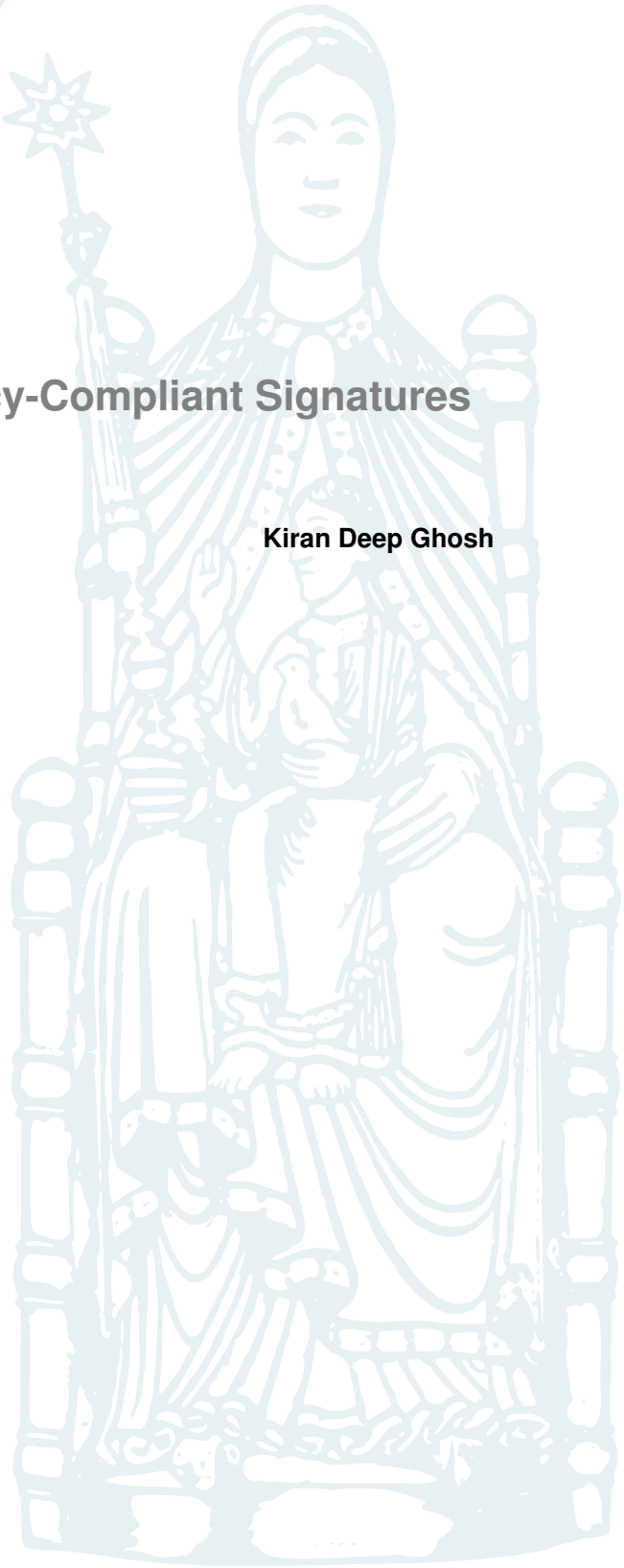
TI-uIPCS

Threshold-Issuance Un-linkable Policy-Compliant Signatures

Kiran Deep Ghosh



1st July 2025

KU LEUVEN

COSIC

TI-ulPCS: Threshold-Issuance - Un-linkable Policy Compliant Signatures

Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Technology
in
Cryptology and Security

by

Kiran Deep Ghosh
Roll No: CrS2310

Daily Supervisor

Dr. Mahdi Sedaghat

Promoters

Prof. Dr. Bimal Kumar Roy
Prof. Dr. ir. Bart Preneel

Indian Statistical Institute
Kolkata – 700108, India

1st July 2025

To my family.

Acknowledgments

I want to show my highest gratitude to Prof. Dr. Bimal Kumar Roy and Prof. Dr. Ir. Bart Preneel for giving me the opportunity to work at COSIC, KU Leuven. I am deeply grateful for their guidance, continuous support and encouragement throughout the journey.

I'm very grateful to my daily supervisor, Dr. Mahdi Sedaghat, for his invaluable guidance and insightful discussions. His problem-solving strategy and his necessities for critical thinking have not only shaped my research but also touched the manner in which I undertake problem-solving in life. I'm also truly grateful to Pela Noe for her support during my stay in Leuven and for the Easter chocolates. I would like to extend my sincere thanks to Prof. Dr. Daniel Slamanig, Prof. Dr. Sherman S. M. Chow, and Dr. Mahdi Sedaghat for co-authoring what could be my first academic publication.

My deepest thanks to all the teachers of the Indian Statistical Institute, for their valuable suggestions and discussions throughout the course which added an important dimension to my research work.

Finally, I am deeply grateful to my parents and my elder brother for their unwavering support. I would also like to sincerely thank all my friends; to save space and printing costs, I have securely hashed your names using SHA-256 (2c5aa62215f1aa12c04030b763d37db3806e4b02c765b4d7896764e9390271b3). I will reveal the names later, or you can try to break the one-wayness and figure them out yourselves. My heartfelt thanks go out to everyone I may have inadvertently left out.

Kiran Deep Ghosh
Indian Statistical Institute
Kolkata-700108, India

Abstract

Digital signatures, as a strong cryptographic primitive, ensure the authenticity and integrity of signed messages. On one hand, no one can forge a verifiable signature without knowing the secret key, on the other hand, any correctly formed signature is always verifiable under the public key of the signer. Besides signing digital data, they serve as foundational components in more advanced cryptographic systems, including blind signatures, group signatures, direct anonymous attestation, e-cash, e-voting protocols, adaptive oblivious transfer, anonymous credential schemes, Policy-Compliant Signatures, etc.
Policy-Compliant Signatures (PCS) enable the enforcement of joint policies between the signer and the verifier. A signature of this type is valid if not only it is correctly signed by the actual signer but also the attributes of both the signer and verifier fulfill a predefined policy. A PCS scheme allows a central authority to enforce a global policy by issuing keys tied to user attributes without revealing the attributes or policy. Unlinkable PCS (ulPCS) strengthens PCS properties by ensuring that signatures generated by the same signer remain unlinkable. However, both PCS and ulPCS rely on a single issuer, which introduces a single point of failure.
In this thesis, we study the concept of Threshold-Issuance Unlinkable PCS (TI-ulPCS). This cryptographic primitive builds on threshold cryptography to distribute the trust among multiple issuers, ensuring that a predefined threshold of issuers must collaborate to issue keys, without any single issuer having full control. We begin by defining and constructing Threshold-Issuance Predicate Encryption (TI-PE), which supports both attribute-hiding and blind-issuance of credentials. We achieve blind-issuance through commitment schemes combined with zero-knowledge proofs. For signing, we use Non-interactive Threshold Structure-Preserving Signatures on Equivalence Classes (NI-TSPS-EQ) and employ a threshold digital signature instead of a single-signer digital signature.

Keywords. Digital Signature; Policy Compliant Signature (PCS); Threshold Structure Preserving Signature (TSPS); Threshold Structure Preserving Signature on Equivalence Class (TSPS-EQ), Unlinkable Policy Compliant Signature (ul-PCS).

Contents

List of Figures

CHAPTER 1
Introduction

1.1 MOTIVATION

With the emergence of our current digital era, people use the Internet on a regular basis to perform many things such as socialization, business, education, entertainment and finance. With online life, concerns about privacy, security, and the authenticity of information have become more pressing. One of the biggest challenges is ensuring that all types of messages and transactions on the Internet are both trustworthy and privacy-preserving for users. This is where cryptography becomes important. It offers techniques and tools for the protection of data, enabling us to protect our own data. As we share more and more personal information on the Internet, be it through social media, banking applications for our mobile phones, or digital profiles, the need for its protection grows. This is an issue that is commonly called *data privacy*.

To protect the user's data governments have tried to respond with legislation. For example, the General Data Protection Regulation (GDPR) [KBDD20] was enacted to give people more control of personal data. In India, for example, despite not having a strong privacy law, massive breaches have exposed biometric information of Aadhaar (India's national ID), which is reportedly being sold for less than 6 euros on Whatsapp. Even in regions where data protection laws exist, as is the case with Facebook, huge amounts of personal information have leaked and been made public in disregard for such laws [Inf25]. Apparently, we need more than legislation—we need *privacy by design* [Sha12]. That is, building systems from scratch with privacy as a design element and not an afterthought. Traditional sanctions like fines are unproductive in bringing big tech to book. What we need are solid cryptographic methods that keep information secure by default and provide users with control of their own information. We require cryptographic mechanisms that provide privacy by design, where the system incorporates privacy from the onset. Outdated measures like fines are not typically effective at stopping huge technology firms from misusing information.

When it comes to online finance, traditional banks have a big weakness — all the user data is stored in one place, making it easier for someone to collect or misuse it. As people want more control over how their financial data is used, decentralization has become an important tool in shaping modern financial systems. One major example of this is Bitcoin [Nak09], a digital currency that runs on a decentralized network. Rather than directly opposing the traditional banking model, Bitcoin offers an alternative framework in which users can transact openly and directly with one another, without relying on intermediaries. However, this openness introduces significant trade-offs. Despite common assumptions, Bitcoin does not guarantee strong privacy. Its public blockchain records all transactions permanently, making them accessible to anyone. Over time, through network analysis and external data, it is often possible to link pseudonymous addresses to real-world identities. This raises critical concerns related to privacy, defined as control over one's personal information, and anonymity, the ability to act without revealing one's identity. In this regard, widely used cryptocurrencies such as Bitcoin and Ethereum [But13] fall short of providing meaningful

privacy or anonymity guarantees.On the other hand, systems that give people more privacy also tend to make it harder for the authorities to monitor things and enforce the law. For example, Tornado Cash [Wik25] ensured powerful privacy through decentralized smart contracts [DZP+22] and zero-knowledge proofs [For87, GMW87]. But hackers abused these anonymity features to do mass money laundering. Most of the users employed a centralized interface with no protection, which left them vulnerable to sanctions [NS23]. This tension creates challenges for regulators, auditors, and compliance professionals, especially when such systems are used to conceal illicit activity.

To address this issue, the concepts of accountability and auditability must be introduced into decentralized financial systems [CBC21]. Accountability within anonymous payment systems is making sure that people can stay private and still be held accountable should they break the rules, using special cryptographic tools to be able to do so. Auditability ensures that, within a given legal framework, authorized parties can request verifiable information about particular transactions. These mechanisms aim to preserve the benefits of decentralization while enabling necessary oversight and trust in financial applications. Practically, auditability needs to work after the fact, only enforced if a transaction breaks the system's normal rules. This is the idea of *accountable privacy* [CBC21, GGM16]—a system which attempts to balance user privacy with auditing and legally justifiable inspection of some activity. This also comes with some trade-offs. The same technology used to allow regulators to spot bad actors can be used to monitor good users at a cost that undermines the privacy guarantees for all parties. In response, new cryptographic primitives have developed that seek to "curb the auditor's powers" without giving up verifiable monitoring. For instance, "zero-knowledge proofs (ZKPs)" allow users to prove compliance with a policy without revealing underlying transaction data. Similarly, "functional commitments" [LRY16] can be employed to commit to transaction metadata so that specific attributes (e.g., compliance flags) are selectively disclosed under exactly specified legal conditions. Another enticing primitive is the use of "traceable ring signatures" [FS07] that provide anonymity in a group but deanonymization in the event that a member is shown to have acted in a way that was problematic. Although such cryptographic tools are hard to apply practically in an efficient way, one useful method that has been introduced is Policy-Compliant Signatures (PCS) [BMW21], put forward by Badertscher *et al.* in 2021. PCS allows a transaction to be verified only if sender and receiver attributes satisfy a globally specified policy, providing fine-grained control on who may exchange.

In the broader privacy and compliance environment, PCS have been a future cryptographic candidate for balancing user anonymity with enforceable policies. PCS is not applicable to fully anonymous decentralized anonymous payment (DAP) [GGM17] protocols like Zcash [KYMM18] or Monero [BFV19]—where transactions are made specifically unlinkable—but does suit very well in compliance-based decentralized payment protocols, where some form of identity or policy linkage is acceptable. PCS breaks the traditional trade-off between supporting user privacy and providing legal or institutional requirements. It allows users to assert compliance with specific policies—e.g., transaction size limitations, blacklisted destination limits, or jurisdictional boundaries—without revealing personal identifiers or transaction contents.

The method is particularly valuable in systems where regulators require assurances of legal behavior but where users still demand minimal disclosure. PCS enables the enforcement of such policies directly at the cryptography level without relying on third-party auditors or revealing sensitive metadata. Such assurances are not, however, cost-free. As Badertscher *et al.* [BSW24] explain, linking a user public key to credentials from an authority introduces a linkability problem: multiple transactions using the same credential can be linked, which can make it possible for observers—perhaps even auditors—to build behavioral profiles or deduce patterns of behavior. This is a limitation on PCS-based systems that claim to support strong anonymity. In opposition, recent work is exploring unlinkable credential systems, transient key schemes, and anonymous attestation methods to divorce identity from transaction metadata without compromising policy adherence.

The Unlinkable Policy-Compliant Signature (ul-PCS) scheme presented by Badertscher, Sedaghat, and Waldner [BSW24] builds on the PCS system immediately by addressing its most extreme privacy limitation—linkability. While PCS links transactions to long-lived user credentials to enable policy compliance, this design allows multiple operations from the same user to become linked. ul-PCS prevents this issue by enabling users to create new, unlinkable public keys for each transaction while their credentials continue to satisfy pre-specified policies. This innovation is designed to provide accountability without third-party verifiers or centralized control. Policy compliance is cryptographically verified in each signature, without the need for external validation. Users are therefore able to prove a transaction meets regulatory requirements—e.g., jurisdictional limits or size limits on transactions—without revealing identity or attribute values. Through this process, ul-PCS is true to the design principles of DAP systems: privacy-preserving, non-custodial, and regulation-aware.

Lastly, ul-PCS argues for privacy-by-design systems that can enforce sophisticated compliance burdens without sacrificing user anonymity or decentralization—a significant step towards bridging cryptographic privacy and real-world policy enforcement. One key innovation of ul-PCS is that users can re-randomize their public keys on a per-transaction basis, such that every action is unlinkable but still associated with a single set of verifiable credentials. Crucially, this re-randomization is non-interactive—no requirement to go back to the issuing authority in order to maintain unlinkability, and thus no scalability or centralization issue. Through co-multiplying credential-bound policy into signature generation, ul-PCS enjoys fine-grained enforcement without permanent identifiers. Although ul-PCS does not immediately improve today's privacy coins like Monero or Zcash, it is a huge step towards building future decentralized payment systems that can meet users' privacy requirements and regulations and rules. It demonstrates that accountability and unlinkability are not mutually exclusive—and that privacy-preserving payment systems can be made to meet real-world compliance demands.

While ul-PCS strengthens the privacy-accountability trade-off for decentralized payment systems, it still assumes the presence of a centralized credential authority distributing user attributes. A such central trust point may become a weak spot—censorship threats, compromise, or misuse. To eliminate this bottleneck, threshold cryptography [DF90] offers a powerful extension: distributing the credential issuance process over a number of independent entities. In a threshold credential system, the master signing key with which credentials are emitted is shared among n issuers, and a credential is validly generated only when a threshold t They all participate together. This ensures that no single issuer can issue, deny, or forge credentials on its own, thereby excluding centralized control without weakening privacy guarantees. Credentials remain interoperable with the ul-PCS infrastructure, so users can still prove compliance with policies without revealing identity—but now with the added security and censorship resistance provided by distributed trust. By pairing threshold-issuance with policy-compliant signatures that are unlinkable, the system is not just accountable and privacy-preserving, but also decentralized and resilient. This gets us one step closer to cryptographic infrastructures where trust is minimized by design, as opposed to simply required by law or institutional promises.

1.2 OVERVIEW OF CONTRIBUTIONS

We introduce an enhanced cryptographical primitive called threshold issuance unlinkable policy compliant signatures (TI-ulPCS), a stronger version of ul-PCS. We aim to combine the privacy benefits of ul-PCS with the security of threshold cryptography. Unlike the original ul-PCS model, which depends on a single issuer to provide user credentials, our new scheme uses a fixed set of issuers who work together to issue credentials. This allows us to keep the unlinkability and policy enforcement features of ul-PCS while removing the central authority. It is designed to give users strong privacy and security guarantees, even in decentralized environments. Unlinkable Policy-Compliant Signatures (ul-PCS) enable the enforcement of policies with

different complexity and scope towards the objective of accountable privacy. The policies are of different structures and enforcement types, for which ul-PCS is designed.

Generic Policies allow for arbitrary joint checks between sender and receiver attributes on the basis of a predicate $F(x, y)$. The inner-product (IP) predicate is most quoted as an example, and this predicate can express complex conditions such as set membership, conjunctive or disjunctive normal form (CNF or DNF) formula, and threshold clauses. Separable Policies adopt a more formalized approach, with the predicate splitting into distinct sender and receiver independent factors: $F(x, y) = S(x) \wedge R(y)$. A common example might be that a sender must be passed Know Your Customer (KYC) checks, $S(x)$, whereas the receiver can be simply alone with $R(y) = 1$. This form is particularly handy to define transaction types in systems like Zcash based upon the abilities or roles of each participant. Role-Based Access Control (RBAC) Policies define permissible interactions between roles using a matrix whose entries indicate if one role can transact with another. For more expressive policies based on predicate encryption, RBAC can be efficiently implemented under cryptographic accumulators, particularly pairing-based signatures.

These diverse policy types are accommodated in ul-PCS in the sense of embracing a modular cryptographic model. Generic policies are typically enforced with predicate encryption (PE), digital signatures, pseudo-random functions (PRFs), and non-interactive zero-knowledge (NIZK) proofs. A common instantiation involves the use of inner-product PE due to expressiveness in policy logic. For separable policies, using regular public key encryption instead of PE improves efficiency without impacting functionality. RBAC policies employ accumulators instead of PE to guarantee enforcement.

Security and privacy are offered by various NIZK proof systems like Sigma protocols, Groth-Sahai (GS) proofs, and range proofs. Dodis-Yampolskiy PRF is used for its security and performance, and structure-preserving signatures and BLS are used if there is a need for compatibility with GS-friendly relations. This modular and adaptive approach allows ul-PCS to offer high performance across a wide spectrum of policy requirements, making it deployable in a wide array of privacy-preserving systems. To extend the functionalities of ul-PCS into a threshold setting, we propose distributed counterparts of the underlying cryptographic primitives. Specifically, instead of relying on digital signatures (DS) from a single issuer, we construct threshold digital signatures (TDS), enabling a set of signers to jointly issue signatures, tolerating partial corruption. This threshold approach is systematically applied to all required primitives: we introduce non-interactive threshold structure preserving signatures on equivalence class (NI-TSPS-EQ) to replace structure preserving signatures on equivalence class (SPS-EQ), threshold issuance (predicate only) predicate encryption (TI-PE) which is a modified version of predicate encryption from [OT12]. These constructions collectively enable the design of a Threshold-Issuance unlinkable PCS (TI-ulPCS) in chapter 4 with robust security in the presence of partial trust. Notably, we go beyond TSPS-EQ and construct a more advanced non-interactive TSPS-EQ (NI-TSPS-EQ) in chapter 3 achieving, to the best of our knowledge, the first such construction in the literature.

1.2.1 Non-Interactive Threshold Structure-Preserving Signatures on Equivalence Class

In the state-of-the-art on threshold structure-preserving signatures (TSPS) [CKP+23, MMS+24, ANPKT24] and structure-preserving signatures on equivalence-classes (SPS-EQ) [FHS14, KSD19], only a few schemes incorporate efficiency, privacy and strong security guarantees. A key challenge in thresholdizing structure-preserving signatures over equivalence-classes (SPS-EQ) is maintaining non-interactivity, which is critical for efficiency and composability in advanced protocols. Our construction overcomes this barrier by achieving a non-interactive variant of TSPS-EQ, where signers can independently contribute to a joint signature without requiring interaction or sequential communication. Some schemes demonstrate robust security but are inefficient for users, while others offer better efficiency at the cost of significant security compromises. Moreover, while some schemes are proven secure in the Generic Group Model (GGM) [FHS14], others provide security proofs in the standard model [KSD19, CLPK22].

Note. Given the recent impossibility result by Bauer et al. [BFR24a] showing that equivalence class signatures can't be proven secure under non-interactive assumptions, the construction by Fuchsbauer et al. [FHS14] now seems impossible to prove secure in the standard model. Although their approach uses the CRS model, which is a weaker assumption than the standard model, the CRS doesn't help in verifying class membership.

However, even though more than ten years have passed since the introduction of TSPS and SPS-EQ, the literature still lacks Threshold Structure-Preserving Signatures on Equivalence-Classes in non-interactive setting (NI-TSPS-EQ) as a natural extension. We start by finding a good foundation for building an SPS-EQ that avoids non-linear operations, so it can later extend to a TSPS-EQ without extra communication. Our initial attempt involved the schemes in paper [MBS+25], but we found it unsuitable for threshold settings due to its reliance on multiplication operations on the shared values, which are not easily avoided. We experimented with using monomials to build commitments, but this led to trivial forgery attacks. Attempts to add randomness to mitigate this issue ultimately circled back to the same threshold-incompatibility. We then explored an indexing-based approach, as examined in previous works such as [CKP+23]; however, this introduced challenges with the random oracle model, particularly when trying to define a monomial set without relying on a single fixed basis. Also it relies on indexed Diffie-Hellman message spaces and does not operate on equivalence-classes, which is a key requirement for our construction. Subsequently, we examined a paper [MBG+23] proposing a multi-issuer signature scheme, but it was not threshold-friendly. Abe *et al.* [ANPKT24] built on the original scheme from [FHS14] and used a multiparty protocol to create signatures. While their approach achieved TSPS-EQ in an interactive setting, the verification process requires interacting with each signer, which makes it inefficient for our use case. We then turned our attention to the paper [MMS+24], which presents another SPS scheme that works well with threshold settings. The main challenge we faced was enabling signing over equivalence-classes. This scheme is based on an extended version of the KPW15 SPS scheme, with some small changes to avoid non-linear operations. However, since it doesn't currently support signing over equivalence-classes, we had to make additional changes.

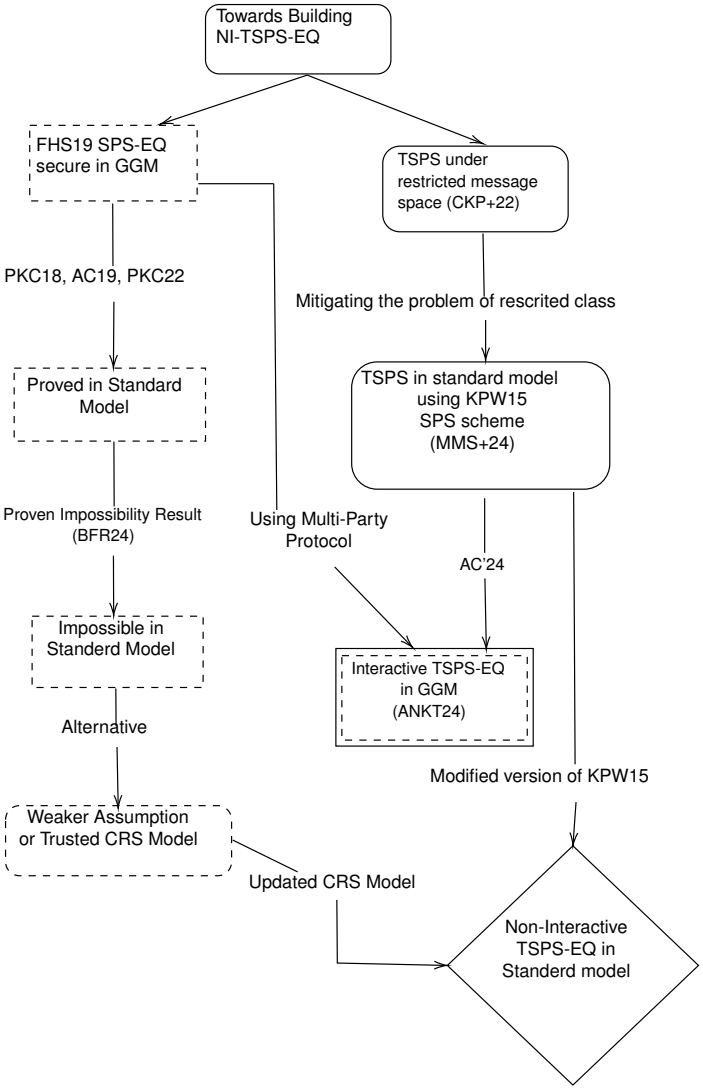


Figure 1.1: NI-TSPS-EQ Related Works.

To summarize, refer to Figure 1.1: the branches on the left illustrate the evolution of signatures on equivalence-classes, while the branches on the right depict the development of threshold structure-preserving signatures over the years.

1.2.2 Threshold Issuance Predicate Encryption (TI-PE)

Our initial attempt for the threshold issuance predicate encryption starts with reviewing the existing constructions from the literature. While most of the constructions are doing multi-authority predicate encryptions (PE) [vdKPJ20] or attribute based encryptions (ABE) [BSW07], where the system has more than one central authority, and each of them can give the users a secret key for some specific attribute. For our proposed TI-uIPCS schemes we need one (predicate only) PE which will be similar to threshold signatures. Our scheme should have multiple authorities and a sufficient number of them is required to generate a valid key. Let's assume that the user has attributes $(x_1, x_2, \ldots, x_n)$ in a (t, n) setting, the users gets a valid key iff at least t of issuers agree on that. As described in [vdKPJ20], Kamp *et al.* have achieved multi-authority PE (MA-PE) by introducing a multi-authority admissible pair encoding scheme (MA-PES). Based on this encoding, they showed that with a general conversion algorithm, different kind of predicate encryptions can be seamlessly combine into a MA-PE framework. While having a freedom of using various predicates, this construction uses composites order group elements. Based on our signature scheme, using composite

order group elements for predicate encryption will not be compatible for TI-ulPCS scheme.

We then turned our attention to the construction of [OT12] (predicate only) PE which is used in the original ul-PCS scheme. While this construction is suitable for single authority case in ul-PCS, we have modified it for multiple authorities. To modify it into threshold issuance setting we have to employ multi-party protocols (MPC) for the interactions between all the certifications authorities (CA). Having said that, as there are no constructions for TI-PE without interaction among the CAs so far, this remains an open problem in the literature.

1.3 OUTLINE OF THIS THESIS

This thesis consists of the following chapters:
Chapter 2 introduces the notation adopted throughout the thesis and revisits the definitions of key cryptographic primitives along with their corresponding security properties. It also examines the mathematical assumptions underlying the hard problems that form the basis for the constructions presented in the following chapters.
Chapter 3 talks about the contributions of concerning non-interactive threshold structure-preserving signatures on equivalence class(NI-TSPS-EQ). This is the one of the key building blocks of this thesis.
Chapter 4 presents the main contributions of this thesis concerning threshold issuance unlinkable policy compliant signatures (TI-ulPCS) and its use cases. It starts by discussing the motivation for these constructions, identifies the key design challenges.
Chapter 5 concludes the thesis and outlines potential directions for future work.

CHAPTER 2
Preliminaries

In this section, we begin by presenting key background concepts that are essential for understanding the main contributions of this thesis. Next, we examine several well-established cryptographic primitives and their security properties, which serve as the foundational building-blocks for the constructions explored in the following chapters.

2.1 BASIC NOTION

Let $\kappa \in \mathbb{N}$ be the security parameter, and its unary encoding is given by 1^κ. An algorithm $\mathcal{A}$ is considered to be running in probabilistic polynomial time (PPT) if there is a polynomial $p(\cdot)$ such that the execution time of the algorithm is less than or equal to $p(|x|)$ for any input x. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all polynomials $f(x)$ with positive coefficients, there exists a threshold x_0 such that for all $x > x_0$, it holds that : $\mathsf{negl}(\kappa) < 1/f(x)$. For the sake of clarity, the parameter κ may be omitted when the context allows. We use the notation $[1, n]$ to denote the set $\{1, \ldots, n\}$ for any positive integer n, and $\{0,1\}^*$ to refer to the set of all finite binary strings. The symbol "=" to signify equality, and the value assignment is represented by ":=". Random sampling of a probabilistic algorithm A is represented as $a \leftarrow A$, omitting the internal randomness of A. When two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable by any efficient algorithm, we say $\mathcal{D}_1 \approx_c \mathcal{D}_2$. Vectors and matrices are represented as $\vec{r}$ and $\mathbf{A}$, respectively. Let q be a prime number of κ-bit length. We define $\mathbb{Z}_q = \mathbb{Z}\backslash q\mathbb{Z}$ as the ring of integers modulo q, and its group of invertible elements is written as $\mathbb{Z}_q^*$. The polynomial ring over $\mathbb{Z}_q$ is denoted $\mathbb{Z}_q[X]$. Consider a cyclic group $\mathbb{G} = \langle G \rangle$ of prime order q, generated by an element G, with the identity element represented by $1_\mathbb{G}$. The set $\mathbb{G}^*$ contains all non-identity elements of $\mathbb{G}$, i.e.,$\mathbb{G}\backslash\{1_\mathbb{G}\}$.

In this thesis, group operations are traditionally expressed in additive notation. We employ an implicit notation for group elements: for a scalar $\alpha \in \mathbb{Z}_q$ and a group element $\mathbf{M} \in \mathbb{G}$, the implicit values of α with respect to $\mathbf{M}$ is denoted by,

$$[\alpha]_\mathbf{M} := \alpha\mathbf{M} \in \mathbb{G}.$$

It is extended naturally to matrices. For a matrix $\mathbf{A} = (\alpha_{ij}) \in \mathbb{Z}_q^{m\times n}$, its implicit representation in the group $\mathbb{G}$ with respect to $\mathbf{M}$ is denoted by $[\mathbf{A}]_\mathbf{M}$, and we have

$$[\mathbf{A}]_\mathbf{M} = \begin{pmatrix} \alpha_{1,1}\mathbf{M} & \cdots & \alpha_{1,n}\mathbf{M} \\ \alpha_{2,1}\mathbf{M} & \cdots & \alpha_{2,n}\mathbf{M} \\ \vdots & \ddots & \vdots \\ \alpha_{m,1}\mathbf{M} & \cdots & \alpha_{m,n}\mathbf{M} \end{pmatrix}.$$

To simplify, when $\mathbf{M}$ is a generator of $\mathbb{G}$, i.e. G, we use $[\alpha]$ instead of $[\alpha]_G$.

2.2 BILINEAR GROUPS

Pairings, or bilinear maps [BMS03], are efficiently computable functions defined over three cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, all of prime order p. These groups are typically subgroups of elliptic curve groups or their extensions, with corresponding generators $\mathsf{P}_1 \in \mathbb{G}_1$, $\mathsf{P}_2 \in \mathbb{G}_2$, and $\mathsf{P}_T \in \mathbb{G}_T$.

Note. A pairing is said to be $symmetric$ or Type-I pairing if $\mathbb{G}_1 = \mathbb{G}_2$, otherwise $asymmetric$. If $\mathbb{G}_1 \neq \mathbb{G}_2$ but there exists an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ then we call it type-II pairing. Lastly, type-III pairing if $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no known homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

In this thesis, we focus exclusively on asymmetric pairings, particularly those instantiated over Type-III bilinear groups.

Definition 1 (Bilinear Groups)**.** *Formally, an asymmetric bilinear group generator* $\mathsf{ABSGen}(1^\kappa)$ *outputs a tuple* $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{P}_1, \mathsf{P}_2, e)$*, where* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *are cyclic groups of the same prime order* p*. Moreover, there is no efficiently computable homomorphism between* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, which characterizes the setting as Type-III.* P_1 *and* P_2 *are the generators of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, respectively, where* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ *is an efficiently computable (non-degenerate) bilinear map with the following properties:*

- $\forall\, a, b \in \mathbb{Z}_p$, $e([a]_1, [b]_2) = [ab]_T = e([b]_1, [a]_2)$,
- $\forall\, a, b \in \mathbb{Z}_p$, $e([a+b]_1, [1]_2) = e([a]_1, [1]_2)e([b]_1, [1]_2)$,

where we use an implicit representation of group elements, in which for $\zeta \in \{1, 2, T\}$ and an integer $\alpha \in \mathbb{Z}_p$, the implicit representation of integer α in group $\mathbb{G}_\zeta$ is defined by $[\alpha]_\zeta = \alpha\mathsf{P}_\zeta \in \mathbb{G}_\zeta$, where $\mathsf{P}_T = e(\mathsf{P}_1, \mathsf{P}_2)$. To be more general, the implicit representation of a matrix $\mathbf{A} = (\alpha_{ij}) \in \mathbb{Z}_p^{m \times n}$ in $\mathbb{G}_\zeta$ is defined by $[\mathbf{A}]_\zeta$ and we have:

$$[\mathbf{A}]_\zeta = \begin{pmatrix} \alpha_{1,1}\mathsf{P}_\zeta & \cdots & \alpha_{1,n}\mathsf{P}_\zeta \\ \alpha_{2,1}\mathsf{P}_\zeta & \cdots & \alpha_{2,n}\mathsf{P}_\zeta \\ \vdots & \ddots & \vdots \\ \alpha_{m,1}\mathsf{P}_\zeta & \cdots & \alpha_{m,n}\mathsf{P}_\zeta \end{pmatrix} .$$

For two matrices $\mathbf{A}$ and $\mathbf{B}$ with matching dimensions we define $e([\mathbf{A}]_1, [\mathbf{B}]_2) = [\mathbf{AB}]_T$. To be more precise,

$$e([\mathbf{A}]_1, [\mathbf{B}]_2) = e\left(\begin{pmatrix} \alpha_{1,1}\mathsf{P}_1 & \cdots & \alpha_{1,n}\mathsf{P}_1 \\ \alpha_{2,1}\mathsf{P}_1 & \cdots & \alpha_{2,n}\mathsf{P}_1 \\ \vdots & \ddots & \vdots \\ \alpha_{m,1}\mathsf{P}_1 & \cdots & \alpha_{m,n}\mathsf{P}_1 \end{pmatrix} \cdot \begin{pmatrix} \alpha_{1,1}\mathsf{P}_2 & \cdots & \alpha_{1,m}\mathsf{P}_2 \\ \alpha_{2,1}\mathsf{P}_2 & \cdots & \alpha_{2,m}\mathsf{P}_2 \\ \vdots & \ddots & \vdots \\ \alpha_{n,1}\mathsf{P}_2 & \cdots & \alpha_{n,m}\mathsf{P}_2 \end{pmatrix} \right) = [\mathbf{AB}]_T$$

2.3 CRYPTOGRAPHIC ASSUMPTIONS

Security is usually based on the idea that certain problems are very hard to solve. These problems, called hardness assumptions, come in different types such that some are more commonly used, some are easier to check, and some involve interaction while others don't.

In cryptography, a standard assumption refers to a computational hardness assumption that has been extensively analyzed and validated over time, often serving as the foundation for the security of numerous cryptographic primitives and protocols. Traditionally, such well-established assumptions have been classified into two broad categories: generic assumptions, which are defined within idealized models, and concrete assumptions, which rely on specific hardness problems in standard computational settings.

Generic and Concrete assumptions. Historically, cryptographic schemes have been proven secure based on a few well-established computational assumptions. These assumptions can be partitioned among two groups, generic and concrete. While generic assumptions refer to the existence of primitives such as one-way functions, one-way permutations, and trapdoor functions, concrete assumptions are more specific and include, for example, the universal one-way function [Gol01], the Factoring and RSA assumptions [RSA78, Rab79], the Discrete Logarithm assumption in various groups [DH76], the Decisional Diffie-Hellman (DDH) assumption [DH76], and the Learning With Errors (LWE) assumption [Reg05], among others.

Beyond this traditional classification, cryptographic assumptions have been categorized through various other frameworks. For example, Naor [Nao03] introduced a distinction between falsifiable and non-falsifiable assumptions, whereas Goldwasser *et al.* [GK15] proposed a classification based on search complexity and decision complexity.

Falsifiable and non-falsifiable assumptions. Naor [Nao03] introduced the concept of falsifiable assumptions, which intuitively refers to assumptions that can be constructively shown to be false, if indeed they are. He proposed three levels of falsifiability: efficiently falsifiable, falsifiable, and somewhat falsifiable. Non-falsifiable assumptions are those where a challenger can not effectively check if the adversary has achieved their goal.

Below is a list of assumptions, which we'll use as the basis for the methods explained in the next chapters.

Definition 2 (Discrete Logarithm (DL) based problem)**.** *Let a cyclic group $\mathbb{G} = \langle \mathsf{P} \rangle$ of prime order p, the DL problem is hard if for all PPT adversaries $\mathcal{A}$, we have*

$$Pr[\forall\ Z \leftarrow \mathbb{G} \mid z \leftarrow \mathcal{A}(\mathsf{P}, Z) : Z = [z]] \leq \mathsf{negl}(\kappa)$$

Definition 3 ((q_1, q_2)-Discrete Logarithm Assumption)**.** *Given an asymmetric bilinear group description $\mathcal{G} := (p, \mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_T, \mathsf{P}_1, \mathsf{P}_2, e) \leftarrow \mathsf{ABSGen}(1^\kappa)$, the (q_1, q_2)-discrete logarithm assumption holds if for all PPT adversaries $\mathcal{A}$, we have:*

$$Pr\left[z \leftarrow \mathbb{Z}_p^*, z' \leftarrow \mathcal{A}(\mathcal{G}, [z, \ldots, z^{q_1}]_1, [z, \ldots, z^{q_2}]_2) : z' = z\right] \leq \mathsf{negl}(\kappa).$$

Decisional Diffie–Hellman Assumption. The hardness of computing discrete logarithms in large finite groups has underpinned many cryptographic protocols over the past decades, beginning with the Diffie–Hellman key exchange and extending to encryption, digital signatures, and more.
A fundamental assumption closely tied to the Diffie–Hellman key exchange is the Computational Diffie–Hellman (CDH) assumption.

CDH Problem : Given a group $\mathbb{G}$, a generator P of $\mathbb{G}$, and two elements $a = \mathsf{P}^x$, $b = \mathsf{P}^y \in \mathbb{G}$, where x and y are unknown, compute the value $c = \mathsf{P}^{xy} \in \mathbb{G}$.
And the **assumption** is that, any probabilistic polynomial time (PPT) algorithm solves the CDH problem only with negligible probability.

While the CDH assumption is a fundamental cryptographic assumption, it often falls short for proving strong security guarantees in some groups. CDH captures only a limited aspect of the problem's hardness, motivating the introduction of the Decisional Diffie–Hellman (DDH) assumption.

Definition 4 (Decisional Diffie-Hellman Assumption [Bon98])**.** *Given a cyclic group $\mathbb{G}$ of prime order p with generator P, the Decisional Diffie-Hellman (DDH) assumption holds if for all PPT adversaries $\mathcal{A}$, and uniformly random integers $x, y, z \leftarrow \mathbb{Z}_p^*$, we have:*

$$Adv_{\mathcal{A}}^{DDH}(\kappa) := |\epsilon_1 - \epsilon_0| \leq \mathsf{negl}(\kappa),$$

where $\epsilon_\beta := Pr[\mathcal{A}([x], [y], [xy + \beta z]) = 1]$.

Definition 5 (Collision Resistant Hash Function (CRHF) [PGV93])**.** *For a given security parameter κ and a family of functions $\mathcal{H} : \{0,1\}^{m(\kappa)} \rightarrow \{0,1\}^{n(\kappa)}$, where $n(\kappa) > m(\kappa)$. $\mathcal{H}$ is a family of CRHF, if for any $H \in \mathcal{H}$ it is difficult to find a pair (X_1, X_2) such that $X_1 \neq X_2$ and $H(X_1) = H(X_2)$.*

Matrix assumptions.

Definition 6 (Matrix Distribution)**.** *Let $k, \ell \in \mathbb{N}^*$ s.t. $k < \ell$. We call $\mathcal{D}_{\ell,k}$ a matrix distribution if it outputs matrices over $\mathbb{Z}_p^{\ell \times k}$ of full rank k in polynomial time. W.l.o.g, we assume the first k rows of matrix $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}$ form an invertible matrix. For $\ell = k+1$, we write $\mathcal{D}_k$ in short.*

Next, we recall the Matrix Decisional Diffie-Hellman assumption, which defines over $\mathbb{G}_\zeta$ for any $\zeta = \{1, 2\}$ and states two distributions $([\mathbf{A}]_\zeta, [\mathbf{Ar}]_\zeta)$ and $([\mathbf{A}]_\zeta, [\mathbf{u}]_\zeta)$, where $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}, \mathbf{r} \leftarrow \mathbb{Z}_p^k, \mathbf{u} \leftarrow \mathbb{Z}_p^\ell$ are computationally indistinguishable.

Definition 7 ($\mathcal{D}_{\ell,k}$-Matrix Decisional Diffie-Hellman ($\mathcal{D}_{\ell,k}$-MDDH) Assumption [EHK+17])**.** *For a given security parameter κ, let $k, \ell \in \mathbb{N}^*$ s.t. $k < \ell$ and $\mathcal{D}_{\ell,k}$ be a matrix distribution, defined in Definition 6. We say $\mathcal{D}_{\ell,k}$-MDDH assumption over $\mathbb{G}_\zeta$ for $\zeta = \{1, 2\}$ holds, if for all PPT adversaries $\mathcal{A}$ we have:*

$$Adv^{\mathsf{MDDH}}_{\mathcal{D}_{\ell,k},\mathbb{G}_\zeta,\mathcal{A}}(\kappa) = \left| \Pr\left[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_\zeta, [\mathbf{Ar}]_\zeta) = 1\right] - \Pr\left[\mathcal{A}(\mathcal{G}, [\mathbf{A}]_\zeta, [\mathbf{u}]_\zeta) = 1\right] \right| \leq \mathsf{negl}(\kappa) \ ,$$

where $\mathcal{G} \leftarrow \mathsf{ABSGen}(1^\kappa)$, $\mathbf{A} \leftarrow \mathcal{D}_{\ell,k}, \mathbf{r} \leftarrow \mathbb{Z}_p^k$ and $\mathbf{u} \leftarrow \mathbb{Z}_p^\ell$.

Definition 8 ($\mathcal{D}_k$-Kernel Matrix Diffie-Hellman ($\mathcal{D}_k$-KerMDH) Assumption [MRV16])**.** *For a given security parameter κ, let $k \in \mathbb{N}^*$ and $\mathcal{D}_k$ is a matrix distribution, defined in Definition 6. We say $\mathcal{D}_k$-KerMDH assumption over $\mathbb{G}_\zeta$ for $\zeta = \{1, 2\}$ holds, if for all PPT adversaries $\mathcal{A}$ we have:*

$$Adv^{\mathsf{KerMDH}}_{\mathcal{D}_k,\mathbb{G}_\zeta,\mathcal{A}}(\kappa) = \Pr\left[\mathbf{c} \in \mathsf{orth}(\mathbf{A}) \mid [\mathbf{c}]_{3-\zeta} \leftarrow \mathcal{A}(\mathcal{G}, [\mathbf{A}]_\zeta))\right] \leq \mathsf{negl}(\kappa) \cdot$$

The Kernel Matrix Diffie-Hellman assumption is a natural computational analog of the MDDH assumption. It is well-known that for all $k \geq 1$, $\mathcal{D}_k$-MDDH $\Rightarrow$ $\mathcal{D}_k$-KerMDH [KPW15, MRV16].

2.4 DIGITAL SIGNATURE

Digital Signatures (DS) [RSA78] are the digital analogue of handwritten signatures, which are traditionally employed to indicate consent, approval, or acknowledgment in various societal and legal contexts. For example, individuals may sign a physical document to authorize a financial transaction or to confirm the receipt of a delivered item. As the world increasingly shifts toward digital platforms, many of these physical interactions have been replaced by their electronic counterparts.

Definition 9 (Digital Signatures (DS))**.** *More formally, a DS over message space $\mathcal{M}$ consists of the following PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{DS.Setup}(\kappa)$*: The setup algorithm takes the security parameter κ as input and returns the set of public parameters* pp *as output.*
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp})$*: The key generation algorithm takes the public parameters* pp *as inputs. It then returns secret/verification keys* $(\mathsf{sk}, \mathsf{vk})$ *as output.*
- $\sigma \leftarrow \mathsf{DS.Sign}(\mathsf{pp}, \mathsf{sk}, M)$*: The signing algorithm as a probabilistic algorithm, takes* pp*, secret key,* sk*, and a message $M \in \mathcal{M}$ as inputs. It then returns a signature σ as output.*
- $0/1 \leftarrow \mathsf{DS.Verify}(\mathsf{pp}, \mathsf{vk}, M, \sigma)$*: The verification algorithm as a deterministic algorithm, takes* pp*, the verification key,* vk*, and message $M \in \mathcal{M}$ along with a signature σ as inputs. It then returns 1 (accept), if the aggregated signature is valid and 0 (reject), otherwise.*

A digital signature (DS) is considered secure if it meets two key properties: **Correctness** and **Existential Unforgeability under Chosen Message Attack (EUF-CMA)**. Correctness ensures that any valid signature can be verified using only the signer's public key. EUF-CMA, on the other hand, guarantees that only the signer can generate valid signatures, and any attempt to alter the signature or the associated message can be easily detected.

Definition 10 (Correctness)**.** *A digital signature is correct for all public parameters* $\mathsf{pp} \leftarrow \mathsf{DS.Setup}(1^\kappa)$*, key-pairs* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp})$ *and messages* $M \in \mathcal{M}$*, if:*

$$Pr[\mathsf{DS.Verify}(\mathsf{pp}, \mathsf{vk}, M, \mathsf{DS.Sign}(\mathsf{pp}, \mathsf{sk}, M)) = 1] \geq 1 - \mathsf{negl}(\kappa)$$

Definition 11 (Existential Unforgeability against Chosen Message Attack (EUF-CMA) [GMR87])**.** *Given the security parameter* κ*, public parameters* pp*, verification key* vk *such that* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{DS.KeyGen}(\mathsf{pp})$*, and oracle* $\mathcal{O}^{\mathsf{DS.Sign}(M)} \rightarrow \sigma$*, the goal of the EUF-CMA game for a DS, i.e.* $G^{EUF-CMA}$ *, is to find a pair* (M^*, σ^*) *such that*

- *Verify*$(\mathsf{pp}, \mathsf{vk}, M^*, \sigma^*) = 1$*, and*
- $M^* \notin \mathcal{Q}$*, where* $\mathcal{Q}$ *is the list of queried messages to oracle* $\mathcal{O}^{\mathsf{DS.Sign}(\cdot)}$*.*

A DS is called EUF-CMA-secure if for all PPT adversaries $\mathcal{A}$*:*

$$Adv_{\mathcal{A}}^{EUF-CMA}(\kappa) := Pr[G^{EUF-CMA}(\kappa) = 1] \leq \mathsf{negl}(\kappa).$$

Note. A signature scheme is **strongly unforgeable** if an adversary cannot produce a valid signature on any new message, nor generate a different valid signature on a previously signed message, even after seeing a valid signature for it.

2.5 UNLINKABLE POLICY COMPLIANT SIGNATURES

An ul-PCS [BSW24] scheme consists of five PPT algorithms: $Setup$, $KeyGen$, $RandKey$, $Sign$, and $Verify$. The $Setup$ algorithm generates the system parameters, $KeyGen$ generates user key pairs encoding attributes, $RandKey$ re-randomizes keys for unlinkability, $Sign$ creates signatures compliant with defined policies, and $Verify$ checks signature validity against those policies. The scheme ensures existential unforgeability, attribute-hiding, and unlinkability, making it suitable for privacy-sensitive applications.

Definition 12 (Unlinkable Policy-Compliant Signatures)**.** *Let* $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family of attribute sets and denote by* $\mathcal{X}_\lambda$ *the powerset of* X_λ*. Further let* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ *be a family of sets* $\mathcal{F}_\lambda$ *of predicates* $F \colon \mathcal{X}_\lambda \times \mathcal{X}_\lambda \rightarrow \{0, 1\}$*. Then an* unlinkable policy-compliant signature (ul-PCS) *scheme for the functionality class* $\mathcal{F}_\lambda$ *is a tuple of four PPT algorithms* $\mathsf{ULPCS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{RandKey}, \mathsf{Sign}, \mathsf{Verify})$*:*

$\mathsf{Setup}(1^n, F)$**:** *After taking a unary representation of the security parameter* n *and a policy* $F \in \mathcal{F}_\lambda$ *the algorithm a master public and secret key pair* $(\mathsf{mpk}, \mathsf{msk})$*.*

$\mathsf{KeyGen}(\mathsf{msk}, x)$**:** *After taking the master secret key* msk *and a set of attributes* $x \in \mathcal{X}_\lambda$ *the algorithm a public and secret key pair* $(\mathsf{vk}, \mathsf{sk})$*.*

$\mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk})$ *After taking the master public key* mpk *and a secret key* sk*the algorithm a new public-secret-key pair* $(\mathsf{vk}', \mathsf{sk}')$*.*

$\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_S, \mathsf{vk}_R, m)$**:** *After taking the master public key* mpk*, a sender secret key* sk_S*, a receiver public key* vk_R *and a message* m *the algorithm either a signature* σ *or* $\perp$*.*

$\mathsf{Verify}(\mathsf{mpk}, \mathsf{vk}_S, \mathsf{vk}_R, m, \sigma)$**:** *After taking the master public key* mpk*, a sender public key* vk_S*, a receiver public key* vk_R*, a message* m *and a signature* σ *the algorithm either* 0 *or* 1*.*

Additionally, ul-PCS schemes must satisfy correctness, detectability, existential unforgeability, attribute-hiding, and unlinkability to ensure both security and privacy.

Correctness.

A ul-PCS scheme is called correct if for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in experiment CORR specified in Figure 2.1, the probability $\Pr[\mathsf{CORR}^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1]$ is negligible in the security parameter.

$\underline{\mathsf{CORR}^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A})}$
$(F, \mathbf{s}^\top) \leftarrow \mathcal{A}_1(1^\lambda)$
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, F);\ \ c \leftarrow 0$
$((i,j),(k,\ell),m) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Gen}}, \mathcal{O}_{\mathsf{ReRand}}}(\mathbf{s}^\top, \mathsf{mpk})$
$(\mathsf{sk}_S, \mathsf{vk}_S, x_S) \leftarrow Q_i[j]$
$(\mathsf{sk}_R, \mathsf{vk}_R, x_R) \leftarrow Q_k[\ell]$
$b \leftarrow \mathsf{Verify}(\mathsf{mpk}, \mathsf{vk}_S, \mathsf{vk}_R, m,$
$\quad \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_S, \mathsf{vk}_R, m))$
Return $b \neq F(x_S, x_R)$

$\underline{\mathcal{O}_{\mathsf{Gen}}(x)}$:
$c \leftarrow c + 1$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$
$Q_c \leftarrow [(\mathsf{sk}, \mathsf{vk}, x)]$
Return $(\mathsf{sk}, \mathsf{vk})$
$\underline{\mathcal{O}_{\mathsf{ReRand}}(j)}$:
If $j > c$ return $\perp$
$(\mathsf{sk}, \mathsf{vk}, x) \leftarrow Q_j[|Q_j|]$
$(\mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk})$
$Q_j \leftarrow Q_j || (\mathsf{sk}', \mathsf{vk}', x)$
Return $(\mathsf{sk}', \mathsf{vk}')$

Figure 2.1: Correctness Experiment of a ul-PCS scheme [BSW24].

2.5.1 Adversarial Capabilities in the Security Games

Before presenting the notions of unforgeability, attribute-hiding and unlinkability, we describe the adversarial capabilities in the different security games.
To keep track of all honestly generated keys, corrupted keys and the list of generated signatures we define the initially empty sets $\mathcal{QK}$, $\mathcal{QC}$ and $\mathcal{QS}$, respectively.

Key-Generation Oracle $\mathsf{QKeyGen}(\cdot)$**:** On the i-th input of an attribute set x_i, generate $(\mathsf{vk}_i^0, \mathsf{sk}_i^0) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x_i)$, add $((i,0), \mathsf{vk}_i^0, \mathsf{sk}_i^0, x_i)$ to $\mathcal{QK}$ and return vk_i^0.

Left-or-Right Key-Generation Oracle $\mathsf{QKeyGenLR}_\beta(\cdot,\cdot)$**:** On the i-th input of a pair of attribute sets $x_{i,0}$ and $x_{i,1}$, generate $(\mathsf{vk}_i^0, \mathsf{sk}_i^0) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x_{i,\beta})$, add $((i,0), \mathsf{vk}_i^0, \mathsf{sk}_i^0, x_{i,0}, x_{i,1})$ to $\mathcal{QK}$, and return vk_i^0.

Key-Randomization Oracle $\mathsf{QRandKey}(\cdot)$**:** After taking an index i, if $\mathcal{QK}$ contains entries $((i,0), \mathsf{vk}_i^0, \mathsf{sk}_i^0, \dots), \dots, ((i,c_i), \mathsf{vk}_i^{c_i}, \mathsf{sk}_i^{c_i}, \dots)$, then compute $(\mathsf{vk}_i^{c_i+1}, \mathsf{sk}_i^{c_i+1}) \leftarrow \mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk}_i^{c_i})$. Add $((i, c_i+1), \mathsf{vk}_i^{c_i+1}, \mathsf{sk}_i^{c_i+1}, \dots)$ to $\mathcal{QK}$ and return $\mathsf{vk}_i^{c_i+1}$.

Corruption Oracle $\mathsf{QCor}(\cdot)$**:** After taking an index i, if $\mathcal{QK}$ contains entries $((i,j), \mathsf{vk}_i^j, \mathsf{sk}_i^j, \dots)$ for $0 \leq j \leq c_i$ for some $c_i \geq 0$, then copy these entries from $\mathcal{QK}$ to $\mathcal{QC}$ and return the list $(\mathsf{sk}_i^0, \dots, \mathsf{sk}_i^{c_i})$.

Signing Oracle $\mathsf{QSign}(\cdot,\cdot,\cdot)$**:** After taking an index pair (i,j) (or a single index i respectively), a public key vk' and a message m, if $\mathcal{QK}$ contains an entry $((i,j), \mathsf{vk}_i^j, \mathsf{sk}_i^j, \dots)$ (or an entry $(i, \mathsf{vk}_i, \mathsf{sk}_i, \dots)$ respectively), then compute $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_i^j, \mathsf{vk}', m)$ (or $\sigma \leftarrow \mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}_i, \mathsf{vk}', m)$ respectively), add $((i,j), \mathsf{vk}_i^j, \mathsf{vk}', m, \sigma)$ to $\mathcal{QS}$ and return the signature.

Randomization-Challenge Oracle $\mathsf{QRandKey}_\beta$ On receiving a query, do the following: if $\beta = 0$ then set $(\mathsf{vk}', \mathsf{sk}') \leftarrow \mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk})$, and if $\beta = 1$ set $(\mathsf{vk}', \mathsf{sk}') \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$ where $(i, \mathsf{sk}, \dots)$ is the last entry of $\mathcal{QK}$. Finally, add $(i+1, \mathsf{vk}', \mathsf{sk}')$ to $\mathcal{QK}$ and return vk'.

For notational ease, if the set $\mathcal{QK}$ contains the sequence $((i,0), \mathsf{vk}_i^0, \mathsf{sk}_i^0, \dots), \dots, ((i,c_i), \mathsf{vk}_i^{c_i}, \mathsf{sk}_i^{c_i}, \dots)$ we denote by $\mathcal{QK}_i$ the corresponding sequence of keys $[(\mathsf{vk}_i^0, \mathsf{sk}_i^0), \dots, (\mathsf{vk}_i^{c_i}, \mathsf{sk}_i^{c_i})]$ for party i.

2.5.2 Detectibility.

Let Detect be an algorithm that takes as input the master public key mpk, a candidate key vk^*, and a list consisting of sequences of key pairs $(Q_1, \ldots, Q_c)$, and outputs an index or $\perp$. A ULPCS scheme is said to have the detectability property if there is an efficiently computable algorithm Detect such that for all efficient adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in experiment DTCT specified in Figure 2.2, the probability $\Pr[\mathrm{DTCT}^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1]$ is negligible in the security parameter.

$\underline{\mathrm{det}^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A})}$
$(F, \mathbf{s}^\top) \leftarrow \mathcal{A}_1(1^\lambda)$
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, F)$
$c := 0$
$(i, j) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{Gen}}, \mathcal{O}_{\mathsf{ReRand}}}(\mathbf{s}^\top, \mathsf{mpk})$
$(\mathsf{sk}^*, \mathsf{vk}^*) \leftarrow Q_i[j]$
$i^* \leftarrow \mathsf{Detect}(\mathsf{mpk}, \mathsf{vk}^*, (Q_1, \ldots, Q_c))$
Return $i^* \neq i$

$\underline{\mathcal{O}_{\mathsf{Gen}}(x)\text{:}}$
$c \leftarrow c + 1$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$
$Q_c \leftarrow [(\mathsf{sk}, \mathsf{vk})]$
Return $(\mathsf{sk}, \mathsf{vk})$
$\underline{\mathcal{O}_{\mathsf{ReRand}}(j)\text{:}}$
If $j > c$, return $\perp$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow Q_j[|Q_j|]$
$(\mathsf{sk}', \mathsf{vk}') \leftarrow \mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk})$
$Q_j \leftarrow Q_j || (\mathsf{sk}', \mathsf{vk}')$
Return $(\mathsf{sk}', \mathsf{vk}')$

Figure 2.2: Detectability Experiment of a ul-PCS scheme [BSW24].

Unforgeability.

Definition 13 (Existential Unforgeability of a PCS Scheme)**.** *Let* $\mathsf{ULPCS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a ul-PCS scheme that satisfies the detectability property. We define the experiment* $\mathsf{EUF\text{-}CMA}^{\mathsf{ULPCS}}$ *in Figure 2.3, where all oracles are defined as in Section 2.5.1. The advantage of an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is defined by*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{ULPCS}, \mathcal{A}}(\lambda) = \Pr[\mathsf{EUF\text{-}CMA}^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1].$$

Such a ul-PCS scheme ULPCS *is called* existential unforgeable under adaptive chosen message attacks *or* existential unforgeable *for short if for any polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathsf{ULPCS}, \mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*. We further call a ul-PCS scheme* T_{Rand}*-unforgeable if the number of key rerandomization queries* q *is less than* T_{Rand}*, i.e.* $q < T_{\mathsf{Rand}}$*.*

$\underline{\mathsf{EUF\text{-}CMA}^{\mathsf{ULPCS}}(1^n, \mathcal{A})}$
$(F, \mathbf{s}^\top) \leftarrow \mathcal{A}_1(1^n)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, F)$
$(\mathsf{vk}, \mathsf{vk}^*, m^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathsf{QKeyGen}(\cdot), \mathsf{QRandKey}(\cdot)\mathsf{QCor}(\cdot), \mathsf{QSign}(\cdot,\cdot,\cdot)}(\mathbf{s}^\top, \mathsf{mpk})$
Let $i_{\max}$ be the number of queries made to $\mathsf{QKeyGen}(\cdot)$
$S \leftarrow \mathsf{Detect}(\mathsf{mpk}, \mathsf{vk}, (\mathcal{QK}_1, \ldots, \mathcal{QK}_{i_{\max}}))$
$R \leftarrow \mathsf{Detect}(\mathsf{mpk}, \mathsf{vk}^*, (\mathcal{QK}_1, \ldots, \mathcal{QK}_{i_{\max}}))$
Let x_S and x_R denote the attributes in case $S, R \neq \perp$
Output: $\mathsf{Verify}(\mathsf{mpk}, \mathsf{vk}, \mathsf{vk}^*, m^*, \sigma^*) = 1 \wedge$
$\Big[\big[\exists (i, j), \mathsf{sk}, x\ \forall (i', j'), \sigma : ((i, j), \mathsf{vk}, \mathsf{sk}, x) \in \mathcal{QK} \setminus \mathcal{QC} \wedge$
$((i', j'), \mathsf{vk}, \mathsf{vk}^*, m^*, \sigma) \notin \mathcal{QS}\big] \vee \big[(S \neq \perp) \wedge (R \neq \perp) \Rightarrow F(x_S, x_R) = 0\big]\Big]$

Figure 2.3: Unforgeability Game of ULPCS. [BSW24]

Attribute-Hiding.

$\mathsf{AH}_{\beta}^{\mathsf{ULPCS}}(1^n, \mathcal{A})$
$(F, \mathbf{s}^\top) \leftarrow \mathcal{A}_1(1^n)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ULPCS.Setup}(1^\lambda, F)$
$\alpha \leftarrow \mathcal{A}_2^{\mathsf{QKeyGenLR}_\beta(\cdot,\cdot),\mathsf{QRandKey}(\cdot),\mathsf{QCor}(\cdot),\mathsf{QSign}(\cdot,\cdot,\cdot)}(\mathbf{s}^\top, \mathsf{mpk})$
Output: α

Figure 2.4: The Attribute-Hiding game for ULPCS. [BSW24]

Definition 14 (IND-Based Attribute Hiding)**.** *Let* $\mathsf{ULPCS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a ul-PCS scheme that satisfies the detectability property. For* $\beta \in \{0,1\}$*, we define the experiment* $\mathsf{AH}_{\beta}^{\mathsf{ULPCS}}$ *in Figure 2.4, where all oracles are defined as in Section 2.5.1. The advantage of an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is defined by*

$$\mathsf{Adv}^{\mathsf{AH}}_{\mathsf{ULPCS},\mathcal{A}}(\lambda) = |\Pr[\mathsf{AH}_0^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1] - \Pr[\mathsf{AH}_1^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1]|.$$

We call an adversary valid *if all of the following hold with probability* 1 *over the randomness of the adversary and all involved algorithms, where* $i_{\max}$ *denotes an upper bound on the number of queries to* $\mathsf{QKeyGenLR}_\beta$*:*

- *for every* $((i,j), \mathsf{vk}_i^j, \mathsf{sk}_i^j, x_{i,0}, x_{i,1}) \in \mathcal{QC}$ *and for all* $((k,\ell), \mathsf{vk}_k^\ell, \mathsf{sk}_k^\ell, x_{k,0}, x_{k,1}) \in \mathcal{QK}$ *we have* $x_{i,0} = x_{i,1} =: x_i$ *and* $F(x_i, x_{k,0}) = F(x_i, x_{k,1})$*,*
- *and for all* $((i,j), \mathsf{vk}_i^j, \mathsf{vk}, m, \sigma) \in \mathcal{QS}$*,* $R \leftarrow \mathsf{Detect}(\mathsf{mpk}, \mathsf{vk}, (\mathcal{QK}_1, \ldots, \mathcal{QK}_{i_{\max}}))$*, and* $((i,j), \mathsf{vk}_i, \mathsf{sk}_i, x_{i,0}, x_{i,1}) \in \mathcal{QK}$*, we either have* $R = \perp$ *or otherwise* $F(x_{i,0}, x_{R,0}) = F(x_{i,1}, x_{R,1})$ *holds.*

Such a ul-PCS scheme ULPCS *is called* attribute hiding *if for any valid polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\mathsf{AH}}_{\mathsf{ULPCS},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*. We call a ul-PCS scheme* T_{Rand}*-attribute-hiding if the number of key rerandomization queries* q *is less than* T_{Rand}*, i.e.* $q < T_{\mathsf{Rand}}$*. Finally, we call a ul-PCS scheme outsider-attribute-hiding (outsider-AH) if the adversary does not have access to the corruption oracle.*

Unlinkability.

Definition 15. *Let* $\mathsf{ULPCS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a ul-PCS scheme that satisfies the detectability property. For* $\beta \in \{0,1\}$*, we define the experiment* $\mathsf{Link}_{\beta}^{\mathsf{ULPCS}}$ *in Figure 2.5, where all oracles are defined as in Section 2.5.1. The advantage of an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *is defined by*

$$\mathsf{Adv}^{\mathsf{Link}}_{\mathsf{ULPCS},\mathcal{A}}(n) = |\Pr[\mathsf{Link}_0^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1] - \Pr[\mathsf{Link}_1^{\mathsf{ULPCS}}(1^\lambda, \mathcal{A}) = 1]|.$$

We call such a ul-PCS scheme ULPCS unlinkable *if for any polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\mathsf{Link}}_{\mathsf{ULPCS},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*.*
We call a ul-PCS scheme T_{Rand}*-unlinkable if the number of key rerandomization queries* q *is less than* T_{Rand}*, i.e.* $q < T_{\mathsf{Rand}}$*.*

$\mathsf{Link}_{\beta}^{\mathsf{ULPCS}}(1^n, \mathcal{A})$

$(F, \mathsf{st}_1) \leftarrow \mathcal{A}_1(n)$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^n, F)$

$(x, \mathsf{st}_2) \leftarrow \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}, \mathsf{st}_1)$

$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{msk}, x)$

$\alpha \leftarrow \mathcal{A}_3^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{QRandKey}_{\beta},\mathsf{QSign}(\cdot,\cdot,\cdot)}(\mathsf{vk}, \mathsf{st}_2)$

Output: α

Figure 2.5: Single-Challenge Unlinkability game of ULPCS. [BSW24]

2.6 THRESHOLD SIGNATURE

While traditional Digital Signatures (DS) involve a single signer, an (n, t)-threshold signature scheme enables collaborative signing among n participants. In such a scheme, any group of at least t members can jointly produce a valid signature, whereas groups with fewer than t participants cannot.

Threshold signatures, introduced by Desmedt in 1989, were designed to distribute trust across multiple signers and enhance key availability [Des90]. Since then, substantial progress has been made in improving the security and efficiency of these schemes [GJKR01, MR01, GGN16, Lin17, DKLS18, DKLS19]. With the emergence of cryptocurrencies, blockchain technology, and self-sovereign identity systems, threshold signatures have garnered increased research attention [Bol02, KG21, CGG+20, KMOS21, CKM23]. Moreover, a standardization initiative in the broader field of threshold cryptography is currently being undertaken by NIST [BDV20].

Definition 16 ((n, t)-Threshold Signatures)**.** *Given a security parameter κ and a message space $\mathcal{M}$, an (n, t)-TS contains the following PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{TS.Setup}(1^\kappa)$*: The setup algorithm takes the security parameter κ as input and returns the set of public parameters* pp *as output.*
- $(\{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[1,n]}, \mathsf{vk}) \leftarrow \mathsf{TS.KeyGen}(\mathsf{pp}, n, t)$*: The key generation algorithm takes the public parameters* pp *along with two integers n, t s.t. $1 \leq t \leq n$ as inputs. It then returns secret/verification keys $\{\mathsf{sk}_i, \mathsf{vk}_i\}$ for $i \in [1, n]$ along with a global verification key* vk *as output.*
- $\sigma_i \leftarrow \mathsf{TS.ParSign}(\mathsf{pp}, \mathsf{sk}_i, M)$*: The partial signing algorithm takes* pp*, the i^{th} party's secret key,* sk_i*, and a message $M \in \mathcal{M}$ as inputs. It then returns a partial signature σ_i as output.*
- $0/1 \leftarrow \mathsf{TS.ParVerify}(\mathsf{pp}, \mathsf{vk}_i, M, \sigma_i)$*: The partial verification algorithm as a deterministic algorithm, takes* pp*, the i^{th} verification key,* vk_i*, and a message $M \in \mathcal{M}$ along with partial signature σ_i as inputs. It then returns 1 (accept), if the partial signature is valid; 0 (reject), otherwise.*
- $\sigma \leftarrow \mathsf{TS.CombineSign}(\mathsf{pp}, T, \{\sigma_i\}_{i\in T})$*: The combine algorithm takes a set of partial signatures that already satisfy the* TS.ParVerify *algorithm σ_i for $i \in T$ along with $T \subseteq [1, n]$ and then returns an aggregated signature σ as output.*
- $0/1 \leftarrow \mathsf{TS.Verify}(\mathsf{pp}, \mathsf{vk}, M, \sigma)$*: The verification algorithm as a deterministic algorithm, takes* pp*, the global verification key,* vk*, and message $M \in \mathcal{M}$ along with an aggregated signature σ as inputs. It then returns 1 (accept), if the aggregated signature is valid and 0 (reject), otherwise.*

Definition 17 (Correctness)**.** *An (n, t)-TS scheme is called correct if we have:*

$$\Pr\left[\begin{array}{l}\forall\ \mathsf{pp} \leftarrow \mathsf{TS.Setup}(1^\kappa), (\{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[1,n]}, \mathsf{vk}) \leftarrow \mathsf{TS.KeyGen}(\mathsf{pp}, n, t), \mathbf{M} \in \mathcal{M}, \\ \sigma_i \leftarrow \mathsf{TS.ParSign}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{M}) \text{ for } i \in [1, n], \forall\ T \subseteq [1, n], |T| \geq t, \\ \sigma \leftarrow \mathsf{TS.CombineSign}\left(\mathsf{pp}, T, \{\sigma_i\}_{i\in T}\right) : \mathsf{TS.Verify}\left(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma\right) = 1\end{array}\right] = 1\ .$$

Unforgeability. As defined by Cremers *et al.* [CPZ24] we have defined threshold Unforgeability as TS-UF-0, TS-UF-1, TS-UF-2, TS-UF-3 and TS-UF-4. As our signature scheme is fully non-interactive the TS-UF-0 and TS-UF-1 unforgeability are needed to show. In particular, the TS-UF-0 notion is the weaker one and prohibits adversaries from querying the signing oracle for partial signatures on the challenge message, i.e., the message corresponding to the forged signature. The stronger TS-UF-1 notion, which will be our main focus, allows adversaries to query the signing oracle up to $t - |\mathsf{CS}|$ times for partial signatures, even on the challenge message. Here CS with $|\mathsf{CS}| < t$ denotes the set of (statically corrupted) signers.

2.7 STRUCTURE-PRESERVING SIGNATURES ON EQUIVALENCE CLASSES

First, we will discuss structure-preserving signatures (SPS), and then go deeper into the notion of SPS on equivalence classes(SPS-EQ). For the rest of this thesis, we use the same equivalence relation used to partition the message space $(\mathbb{G}^*)^\ell$ as described in [FHS14], defined as follows:

$$\mathcal{R} := \left\{ (\mathbf{M}, \mathbf{M}') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell \;\middle|\; \exists \mu \in \mathbb{Z}_p^* \colon \mu\mathbf{M} = \mathbf{M}' \right\}.$$

Therefore, $[\mathbf{M}]_\mathcal{R}$ represents the set of all $\mathbf{M}' = \mu\mathbf{M}$, where $\mu \in \mathbb{Z}_p^*$.

Structure-preserving signatures (SPS) [AFG+10] are pairing-based signatures where all the messages, signatures and public keys are group elements, and verifying signatures only involves deciding group membership of the signature components and evaluating Pairing Product Equations (PPE).

Definition 18 (Structure-preserving signature)**.** *A structure-preserving signature scheme SPS is defined as a triple of probabilistic polynomial time (PPT) algorithms:*

- $(\mathsf{vk}, \mathsf{sk}) \leftarrow Gen(1^\kappa)$*: The probabilistic key generation algorithm returns the public/secret key* $(\mathsf{vk}, \mathsf{sk})$*, where* $\mathsf{vk} \in \mathbb{G}^{n_{vk}}$ *for some* $n_{vk} \in poly(\kappa)$*. We assume that* vk *implicitly defines a message space* $M := \mathbb{G}^n$ *for some* $n \in poly(\kappa)$*.*
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, M)$*: The probabilistic signing algorithm returns a signature* $\sigma \in \mathbb{G}^{n_\sigma}$ *for* $n_\sigma \in poly(\kappa)$
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{vk}, M, \sigma)$*: The deterministic verification algorithm only consists of pairing product equations and returns 1 (accept) or 0 (reject).*

Now, we recall the definition and the security model of SPS-EQ scheme.

Definition 19 (Structure-Preserving Signatures on Equivalence classes [FHS14])**.** *Given an asymmetric bilinear group, a SPS-EQ over message space* $\mathcal{M} := (\mathbb{G}_i^*)^\ell$ *consists of the following PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}_\mathcal{R}(1^\lambda)$*: A probabilistic algorithm that takes the security parameter* λ *in its unary representation as input, and outputs public parameters* pp*.*
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_\mathcal{R}(\mathsf{pp}, \ell)$*: A probabilistic algorithm that takes the public parameters* pp *and a vector length* $\ell > 1$ *as inputs, and outputs the key-pair* $(\mathsf{sk}, \mathsf{vk})$*.*
- $\sigma \leftarrow \mathsf{Sign}_\mathcal{R}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})$*: A probabilistic algorithm that takes public parameters* pp*, secret key* sk *and a representative message* $\mathbf{M} \in \mathcal{M}$ *for class* $[\mathbf{M}]_\mathcal{R}$ *as inputs. It outputs the signature* σ *on message* $\mathbf{M}$*.*
- $0/1 \leftarrow \mathsf{Verify}_\mathcal{R}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma)$*: A deterministic algorithm that takes public parameters* pp*, a verification key* vk*, representative message* $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$*, a signature* σ *as inputs, and outputs* 1 *if the signature* σ *is valid for* $\mathbf{M}$*, and* 0 *otherwise.*
- $(\sigma', \mathbf{M}') \leftarrow \mathsf{ChgRep}_\mathcal{R}(\mathsf{pp}, \mathbf{M}, \sigma, \mu, \mathsf{vk})$*: The change representation algorithm is a probabilistic algorithm and takes public parameters* pp*, a representative message* $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$*, a signature* σ*, a scalar* $\mu \in \mathbb{Z}_p^*$ *and the verification key* vk *as inputs. It outputs a randomized signature* σ' *on a new representative message* $\mathbf{M}' = \mu\mathbf{M}$*.*

Security Properties The primary security requirements for a SPS-EQ scheme are *correctness* and *existential unforgeability against chosen message attack*, which are defined as follows:

Definition 20 (Correctness)**.** *A SPS-EQ scheme over $\mathcal{M}$ is called* correct*, if for a valid setup* pp*, any message $\mathbf{M} \in \mathcal{M}$, any (valid) key pair* $(\mathsf{sk}, \mathsf{vk})$ *in the support of* $\mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell)$*, and any scalar $\mu \in \mathbb{Z}_p^*$, we have:*

$$\Pr\left[\begin{array}{l}\mathsf{Verify}_{\mathcal{R}}\left(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})\right) = 1 \wedge \\ \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mu\mathbf{M}, \\ \qquad \mathsf{ChgRep}_{\mathcal{R}}(\mathbf{M}, \mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M}), \mu, \mathsf{vk})) = 1\end{array}\right] = 1 \ .$$

Definition 21 (Existential Unforgeability)**.** *A SPS-EQ over $\mathcal{M}$ is called adaptively* EUF-CMA*-secure if for all PPT adversaries $\mathcal{A}$ with access to the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ we have:*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^\lambda), (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, \ell), \\ (\mathbf{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}(\cdot)}(\mathsf{pp}, \mathsf{vk}) : \\ \mathbf{M}^* \notin \mathcal{Q}^{\mathsf{Sign}} \ \wedge \ \mathsf{Verify}_{\mathcal{R}}\left(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma^*\right) = 1\end{array}\right] \leq \mathsf{negl}(\lambda) \ ,$$

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}(\cdot)$ takes a message $\mathbf{M} \in \mathcal{M}$ as inputthe algorithms $\mathsf{Sign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}, \mathbf{M})$ and updates the query set $\mathcal{Q}^{\mathsf{Sign}} = \mathcal{Q}^{\mathsf{Sign}} \cup \{[\mathbf{M}]_{\mathcal{R}}\}$.

Additionally, an SPS-EQ achieves Class-Hiding (cf. Definition 22) and Perfect Adaptation (cf. Definition 23).

Definition 22 (Class-Hiding [FHS14])**.** *A relation $\mathcal{R}$ is called class-hiding if for all PPT adversaries, $\mathcal{A}$, and $\ell > 1$ we have:*

$$\left|\Pr\left[\begin{array}{l}\mathbf{M} \leftarrow (\mathbb{G}^*)^\ell, \mathbf{M}_0 \leftarrow (\mathbb{G}^*)^\ell, \mathbf{M}_1 \leftarrow [\mathbf{M}]_{\mathcal{R}}, \\ b \leftarrow \{0,1\}, b' \leftarrow \mathcal{A}(\mathbf{M}, \mathbf{M}_b) \mid b = b'\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Definition 23 (Signature Adaptation)**.** *An SPS-EQ scheme over message space $\mathcal{M}$ perfectly adapts signatures if for all tuples* $(\mathsf{sk}, \mathsf{vk}, \mathbf{M}, \sigma, \mu)$*, where* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \ell)$*, $\mathbf{M} \in (\mathbb{G}_i^*)^\ell$ and* $\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma) = 1$*, the two distributions $\sigma(\mathsf{pp}, \mathsf{sk}, \mathbf{M}^\mu)$ and* $\mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathbf{M}, \sigma, \mathsf{vk}, \mu)$ *are identical.*

2.8 THRESHOLD STRUCTURE-PRESERVING SIGNATURES ON EQUIVALENCE CLASS

First, we recall the definition of the threshold structure-preserving signatures on equivalence classes (TSPS-EQ) from [ANPKT24] and their main security properties: correctness, threshold unforgeability and perfect adaption. In this thesis, we assume the existence of a trusted dealer who shares the secret key among the signers. However, there are straightforward and well-known techniques in particular distributed key generation (DKG) protocols (e.g., [Ped92]) without such an assumption.

Definition 24 (Threshold structure-preserving signatures on equivalence classes)**.** *Given an asymmetric bilinear group and the relation described in Equation (2.1), a TSPS-EQ over message space $\mathcal{M} := (\mathbb{G}_i^*)^\ell$ consists of the following PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^\kappa)$*: The setup algorithm takes the security parameter κ as input and returns the set of public parameters* pp *as output.*
- $(\{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[1,n]}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, n, t)$*: The key generation algorithm takes the public parameters* pp *along with two integers n, t s.t. $1 \leq t \leq n$ as inputs. It then returns secret/verification keys $\{\mathsf{sk}_i, \mathsf{vk}_i\}$ for $i \in [1, n]$ along with a global verification key* vk *as output.*
- $\sigma_i \leftarrow \mathsf{ParSign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{M})$*: The partial signing algorithm takes* pp*, the i^{th} party's secret key, sk_i, and a message $\mathbf{M} \in \mathcal{M}$ as inputs. It then returns a partial signature σ_i as output.*

- $0/1 \leftarrow \mathsf{ParVerify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}_i, \mathbf{M}, \sigma_i)$: *The partial verification algorithm as a deterministic algorithm, takes* pp*, the* i^{th} *verification key,* vk_i*, and a message* $\mathbf{M} \in \mathcal{M}$ *along with partial signature* σ_i *as inputs. It then returns* 1 *(accept), if the partial signature is valid;* 0 *(reject), otherwise.*
- $\sigma \leftarrow \mathsf{CombineSign}_{\mathcal{R}}(\mathsf{pp}, T, \{\sigma_i\}_{i\in T})$: *The combine algorithm takes a set of partial signatures that already satisfy the* ParVerify *algorithm* σ_i *for* $i \in T$ *along with* $T \subseteq [1, n]$ *and then returns an aggregated signature* σ *as output.*
- $0/1 \leftarrow \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma)$: *The verification algorithm as a deterministic algorithm, takes* pp*, the global verification key,* vk*, and message* $\mathbf{M} \in \mathcal{M}$ *along with an aggregated signature* σ *as inputs. It then returns* 1 *(accept), if the aggregated signature is valid and* 0 *(reject), otherwise.*
- $(\mathbf{M}', \sigma') \leftarrow \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \mu, \sigma)$: *The adaptation algorithm as a probabilistic algorithm, takes* pp*, the global verification key,* vk*, message* $\mathbf{M} \in \mathcal{M}$ *and a scalar* $\mu \leftarrow \mathbb{Z}_p^*$ *along with an aggregated signature* σ *as inputs. This PPT algorithm then returns* $\mathbf{M}' \in [\mathbf{M}]_{\mathcal{R}}$*, and a randomized signature* σ'*.*

Correctness. Correctness guarantees that a signature obtained from a set $T \subseteq [1, n]$ s.t. $|T| \geq t$ of honest signers always verifies.

Definition 25 (Correctness)**.** *An* (n, t)*-TSPS-EQ scheme is called correct if we have:*

$$\Pr\left[\begin{array}{l} \forall\ \mathsf{pp} \leftarrow \mathsf{Setup}_{\mathcal{R}}(1^{\kappa}), (\{\mathsf{sk}_i, \mathsf{vk}_i\}_{i\in[1,n]}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, n, t), \mathbf{M} \in \mathcal{M}, \\ \Sigma_i \leftarrow \mathsf{ParSign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}_i, \mathbf{M}) \text{ for } i \in [1, n], \forall\ T \subseteq [1, n], |T| \geq t, \\ \Sigma \leftarrow \mathsf{CombineSign}_{\mathcal{R}}\left(\mathsf{pp}, T, \{\Sigma_i\}_{i\in T}\right) : \mathsf{Verify}_{\mathcal{R}}\left(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}], \Sigma\right) = 1 \\ \quad \wedge\ \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mu \cdot \mathbf{M}, \mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathbf{M}, \Sigma, \mu, \mathsf{vk}), \mathsf{vk}) = 1 \end{array}\right] = 1\ .$$

Unforgettability. As discussed in the Section 2.6, the following defines the unforgeability for TSPS-EQ

Definition 26 (Threshold Unforgeability)**.** *Let* $\mathsf{TSPS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ParSign}, \mathsf{ParVerify}, \mathsf{CombineSign}, \mathsf{Verify})$ *be an* (n, t)*-TSPS scheme over message space* $\mathcal{M}$ *and let* $\mathsf{prop} \in \{\mathsf{TS\text{-}UF\text{-}b}, \mathsf{adp\text{-}TS\text{-}UF\text{-}b}\}_{\mathsf{b}\in\{0,1\}}$*. The advantage of a PPT adversary* $\mathcal{A}$ *playing described security games in Figure 2.6, is defined as,*

$$\mathbf{Adv}^{\mathsf{prop}}_{\mathsf{TSPS},\mathcal{A}}(\kappa) = \Pr\left[\mathbf{G}^{\mathsf{prop}}_{\mathsf{TS},\mathcal{A}}(\kappa) = 1\right]\ .$$

A TSPS achieves prop-security if we have, $\mathbf{Adv}^{\mathsf{prop}}_{\mathsf{TSPS},\mathcal{A}}(\kappa) \leq \mathsf{negl}(\kappa)$.

Definition 27 (Signature adaptation under malicious keys)**.** *Let* $\ell > 1$*. An SPS-EQ scheme on* $(\mathbb{G}_i^*)^{\ell}$ *perfectly adapts signatures if for all tuples* $(sk, pk, \mathbf{M}, \sigma, \mu)$ *with*

$$\mathbf{M} \in (\mathbb{G}_i^*)^{\ell} \qquad \mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mathbf{M}, \sigma) = 1 \qquad \mu \in \mathbb{Z}_p^*$$

We have that the output of $\mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathbf{M}, \sigma, \mu, \mathsf{vk})$ *is a uniformly random element is the space of signature space, conditioned on* $\mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, \mu\mathbf{M}, \sigma') = 1$.

Definition 28 (Class-hiding)**.** *Let* $\ell > 1$ *and* $\mathbb{G}_i^*$ *be a base group of a bilinear group. The message space* $(\mathbb{G}_i^*)^{\ell}$ *is class-hiding if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(.)$ *such that*

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}, BG \leftarrow BGGen_{\mathcal{R}}, \mathbf{M} \leftarrow (\mathbb{G}_i^*)^{\ell}, \\ \mathbf{M}^{(0)} \leftarrow (\mathbb{G}_i^*)^{\ell}, \mathbf{M}^{(1)} \leftarrow \mathbf{M}_{\mathcal{R}}, b^* \leftarrow \mathcal{A}(BG, \mathbf{M}, \mathbf{M}^{(b)}) \end{array} : b^* = b\right] - \frac{1}{2} \leq \epsilon(\kappa)$$

2.9 SECRET SHARING SCHEME

Secret sharing schemes enable a secret to be distributed among multiple parties such that only a predefined number of parties can collaboratively reconstruct it. In this thesis, we utilize Shamir's secret sharing scheme [Sha79], a well-established method based on Lagrange interpolation [29].

$\boxed{G^{\text{TS-UF-0}}_{\text{TS},\mathcal{A}}(\kappa)}$, $\boxed{G^{\text{TS-UF-1}}_{\text{TS},\mathcal{A}}(\kappa)}$, $\boxed{G^{\text{adp-TS-UF-0}}_{\text{TS},\mathcal{A}}(\kappa)}$, $\boxed{G^{\text{adp-TS-UF-1}}_{\text{TS},\mathcal{A}}(\kappa)}$:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$

$(n, t, \mathsf{CS}, \mathsf{st}_0) \leftarrow \mathcal{A}(\mathsf{pp})$

$\mathsf{HS} \coloneqq [1, n] \setminus \mathsf{CS}$

$(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[1,n]}, \{\mathsf{vk}_i\}_{i\in[1,n]}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, n, t)$

$([\mathbf{m}^*], \Sigma^*, \mathsf{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PSign}}(.),\ \mathcal{O}^{\mathsf{Corrupt}}(.)}(\mathsf{st}_0, \mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathsf{CS}}, \{\mathsf{vk}_i\}_{i\in[1,n]})$

$\mathbf{return}\ \Big(\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*], \Sigma^*)\ \wedge\ |\mathsf{CS}| < t\ \wedge$
$\big(\boxed{S_1([\mathbf{m}^*]) = \emptyset}\ \vee\ \boxed{|S_1([\mathbf{m}^*])| < t - |\mathsf{CS}|}\big)\Big)$

$\underline{\mathcal{O}^{\mathsf{PSign}}(i, [\mathbf{m}]):}$
$\mathsf{Assert}\ \big([\mathbf{m}] \in \mathcal{M}\ \wedge\ i \in \mathsf{HS}\big)$
$\Sigma_i \leftarrow \mathsf{ParSign}(\mathsf{pp}, \mathsf{sk}_i, [\mathbf{m}])$
$\Sigma_i \neq \perp :$
$\quad S_1([\mathbf{m}]) \leftarrow S_1([\mathbf{m}]) \cup \{i\}$
$\mathbf{return}\ (\Sigma_i)$

$\underline{\mathcal{O}^{\mathsf{Corrupt}}(k):}$
$k \in \mathsf{CS} :$
$\quad \mathbf{return}\ \perp$
$: \mathsf{CS} \leftarrow \mathsf{CS} \cup \{k\}$
$\quad \mathsf{HS} \leftarrow \mathsf{HS} \setminus \{k\}$
$\quad \mathbf{return}\ (\mathsf{sk}_k)$

Figure 2.6: Games defining the $\boxed{\text{TS-UF-0}}$, $\boxed{\text{TS-UF-1}}$, $\boxed{\text{adp-TS-UF-0}}$, and $\boxed{\text{adp-TS-UF-1}}$ unforgeability notions of threshold signatures. [MMS+24]

In a (n, t)-Shamir Secret Sharing scheme, a secret s is encoded as the constant term of a randomly chosen polynomial of degree $t-1$, and n shares are generated by evaluating this polynomial at n distinct nonzero points. To recover the secret, one uses Lagrange interpolation .

Definition 29 (Lagrange's Interpolation)**.** *If S is a subset of $\mathbb{F}$ with $|S| > t-1$ and $h(x)$ is polynomial of degree at most $t-1$, then by Lagrange's interpolation we can find a unique polynomial $h(x)$ such that,*

$$h(X) = \sum_{i\in S} h(i)\lambda_i(X)$$

where $\lambda_i(X)$ is the degree t polynomial such that, for all $i, j \in S$, $\lambda_i(i)$ = 0 if $i \neq j$ and $\lambda_i(j) = 1$, if $i = j$. In other words,

$$\lambda_i(X) = \prod_{j\in C, j\neq i} \frac{(X-j)}{(i-j)}$$

In this work, we use Shamir Secret Sharing to secret share a matrix of size $a \times b$, i.e., we use ab-many parallel instances of Shamir Secret Sharing. To keep our exposition simpler, we however assume that we have an (n, t)-Shamir Secret Sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ which operates on matrices. Since, our work here uses Shamir Secret Sharing quite generically, it is convenient to make such abstraction without going into the details.

Definition 30 (Secret Sharing)**.** *For any two positive integers $n, t < n$, an $(n, t)_{\mathbb{Z}_p^{a\times b}}$-secret-sharing scheme over $\mathbb{Z}_p^{a\times b}$ for $a, b \in \mathbb{N}$ consists of two functions* Share *and* Rec*.* Share *is a randomized function that takes a secret $\mathbf{M} \in \mathbb{Z}_p^{a\times b}$ and outputs $(\mathbf{M}_1, \ldots, \mathbf{M}_n) \leftarrow \mathsf{Share}(\mathbf{M}, \mathbb{Z}_p^{a\times b}, n, t)$ where $\mathbf{M}_i \in \mathbb{Z}_p^{a\times b}\ \forall i \in [1, n]$. The pair of functions $(\mathsf{Share}, \mathsf{Rec})$ satisfy the following requirements.*

- ***Correctness**: For any secret $\mathbf{M} \in \mathbb{Z}_p^{a\times b}$ and a set of parties $\{i_1, i_2, \ldots, i_k\} \subseteq [1, n]$ such that $k \geq t$, we have*

$$\Pr[\mathsf{Rec}(\mathbf{M}_{i_1}, \ldots, \mathbf{M}_{i_k} : (\mathbf{M}_1, \ldots, \mathbf{M}_n) \leftarrow \mathsf{Share}(\mathbf{M}, \mathbb{Z}_p^{a\times b}, n, t)) = \mathbf{M}] = 1 \ .$$

- ***Security**: For any secret $\mathbf{M} \in \mathbb{Z}_p^{a\times b}$ and a set of parties $S \subseteq [1, n]$ such that $|S| = k < t$, for all information-theoretic adversary $\mathcal{A}$ we have*

$$\Pr\left[S = \{i_i\}_{i\in[1,k]} \wedge \mathbf{M}^* = \mathbf{M} \;\middle|\; \begin{array}{l}(\mathbf{M}_1, \ldots, \mathbf{M}_n) \leftarrow \mathsf{Share}(\mathbf{M}, \mathbb{Z}_p^{a\times b}, n, t)\\ S \leftarrow \mathcal{A}()\\ \mathbf{M}^* \leftarrow \mathcal{A}(\mathbf{M}_{i_1}, \ldots, \mathbf{M}_{i_k})\end{array}\right] = 1/p \ .$$

  We follow standard nomenclature to call this "selective security". In case of "adaptive security", $\mathcal{A}$ adaptively chooses $i_j \in [1, n]$ to get $\mathbf{M}_{i_j}$ one at a time.

2.10 PSEUDORANDOM FUNCTION

We recall the definition of a pseudorandom function (PRF) as it has been defined in [GGM86].

Definition 31 (Pseudorandom Function)**.** *A pseudo-random function is a keyed function* $\mathsf{PRF} : \{0,1\}^n \times \mathcal{X} \to \mathcal{Y}$*, where evaluation is done via an efficient algorithm* $\mathsf{PRF.Eval}(k, x)$*. For* $\beta \in \{0, 1\}$*, we define the experiment* $\mathsf{IND}_\beta^{\mathsf{PRF}}$ *in Figure 2.7, where the oracle* $\mathcal{O}$ *is defined as:*

$$\mathcal{O}(x) = \begin{cases} \mathsf{PRF.Eval}(k, x) & \text{if } \beta = 0 \\ \mathsf{RF}(x) & \text{if } \beta = 1 \end{cases} \ .$$

with $\mathsf{RF}(x)$ *denoting a random function. We define the advantage of an adversary* $\mathcal{A}$ *in the following way:*

$$\mathsf{Adv}^{\mathsf{IND}}_{\mathsf{PRF},\mathcal{A}}(\lambda) = |\Pr[\mathsf{IND}_0^{\mathsf{PRF}}(\lambda, \mathcal{A})] - \Pr[\mathsf{IND}_1^{\mathsf{PRF}}(\lambda, \mathcal{A})]| \ .$$

A pseudorandom function PRF *is secure, if for any polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that:* $\mathsf{Adv}^{\mathsf{IND}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$*.*

$\mathsf{IND}_\beta^{\mathsf{PRF}}(\lambda, \mathcal{A})$
$k \leftarrow \{0,1\}^n$
$\alpha \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(1^\lambda)$
Output: α

Figure 2.7: Security Games for PRF. [BSW24]

2.11 PREDICATE ENCRYPTION

To allow for oblivious policy evaluations, we also recap the notion of *predicate-only predicate encryption* as it has been introduced by Katz et al. [KSW08].

Definition 32 (Predicate-Only Predicate Encryption)**.** *Let* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *be a family of sets* $\mathcal{F}_\lambda$ *of predicates* $f\colon \mathcal{X}_\lambda \to \{0, 1\}$*. A* predicate-only predicate encryption *(PE) scheme for the functionality class* $\mathcal{F}_\lambda$ *is a tuple of four algorithms* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda)$*: Takes as input a unary representation of the security parameter* λ *and outputs the master public key* mpk *and the master secret key* msk*.*

- $\mathsf{KeyGen}(\mathsf{msk}, f)$: *Takes as input the master secret key* msk *and a function* $f \in \mathcal{F}$*, and outputs a functional key* sk_f.
- $\mathsf{Enc}(\mathsf{mpk}, x)$: *Takes as input the master public key* mpk *and an attribute* $x \in \mathcal{X}_\lambda$*, and outputs a ciphertext* ct.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct})$: *Takes as input a functional key* sk_f *and a ciphertext* ct *and outputs* 0 *or* 1.

A predicate-only predicate encryption scheme ABE *is* correct *if for all* $\lambda \in \mathbb{N}$*, for all* $(\mathsf{mpk}, \mathsf{msk})$ *in the support of* $\mathsf{Setup}(1^\lambda)$*, all functions* $f \in \mathcal{F}_\lambda$*, all secret keys* sk_f *in the support of* $\mathsf{KeyGen}(\mathsf{msk}, f)$*, and for all attributes* $x \in \mathcal{X}_\lambda$*, we have*

$$\Pr\big[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{mpk}, x)) = f(x)\big] = 1.$$

In the initial work of Katz et al. [KSW08], the authors only introduce the notion of selective security. The corresponding indistinguishability based adaptive security notion for predicate encryption has been introduced in [OT12]. We present a modification of this definition where the adversary has access to a challenge oracle to which it can submit multiple challenges instead of being able to only submit a single challenge. This security definition directly follows from the standard security definition using a simple hybrid argument.

Definition 33 (Indistinguishability-Based Attribute Hiding)**.** *Let* $\mathsf{ABE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a PE scheme for a function family* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ *as defined above. For* $\beta \in \{0, 1\}$*, we define the experiment* $\mathsf{AH}_\beta^{\mathsf{ABE}}$ *in Figure 2.8, where the left-or-right oracle is defined as:*

$\mathsf{QEncLR}_\beta(\cdot, \cdot)$**:** *After taking two attribute sets* x_0 *and* x_1 *the algorithm* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{msk}, x_\beta)$.

The advantage of an adversary $\mathcal{A}$ *is defined as:*

$$\mathsf{Adv}^{\mathsf{AH}}_{\mathsf{ABE},\mathcal{A}}(\lambda) = |\Pr[\mathsf{AH}_0^{\mathsf{ABE}}(1^\lambda, \mathcal{A}) = 1] - \Pr[\mathsf{AH}_1^{\mathsf{ABE}}(1^\lambda, \mathcal{A}) = 1]|.$$

We call an adversary valid *if for all queries* (x_0, x_1) *to the oracle* $\mathsf{QEncLR}_\beta(\cdot, \cdot)$ *and for any function* f *queried to the key generation oracle* $\mathsf{KeyGen}(\mathsf{msk}, \cdot)$*, we have* $f(x_0) = f(x_1)$ *(with probability* 1 *over the randomness of the adversary and the involved algorithms).*
A predicate-only predicate encryption scheme ABE *is called* attribute hiding *if for any valid polynomial-time adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that* $\mathsf{Adv}^{\mathsf{AH}}_{\mathsf{ABE},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

$\mathsf{AH}_\beta^{\mathsf{ABE}}(1^n, \mathcal{A})$

$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$

$\alpha \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot), \mathsf{QEncLR}_\beta(\cdot,\cdot)}(\mathsf{mpk})$

Output: α

Figure 2.8: Attribute-Hiding game of ABE. [BSW24]

2.12 NON-INTERACTIVE ZERO-KNOWLEDGE PROOFS

In this section, we introduce the notion of non-interactive zero knowledge (NIZK) proofs [For87, GMW87].

Definition 34 (Non-Interactive Zero-Knowledge Proofs)**.** *Let* R *be an NP Relation and consider the language* $L = \{x \mid \exists\, w$ *with* $(x, w) \in R\}$ *(where* x *is called a statement or instance). A non-interactive zero-knowledge proof (NIZK) for the relation* R *is a triple of PPT algorithms* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda)$: *Takes as input the unary representation of the security parameter* λ *and outputs a common reference string* CRS.

- $\mathsf{Prove}(\mathsf{CRS}, x, w)$*: Takes as input the common reference string* CRS*, a statement* x *and a witness* w*, and outputs a proof* π*.*
- $\mathsf{Verify}(\mathsf{CRS}, x, \pi)$*: Takes as input the common reference string* CRS*, a statement* x *and a proof* π*, and outputs* 0 *or* 1*.*

A system NIZK *is complete, if (for all* $\lambda \in \mathbb{N}$*), for all* CRS *in the support of* $\mathsf{Setup}(1^\lambda)$ *and all statement-witness pairs in the relation* $(x, w) \in R$*,*

$$\Pr[\mathsf{Verify}(\mathsf{CRS}, x, \mathsf{Prove}(\mathsf{CRS}, x, w)) = 1] = 1.$$

Besides completeness, a NIZK system should also fulfill the notions of soundness and zero-knowledge, which we introduce in the following two definitions:

$\mathsf{zk}_0^{\mathsf{NIZK}}(1^\lambda, \mathcal{A}, \mathsf{Sim})$	$\mathsf{zk}_1^{\mathsf{NIZK}}(1^\lambda, \mathcal{A}, \mathsf{Sim})$
$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$	$(\mathsf{CRS}, \tau) \leftarrow \mathsf{Sim}_1(1^\lambda)$
$\alpha \leftarrow \mathcal{A}^{\mathsf{Prove}(\mathsf{CRS},\cdot,\cdot)}(\mathsf{CRS})$	$\alpha \leftarrow \mathcal{A}^{\mathsf{Sim}'(\mathsf{CRS},\tau,\cdot,\cdot)}(\mathsf{CRS})$
Output: α	Output: α

Figure 2.9: Zero-knowledge property of NIZK. [BSW24]

Definition 35 (Zero-Knowledge)**.** *Let* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a NIZK proof system for a relation* R *and the corresponding language* L*,* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *a pair of algorithms (the simulator), with* $\mathsf{Sim}'(\mathsf{CRS}, \tau, x, w) = \mathsf{Sim}_2(\mathsf{CRS}, \tau, x)$ *for* $(x, w) \in R$*, and* $\mathsf{Sim}'(\mathsf{CRS}, \tau, x, w) = \mathsf{failure}$ *for* $(x, w) \notin R$*. For* $\beta \in \{0, 1\}$*, we define the experiment* $\mathsf{zk}_\beta^{\mathsf{NIZK}}(1^n, \mathcal{A})$ *in Figure 2.9. The associated advantage of an adversary* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) \coloneqq |\Pr[\mathsf{zk}_0^{\mathsf{NIZK}}(1^n, \mathcal{A}, \mathsf{Sim}) = 1] - \Pr[\mathsf{zk}_1^{\mathsf{NIZK}}(1^n, \mathcal{A}, \mathsf{Sim}) = 1]| \ .$$

A NIZK proof system NIZK *is called perfect zero-knowledge, with respect to a simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$*, if* $\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) = 0$ *for all algorithms* $\mathcal{A}$*, and computationally zero-knowledge, if* $\mathsf{Adv}^{\mathsf{zk}}_{\mathsf{NIZK},\mathcal{A},\mathsf{Sim}}(\lambda) \leq \mathsf{negl}(\lambda)$ *for all PPT algorithms* $\mathcal{A}$*.*

Besides zero-knowledge and soundness, we rely on the notion of extractability [CKLM12].

Definition 36 (Extractability)**.** *Let* $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a NIZK proof system for a relation* R *and the corresponding language* L*,* $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$ *a pair of algorithms (the extractor). We define the extraction advantages of an adversary* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathrm{CRS}}_{\mathsf{NIZK},\mathcal{A}} \coloneqq |\Pr[\mathsf{CRS} \leftarrow \mathsf{Setup}(1^n); 1 \leftarrow \mathcal{A}(\mathsf{CRS})] - \Pr[(\mathsf{CRS}, \mathsf{st}) \leftarrow \mathsf{Ext}_1(1^\lambda); 1 \leftarrow \mathcal{A}(\mathsf{CRS})]|,$$

and

$$\mathsf{Adv}^{\mathsf{Extract}}_{\mathsf{NIZK},\mathcal{A}}(n) \coloneqq \Pr\left[(\mathsf{CRS}_{\mathsf{Ext}}, \mathsf{st}_{\mathsf{Ext}}) \leftarrow \mathsf{Ext}_1(1^\lambda); \begin{matrix} \mathsf{Verify}(\mathsf{CRS}_{\mathsf{Ext}}, x, \pi) = 1 \wedge \\ R(x, \mathsf{Ext}_2(\mathsf{CRS}_{\mathsf{Ext}}, \mathsf{st}_{\mathsf{Ext}}, x, \pi)) = 0 \end{matrix}\right]$$

A NIZK proof system NIZK *is called extractable, with respect to an extractor* $\mathsf{Ext} = (\mathsf{Ext}_1, \mathsf{Ext}_2)$*, if* $\mathsf{Adv}^{\mathrm{CRS}}_{\mathsf{NIZK},\mathcal{A}} \leq \mathsf{negl}(n)$ *and* $\mathsf{Adv}^{\mathsf{Extract}}_{\mathsf{NIZK},\mathcal{A}}(n) \leq \mathsf{negl}(n)$*. Additionally, we call an extractable non-interactive zero-knowledge proof a non-interactive zero-knowledge proof of knowledge (NIZKPoK).*

2.13 UPDATABLE COMMON REFERENCE STRINGS

As defined in [GKM+18] updatable common reference strings (CRS) guarantees security if at least one of the users updates the CRS honestly.

Definition 37 (Updateable CRS)**.** *An updatable CRS scheme can be define by PPT algorithms* $Setup$*,* $Update$ *and a DPT algorithm* $VerifyCRS$ *that behave as follows:*

- $\mathsf{Setup}(1^\lambda)$*: Takes as input the unary representation of the security parameter* λ *and outputs a common reference string* CRS *and a proof of correctness* ρ*.*
- $\mathsf{Update}(1^\lambda, \mathsf{CRS}, (\rho_i)_{i=1}^n)$*: takes as input the security parameter, a common reference string, and a list of update proofs for the common reference string* $(\rho_i)_{i=1}^n$*. It outputs an updated common reference string* CRS′ *and a proof of the correctness* ρ' *of the update .*
- $\mathsf{VerifyCRS}(1^\lambda, \mathsf{CRS}, (\rho_i)_{i=1}^n)$*: Takes as input the security parameter, a common reference string, and a list of proofs. It outputs a bit indicating acceptance,* $b = 1$*, or rejection* $b = 0$*.*

Definition 38 (Correctness)**.** *An updatable CRS scheme is perfectly correct if,*

- *for all* $(\mathsf{CRS}, \rho) \leftarrow \mathsf{Setup}(1^\lambda)$ *we have* $\mathsf{VerifyCRS}(1^\lambda, \mathsf{CRS}, \rho) = 1$*;*
- *for all* $(1^\lambda, \mathsf{CRS}, (\rho_i)_{i=1}^n)$ *such that* $\mathsf{VerifyCRS}(1^\lambda, \mathsf{CRS}, (\rho_i)_{i=1}^n) = 1$
  we have for $(\mathsf{CRS}', \rho^{n+1}) \leftarrow \mathsf{Update}(1^\lambda, \mathsf{CRS}, (\rho_i)_{i=1}^n)$ *that* $\mathsf{VerifyCRS}(\mathsf{CRS}', \rho^{n+1}) = 1$*.*

CHAPTER 3

NI-TSPS-EQ

The use of SPS-EQ is crucial in the UL-PCS scheme for unlinkability, as it reduces dependency on heavy Zk-SNARKs and Zk proofs. As it is primarily needed in the UL-PCS scheme for transferring the issuance of attributes in generic policy schemes and when dealing with role-based policies (RBAC). In RBAC policies, SPS-EQ is utilized to handle the structure of attribute management and re-randomization differently than in the separable or generic policy schemes. As in the generic scheme signing the functional keys, instead the scheme signs witnesses for an accumulator that holds roles. SPS-EQ can keep the witnesses intact while enabling the re-randomization of public accumulator values, which is crucial for maintaining unlinkability [1.2]. In the context of the generic UL-PCS construction, SPS-EQ is used for the transportation of attribute issuance from a central authority to re-randomization. An initial encryption of attributes often done by a generalized version of pedersen commitments, is issued by the authority along with an SPS-EQ signature is provided on a vector related to these attributes. Combining homomorphic properties of commitments and the signature adaptation or unforgeability properties of SPS-EQ, a user can then generate a commitment to a scaled vector of its attributes and verify that this is done correctly.

Imagine having all the features in threshold setting. We can distribute all that trust between n number of authorities by using TSPS-EQ. To add extra efficiency in the system our proposed schemes requires no interactions among the authorities. So, it is possible to have a non interactive TSPS-EQ ? In this chapter we tried to answer this question in a positive manner.

As mentioned earlier, we begin with an observation about the threshold structure-preserving signature (TSPS) scheme proposed by Mitrokotsa et al. [MMS+24]. Their construction is a slightly modified version of the scheme by [KPW15], which results to the first TSPS scheme for general message spaces in the standard model that remains secure against adaptive adversaries. In this scheme, signature consists of four components and each issuer generates the first and second components of their partial signature on a message $[\mathbf{m}]_1 \in \mathbb{G}_1^\ell$ as follows:

$$\text{MMS}^+24: \ (\sigma_1, \sigma_2) := \left(\underbrace{\left[\begin{pmatrix} 1 & \mathbf{m}^\top \end{pmatrix}\right]_1 \mathbf{K}_i}_{\text{SP-OTS}} + \overbrace{\mathbf{r}_i^\top \left[\mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})\right]_1, \left[\mathbf{r}_i^\top \mathbf{B}^\top\right]_1}^{\text{randomized PRF}} \right),$$

where $\mathbf{K}$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U}$ and $\mathbf{V}$ are random matrices of appropriate dimensions.
Here, τ is a fresh random integer, and $\mathbf{r}$ is a newly sampled random vector of appropriate dimension.[1]

[1] Here, we follow the group notation by Escala *et al.* [EHK+17]. See Definition 1 for more details.

Additionally, the secret key for partial signing and verification keys are $\mathsf{sk}_i := \mathbf{K}_i$, for all $i \in [n]$ (where n is total number of signers) and $\mathsf{vk}_i := [\mathbf{K}_i\mathbf{A}]_2$. Accordingly, the global verification key is defined as $\mathsf{vk} := [\mathbf{K}\mathbf{A}]_2$.
Building on this scheme, our main concern is to make it *TSPS-EQ*. As in SPS-EQ, signing an equivalence class involves signing any representative, from which signatures on other representatives can be derived without the secret key. Obviously, this scheme does not support controlled malleability, as it is secure under standard EUF-CMA.

To overcome this challenge, the signer must sign the message $\left[\mathbf{m}^\top\right]_1$, instead of $\left[\begin{pmatrix}1 & \mathbf{m}^\top\end{pmatrix}\right]_1$. By removing $[1]_1$, as observed by [KPW15], the resulting scheme produces a homomorphic signature.[2] To achieve controlled malleability in our signature scheme, we introduce an additional secret key (namely, $[\mathbf{L}]_1$), along with the corresponding verification key (namely, $[\mathbf{L}\mathbf{A}]_2$). These modifications to the signature scheme of Mitrokotsa *et al.* [MMS+24] impacted the signing algorithm. Specifically, we had to adapt it by adding an extra component (the fifth component σ_5) to the signature. Consequently, the verification algorithm now requires three pairing product equations instead of two.

To be more precise, this adjustment involves defining the secret key as $\mathsf{sk} := (\mathsf{sk}^{(1)}, \mathsf{sk}^{(2)}) := (\mathbf{K}, \mathbf{L})$ and transferring the remaining parameters to the set of public parameters, i.e.,

$$\mathsf{pp} := ([\mathbf{A}]_2, [\mathbf{U}\mathbf{A}]_2, [\mathbf{V}\mathbf{A}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1)$$

We then adjust the SP-OTS part of the signature from $(\left[\begin{pmatrix}1 & \mathbf{m}^\top\end{pmatrix}\right]_1 \mathbf{K})$ to $(\left[\mathbf{m}^\top\mathbf{K} + \tau\mathbf{L}\right]_1)$ which supports controlled malleability for equivalence class messages and the verification key is defined as $\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)}) := ([\mathbf{K}\mathbf{A}]_2, [\mathbf{L}\mathbf{A}]_2)$. This rather simple structure allows to obtain the first TSPS-EQ for general message spaces in the standard model and can be proven secure in the TS-UF-1 model.

Similar to [MMS+24], consider the following setting. Imagine there are n signers, each equipped with their own signing key, either obtained through the involvement of a trusted dealer or by conducting a distributed key generation (DKG). Their collective objective is to generate a signature for a given message $[\mathbf{m}]_1 \in \mathbb{G}_1^\ell$. To have a linear structure of the SP-OTS $\{\left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i\right]_1\}_{i\in S}$ and a randomized PRF in respective spaces for each signers we employ a collision-resistant hash function (CRHF), $\mathcal{H}(.)$, to derive τ from $[\mathbf{m}]_1$. This gives the basis of our construction, where each signer $i \in [1, n]$ computes a partial signature on $[\mathbf{m}]_1$ as

$$(\sigma_1, \sigma_2) = \left(\left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i\right]_1 + \mathbf{r}_i^\top \left[\mathbf{B}^\top(\mathbf{U} + \tau \cdot \mathbf{V})\right]_1, \left[\mathbf{r}_i^\top\mathbf{B}^\top\right]_1\right) .$$

Here the signer i is holding the secret share $\mathsf{sk}_i := (\mathbf{K}_i, \mathbf{L}_i)$ and chooses a random quantity $\mathbf{r}_i$ of appropriate size and uses $\tau = \mathcal{H}([\mathbf{m}]_1)$. It is easy to verify that this signature can be aggregated in a non-interactive manner. Looking ahead, as a first step we prove that this construction achieves TS-UF-0 security, relying on the well-established and non-interactive standard assumption, i.e., the MDDH assumption.

In the following section, we provide a detail discussion of each algorithm that is defined in Definition 24. we have talked about each algorithm focusing on its correctness, ensuring that it functions as intended under all specified conditions and it's security properties, which assess the algorithm's ability towards attacks.

3.1 NON-INTERACTIVE TSPS-EQ

There are seven algorithms, we need to discuss namely Setup, KeyGen, ParSign, ParVerify, CombineSign, and ChgRep.

[2] A signature is called *homomorphic* if one can generate a valid signature on the span of all signed messages, without any control over the equivalence classes.

The Setup. Having taken input of the security parameter κ the algorithm $\mathsf{Setup}_{\mathcal{R}}(1^{\kappa})$ runs $\mathsf{ABSGen}(1^{\kappa})$ and sets $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{T}}, \mathsf{P}_1, \mathsf{P}_2, e)$. We are going to use asymmetric bilinear pairing [see 1], where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two different groups and $\mathbb{G}_{\mathsf{T}}$ is the target group. All three groups are cyclic groups of order p, where P_1 and P_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. After randomly taking matrices $\mathbf{A}$, $\mathbf{B}$ from the matrix distribution $\mathcal{D}_k$ [see 6] and matrices $\mathbf{U}$, $\mathbf{V}$ from the space $\mathbb{Z}_p^{(k+1)\times(k+1)}$, it outputs the public parameter pp as $\mathsf{pp} := \left([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, \left[\mathbf{B}^\top\mathbf{U}\right]_1, \left[\mathbf{B}^\top\mathbf{V}\right]_1\right)$. One can observe that no matrices are in the plain, $\mathbf{A}$ is in group $\mathbb{G}_2$, $\mathbf{B}$ is in group $\mathbb{G}_1$, and so on. This is the reason that having these matrices as public parameter won't leak any extra information to the adversary. Adversary can compute $\mathbf{U}$ by inverting $\mathbf{A}$ and doing product with $\mathbf{UA}$ on the right hand side. But she can only do a pairing product between $\mathbb{G}_1$ and $\mathbb{G}_2$ to have the result in $\mathbb{G}_{\mathsf{T}}$, not in the plain.

Key Generation. The algorithm $\mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, n, t)$ takes input the public parameters pp, and two positive integers n and t. The number n is the number of signers that are participating in the protocol and t is the threshold value for the corrupt signers in the protocol. To generate secret keys sk, first randomly taking two matrices $\mathbf{K}$ and $\mathbf{L}$ of size $\ell \times (k+1)$ and $1 \times (k+1)$ respectively from the space $\mathbb{Z}_p^{\ell\times(k+1)}$ and $\mathbb{Z}_p^{1\times(k+1)}$. To have partial secret keys sk_i, the algorithm uses shamir secret sharing [Definition 30] on those matrices $\mathbf{K}$ and $\mathbf{L}$. So it outputs all the shares of the secret key and corresponding partial verification keys vk_i, as well as verification key vk. It outputs $\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)}) := ([\mathbf{KA}]_2, [\mathbf{LA}]_2)$, $\mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}) := (\mathbf{K}_i, \mathbf{L}_i)$ and $\mathsf{vk}_i := (\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)}) := ([\mathbf{K}_i\mathbf{A}]_2, [\mathbf{L}_i\mathbf{A}]_2)$. The correctness of secret sharing ensures that with sufficient number of shares of secret one can obtain the actual secret by using Rec. For the security of secret sharing, having all the shares that are under influence of the adversary, they can reconstruct the secret with negligible probability.

Partial signatures. Each signer runs algorithm $\mathsf{ParSign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}_i, [\mathbf{m}]_1)$, which takes inputs the public parameter pp, their share of secret key sk_i, and the message $[\mathbf{m}]_1$. Given a collision resistant hash function, $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, [Definition 5] and message space $\mathcal{M} := \mathbb{G}_1^\ell$, each signer can produce the partial signatures. First it takes one randomness r_i from the space $\mathbb{Z}_p^k$, then it creates a tag $\tau := \mathcal{H}([\mathbf{m}]_1)$. Creation of these tags helps all the signers to have a common randomness among themselves, without doing any extra interactions between them. So, the partial signatures on message $[\mathbf{m}]_1$ done by i^{th} signer is $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ s.t. $\sigma_1 := \left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i\right]_1 + \mathbf{r}_i^\top\left[\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})\right]_1$, $\sigma_2 := \left[\mathbf{r}_i^\top\mathbf{B}^\top\right]_1$, $\sigma_3 := \left[\tau\mathbf{r}_i^\top\mathbf{B}^\top\right]_1$, $\sigma_4 := [\tau]_2$, $\sigma_5 := [\tau]_1$. Loosely speaking, having common randomness τ for all the signers makes the signature scheme threshold friendly [3]. Observe that first signature component σ_1 has product of τ and i^{th} share of secret key component $\mathbf{L}_i$, still it fulfills the requirements of threshold friendliness.

Partial Signature verification. In threshold signatures, verifying each partial signatures produced by each signers is crucial for early error detection in the system. To verify the partial signatures Σ_i produced by i^{th} signer, the algorithm $\mathsf{ParVerify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}_i, [\mathbf{m}]_1, \Sigma_i)$ takes public parameters pp, partial verification keys vk_i, the message $[\mathbf{m}]_1$ and the partial signature Σ_i. This algorithm essentially checks three pairing product equations (PPE) and outputs 1 if the checks hold, otherwise outputs 0. The left hand side of the first PPE has first component of the signature σ_1, and the right hand side takes σ_2,σ_3, and σ_5 along with the public parameters $[\mathbf{UA}]_2$ and $[\mathbf{VA}]_2$. Let us observe the mathematical identity of the PPE for correctness,

[3] Three Requirements to call a SPS threshold friendly, (i) SPS should not have inversion on randomness ($\frac{1}{r}$) and/or on secret shares($\frac{1}{sk}$). (ii) SPS should not have multiplication operations between randomness and secret shares ($r_i \cdot sk_i$). (iii) SPS should not any non-identity powers (r_i^x) on randomness and/or on secret shares.

$$
\begin{aligned}
e(\sigma_1, [\mathbf{A}]_2) &= e\left(\left[\mathbf{m}^\top \mathbf{K}_i + \tau \mathbf{L}_i\right]_1 + \mathbf{r}_i^\top \left[\mathbf{B}^\top(\mathbf{U} + \tau \mathbf{V})\right]_1, [\mathbf{A}]_2\right) \\
\implies &= e\left(\left[\mathbf{m}^\top\right]_1, [\mathbf{K}_i\mathbf{A}]_2\right) \cdot e\left([\tau \mathbf{L}_i]_1, [\mathbf{A}]_2\right) \cdot e\left(\left[\mathbf{r}_i^\top \mathbf{B}^\top\right]_1, [\mathbf{U}\mathbf{A}]_2\right) \cdot e\left(\left[\tau \mathbf{r}_i^\top \mathbf{B}^\top\right]_1, [\mathbf{V}\mathbf{A}]_2\right) \\
\implies &= e\left(\left[\mathbf{m}^\top\right]_1, [\mathbf{K}_i\mathbf{A}]_2\right) \cdot e\left([\tau]_1, [\mathbf{L}_i\mathbf{A}]_2\right) \cdot e\left(\sigma_2, [\mathbf{U}\mathbf{A}]_2\right) \cdot e\left(\sigma_3, [\mathbf{V}\mathbf{A}]_2\right) \\
\implies &= e\left(\left[\mathbf{m}^\top\right]_1, \mathsf{vk}_i^{(1)}\right) \cdot\ e\left(\sigma_5, \mathsf{vk}_i^{(2)}\right) \cdot\ e\left(\sigma_2, [\mathbf{U}\mathbf{A}]_2\right) \cdot\ e\left(\sigma_3, [\mathbf{V}\mathbf{A}]_2\right).
\end{aligned}
$$

Next PPE checks the pairings between σ_2 and σ_4 with σ_3 and identity element of $\mathbb{G}_2$.

$$
\begin{aligned}
e\left(\sigma_2, \sigma_4\right) &= e\left(\left[\mathbf{r}_i^\top \mathbf{B}^\top\right]_1, [\tau]_2\right) \\
\implies &= e\left(\left[\tau \mathbf{r}_i^\top \mathbf{B}^\top\right]_1, [1]_2\right) \\
\implies &= e(\sigma_3, [1]_2).
\end{aligned}
$$

The last PPE checks consistency between σ_4 and σ_5. One can observe that those two components are same in value while being in different groups. So, checking this PPE will be invariant in target group $\mathbb{G}_\mathsf{T}$.

$$
\begin{aligned}
e\left(\sigma_5, [1]_2\right) &= e\left([\tau]_1, [1]_2\right) \\
\implies &= e\left([1]_1, [\tau]_2\right) \\
\implies &= e([1]_1, \sigma_4).
\end{aligned}
$$

Combining Partial Signatures. Given that the algorithm ParVerify outputs 1, and having sufficient number of partial signatures Σ_i one can combine the partial signatures into a signature Σ for the message $[\mathbf{m}]_1$. For a set $S \subseteq [1, n]$ such that $|S| \geq t$ of honest signers having $\{\mathsf{sk}_i\}_{i \in S}$ always combines to correct signature Σ. To achieve non-interactivity, during the partial signing phase the tag should be the hash of message $\mathbf{m}$, to make sure that every signer picking the same tag, which is important for the reconstruction. First, the algorithm parses Σ_i as $(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, \sigma_4, \sigma_5)$ for all $i \in S$. As one can observe that the fourth and fifth components of partial signatures are consistent as $[\tau]_1$ and $[\tau]_2$. So, in the combined signature those components are $\widehat{\sigma}_4 := \sigma_4$ and $\widehat{\sigma}_5 := \sigma_5$. Let us look at other three components σ_1, σ_2, and σ_3. Combining these components by using Shamir secret sharing [Definition 30] in the exponent. So, raising shares of each components by its corresponding lagrange coefficients λ_i [Definition 29] and then doing product over all the shares indexed from the set S. We will get,

$$
\begin{aligned}
\widehat{\sigma}_1 := \prod_{i \in S} \sigma_{i,1}^{\lambda_i} &= \left[\mathbf{m}^\top \sum_{i \in S} \lambda_i \mathbf{K}_i + \tau \sum_{i \in S} \lambda_i \mathbf{L}_i\right]_1 + \sum_{i \in S} \lambda_i \mathbf{r}_i^\top \left[\mathbf{B}^\top(\mathbf{U} + \tau \mathbf{V})\right]_1 \\
\implies &= \left[\mathbf{m}^\top \mathbf{K} + \tau \mathbf{L}\right]_1 + \mathbf{r}^\top \left[\mathbf{B}^\top(\mathbf{U} + \tau \mathbf{V})\right]_1
\end{aligned}
$$

Similarly for $\widehat{\sigma}_2$ and $\widehat{\sigma}_3$,

$$
\begin{aligned}
\widehat{\sigma}_2 := \prod_{i \in S} \sigma_{i,2}^{\lambda_i} &= \left[\sum_{i \in S} \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top\right]_1 \\
\implies &= \left[\mathbf{r}^\top \mathbf{B}^\top\right]_1.
\end{aligned}
$$

And

$$
\begin{aligned}
\widehat{\sigma}_3 := \prod_{i \in S} \sigma_{i,3}^{\lambda_i} &= \left[\sum_{i \in S} \tau \lambda_i \mathbf{r}_i^\top \mathbf{B}^\top\right]_1 \\
\implies &= \left[\tau \mathbf{r}^\top \mathbf{B}^\top\right]_1.
\end{aligned}
$$

Here we are relying on the correctness and security properties of the Shamir's Secret Sharing Scheme.

Verification. To verify the combine signature Σ the algorithm $\mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}]_1, \Sigma)$ takes public parameters pp, verification key vk, the message $[\mathbf{m}]_1$, and the combined signature Σ. The algorithm checks same PPEs as it has checked in ParVerify. Here using combined signature instead of partial signature and using verification keys instead of shares of them.

Signature adaptation. The probabilistic algorithm ChgRep gives TSPS scheme a controlled form of malleability, which is necessary for TSPS-EQ. By controlled form of malleability, we meant that knowing a signature for any message $\mathbf{M}$ from message space will make sure knowing the signatures of all the corresponding messages that are in the same class of $\mathbf{M}$. Here we will call $\mathbf{M}$ as representative of its class. The algorithm $\mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}]_1, \mu, \Sigma)$ takes a scaler $\mathbf{m} = (m_i)_{i=1}^{l}$, public key vk, representative message $\mathbf{m}$ from message space $(\mathbb{G}_1^*)^l$, corresponding signature Σ as inputs and outputs a signature Σ' corresponding to the new representative $\mathbf{m}' \leftarrow \mu \cdot (m_i)_{i=1}^{l}$. Since, we are treating the collision resistance hash function as random oracle one can observe that the distributions of algorithms "ParSign, ParVerify, and CombineSign" together and the distribution of ChgRep algorithm is indistinguishable to a PPT adversary [4]. So, the perfect adaptation holds. The verification for the adapted signature will happen by the Verify algorithm using the product of scaler μ and hash of $\mathbf{m}$, not the hash of $\mu\mathbf{m}$ as the new tag.

Non-Interactivity. To combine these algorithms in one signature scheme, one signer (say, P_i) who wants to sign a message $\mathbf{m}$ will do the following. After getting the secret key $\mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)})$ from the trusted dealer, it signs the message and outputs the partial signature Σ_i. The party who wants to have the signature will collect t number of correctly produced (correctness will be checked by ParVerify) partial signatures and compute the aggregated signature Σ by running CombineSign. As one can observe that no interaction is required among the signers. In the next section we have discussed the construction in details.

3.1.1 Threshold SPS-EQ: Construction

Given a collision resistant hash function, $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$, and message space $\mathcal{M} := \mathbb{G}_1^\ell$, we present our (n,t)-TSPS-EQ construction in Figure 3.1. This consists of seven main PPT algorithms: Setup, KeyGen, ParSign, ParVerify, CombineSign, Verify and ChgRep, as defined in Definition 24. Similar to the settings of Bellare *et al.* [BCK+22], we also assume there is a dealer who is responsible for generating key pairs for all signers and a general verification key.

[4] Indistinguishable as long as the adversary don't know the scelar μ.

$\mathsf{Setup}_{\mathcal{R}}(1^\kappa)$:

1: $\mathcal{G} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, \mathsf{P}_1, \mathsf{P}_2, e) \leftarrow \mathsf{ABSGen}(1^\kappa)$.
2: $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$, $\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$.
3: $\mathsf{pp} := \left([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, \left[\mathbf{B}^\top\mathbf{U}\right]_1, \left[\mathbf{B}^\top\mathbf{V}\right]_1\right)$.

$\mathsf{KeyGen}_{\mathcal{R}}(\mathsf{pp}, n, t)$:

1: $\mathbf{K} \leftarrow \mathbb{Z}_p^{\ell\times(k+1)}$,
2: $\mathbf{L} \leftarrow \mathbb{Z}_p^{1\times(k+1)}$.
3: $\mathbf{K}_1, \ldots, \mathbf{K}_n \leftarrow \mathsf{Share}(\mathbf{K}, \mathbb{Z}_p^{(\ell+1)\times(k+1)}, n, t)$,
4: $\mathbf{L}_1, \ldots, \mathbf{L}_n \leftarrow \mathsf{Share}(\mathbf{L}, \mathbb{Z}_p^{1\times(k+1)}, n, t)$.
5: Set $\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)}) := ([\mathbf{KA}]_2, [\mathbf{LA}]_2)$, $\mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}) := (\mathbf{K}_i, \mathbf{L}_i)$ and $\mathsf{vk}_i := (\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)}) := ([\mathbf{K}_i\mathbf{A}]_2, [\mathbf{L}_i\mathbf{A}]_2)$.

$\mathsf{ParSign}_{\mathcal{R}}(\mathsf{pp}, \mathsf{sk}_i, [\mathbf{m}]_1)$:

1: $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$.
2: $\tau := \mathcal{H}([\mathbf{m}]_1)$.
3: Output $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$ s.t.
$\sigma_1 := \left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i\right]_1 + \mathbf{r}_i^\top\left[\mathbf{B}^\top(\mathbf{U}+\tau\mathbf{V})\right]_1$, $\sigma_2 := \left[\mathbf{r}_i^\top\mathbf{B}^\top\right]_1$, $\sigma_3 := \left[\tau\mathbf{r}_i^\top\mathbf{B}^\top\right]_1$, $\sigma_4 := [\tau]_2$, $\sigma_5 := [\tau]_1$.

$\mathsf{ParVerify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}_i, [\mathbf{m}]_1, \Sigma_i)$: Output 1 if the following checks hold; else output 0.

1: $e(\sigma_1, [\mathbf{A}]_2) = e\left(\left[\mathbf{m}^\top\right]_1, \mathsf{vk}_i^{(1)}\right) \cdot e\left(\sigma_5, \mathsf{vk}_i^{(2)}\right) \cdot e(\sigma_2, [\mathbf{UA}]_2) \cdot e(\sigma_3, [\mathbf{VA}]_2)$.
2: $e(\sigma_2, \sigma_4) = e(\sigma_3, [1]_2)$.
3: $e(\sigma_5, [1]_2) = e([1]_1, \sigma_4)$.

$\mathsf{CombineSign}_{\mathcal{R}}(\mathsf{pp}, S, \{\Sigma_i\}_{i\in S})$:

1: Parse $\Sigma_i = (\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}, \sigma_4, \sigma_5)$ for all $i \in S$.
2: Compute Lagrange polynomials λ_i for $i \in S$.
3: Output $\Sigma := (\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5)$ s.t.
$\widehat{\sigma}_1 := \prod_{i\in S}\sigma_{i,1}^{\lambda_i} = \left[\mathbf{m}^\top\sum_{i\in S}\lambda_i\mathbf{K}_i + \tau\sum_{i\in S}\lambda_i\mathbf{L}_i\right]_1 + \sum_{i\in S}\lambda_i\mathbf{r}_i^\top\left[\mathbf{B}^\top(\mathbf{U}+\tau\mathbf{V})\right]_1 = \left[\mathbf{m}^\top\mathbf{K} + \tau\mathbf{L}\right]_1 + \mathbf{r}^\top\left[\mathbf{B}^\top(\mathbf{U}+\tau\mathbf{V})\right]_1$,
$\widehat{\sigma}_2 := \prod_{i\in S}\sigma_{i,2}^{\lambda_i} = \left[\sum_{i\in S}\lambda_i\mathbf{r}_i^\top\mathbf{B}^\top\right]_1 = \left[\mathbf{r}^\top\mathbf{B}^\top\right]_1$,
$\widehat{\sigma}_3 := \prod_{i\in S}\sigma_{i,3}^{\lambda_i} = \left[\sum_{i\in S}\tau\lambda_i\mathbf{r}_i^\top\mathbf{B}^\top\right]_1 = \left[\tau\mathbf{r}^\top\mathbf{B}^\top\right]_1$, $\widehat{\sigma}_4 := \sigma_4$, and $\widehat{\sigma}_5 := \sigma_5$.

$\mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}]_1, \Sigma)$: Output 1 if the following checks satisfy; else output 0.

1: $e(\widehat{\sigma}_1, [\mathbf{A}]_2) = e\left(\left[\mathbf{m}^\top\right]_1, \mathsf{vk}^{(1)}\right) \cdot e\left(\widehat{\sigma}_5, \mathsf{vk}^{(2)}\right) \cdot e(\widehat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\widehat{\sigma}_3, [\mathbf{VA}]_2)$.
2: $e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2)$.
3: $e(\widehat{\sigma}_5, [1]_2) = e([1]_1, \widehat{\sigma}_4)$.

$\mathsf{ChgRep}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}]_1, \mu, \Sigma)$: On input a representative $\mathbf{m} = (m_i)_{i=1}^l \in (\mathbb{G}_1^*)^l$ of equivalence class $[\mathbf{m}]_\mathcal{R}$, corresponding signature Σ, $\mu \in \mathbb{Z}_p^*$ and $\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)})$, returns $\perp$ if $\mathsf{Verify}_{\mathcal{R}}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}]_1, \Sigma) = 0$. Otherwise return $(\mathbf{m}', \Sigma')$, where $\Sigma' \leftarrow (\mu\sigma_1, \mu\sigma_2, \mu\sigma_3, \mu\sigma_4, \mu\sigma_5)$ is the the updated signature for the new representative $\mathbf{m}' \leftarrow \mu \cdot (m_i)_{i=1}^l$.

Figure 3.1: Our proposed TSPS-EQ construction

3.1.2 Threshold SPS-EQ: Security

First we prove our proposed scheme achieves TS-UF-0 security for the TSPS part and then we will prove the signatures on equivalence class part by using updatable CRS model.

Theorem 1. *Under the* $\mathcal{D}_k$-MDDH *Assumption in* $\mathbb{G}_1$ *and* $\mathcal{D}_k$-KerMDH *Assumption in* $\mathbb{G}_2$*, the proposed Threshold Structure-Preserving Signature construction in Figure 3.1 achieves* TS-UF-0 *security against an efficient adversary making at most* q *partial signature queries.*

Proof. We prove the above theorem through a series of games and we use $\mathbf{Adv}_i$ to denote the advantage of the adversary $\mathcal{A}$ in winning the Game i. The games are described below.

Game 0. This is the TS-UF-0 security game described in Definition 26. As shown in Figure 3.2, an adversary $\mathcal{A}$ after receiving the set of public parameters, pp, returns (n, t, CS), where n, t and CS represents the total number of signers, the threshold, and the set of corrupted signers, respectively. The adversary can query the partial signing oracle $\mathcal{O}^{\mathsf{PSign}}(\cdot)$ to receive partial signatures and q represents the total number of these queries. In the end, the adversary outputs a message $[\mathbf{m}^*]_1$ and a forged signature Σ^*.

Game 1. We modify the verification procedure to the one described in Figure 3.3. Consider any forged message/signature pair $([\mathbf{m}^*]_1, \Sigma^* = (\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5))$, where $e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2)$, $e(\widehat{\sigma}_5, [1]_{\mathbf{G}_2}) = e([1]_{\mathbf{G}_1}, \widehat{\sigma}_4)$, $|\mathsf{CS}| < t$ and $S_1([\mathbf{m}^*]_1) = \emptyset$. It is easy to observe that if the pair $([\mathbf{m}^*]_1, \Sigma^*)$ meets the $\mathsf{Verify}^*(\cdot)$ criteria, outlined in Figure 3.3, it also satisfies $\mathsf{Verify}(\cdot)$ procedure, described in Figure 3.2. This is primarily due to the fact that:

$$\begin{aligned} & e(\widehat{\sigma}_1, [\mathbf{A}]_2) = e\left([\mathbf{m}^{*\top}]_1, [\mathbf{K}\mathbf{A}]_2\right) \cdot e(\widehat{\sigma}_5, [\mathbf{L}\mathbf{A}]_2) \cdot e(\widehat{\sigma}_2, [\mathbf{U}\mathbf{A}]_2) \cdot e(\widehat{\sigma}_3, [\mathbf{V}\mathbf{A}]_2) \\ \Longleftarrow\ & e(\widehat{\sigma}_1, [1]_2) = e([\mathbf{m}^{*\top}]_1, [\mathbf{K}]_2) \cdot e(\widehat{\sigma}_5, [\mathbf{L}]_2) \cdot e(\widehat{\sigma}_2, [\mathbf{U}]_2) \cdot e(\widehat{\sigma}_3, [\mathbf{V}]_2) \\ \Longleftrightarrow\ & e(\widehat{\sigma}_1, [1]_2) = e([\mathbf{m}^{*\top}\mathbf{K} + \tau^*\mathbf{L}]_1, [1]_2) \cdot e(\widehat{\sigma}_2, [\mathbf{U} + \tau^*\mathbf{V}]_2). \end{aligned}$$

Assume there exists a message/signature pair like $([\mathbf{m}^*]_1, \Sigma^* = (\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5))$ that satisifies $\mathsf{Verify}(\cdot)$ and not $\mathsf{Verify}^*(\cdot)$, then we can compute a non-zero vector $\mathbf{c}$ in the kernal of $\mathbf{A}$ as follows:

$$\mathbf{c} := \widehat{\sigma}_1 - ([\mathbf{m}^{*\top}\mathbf{K} + \tau^*\mathbf{L}]_1 + \widehat{\sigma}_2\mathbf{U} + \widehat{\sigma}_3\mathbf{V}) \in \mathbb{G}_1^{1\times(k+1)}.$$

According to $\mathcal{D}_k$-KerMDH assumption over $\mathbb{G}_2$ described in Definition 8, computing such a vector $\mathbf{c}$ is considered computationally hard. Thus,

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

$\underline{\mathsf{Verify}^*(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*)}$:
1: Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4 = [\tau^*]_2, \widehat{\sigma}_5 = [\tau^*]_1)$.
2: **return** $\Big(e(\widehat{\sigma}_1, [1]_2) = e\left([\mathbf{m}^{*\top}\mathbf{K} + \tau^*\mathbf{L}]_1, [1]_2\right) \cdot e(\widehat{\sigma}_2, [\mathbf{U} + \tau^*\mathbf{V}]_2) \wedge$
$e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2) \ \wedge e(\widehat{\sigma}_5, [1]_2) = e([1]_1, \widehat{\sigma}_4)\Big)$

Figure 3.3: Modifications in Game_1

Game 2. On receiving a partial signature query on a message $[\mathbf{m}_i]_1$, the query list is updated to include the message $[\mathbf{m}_i]_1$ along with its corresponding tag, $\tau_i := \mathcal{H}([\mathbf{m}_i]_1)$. The challenger aborts if an adversary can generate two tuples $([\mathbf{m}_i]_1, \tau_i)$, $([\mathbf{m}_j]_1, \tau_j)$ with $[\mathbf{m}_i]_1 \neq [\mathbf{m}_j]_1$ and $\tau_i = \tau_j$. By the collision resistance property of the underlying hash function we have,

$$|\mathbf{Adv}_1 - \mathbf{Adv}_2| \leq Adv^{\mathsf{CRHF}}_{\mathcal{H}}(\kappa) \cdot$$

$\underline{\boldsymbol{G}_0(\kappa)}$:

$\mathcal{G} \leftarrow \mathsf{ABSGen}(1^\kappa)$,
$\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$,
$\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$.
$\mathsf{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1)$.
$(n, t, \mathsf{CS}, \mathsf{st}_0) \leftarrow \mathcal{A}(\mathsf{pp})$.
Assert $\mathsf{CS} \subset [1, n]$.
Sample $\mathbf{K} \leftarrow \mathbb{Z}_p^{\ell\times(k+1)}$.
Sample $\mathbf{L} \leftarrow \mathbb{Z}_p^{1\times(k+1)}$.
$(\mathbf{K}_1, \ldots, \mathbf{K}_n) \leftarrow \mathsf{Share}(\mathbf{K}, \mathbb{Z}_p^{\ell\times(k+1)}, n, t)$.
$(\mathbf{L}_1, \ldots, \mathbf{L}_n) \leftarrow \mathsf{Share}(\mathbf{L}, \mathbb{Z}_p^{1\times(k+1)}, n, t)$.
$\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)}) := ([\mathbf{KA}]_2, [\mathbf{LA}]_2)$.
for $i \in [1, n]$:
  $\mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}) := (\mathbf{K}_i, \mathbf{L}_i)$; $\mathsf{vk}_i := (\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)}) := ([\mathbf{K}_i\mathbf{A}]_2, [\mathbf{L}_i\mathbf{A}]_2)$;
$([\mathbf{m}^*]_1, \Sigma^*, \mathsf{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PSign}(\cdot)}}\left(\mathsf{st}_0, \mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathsf{CS}}, \{\mathsf{vk}_i\}_{i\in[1,n]}\right)$.
return $(\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*) \wedge |\mathsf{CS}| < t \wedge S_1([\mathbf{m}^*]_1) = \emptyset)$

$\underline{\mathcal{O}^{\mathsf{PSign}}(i, [\mathbf{m}]_1)}$:

Assert $\left([\mathbf{m}]_1 \in \mathcal{M} \wedge i \in \mathsf{HS}\right)$.
$\mathbf{r}_i \leftarrow \mathbb{Z}_p^k$.
$\tau := \mathcal{H}([\mathbf{m}]_1)$.
$\sigma_1 := \left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i + \mathbf{r}_i^\top[\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]\right]_1$,
$\sigma_2 := [\mathbf{r}_i^\top\mathbf{B}^\top]_1$,
$\sigma_3 := [\tau\mathbf{r}_i^\top\mathbf{B}^\top]_1$,
$\sigma_4 := [\tau]_2$.
$\sigma_5 := [\tau]_1$.
$\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.
$\Sigma_i \neq \perp$:
  $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$.
return Σ_i

$\underline{\mathsf{Verify}(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*)}$:

Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5)$.
return $\Big(e(\widehat{\sigma}_1, [\mathbf{A}]_2) = e\left([\mathbf{m}^{*\top}]_1, [\mathbf{KA}]_2\right) \cdot e(\widehat{\sigma}_5, [\mathbf{LA}]_2) \cdot e(\widehat{\sigma}_2, [\mathbf{UA}]_2) \cdot e(\widehat{\sigma}_3, [\mathbf{VA}]_2)$
$\wedge\, e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2) \wedge\; e(\widehat{\sigma}_5, [1]_2) = e([1]_1, \widehat{\sigma}_4)\Big)$

Figure 3.2: Game_0

$\underline{\mathcal{O}^{\mathsf{PSign}^*}(i, [\mathbf{m}]_1):}$
1: Assert $\big([\mathbf{m}]_1 \in \mathcal{M} \ \wedge \ i \in \mathsf{HS}\big)$.
2: $\mathbf{r}_i \leftarrow \mathbb{Z}_p^k, \tau := \mathcal{H}([\mathbf{m}]_1), \mu \leftarrow \mathbb{Z}_p$.
3: $\sigma_1 := [\mathbf{m}^\top \mathbf{K}_i + \tau \mathbf{L}_i + \mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1$,
$\sigma_2 := [\mathbf{r}_i^\top \mathbf{B}^\top]_1$,
$\sigma_3 := [\tau \mathbf{r}_i^\top \mathbf{B}^\top]_1$,
$\sigma_4 := [\tau]_2$,
$\sigma_5 := [\tau]_1$.
4: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.
5: $\Sigma_i \neq \perp$:
6: $S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$.
7: **return** Σ_i

Figure 3.4: Modifications in Game_3

Game 3. In this game, we introduce randomness to the partial signatures by adding $\mu \mathbf{a}^\perp$ to each partial signature, where μ is chosen uniformly at random and the vector $\mathbf{a}^\perp$ is a non-zero vector in the kernel of $\mathbf{A}$. The new partial signatures satisfy the verification procedure as $\mathbf{a}^\perp \mathbf{A} = 0$. Figure 3.4 describes the new partial signing oracle, $\mathcal{O}^{\mathsf{PSign}^*}(\cdot)$.

Lemma 1. $|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq 2q Adv^{\mathsf{MDDH}}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}_1}(\kappa) + q/p$.

Proof. We prove this lemma through a reduction to the core lemma, Lemma 2. Let us assume there exists an adversary $\mathcal{A}$ that can distinguish the games Game_2 and Game_3, we can use it to build an adversary $\mathcal{B}_1$, defined in Figure 3.5, which breaks the core lemma, Lemma 2. The adversary $\mathcal{B}_1$ has access to four oracles, $\mathsf{Init}(\cdot), \mathcal{O}_b(\cdot), \mathcal{O}^*(\cdot), \mathcal{O}^{**}(\cdot)$, however in this reduction, we only use the first three oracles, defined as follows:

Oracle $\mathsf{Init}(\cdot)$**:** The oracle Init provides the set of public parameters pp.

Oracle $\mathcal{O}_b(\cdot)$**:** On the i-th query to this oracle on $[\tau]_1$, it outputs $\big([b\mu \mathbf{a}^\perp + \mathbf{r}_i^\top \mathbf{B}^\top (\mathbf{U} + \tau \cdot \mathbf{V})]_1, [\mathbf{r}_i^\top \mathbf{B}^\top]_1\big)$ depending on a random bit b.

Oracle $\mathcal{O}^*(\cdot)$**:** On input $[\tau^*]_2$, it returns $[\mathbf{U} + \tau^* \mathbf{V}]_2$.

When the lemma challenger selects the challenge bit as $b = 0$, it leads to the game Game_2, and when $b = 1$, it results in the game Game_3. All the other values are simulated perfectly. Thus, $|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq Adv^{\mathsf{Core}}_{\mathcal{D}_k, \mathsf{ABSGen}, \mathcal{B}_1}(\kappa)$ holds and therefore we have

$$|\mathbf{Adv}_2 - \mathbf{Adv}_3| \leq 2q Adv^{\mathsf{MDDH}}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}}(\kappa) + q/p. \qquad \square$$

Game 4. In this game, we apply the modifications described in Figure 3.6. Shamir secret sharing (see Definition 30) ensures that $(\mathbf{K}_1, \ldots, \mathbf{K}_n)$ & $(\mathbf{L}_1, \ldots, \mathbf{L}_n)$ in Game_3 and $(\widetilde{\mathbf{K}}_1, \ldots, \widetilde{\mathbf{K}}_n)$ & $(\widetilde{\mathbf{L}}_1, \ldots, \widetilde{\mathbf{L}}_n)$ in Game_4 have identical distributions respectively. W.l.o.g, $\mathbf{K}_i$ & $\mathbf{L}_i$ in Game_3 and $\widetilde{\mathbf{K}}_i$ & $\widetilde{\mathbf{L}}_i$ in Game_4 are identically distributed respectively. In Game_4, on the other hand, $\widetilde{\mathbf{K}}_i$ and $\mathbf{K}_i = \widetilde{\mathbf{K}}_i - \mathbf{u}_i \mathbf{a}^\perp$ are identically distributed as well as $\widetilde{\mathbf{L}}_i$ and $\mathbf{L}_i = \widetilde{\mathbf{L}}_i - \mathbf{v}_i \mathbf{a}^\perp$. Combining these observations, it follows that $\mathbf{K}_i$ & $\mathbf{L}_i$ in Game_3 and $\mathbf{K}_i$ & $\mathbf{L}_i$ in Game_4 are identically distributed for all $i \in [1, n]$. Consequently, it can be deduced that $\mathbf{K}$ in Game_3 and $\mathbf{K} + \mathbf{u}_0 \mathbf{a}^\perp$ in Game_4 are identically distributed as well as for the case of $\mathbf{L}$ in Game_3 and $\mathbf{L} + \mathbf{v}_0 \mathbf{a}^\perp$ in Game_4. Therefore, this change is just a conceptual change and we have,

$\underline{\mathcal{B}_1^{\mathsf{Init}(\cdot),\mathcal{O}_b(\cdot),\mathcal{O}^*(\cdot),\mathcal{O}^{**}(\cdot)}:}$

1: Assert $\big([\mathbf{m}]_1 \in \mathcal{M} \ \wedge \ i \in \mathsf{HS}\big)$.
2: $(\mathbf{A}, \mathbf{UA}, \mathbf{VA}, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1) \leftarrow \mathsf{Init}()$.
3: $\mathsf{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1)$.
4: $(n, t, \mathsf{CS}, \mathsf{st}_0) \leftarrow \mathcal{A}(\mathsf{pp})$.
5: Assert $\mathsf{CS} \subset [1, n]$.
6: Sample $\mathbf{K} \leftarrow \mathbb{Z}_p^{\ell\times(k+1)}$.
7: Sample $\mathbf{L} \leftarrow \mathbb{Z}_p^{1\times(k+1)}$.
8: $(\mathbf{K}_1, \ldots, \mathbf{K}_n) \leftarrow \mathsf{Share}(\mathbf{K}, \mathbb{Z}_p^{\ell\times(k+1)}, n, t)$.
9: $(\mathbf{L}_1, \ldots, \mathbf{L}_n) \leftarrow \mathsf{Share}(\mathbf{L}, \mathbb{Z}_p^{1\times(k+1)}, n, t)$.
10: $\mathsf{vk} := (\mathsf{vk}^{(1)}, \mathsf{vk}^{(2)}) := ([\mathbf{KA}]_2, [\mathbf{LA}]_2)$.
11: **for** $i \in [1, n]$:
12: $\quad \mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}) := (\mathbf{K}_i, \mathbf{L}_i)$; $\quad \mathsf{vk}_i := (\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)}) := ([\mathbf{K}_i\mathbf{A}]_2, [\mathbf{L}_i\mathbf{A}]_2)$.
13: **if** $(\mathbf{m}^*, \Sigma^*, \mathsf{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PSign}^*(\cdot)}}(\mathsf{st}_0, \mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathsf{CS}}, \{\mathsf{vk}_i\}_{i\in[1,n]})$ **then**
14: $\quad$ Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5)$
15: $\quad \mathsf{result} := \mathsf{Verify}^*(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*) \wedge |\mathsf{CS}| < t \wedge S_1([\mathbf{m}^*]_1) \stackrel{?}{=} \emptyset$
16: **end if**
17: **return** $\tilde{b} \leftarrow \mathcal{A}(\mathsf{result})$

$\underline{\mathcal{O}^{\mathsf{PSign}^*}(i, [\mathbf{m}]_1):}$

1: $\tau := \mathcal{H}([\mathbf{m}]_1)$.
2: $(val_1, val_2) \leftarrow \mathcal{O}_b(\tau)$.
3: $\sigma_1 := \left[\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i\right]_1 \cdot val_1$.
$\quad \sigma_2 := val_2$,
$\quad \sigma_3 := [\tau]_1 \cdot val_2$,
$\quad \sigma_4 := [\tau]_2$
$\quad \sigma_5 := [\tau]_1$.
4: $\Sigma_i := (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$.
5: $\Sigma_i \neq \perp$:
6: $\quad S_1([\mathbf{m}]_1) := S_1([\mathbf{m}]_1) \cup \{i\}$.
7: **return** Σ_i

$\underline{\mathsf{Verify}^*(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*):}$

1: Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4, \widehat{\sigma}_5)$.
2: **return** $\Big(e\left(\widehat{\sigma}_1, [1]_2\right) = e\left(\left[\mathbf{m}^{*\top}\mathbf{K} + \tau^*\mathbf{L}\right]_1, [1]_2\right) \cdot \ e(\widehat{\sigma}_2, \mathcal{O}^*(\widehat{\sigma}_4))$
$\qquad \wedge\, e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2) \ \wedge e(\widehat{\sigma}_5, [1]_2) = e([1]_1, \widehat{\sigma}_4)\Big)$

Figure 3.5: Reduction to the core lemma in Lemma 2

$\boldsymbol{G}_3(\kappa)$: $\boldsymbol{G}_4(\kappa)$:

$\mathcal{G} \leftarrow \mathsf{ABSGen}(1^\kappa)$,
$\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$,
$\mathbf{U}, \mathbf{V} \leftarrow \mathbb{Z}_p^{(k+1)\times(k+1)}$.
$\mathsf{pp} := ([\mathbf{A}]_2, [\mathbf{UA}]_2, [\mathbf{VA}]_2, [\mathbf{B}]_1, [\mathbf{B}^\top\mathbf{U}]_1, [\mathbf{B}^\top\mathbf{V}]_1)$.
$(n, t, \mathsf{CS}, \mathsf{st}_0) \leftarrow \mathcal{A}(\mathsf{pp})$.
Assert $\mathsf{CS} \subset [1, n]$.
Sample $\mathbf{K} \leftarrow \mathbb{Z}_p^{\ell\times(k+1)}$.
Sample $\mathbf{L} \leftarrow \mathbb{Z}_p^{1\times(k+1)}$.
$(\mathbf{K}_1, \ldots, \mathbf{K}_n) \leftarrow \mathsf{Share}(\mathbf{K}, \mathbb{Z}_p^{\ell\times(k+1)}, n, t)$
$(\mathbf{L}_1, \ldots, \mathbf{L}_n) \leftarrow \mathsf{Share}(\mathbf{L}, \mathbb{Z}_p^{1\times(k+1)}, n, t)$
Sample $\mathbf{u}_0 \leftarrow \mathbf{Z}_p^\ell$ and $\mathbf{v}_0 \leftarrow \mathbf{Z}_p$
$(\mathbf{u}_1, \ldots, \mathbf{u}_n) \leftarrow \mathsf{Share}(\mathbf{u}_0, \mathbb{Z}_p^\ell, n, t)$
$(\mathbf{v}_1, \ldots, \mathbf{v}_n) \leftarrow \mathsf{Share}(\mathbf{v}_0, \mathbb{Z}_p, n, t)$
$(\widetilde{\mathbf{K}}_1, \ldots, \widetilde{\mathbf{K}}_n) \leftarrow \mathsf{Share}(\mathbf{K}, \mathbb{Z}_p^{\ell\times(k+1)}, n, t)$
$(\widetilde{\mathbf{L}}_1, \ldots, \widetilde{\mathbf{L}}_n) \leftarrow \mathsf{Share}(\mathbf{L}, \mathbb{Z}_p^{1\times(k+1)}, n, t)$
$\mathbf{K}_i := \widetilde{\mathbf{K}}_i + \mathbf{u}_i\mathbf{a}^\perp, \forall i \in [1, n]$ and $\mathbf{L}_i := \widetilde{\mathbf{L}}_i + \mathbf{v}_i\mathbf{a}^\perp, \forall i \in [1, n]$
$\mathsf{vk}^{(1)} := [\mathbf{KA}]_2$, and $\mathsf{vk}^{(2)} := [\mathbf{LA}]_2$.
for $i \in [1, n]$:
  $\mathsf{sk}_i := (\mathsf{sk}_i^{(1)}, \mathsf{sk}_i^{(2)}) := (\mathbf{K}_i, \mathbf{L}_i)$; $\mathsf{vk}_i := (\mathsf{vk}_i^{(1)}, \mathsf{vk}_i^{(2)}) := ([\mathbf{K}_i\mathbf{A}]_2, [\mathbf{L}_i\mathbf{A}]_2)$.
$([\mathbf{m}^*]_1, \Sigma^*, \mathsf{st}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{PSign}(\cdot)}}(\mathsf{st}_0, \mathsf{vk}, \{\mathsf{sk}_i\}_{i\in\mathsf{CS}}, \{\mathsf{vk}_i\}_{i\in[1,n]})$.
return $\left(\mathsf{Verify}^*(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*) \wedge\ |\mathsf{CS}| < t\ \wedge\ S_1([\mathbf{m}^*]_1) = \emptyset\right)$

$\mathsf{Verify}^*(\mathsf{pp}, \mathsf{vk}, [\mathbf{m}^*]_1, \Sigma^*)$:

Parse Σ^* as $(\widehat{\sigma}_1, \widehat{\sigma}_2, \widehat{\sigma}_3, \widehat{\sigma}_4 = [\tau^*]_2, \widehat{\sigma}_5 = [\tau^*]_1)$.
return $\Big(e(\widehat{\sigma}_1, [1]_2) = e\left(\left[\mathbf{m}^{*\top}(\mathbf{K} + \boxed{\mathbf{u}_0\mathbf{a}^\perp}) + \tau^*(\mathbf{L} + \boxed{\mathbf{v}_0\mathbf{a}^\perp})\right]_1, [1]_2\right)$
$e(\widehat{\sigma}_2, [\mathbf{U} + \tau^*\mathbf{V}]_2) \wedge\ e(\widehat{\sigma}_2, \widehat{\sigma}_4) = e(\widehat{\sigma}_3, [1]_2)$
$\wedge e(\widehat{\sigma}_5, [1]_2) = e([1]_1, \widehat{\sigma}_4)\Big)$

Figure 3.6: Modification from Game_3 to Game_4

$$|\mathbf{Adv}_3 - \mathbf{Adv}_4| = 0.$$

Now, we give a bound on $\mathbf{Adv}_4$ via an information-theoretic argument. We first consider the information about $\mathbf{u}_0$ (and subsequently $\{\mathbf{u}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$) and $\mathbf{v}_0$ (and subsequently $\{\mathbf{v}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$) leaked from $\mathsf{vk}^{(1)}$ (and subsequently $\{\mathsf{vk}_i^{(1)}\}_{i\in[1,n]}$) and $\mathsf{vk}^{(2)}$ (and subsequently $\{\mathsf{vk}_i^{(2)}\}_{i\in[1,n]}$) respectively and partial signing queries:

- $\mathsf{vk}^{(1)} := [\mathbf{KA}]_2 = \left[\widetilde{\mathbf{K}}\mathbf{A}\right]_2$ and $\mathsf{vk}_i^{(1)} := [\mathbf{K}_i\mathbf{A}]_2 = \left[\widetilde{\mathbf{K}}_i\mathbf{A}\right]_2$ for all $i \in [1, n]$.
- $\mathsf{vk}^{(2)} := [\mathbf{LA}]_2 = \left[\widetilde{\mathbf{L}}\mathbf{A}\right]_2$ and $\mathsf{vk}_i^{(2)} := [\mathbf{L}_i\mathbf{A}]_2 = \left[\widetilde{\mathbf{L}}_i\mathbf{A}\right]_2$ for all $i \in [1, n]$.
- The output of the j^{th} partial signature query on $(i, [\mathbf{m}]_1)$ for $[\mathbf{m}]_1 \neq [\mathbf{m}^*]_1$ completely hides $\{\mathbf{u}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$ (and subsequently $\mathbf{u}_0$ as the adversary has only $|\mathsf{CS}|$ many $\mathbf{u}_i$ with $|\mathsf{CS}| < t$) as well as $\{\mathbf{v}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$ (and subsequently $\mathbf{v}_0$ as the adversary has only $|\mathsf{CS}|$ many $\mathbf{v}_i$ with $|\mathsf{CS}| < t$), since

  $$\mathbf{m}^\top\mathbf{K}_i + \tau\mathbf{L}_i + \mu_j\mathbf{a}^\perp = \mathbf{m}^\top\widetilde{\mathbf{K}}_i + \tau\widetilde{\mathbf{L}}_i + \tau\mathbf{v}_i\mathbf{a}^\perp + \mathbf{m}^\top\mathbf{u}_i\mathbf{a}^\perp + \mu_j\mathbf{a}^\perp.$$

  distributed identically to $\mathbf{m}^\top\widetilde{\mathbf{K}}_i + \tau\widetilde{\mathbf{L}}_i + \mu_j\mathbf{a}^\perp$. This is because $\mu_j\mathbf{a}^\perp$ already hides $(\mathbf{m}^\top\mathbf{u}_i\mathbf{a}^\perp + \tau\mathbf{v}_i\mathbf{a}^\perp)$ for uniformly random $\mu_j \leftarrow \mathbb{Z}_p$.

The only way to successfully convince the verification to accept a signature Σ^* on $\mathbf{m}^*$, the adversary must correctly compute $\mathbf{m}^{*\top}(\mathbf{K} + \mathbf{u}_0\mathbf{a}^\perp) + \tau^*(\mathbf{L} + \mathbf{v}_0\mathbf{a}^\perp)$ and thus $\mathbf{m}^{*\top}\mathbf{u}_0$ and $\tau^*\mathbf{v}_0$. Observe that, $\{\mathbf{u}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$ (and thereby $\mathbf{u}_0$) also $\{\mathbf{v}_i\}_{i\in[1,n]\setminus \mathsf{CS}}$ (and thereby $\mathbf{v}_0$) are completely hidden to the adversary, $\mathbf{m}^{*\top}\mathbf{u}_0$ and τ^*v_0 are uniformly random from $\mathbb{Z}_p$ from the adversary's viewpoint. Therefore, $\mathbf{Adv}_4 = 1/p^2$.

□

Impossibility Result and updateable CRS model. Recent work by Bauer et al. [BFR24b] (AC'24) pointed out flaws in the security proofs of constructions of well known SPS-EQ in the literature. In a follow-up [BFR24a] (PKC'24), the same authors proved an impossibility result: SPS-EQ schemes cannot achieve security under standard assumptions using current techniques. That is achieving both unlinkability and unforgeability for SPS-EQ is impossible. The reason behind it is that if challenger wants to have unlinkability and provide the signatures to adversary, after getting response from the adversary the challenger has to make sure that the claim of the adversary is falsifiable [see Section 2.3]. To show that the unforgeability game is falsifiable, we meant that the challenger can determine if the forgery is coming from equivalence class of the previously queried messages or not. If the challenger can efficiently determine whether response from adversary for forgery is from equivalence class of the previously queried message will weaken the concept of unlinkablity property.

To solve this dilemma we enforce the updateable CRS [Definition 37] model for the threshold setting. In this scenario, we just need to run multiple extraction processes to get the trapdoors. The key point here is that one of those involved parties in the update phase should be following the protocol honestly. Making sure that at least one party updated the CRS honestly is important, because in real world setting any user can contribute, not just the signers. To make the unforgeability game falsifiable the challenger should extract the scaler μ to identify the class of forged message.

CHAPTER 4

TI-ulPCS

In this chapter we will be looking at the main content of this thesis. Having a secure non-interactive TSPS-EQ in Chapter 3, next we need is a (predicate only) predicate encryption in threshold setting. As discussed in Section 1.2.2, our predicate encryption uses a multi-party protocol.

4.1 THRESHOLD ISSUANCE PREDICATE ENCRYPTION

The (predicate-only) predicate encryption is used to have fine-grained control over the attribute and privacy over the attributes of the users. Also, it is important for attribute hiding and policy compliance. Let us look into what we briefly refer to as OT12 scheme [OT12]. We briefly describe the basics behind public-key generation, encryption and decryption. Assume we are in a bilinear group setting $\mathsf{pp} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, e, \mathsf{P}_1, \mathsf{P}_2)$ as before. We describe the predicate-only version already adapted to the asymmetric pairing case. This description can be found in [BSW24].

Public key and master secret. We first sample an invertible matrix X of dimension $N = 4n + 2$ (where n is the number of attributes) with elements in $\mathbb{Z}_p^*$ and consider the matrix $\psi \cdot X^{-1}$ for a random, non-zero field element ψ. For syntactical purposes only, the transpose is actually considered, i.e., we define $Y = \psi \cdot (X^{-1})^T$.

For notational purposes, we define basis vectors $\vec{a}_i = (\underbrace{1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2}}_{i-1}, \mathsf{P}_2, \underbrace{1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2}}_{N-i})$ and $\vec{a}_i^* = (\underbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}_{i-1}, \mathsf{P}_1, \underbrace{1_{\mathbb{G}_1}, \ldots, 1_{\mathbb{G}_1}}_{N-i})$. For a an element $c \in \mathbb{Z}_p$ the notation $c\vec{a}_i$ is shorthand for $(1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2}, \mathsf{P}_2^c, 1_{\mathbb{G}_2}, \ldots, 1_{\mathbb{G}_2})$. And for two vectors $\vec{b} = (B_1, \ldots, B_N) \in \mathbb{G}_i$ and $\vec{b'} = (B'_1, \ldots, B'_N) \in \mathbb{G}_i$, we write $\vec{b} \odot \vec{b'} := (B_1 B'_1, \ldots, B_N B'_N)$.

Finally, a pairing operation continued for vectors is defined: let $\vec{c} = (C_1, \ldots, C_N) \in \mathbb{G}_1^N$ and $\vec{c'} = (C'_1, \ldots, C'_N) \in \mathbb{G}_2^N$, then $\hat{e}(\vec{c}, \vec{c'}) := \prod_{i=1}^N e(C_i, C'_i)$.

Having these matrices $X = (x_{i,j})$ and $Y = (y_{i,j})$, we now define the public key and the master secret key: the public key $\mathbb{B} = (\vec{b}_1, \ldots, \vec{b}_N)$ consists of N vectors $\vec{b}_i := \bigodot_{j=1}^N x_{i,j}\vec{a}_j$. The master secret key $\mathbb{B}^* = (\vec{b}_1^*, \ldots, \vec{b}_N^*)$ consists of N vectors $\vec{b}_i^* := \bigodot_{j=1}^N y_{i,j}\vec{a}_j^*$.

We observe that there is the following relationship between $\mathbb{B}$ and $\mathbb{B}^*$ that follows from the definition of matrices X and Y:

$$\hat{e}(\vec{b}_i^*, \vec{b}_j) = e(\mathsf{P}_1, \mathsf{P}_2)^{x_{i,1}y_{j,1}+x_{i,2}y_{j,2}+\ldots+x_{i,N}y_{j,N}} = \begin{cases} e(\mathsf{P}_1, \mathsf{P}_2) =: \mathsf{P}_T, & \text{if } i = j, \\ 1_{\mathbb{G}_T}, & \text{if } i \neq j. \end{cases}$$

Key generation. For an attribute vector $\vec{v} \in \mathbb{Z}_p^n \setminus \{\vec{0}\}$, one first samples $\sigma \leftarrow \mathbb{Z}_p$ and $\vec{n} \leftarrow \mathbb{Z}_p^n$ at random. We then form the vector $\vec{z}^* = (1, \sigma\vec{v}, \underbrace{0, \ldots, 0}_{2n}, \vec{n}, 0)$. The key for attribute $\vec{v}$ is defined as $\vec{k}^* := \bigodot_{i=1}^N z_i^* \vec{b}_i^*$.

Encryption. For encryption, which is done relative to attribute vector $\vec{x} \in \mathbb{Z}_p^n \setminus \{\vec{0}\}$, one samples random values $\omega, \phi \leftarrow \mathbb{Z}_p$ and defines the helper vector $\vec{z} := (1, \omega\vec{x}, \underbrace{0, \ldots, 0}_{3n}, \phi)$. The ciphertext is defined as $\vec{c} := \bigodot_{i=1}^N z_i \vec{b}_i$.

Decryption. In the predicate-only case, a key $\vec{k}^*$ decrypts a ciphertext $\vec{c}$ iff $\hat{e}(\vec{k}^*, \vec{c}) = \mathsf{P}_T$.
We observe, for correctness, that due to the above relation between $\mathbb{B}^*$ and $\mathbb{B}$, the operation $\hat{e}$ in fact computes the inner product of the vectors $\vec{z}^*$ and $\vec{z}$ in the exponent of P_T. That is, $\hat{e}(\vec{k}^*, \vec{c}) = e(\mathsf{P}_1, \mathsf{P}_2)^{1+\omega\sigma\langle\vec{v},\vec{x}\rangle}$, and therefore $\hat{e}(\vec{k}^*, \vec{c}) = \mathsf{P}_T$ when the inner product $\langle\vec{v}, \vec{x}\rangle$ is zero.
We refer to [OT12] for the proof that this scheme is attribute hiding in the sense defined in Section 2.11.

So the main question here is that can we have a TI-PE by modifying the above construction from [OT12]. To answer this question, lets try to construct one scheme on our own. Given that there are n number of certificate authority (CA) in the system. We are assuming a trusted dealer to share the secret among the CAs. Also we can use well known DKG protocols pederson92 for this. So by using Shamir secret sharing Definition 30 to make n shares of the matrices X and Y, such that,

$$\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(n)} \leftarrow \mathsf{Share}(\mathbf{X}, \mathbb{Z}_p^{N\times N}, n, t),$$

and

$$\mathbf{Y}^{(1)}, \ldots, \mathbf{Y}^{(n)} \leftarrow \mathsf{Share}(\mathbf{Y}, \mathbb{Z}_p^{N\times N}, n, t).$$

Recombination will be done by Rec algorithm.

Partial Key generation. Having the shares of the matrices $\mathbf{X}$ and $\mathbf{Y}$. Lets say k^{th} CA has the public key $\mathbb{B}^{(k)}$ and master secret key $\mathbb{B}^{*(k)}$ of the form, $\mathbb{B}^{(k)} := \left(\vec{b}_1^{(k)}, \ldots, \vec{b}_N^{(k)}\right)$ where, $\vec{b}_i^{(k)} := \bigodot_{j=1}^N x_{i,j}^{(k)} \vec{a}_j$.
Similarly, $\mathbb{B}^{*(k)} = (\vec{b^*}_1^{(k)}, \ldots, \vec{b^*}_N^{(k)})$ consists of N vectors $\vec{b^*}_i^{(k)} := \bigodot_{j=1}^N y_{i,j}^{(k)} \vec{a}_j^*$.

For an attribute vector $\vec{v} \in \mathbb{Z}_p^n \setminus \{\vec{0}\}$, the k^{th} CA generate partial secret key by the following steps,

- $\sigma^{(k)} \leftarrow \mathbb{Z}_p$ and $\vec{n}^{(k)} \leftarrow \mathbb{Z}_p^n$.
- Forms the helper vector $\vec{Z^*}^{(k)} := (1, \sigma^{(k)} \cdot \vec{v}, 0, \ldots, 0, \vec{n}^{(k)}, 0)$.
- computes $\vec{k^*}^{(k)} := \bigodot_{i=1}^N z^*{}_i^{(k)} \vec{b^*}_i^{(k)}$.

Observe that

$$\vec{k^*}^{(k)} := \bigodot_{i=1}^N z^*{}_i^{(k)} \vec{b^*}_i^{(k)} := \left(\mathbb{G}_1^{y_{1,1}^{(k)}+\sigma^{(k)}(\vec{v}\cdot\vec{y}^{(k)})+(\vec{n}\cdot\vec{y}^{(k)})}, \ldots, \mathbb{G}_1^{y_{1,N}^{(k)}+\sigma^{(k)}(\vec{v}\cdot\vec{y}^{(k)})+(\vec{n}\cdot\vec{y}^{(k)})}\right). \tag{4.1}$$

In our case, we are distributing the power of the key generation in the system. So that the encryption and decryption remains as it has been described above. We can also achieve our goal of threshold issuance, because any user and the centralized authority generating the attributes for the receiver, but since the sender's power is being distributed we are relaxing the trust assumptions on the sender's side. One can observe that without having one part of the system the policy will not be fulfilled.

So, any party who wants to encrypt certain ciphertext relative to attribute vector $\vec{x} \in \mathbb{Z}_p^n \setminus \{\vec{0}\}$, will need to combine the secret keys first. After getting all sufficient number of shares from corresponding certificate authorities. For a set $S \subseteq [1, n]$ such that $|S| \geq t$ of correctly generated $\{\vec{k^*}^{(i)}\}_{i \in S}$ [4.1] it will do the following,

- using lagrange's coefficients λ_i,

$$\vec{k^*} := \left(\mathbb{G}_1^{\sum_{i \in S} \lambda_i y_{1,1}^{(i)} + \sum_{i \in S} \lambda_i \sigma^{(i)} (\vec{v} \cdot \vec{y}^{(i)}) + \sum_{i \in S} \lambda_i (\vec{n}^{(i)} \cdot \vec{y}^{(i)})}, \ldots, \mathbb{G}_1^{\sum_{i \in S} \lambda_i y_{1,N}^{(i)} + \sum_{i \in S} \lambda_i \sigma^{(i)} (\vec{v} \cdot \vec{y}^{(i)}) + \sum_{i \in S} \lambda_i (\vec{n}^{(i)} \cdot \vec{y}^{(i)})} \right)$$

Observe that the recombination is not trivia here, the term $\sum_{i \in S} \lambda_i \sigma^{(i)} (\vec{v} \cdot \vec{y}^{(i)})$ doesn't support homomorphism as it has a non-uniform term $\sigma^{(i)}$ for all $i \in S$ (similarly for $\vec{n}^{(i)}$). To remove these randomnesses from the exponent and reconstruct using Rec, each party needs to agree on an uniform randomness σ (also, $\vec{n}$) before the protocol begins. For this need of agreement in the protocol we need to employ an efficient MPC protocol.

4.1.1 Instantiation of MPC protocol

Lets assume that there are n number of parties participating in the key generation phase as before. In this instantiation of the MPC protocol, we need each party to contribute. Because if any party does not participates, others can collude their information and leak some information.

This protocol should satisfy some properties,

- **Consistency/Agreement.** Each honest party should have some common output. i.e. the combined key $\vec{k^*}$.
- **Termination/Liveness.** Every party obtains an output in some feasiable time, such that the output is not going to leak any information about individual shares.

Each party inputs their shares of secret key and the randomness i.e. party p_i will input $\vec{k^*}^{(i)}$, $\sigma^{(i)}$ and $\vec{n}^{(i)}$. After one round of communication the protocol will output desired secret key $\vec{k^*}$ and every party will agree for one randomness σ. We need this key generation going to be done in distributed manner. Observe that this not an efficient way to generate key for predicate encryption. Although (Predicate only) predicate encryption is only needed for ul-PCS for generic policies. Ul-PCS for separable policies and RBAC policies are still efficient.

4.2 THRESHOLD ISSUANCE-UNLINKABLE POLICY COMPLIANT SIGNATURE

As described in Section 1.2 we are going to give methods to achieve threshold-issuance ul-PCS for each policies. Lets start with TI-ulPCS for generic policies, then move on to more efficient schemes like generic policies and RBAC policies.

4.2.1 TI-uIPCS for generic policies.

We begin with explaining how the generic UL-PCS scheme of Figure 4.1 and its instantiation from OT12's inner product predicate encryption [OT12] has been decentralized. The master secret key, $\mathsf{msk}_{\mathsf{ABE}}$, of the predicate encryption scheme and its signing key $\mathsf{sk}_{\mathsf{sig}}^{A}$ are kept by the Certification Authority (CA). All users, however, receive the PRF seed k, a predicate encryption secret key sk_f, and a root signature key-pair $(\mathsf{sk}_{\mathsf{sig}}, \mathsf{vk}_{\mathsf{sig}})$. To enable distributed generation of these secret material, one can proceed as follows.

CA-side setup: As we have established the OT12 IP-PE scheme in Section 4.1, we can generate the keys in a distributed way. Each CA maintains its own signature key-pair. Registration of a client for attributes x:
Each CA generates a random seed k_i and a public key share vk_i. The CAs produce additional shared randomness in anticipation of generating the secret functional key. They compute a sum-sharing of $n+1$ random elements r_k. They execute a multiplication protocol to obtain a sharing of selected matrix elements: $r_1 Y_{i,2}, \ldots, r_1 Y_{i,n+1}$ (those are the generation indices for a scaled vector of attributes) and $r_j Y_{i,3n+j}$ for $1 < j \leq n+1$ (those are the indices where random exponents must be used).
In OT12, the functional key sk_f is a vector whose ith component is $\prod_{j=1}^{N} \mathsf{P}_1^{Y_{ij} z_j}$. We observe that every CA can compute an useful share sk_f^i by carrying out this computation as a function of attribute x and of the share of elements Y_{ij}, respectively $r_k Y_{ij}$ (for those indices where additional randomness is needed). The CA signs the triples (k_i, x), (k, sk_f^i), and (k, vk_i).

Aggregation step: Functional key shares are summed by component-wise product of the vectors sk_f^i. Incorporating the exponent gives the expression in the functional key definition definition, as everything has already been computed as a sum-sharing. Seed shares are summed as $k = \sum_i k_i$. Root key pairs are summed as well (e.g., taking a simple DL-based signature scheme into account).
Lastly, the pairs sk_f, (k, x), and $(k, \mathsf{vk}_{\mathsf{sig}})$ aggregated together are certifiable. In either language $\mathcal{L}_1$ and $\mathcal{L}_2$, instead of proving knowledge of a signature from the CA, one would have to prove that the aggregation is done properly according to signed shares by each CA. Though a conceptually achievable step, one can even push more computational effort to the registration phase, as discussed below.

Alternative solution (optional extra round): Alternatively to the above certification, one can incorporate an extra round of interaction in which the client commits to every aggregated pair, verifies their well-formedness with a NIZK, and receives a threshold signature on the commitments. In this instance, each pair is certifiable in the first and second NIZK languages by incorporating one extra commitment and a signature upon it. We utilize our TSPS-EQ proposed in Chapter 3 for further efficiency. In this case, each CA has its own SPS signature key-pair, and many issuers need to obtain a valid signature (with respect to the aggregated public key).
Furthermore, the simplifications are possible based on the adversary model. We may observe that the base key pair $(\mathsf{sk}_{\mathsf{sig}}, \mathsf{vk}_{\mathsf{sig}})$ for the party is never kept by the party in any process. Thus, in an honest-but-curious setting, we may simply have one server decide on this key-pair.

4.2.2 TI-uIPCS for separable policies

Similarly, we will disclose the generation of secret keys for the TI-uIPCS scheme with separable policies. In this scheme, the PE functionality is implemented in terms of a plain public key encryption (PKE) scheme with keys $(\mathsf{vk}_{\mathsf{PKE}}, \mathsf{sk}_{\mathsf{PKE}})$. Furthermore, signature keys are used to sign receiver and sender predicates. This scheme is far less complex than the generic scheme shown previously. Advance distributed key generation is possible, and each Certification Authority (CA) possesses its own signature key-pair. We now very briefly describe how the registration process of the user is actually carried out.

$\underline{\mathsf{Setup}(1^\lambda, F)}$:
$\mathsf{CRS}_{\mathsf{Rand}} \leftarrow \mathsf{NIZK}_{\mathcal{L}_1}.\mathsf{Setup}(1^n)$
$\mathsf{CRS}_{\mathsf{sig}} \leftarrow \mathsf{NIZK}_{\mathcal{L}_2}.\mathsf{Setup}(1^n)$
$(\mathsf{sk}^A_{\mathsf{sig}}, \mathsf{vk}^A_{\mathsf{sig}}) \leftarrow \mathsf{DS.Setup}(1^n)$
$(\mathsf{mpk}_{\mathsf{ABE}}, \mathsf{msk}_{\mathsf{ABE}}) \leftarrow \mathsf{ABE.Setup}(1^n)$
$\mathsf{mpk} := (T_{\mathsf{Rand}}, F, \mathsf{CRS}_{\mathsf{Rand}}, \mathsf{CRS}_{\mathsf{sig}}, \mathsf{vk}^A_{\mathsf{sig}}, \mathsf{mpk}_{\mathsf{ABE}})$
$\mathsf{msk} := (F, \mathsf{sk}^A_{\mathsf{sig}}, \mathsf{msk}_{\mathsf{ABE}})$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, x)}$:
Parse msk as defined above
$k \leftarrow \{0,1\}^n$
$(\mathsf{sk}_{\mathsf{sig}}, \mathsf{vk}_{\mathsf{sig}}) \leftarrow \mathsf{DS.Setup}(1^n)$
$\mathsf{sk}_{f_x} \leftarrow \mathsf{ABE.KeyGen}(\mathsf{msk}_{\mathsf{ABE}}, f_x)$
$\sigma^1_{\mathsf{sig}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^A_{\mathsf{sig}}, (k, x)), \sigma^2_{\mathsf{sig}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^A_{\mathsf{sig}}, (k, \mathsf{vk}_{\mathsf{sig}}))$
$\sigma^3_{\mathsf{sig}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}^A_{\mathsf{sig}}, (k, \mathsf{sk}_{f_x}))$
$\mathsf{usk} := (k, \mathsf{vk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}}, \sigma^1_{\mathsf{sig}}, \sigma^2_{\mathsf{sig}}, \sigma^3_{\mathsf{sig}}, x, \mathsf{sk}_{f_x})$
Return $(\mathsf{vk}_0, \mathsf{sk}_0) \leftarrow \mathsf{RandKey}(\mathsf{mpk}, (\mathsf{usk}, -1, \perp))$

$\underline{\mathsf{RandKey}(\mathsf{mpk}, \mathsf{sk})}$:
Parse mpk, usk as defined above and $\mathsf{sk} = (\mathsf{usk}, \mathsf{ctr}, \cdot)$
$\mathsf{ctr} := \mathsf{ctr} + 1$
If $\mathsf{ctr} \geq T_{\mathsf{Rand}}$: return $\perp$
$\mathcal{ID}_{\mathsf{ctr}} := \mathsf{PRF.Eval}(k, \mathsf{ctr})$
$(\mathsf{vk}^{\mathsf{ctr}}_{\mathsf{sig}}, \mathsf{sk}^{\mathsf{ctr}}_{\mathsf{sig}}) \leftarrow \mathsf{DS.Setup}(1^n)$
$\sigma_{\mathsf{ctr}} \leftarrow \mathsf{DS.Sign}(\mathsf{sk}_{\mathsf{sig}}, (\mathsf{vk}^{\mathsf{ctr}}_{\mathsf{sig}}, \mathcal{ID}_{\mathsf{ctr}}))$
$\mathsf{ct}_{\mathsf{ctr}} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}_{\mathsf{ABE}}, x)$
$\pi_{\mathsf{ctr}} \leftarrow \mathsf{NIZK}_{\mathcal{L}_1}.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{Rand}}, (T_{\mathsf{Rand}}, \mathcal{ID}_{\mathsf{ctr}}, \mathsf{vk}^{\mathsf{ctr}}_{\mathsf{sig}}, \mathsf{ct}_{\mathsf{ctr}}, \mathsf{vk}^A_{\mathsf{sig}}, \mathsf{mpk}_{\mathsf{ABE}}), (\mathsf{usk}, \sigma_{\mathsf{ctr}}))$
$\mathsf{vk}_{\mathsf{ctr}} := (\mathcal{ID}_{\mathsf{ctr}}, \mathsf{vk}^{\mathsf{ctr}}_{\mathsf{sig}}, \mathsf{ct}_{\mathsf{ctr}}, \pi_{\mathsf{ctr}})$
Return $(\mathsf{vk}_{\mathsf{ctr}}, \mathsf{sk}_{\mathsf{ctr}} := (\mathsf{usk}, \mathsf{ctr}, \mathsf{sk}^{\mathsf{ctr}}_{\mathsf{sig}}))$

$\underline{\mathsf{Sign}(\mathsf{mpk}, \mathsf{sk}, \mathsf{vk}_R, m)}$:
Parse $\mathsf{mpk}, \mathsf{sk} := (\mathsf{usk}, \mathsf{ctr}, \mathsf{sk}_{\mathsf{ctr}})$ and usk as above
If $\mathsf{VerifyPK}(\mathsf{mpk}, \mathsf{vk}_R) = 0$: return $\perp$
$\mathcal{ID}_S := \mathsf{PRF.Eval}(k, \mathsf{ctr})$
If $\mathsf{ABE.Dec}(\mathsf{sk}_{f_x}, \mathsf{ct}_R) = 0$: return $\perp$
$\pi_s \leftarrow \mathsf{NIZK}_{\mathcal{L}_2}.\mathsf{Prove}(\mathsf{CRS}_{\mathsf{sig}}, (\mathcal{ID}_S, \mathsf{ct}_R, \mathsf{vk}^A_{\mathsf{sig}}), \mathsf{sk})$
$\sigma \leftarrow \mathsf{DS.Sign}(\mathsf{sk}_{\mathsf{ctr}}, (m, \mathsf{vk}_R, \pi_s))$
Return (π_s, σ)

$\underline{\mathsf{Verify}(\mathsf{mpk}, \mathsf{vk}_S, \mathsf{vk}_R, m, \sigma)}$:
Parse mpk as defined above and $\sigma = (\pi, \sigma')$
If $\mathsf{VerifyPK}(\mathsf{mpk}, \mathsf{vk}_S) = 0$ or $\mathsf{VerifyPK}(\mathsf{mpk}, \mathsf{vk}_R) = 0$, return $\perp$
Return $(\mathsf{NIZK}_{\mathcal{L}_2}.\mathsf{Verify}(\mathsf{CRS}_{\mathsf{sig}}, (\mathsf{vk}_S, \mathsf{vk}_R), \pi) \ \wedge \ \mathsf{DS.Verify}(\mathsf{vk}_S, (m, \mathsf{vk}_R, \pi), \sigma'))$

Figure 4.1: The algorithms of the unlinkable PCS scheme for generic policies. [BSW24]

CA-side setup: Each CA receives a random PRF seed $k_i \leftarrow \mathbb{Z}_p^*$ and public key share vk_i.
Registration of a client for attributes x: Every CA signs the value (k_i, vk_i, m), where $m \in \{0, 1\}$ is a bit. If $m = 1$, the client also receives a signature on (k_i, sk_i). The client then uses standard aggregation to derive all the relevant values, namely $\mathsf{sk}_{\mathsf{sig}}$ of all sk_is and global PRF seed k of all k_is.
Certification is once more available either by a non-interactive zero-knowledge proof (NIZK), or in a further round of interaction, as described above.

4.2.3 TI-ulPCS Role-based (RBAC) policies

The process of signing on behalf of role-based TI-ulPCS is also as defined above. With particular care, however, given to values that would infringe on privacy—i.e., the accumulator witnesses (possibly exploitable to check against which attributes a given public key is signed) and the PRF seed values. To promote privacy in this case, the following need to occur:

CA Setup: Since accumulator witnesses are essentially signatures on roles with an associated accumulator value, we employ a threshold signature scheme for their production. Every Certification Authority (CA) possesses a share of the signature of a role i on an accumulator A, where the latter is routed by its signature verification key.

Client registration for Attribute set x: The CA each select a random PRF seed k and generates a public key share. The CAs give partial signature shares to each client, combines them, and calculates a full witness for his/her role. The client derives the whole PRF seed, the aggregate signature key pair $(\mathsf{vk}_{\mathsf{sig}}, \mathsf{sk}_{\mathsf{sig}})$, and collects all witnesses required.

Certification: This is possible either via a non-interactive zero-knowledge proof (NIZK) or otherwise via an additional interactive round, as detailed above.

CHAPTER 5

Conclusion & Future Directions

Here, we present Threshold Issuance Unlinkable Policy-Compliant Signatures (TI-ulPCS), an enhanced and more secure cryptographic building block that extends the existing unlinkable policy-compliant signature (ulPCS) schemes by integrating threshold cryptography. Our construction addresses a critical flaw in standard ulPCS constructions—the centralized issuance of credentials—that necessarily creates a single point of failure and violates the security and trust assumptions within decentralized environments. By distributing the issuing process over a steady set of issuers with threshold techniques, we eliminate any point of trusted authority, thereby increasing the resistance of the system to compromise, increasing fault tolerance, and placing it more in the spirit of the goals of privacy-preserving decentralized identity systems. In order to attain this construction, we utilize the non-iterative threshold structure-preserving signatures on equivalence class (NI-TSPS-EQ) specified in Chapter 3 such that secure threshold signing with the algebraic structure that supports efficient cryptographic computation is enabled. For enforcing policy, we utilize the threshold predicate encryption scheme described in Section 1.2.2, with respect to which decryption—and, as a consequence, signature verification—is feasible only upon the fulfillment of access policies by a set of valid issuers. In addition to this, we employ pseudorandom functions (PRFs) to provide unlinkability as well as effective key derivation in the protocol.
The full instantiation of the TI-ulPCS scheme is given in Chapter 4, where we demonstrate its application for three various kinds of policies. While our construction supports generic policies, we observe that such implementations may lead to inefficiencies in real-world scenarios due to the computational and structural cost involved. Therefore, in a more efficient and practical deployment, we recommend the use of separable policies or role-based access control (RBAC) policies that supplement the threshold structure to enable scalable efficient enforcement of the access rules without compromising security or unlinkability.

Future Work. One of the most important aspects for future research is to push the challenge of adding a secure Multi-Party Computation (MPC) protocol to the threshold predicate encryption system described in Chapter 4. The current protocol takes one round of interaction from all Certificate Authorities (CAs), and this need for full participation renders the system non-threshold tolerant to failures. Schemes based on other policies(separable and RBAC), however, are efficient and practicable. The construction of an efficient Threshold-Issuance Predicate Encryption scheme is a hard problem which so far has not been explored thoroughly in the literature. Thus it is an interesting and promising research direction. Also one can observe that we only described the key generation in distributed manner, generating cipher text in distributed manner can also be good problem statement to have on mind.

Bibliography

[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, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Berlin, Heidelberg, August 2010.

[ANPKT24] Masayuki Abe, Masaya Nanri, Octavio Perez-Kempner, and Mehdi Tibouchi. Interactive threshold mercurial signatures and applications. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 69–103. Springer, Singapore, December 2024.

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 517–550. Springer, Cham, August 2022.

[BDV20] Luis T A N Brandao, Michael Davidson, and Apostol Vassilev. NIST roadmap toward criteria for threshold schemes for cryptographic primitives, 2020-07-01 04:07:00 2020.

[BFR24a] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On proving equivalence class signatures secure from non-interactive assumptions. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 3–36. Springer, Cham, April 2024.

[BFR24b] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On security proofs of existing equivalence class signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 3–37. Springer, Singapore, December 2024.

[BFV19] Alex Biryukov, Daniel Feher, and Giuseppe Vitto. Privacy aspects and subliminal channels in zcash. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1813–1830, New York, NY, USA, 2019. Association for Computing Machinery.

[BMS03] Dan Boneh, Ilya Mironov, and Victor Shoup. A secure signature scheme from bilinear maps. In Marc Joye, editor, *Topics in Cryptology — CT-RSA 2003*, pages 98–110, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[BMW21] Christian Badertscher, Christian Matt, and Hendrik Waldner. Policy-compliant signatures. Cryptology ePrint Archive, Paper 2021/1234, 2021.

[Bol02] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Bon98] Dan Boneh. The decision diffie-hellman problem. In Joe P. Buhler, editor, *Algorithmic Number Theory*, pages 48–63, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.

[BSW24] Christian Badertscher, Mahdi Sedaghat, and Hendrik Waldner. Unlinkable policy-compliant signatures for compliant and decentralized anonymous payments. *PoPETs*, 2024(4):226–267, October 2024.

[But13] Vitalik Buterin. Ethereum white paper: A next generation smart contract & decentralized application platform. 2013.

[CBC21] Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. SoK: Auditability and accountability in distributed payment systems. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21International Conference on Applied Cryptography and Network Security, Part II*, volume 12727 of *LNCS*, pages 311–337. Springer, Cham, June 2021.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ECDSA with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1769–1787, New York, NY, USA, 2020. Association for Computing Machinery.

[CKLM12] Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. Malleable proof systems and applications. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 281–300. Springer, Berlin, Heidelberg, April 2012.

[CKM23] Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 678–709, Cham, 2023. Springer Nature Switzerland.

[CKP+23] Elizabeth C. Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 348–382. Springer, Singapore, December 2023.

[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, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 409–438. Springer, Cham, March 2022.

[CPZ24] Cas Cremers, Aleksi Peltonen, and Mang Zhao. An extended hierarchy of security notions for threshold signature schemes and automated analysis of protocols that use them. Cryptology ePrint Archive, Paper 2024/1920, 2024.

[Des90] Yvo G. Desmedt. Making conditionally secure cryptosystems unconditionally abuse-free in a general context. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 6–16, New York, NY, 1990. Springer New York.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 307–315, New York, NY, 1990. Springer New York.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DKLS18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 980–997, 2018.

[DKLS19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1051–1066, 2019.

[DZP+22] Yue Duan, Xin Zhao, Yu Pan, Shucheng Li, Minghao Li, Fengyuan Xu, and Mu Zhang. Towards automated safety vetting of smart contracts in decentralized applications. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 921–935, New York, NY, USA, 2022. Association for Computing Machinery.

[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.

[FHS14] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. Cryptology ePrint Archive, Paper 2014/944, 2014.

[For87] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). In Alfred Aho, editor, *19th ACM STOC*, pages 204–209. ACM Press, May 1987.

[FS07] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography – PKC 2007*, pages 181–200, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

[GGM16] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In Jens Grossklags and Bart Preneel, editors, *FC 2016*, volume 9603 of *LNCS*, pages 81–98. Springer, Berlin, Heidelberg, February 2016.

[GGM17] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. In Jens Grossklags and Bart Preneel, editors, *Financial Cryptography and Data Security*, pages 81–98, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security*, pages 156–174, Cham, 2016. Springer International Publishing.

[GJKR01] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold dss signatures. *Information and Computation*, 164(1):54–84, 2001.

[GK15] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. Cryptology ePrint Archive, Paper 2015/907, 2015.

[GKM+18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Cham, August 2018.

[GMR87] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen message attack**this research was supported by nsf grant mcs-80-06938, an ibm/mit faculty development award, and darpa contract n00014-85-k-0125.: Extended abstract. In David S. Johnson, Takao Nishizeki, Akihiro Nozaki, and Herbert S. Wilf, editors, *Discrete Algorithms and Complexity*, pages 287–310. Academic Press, 1987.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[Gol01] Oded Goldreich. *Computational Difficulty*, page 30–100. Cambridge University Press, 2001.

[Inf25] Information is Beautiful. World's biggest data breaches & hacks. https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/, 2025. Accessed: 2025-07-10.

[KBDD20] Christopher Kuner, Lee A Bygrave, Christopher Docksey, and Laura Drechsler. *The EU General Data Protection Regulation (GDPR): A Commentary*. Oxford University Press, 02 2020.

[KG21] Chelsea Komlo and Ian Goldberg. Frost: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography*, pages 34–65, Cham, 2021. Springer International Publishing.

[KMOS21] Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh when you wake up: Proactive threshold wallets with offline devices. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 608–625, 2021.

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Berlin, Heidelberg, August 2015.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian. Structure-preserving signatures on equivalence classes from standard assumptions. Cryptology ePrint Archive, Paper 2019/1120, 2019.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Berlin, Heidelberg, April 2008.

[KW15] Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 101–128. Springer, Berlin, Heidelberg, April 2015.

[KYMM18] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in Zcash. In *Proceedings of the 27th USENIX Conference on Security Symposium*, SEC'18, page 463–477, USA, 2018. USENIX Association.

[Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Cham, August 2017.

[LRY16] Benoît Libert, Somindu C. Ramanna, and Moti Yung. Functional Commitment Schemes: From Polynomial Commitments to Pairing-Based Accumulators from Simple Assumptions. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*,

volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[MBG+23] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 30–44. ACM Press, November 2023.

[MBS+25] Omid Mirzamohammadi, Jan Bobolz, Mahdi Sedaghat, Emad Heydari Beni, Aysajan Abidin, Dave Singelee, and Bart Preneel. Keyed-verification anonymous credentials with highly efficient partial disclosure. Cryptology ePrint Archive, Paper 2025/041, 2025.

[MMS+24] Aikaterini Mitrokotsa, Sayantan Mukherjee, Mahdi Sedaghat, Daniel Slamanig, and Jenit Tomy. Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 163–195. Springer, Cham, April 2024.

[MR01] Philip MacKenzie and Michael K. Reiter. Two-party generation of DSA signatures. In Joe Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 137–154, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Berlin, Heidelberg, December 2016.

[Nak09] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. May 2009.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003. Invited paper.

[NS23] Matthias Nadler and Fabian Schär. Tornado cash and blockchain privacy: A primer for economists and policymakers. *Federal Reserve Bank of St. Louis Review*, 105(2):122–136, 2023.

[OT12] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Berlin, Heidelberg, April 2012.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Berlin, Heidelberg, August 1992.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. On the power of memory in the design of collision resistant hash functions. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, pages 105–121, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[Rab79] Michael O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. MIT Laboratory for Computer Science, 1979. URL: `http://ncstrl.mit.edu/Dienst/UI/2.0/Describe/ncstrl.mit_lcs/MIT/LCS/TR-212`. Note: Technical Report 212.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979.

[Sha12] Stuart S. Shapiro. The state and evolution of privacy by design. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 1053, New York, NY, USA, 2012. Association for Computing Machinery.

[vdKPJ20] Tim van de Kamp, Andreas Peter, and Willem Jonker. A multi-authority approach to various predicate encryption types. *DCC*, 88(2):363–390, 2020.

[Wik25] Wikipedia contributors. Tornado cash. https://en.wikipedia.org/wiki/Tornado_Cash, 2025. Accessed: 2025-07-10.

CHAPTER A
Appendix

Lemma 2 (Core Lemma). *Let the game* $\mathbf{G}^{\mathsf{Core}}_{\mathcal{D}_k,\mathsf{ABSGen}}(\kappa)$ *be defined as Figure A.1. For any adversary* $\mathcal{A}$ *with the advantage of* $Adv^{\mathsf{Core}}_{\mathcal{D}_k,\mathsf{ABSGen},\mathcal{A}}(\kappa) := |\Pr[\mathbf{G}^{\mathsf{Core}}_{\mathcal{D}_k,\mathsf{ABSGen}}(\kappa)] - 1/2|$*, there exists an adversary* $\mathcal{B}$ *against the* $\mathcal{D}_k$-MDDH *assumption such that with the running time* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *it holds that*

$$Adv^{\mathsf{Core}}_{\mathcal{D}_k,\mathsf{ABSGen},\mathcal{A}}(\kappa) \leq 2qAdv^{\mathsf{MDDH}}_{\mathcal{D}_k,\mathbb{G}_1,\mathcal{B}}(\kappa) + q/p,$$

where q *is a bound on the number of queries requested by adversary* $\mathcal{A}$ *for oracle* $\mathcal{O}_b(\cdot)$*. Note that* $\mathcal{A}$ *can only query the other oracles only once.*

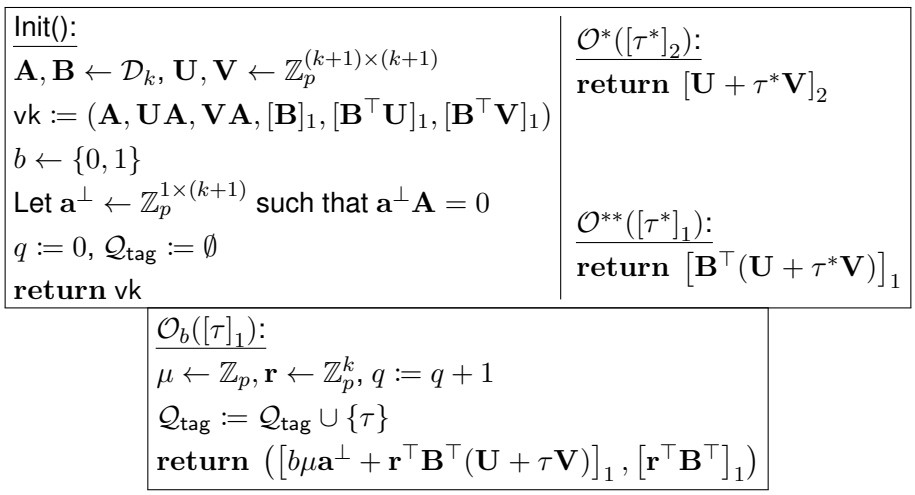


Figure A.1: Game defining the core lemma, $\mathbf{G}^{\mathsf{Core}}_{\mathcal{D}_k,\mathsf{ABSGen}}(\kappa)$. [MMS+24]

Proof. We proceed through a series of games from Game_0 to Game_q. Note that, Init outputs the same in all the games. In Game_i, the first i queries to the oracle $\mathcal{O}_b(\cdot)$ are responded with $([\mu\mathbf{a}^\perp + \mathbf{r}^\top\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]_1, [\mathbf{r}^\top\mathbf{B}^\top]_1)$ and the next $q - i$ queries are responded with $([\mathbf{r}^\top\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]_1, [\mathbf{r}^\top\mathbf{B}^\top]_1)$. The intermediate games Game_i and Game_{i+1} respond differently to the $i+1$-th query to $\mathcal{O}_b(\cdot)$. The Game_i responds with $([\mathbf{r}^\top\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]_1, [\mathbf{r}^\top\mathbf{B}^\top]_1)$ whereas Game_{i+1} responds with $([\mu\mathbf{a}^\perp + \mathbf{r}^\top\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]_1, [\mathbf{r}^\top\mathbf{B}^\top]_1)$. We compute the advantage of the adversary in differentiating the two games below. The advantage of the adversary in Game_i is denoted by $\mathbf{Adv}_i$ for $i = 0, \ldots, q$. On querying $\mathcal{O}_b(\cdot)$, Game_i picks $\mathbf{r} \leftarrow \mathbb{Z}_p^k$ and responds to $i+1$-th query with

$$([\mathbf{r}^\top\mathbf{B}^\top(\mathbf{U} + \tau\mathbf{V})]_1, [\mathbf{r}^\top\mathbf{B}^\top]_1),$$

We define a sub-game $\mathsf{Game}_{i.1}$ where $[\mathbf{Br}]_1$ is replaced with $[\mathbf{w}]_1$, $[\mathbf{w}]_1 \leftarrow \mathbb{G}_1^{k+1}$. From the MDDH assumption, a MDDH adversary cannot distinguish between the distributions $([\mathbf{B}]_1, [\mathbf{Br}]_1)$ and $([\mathbf{B}]_1, [\mathbf{w}]_1)$.

Thus,

$$([\mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{r}^\top \mathbf{B}^\top]_1) \approx_c ([\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{w}]_1).$$

All the other values can be perfectly simulated in the reduction by choosing $\mathbf{U}$ and $\mathbf{V}$ from the appropriate distributions. In the next sub-game $\mathsf{Game}_{i.2}$, we introduce the randomness $\mu \mathbf{a}^\perp$ to $[\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1$ and proceed to use an information-theoretic argument to bound the advantage in this experiment. As shown in [KW15], for every $\mathbf{A}, \mathbf{B} \leftarrow \mathcal{D}_k$, $\tau \neq \tau^*$, the following distributions are identically distributed

$$(\mathsf{vk}, [\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, \mathbf{U} + \tau^* \mathbf{V}) \text{ and } (\mathsf{vk}, [\mu \mathbf{a}^\perp + \mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, \mathbf{U} + \tau^* \mathbf{V}).$$

with probability $1 - 1/p$ over $\mathbf{w}$. The values $[\mathbf{B}^\top \mathbf{U}]_1$ and $[\mathbf{B}^\top \mathbf{V}]_1$ are part of the public values $\mathsf{vk} := (\mathbf{A}, \mathbf{UA}, \mathbf{VA}, [\mathbf{B}]_1, [\mathbf{B}^\top \mathbf{U}]_1, [\mathbf{B}^\top \mathbf{V}]_1)$ and anyone can compute $[\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1$ corresponding to a τ^*. Thus, for $\tau \neq \tau^*$, we have the two following identical distributions:

$$\begin{aligned} &(\mathsf{vk}, [\mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1) \text{ and} \\ &(\mathsf{vk}, [\mu \mathbf{a}^\perp + \mathbf{w}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1). \end{aligned} \tag{A.1}$$

From Equation (A.1), the subgames $\mathsf{Game}_{i.1}$ and $\mathsf{Game}_{i.2}$ are statistically close. We use the MDDH assumption again in the next sub-game $\mathsf{Game}_{i.3}$ and replace $[\mathbf{w}]_1$ with $[\mathbf{Br}]_1$. The resulting distribution is

$$(\mathsf{vk}, [\mu \mathbf{a}^\perp + \mathbf{r}^\top \mathbf{B}^\top (\mathbf{U} + \tau \mathbf{V})]_1, [\mathbf{U} + \tau^* \mathbf{V}]_2, [\mathbf{B}^\top (\mathbf{U} + \tau^* \mathbf{V})]_1),$$

which is same as Game_{i+1}. Thus, from the two MDDH instances as well as the information-theoretic argument,

$$|\mathbf{Adv}_i - \mathbf{Adv}_{i+1}| \leq 2 Adv^{\mathsf{MDDH}}_{\mathcal{D}_k, \mathbb{G}_1, \mathcal{B}}(\kappa) + 1/p. \qquad \square$$