow and a deny list of revocation handlers. The authority publishes an accumulator RevAcc representing the deny list, and maintains a public list of non-membership witnesses for unrevoked users. During the issuing protocol, users are given a revocation handle that is encoded in the credential. To prove that they are not revoked during a showing, the user consistently randomises its credential with the accumulator and the corresponding non-membership witness. Then, the verifier checks that the (randomised) witness is valid for the revocation handle (encoded in the user credential), and with respect to the (randomised) accumulator. To work, the user must compute a ZKPoK on the correct randomisation of the non-membership witness and the accumulator. As explained in [DHS15a], the revocation handle encoded in the user's credential is of the form $\mathsf{usk}_2(b + \mathsf{nym})P_1$, where usk_2 is an additional user secret key required for anonymity and nym is the pseudonym used for revocation. For this reason, users are required to manage augmented keys of the form $\mathsf{upk} = (\mathsf{upk}_1, \mathsf{upk}_2)$, $\mathsf{usk} = (\mathsf{usk}_1, \mathsf{usk}_2)$. Furthermore, for technical reasons, another component, $\mathsf{usk}_2 Q$, where Q is a random element in $\mathbb{G}_1$ with unknown discrete logarithm, must be included in the credential.

Auditability. A credential in [FHS19, CLPK22] and [DHS15a] contains a tuple (C, rC, P_1) where C is the set commitment on the user attributes, r is a random value used for technical purposes and P_1 is used to compute a ZKPoK of the randomiser μ in $(\mu C, \mu rC, \mu P_1)$ during a showing. We borrow the idea of using a verifiable variant of ElGamal from [BDCET21] to prove the well-formedness of a ciphertext (encrypting the user's public key) with respect to the auditor's key. Therefore, we add the user's public key upk_1 and the auditor's public key apk as the credentials' sixth and seventh components. Thus, we now have revocable credentials of the form $(C, rC, P_1, \mathsf{usk}_2(b+\mathsf{nym})P_1, \mathsf{usk}_2 Q, \mathsf{upk}_1, \mathsf{apk})$, which can be randomised to obtain a tuple (μC, $\mu r C$, μP_1, $\mu\mathsf{usk}_2(b+\mathsf{nym})P_1$, $\mu\mathsf{usk}_2 Q$, $\mu\mathsf{usk}_1 P_1$, $\mu\mathsf{apk}$). We exploit this fact to allow the user to generate an *audit* proof that can be publicly verified without leaking information about the user's public key. This way, verifiers can check a proof using the sixth and seventh components in the credential to be sure that (1) the user encrypted a public key for which it has the corresponding secret key, and (2) using the correct one. Since the issuing authority signs the credential, the randomisation needs to be consistent. Modifications required to implement our auditability approach are as follows:

1. The user randomises its credential as usual to obtain a new one of the form (C'_1, C'_2, C'_3, C'_4, C'_5, C'_6, C'_7)=(μC_1, μC_2, μP_1, μC_4, μC_5, $\mu\mathsf{upk}_1$, $\mu\mathsf{apk}$). Since only the user knows the randomiser μ, its public key remains hidden.
2. The user picks $\alpha \in \mathbb{Z}_p$ and encrypts its own public key using ElGamal encryption with auditor's public key apk and randomness α to obtain a ciphertext $\mathsf{enc} = (\mathsf{enc}_1, \mathsf{enc}_2) = (\mathsf{upk}_1 + \alpha\mathsf{apk}, \alpha P_1)$.
3. The user runs the algorithm $\mathsf{AuditPrv}$ (Figure 6.2) with input $(\mathsf{enc}, \alpha, \mathsf{usk}_1, \mathsf{apk})$ to obtain c, z_1 and z_2.
4. Then, the user picks $\beta \leftarrow \mathbb{Z}_p$, computes $t_1 = \beta P_2$, $t_2 = \beta\mu P_2$, $t_3 = \alpha\beta P_2$ and sends $(\mathsf{enc}, c, z_1, z_2, t_1, t_2, t_3)$ to the verifier alongside the randomised credential from step 1.
5. The verifier checks the well-formedness of the ElGamal encryption pair running the algorithm $\mathsf{AuditVerify}$ (Figure 6.2) with input $(c, \mathsf{enc}, z_1, z_2)$. If the check succeeds, it checks the following pairing equations to verify that the encrypted public key is the one in the credential:
$$e(\mathsf{enc}_2, t_2){=}e(C'_3, t_3) \wedge e(\mathsf{enc}_2, t_1){=}e(P_1, t_3) \wedge e(\mathsf{enc}_1, t_2){=}e(C'_6, t_1){+}e(C'_7, t_3)$$

Observe that the verifier knows $\mu P_1 = C'_3$, $\mu\mathsf{usk}_1 P_1 = C'_6$, $\mu\mathsf{ask}P_1 = C'_7$, $(\mathsf{usk}_1 + \alpha\mathsf{ask})P_1 = \mathsf{enc}_1$, $\alpha P_1 = \mathsf{enc}_2$, $\beta P_2 = t_1$, $\beta\mu P_2 = t_2$ and $\alpha\beta P_2 = t_3$. With β the user is able to randomise the other values so that the pairing equation can be checked to verify the relation between the ElGamal ciphertext and the randomised public key in C'_6, without leaking information about the user's public key. Furthermore, the first two pairing equations verify the well-formedness of t_1, t_2 and t_3 with respect to the user's credential and the ciphertext. Hence, the verifier will not be able to recover the user's public key nor the user cheat.

Our solution only adds two elements to the credential while requiring the user to send two more elements in $\mathbb{G}_1$, three in $\mathbb{Z}_p$ and three in $\mathbb{G}_2$, for a total of eight. Furthermore, computational cost remains low as it just involves the computation of seven pairings, the ElGamal encryption and two Schnorr proofs [Sch90].

Issuer-hiding. In permissioned blockchains where there are multiple organisations that issue credentials, the issuer-hiding strategy discussed in Chapter 5 can be used. This would allow users holding valid signatures to pick any subset of issuer's public keys to generate a proof. In this regard, we adapt the proof system from Chapter 5 to the signature used and make it malleable so that users can compute the proof once and then adapt it during showings with little computational cost.

In Figure 6.1, we build a fully adaptive malleable NIZK argument following the construction from Chapter 5. The main idea is that given two proofs, π_1 and π_2,

```
$\underline{\mathsf{SH.PGen}(1^\lambda):}$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda);\ z \xleftarrow{\$} \mathbb{Z}_p$
$\mathbf{return}\ (\mathsf{BG}, [z]_1)$

$\underline{\mathsf{SH.PSim}(\mathsf{crs}, \tau, (\mathbf{v}_i)_{i\in[n]}, [\mathbf{x}_1]_2, [\mathbf{x}_2]_2):}$
$\delta, z_1, ..., z_{n-1} \xleftarrow{\$} \mathbb{Z}_p^*$
$z_n \leftarrow \delta\tau - \sum_{i=1}^{i=n-1} z_j$
$\mathbf{for\ all\ i} \in [n]\ \mathbf{do}$
    $d_i \xleftarrow{\$} \mathbb{Z}_p;\ [\mathbf{a}_i]_2 \leftarrow d_i \cdot \mathbf{v}_i - z_i \cdot \mathbf{x}$
$\mathbf{return}\ (([\mathbf{a}_n]_2, [d_n]_1, [z_n]_1)_{n\in[n]}, \delta P_2)$

$\underline{\mathsf{SH.ZKEval}(\mathsf{crs}, [\mathbf{x}_1]_2, [\mathbf{x}_2]_2, \pi; \alpha, \beta):}$
$//\ [\mathbf{x}']_2 = (\alpha w_1 + \beta w_2)[\mathbf{v}_i]_2$
$(([\mathbf{a}_n^j]_2, [d_n^j]_1, [z_n]_1)_{n\in[n]}^{j\in[2]}, Z_2) \leftarrow \pi$
$\delta \xleftarrow{\$} \mathbb{Z}_p^*;\ Z_2' \leftarrow \delta Z_2$
$\mathbf{for\ all}\ i \in [n]\ \mathbf{do}$
    $[z_i']_1 \leftarrow \delta[z_i]_1;$
    $[d_i']_2 \leftarrow \delta\alpha[d_i^1]_2 + \delta\beta[d_i^2]_2;$
    $[\mathbf{a}_i']_2 \leftarrow \delta\alpha[\mathbf{a}_i^1]_2 + \delta\beta[\mathbf{a}_i^2]_2;$
$\mathbf{return}\ (([\mathbf{a}_n']_2, [d_n']_1, [z_n']_1)_{n\in[n]}, Z_2')$
```

```
$\underline{\mathsf{SH.PTGen}(1^\lambda):}$
$\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda);\ z \xleftarrow{\$} \mathbb{Z}_p;\ \tau \leftarrow z$
$\mathbf{return}\ (\mathsf{BG}, [z]_1, \tau)$

$\underline{\mathsf{SH.PPro}(\mathsf{crs}, ([\mathbf{v}_i]_2)_{i\in[n]}, ([\mathbf{x}_j]_2, w_j)_{j\in[2]}):}$
$//\ [\mathbf{x}_1]_2 = w_1[\mathbf{v}_i]_2,\ [\mathbf{x}_2]_2 = w_2[\mathbf{v}_i]_2$
$\delta, r_1, r_2, z_1, ..., z_{n-1} \xleftarrow{\$} \mathbb{Z}_p^*$
$[z_n]_1 \leftarrow \delta[z]_1 - \sum_{i=1}^{i=n-1}[z_i]_1$
$([\mathbf{a}_i^j]_2, [d_i^j]_1) \leftarrow (r_j[\mathbf{v}_i]_2, w_j[z_i]_1 + [r_j]_1)$
$\mathbf{for\ all}\ k \neq i \in [n],\ j \in [2]\ \mathbf{do}$
    $d_k^j \xleftarrow{\$} \mathbb{Z}_p;\ [\mathbf{a}_k^j]_2 \leftarrow d_k^j[\mathbf{v}_k]_2 - z_k[\mathbf{x}_j]_2$
$\mathbf{return}\ (([\mathbf{a}_n^j]_2, [d_n^j]_1, [z_n]_1)_{n\in[n]}^{j\in[2]}, \delta P_2)$

$\underline{\mathsf{SH.PVer}(\mathsf{crs}, ([\mathbf{v}_i]_2)_{i\in[n]}, [\mathbf{x}]_2, \pi):}$
$(([\mathbf{a}_n]_2, [d_n]_1, [z_n]_1)_{n\in[n]}, Z_2) \leftarrow \pi$
$\mathbf{check}\ e([z]_1, Z_2) = e(\sum_{i=1}^{i=n}[z_i]_1, [1]_2)$
$\mathbf{for\ all\ i} \in [n]\ \ \mathbf{do}$
 $\mathbf{check}\ e([d_i]_1, [\mathbf{v}_i]_2) = e([z_i]_1, [\mathbf{x}]_2) + e([1]_1, [\mathbf{a}_i]_2)$
$\mathbf{return}\ 1$
```

Figure 6.1: Our fully adaptive malleable NIZK argument

for statements $\mathbf{x}_1 = w_1\mathbf{v}_i$ and $\mathbf{x}_2 = w_2\mathbf{v}_i$; one can compute a valid proof π for the statement $\mathbf{x} = (\alpha w_1 + \beta w_2)\mathbf{v}_i$ with fresh α and β. The derivation privacy property of the proof system ensures that π looks like a freshly computed proof.

Theorem 25. The proof system given in Figure 6.1 is a fully adaptive malleable NIZK argument for the language $\mathcal{L}_{\bigvee(\mathbf{v}_i)_{i\in[n]}}$, defined as:

$$\mathcal{L}_{\bigvee(\mathbf{v}_i)_{i\in[n]}} = \{(\mathbf{v}_i) \in \mathbb{G}_2^\ell |\ \exists\ w \in \mathbb{Z}_p^* : \vee\ (\mathbf{v}_i' = w\mathbf{v}_i)_{i\in[n]}\}$$

The proof of Theorem 25 follows from the one given in Theorem 15, but considering a 1-out-of-n OR-Proof as in Theorem 21.

Now we elaborate on a second strategy based on the approach from [BEK⁺21]. We recall that an issuer-policy in [BEK⁺21] is a set $\{(\sigma_i, \mathsf{opk}_i)_{i\in[n]}\}$ of signatures on issuers public keys generated by some verification secret key vsk. To hide the identity of an issuer j, a user consistently randomises the pair $(\sigma_j, \mathsf{opk}_j)$ to obtain a randomised public key opk_j'. It then adapts the signature σ on its credential the same way and presents opk_j' to the verifier. If the verifier accepts the signature σ_j on opk_j' (using vpk), it proceeds to verify σ using opk_j'. Issuer-policies can be specified by the entity that created the smart contract and defined within using the entity's verification key pair. Unlike the first approach, where users choose the issuer's anonymity set, here, it is determined by the policy maker.

We observe that the mercurial signature used in this chapter only provides a weak form of issuer-hiding. Given a signature that has been adapted to verify under a randomised public key pk' in the equivalence class of pk, the owner of

pk can recognize it. Thus, issuers can know which transactions belong to users from their organisations (but not to which particular user) and which ones don't by reading the non-interactive showing proof (it contains the issuer's randomised public key). However, we argue that in the permissioned blockchain setting this provides a fair trade-off as a minimum traceability level is usually required.

Compared to the ABC construction from Chapter 5, we make use of a cryptographic hash function $\mathcal{H}$ to apply the (strong) Fiat-Shamir transform while adding the previously discussed auditability and revocation features. Therefore we

```
$\underline{\mathsf{Setup}(1^\lambda, \mathsf{aux})}$:
$(q, q') \leftarrow \mathsf{aux}$; pick $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$; $Q \xleftarrow{\$} \mathbb{G}_1$; $(\mathsf{rev_{pp}}, \mathsf{rev}_\tau) \xleftarrow{\$} \mathsf{RevAcc.Setup}(1^\lambda, q')$
$(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{scds}_\tau) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$; $(\mathsf{sps_{pp}}, \mathsf{sps}_\tau) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PGen}(1^\lambda; \mathsf{BG})$
return $(\mathcal{H}, \mathsf{BG}, \mathsf{rev_{pp}}, Q, \mathsf{scds_{pp}}, \mathsf{sps_{pp}})$

$\underline{\mathsf{TSetup}(1^\lambda, \mathsf{aux})}$:
$(q, q') \leftarrow \mathsf{aux}$; pick $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$; $Q \xleftarrow{\$} \mathbb{G}_1$; $(\mathsf{rev_{pp}}, \mathsf{rev}_\tau) \xleftarrow{\$} \mathsf{RevAcc.Setup}(1^\lambda, q')$
$(\mathsf{BG}, \mathsf{scds_{pp}}, \mathsf{scds}_\tau) \xleftarrow{\$} \mathsf{SCDS.Setup}(1^\lambda, q)$; $(\mathsf{sps_{pp}}, \mathsf{sps}_\tau) \xleftarrow{\$} \mathsf{SPS\text{-}EQ.PGen}(1^\lambda; \mathsf{BG})$
$\tau = (\mathsf{rev}_\tau, \mathsf{scds}_\tau, \mathsf{sps}_\tau)$; return $(\mathcal{H}, \mathsf{BG}, \mathsf{rev_{pp}}, Q, \mathsf{scds_{pp}}, \mathsf{sps_{pp}}, \tau)$

$\underline{\mathsf{RevAcc.Setup}(1^\lambda, 1^q)}$: $\mathsf{BG} \xleftarrow{\$} \mathsf{BGGen}(1^\lambda)$; $b \xleftarrow{\$} \mathbb{Z}_p^*$; return $(\mathsf{BG}, (b^i P_1, b^i P_2)_{i \in [q]})$

$\underline{\mathsf{AAKGen}(\mathsf{pp})}$: $\mathsf{ask} \xleftarrow{\$} Z_p^*$; $\mathsf{apk} \leftarrow \mathsf{ask} P_1$; return $(\mathsf{ask}, \mathsf{apk})$

$\underline{\mathsf{RAKGen}(\mathsf{pp})}$: $\mathsf{rsk} \xleftarrow{\$} Z_p^*$; $\mathsf{rpk} \leftarrow \mathsf{rsk} P_2$; return $(\mathsf{rpk}, \mathsf{rsk})$

$\underline{\mathsf{OKGen}(\mathsf{pp})}$: return $\mathsf{SPS\text{-}EQ.KGen}(\mathsf{BG}, \mathsf{sps_{pp}}, 7)$

$\underline{\mathsf{UKGen}(\mathsf{pp})}$: $\mathsf{usk}_1, \mathsf{usk}_2 \xleftarrow{\$} Z_p^*$; $(\mathsf{upk}_1, \mathsf{upk}_2) \leftarrow (\mathsf{usk}_1 P_1, \mathsf{usk}_2 P_1)$
return $((\mathsf{usk}_1, \mathsf{usk}_2), (\mathsf{upk}_1, \mathsf{upk}_2))$
```

Figure 6.2: Protego: setup and key generation algorithms.

```
$\underline{\mathsf{AuditEnc}(\mathsf{upk}, \mathsf{apk})}$: $\alpha \leftarrow \mathbb{Z}_p$; $\mathsf{enc} \leftarrow (\mathsf{upk} + \alpha\mathsf{apk}, \alpha P_1)$; return $(\mathsf{enc}, \alpha)$

$\underline{\mathsf{AuditDec}(\mathsf{enc}, \mathsf{ask})}$: $(\mathsf{enc}_1, \mathsf{enc}_2) \leftarrow \mathsf{enc}$; return $(\mathsf{enc}_1 - \mathsf{ask} \cdot \mathsf{enc}_2)$

$\underline{\mathsf{AuditPrv}(\mathsf{enc}, \alpha, \mathsf{usk}, \mathsf{apk})}$:
$r_1, r_2 \leftarrow \mathbb{Z}_p$; $\mathsf{com}_1 \leftarrow r_1 P_1 + r_2 \mathsf{apk}$; $\mathsf{com}_2 \leftarrow r_2 P_1$; $c \leftarrow \mathcal{H}(\mathsf{com}_1, \mathsf{com}_2, \mathsf{enc})$
$z_1 \leftarrow r_1 + c \cdot \mathsf{usk}$; $z_2 \leftarrow r_2 + c \cdot \alpha$; return $(c, z_1, z_2)$

$\underline{\mathsf{AuditVerify}(\mathsf{apk}, c, \mathsf{enc}, z_1, z_2)}$:
$\mathsf{com}_1 \leftarrow z_1 P_1 + z_2 \mathsf{apk} - c\mathsf{enc}_1$; $\mathsf{com}_2 \leftarrow z_2 P_1 - c\mathsf{enc}_2$; $c' \leftarrow \mathcal{H}(\mathsf{com}_1, \mathsf{com}_2, \mathsf{enc})$
return $c' = c$
```

Figure 6.3: Protego: Auditing algorithms.

```
$\underline{\mathsf{RSetup}(\mathsf{pp}, (\mathsf{rsk}, \mathsf{rpk}), \mathsf{NYM}, \mathsf{RNYM}):}$
$(\Pi_{\mathsf{rev}}, \mathsf{aux}_{\mathsf{rev}}) \leftarrow \mathsf{RevAcc.Commit}(\mathsf{rev}_{\mathsf{pp}}, \mathsf{RNYM})$
$\textbf{foreach } \mathsf{nym} \in \mathsf{NYM} \textbf{ do } \mathtt{WIT}[\mathsf{nym}] \leftarrow \mathsf{RevAcc.NonMemWit}(\mathsf{pp}, \Pi_{\mathsf{rev}}, \mathsf{aux}_{\mathsf{rev}}, \mathsf{nym})$
$\textbf{return } ((\Pi_{\mathsf{rev}}, \mathtt{WIT}), \mathsf{aux}_{\mathsf{rev}})$

$\underline{\mathsf{RevAcc.Commit}(\mathsf{pp}, \mathcal{X}; \mathsf{rsk}):}$
$\textbf{check } |\mathcal{X}| \leq q \wedge \nexists\, b' \in \mathcal{X} : b'P_1 = bP_1;\ \Pi_{\mathsf{rev}} \leftarrow \mathsf{rsk}^{-1} \cdot \mathsf{Ch}_{\mathcal{X}}(s)P_1;\ \mathsf{aux}_{\mathsf{rev}} \leftarrow \mathcal{X}$
$\textbf{return } (\Pi_{\mathsf{rev}}, \mathsf{aux}_{\mathsf{rev}})$

$\underline{\mathsf{Revoke}(\mathsf{pp}, \mathbb{R}, \mathsf{aux}_{\mathsf{rev}}, \mathsf{nym}, b):}$
$(\Pi_{\mathsf{rev}}, \mathtt{WIT}) \leftarrow \mathbb{R};\ \mathsf{RNYM} \leftarrow \mathsf{aux}_{\mathsf{rev}}$
$\textbf{if } b = 1$
    $\mathsf{NYM} \leftarrow \mathsf{NYM} \setminus \{\mathsf{nym}\};\ \mathsf{RNYM} \leftarrow \mathsf{RNYM} \cup \{\mathsf{nym}\}$
    $(\Pi'_{\mathsf{rev}}, \mathsf{aux}'_{\mathsf{rev}}) \leftarrow \mathsf{RevAcc.Add}(\mathsf{pp}, \Pi_{\mathsf{rev}}, \mathsf{RNYM}, \mathsf{nym})$
$\textbf{else}$
    $\mathsf{NYM} \leftarrow \mathsf{NYM} \cup \{\mathsf{nym}\}; \mathsf{RNYM} \leftarrow \mathsf{RNYM} \setminus \{\mathsf{nym}\}$
    $(\Pi'_{\mathsf{rev}}, \mathsf{aux}'_{\mathsf{rev}}) \leftarrow \mathsf{RevAcc.Del}(\mathsf{pp}, \Pi_{\mathsf{rev}}, \mathsf{RNYM}, \mathsf{nym})$
$\textbf{foreach } \mathsf{nym}' \in \mathsf{NYM} \textbf{ do } \mathtt{WIT}[\mathsf{nym}'] \leftarrow \mathsf{RevAcc.NonMemWit}(\mathsf{pp}, \Pi'_{\mathsf{rev}}, \mathsf{aux}'_{\mathsf{rev}}, \mathsf{nym}')$
$\textbf{return } ((\Pi'_{\mathsf{rev}}, \mathtt{WIT}), \mathsf{aux}'_{\mathsf{rev}})$

$\underline{\mathsf{RevAcc.Add}(\mathsf{pp}, \mathsf{rsk}, \Pi_{\mathsf{rev}}, \mathsf{aux}_{\mathsf{rev}}, \mathsf{nym}):}$
$\mathcal{X} \leftarrow \mathsf{aux}_{\mathsf{rev}};\ \mathcal{X} \leftarrow \mathcal{X} \cup \{\mathsf{nym}\};\ \textbf{return } \mathsf{RevAcc.Commit}(\mathsf{pp}, \mathcal{X}; \mathsf{rsk})$

$\underline{\mathsf{RevAcc.Del}(\mathsf{pp}, \mathsf{rsk}, \Pi_{\mathsf{rev}}\mathsf{aux}_{\mathsf{rev}}, \mathsf{nym}):}$
$\mathcal{X} \leftarrow \mathsf{aux}_{\mathsf{rev}};\ \mathcal{X} \leftarrow \mathcal{X} \setminus \{\mathsf{nym}\};\ \textbf{return } \mathsf{RevAcc.Commit}(\mathsf{pp}, \mathcal{X}; \mathsf{rsk})$

$\underline{\mathsf{RevAcc.NonMemWit}(\mathsf{pp}, \Pi_{\mathsf{rev}}, \mathsf{aux}_{\mathsf{rev}}, \mathsf{nym}):}$
$\mathcal{X} \leftarrow \mathsf{aux}_{\mathsf{rev}};\ \textbf{check } \mathsf{nym} \notin \mathcal{X};$ Let $q(X)$ and $d \in \mathbb{Z}_p^*$ s.t. $\mathsf{Ch}_{\mathcal{X}}(X) = q(X)(X + \mathsf{nym}) + d$
$\textbf{return } (q(b)P_2, d)$

$\underline{\mathsf{RevAcc.VerifyWit}(\mathsf{pp}, \Pi_{\mathsf{rev}}, \mathsf{nym}, \mathsf{wss}_{\mathsf{rev}}):}$
$(\mathsf{wss}^1_{\mathsf{rev}}, \mathsf{wss}^2_{\mathsf{rev}}) \leftarrow \mathsf{wss}_{\mathsf{rev}};\ \textbf{return } e(\Pi_{\mathsf{rev}}, \mathsf{rpk}) = e((b + \mathsf{nym})P_1, \mathsf{wss}^1_{\mathsf{rev}})e(\mathsf{wss}^2_{\mathsf{rev}}P_1, P_2)$
```

Figure 6.4: Protego: Revocation algorithms.

implement the ZKPoK's as Schnorr proofs (instead of following Remark 1 from [FHS19]).

In Figure 6.2, we present the setup and key generation algorithms. The setup algorithm also takes a bound q' on the maximum number of revoked pseudonyms for the revocation accumulator. Revocation and auditing algorithms are given in figures 6.4 and 6.4, respectively. The revocation authority is responsible for running the Revoke algorithm and updating the accumulator. Obtain and Issue have constant-size communication and are given in Figure 6.5.

We present *Protego* and *Protego Duo* for Show and Verify, depending on the issuer hiding approach. Protego is given in Figure 6.6 and produces a variable-length proof as it relies on the (malleable) NIZK proof. Protego Duo produces a constant-

```
Obtain(pp, usk, opk, apk, $\mathcal{X}$, nym)
$r_1, r_2, r_3 \xleftarrow{\$} \mathbb{Z}_p^*$; $a_1 \leftarrow r_1 P_1$; $a_2 \leftarrow r_2 P_1$
$a_3 \leftarrow r_3 Q$; $C_4 \leftarrow \mathsf{usk}_2(b + \mathsf{nym})P_1$
$C_5 \leftarrow \mathsf{usk}_2 Q$; $e \leftarrow \mathcal{H}(\mathsf{upk}_1, \mathsf{upk}_2, C_5, a_1, a_2, a_3)$
$z_1 \leftarrow r_1 + e \cdot \mathsf{usk}_1$
$z_2 \leftarrow r_2 + e \cdot \mathsf{usk}_2$; $z_3 \leftarrow r_3 + e \cdot \mathsf{usk}_2$
$(C_1, O) \leftarrow \mathsf{SCDS.Commit}(\mathsf{scds_{pp}}, \mathcal{X}; \mathsf{usk}_1)$
$r_4 \xleftarrow{\$} \mathbb{Z}_p^*$; $C_2 \leftarrow r_4 \cdot C_1$
$\Sigma \leftarrow (C_1, C_2, C_4, C_5, (a_i, z_i)_{i \in [3]})$    $\xrightarrow{\Sigma}$

                                   Issue(pp, upk, osk, apk, $\mathcal{X}$, nym)
                                   $e \leftarrow \mathcal{H}(\mathsf{upk}_1, \mathsf{upk}_2, C_5, a_1, a_2, a_3)$
                                   check
                                     $z_1 P_1 = a_1 + e \cdot \mathsf{upk}_1$
                                     $z_2 P_1 = a_2 + e \cdot \mathsf{upk}_2$
                                     $z_3 Q = a_3 + e \cdot C_5$
                                     $e(C_1, P_2) \neq e(\mathsf{upk}_1, \mathsf{Ch}_{\mathcal{X}}(s)P_2)$
                                     $\forall\, x \in \mathcal{X} : xP_1 \neq \mathsf{ek}_1^0$
                                     $e(C_4, P_2) = e(\mathsf{upk}_2, (b + \mathsf{nym})P_2)$
                                   $\sigma \leftarrow \mathsf{SPS\text{-}EQ.Sign}(\mathsf{sps_{pp}},$
                                   $(C_1, C_2, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \mathsf{osk})$
                                   $\xleftarrow{\sigma}$
check
  $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{sps_{pp}},$
  $(C_1, C_2, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \sigma, \mathsf{opk})$
return cred $\leftarrow (C_1, C_4, C_5, \sigma, r_4, \mathsf{nym}, O)$
```

Figure 6.5: Protego: Obtain and issue algorithms.

size proof and is depicted in Figure 6.7. The differences are highlighted in grey. After the credential is updated, the user randomises the revocation accumulator, witnesses and generates the Schnorr proofs. Following the auditing proof, the Fiat-Shamir transform is applied, the ZKPoK's and PoE's are computed, returning the showing proof. Finally, Verify takes proof (depending on the case), computes the challenge and verifies each statement.

6.5.4 Security Proofs

As in Chapter 5, the security of Protego and Protego Duo relies on the security properties of the different building blocks. Furthermore, the proofs are analogous and very similar to those discussed in the previous chapter. For this reason, when presenting the security proofs for Protego, we only discuss the differences and required changes. We omit the proof of correctness, as it follows from inspection. Subsequently, we discuss unforgeability, anonymity and issuer-hiding.

Theorem 26. If the q-co-DL assumption holds (Section 2.3.3 on page 11), the ZKPoK's have perfect ZK, SCDS is sound, SPS-EQ is EUF-CMA (Definition 27 on page 62) and RevAcc is collision-free (as in Definition 16 from [DHS15a]), then Protego is unforgeable (Definition 39 on page 121).

Proof. The proof follows from the one given in Chapter 5 (Theorem 22 on page 102) and the unforgeability proof from [DHS15a] (Theorem 3), which considers the revocation part. The required changes are as follows. We assume there is

```
$\underline{\mathsf{Show}(\mathsf{pp}, \mathsf{usk}, \mathsf{upk}, \mathsf{opk}_j, \mathsf{cred}, \mathcal{S}, \mathcal{D}, (\mathsf{opk}_i)_{i\in[n]}, (\mathsf{opk}_j^i, w_j^i)_{i\in[2]}, \Omega, \mathbb{R}, \mathsf{apk}, \mathsf{tx})}$
$(C_1, C_4, C_5, \sigma, r, \mathsf{nym}, O) \leftarrow \mathsf{cred};\ (\Pi_{\mathsf{rev}}, \mathtt{WIT}) \leftarrow \mathbb{R}; \beta, \mu, \rho, \gamma, \tau, (r_i)_{i\in[5]} \xleftarrow{\$} \mathbb{Z}_p^*$
$\mathbf{if}\ O = (1, (o_1, o_2))\ \ \mathbf{then}\ O' = (1, (\mu \cdot o_1, o_2))\ \mathbf{else}\ O' = \mu \cdot O$
$\mathsf{opk}_j' \leftarrow \mathsf{ConvertPK}(\mathsf{opk}_j, \rho w_j^1 + \gamma w_j^2); \Omega' \leftarrow \mathsf{SH.ZKEval}(\mathsf{opk}_j^1, \mathsf{opk}_j^2, \Omega; \rho, \gamma)$
$\sigma' \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{sps}_{\mathsf{pp}}, (C_1, rC_1, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \sigma, \mu, \rho w_j^1 + \gamma w_j^2, \mathsf{opk}_j)$
$\mathsf{cred}' \leftarrow ((C_i)_{i\in[7]} = \mu \cdot (C_1, rC_1, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \sigma')$
$\mathsf{wss} \leftarrow \mathsf{SCDS.OpenSS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{S}, O'); \mathsf{wds} \leftarrow \mathsf{SCDS.OpenDS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{D}, O')$
$\mathsf{wss}_{\mathsf{rev}} \leftarrow \mathtt{WIT}[\mathsf{nym}]; \mathsf{wss}'_{\mathsf{rev}} \leftarrow (\tau \mathsf{wss}^1_{\mathsf{rev}}, \mathsf{usk}_2\mu\tau\mathsf{wss}^2_{\mathsf{rev}}P_1)$
$a_1 \leftarrow r_1C_1; a_2 \leftarrow r_2P_1; a_3 \leftarrow r_3\Pi_{\mathsf{rev}}; a_4 \leftarrow r_4Q; a_5 \leftarrow r_5P_1; \Pi'_{\mathsf{rev}} \leftarrow (\mathsf{usk}_2\mu\tau)\Pi_{\mathsf{rev}}$
$(\mathsf{enc}, \alpha) \leftarrow \mathsf{AuditEnc}(\mathsf{apk}, \mathsf{upk}_1); t_1 = \beta P_2; t_2 = \beta\mu P_2; t_3 = \alpha\beta P_2$
$\Pi \leftarrow \mathsf{AuditPrv}(\mathsf{enc}, \alpha, \mathsf{usk}, \mathsf{apk})$
$e \leftarrow \mathcal{H}(\mathcal{S}, \mathcal{D}, \mathsf{apk}, \mathsf{tx}, \mathsf{enc}, \Pi, \mathsf{opk}_j', (\mathsf{opk}_i)_{i\in[n]}, \Omega', (a_i)_{i\in[5]}, (t_i)_{i\in[3]}, \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{tx},$
$C_2, C_3, \Pi'_{\mathsf{rev}}, C_5, \mathsf{wss}'_{\mathsf{rev}})$
$z_1 \leftarrow r_1 + e \cdot r;\ z_2 \leftarrow r_2 + e \cdot \mu;\ z_3 \leftarrow r_3 + e \cdot (\mathsf{usk}_2\mu\tau);\ z_4 \leftarrow r_4 + e \cdot (\mathsf{usk}_2\mu)$
$z_5 \leftarrow r_5 + e \cdot (\mathsf{usk}_2\mu\tau\mathsf{wss}^2_{\mathsf{rev}})$
$\pi_1 \leftarrow \mathsf{SCDS.PoE}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{S}, e); \pi_2 \leftarrow \mathsf{SCDS.PoE}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{D}, e)$
$\mathbf{return}\ (\mathsf{enc}, (t_i)_{i\in[3]}, \mathsf{opk}', (\mathsf{opk}_i)_{i\in[n]}, \Omega', \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{wss}'_{\mathsf{rev}}, \Pi'_{\mathsf{rev}}, \Pi, \pi_1, \pi_2, (a_i, z_i)_{i\in[5]})$

$\underline{\mathsf{Verify}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \Pi_{\mathsf{rev}}, \mathsf{rpk}, \mathsf{apk}, \mathsf{tx}, \Omega)}$
$(\mathsf{enc}, (t_i)_{i\in[3]}, \mathsf{opk}', (\mathsf{opk}_i)_{i\in[n]}, \Omega', \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{wss}'_{\mathsf{rev}}, \Pi'_{\mathsf{rev}}, \Pi, \pi_1, \pi_2, (a_i, z_i)_{i\in[5]}) \leftarrow \Omega$
$(C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma) \leftarrow \mathsf{cred}'$
$e \leftarrow \mathcal{H}(\mathcal{S}, \mathcal{D}, \mathsf{apk}, \mathsf{tx}, \mathsf{enc}, \Pi, \mathsf{opk}', (\mathsf{opk}_i)_{i\in[n]}, \Omega', (a_i)_{i\in[5]}, (t_i)_{i\in[3]}, \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{tx},$
$C_2, C_3, \Pi'_{\mathsf{rev}}, C_5, \mathsf{wss}'_{\mathsf{rev}})$
$\mathbf{check}$
  $z_1C_1 = a_1 + eC_2;\ z_2P_1 = a_2 + eC_3;\ z_3\Pi_{\mathsf{rev}} = a_3 + e\Pi'_{\mathsf{rev}};\ z_4Q = a_4 + eC_5$
  $z_5P_1 = a_5 + e\mathsf{wss}'_{\mathsf{rev}};\ \mathsf{RevAcc.VerifyWit}(\Pi'_{\mathsf{rev}}, C_4, \mathsf{wss}'_{\mathsf{rev}});\ \mathsf{AuditVerify}(\mathsf{enc}, \Pi_2)$
  $e(\mathsf{enc}_1, t_2) = e(C_6, t_1) + e(C_7, t_3);\ e(\mathsf{enc}_2, t_2) = e(C_3, t_3);\ e(\mathsf{enc}_2, t_1) = e(P_1, t_3)$
  $\mathsf{SCDS.VerifySS}(C_1, \mathcal{S}, \mathsf{wss}; \pi_1, e);\ \mathsf{SCDS.VerifyDS}(C_1, \mathcal{D}, \mathsf{wds}; \pi_2, e)$
  $\mathsf{SH.PVer}((\mathsf{opk}_i)_{i\in[n]}, \mathsf{opk}', \Omega');\ \mathsf{SPS\text{-}EQ.Verify}(\mathsf{cred}', \mathsf{opk}')$
```

Figure 6.6: Protego: Show and verify algorithms.

an efficient adversary $\mathcal{A}$ winning the unforgeability game with non-negligible probability. We use $\mathcal{A}$ considering the following types of attacks:

Type 1. Adversary $\mathcal{A}$ conducts a valid showing so that $\mathsf{nym}^* = \perp$. Then we construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the EUF-CMA security.

Type 2. Adversary $\mathcal{A}$ manages to conduct a showing accepted by the verifier using the credential of user i^* under nym^* with respect to $\mathcal{S}^*$ such that $\mathcal{S}^* \not\subseteq \mathtt{ATTR}[\mathsf{nym}]$ or with respect to $\mathcal{D}^*$ such that $\mathcal{D}^* \subseteq \mathtt{ATTR}[\mathsf{nym}]$ holds. Then we construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the soundness of the set-commitment scheme SCDS.

Type 3. Adversary $\mathcal{A}$ manages to conduct a showing accepted by the verifier reusing a showing based on the credential of a user i^* under nym^* with $i^* \in \mathtt{HU}$, whose secret usk_{i^*} and credentials it does not know.

Type 4. Adversary $\mathcal{A}$ manages to conduct a showing accepted by the verifier using

```
$\underline{\mathsf{Show}(\mathsf{pp}, \mathsf{usk}, \mathsf{upk}, \mathsf{opk}_j, \mathsf{cred}, \mathcal{S}, \mathcal{D}, \mathsf{opk}_j, \sigma_j, \mathbb{R}, \mathsf{apk}, \mathsf{tx})}$
$(C_1, C_4, C_5, \sigma, r, \mathsf{nym}, O) \leftarrow \mathsf{cred};\ (\Pi_{\mathsf{rev}}, \mathtt{WIT}) \leftarrow \mathbb{R}; \beta, \mu, \rho, \gamma, \tau, (r_i)_{i\in[5]} \xleftarrow{\$} \mathbb{Z}_p^*$
$\textbf{if } O = (1, (o_1, o_2)) \ \textbf{ then } O' = (1, (\mu \cdot o_1, o_2)) \textbf{ else } O' = \mu \cdot O$
$\mathsf{opk}'_j \leftarrow \mathsf{ConvertPK}(\mathsf{opk}_j, \rho); \sigma'_j \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{sps}_{\mathsf{pp}}, \mathsf{opk}_j, \sigma_j, \rho)$
$\sigma' \xleftarrow{\$} \mathsf{SPS\text{-}EQ.ChgRep}(\mathsf{sps}_{\mathsf{pp}}, (C_1, rC_1, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \sigma, \mu, \rho, \mathsf{opk}_j)$
$\mathsf{cred}' \leftarrow ((C_i)_{i\in[7]} = \mu \cdot (C_1, rC_1, P_1, C_4, C_5, \mathsf{upk}_1, \mathsf{apk}), \sigma')$
$\mathsf{wss} \leftarrow \mathsf{SCDS.OpenSS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{S}, O'); \mathsf{wds} \leftarrow \mathsf{SCDS.OpenDS}(\mathsf{scds}_{\mathsf{pp}}, C_1, \mathcal{D}, O')$
$\mathsf{wss}_{\mathsf{rev}} \leftarrow \mathtt{WIT}[\mathsf{nym}]; \mathsf{wss}'_{\mathsf{rev}} \leftarrow (\tau \mathsf{wss}^1_{\mathsf{rev}}, \mathsf{usk}_2 \mu \tau \mathsf{wss}^2_{\mathsf{rev}} P_1)$
$a_1 \leftarrow r_1 C_1; a_2 \leftarrow r_2 P_1; a_3 \leftarrow r_3 \Pi_{\mathsf{rev}}; a_4 \leftarrow r_4 Q; a_5 \leftarrow r_5 P_1; \Pi'_{\mathsf{rev}} \leftarrow (\mathsf{usk}_2 \mu \tau)\Pi_{\mathsf{rev}}$
$(\mathsf{enc}, \alpha) \leftarrow \mathsf{AuditEnc}(\mathsf{apk}, \mathsf{upk}_1); t_1 = \beta P_2; t_2 = \beta \mu P_2; t_3 = \alpha \beta P_2$
$\Pi \leftarrow \mathsf{AuditPrv}(\mathsf{enc}, \alpha, \mathsf{usk}, \mathsf{apk})$
$e \leftarrow \mathcal{H}(\mathcal{S}, \mathcal{D}, \mathsf{apk}, \mathsf{tx}, \mathsf{enc}, \Pi, \mathsf{opk}'_j, \sigma'_j, (a_i)_{i\in[5]}, (t_i)_{i\in[3]}, \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{tx},$
$C_2, C_3, \Pi'_{\mathsf{rev}}, C_5, \mathsf{wss}'_{\mathsf{rev}})$
$z_1 \leftarrow r_1 + e \cdot r;\ z_2 \leftarrow r_2 + e \cdot \mu;\ z_3 \leftarrow r_3 + e \cdot (\mathsf{usk}_2 \mu \tau);\ z_4 \leftarrow r_4 + e \cdot (\mathsf{usk}_2 \mu)$
$z_5 \leftarrow r_5 + e \cdot (\mathsf{usk}_2 \mu \tau \mathsf{wss}^2_{\mathsf{rev}})$
$\pi_1 \leftarrow \mathsf{SCDS.PoE}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{S}, e); \pi_2 \leftarrow \mathsf{SCDS.PoE}(\mathsf{scds}_{\mathsf{pp}}, \mathcal{D}, e)$
$\textbf{return } (\mathsf{enc}, (t_i)_{i\in[3]}, \mathsf{opk}'_j, \sigma'_j, \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{wss}'_{\mathsf{rev}}, \Pi'_{\mathsf{rev}}, \Pi, \pi_1, \pi_2, (a_i, z_i)_{i\in[5]})$

$\underline{\mathsf{Verify}(\mathsf{pp}, \mathcal{S}, \mathcal{D}, \Pi_{\mathsf{rev}}, \mathsf{rpk}, \mathsf{apk}, \mathsf{vpk}, \mathsf{tx}, \Omega)}$
$(\mathsf{enc}, (t_i)_{i\in[3]}, \mathsf{opk}', \sigma', \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{wss}'_{\mathsf{rev}}, \Pi'_{\mathsf{rev}}, \Pi, \pi_1, \pi_2, (a_i, z_i)_{i\in[5]}) \leftarrow \Omega$
$(C_1, C_2, C_3, C_4, C_5, C_6, C_7, \sigma) \leftarrow \mathsf{cred}'$
$e \leftarrow \mathcal{H}(\mathcal{S}, \mathcal{D}, \mathsf{apk}, \mathsf{tx}, \mathsf{enc}, \Pi, \mathsf{opk}', \sigma', (a_i)_{i\in[5]}, (t_i)_{i\in[3]}, \mathsf{cred}', \mathsf{wss}, \mathsf{wds}, \mathsf{tx},$
$C_2, C_3, \Pi'_{\mathsf{rev}}, C_5, \mathsf{wss}'_{\mathsf{rev}})$
$\textbf{check}$
  $z_1 C_1 = a_1 + eC_2;\ z_2 P_1 = a_2 + eC_3;\ z_3 \Pi_{\mathsf{rev}} = a_3 + e\Pi'_{\mathsf{rev}};\ z_4 Q = a_4 + eC_5$
  $z_5 P_1 = a_5 + e\mathsf{wss}'_{\mathsf{rev}};\ \mathsf{RevAcc.VerifyWit}(\Pi'_{\mathsf{rev}}, C_4, \mathsf{wss}'_{\mathsf{rev}});\ \mathsf{AuditVerify}(\mathsf{enc}, \Pi_2)$
  $e(\mathsf{enc}_1, t_2) = e(C_6, t_1) + e(C_7, t_3);\ e(\mathsf{enc}_2, t_2) = e(C_3, t_3);\ e(\mathsf{enc}_2, t_1) = e(P_1, t_3)$
  $\mathsf{SCDS.VerifySS}(C_1, \mathcal{S}, \mathsf{wss}; \pi_1, e);\ \mathsf{SCDS.VerifyDS}(C_1, \mathcal{D}, \mathsf{wds}; \pi_2, e)$
  $\mathsf{SPS\text{-}EQ.Verify}(\mathsf{opk}', \mathsf{vpk});\ \mathsf{SPS\text{-}EQ.Verify}(\mathsf{cred}', \mathsf{opk}')$
```

Figure 6.7: Protego Duo: Show and verify algorithms.

some credential corresponding to a revoked pseudonym $\mathsf{nym}^* \in \mathsf{RNYM}$. Then, we construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the binding property of the revocation accumulator RevAcc.

Types 1 and 2 follow the proofs of Theorem 22 as the underlying primitives remain unchanged. For Type 3, we leverage the fact that reusing a showing would only allow the adversary to generate a valid showing for *the same* original transaction tx (that is timestamped), and hence, we do not consider it as an attack. Observe that any modification done to the original tx will lead to a different challenge; thus, the rest of the proofs (showing, revocation and auditing) will not verify. Finally, Type 4 follows from [DHS15a] (Theorem 3). □

Theorem 27. If the DDH assumption holds, the $\mathsf{SPS\text{-}EQ}$ perfectly adapts signatures (Definition 28 on page 63), and $\mathcal{H}$ is assumed to be a random oracle, then Protego is anonymous (Definition 40 on page 121).

Proof. The proof follows Theorem 23 and [DHS15a] (Theorem 4). However, we must also to take into account the ROM and the addition of the auditing features. The extra credential components for the auditing are randomised during every credential showing like the rest of the components. Similarly, the user generates a new encryption of the auditor's public key with a fresh α while a fresh β is used to randomise the values t_i. Since ElGamal encryption is IND-CPA secure and key-private [BBDP01], the ciphertexts produced by the user are indistinguishable and do not leak information about the user's public key nor the auditor's. $\square$

Theorem 28. If the algorithms $\mathsf{AuditPrv}$ and $\mathsf{AuditVerify}$ are a NIZK proof system and the $\mathsf{SPS\text{-}EQ}$ is EUF-CMA (Definition 27 on page 62) then Protego is auditable (Definition 41 on page 122).

Proof. If the verification returns true, we have that $\exists\ (\mathsf{enc}_1^*, \mathsf{enc}_2^*) = ((\delta^* + \alpha^*\mathsf{ask})P_1, \alpha^* P_1)$ for some δ^* and α^* chosen by the adversary. Moreover, because of the unforgeability of the signature scheme, the verification implies that $C_3 = \mu^* P_1$, $C_6 = \mu^*\mathsf{usk}_1 P_1$ and $C_7 = \mu^*\mathsf{ask}P_1$ for some μ^* chosen by the adversary. As a result, we can re-write the pairing equations for the audit proof as:

$$\begin{aligned} e(\alpha^* P_1, t_2^*) &= e(\mu^* P_1, t_3^*) \\ e(\alpha^* P_1, t_1^*) &= e(P_1, t_3^*) \\ e((\delta^* + \alpha^*\mathsf{ask})P_1, t_2^*) &= e(\mu^*\mathsf{usk}_1 P_1, t_1^*) + e(\mu^*\mathsf{ask}P_1, t_3^*) \end{aligned}$$

where t_1^*, t_2^* and t_3^* are also chosen by the adversary. We show that $\delta^* = \mathsf{usk}_1$, which implies that $\mathsf{upk}_1 = \mathsf{AuditDec}(\mathsf{enc}, \mathsf{ask})$. Looking at the first two equations in the target group, we have that $\alpha^* t_2^* = \mu^* t_3^*$ and $\alpha^* t_1^* = t_3^*$, concluding that $t_2^* = \mu^* t_1^*$. Replacing t_2^* and t_3^* in third one and simplyfing we obtain:

$$\begin{aligned} (\delta^* + \alpha^*\mathsf{ask})\mu^* t_1^* &= \mu^*\mathsf{usk}_1 t_1^* + \mu\mathsf{ask}\alpha^* t_1^* \\ \mu^*\delta^* t_1^* + \mu^*\alpha^*\mathsf{ask}t_1^* &= \mu^*\mathsf{usk}_1 t_1^* + \mu^*\alpha^*\mathsf{ask}t_1^* \end{aligned}$$

deducing that $\delta^* = \mathsf{usk}_1$. $\square$

Theorem 29. If the underlying SPS-EQ perfectly adapts signatures (Definition 31), then Protego is issuer-hiding (Definition 38 on page 95).

Proof. Analogous to the one given in Chapter 5 (Theorem 24 on page 108) $\square$

Integration with Fabric. A multi-party computation ceremony can be run for the CRS generation of the Setup algorithm, ensuring that no organisation knows the trapdoors of the different components. As we are in the permissioned setting it is plausible to assume that at least one of the organisations is honest. By allowing *users and endorsers* to obtain credentials, both can produce showing proofs. Users can generate showing proofs to prove that they satisfy the access policy for the execution of a particular transaction proposal. Furthermore, by computing the PoE's, the verification time for endorsers improves substantially. Similarly, endorsers can prove that they satisfy a given endorsement policy attaching a showing proof to their endorsements. Even if the endorsement policies are defined in a privacy-preserving way as suggested in [ADCNS19], endorsers can still compute selective AND and NAND clauses for the respective pseudonyms defined by the policy using their credentials. Endorsers should also use the read and write sets to from the transaction proposals to generate showing proofs.

6.6 Implementation and Evaluation

We implemented a prototype version of Protego and Protego Duo in Rust using the bls12-381 curve[1] and the BLAKE3 hash function[2]. This choice is based on their performance, wide adoption and available open-sourced implementations in Rust. The source code and related documentation are provided in [CDLPK22]. Our signature implementation is based on the one from [Bur20] but using the bls12-381 curve instead of Barreto-Naehrig curves [BN06]. As a result, we obtain times up to 67% faster when compared to [Bur20]. To run the benchmarks a Macbook Air (Chip M2 & 16GB RAM) was used with no extra optimizations, using the nightly compiler and the *Criterion* library. For all values, the standard deviation was at most 1 millisecond.

Issue and Obtain take roughly 20 ms each when issuing a credential for 10 attributes. Both scale linearly on the number of attributes. To evaluate showing and verification, we considered the PoE in the showing protocol. Therefore, verification running time remains (almost) constant[3] regardless the number of shown attributes, credential size, and issuer-hiding approach. If the PoE is disabled, showing running time would be smaller while verification would increase linearly with the number of shown attributes.

	Revocation				Signature			Issuer-hiding NIZK		
	$n = 10$		$n = 100$		$\ell = 7$ (for Protego)			$n = 5$		
Scheme	Prove	Verify	Prove	Verify	Sign	Verify	ChgRep	Prove	Verify	ZKEval
[BDCET21]	28	64	**28**	64	23	59	N/A	N/A	N/A	N/A
Protego	**7.7**	**4.2**	77.4	**4.2**	**3.4**	**11**	8.8	103	118	59

Table 6.1: Running time for the different algorithms in milliseconds.

[1] https://electriccoin.co/blog/new-snark-curve

[2] https://github.com/BLAKE3-team/BLAKE3

[3] Asymptotic complexity is O(1) (considering exponentiations and pairings) but some multiplications depending on the shown attributes are required, hence the difference.

An auditing proof in Protego takes roughly 1 and 1.5 ms for proof generation and verification, surpassing the values from [BDCET21]. In Table 6.1 we report the revocation and signing algorithms, including our issuer-hiding NIZK with $n = 5$. For Protego, we consider a signature for vectors of length seven (the size of a credential). In our case, the revocation witnesses are computed by the authority (in linear time) and then randomized by the users (in constant time). For this reason we consider the generation of a single witness for a revocation lists of 10 and a hundred elements (although in practice one would expect it to be closer to 10). For [BDCET21], we consider the total time to generate and verify a signature in a user level $L = 2$ (involving two delegations), with revocation times in $\mathbb{G}_2$.

Comparison with the Idemix extension from [BDCET21]. The computational cost for the prover and verifier grows linearly with the number of attributes in the credential and delegation levels for [BDCET21]. In Protego Duo, the prover computational cost is $O(n - k)$ for showings involving k-attributes out of n, which in practice is much better. Verification cost in Protego and Protego Duo is almost constant (or O(k) if the PoE is disabled).

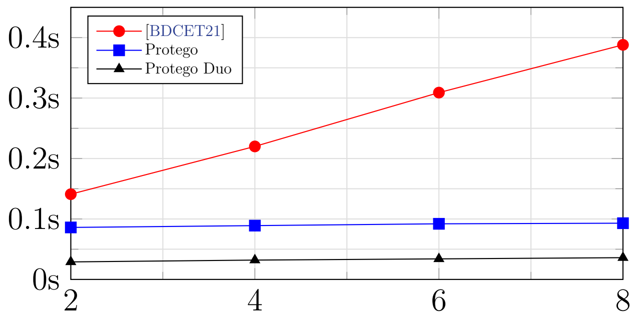


Figure 6.8: Showing times in seconds considering 2, 4, 6 and 8 attributes.

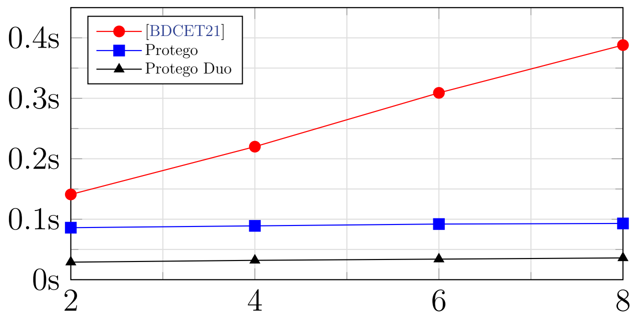


Figure 6.9: Verification times in seconds considering 2, 4, 6 and 8 attributes.

The two works are compared in figures 6.8 and 6.9, using the same hardware and considering credential showings and verification times, respectively. Exact times for the values presented in both figures are also given in Table 6.2. For [BDCET21], we consider a delegation level $L = 2$, which corresponds to a user level given that the root authority is at $L = 0$ and organizations start at $L = 1$. Regarding the

	$k = 2$		$k = 4$		$k = 6$		$k = 8$		$k = 10$	
Scheme	Show	Verify	Show	Verify	Show	Verify	Show	Verify	Show	Verify
[BDCET21]	141	106	220	170	309	266	388	356	-	-
Protego	86	142	89	140	92	141	93	145	96	145
Protego Duo	**29**	**35**	**32**	**35**	**34**	**36**	**37**	**36**	**39**	**38**

Table 6.2: Protocols' comparison showing the running times in milliseconds.

attributes, [BDCET21] we could only retrieve information considering proofs for credential possesion below ten attributes (assuming a minimal overhead when all attributes are shown as authors suggest). Therefore, we report credential possesions for [BDCET21] considering up to 8 attributes, and selective disclosures of k-out-of-10 attributes in ours. For Protego, we consider five authorities for the NIZK proof, which would suffice for practical scenarios like a consortium of pharmaceuticals.

6.7 Conclusions and Future Work

We have seen the development of two "breeds" of credential schemes, one based on the line of work following blind signatures and delegatable credentials and the other on SPS-EQ. We presented here the first SPS-EQ credential scheme modified to work with permissioned blockchains. The versatility of Protego alongside the efficiency gains (at least twice as fast as the most recent Idemix extension), enables a broader scope of applications in such a setting. Depending on the context, the PoE's can be computed or not, the credential issuer can be hidden or not, and one can select only subsets or disjoint sets to generate the proofs. Similarly, auditability and revocation features can be considered as optional, showing its flexibility.

As future directions to explore, we consider the following points:

1. Adding confidentiality of transactions to a Protego-like credential scheme.
2. Adding more power to the users (*i.e.,* how to define precise notions of user-invoked regulatory measures).
3. Extend our results to the multi-authority setting, where users can get attributes from multiple authorities instead of a single one.

7
Conclusion

Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza.

— Dante Alighieri

This thesis studied malleability in the context of PKE and digital signatures.

For PKE, we characterised malleability in terms of generic definitions for randomisable encryption. We presented interactive and non-interactive zero-knowledge protocols for plaintext equality and inequality based on those definitions. While previous work relied on specific constructions that did not allow the scheme to be separated from the protocols requirements, our constructions are more versatile. They can be easily integrated with other building blocks. This makes it easier to adapt or extend use cases implementing privacy-enhancing technologies.

Regarding digital signatures, we proposed new constructions in the standard model and applications in the context of anonymous credentials, one of the most prominent privacy-enhancing technology. In particular, the problem of identity management in usable real-world applications has had a bumpy history. Its foundations remain weak as we progress into a more federated and decentralised world. Cryptographic primitives used to build early credential systems falter when ported to industrial applications. The increasing need for fine-grained access control, coupled with the greater awareness and demand for user-privacy, leaves many challenges to be addressed. For this reason, we pushed towards more expressive and efficient constructions. As a result, we extended previous credential systems and proposed new ones tailored to the enterprise blockchain setting. These contributions enable more use cases, which could not easily be implemented with the technology that was previously available.

To show the feasibility of our results, we provided prototype implementations for the cryptographic primitives and protocols presented in chapters 3 and 6. Our evaluations confirm the practicality of our contributions and set the path for future industrialisation of the presented credential schemes.

Open issues and future work. Considering zero-knowledge protocols for plaintext equality and inequality, we have seen how they are used in online voting schemes, reputation systems and cloud-based applications. In this sense, we stress that having generic protocols for inequality tests could also enable more use cases. Likewise, the design of generic non-interactive protocols for plaintext inequality can help in the same direction.

We provided a SPS-EQ from standard assumptions in Chapter 4, which we later extended to obtain the first mercurial signature in the same setting. In this regard, we recall that our signature construction requires a CRS and that it only achieves a weak form of anonymity with respect to the key space. Although we do not consider the use of a CRS a major drawback, it would be desirable to achieve a construction secure in the standard model alone. Similarly, we leave the construction of mercurial signatures from standard assumptions and achieving a stronger notion of perfect adaption, as an interesting open problem.

We have seen how SPS-EQ provides a host of functionalities for designing other primitives with many efficiencies and interoperability gains. Unfortunately, they do not provide post-quantum security guarantees. Therefore, studying alternative constructions that could offer post-quantum security is an open issue and promising future work. While these lines were being written, the National Institute of Standards and Technology (NIST) announced the first group of winners from its six-year competition on post-quantum cryptographic algorithms to be standardised. Furthermore, another call for digital signature algorithms with short signatures and fast verification was also announced. This situation reflects how active this research field is and the importance of developing signatures considering different trade-offs. The ideas developed in this thesis shed light in that direction even if the presented constructions do not offer post-quantum security.

While the work on anonymous credentials validated many of the underlying building blocks for the first time, providing an implementation, there is still much room for improvement. Regarding security, the extensions presented in Chapter 5 and 6 were studied independently. Analyzing security using the universal composability framework [Can01] is left as interesting future work. In terms of functionalities, incorporating other building blocks like aggregatable signatures or defining equivalence classes in a different way are also interesting directions to explore. In this regard, extending the contributions from Chapter 6 to the multi-authority setting is also an interesting future work.

Bibliography

[Abd14] Michel Abdalla. Brief Introduction to Provable Security. https://www.di.ens.fr/~mabdalla/coursedocs/provablesecurity.pdf, 2014. [Online; accessed 01-April-2022]. Cited on page 7.

[ACC$^+$20] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, page 255267, New York, NY, USA, 2020. Association for Computing Machinery. Cited on page 113.

[ADCNS19] Elli Androulaki, Angelo De Caro, Matthias Neugschwandtner, and Alessandro Sorniotti. Endorsement in hyperledger fabric. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 510–519, 2019. Cited on pages 113 and 131.

[ADKL19] Prabhanjan Ananth, Apoorvaa Deshpande, Yael Tauman Kalai, and Anna Lysyanskaya. Fully homomorphic NIZK and NIWI proofs. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 356–385, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany. Cited on page 3.

[AFG$^+$10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. Cited on page 59.

[AFG$^+$16] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. *Journal of Cryptology*, 29(2):363–421, April 2016. Cited on page 59.

[AGOT14] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 688–712, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. Cited on page 59.

[AJO$^+$19] Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with tighter security. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 669–699, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany. Cited on page 69.

[ATSM09] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In Marc Fischlin, editor, *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 295–308, San Francisco, CA, USA, April 20–24, 2009. Springer, Heidelberg, Germany. Cited on page 101.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. Cited on pages 18 and 130.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 561–586, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. Cited on page 82.

[BBLPK21a] Olivier Blazy, Xavier Bultel, Pascal Lafourcade, and Octavio Perez Kempner. Generic Plaintext Equality and Inequality Proofs. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security*, pages 415–435, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg. Cited on pages 3, 6, 27, and 57.

[BBLPK21b] Olivier Blazy, Xavier Bultel, Pascal Lafourcade, and Octavio Perez Kempner. Generic Plaintext Equality and Inequality Proofs (Extended Version). Cryptology ePrint Archive, Report 2021/426, 2021. `https://ia.cr/2021/426`. Cited on page 3.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in*

Computer Science, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. Cited on pages 23 and 50.

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 145–164, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany. Cited on page 29.

[BDCET21] Dmytro Bogatov, Angelo De Caro, Kaoutar Elkhiyaoui, and Björn Tackmann. Anonymous Transactions withăRevocation andăAuditing inăHyperledger Fabric, 2021. Cited on pages 5, 112, 113, 116, 122, 131, 132, and 133.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. Cited on page 17.

[BDSS16] Olivier Blazy, David Derler, Daniel Slamanig, and Raphael Spreitzer. Non-interactive plaintext (in-)equality proofs and group signatures with verifiable controllable linkability. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 127–143, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany. Cited on page 29.

[BEK$^+$20] Jan Bobolz, Fabian Eidens, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Privacy-preserving incentive systems with highly efficient point-collection. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*, pages 319–333, Taipei, Taiwan, October 5–9, 2020. ACM Press. Cited on page 60.

[BEK$^+$21] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-Hiding Attribute-Based Credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 158–178, Cham, 2021. Springer International Publishing. Cited on pages 5, 97, and 124.

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume

11692 of *Lecture Notes in Computer Science*, pages 413–434, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany. Cited on page 2.

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider. Signatures with flexible public key: Introducing equivalence classes for public keys. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 405–434, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. Cited on page 60.

[BHSB19] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2181–2198. ACM Press, November 11–15, 2019. Cited on page 60.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 408–425, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. Cited on page 61.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 1087–1098, Berlin, Germany, November 4–8, 2013. ACM Press. Cited on page 80.

[BL16] Xavier Bultel and Pascal Lafourcade. A Posteriori Openable Public Key Encryption. In Jaap-Henk Hoepman and Stefan Katzenbeisser, editors, *ICT Systems Security and Privacy Protection*, pages 17–31, Cham, 2016. Springer International Publishing. Cited on page 19.

[BLL$^+$19] Xavier Bultel, Pascal Lafourcade, Russell W. F. Lai, Giulio Malavolta, Dominique Schröder, and Sri Aravinda Krishnan Thyagarajan. Efficient invisible and unlinkable sanitizable signatures. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 11442 of *Lecture Notes in Computer Science*, pages 159–189, Beijing, China, April 14–17, 2019. Springer, Heidelberg, Germany. Cited on page 60.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331, Kingston, Ontario, Canada, August 11–12, 2006. Springer, Heidelberg, Germany. Cited on page 131.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. Cited on page 89.

[BPW12] David Bernhard, Olivier Pereira, and Bogdan Warinschi. How not to prove yourself: Pitfalls of the Fiat-Shamir heuristic and applications to Helios. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 626–643, Beijing, China, December 2–6, 2012. Springer, Heidelberg, Germany. Cited on page 23.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. Cited on page 10.

[Bra00] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000. Cited on page 80.

[BS99] Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. Cited on page 2.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 350–366, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery. Cited on page 3.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes* *in Computer Science*, pages 136–153, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany. Cited on page 91.

[Bur20] Michael Burkhart. Mercurial signatures implementatio. Github, 2020. `https://github.com/burkh4rt/Mercurial-Signatures`. Cited on page 131.

[BVE$^+$03] John Borking, P. Verhaar, B.M.A. Eck, P. Siepel, G.W. Blarkom, R. Coolen, M. Uyl, J. Holleman, P. Bison, R. Veer, J. Giezen, Andrew Patrick, C. Holmes, J.C.A. Lubbe, Roy Lachman, S. Kenny, Randy Song, K. Cartrysse, J. Huizenga, and X. Zhou. *Handbook of Privacy and Privacy-Enhancing Technologies The case of Intelligent Software Agents*. 11 2003. Cited on page 2.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. Cited on page 136.

[CDHK15] Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 262–288, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany. Cited on pages 80, 81, and 82.

[CDLPK22] Aisling Connolly, Jerome Deschamps, Pascal Lafourcade, and Octavio Perez Kempner. Protego: Efficient, Revocable and Auditable Anonymous Credentials with Applications to Hyperledger Fabric. Cryptology ePrint Archive, Report 2022/661, 2022. `https://ia.cr/2022/661`. Cited on pages 3, 6, 111, and 131.

[CEJ$^+$07] Seung Geol Choi, Ariel Elbaz, Ari Juels, Tal Malkin, and Moti Yung. Two-party computing with encrypted data. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 298–314, Kuching, Malaysia, December 2–6, 2007. Springer, Heidelberg, Germany. Cited on page 29.

[CFGL12] Sébastien Canard, Georg Fuchsbauer, Aline Gouget, and Fabien Laguillaumie. Plaintext-checkable encryption. In Orr Dunkelman, editor, *Topics in Cryptology – CT-RSA 2012*, volume 7178 of *Lecture Notes in Computer Science*, pages 332–348, San Francisco, CA, USA, February 27 – March 2, 2012. Springer, Heidelberg, Germany. Cited on page 30.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press. Cited on page 10.

[CH20] Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 768–798, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. Cited on pages 4, 11, 12, 64, 65, 66, 67, 77, 99, and 100.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 281–300, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. Cited on pages 3 and 24.

[CKLM14] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable signatures: New definitions and delegatable anonymous credentials. In Anupam Datta and Cedric Fournet, editors, *CSF 2014: IEEE 27st Computer Security Foundations Symposium*, pages 199–213, Vienna, Austria, jul 19-22 2014. IEEE Computer Society Press. Cited on page 3.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. Cited on page 30.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289, Amalfi, Italy, September 12–13, 2003. Springer, Heidelberg, Germany. Cited on page 80.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. Cited on pages 80, 81, 82, and 114.

[CL11] Sébastien Canard and Roch Lescuyer. Anonymous Credentials from (Indexed) Aggregate Signatures. In *Proceedings of the 7th ACM Workshop on Digital Identity Management*, DIM '11, page 5362, New York, NY, USA, 2011. Association for Computing Machinery. Cited on pages 80, 81, and 82.

[CL13] Sébastien Canard and Roch Lescuyer. Protecting privacy by sanitizing personal data: a new approach to anonymous credentials. In Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng, editors, *ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*, pages 381–392, Hangzhou, China, May 8–10, 2013. ACM Press. Cited on pages 81 and 82.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *Topics in Cryptology – CT-RSA 2019*, volume 11405 of *Lecture Notes in Computer Science*, pages 535–555, San Francisco, CA, USA, March 4–8, 2019. Springer, Heidelberg, Germany. Cited on pages 4, 60, 61, 63, 75, 77, 112, and 114.

[CL21] Elizabeth C. Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *Proceedings on Privacy Enhancing Technologies*, 2021(4):441–463, October 2021. Cited on pages 60, 61, and 75.

[CLPK21] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved Constructions of Anonymous Credentials From Structure-Preserving Signatures on Equivalence Classes. Cryptology ePrint Archive, Report 2021/1680, 2021. https://ia.cr/2021/1680. Cited on pages 3, 59, and 79.

[CLPK22] Aisling Connolly, Pascal Lafourcade, and Octavio Perez Kempner. Improved Constructions of Anonymous Credentials from Structure-Preserving Signatures on Equivalence Classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 409–438, Cham, 2022. Springer International Publishing. Cited on pages 3, 5, 6, 75, 93, 97, 98, 112, and 122.

[Con19] Aisling Connolly. *Try Again. Fail Again. Fail Better : New notions of security, broken assumptions, and increased efficiency in cryptography.* Theses, Université Paris sciences et lettres, September 2019. Cited on page 7.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. Cited on page 58.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. Cited on page 29.

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. Cited on page 58.

[CV02] Jan Camenisch and Els Van Herreweghen. Design and implementation of the idemix anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM CCS 2002: 9th Conference on Computer and Communications Security*, pages 21–30, Washington, DC, USA, November 18–22, 2002. ACM Press. Cited on page 112.

[Dam88] Ivan Damgård. Collision free hash functions and public key signature schemes. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology – EUROCRYPT'87*, volume 304 of *Lecture Notes in Computer Science*, pages 203–216, Amsterdam, The Netherlands, April 13–15, 1988. Springer, Heidelberg, Germany. Cited on page 13.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. Cited on pages 29 and 48.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. Cited on pages 10 and 95.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. Cited on page 2.

[Des19] Jeff Desjardins. How much data is generated each day? , 2019. https://www.weforum.org/agenda/2019/04/

how-much-data-is-generated-each-day-cf4bddf29f/. Cited on page 1.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. Cited on page 1.

[DHS15a] David Derler, Christian Hanser, and Daniel Slamanig. A new approach to efficient revocable attribute-based anonymous credentials. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *Lecture Notes in Computer Science*, pages 57–74, Oxford, UK, December 15–17, 2015. Springer, Heidelberg, Germany. Cited on pages 5, 60, 80, 101, 102, 112, 117, 119, 122, 127, 129, and 130.

[DHS15b] David Derler, Christian Hanser, and Daniel Slamanig. Revisiting cryptographic accumulators, additional properties and relations to other primitives. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 127–144, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany. Cited on pages 82, 83, and 84.

[Dif81] Whitfield Diffie. Cryptography, the next two decades. In Allen Gersho, editor, *Advances in Cryptology – CRYPTO'81*, volume ECE Report 82-04, pages 84–108, Santa Barbara, CA, USA, 1981. U.C. Santa Barbara, Dept. of Elec. and Computer Eng. Cited on page 15.

[DJN10] Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. A Generalization of Paillier's Public-Key System with Applications to Electronic Voting. *Int. J. Inf. Secur.*, 9(6):371385, 2010. Cited on page 27.

[DM14] Tassos Dimitriou and Antonis Michalas. Multi-Party Trust Computation in Decentralized Environments in the Presence of Malicious Adversaries. *Ad Hoc Netw.*, 15:5366, 2014. Cited on page 27.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010: 1st Innovations in Computer Science*, pages 434–452, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press. Cited on page 2.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18: 13th ACM Symposium on Information, Computer* *and Communications Security*, pages 551–565, Incheon, Republic of Korea, April 2–6, 2018. ACM Press. Cited on page 60.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. Cited on pages 12 and 61.

[EHK$^+$17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, January 2017. Cited on page 11.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany. Cited on pages 19 and 48.

[ElG86] Taher ElGamal. On computing logarithms over finite fields. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 396–402, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany. Cited on page 29.

[FFHR19] Antonio Faonio, Dario Fiore, Javier Herranz, and Carla Ràfols. Structure-preserving and re-randomizable RCCA-secure public key encryption and its applications. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 159–190, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany. Cited on page 30.

[FG18] Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 153–183, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany. Cited on pages 60 and 61.

[FGKO17] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi. Access control encryption for equality, comparison, and more. In Serge Fehr, editor, *PKC 2017: 20th International*

Conference on Theory and Practice of Public Key Cryptography, Part II, volume 10175 of *Lecture Notes in Computer Science*, pages 88–118, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany. Cited on page 60.

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 391–408, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany. Cited on pages 60 and 80.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 233–253, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. Cited on page 60.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019. Cited on pages xvi, 3, 4, 5, 11, 20, 60, 61, 79, 80, 81, 82, 83, 84, 85, 87, 91, 92, 93, 95, 96, 102, 105, 108, 109, 110, 112, 117, 122, and 126.

[FKPS21] Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable Time-Lock Puzzles and Applications. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 447–479, Cham, 2021. Springer International Publishing. Cited on page 2.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. Cited on page 23.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press. Cited on page 3.

[GHK17] Romain Gay, Dennis Hofheinz, and Lisa Kohl. Kurosawa-desmedt meets tight security. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 133–160, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. Cited on pages 61 and 64.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 230–258, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. Cited on pages 12, 61, 64, 67, and 69.

[GJJS04] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul F. Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 163–178, San Francisco, CA, USA, February 23–27, 2004. Springer, Heidelberg, Germany. Cited on page 30.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 685–698, Los Angeles, CA, USA, June 25–29, 2018. ACM Press. Cited on page 2.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th Annual ACM Symposium on Theory of Computing*, pages 365–377, San Francisco, CA, USA, May 5–7, 1982. ACM Press. Cited on pages 16, 29, and 48.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press. Cited on page 21.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. Cited on page 20.

[GOP$^+$16] Esha Ghosh, Olga Ohrimenko, Dimitrios Papadopoulos, Roberto Tamassia, and Nikos Triandopoulos. Zero-knowledge accumulators and set algebra. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 67–100, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. Cited on pages 11, 82, and 83.

[GR04] Craig Gentry and Zulfikar Ramzan. Eliminating random permutation oracles in the Even-Mansour cipher. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 32–47, Jeju Island, Korea, December 5–9, 2004. Springer, Heidelberg, Germany. Cited on page 10.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. Cited on pages 29, 58, and 59.

[GT20] Oded Goldreich and Liav Teichner. *Super-Perfect Zero-Knowledge Proofs*. Springer International Publishing, Cham, 2020. Cited on page 20.

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 469–477, Portland, OR, USA, June 14–17, 2015. ACM Press. Cited on page 3.

[HBBS13] O. Hasan, L. Brunie, E. Bertino, and N. Shang. A Decentralized Privacy Preserving Reputation Protocol for the Malicious Adversarial Model. *IEEE Transactions on Information Forensics and Security*, 8(6):949–962, 2013. Cited on page 27.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, Heidelberg, Germany, 2010. Cited on pages 22 and 23.

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 489–518, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. Cited on page 64.

[HP20] Chloé Hébant and David Pointcheval. Traceable constant-size multi-authority credentials. Cryptology ePrint Archive, Report 2020/657, 2020. `https://eprint.iacr.org/2020/657`. Cited on pages 80, 81, 82, 108, and 110.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. Cited on page 30.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. Cited on pages 59, 60, and 82.

[JJ00] Markus Jakobsson and Ari Juels. Mix and match: Secure function evaluation via ciphertexts. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 162–177, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. Cited on page 29.

[Kat02] Jonathan Katz. CMSC 456 - Introduction to Cryptography. Lecture Notes. `https://www.cs.umd.edu/~jkatz/crypto/`, 2002. [Online; accessed 01-April-2022]. Cited on page 10.

[KDJL+19] Hui Kang, Ting Dai, Nerla Jean-Louis, Shu Tao, and Xiaohui Gu. Fabzk: Supporting privacy-preserving, auditable smart contracts in hyperledger fabric. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 543–555, 2019. Cited on page 113.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Third Edition.* Chapman & Hall/CRC, 3ed edition, 2021. Cited on page 1.

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 275–295, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. Cited on page 61.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 63–93, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany. Cited on pages 4, 24, 60, 61, 62, 64, 65, 66, 67, 69, 74, and 77.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 101–128, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. Cited on page 65.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany. Cited on page 82.

[Lin01] Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 171–189, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. Cited on page 23.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany. Cited on page 10.

[MPT20] Eleanor McMurtry, Olivier Pereira, and Vanessa Teague. When is a test not a proof? In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 23–41, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany. Cited on page 29.

[MR19] Subhra Mazumdar and Sushmita Ruj. Design of anonymous endorsement system in hyperledger fabric. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2019. Cited on page 113.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 729–758, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. Cited on pages 11 and 12.

[Ngu05] Lan Nguyen. Accumulators from bilinear pairings and applications. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 275–292, San Francisco, CA, USA, February 14–18, 2005. Springer, Heidelberg, Germany. Cited on page 82.

[NVV18] Neha Narula, Willy Vasquez, and Madars Virza. Zkledger: Privacy-preserving auditing for distributed ledgers. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, NSDI'18, page 6580, USA, 2018. USENIX Association. Cited on page 113.

[OO14] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association. Cited on page 114.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. Cited on pages 29 and 48.

[Pas04] Rafael Pass. Alternative Variants of Zero-Knowledge Proofs. Technical report, KTH Royal Institute of Technology, 2004. Cited on page 20.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. Cited on pages 13 and 14.

[Plo21] Antoine Plouviez. *The security of the One-More Discrete-Logarithm assumption and Blind Schnorr Signatures.* Theses, ENS Paris, October 2021. Cited on page 10.

[PoEU16] European Parliament and Council of European Union. Regulation (eu) 2016/679. `https://eur-lex.europa.eu/eli/reg/2016/679/oj`, 2016. Accessed: 2021-02-11. Cited on page 79.

[PR07] Manoj Prabhakaran and Mike Rosulek. Rerandomizable RCCA encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 517–534, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. Cited on pages 2, 17, 29, 30, and 31.

[PRST06] D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe. Practical Secrecy-Preserving, Verifiably Correct and Trustworthy

Auctions. ICEC ’06, page 7081, New York, NY, USA, 2006. Association for Computing Machinery. Cited on page 29.

[PTT11] Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal verification of operations on dynamic sets. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 91–110, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. Cited on page 82.

[PZ13] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1, revision 3. Technical report, Microsoft Corporation, 2013. Cited on page 80.

[Ràf15] Carla Ràfols. Stretching groth-sahai: NIZK proofs of partial satisfiability. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 247–276, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. Cited on pages 64 and 65.

[Rei18] Manuel Reinert. *Cryptographic Techniques for Privacy and Access Control in Cloud-Based Applications.* PhD thesis, Saarland University, Saarbrücken, Germany, 2018. Cited on page 27.

[Rog06] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228, Hanoi, Vietnam, September 25–28, 2006. Springer, Heidelberg, Germany. Cited on page 13.

[RS06] Peter Y. A. Ryan and Steve A. Schneider. Prêt à voter with re-encryption mixes. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS 2006: 11th European Symposium on Research in Computer Security*, volume 4189 of *Lecture Notes in Computer Science*, pages 313–326, Hamburg, Germany, September 18–20, 2006. Springer, Heidelberg, Germany. Cited on page 27.

[RS21] Lior Rotem and Gil Segev. Non-malleable Vector Commitments via Local Equivocability. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography*, pages 415–446, Cham, 2021. Springer International Publishing. Cited on page 2.

[Rya08] P. Y. A. Ryan. Prêt à Voter with Paillier Encryption. *Math. Comput. Model.*, 48(910):16461662, 2008. Cited on page 27.

[SAB^{+}19] Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *ISOC Network and Distributed System Security Symposium – NDSS 2019*, San Diego, CA, USA, February 24-27, 2019. The Internet Society. Cited on page 80.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. Cited on page 2.

[San20] Olivier Sanders. Efficient redactable signature and application to anonymous credentials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 628–656, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany. Cited on pages 80, 81, 82, 108, and 110.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. Cited on pages 58 and 123.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. Cited on page 29.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949. Cited on page 16.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. Cited on page 10.

[Sho01] Victor Shoup. A Proposal for an ISO Standard for Public Key Encryption. `https://www.shoup.net/papers/iso-2_1.pdf`, 2001. [Online; accessed 01-April-2022]. Cited on pages 2 and 3.

[Tan12] Qiang Tang. Public key encryption supporting plaintext equality test and user-specified authorization. *Security and Communication Networks*, 5(12):1351–1362, 2012. Cited on page 30.

[TG20] Syh-Yuan Tan and Thomas Groß. MoniPoly - an expressive q-SDH-based anonymous attribute-based credential system. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 498–526, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. Cited on pages 80, 81, 82, 83, 91, 109, and 110.

[Tha19] Steve Thakur. Batching non-membership proofs with bilinear accumulators. Cryptology ePrint Archive, Report 2019/1147, 2019. `https://eprint.iacr.org/2019/1147`. Cited on pages 82, 83, and 89.

[Tur36] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936. Cited on page 9.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. Cited on page 82.

[YTHW10] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. Probabilistic public key encryption with equality test. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 119–131, San Francisco, CA, USA, March 1–5, 2010. Springer, Heidelberg, Germany. Cited on page 30.

[Zur13] IBM Research Zurich. Specification of the identity mixer cryptographic library v2.3.0., 2013. Cited on pages 5, 80, 81, 112, and 113.

RÉSUMÉ

Cette thèse étudie la malléabilité dans le contexte du chiffrement à clé publique et des signatures numériques, en présentant les avancées et les applications des technologies améliorant la confidentialité.
La première partie aborde le problème de l'égalité générique des textes en clair et les preuves d'inégalité. Étant donné deux textes chiffrés générés par un schéma de chiffrement à clé publique, le problème de l'égalité des textes chiffrés consiste à déterminer si les textes chiffrés contiennent la même valeur. Parallèlement, le problème de l'inégalité du texte clair consiste à déterminer s'ils contiennent une valeur différente. Les travaux précédents se sont concentrés sur la construction de nouveaux schémas ou sur l'extension de schémas existants afin d'inclure le support de l'égalité/inégalité du texte en clair. Nous proposons des preuves génériques et simples à connaissance zéro pour les deux problèmes, qui peuvent être instanciées avec divers schémas de chiffrement. Pour ce faire, nous formalisons les notions liées à la malléabilité dans le contexte du chiffrement à clé publique et proposons un cadre de définition pour le chiffrement générique aléatoire, que nous utilisons pour construire nos protocoles.
La partie suivante est consacrée aux signatures préservant la structure sur les classes d'équivalences, le principal élément constitutif des parties suivantes. Initialement, nous proposons des constructions nouvelles et plus efficaces sous des hypothèses standard. Ensuite, nous construisons un schéma d'accréditation établi sur les attributs sous des hypothèses standard, qui étend les travaux précédents de plusieurs façons. Nous améliorons notamment l'expressivité, les compromis d'efficacité et proposons une notion de dissimulation de l'émetteur qui permet aux détenteurs de lettres de créance de cacher l'identité de l'émetteur pendant les utilisations.
La dernière partie est consacrée à la présentation de Protego, un nouveau schéma d'accréditation pour les blockchains à autorisation. Il s'appuie sur les contributions précédentes et bien qu'il soit discuté dans le contexte des blockchains à autorisation, il peut également être utilisé dans d'autres contextes. Pour démontrer l'aspect pratique de Protego, nous fournissons un prototype et des benchmarks montrant que Protego est plus de deux fois plus rapide que les approches de l'état de l'art basées sur Idemix, le schéma d'accréditation le plus largement utilisé pour les blockchains à autorisation.

MOTS CLÉS

Cryptographie à clé publique, signatures numériques, Preuve à divulgation nulle de connaissance, sécurité prouvable, informations d'identification anonymes.

ABSTRACT

This thesis studies malleability in the context of public-key encryption and digital signatures, presenting advances and applications to privacy-enhancing technologies.
The first part addresses the problem of Generic Plaintext Equality and Inequality Proofs. Given two ciphertexts generated with a public-key encryption scheme, the problem of plaintext equality consists in determining whether the ciphertexts hold the same value. Similarly, the problem of plaintext inequality consists in deciding whether they hold different values. Previous work has focused on building new schemes or extending existing ones to include support for plaintext equality/inequality. We propose generic and simple zero-knowledge proofs for both problems, which can be instantiated with various encryption schemes. We do so by formalizing notions related to malleability in the context of public-key encryption and proposing a definitional framework for Generic Randomisable Encryption, which we use to build our protocols.
The next part turns to Structure-Preserving Signatures on Equivalence Classes, the main building block of subsequent parts. First, we propose new and more efficient constructions under standard assumptions. Then, we build an anonymous attribute-based credential (ABC) scheme under standard assumptions, which extends previous work in several ways. We improve expressiveness, provide efficiency trade-offs and propose an issuer-hiding notion that allows credential holders to hide the issuer's identity during showings.
The last part is devoted to presenting Protego, a new ABC scheme for permissioned blockchains. It builds upon the previous contributions, and although it is discussed in the context of permissioned blockchains, it can also be used in other settings. To show the practicality of Protego, we provide a prototype implementation and benchmarks showing that Protego is more than twice faster than state-of-the-art approaches based on Idemix, the most widely used ABC scheme for permissioned blockchains.

KEYWORDS

Public-key Cryptography, Digital Signatures, Zero-Knowledge Proofs, Provable Security, Anonymous Credentials.