NON-INTERACTIVE PROTOCOLS FOR BLIND DIGITAL SIGNATURES

by

Aayush Yadav
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Discipline

Committee:

_____ Dr. Foteini Baldimtsi, Dissertation Director

_____ Dr. Giuseppe Ateniese, Committee Member

_____ Dr. Krzysztof M. Gaj, Committee Member

_____ Dr. Samuel Dov Gordon, Committee Member

_____ Dr. Rishab Goyal, Committee Member

_____ Dr. Songqing Chen, Department Head

_____ Dr. Kenneth S. Ball, Dean

Date: _____ Summer Semester 2026
George Mason University
Fairfax, VA

Non-interactive Protocols for Blind Digital Signatures

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Aayush Yadav
Master of Science
Rutgers University, 2021
Bachelor of Engineering
University of Pune, 2017

Director: Dr. Foteini Baldimtsi, Professor
Department of Computer Science

Summer Semester 2026
George Mason University
Fairfax, VA

Acknowledgments

As I recall my journey through graduate school, I am filled with gratitude for all the people who contributed to my success.

My wonderful advisor, Foteini, was also my first cryptography teacher. Her excitement for the subject instantly drew me to the field. She has been pivotal in my progress as an academic. By being flexible and supportive, she allowed me to grow independently as a cryptographer and perhaps also shielded me from the many anxieties of the doctoral student life. The early guidance I received from her and Dov was formative to my research work. I am thankful to them for being patient as I fumbled my way through research during my first year. In hindsight, I was very fortunate to have the insight of two brilliant mentors.

A significant part of my time was spent at the University of Wisconsin. I am thankful to Rishab Goyal for hosting and mentoring me. I will always appreciate his words of encouragement before my first conference talk, and many more since.

I am also grateful to the other members of my dissertation committee, Giuseppe Ateniese and Kris Gaj, and the graduate program director, Hakan Aydin, for their invaluable feedback on my work.

During my time in the program, I had the opportunity to work with many excellent researchers including Daniel McVicker, Ojaswi Acharya, Jiaqi Cheng, Quan Nguyen, Abtin Afshar, Saikumar Yaduguri, Kostas Chalkias, Deepak Maram, Arnab Roy, Joy Wang, Lucjan Hanzlik, Amit Agarwal, Kushal Babel, Sourav Das, Babak Gilkalaye, Ari Juels, Benny Pinkas, Arup Mondal and Peter Rindal. In addition, I was also graciously hosted by Matthew Green, Stefano Tessaro and Fan Zhang. I am thankful to all of them as well as to the several other people I met along the way at conferences and talks for our edifying discussions and productive collaborations. How lucky I must be to have found such a vibrant research community!

I am indebted to Sunil Shende for setting me on the path I am on today. If not for his early encouragement, I might have missed out on a deeply fulfilling career.

Finally, I owe my deepest debt of gratitude to my family. To my parents for supporting my academic journey; and most importantly, to my partner, Erin, who has walked alongside me every step of the way, sharing in the highs and the lows. Without her, none of this would have been possible. In that sense, this work is as much hers as it is mine.

Funding acknowledgment. The works presented in this dissertation were supported by NSF awards #2143287 and #2247304.

Dedicated to Erin.

Table of Contents

List of Tables

List of Figures

Abstract

NON-INTERACTIVE PROTOCOLS FOR BLIND DIGITAL SIGNATURES
Aayush Yadav, PhD
George Mason University, 2026
Dissertation Director: Dr. Foteini Baldimtsi

Blind digital signatures allow a user to obtain a signature from a signing authority on any message, without revealing the message to said signer. Interestingly, in many modern applications, users are required to set these messages as random (unstructured) strings.

Non-interactive blind signatures (NIBS) are a variation of blind signatures that exploit this design choice to allow signers to asynchronously generate partial signatures such that only the intended user can extract a blinded signature on a random message. This bypasses the two-move barrier for traditional blind signatures and improves the efficiency of many known applications. In particular, non-interactivity simplifies protocol design, reduces latency, and enables asynchronous pre-computation for Internet-scale systems.

This work establishes rigorous security definitions that place NIBS on firm theoretical foundation and advances the state of the art by resolving efficiency gaps and addressing contemporary cryptographic challenges. Specifically, we propose efficient constructions from classical (pairing-based) assumptions as well as post-quantum secure (lattice-based) constructions to ensure long-term cryptographic viability. We show how issuance of signatures can be batched with minimal communication overhead. We further extend the applicability of non-interactive protocols to low-trust environments by distributing the secret signing key with threshold NIBS.

While blind signatures address concerns of privacy, we also investigate the complementary problem of accountability. Drawing on our NIBS techniques, we suggest a non-interactive model for anonymous tokens with a private metadata bit. The non-interactive anonymous token scheme with a private metadata bit (NIAT) enforces user accountability by allowing signers to privately embed a trust signal in their blind signatures. This secret flag allows signers to tag suspicious users without alerting them to their detection. We further present an extension facilitating the identification of double-spenders, a capability uniquely enabled by the presence of user public keys in our non-interactive setting. Finally, we propose a lattice-based construction for (interactive) anonymous tokens with private metadata, addressing the pressing requirement for quantum-resistant deployments.

Chapter 1: Introduction

The counter-cultural revolution gave rise to a new intellectual movement led by a small group of Bay Area technologists arguing for privacy in the early days of the Internet. The *cypherpunks* were an ideological collective, who were concerned about the risk of mass surveillance over digital communication, and foretold the end of private lives of citizens. They were, however, not mere harbingers—they were activists who sought to wield cryptography as a tool against what they saw as overreach by a neoliberal order.

It was in the background of these conversations, dominated largely by civil-libertarian thought around digital rights that another Northern California technologist introduced the idea that would lay the groundwork for the modern crypto-economic infrastructure [62, 63]. In a 1985 article titled 'Transaction systems to make big brother obsolete,' David Chaum opened with the following statement [63]:

> *"Computerization is robbing individuals of the ability to monitor and control the ways information about them is used."*

He then posited that "[t]he foundation [was] being laid for a dossier society, in which computers could be used to infer individuals' life-styles, habits, whereabouts, and associations from data collected in ordinary consumer transactions." This datafication of individual lives, as Chaum argues, has a second order 'chilling' effect wherein people modify their online behaviors so as to remain in good standing with the titular big brother.

Our evolving digital landscapes continue to echo Chaum's warnings. Indeed, as the world makes a collective heave towards authoritarianism, one need only look outside their window to observe the steady creep of Ring doorbells and Flock cameras pointed inward in a panopticon arrangement. If the goal in 1985 was to lay the foundation for a 'dossier society' then, today, the structure is being decorated with ornamental cupolas and friezes.

Chaum's prescient writings and much of the cypherpunk literature [108, 126] can therefore be understood as a sort of call for subversion of any such surveillance apparatus through the use of cryptography. Many commonplace privacy-enhancing technologies (PETs)—end-to-end encrypted messaging services, encrypted email protocols, virtual private network (VPN), Tor browser, whistle-blowing platforms, and crypto-currencies—are realizations of these early ideas.

Among the cryptographic tools that have emerged in the realm of PETs, *blind digital signatures* (hereafter simply 'blind signatures') [62] represent a particularly significant advancement. At their core lies a seemingly paradoxical question: how can users authenticate themselves without revealing their identity? By resolving this apparent contradiction of authentication without identification, blind signatures manage to capture a central aspect of digital rights, reinforcing the ability to maintain individual privacy in the face of pervasive oversight.

1.1 Blind Signatures

A blind signature scheme allows a user to obtain a signature on any message, under the key of some signing authority, without revealing the message. Typically, blind signatures are modeled as interactive protocols between a signer, S, holding a secret key, sk, and a user, U, holding the message m, and the signer's public verification key, pk. Only at the end of their interaction should the user be able to obtain a valid signature σ on m, without the signer having learned anything about m. The former condition conceptually captures the property of *unforgeability*, while the latter captures the property of *blindness* (hence the name).

To fully appreciate the usefulness of this primitive, let us consider the example of Chaum's anonymous electronic cash (e-cash) [62]. The idea is that in an e-cash system, a bank (acting as the signer) creates electronic coins through the use of blind signatures. Each coin is associated with a unique serial number selected by the user, and the bank's signature on it serves as proof of its validity. Crucially, the property of blindness guarantees that nobody (including the bank) can trace spent coins. A high-level outline of the e-cash protocol is illustrated in Figure 1.1.

In order to obtain an anonymous electronic coin (e-coin), a user engages in an *issuance* protocol with the bank possessing the key-pair $(\mathsf{sk}, \mathsf{pk})$. To initiate issuance, the user samples a string m and

sends the corresponding "blinded" value β to the bank along with a (fiat) deposit backing the e-coin. The bank issues the user a *pre-signature*, $\overline{\sigma}$ using its secret key, sk. When the user wishes to spend the e-coin, they initiate a *redemption* protocol with a vendor. More precisely, they "un-blind" the pre-signature in order to obtain a (blind) signature, σ, and send both m and σ to the vendor as the e-coin. The vendor simply verifies whether σ is a valid signature on m under the bank's verification key pk and is thus convinced that they have received a legitimate e-coin, with the bank acting as the guarantor.

In general, the issuance protocol may require multiple rounds of interaction between the user and the signer. So, in the e-cash example—depending on the specific blind signature protocol—the user and the bank may exchange more than just two messages. When the blind signature issuance protocols requires exactly two-moves (equivalently, one-round), it is said to be *round-optimal*.

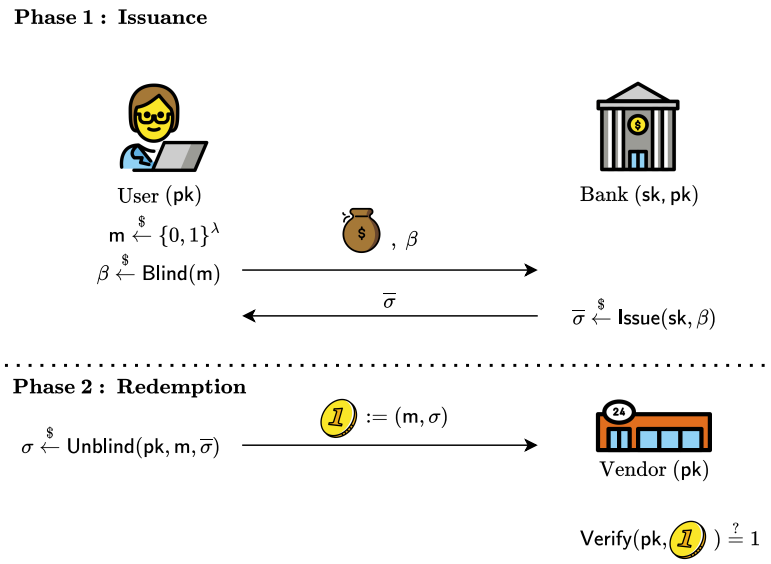

Figure 1.1: A simplified illustration of anonymous electronic cash. The user receives an electronic coin that is unlinkable to them. The vendor is assured that the coin is backed by fiat money in the bank.

Evolution of blind signatures. Since their initial proposal as the cryptographic foundation for e-cash, blind signatures have evolved into an independent research topic with vast academic literature. The early work [62, 64] explored the feasibility of constructing untraceable electronic payment systems. The subsequent decades saw rapid development as the security properties of blind signatures were refined and a formal framework was established. Pointcheval and Stern provided the first rigorous provable security analysis [149], although their scheme only guaranteed unforgeability for logarithmically many signatures. Juels, Luby, and Ostrovsky provided the first standard model scheme using generic one-way trapdoor permutations [113]. Their scheme only achieved sequential security, where sessions must execute non-overlappingly. Abe [3] gave a three-move blind signature protocol based on the discrete logarithm and decisional Diffie-Hellman assumptions, but later showed it to be flawed in a follow-up work [142]. Bellare, et al. [37] and Boldyreva [43] separately gave round-optimal constructions for blind signatures in the ROM. Both constructions rely on non-standard assumptions; the former on the RSA-based one-more-RSA-inversion problem, and the latter on hard problem in the (pairing-based) Gap Diffie-Hellman (GDH) group. Camensich, et al. [55] gave a multi-round standard model construction under the strong-RSA assumption. Fischlin [84] proposed a common reference string (CRS) based template for building blind signatures using only a commitment scheme, digital signatures, public key encryption and a non-interactive zero knowledge (NIZK) arguments. Looking ahead, this generic idea has had a significant impact on the further development of lattice-based blind signatures. Additional round-optimal standard model schemes followed [87, 92, 93], though all relied on non-standard assumptions.

Recent work has revisited classical constructions with modern techniques. Kastner, et al. [115] proved security of both Abe's scheme and Schnorr's blind signature [157] in the algebraic group model (AGM) [89], providing renewed confidence in their practical viability despite the failure of earlier proofs [38, 90]. In another recent result, Tessaro and Zhu [163] gave a pairing-free three-move blind signature scheme that is provably secure in the AGM and ROM under the discrete logarithm assumption. This dependence on the AGM was removed by Chairattana-Apirom, et al. [58] but under non-standard, interactive assumptions. Using a similar set of techniques, Klooß, et al. [121, 122] gave a secure blind signature construction under DDH.

The growing interest in post-quantum secure cryptography has also led to several lines of work in lattice-based blind signature constructions [6, 41, 70, 112, 131]. We discuss these works further below.

Impossibility results and fundamental barriers. On the theoretical side, Lindell [128] studied the concurrent security of general two-party protocols and concluded that black-box security cannot be achieved for the case of unbounded concurrency, i.e., protocols running polynomial number of concurrent executions. This result naturally also extends to blind signature protocols with three or more rounds.[1] In the light of this result, much of the work that followed [84] was focused on proving concurrent security in a non-standard model, or were only sequential-secure [55]. However as Hazay, et al. [104] pointed out, Lindell's impossibility result only applies to simulation-based definitions of security, and that it can indeed be bypassed if game-based definitions are used instead.

Beyond concurrency, deeper obstacles lie in the hardness assumptions themselves. In separate works, Brown [51] and Bresson, et al. [49] showed via a meta-reduction that the 'one-more' variants of the RSA and discrete-log problems can not be reduced to their respective plain versions. This is an important result, as the 'one-more' characterization of hardness assumptions is commonly used in security reductions for the one-more unforgeability of blind signatures [6, 61, 149, 150]. Fischlin and Schröder [85] generalized this further, showing that in fact no black-box reduction to a non-interactive cryptographic assumption can exist for three-move blind signatures (with signature derivation checks). This suggests that interactive hardness assumptions such as the one-more variants are inherently weaker than their non-interactive counterparts in the blind signature setting. As Fischlin and Schröder point out, this is inherent given the competing goals of unforgeability and blindness which they leverage in the meta-reduction.

Pass [145] further ruled out black-box reductions for *unique* blind signature schemes from hardness assumptions with polynomially-many rounds of interaction. Baldimtsi and Lysyanskaya [29] extended Pass' result for the Schnorr family of blind signatures [157]. Specifically, they showed that

[1] Protocols with one and two rounds are inherently concurrent-secure.

Schnorr blind signatures can not be proven secure in the ROM (via replay attacks) under any hardness assumption. A complementary result by Kastner, et al. [115] shows that the same also holds in the AGM for an unrestricted random oracle.

Finally, consider the most fundamental obstacle. Katz, et al. [117] demonstrated that black-box blind signature constructions from one-way functions are impossible in a strong sense. This applies even for single-bit messages and under semi-honest security. These impossibility results collectively establish that blind signatures require either structured assumptions or non-black-box techniques, fundamentally constraining what can be achieved.

Modern PETs applications. Contemporaneously, specialized applications for blind signatures emerged, demonstrating their utility for a broad range of privacy and accountability requirements. Electronic voting systems [56] leverage blind signatures to enable verifiable voting with voter privacy, while anonymous credentials [28, 63] and direct anonymous attestation [50] provide authenticated identity systems without linkability. In the crypto-currency domain, tumblers [105, 106] employed blind signatures to break transaction linkability and enhance privacy.

Perhaps most significantly, blind signatures are being used to construct *anonymous tokens*, achieving widespread industry adoption. Google's Private State Tokens [96], Cloudflare's Privacy Pass [135], Apple's iCloud Private Relay [16], and Apple's Private Cloud Compute [17] all rely on blind signature mechanisms to provide privacy-preserving services at scale. This push for deployment has driven standardization efforts, with the IETF [109] developing formal specifications supported by major industry stakeholders including Google, Apple, Cloudflare, and Fastly. This convergence underscores the growth of blind signatures from conceptual objects to a practical privacy tool.

1.2 Non-interactive Blind Signatures

Blind signatures are inherently interactive—the issuance necessitates that the user send the blinded message; but this limitation applies only when the message to be signed is set by the user. However, in many applications, such as e-cash and anonymous tokens, this is not the case, as the coins (or

tokens) are simply signatures on random identifiers. This critical insight led to the proposition of *non-interactive* protocols for blind signatures (NIBS) on random messages [101].

In a non-interactive blind signature scheme, a signer with key-pair $(\mathsf{sk}, \mathsf{pk})$ issues a pre-signature $\bar{\sigma}$ under a user public key pk_U and some public *context* c. The context is a nonce computed from known public information, such as $\mathsf{H}(\mathsf{pk}_U \| \texttt{timestamp})$. Given the pre-signature, the user can derive both a message m and a blind signature σ such that (m, σ) verifies under pk. The message m itself is an unstructured pseudo-random value, derived by the user from its secret key sk_U and the context c, for which it possesses the pre-signature.

Non-interactive blind signatures enable minimal communication overhead consisting of a single message from signer to user. This efficiency gain allows for new deployment scenarios. In anonymous token architectures such as Privacy Pass [57, 69, 146], the signer can pre-compute the pre-signatures for the user in advance, thus removing the latency due to online computation. Beyond these applications, NIBS extend to lottery systems and crypto-currency airdrops, where the ability to issue signatures non-interactively simplifies the distribution of cryptographic credentials at scale. This observation leads to our core thesis.

Thesis

Non-interactive blind signatures are an appealing alternative when there is no a priori structure on the message. This insight fundamentally simplifies protocol design and reduces latency by enabling asynchronous pre-computation in large-scale systems.

This dissertation contributes to the theory and practice of non-interactive blind signatures by establishing rigorous formal foundations for their design and security. We advance the state of the art across two axes. First, we develop constructions that minimize computational overhead and reduce the size of pre-signatures, making non-interactive blind signatures practical for large-scale deployment. Second, we introduce novel extensions that enhance the applicability of NIBS across diverse use cases.

A formal security framework. The NIBS security model proposed in [101] is insufficient and fails to provide robust security guarantees for real-world applications. In Chapter 3 based on our work in [24], we address these limitations by revisiting the existing framework, identifying its weaknesses, and proposing a strengthened formal model that provides robust privacy guarantees. This enhanced framework places non-interactive blind signatures on a secure theoretical footing suitable for deployment.

Within this new framework, we provide two NIBS constructions that satisfy the stronger security model. The first employs general-purpose NIZK proofs, while the second introduces a generic paradigm for constructing NIBS from any circuit-private leveled homomorphic encryption scheme. Both approaches achieve provable security and offer distinct trade-offs in terms of efficiency and implementation complexity.

Post-quantum security and batch issuance. The design of practically efficient NIBS resistant to quantum adversaries remains a compelling and timely challenge [140]. Recent advances in lattice-based blind signature constructions [6, 41, 70, 112, 131] have opened new avenues for building plausibly post-quantum secure blind signatures. However, progress on NIBS specifically has been limited. A recent construction by Zhang, et al. [169] adapted the generic VRF-based approach of [101] to lattices in the random oracle model, though subsequent work [91] identified security flaws and proposed corrections under the randomized one-more ISIS (rOM-ISIS) [6, 24]. Beyond post-quantum considerations, many real-world applications require users to obtain multiple blind signatures in a single interaction. For example, Privacy Pass users receive batches of 30–100 tokens per session, making batch efficiency critical. A natural optimization is to design protocols where the total communication cost remains independent of the batch size, reducing overhead as users request more signatures. The interactive setting has seen limited exploration of this optimization, with Hanzlik, Loss, and Wagner [102] providing one of the few systematic treatments.[2] However, their construction achieves only amortized cost benefits rather than true independence from batch size.

[2] Although the 'compact e-cash literature' [21, 22, 54] also considers this problem, to the best of my knowledge, it does not similarly develop a formal security framework.

In Chapter 4 based on our work in [24, 25], we formalize the notion of non-interactive *batched* blind signatures (NIBBS), requiring that the cost from signer to user remain independent of batch size. We provide a practically efficient NIBBS from lattice-based assumptions, addressing both the post-quantum challenge and the batch efficiency problem. Specifically, our scheme is provably secure under the hardness the randomized one-more ISIS (rOM-ISIS) [24] or the ISIS_f assumptions [48]. Our construction combines the low-depth Crypto Dark Matter PRF [45] with succinct lattice-based proofs for rank-1 constraint systems [42]. The resulting scheme has an estimated signature size of around 300 KB and communication under 1 KB.

Threshold issuance. Considering that blind signatures are often used as conveyors of a signers' trust, we must also evaluate our assumptions about trustworthiness of the signers themselves. In order to mitigate such risk's such as the signer being compromised, a few works have proposed distributing the task of signing across a threshold of signers, each possessing a share of the full secret signing key [67, 120, 124, 166]. This yields a threshold signature [71, 72] analogue for blind signature wherein, as before, the final message and signature remain untraceable to any user for any set of (colluding) signers.

While [120, 166] are insecure due to the attack of [158]. The other two works give either two- [124] or three-round [67] protocols. In Chapter 5 based on our work in [27], we address the problem of threshold issuance in the non-interactive setting. We introduce non-interactive *threshold* blind signatures (NITBS), where a user gathers partial pre-signatures from a threshold of signers and locally combines them into a valid blind signature. This approach eliminates the multi-round interaction inherent in prior work while maintaining blindness and unforgeability guarantees.

We provide a formal treatment of NITBS, defining the security notions of blindness and one-more unforgeability. Our concrete construction adapts the Pointcheval-Sanders signature scheme [148] and achieves provable security in the AGM. Micro-benchmarking demonstrates that our scheme offers the smallest pre-signature and signature sizes and the fastest issuance among existing NIBS schemes making it practical for deployment.

1.3 (Non-interactive) Anonymous Tokens with a Private Metadata Bit

The Internet today is fraught with web-crawlers or bots. In fact, they account for nearly half the web traffic, generating request volumes that are easily measured in the order of billions of requests per day [8, 9]. Although many of these bots are benign, a substantial and growing fraction are malicious scrapers, fraud bots, etc., that put significant strain on web servers.

A common mitigation against bots is to use challenge-response Turing tests such as CAPTCHAs. However, the detection-systems used to identify bots often also end up flagging legitimate users. These systems largely target privacy-conscious users accessing the Internet behind VPN, Tor, shared proxies or even privacy-enhancing browser extensions [118]. A consequence of this is that users are being disincentivized from browsing the web anonymously as they end up having to solve CAPTCHAs repeatedly to access basic web resources.

The major design objectives of anonymous web browsing address this state of affairs by (i) establishing some notion of 'trust' in honest users; (ii) conveying said trust to various web servers; and (iii) doing so without revealing specific identifiable user information.

A natural approach to the trust problem in anonymous web browsing is to provide users with single-use, privacy-preserving tokens for solving a CAPTCHA that can be used to validate them as human users. The user can later present a token along with their resource request in lieu of a CAPTCHA solution and access the web seamlessly. This is the core premise of anonymous tokens [68].

An anonymous token system involves three parties: a user, a signer (or an signer), and a redemption server (acting as a verifier). The user engages in an issuance protocol with the signer who first verifies the trustworthiness of the user and then issues them a token. Later, the user can present the token to a redeemer, who simply verifies its authenticity to grant the user access to a service or resource, and additionally checks that the same token was not redeemed before (double-spending *detection*). The basic properties satisfied by an anonymous token system are unforgeability, which ensures that a user cannot issue tokens on its own, and unlinkability (or anonymity) which prevents signers and redeemers to link issued tokens to any token later redeemed.

Beyond providing controlled access to content delivery networks (CDNs) without CAPTCHA solving [68], anonymous tokens have many other practical applications including, private web-browsing [16], private contact tracing [161], fraud detection [79], private click measurement [167] and privacy-perserving blockchain transactions [5, 105] to name a few.

The literature contains numerous protocol proposals, each realizing different tradeoffs [39, 60, 68, 123, 161]. An essential distinction among existing works is on whether they support the embedding of public [161] or private metadata [39, 60, 123]. Anonymous tokens with public metadata support applications that require public labeling and can facilitate the embedding of geographical tags or expiration dates, while anonymous tokens with private metadata bit (ATPM) can be used to privately convey trust signals. The most common instantiation of private metadata involves a single hidden bit. The signer encodes this bit within the token in such a way that the user cannot discern its value, yet the bit can be extracted by the signer upon redemption. This seemingly modest extension proves remarkably powerful in practice by enabling the signer to confidentially flag certain tokens without the user's awareness. When adversaries remain oblivious to whether they have been detected, the task of developing sophisticated tools in order to avoid being detected gets significantly more challenging.

Non-interactivity. In Chapter 6 based on our work in [26], we present the first *non-interactive anonymous token* (NIAT) scheme with a private metadata bit. Our design builds on structure-preserving signatures on equivalence classes (SPS-EQ) [88] and recent advances in non-interactive blind signatures [24, 101]. Our main observation is that at the core of many of the proposed protocols is a blind signature scheme on a randomly selected message [60, 68, 123, 161], thus we can take our techniques from NIBS and extend them with a private metadata bit.

The benefits of non-interactivity in the anonymous token setting are substantial. By eliminating the need for online interaction between user and signer during token issuance, NIAT enables the signer to pre-compute and distribute tokens asynchronously. This drastically reduces the system latency, allowing tokens to be pre-computed at scale without constraining user demand during peak load times. We provide a complete formal treatment of NIAT security propose an efficient construction under standard cryptographic hardness assumptions, and experimentally validate its practical

performance. We additionally present an alternative instantiation based on circuit-private leveled homomorphic encryption, following the generic template developed in Chapter 3.

Double-spend identification. A NIAT system by itself does not preclude a user from attempting to redeem the same token twice. More importantly, even if an signer (or group of signers) were able to *detect* such a double-spending attempt by keeping track of all redeemed tokens, unlinkability seems to suggest that it should be hard to identify the offending user. However, this is not the case, since unlinkability with respect to a fixed user is defined for distinct contexts, and it turns out a NIAT scheme that allows *identification* of a user attempting to redeem the same token more than once is perfectly within the bounds of the definition.

Notably, our NIAT scheme is uniquely positioned for identifying violators who double-spend since the system necessitates user public keys which is not the case for existing (interactive) anonymous tokens schemes. In Chapter 6, we also develop a concrete extension of our primary NIAT protocol that realize this double-spend identification capability, permitting signers to respond to such misuse while maintaining anonymity for honest users.

Post-quantum security. In Chapter 7 based on our work in [30], we present the first (interactive) ATPM scheme from hard lattice assumptions. Our design follows the Fischlin blind-signature paradigm [84] and enriches it with lattice-based linearly-homomorphic encryption to carry the hidden bit. We instantiate our scheme from Falcon-512 and the efficient LNP22 lattice NIZK proof system [132]. The resulting protocol, which we call Atlantis, requires 70 KB of user-signer communication and yields 129 KB tokens. Our solution also addresses an open problem posed by Amjad, Yeo, and Yung [15] on generically extending blind signatures to support private metadata bit.

Chapter 2: Essential Cryptographic Background

The work presented in this thesis rely upon various fundamental cryptographic primitives. We introduce these basic building blocks in this chapter to establish a common vocabulary and provide a reference point for the more intricate schemes presented in subsequent chapters. Beyond these common definitions and background, each subsequent chapter introduces technical preliminaries tailored to its particular focus as necessary. We begin by establishing some notation.

Notation. We use λ to denote the security parameter. We use $\leftarrow\!\!\$$ to denote the output of a randomized algorithm and $\leftarrow$ to denote output of a deterministic algorithm. We denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$ and the set of all non-negative integers up to n as $[0, n] \coloneqq \{0\} \cup [n]$. With slight abuse of notation, we also use $[x]_i$ to denote the i^{th} secret share of x, although the distinction will always be clear from context.

Following common convention, we use lowercase bold-face letters to denote vectors and uppercase bold-face letters for matrices. For a vector $\mathbf{x}$, x_i denotes its i^{th} element. For a vector $\mathbf{x}$, we write its ℓ_2 norm as $\|\mathbf{x}\|_2$, often dropping the subscript and writing it simply as $\|\mathbf{x}\|$. We write the ℓ_∞ norm of $\mathbf{x}$ as $\|\mathbf{x}\|_\infty$.

We write polynomials $u(X)$ in indeterminate X simply as u when it is clear from context. The vector of coefficients of a polynomial u is written as $\vec{u}$. The norms of a polynomial u is its norm $\|\vec{u}\|$ taken over the vector of its coefficients. In all that follows, $\mathbb{Z}$ is the set of integers, $\mathbb{F}$ is a field, and $\mathbb{Z}_p$ denotes the set of integers modulo p. For a power of two d, we denote by $\mathcal{R}$ the ring $\mathbb{Z}[X]/(X^d+1)$. We write the ring $\mathbb{Z}_p[X]/(X^d+1) \cong \mathcal{R}/p\mathcal{R}$ as $\mathcal{R}_{p,d}$. When the degree is omitted from the subscript, it is assumed to be the fixed protocol parameter d.

2.1 Pseudorandom Functions

A pseudorandom function (PRF) $\mathrm{F} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a keyed function,[1] that takes a λ-bit key as input, and on input $x \in \{0,1\}^\lambda$, it outputs a value $y = \mathrm{F}_K(x)$.
It must further satisfy the property of **pseduorandomness** which requires that for every *stateful* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[b = \mathcal{A}^{\mathcal{O}_{K,b}(\cdot)}(1^\lambda) \ : \ K \leftarrow\$ \{0,1\}^\lambda,\ b \leftarrow\$ \{0,1\}\right] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the oracle $\mathcal{O}_{K,b}$ is defined as $\mathrm{F}_K(\cdot)$ if $b = 0$; otherwise it is defined as a random function from $\{0,1\}^\lambda \to \{0,1\}^\lambda$.

2.2 Public Key Encryption

A public key encryption (PKE) scheme E over message space $\mathcal{M}$ consists of a tuple of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$. Given the security parameter $\lambda \in \mathbb{N}$, the key generation algorithm outputs a secret decryption key $\mathsf{sk} \in \mathcal{SK}$ and a public encryption key $\mathsf{pk} \in \mathcal{PK}$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \to \psi$. Given a public encryption key $\mathsf{pk} \in \mathcal{PK}$ and a message $\mathsf{m} \in \mathcal{M}$, the probabilistic encryption algorithm outputs a ciphertext $\psi \in \Psi$.
- $\mathsf{Dec}(\mathsf{sk}, \psi) \to \mathsf{m}$. Given a secret decryption key $\mathsf{sk}\mathcal{SK}$ and a ciphertext $\psi \in \Psi$, the deterministic decryption algorithm outputs a message $\mathsf{m} \in \mathcal{M}$.

A PKE scheme must satisfy the following properties:

[1] For simplicity, we fix the key space, input space, and output space to be λ-bit strings, but this is not essential to the definition.

(i) **Correctness.** For every $\lambda \in \mathbb{N}$ and $\mathsf{m} \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \psi) = \mathsf{m} \ : \begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda) \\ \psi \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathsf{m}) \end{array}\right].$$

(ii) **Indistinguishability under chosen-plaintext attack (IND-CPA).** For every *stateful* PPT adversary $\mathcal{A}$ there exists a negligible functions $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[\mathcal{A}(\psi) = b : \begin{array}{r} b \leftarrow\!\$\ \{0,1\} \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \psi \leftarrow\!\$\ \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b) \end{array}\right] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.3 Signature Schemes

A signature scheme Σ over message space $\mathcal{M}$ consists of a tuple of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$. Given the security parameter $\lambda \in \mathbb{N}$, the probabilistic key generation algorithm outputs a public signature verification key $\mathsf{pk} \in \mathcal{PK}$ and a secret signing key $\mathsf{sk} \in \mathcal{SK}$.
- $\mathsf{Sign}(\mathsf{sk}, \mathsf{m}) \to \sigma$. Given the signing key $\mathsf{sk} \in \mathcal{SK}$ and a message $\mathsf{m} \in \mathcal{M}$, the signature generation algorithm returns a signature $\sigma \in \Sigma$.
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \to 1/0$. Given the public key $\mathsf{pk} \in \mathcal{PK}$, a message $\mathsf{m} \in \mathcal{M}$ and a signature σ, the deterministic signature verification algorithm outputs 1 (accept) or 0 (reject).

A signature scheme must satisfy the following properties:

(i) **Correctness.** For every $\lambda \in \mathbb{N}$ and $\mathsf{m} \in \mathcal{M}$,

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

(ii) **Existential unforgeability under chosen message attacks.** For every PPT adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{pk},\mathsf{m}^*,\sigma^*)=1 \;:\; \begin{array}{r}(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\; \mathsf{KeyGen}(1^\lambda)\\ (\mathsf{m}^*,\sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk})\end{array}\right] \leqslant \mathsf{negl}(\lambda)\ .$$

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}$, on query m, outputs $\mathsf{Sign}(\mathsf{sk},\mathsf{m})$. We say that $\mathcal{A}$ is admissible if it never queries m^* to $\mathcal{O}_{\mathsf{Sign}}$.

2.4 Commitment Schemes

A commitment scheme C over message space $\mathcal{M}$ consists of a tuple of polynomial-time algorithms $(\mathsf{Setup},\mathsf{Commit},\mathsf{Verify})$ defined as follows:

- $\mathsf{Setup}(1^\lambda, 1^n) \to \mathsf{crs}$. Given the security parameter $\lambda \in \mathbb{N}$ and a message length $n \in \mathbb{N}$, the setup algorithm outputs a reference string crs.
- $\mathsf{Commit}(\mathsf{crs},\mathsf{m};\mathsf{op}) \to \mathsf{com}$. Given the reference string crs, a message $\mathsf{m} \in \mathcal{M}$, and a random opening $\mathsf{op} \in \{0,1\}^\lambda$, the commit algorithm outputs a commitment com.[2]
- $\mathsf{Verify}(\mathsf{crs},\mathsf{m},\mathsf{com},\mathsf{op}) \to 0/1$. Given the reference string crs, a message $\mathsf{m} \in \mathcal{M}$, a commitment com and a random opening $\mathsf{op} \in \{0,1\}^\lambda$, the deterministic verification algorithm outputs either 1 (accept) or 0 (reject).

[2] We assume for simplicity that op is always a λ-bit string. Note that this follows without any loss of generality.

A commitment scheme must satisfy the following properties:

(i) **Correctness.** For every $\lambda, n \in \mathbb{N}$ and $\mathsf{m} \in \{0,1\}^n$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs},\mathsf{m},\mathsf{com},\mathsf{op}) = 1 \ : \begin{array}{r} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n) \\ \mathsf{op} \leftarrow\!\!\$\ \{0,1\}^\lambda \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{crs},\mathsf{m};\mathsf{op}) \end{array}\right] = 1 \ .$$

(ii) **Binding.** For every adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda, n \in \mathbb{N}$,

$$\Pr\left[\bigwedge_{b\in\{0,1\}} \mathsf{Verify}(\mathsf{crs},\mathsf{m}_b,\mathsf{com},\mathsf{op}_b) = 1 \ : \begin{array}{r} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{com},\mathsf{m}_0,\mathsf{m}_1,\mathsf{op}_0,\mathsf{op}_1) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leqslant \mathsf{negl}(\lambda) \ .$$

(iii) **Hiding.** For every *stateful* adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} b = \mathcal{A}(\mathsf{com}) \\ \wedge\ \mathsf{m}_0,\mathsf{m}_1 \in \{0,1\}^n \end{array} \ : \begin{array}{r} 1^n \leftarrow \mathcal{A}(1^\lambda) \\ \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathsf{m}_0,\mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{crs}) \\ b \leftarrow\!\!\$\ \{0,1\}, \mathsf{op} \leftarrow\!\!\$\ \{0,1\}^\lambda \\ \mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{crs},\mathsf{m}_b;\mathsf{op}) \end{array}\right] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda) \ .$$

Statistical and computational adversaries. If the adversary in the binding (resp. hiding) experiment is computationally unbounded, then the commitment scheme is *statistically* binding (resp. hiding); otherwise if the adversary is a PPT machine, the commitment scheme is *computationally* binding (resp. hiding). It is a well known fact that a commitment scheme cannot simultaneously be statistically binding and hiding.

2.5 Trapdoor Permutations

A trapdoor permutation P is a tuple of polynomial time algorithms $(\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Inv})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$. Given the security parameter $\lambda \in \mathbb{N}$, the key generation algorithm outputs a secret key $\mathsf{sk} \in \mathcal{SK}$ and a public key $\mathsf{pk} \in \mathcal{PK}$.
- $\mathsf{Eval}(\mathsf{pk}, x) \to y$. Given a public key $\mathsf{pk} \in \mathcal{PK}$ and an input $x \in \mathcal{X}$, the evaluation algorithm outputs $y \in \mathcal{X}$.
- $\mathsf{Inv}(\mathsf{sk}, y) \to x$. Given a secret key $\mathsf{sk} \in \mathcal{SK}$ and input $y \in \mathcal{X}$, the inverse algorithm outputs $x \in \mathcal{X}$.

A trapdoor permutation must further satisfy the following properties:

(i) **Correctness.** For every $\lambda \in \mathbb{N}$ and every $x \in \mathcal{X}$,

$$\Pr\left[\mathsf{Inv}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, x)) = x \ : \ (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)\right] = 1 \ .$$

(ii) **Pre-image resistance.** For every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathcal{A}(y) = x \ : \begin{array}{r} (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda) \\ x \leftarrow\!\!\$\ \mathcal{X} \\ y \leftarrow \mathsf{Eval}(\mathsf{pk}, x) \end{array}\right] \leqslant \mathsf{negl}(\lambda) \ .$$

For brevity, we will write Eval as the function $\mathrm{P} : \mathcal{SK} \times \mathcal{X} \to \mathcal{X}$ and Inv as the function $\mathrm{P}^{-1} : \mathcal{SK} \times \mathcal{X} \to \mathcal{X}$.

2.6 Non-interactive Zero Knowledge Proof Systems

A non-interactive zero-knowledge (NIZK) proof system Π for relation $\mathcal{R}$ is a tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ defined as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$. Given the security parameter $\lambda \in \mathbb{N}$, the probabilistic setup algorithm outputs a common reference string crs.
- $\mathsf{Prove}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \to \pi$. Given the crs, an instance $\mathbb{x} \in \mathcal{L}_\mathcal{R}$, and a witness $\mathbb{w}$, the probabilistic prover algorithm outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, \mathbb{x}, \pi) \to 1/0$. Given the crs, an instance $\mathbb{x}$, and a proof π, the deterministic verification algorithm either 1 (accept) or 0 (reject).

A NIZK proof system must satisfy the following properties:

(i) **Completeness.** For every $\lambda \in \mathbb{N}$, any instance and witness pair $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$,

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathbb{x}, \pi) = 1 \ : \ \begin{array}{r} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ \pi \leftarrow\!\!\$\ \mathsf{Prove}(\mathsf{crs}, \mathbb{x}, \mathbb{w}) \end{array}\right] = 1.$$

(ii) **Soundness.** For every *stateful* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{crs}, \mathbb{x}, \pi) = 1 \\ \nexists \mathbb{w} \ : \ (\mathbb{x}, \mathbb{w}) \in \mathcal{R} \end{array} \ : \ \begin{array}{r} \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ (\mathbb{x}, \pi) \leftarrow \mathcal{A}(\mathsf{crs}) \end{array}\right] \leqslant \mathsf{negl}(\lambda)\ .$$

(iii) **Zero-knowledge.** There exists a simulator $\mathcal{S}$ such that for every *stateful* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{rr} & b \leftarrow\!\!\$\ \{0,1\} \\ & \mathsf{crs}_0 \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda),\ \mathsf{crs}_1 \leftarrow \mathcal{S}(1^\lambda) \\ \mathcal{A}\left(\pi_b\right) = b \ \wedge\ (\mathbb{x},\mathbb{w}) \in \mathcal{R} \quad : & (\mathbb{x},\mathbb{w}) \leftarrow \mathcal{A}(\mathsf{crs}_b) \\ & \pi_0 \leftarrow\!\!\$\ \mathsf{Prove}(\mathsf{crs}_0,\mathbb{x},\mathbb{w}) \\ & \pi_1 \leftarrow \mathcal{S}(\mathsf{crs}_1,\mathbb{x}) \end{array}\right] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Argument of knowledge. A NIZK proof system Π is an argument of knowledge (NIZKAoK) if it additionally satisfies the following property:

(iv) **Knowledge soundness.** There exists a PPT extractor $\mathcal{E}$ such that for every *stateful* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{rr} & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{Verify}(\mathsf{crs},\mathbb{x},\pi) = 1 & (\mathbb{x},\pi) \leftarrow \mathcal{A}(\mathsf{crs}) \\ \wedge\ (\mathbb{x},\mathbb{w}) \notin \mathcal{R} \quad : & \mathbb{w} \leftarrow \mathcal{E}(\mathbb{x},\pi) \end{array}\right] \leqslant \mathsf{negl}(\lambda)\,.$$

Online knowledge extraction. A knowledge extractor $\mathcal{E}$ is called 'online' (also, 'straight-line') if it does not rewind the adversary.

2.7 Leveled Homomorphic Encryption

Let C_d denote the class of boolean valued circuits of depth d. A leveled homomorphic encryption scheme HE with message space $\{0,1\}$ for circuit class $\{C_d\}_{d\in\mathbb{N}}$ is a tuple of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda, 1^d) \to (\mathsf{sk}, \mathsf{ek})$. Given the security parameter $\lambda \in \mathbb{N}$ and the maximum circuit depth $d \in \mathbb{N}$, the key generation algorithm outputs a secret key $\mathsf{sk} \in \mathcal{SK}$ and an evaluation key $\mathsf{ek} \in \mathcal{EK}$.

- $\mathsf{Enc}(\mathsf{sk}, \mathsf{m} \in \{0,1\}) \to \psi$. Given a secret key $\mathsf{sk} \in \mathcal{SK}$, message $\mathsf{m} \in \{0,1\}$, the probabilistic encryption algorithm outputs a ciphertext ψ.

- $\mathsf{Eval}(\mathsf{ek}, \mathsf{C} \in \mathcal{C}_d, \vec{\psi}) \to \psi'$ Given an evaluation key $\mathsf{ek} \in \mathcal{EK}$, a circuit $\mathsf{C} \in \mathcal{C}_d$, and a sequence of ciphertexts $\vec{\psi} = (\psi_1, \ldots, \psi_\ell)$ for some $\ell > 0$, the evaluation algorithm outputs a ciphertext $\widehat{\psi}$. Here ℓ denotes the input length of C.

- $\mathsf{Dec}(\mathsf{sk}, \psi) \to \mathsf{m}/\perp$ Given a secret key $\mathsf{sk} \in \mathcal{SK}$ and a ciphertext ψ, the deterministic decryption algorithm outputs $\mathsf{m} \in \{0,1\}$ or $\perp$.

An LHE scheme must satisfy the following properties:

(i) **Correctness.** For every $\lambda, d \in NN$, every $\mathsf{C} \in \mathcal{C}_d$ with input length $\ell = \mathsf{poly}(\lambda)$ and every sequence of messages $(\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_\ell) \in \{0,1\}^\ell$

$$\Pr\left[\mathsf{Dec}\left(\mathsf{sk}, \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, \{\psi_i\}_{i\in[\ell]}\right) = \mathsf{C}\left(\{\mathsf{m}_i\}_{i\in[\ell]}\right) \ : \ \begin{array}{r} (\mathsf{sk}, \mathsf{ek}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda, 1^d) \\ \forall i \in [\ell] \ : \ \psi_i \leftarrow\!\$\ \mathsf{Enc}(\mathsf{sk}, m_i) \end{array}\right] = 1\,.$$

(ii) **IND-CPA.** For every *stateful* PPT adversary $\mathcal{A}$ there exists a negligible functions $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$

$$\Pr\left[\mathcal{A}(\psi) = b : \begin{array}{r} b \leftarrow\!\$\ \{0,1\} \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \psi \leftarrow\!\$\ \mathsf{Enc}(\mathsf{sk}, \mathsf{m}_b) \end{array}\right] \leqslant \frac{1}{2} + \mathsf{negl}(\lambda).$$

Circuit privacy. Additionally, an HE scheme is said to be circuit private if there exists a PPT simulator $\mathcal{S}$ such that for every $d \in \mathbb{N}$, every circuit $\mathsf{C} \in \mathcal{C}_d$ with input length $\ell = \mathsf{poly}(\lambda)$, any sequence of message bits $\mathsf{m}_1, \ldots, \mathsf{m}_\ell \in \{0,1\}$, $(\mathsf{sk}, \mathsf{ek}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda, 1^d)$, and for each $i \in [\ell]$: $\psi_i \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{m}_i)$,

$$\{\mathsf{ek}, \mathsf{Eval}(\mathsf{ek}, \mathsf{C}, (\psi_1, \ldots, \psi_\ell)), \psi_1, \ldots, \psi_\ell\} \approx_c \{\mathsf{ek}, \varphi, \psi_1, \ldots, \psi_\ell\} \ ,$$

where $\varphi \leftarrow \mathcal{S}(\mathsf{ek}, \mathsf{C}(m_1, \ldots, m_\ell), \psi_1, \ldots, \psi_\ell)$.

Part I

Non-interactive Blind Signatures

Chapter 3: Non-interactive Blind Signatures

As we have already established, the original blind signature scheme by Chaum [62] requires one full round of interaction (Fig 1.1). The user sends a blinded message to the signer, who responds with a pre-signature on the blinded message; the user then locally unblind this pre-signature to obtain the final message and signature pair. Further, it is clear that blind signatures on user specified messages are impossible unless the user and signer both send at least one message. Thus, two-move (equivalently, one round) blind signatures are round-optimal, making them particularly interesting for real-world applications. Beyond optimal communication, unforgeability of round-optimal blind signatures naturally extends to the concurrent setting [129], meaning that security is maintained even when an adversary engages in multiple concurrent sessions with the signer. Given these observations, there has been a long line of research on round-optimal blind signatures under different models and assumptions [6, 36, 41, 43, 84, 86, 87, 93, 95, 116, 160].

Interestingly, in many applications, the message to be signed is selected randomly, and does not come from a specific distribution or have a specific structure [62, 68]. Based on this observation, Hanzlik [101] demonstrated that if the underlying message, a priori, has no structure, then one can design a non-interactive protocol for blind signature issuance such that a signer can directly issue a pre-signature to the user (possessing its own key-pair) without requiring the user to make a request. This idea is illustrated in Figure 3.1.

More generally, non-interactive blind signatures (NIBS) prove most valuable in settings where the choice of message is not central to the application. When users are content to work with unstructured random messages determined by the system, NIBS eliminate the communication overhead inherent in traditional blind signatures, yielding protocols with minimal latency. Moreover, when user public keys are known through established public-key infrastructure (PKI), the signer

This chapter is based on [24].

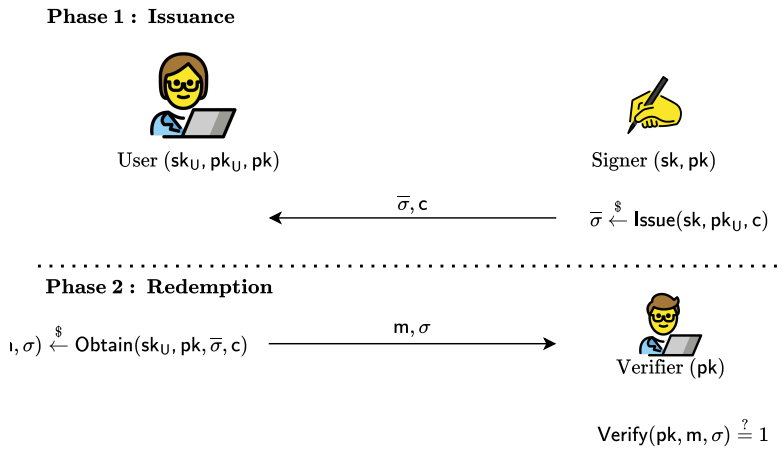

Figure 3.1: A simplified illustration of the non-interactive blind signature protocol. The user with key-pair $(\mathsf{sk}_\mathsf{U}, \mathsf{pk}_\mathsf{U})$ derives both the random message m as well as the blind signature σ given the pre-signature $\bar{\sigma}$ and a context c. Notice that the user gets to see the signer's randomness (c) but only it controls the final algorithm used to obtain the random message.

can generate and publish pre-signatures without any involvement from the user. This enables entirely new deployment scenarios. For instance, anonymous token systems following the Privacy Pass model [57, 68, 69, 146] can issue and hold tokens in advance of use, reducing latency at the critical moment of redemption. Similarly, lottery systems and other single-use credential distribution mechanisms become feasible at scale, freeing the design from the constraint that each credential must be personally negotiated.

Another example considered is crypto-currency airdrops, a mechanism used to bootstrap interest in new digital assets by distributing coins to registered users. Rather than requiring each user to engage in an issuance protocol, the project can compute and publish pre-signatures corresponding to user public keys already registered in the system. Users then privately derive the final coins by computing signatures from these pre-signatures, entirely offline and without further signer involvement.

Having established the practical utility of non-interactive blind signatures, the remainder of this chapter focuses on the fundamental concern of ensuring that NIBS rest on rigorous foundations.

Specifically, the existing security model for NIBS [101], while intuitive, contains gaps that fail to capture the full threat landscape of real-world deployments. We identify these limitations and propose a strengthened formal framework that provides robust security guarantees appropriate for practical applications. Building on this enhanced model, we present two concrete constructions of provably secure NIBS, each employing standard cryptographic primitives and representing distinct tradeoffs between efficiency and implementation complexity. Specifically, the first construction, based on general-purpose NIZKs has optimal pre-signatures but large final signatures, whereas the second construction which is based on circuit-private LHE has optimal final signatures but large pre-signatures.

3.1 The Formal NIBS Framework

In a NIBS scheme, a signer S issues a random pre-signature $\bar{\sigma}$ to a user U with public key pk_U, allowing U to extract a blind signature σ for a random message m using its secret key sk_U. Syntactically, a non-interactive blind signature scheme consists of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. Given the security parameter $\lambda \in \mathbb{N}$, the probabilistic setup algorithm generates the collection of public parameters pp, which is an implicit input to the other algorithms.
- $\mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$. Given the public parameters pp, the probabilistic (signer) key generation algorithm outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- $\mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \to (\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})$. Given the public parameters pp, the probabilistic user key generation algorithm outputs a $\mathsf{pk}_\mathsf{U} \in \mathcal{PK}_\mathsf{U}$ and a secret key $\mathsf{sk}_\mathsf{U} \in \mathcal{SK}_\mathsf{U}$.
- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}, \mathsf{c}) \to \bar{\sigma}$. Given a signing key share $\mathsf{sk} \in \mathcal{SK}$, a user public key $\mathsf{pk}_\mathsf{U} \in \mathcal{PK}_\mathsf{U}$ and context $\mathsf{c} \in \{0,1\}^\lambda$, the probabilistic issuance algorithm outputs a pre-signature share $\bar{\sigma} \in \overline{\Sigma}$.

- $\mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \mathsf{c}, \bar{\sigma}) \rightarrow (\mathsf{m}, \sigma)/(\perp, \perp)$. Given the user secret key $\mathsf{sk}_\mathsf{U} \in \mathcal{SK}_\mathsf{U}$, the public key $\mathsf{pk} \in \mathcal{PK}$, signing context $\mathsf{c} \in \{0,1\}^\lambda$, and a pre-signature $\bar{\sigma} \in \overline{\Sigma}$, the probabilistic obtain algorithm outputs a message and signature pair $(\mathsf{m}, \sigma) \in \mathcal{M} \times \Sigma$ or $(\perp, \perp)$.

- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \rightarrow 1/0$. Given the master public key $\mathsf{pk} \in \mathcal{PK}$, a message $\mathsf{m} \in \mathcal{M}$ and a signature $\sigma \in \Sigma$, the deterministic signature verification algorithm outputs 1 (accept) or 0 (reject).

Remark 3.1 (Context c as input or output.)**.** One can specify the NIBS syntax in two ways. As shown above, we use the 'context as input' convention followed in [27, 101] wherein the signing context c is given as input to the Issue algorithm. Under this approach, the signing context acts as a random nonce that ensures reusability of a user's public key for obtaining multiple message and signature pairs from a single signer. The alternate convention followed of 'context as output' of Issue is followed in [24, 25]. It allows one to define the reusability of a NIBS scheme as a formal property.

It is straightforward to show that a NIBS scheme in the 'context as output' syntax can be generically translated into one in the 'context as input' syntax. The idea is to generate the randomness for the former's $\mathsf{Issue}_\mathsf{O}$ algorithm by evaluating a PRF on the signing context c and the public key pk_U provided as input in the latter's $\mathsf{Issue}_\mathsf{I}$ algorithm. Concretely, $\mathsf{Issue}_\mathsf{I}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}, \mathsf{c})$ outputs the pre-signature as $(\bar{\sigma}', \mathsf{c}')$ where $(\bar{\sigma}', \mathsf{c}') \leftarrow\!\$\ \mathsf{Issue}_\mathsf{O}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}; \mathrm{F}_K(\mathsf{c}, \mathsf{pk}_\mathsf{U}))$.

We adopt the former 'context as input' convention for this part of the thesis, however for completeness, we provide a detailed proof of equivalence as well as a discussion on the benefits and drawbacks of both models in Appendix A.

A NIBS scheme must be correct as defined next.

Definition 3.1 (Correctness)**.** For every $\lambda \in \mathbb{N}$ and $\mathsf{c} \in \{0,1\}^\lambda$, a non-interactive blind signature scheme satisfies *correctness* if

$$\Pr\left[\mathsf{Verify}(\mathsf{pk},\mathsf{m},\sigma)=1 \;:\; \begin{array}{r} \mathsf{pp} \leftarrow\!\$\; \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk},\mathsf{sk}) \leftarrow\!\$\; \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}), \\ (\mathsf{pk}_\mathsf{U},\mathsf{sk}_\mathsf{U}) \leftarrow\!\$\; \mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \\ \bar{\sigma} \leftarrow\!\$\; \mathsf{Issue}(\mathsf{sk},\mathsf{pk}_\mathsf{U},\mathsf{c}) \\ (\mathsf{m},\sigma) \leftarrow\!\$\; \mathsf{Obtain}(\mathsf{sk}_\mathsf{U},\mathsf{pk},\mathsf{c},\bar{\sigma}) \end{array}\right] = 1\ .$$

Game $\mathsf{OM\text{-}Unf}_{\mathsf{NIBS},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\; \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{sk},\mathsf{pk}) \leftarrow\!\$\; \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp})$
3 : $\{(\mathsf{m}_i,\sigma_i)\}_{i=1}^{\ell+1} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Issue}}}(\mathsf{pp},\mathsf{pk})$
4 : **if** $\bigvee_{1\leqslant j<j'\leqslant \ell+1} (\mathsf{m}_j = \mathsf{m}_{j'})$ $\bigvee_{j\in[\ell+1]} \mathsf{Verify}\,(\mathsf{pk},\mathsf{m}_j,\sigma_j) \neq 1$ **then**
5 : **reject**
6 : **accept**

Oracle $\mathcal{O}_{\mathsf{Issue}}(\mathsf{pk}_\mathsf{U},\mathsf{c})$

1 : **if** ℓ is uninitialized **then**
2 : $\ell := 0$
3 : $\ell \leftarrow \ell + 1$
4 : **return** $\mathsf{Issue}\,(\mathsf{sk},\mathsf{pk}_\mathsf{U},\mathsf{c})$

Figure 3.2: The one-more unforgeability experiment for NIBS.

A NIBS scheme must also be unforgeable, which intuitively asks that a user should not be able to derive a valid (m,σ) pair on their own. The *one-more* unforgeability in the NIBS context is a natural extension of the same property for traditional blind signatures. As defined in [101], the adversary is granted access to a presignature oracle, and after obtaining ℓ presignatures for $(\mathsf{pk}_\mathsf{R},\mathsf{c})$ pairs of its choice, it must produce $(\ell+1)$ valid signatures to succeed.

Definition 3.2 (One-more unforgeability)**.** For all $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathsf{OM\text{-}Unf}}_{\mathsf{NIBS},\mathcal{A}} = \Pr\left[\mathsf{OM\text{-}Unf}_{\mathsf{NIBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 3.2. We say a non-interactive blind signature scheme, NIBS, is *one-more unforgeable* if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

Game U-Blind$_{\mathsf{NIBS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda),\ b \leftarrow\!\$\ \{0,1\}$
foreach $\omega \in \{0,1\}$ **do**
  $(\mathsf{pk}_{\mathsf{U}_\omega}, \mathsf{sk}_{\mathsf{U}_\omega}) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp})$
$\left(\mathsf{pk}, \{\mathsf{c}_\omega, \bar{\sigma}_\omega\}_{\omega\in\{0,1\}}\right) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{U}}_{\mathsf{Obtain}}}\left(\mathsf{pk}_{\mathsf{U}_0}, \mathsf{pk}_{\mathsf{U}_1}\right)$
foreach $\omega \in \{0,1\}$ **do**
  if $\mathsf{c}_\omega \in Q_{\mathsf{U}}$ **then**
    reject
  $(\mathsf{m}_\omega, \sigma_\omega) \leftarrow\!\$\ \mathsf{Obtain}\left(\begin{array}{c}\mathsf{sk}_{\mathsf{U}_\omega}, \mathsf{pk},\\ \mathsf{c}_\omega, \bar{\sigma}_\omega\end{array}\right)$
  if $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}_\omega, \sigma_\omega) \neq 1$ **then**
    reject
$b^* \leftarrow \mathcal{A}\left(\mathsf{m}_b, \sigma_b, \mathsf{m}_{1-b}, \sigma_{1-b}\right)$
if $b^* = b$ **then**
  accept
reject

Oracle $\mathcal{O}^{\mathsf{U}}_{\mathsf{Obtain}}\left(\omega, \mathsf{pk}, \mathsf{c}, \bar{\sigma}\right)$

if Q_{U} is uninitialized **then**
  $Q_{\mathsf{U}} \coloneqq \varnothing$
$Q_{\mathsf{U}} \coloneqq Q_{\mathsf{U}} \cup \{\mathsf{c}\}$
return $\mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}_\omega}, \mathsf{pk}, \mathsf{c}, \bar{\sigma}\right)$

Game C-Blind$_{\mathsf{NIBS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda),\ b \leftarrow\!\$\ \{0,1\}$
$(\mathsf{pk}_{\mathsf{U}}, \mathsf{sk}_{\mathsf{U}}) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp})$
$\left(\mathsf{pk}, \{\mathsf{c}_\omega, \bar{\sigma}_\omega\}_{\omega\in\{0,1\}}\right) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{C}}_{\mathsf{Obtain}}}\left(\mathsf{pk}_{\mathsf{U}}\right)$
foreach $\omega \in \{0,1\}$ **do**
  if $\mathsf{c}_\omega \in Q_{\mathsf{C}}$ **then**
    reject
  $(\mathsf{m}_\omega, \sigma_\omega) \leftarrow\!\$\ \mathsf{Obtain}\left(\begin{array}{c}\mathsf{sk}_{\mathsf{U}}, \mathsf{pk},\\ \mathsf{c}_\omega, \bar{\sigma}_\omega\end{array}\right)$
  if $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}_\omega, \sigma_\omega) \neq 1$ **then**
    reject
$b^* \leftarrow \mathcal{A}\left(\mathsf{m}_b, \sigma_b, \mathsf{m}_{1-b}, \sigma_{1-b}\right)$
if $b^* = b$ **then**
  accept
reject

Oracle $\mathcal{O}^{\mathsf{C}}_{\mathsf{Obtain}}\left(\mathsf{pk}, \mathsf{c}, \bar{\sigma}\right)$

if Q_{C} is uninitialized **then**
  $Q_{\mathsf{C}} \coloneqq \varnothing$
$Q_{\mathsf{C}} \coloneqq Q_{\mathsf{C}} \cup \{\mathsf{c}\}$
return $\mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, \mathsf{c}, \bar{\sigma}\right)$

Figure 3.3: The user (left) and context (right) blindness experiments for NIBS.

A NIBS scheme, like its interactive counterpart must also satisfy blindness, meaning that a signer should not be able to link a (m, σ) pair to any specific issuance, and by extension to any specific user. Hanzlik [101] proposed two distinct security experiments to capture blindness: (i) *user blindness*, ensuring that a malicious signer cannot determine which of two possible presignatures led to the final signature; and (ii) *context blindness*, which ensures that when a signer issues two presignatures to a specific recipient, they cannot link the final message-signature pairs to the respective presignatures. Both definitions are given as indistinguishability-based experiments.

Hanzlik's definition for blindness, however, proves insufficient for the privacy requirements of practical NIBS applications. The fundamental limitation is that it only guarantees unlinkability between any two pre-signatures and their derived message and signature pairs. Consequently, a scheme satisfying this definition need not maintain unlinkability across multiple pre-signatures issued to the same or different users, which can be disastrous is practice.

To fully appreciate this issue, consider the following simple scenario: an adversary issues two pre-signatures to a user with $\mathsf{pk}_{\mathsf{U}_0}$ and one pre-signature to a user with $\mathsf{pk}_{\mathsf{U}_1}$. Once an eavesdropping adversary learns two signatures from the same user, they will know with certainty that they mustbelong to the user with $\mathsf{pk}_{\mathsf{U}_0}$.[1]

A better blindness definition. To address the attacks described above, we propose a stronger blindness framework for NIBS. Our goal is to capture scenarios where an adversary learns correlations between pre-signature and context pairs, and their corresponding message and signature pairs. We formalize this by giving the adversary oracle access to the user's secret key through Obtain queries. For non-triviality, we restrict the adversary from querying the two challenge pre-signatures $\bar{\sigma}_0, \bar{\sigma}_1$.

Our new model functions as a sort of 'chosen-ciphertext' variant of blindness, analogous to IND-CCA security where the adversary gains query access to the secret key. These definitions ensure that blindness holds even when a malicious signer successfully breaks blindness on signatures from previous sessions. Any NIBS scheme secure under these stronger definitions prevents the attacks we described earlier. Intuitively, we allow the adversary to fully break blindness on all non-challenge

[1]This is not a problem in interactive blind signatures where blindness is defined as an indistinguishability game. It is the creation of user specific pre-signatures that complicates the definition of NIBS blindness.

pre-signatures, yet the challenge pre-signatures remain hidden. This robustness is what distinguishes our framework from weaker notions. Formally, we have the following definition—

Definition 3.3 (Blindness)**.** For all $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathsf{U\text{-}Blind}}_{\mathsf{NIBS},\mathcal{A}} = \Pr\left[\mathsf{U\text{-}Blind}_{\mathsf{NIBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of a *stateful* adversary $\mathcal{A}$ in the experiment defined in Figure 3.3 (left), and

$$\mathsf{Adv}^{\mathsf{C\text{-}Blind}}_{\mathsf{NIBS},\mathcal{A}} = \Pr\left[\mathsf{C\text{-}Blind}_{\mathsf{NIBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be its advantage in the experiment defined in Figure 3.3 (right). We say a non-interactive blind signature scheme, NIBS, satisfies *blindness* if for any PPT adversary $\mathcal{A}$ each of these advantages is at most $1/2 + \mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\cdot)$.

When is weak-blindness sufficient? In most applications of interest the weaker blindness definition given in [101] is insufficient; and one of our main assertions and contributions is that for typical applications of NIBS, where multiple pre-signatures are issued to multiple users, our stronger blindness definition *is* the correct definition. That said, one could potentially envision possible applications where the weaker definition may be satisfactory. For example, consider a one-signature-per-user system for anonymous airdrops or e-voting tokens, where only a single pre-signature will be generated for each user. In such applications, cardinality-based linking becomes impossible. Each derived signature carries no information about which user produced it, and the adversary gains nothing from counting or correlation. Essentially, in scenarios where credentials are issued in bulk and users claim them privately, the one-signature-per-user constraint ensures that the weaker user blindness notion from [101] suffices. The point is that weak blindness breaks precisely when multiple pre-signatures enable counting attacks, so if we eliminate that possibility, the weaker definition becomes adequate.

3.2 NIBS from General Purpose NIZK

In this section we present a Fischlin-like compiler for NIBS that builds from a standard signature scheme, an encryption scheme, a commitment scheme, general-purpose NIZK proofs and a pseudorandom function. Hanzlik [101] proposed a similar generic template using verifiable random functions (VRFs) and general-purpose dual mode witness indistinguishable proofs [99], but only proved security under the weaker blindness notions. Our construction is notably simpler, requiring only a PRF and general-purpose NIZK, while providing stronger blindness security guarantees.

Our starting point is the classical Fischlin's paradigm [84] for constructing traditional blind signatures. Fischlin's construction operates in the common reference string (CRS) model and employs a standard signature scheme, an encryption scheme, a commitment scheme, and general-purpose NIZK proofs. The CRS contains an encryption public key pk_{E}. Each signer generates its key pair $(\mathsf{sk}, \mathsf{pk})$ via the signature scheme's setup. The user computes a commitment $\mathsf{com} \leftarrow \mathsf{Commit}(\mathsf{m})$ and sends it to the signer. The signer signs the commitment, producing a pre-signature $\bar{\sigma}$, which it sends to the user. Upon verification, the user encrypts the pair $(\mathsf{com}, \bar{\sigma})$ under pk_{E} to obtain ψ, then generates a NIZK proof π demonstrating knowledge of a valid pre-signature and an opening of com to some message m. The user outputs the ciphertext ψ and proof π as the blind signature. Unforgeability follows from commitment binding and signature unforgeability, with straight-line extraction enabled by the trapdoor key sk_{E}. Blindness follows from encryption security, NIZK zero-knowledge, and commitment hiding. We now explain how to extend this paradigm to the non-interactive setting.

Fischlin's paradigm for non-interactive issuance. In the absence of interaction between signer and user, the signer can no longer rely on a commitment submitted by the user in a first message. Instead, the receiver's public key itself must serve this role, implicitly committing to a set of candidate messages. The challenge now is that this commitment must be succinct enough to represent an *exponential* number of possible messages, while the signer's response at the same time, must obliviously bind to exactly one of them, chosen at random. This oblivious binding to a single message from an exponentially large set is essential for one-more unforgeability as otherwise the receiver could easily cheat by later claiming that multiple distinct messages were signed during the protocol.

To instantiate this template, we must overcome two interconnected challenges; (i) we need a way to commit to an exponential number of messages efficiently; and (ii) we need to enable the signer to obliviously select one of those messages. Our approach is to set the receiver's public key pk_{U} as a commitment to a PRF key K, with K and its opening held as the corresponding secret key sk_{U}. Notice that with this construction we have implicitly defined the exponential message space as all possible outputs of the PRF. During the issuance of a pre-signature, the signer obliviously samples a message by evaluating the PRF on the context c, and signs both this message and pk_{U}. The receiver then derives the final signature on the message $\mathsf{m} \coloneqq \mathrm{F}_K(\mathsf{c})$ by constructing a NIZK proof of knowledge of a (pre-)signature on both pk_{U} and the context c corresponding to the obliviously selected message m.

Construction 3.1. Our construction relies on a pseudorandom function F, a statistically binding commitment scheme $\mathrm{C} = (\mathrm{C}.\mathsf{Setup}, \mathrm{C}.\mathsf{Commit}, \mathrm{C}.\mathsf{Verify})$, a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$, and a NIZKAoK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for the language $\mathcal{L}_{\Pi}^{\mathsf{NIBS}}$ as described.

Language $\mathcal{L}_{\Pi}^{\mathsf{NIBS}}$

Instance: Each instance $\mathbb{x}$ is interpreted as a verification key pk, CRS for a commitment scheme $\mathsf{crs}_{\mathrm{C}}$, and a message m.

Witness: Witness $\mathbb{w}$ consists of a signature $\bar{\sigma}$, context c, PRF key K, and commitment opening op.

Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\Sigma.\mathsf{Verify}(\mathsf{pk}, \mathrm{C}.\mathsf{Commit}(\mathsf{crs}_{\mathrm{C}}, K; \mathsf{op}) \| \mathsf{c}, \bar{\sigma}) = 1 \ \wedge \ \mathsf{m} = \mathrm{F}_K(\mathsf{c})$$

Concretely, the NIBS construction is defined by the following algorithms:

- $\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}$. The setup algorithm runs the setup algorithms for the NIZK proof system Π (for language $\mathcal{L}_{\Pi}^{\mathsf{NIBS}}$) and C schemes, i.e., it generates

$$\mathsf{crs}_{\Pi} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^{\lambda}) \text{ and } \mathsf{crs}_{\mathrm{C}} \leftarrow\!\!\$\ \mathrm{C}.\mathsf{Setup}(1^{\lambda})\ .$$

It outputs $\mathsf{pp} \coloneqq (\mathsf{crs}_{\Pi}, \mathsf{crs}_{\mathrm{C}})$.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$. The signer's key generation algorithm runs the key generation for the signature scheme Σ. Namely, it generates keys as

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \Sigma.\mathsf{Setup}(1^{\lambda})\ .$$

- $\mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \to (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. The user's key generation algorithm first samples a random PRF key $K \leftarrow\!\!\$\ \{0,1\}^{\lambda}$ and commitment randomness $\mathsf{op} \leftarrow\!\!\$\ \{0,1\}^{\lambda}$. Next, it computes a commitment, $\mathsf{com} \leftarrow \mathrm{C}.\mathsf{Commit}(\mathsf{crs}_{\mathrm{C}}, K; \mathsf{op})$. Finally, it outputs user's secret key and public key as $\mathsf{sk}_{\mathsf{U}} \coloneqq (K, \mathsf{op})$ and $\mathsf{pk}_{\mathsf{U}} \coloneqq \mathsf{com}$.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}) \to \bar{\sigma}$. The pre-signature issuance algorithm creates a signature on the message $\mathsf{pk}_{\mathsf{U}} \| \mathsf{c}$ as

$$\bar{\sigma} \leftarrow\!\!\$\ \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}} \| \mathsf{c})\ ,$$

and outputs $\bar{\sigma}$.

- $\mathsf{Obtain}(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}) \to (m, \sigma)$. The user's obtain algorithm first recalls $\mathsf{com} \leftarrow \mathrm{C}.\mathsf{Commit}(\mathsf{crs}_{\mathrm{C}}, K; \mathsf{op})$ and checks if $\bar{\sigma}$ is a valid signature on $\mathsf{com} \| \mathsf{c}$. If not, it outputs $\perp$ and aborts. Otherwise, it computes the message as $\mathsf{m} \coloneqq \mathrm{F}_K(\mathsf{c})$, where $(K, \mathsf{op}) =: \mathsf{sk}_{\mathsf{U}}$. It then runs the NIZK prover to create the signature σ as the following NIZK proof:

$$\sigma \leftarrow\!\!\$\ \Pi.\mathsf{Prove}\left(\mathsf{crs}_{\Pi}, \mathbb{x} \coloneqq (\mathsf{pk}, \mathsf{crs}_{\mathrm{C}}, \mathsf{m}), \mathbb{w} \coloneqq (\bar{\sigma}, \mathsf{c}, K, \mathsf{op})\right)\ .$$

Finally, it outputs m and σ.

- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \rightarrow 1/0$. The blind signature verification algorithm returns the output of $\Pi.\mathsf{Verify}(\mathsf{crs}_{\Pi}, \mathbb{x} := (\mathsf{pk}, \mathsf{crs}_{\mathrm{C}}, \mathsf{m}), \sigma)$.

We now state the main security theorem for our construction and sketch a high-level proof. The detailed proofs are provided in the full version of [24].

Theorem 3.4 (Security)**.** *Construction 3.1 is:*

(i) One-more unforgeable (Def. 3.2) assuming that the NIZK proof system Π *is an argument of knowledge, the commitment scheme* C *is statistically hiding, and the signature scheme* Σ *is existentially unforgeable under chosen message attacks.*

(ii) User and context blind (Def. 3.3) assuming that the NIZK proof system Π *satisfies zero-knowledge, the commitment scheme* C *is computationally hiding, and* F *is a secure pseudorandom function.*

Pf. sketch. Given that the PRF is a deterministic function and commitment is binding, our approach guarantees that the receiver can only obtain a single final signature from a pre-signature which allows us to prove one-more unforgeability. Further, blindness properties can be reduced to the zero-knowledge property of the underlying NIZK and hiding of the commitment scheme. The core intuition is that the NIZK proof systems are *multi-theorem* zero-knowledge, thus they can be used to simulate responses to all Obtain queries including the challenge queries, thus each blinded signature is completely unlinkable to its pre-signature, since the blinded signature is simulated without any information about the pre-signature. □

3.3 NIBS from Leveled Homomorphic Encryption

We now describe another template for designing NIBS with stronger blindness based on circuit-private leveled homomorphic encryption (LHE). We already know that homomorphic encryption with special properties [19, 141] are useful in designing round-optimal MPC protocols. Moreover, such protocols can be client-optimized where the communication complexity of a client is extremely

low. Our main observation is that we can instantiate a NIBS scheme as a specialized two-round MPC protocol where the first round message can be reused [4, 23, 59, 110, 139], and by using FHE to instantiate the protocol, we can optimize the communication complexity for the user to nearly optimal. Let us elaborate on our main ideas below.

Our key idea is that rather than using NIZKs to hide the user's secrets, we can leverage the fact that LHE enables arbitrary homomorphic operations on the user's commitment. Specifically, the user commits to a PRF secret key K by encrypting it under LHE. Using this commitment, a signer issues a pre-signature as follows: it first homomorphically evaluates the PRF $\mathrm{F}_K(\cdot)$ on some randomness c of its choice and then homomorphically generates a signature on $\mathrm{F}_K(\mathsf{c})$ treated as a message. The resulting ciphertext $\bar{\psi}$ is, therefore, an encryption of the signature on $\mathrm{F}_K(\mathsf{c})$. Under the mild assumption that the LHE is circuit-private [144], the signer can simply send $\bar{\psi}$ as the pre-signature along with an argument of knowledge that $\bar{\psi}$ was evaluated using the signing key as input to the circuit. The user sets its message to $\mathrm{F}_K(\mathsf{c})$ and obtains the corresponding signature by decrypting $\bar{\psi}$. Notably, the final signature is just a regular signature and is thus optimal in size.

We now describe our NIBS scheme from circuit-private LHE. The resulting construction has optimal-size signatures and strong security.

Construction 3.2. Our construction relies on a pseudorandom function F, a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$, a leveled homomorphic encryption scheme $\mathrm{HE} = (\mathrm{HE}.\mathsf{KeyGen}, \mathrm{HE}.\mathsf{Enc}, \mathrm{HE}.\mathsf{Dec}, \mathrm{HE}.\mathsf{Eval})$ and a NIZKAoK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIBS}}$ as described.

Language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIBS}}$

Instance: Each instance $\mathbb{x}$ is interpreted as an LHE evaluation key $\mathsf{ek}_{\mathrm{HE}}$, ciphertexts ψ and $\bar{\psi}$ and context $\mathsf{c} \in \{0,1\}^{\lambda}$.

Witness: Witness $\mathbb{w}$ consists of a secret signing key sk and randomness ρ.

Membership: Let $\mathsf{C}_{\mathsf{sk},\mathsf{c}}$ encode the circuit $\Sigma.\mathsf{Sign}(\mathsf{sk}, \mathrm{F}_{\circ}(\mathsf{c}))$ for a public pseudorandom function F. Then $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\bar{\psi} = \mathrm{HE}.\mathsf{Eval}(\mathsf{ek}_{\mathrm{HE}}, \mathsf{C}_{\mathsf{sk},\mathsf{c}}, \psi; \rho)$$

▷ $\bar{\psi}$ is the homomorphic evaluation of the ciphertext ψ over the circuit $\mathsf{C}_{\mathsf{sk},\mathsf{c}}$

Concretely, the NIBS construction is defined by the following algorithms:

- $\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}$. The setup algorithm runs the setup for the NIZK proof system Π (for language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIBS}}$):

  $$\mathsf{crs} \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^{\lambda}) \ ,$$

  and outputs $\mathsf{pp} = \mathsf{crs}$.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$. The signer's key generation algorithm runs the key generation algorithm for the signature scheme Σ. Specifically, it generates keys as

  $$(\mathsf{sk}, \mathsf{pk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^{\lambda}) \ .$$

- $\mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \to (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. The user's key generation algorithm first samples a LHE key-pair $(\mathsf{sk}_{\mathrm{HE}}, \mathsf{ek}_{\mathrm{HE}}) \leftarrow\!\!\$\ \mathrm{HE}.\mathsf{Setup}(1^{\lambda}, 1^{d})$, and a random PRF key $K \leftarrow\!\!\$\ \{0,1\}^{\lambda}$. Next, it encrypts K as $\psi \leftarrow\!\!\$\ \mathrm{HE}.\mathsf{Enc}(\mathsf{sk}_{\mathrm{HE}}, K)$ and finally outputs the user's secret and public keys as $\mathsf{sk}_{\mathsf{U}} \coloneqq (\mathsf{sk}_{\mathrm{HE}}, K)$ and $\mathsf{pk}_{\mathsf{U}} \coloneqq (\mathsf{ek}_{\mathrm{HE}}, \psi)$ respectively.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}) \to \bar{\sigma}$. Let $\mathsf{C}_{\mathsf{sk},\mathsf{c}}(\cdot)$ be the following circuit (with key sk and message c hardwired):

  $$\mathsf{C}_{\mathsf{sk},\mathsf{c}}(K) \coloneqq \Sigma.\mathsf{Sign}(\mathsf{sk}, \mathrm{F}_K(\mathsf{c})) \ .$$

That is, $\mathsf{C}_{\mathsf{sk},\mathsf{c}}$ runs the PRF function using the circuit input as the PRF key on message c, and then runs the signing algorithm to sign the output of the PRF. Let d denote the depth of the circuit $\mathsf{C}_{\mathsf{sk},\mathsf{c}}$.

The pre-signature issuance algorithm runs the homomorphic evaluation algorithm under uniformly random $\rho \leftarrow\!\$\ \{0,1\}^\lambda$, and $(\mathsf{ek}_{\mathrm{HE}}, \psi) := \mathsf{pk}_{\mathsf{U}}$, as follows:[2]

$$\bar{\psi} \leftarrow \mathrm{HE.Eval}(\mathsf{ek}_{\mathrm{HE}}, \mathsf{C}_{\mathsf{sk},\mathsf{c}}, \psi; \rho)\ ,$$

and creates a NIZK proof π for the language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIBS}}$ as

$$\pi \leftarrow\!\$\ \Pi.\mathsf{Prove}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_{\mathsf{U}}, \bar{\psi}, \mathsf{c}), \mathbb{w} := (\mathsf{sk}, \rho))$$

Finally, it outputs the pre-signature $\bar{\sigma} := (\bar{\psi}, \pi)$.

- $\mathsf{Obtain}(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}) \rightarrow (\mathsf{m}, \sigma)$. The user's obtain algorithm parses $(\bar{\psi}, \pi) =: \bar{\sigma}$ and then runs the NIZK verifier as $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}_{\mathsf{U}}, \bar{\psi}, \mathsf{c}), \pi)$. It outputs $\perp$ and aborts if the NIZK verification outputs 0. Otherwise it continues by computing the message as $\mathsf{m} := \mathrm{F}_K(\mathsf{c})$, where $(\mathsf{sk}_{\mathrm{HE}}, K) =: \mathsf{sk}_{\mathsf{U}}$. It then runs LHE decryption algorithm to compute the signature $\sigma \leftarrow\!\$\ \mathrm{HE.Dec}(\mathsf{sk}_{\mathrm{HE}}, \bar{\psi})$, and verifies that σ is a valid signature for m under pk. If the check fails, then it outputs $\perp$ and aborts; and otherwise outputs m and σ.
- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \rightarrow 1/0$. The blind signature verification algorithm returns the output of $\Sigma.\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma)$.

We now state the main security theorem for our construction and sketch a high-level proof. The detailed proofs are provided in the full version of [24].

Remark 3.2 (Honest-key model)**.** The unforgeability and blindness definitions discussed thus far have been defined in the general 'chosen-key' model where the attacker can specify arbitrary (possibly malformed) user and verification keys. For Construction 3.2, we will consider definitions under

[2] This randomness is needed for circuit privacy.

the 'honest-key' model [43, 136] as defined in Appendix B. At a high level, under this model, the attacker must provide the user's secret key in unforgeability experiment and the signer's signing key in blindness experiments, respectively.

We further remark that any NIBS scheme satisfying one-more unforgeability, user blindness, and context blindness in the honest-key model can be generically upgraded into another NIBS scheme satisfying the same security properties by using a NIZKAoK scheme. We show this formally in Appendix B.

Theorem 3.5 (Security)**.** *Construction 3.2 is:*

(i) One-more unforgeable in the honest key model (Def B.3) assuming that the NIZK proof system Π *satisfies zero knowledge, the LHE scheme* HE *is circuit private, and the signature scheme* Σ *is existentially unforgeable under chosen message attacks.*

(ii) User and context blind (Def. 3.3) assuming that the NIZK proof system Π *is an argument of knowledge, the LHE scheme* HE *is IND-CPA secure, and* F *is a secure pseudorandom function.*

Pf. sketch. At a high level, both user and context blindness follow from the IND-CPA security of the encryption scheme which ensures that ψ does not reveal anything about K, and the pseudorandomness of F, which ensures that two messages on different contexts look random. The one-more unforgeability of the protocol follows from the unforgeability of the underlying signature scheme and the circuit-privacy of the LHE. For the proof to go through, we additionally require that the adversary also prove knowledge of the users' secret keys. □

Chapter 4: Non-interactive and Batched Blind Signatures from Hard Lattice Assumptions

In many applications, users require *multiple* tokens at once rather than a single token each time they contact the issuer. Consider Privacy Pass, where users receive batches of 30–100 tokens per session [68]. This choice is not incidental, rather, it reflects a recurring need for real-world deployments as issuing tokens one at a time introduces excessive communication overhead. The importance of batched issuance is equally reflected in ongoing standardization efforts, including a recent IETF specification that proposes batched issuance techniques for privacy-preserving tokens [109].

Recognizing this gap, Hanzlik, Loss, and Wagner recently introduced the concept of batching in interactive blind signatures [102]. However, their construction only achieves amortized cost benefits. Our observation is that batching is fundamentally more natural in the non-interactive setting. In NIBS, the architecture is already aligned with batch issuance—a user simply specifies N, the desired number of signatures, and the signer issues a single pre-signature that can be un-blinded to obtain N blind signatures. Since each blind signature corresponds to a random message drawn from the message space, communication cost can remain sub-linear, and ideally constant, in batch size.

NIBS already eliminates back-and-forth communication, and a batched variant would significantly reduce both computation and communication costs, enabling large-scale deployments with minimal signer-side effort. The central question we address in this chapter, then, is on designing a practical batched NIBS scheme with provable security while maintaining communication cost independent of batch size.

After a discussion on prior works and necessary preliminaries, we introduce the concept of batching for NIBS and formalize the notion of non-interactive *batched* blind signatures (NIBBS). While the previous chapter provided two main NIBS constructions with strong security guarantees, both

This chapter is based on [24, 25].

were generic and lacked practical efficiency. Our work here takes a complementary approach, prioritizing concreteness and feasibility. We present the first practical batched NIBS scheme based on hard lattice assumptions, making it plausibly post-quantum secure. The key achievement is constant communication cost independent of batch size. At just under 1 KB, this overhead is nearly optimal compared to individual signature sizes of approximately 308 KB.

Our construction is provably secure under the hardness of the randomized one-more ISIS (rOM-ISIS) assumption, which we will also introduce and analyze. Additionally, we present an alternate construction whose security relies on the ISIS_f assumption [48], which circumvents technical limitations of the previous approach. Despite their different hardness assumptions, both constructions attain identical communication efficiency. As a special case of $N = 1$, our NIBBS scheme provides the first concretely efficient post-quantum secure NIBS with rigorous provable security. In this setting, the final blind signature occupies approximately 306 KB, with communication remaining minimal at under 1 KB.

4.1 Prior Work on Lattice-based Blind Signatures

Several round-optimal lattice-based blind signature schemes have emerged in recent years. Lyubashevsky, et al. [131] introduced a scheme based on one-time signatures under Module-SIS and Module-LWE assumptions, but it supports only a bounded number of signatures per key, with each signature approximately 150 KB. Del Pino and Katsumata [70] adapted Fischlin's paradigm to achieve unbounded signatures while reducing the size to 102 KB. Agrawal, et al. [6] employed lattice-based NIZKs to further shrink signature size to 45 KB with transcript size reduced to 1 KB under the one-more ISIS assumption (OM-ISIS). Beullens, et al. [41] pursued a different trade-off by shifting costly computation to the receiver, achieving a signature size of 22 KB at the cost of increasing the first message to several hundred kilobytes. Most recently, Jeudy and Sanders [111] demonstrated that reusing randomness from commitments to mask pre-signatures yields a 41 KB signature size alongside faster signing than previous works.

Scheme	Post-quantum secure	#Moves	Communication $N = 4$	$N = 16$	$N = 256$	$\lvert\sigma\rvert$
[102]	✗	3	67.92–175.88	206.72–588.32	2982.4–8837.12	9.41–13.98
Const. 4.1	✓	1	——— 0.74 ———			308.78

Table 4.1: Communication and signature sizes of practical batched blind signature schemes (in kilobytes). Communication is given for batches of N messages. Horizontal line through a row indicates that the number is common across all batch sizes.

Lattice-based NIBS. In our work [24] (Chapter 3), we also introduced a practical lattice-based NIBS scheme under the randomized one-more-ISIS (rOM-ISIS) assumption. While this construction satisfied the weaker security properties from Hanzlik's original NIBS framework, the generic construction from NIZK (§ 3.2) will prove crucial for the work presented in this chapter. Zhang, et al. [169] subsequently gave a lattice-based NIBS construction by partially instantiating the generic VRF approach of [101]. However, a concurrent work [91] demonstrated security flaws in their construction and proposed corrections under the rOM-ISIS assumption. We omit this work from our concrete comparisons as it provides no efficiency estimates. We note, however, that the VRF instantiation proposed by the authors [81] severely limits reusability, supporting only up to 5 VRF evaluations under a single key. While this restriction can be bypassed using the lattice-based VRF of Esgin, et al. [82], doing so incurs the cost of additional proofs for hash computations. We therefore believe our scheme offers better concrete efficiency.

Scheme	Assumption	Communication $\mathsf{U} \to \mathsf{S}$	$\mathsf{S} \to \mathsf{U}$	$\lvert\sigma\rvert$
[6]	OM-ISIS	0.96	0.56	45
[41]	MLWE, MSIS	~ 50	~ 60	22
[111]	MLWE, MSIS	8.73	50.89	41.12
Const. 4.1 ($N = 1$)	rOM-ISIS	0	0.68	306.18

Table 4.2: Comparison of state-of-the-art practical lattice-based blind signatures, all sizes in kilobytes. $\mathsf{U} \to \mathsf{S}$ indicates the total communication from the recipient to the signer (similarly $\mathsf{S} \to \mathsf{U}$). [24] is omitted from the list as it does not achieve full blindness.

Batching. Hanzlik, et al. [102] introduced the concept of batching in blind signatures and established a formal framework for batched blind signatures. They constructed a concurrently secure round-optimal blind signature based on cut-and-choose, secure under the computational Diffie-Hellman (CDH) assumption; but this is not post-quantum secure. They also introduced a formal model for batched blind signatures, and proposed a batching technique for their scheme in order to achieve reduced (amortized) communication complexity.

We give a comparison of our NIBBS scheme with other batched blind signatures in Table 4.1 and of our NIBS scheme with other state-of-the-art lattice-based blind signatures in Table 4.2.

4.2 Preliminaries

In this section, we give the preliminaries necessary for this chapter. Many of these foundations will be useful in subsequent chapters as well.

4.2.1 Gadgets

Decomposition gadget. We will utilize a suitable gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times n \cdot \ell}$, such that there exists a deterministic inverse function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \mathbb{Z}^{n \cdot \ell}$, with a "short" image and $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$ for any $\mathbf{x} \in \mathbb{Z}_q^n$. A typical example is $\mathbf{G} = \mathbf{I}_n \otimes \begin{bmatrix} 2^0 & 2^1 & \dots & 2^{\ell-1} \end{bmatrix}$ with $\mathbf{G}^{-1}(\mathbf{x})$ defined such that each of its ℓ entries has $\{0, 1\}$ coefficients. More simply, $\mathbf{G}^{-1}$ may be viewed as the (entry-wise) binary decomposition function.

Vectorization. The vectorization operation $\mathsf{Vec} : \mathbb{Z}^{n \times m} \to \mathbb{Z}^{nm}$ is a linear algebraic transformation which maps an $n \times m$ matrix to a unique vector of dimension nm. Let $\mathbf{e}_i$ be the i^{th} canonical basis vector for the m-dimensional space, then for a row-major vectorization, it holds that $\mathsf{Vec}(\mathbf{X}) = \sum_{i=1}^{m} (\mathbf{e}_i \otimes \mathbf{I}_n)\, \mathbf{X}\mathbf{e}_i$. Moreover, for $\mathbf{x} \in \mathbb{Z}^{nm}$ the inverse vectorization is given by $\mathsf{Vec}^{-1}(\mathbf{x}) \coloneqq \left(\mathsf{Vec}(\mathbf{I}_m)^\top \otimes \mathbf{I}_n\right) \cdot (\mathbf{I}_m \otimes \mathbf{x})$.

4.2.2 Crypto dark matter

Boneh, et al. [45] gave a novel strong PRF candidate which can be computed via a depth-3 ACC $[p, q]$ circuit for primes $p < q$. As stated in the overview, this bypasses the supposed PRF barrier [33] and gives a simple, low-depth PRF $\mathrm{F} : \mathbb{Z}_p^{n' \times \ell m'} \times \mathbb{Z}_p^m \to \mathbb{Z}_q^n$, for $\ell = \lceil \log_p(q) \rceil$, defined as follows:

$$\mathrm{F}_{\mathbf{K}}(\mathbf{x}) \coloneqq \mathbf{Y}_1 \cdot \left(\mathbf{K} \cdot \mathbf{G}^{-1} \left(\mathbf{Y}_0 \cdot \mathbf{x} \mod q \right) \mod p \right) \mod q \,,$$

where $\mathbf{K} \in \mathbb{Z}_p^{n' \times m'}$ is the secret PRF key, $\mathbf{x} \in \mathbb{Z}_p^m$ is the input, $\mathbf{Y}_0 \in \mathbb{Z}_q^{m' \times m}$ and $\mathbf{Y}_1 \in \mathbb{Z}_q^{n \times n'}$ ($n < n'$) are fixed public matrices and $\mathbf{G}^{-1}$ is the decomposition gadget which maps $\mathbb{Z}_q^{m'} \to \mathbb{Z}_q^{\ell m'}$.

Security. The security of the PRF is known to degrade when the key $\mathbf{K}$ is chosen to be a circulant matrix (instead of random) [65]. Aside from this, the scheme has held up to the recent cryptanalytic scrutiny [45, 74] where the best attacks require exponential time and memory.

4.2.3 Randomness Extraction

Lemma 4.1 (Leftover hash lemma [75, 76])**.** *Let $\mathcal{H} = \{\mathsf{H} : X \to Y\}_{\mathsf{H} \in \mathcal{H}}$ be a universal hash family, then for any random variable W taking values in X, the following holds*

$$\Delta_s \left((\mathsf{H}, \mathsf{H}(W)) \,, (\mathsf{H}, \mathfrak{U}_Y) \right) \leqslant \frac{1}{2} \sqrt{2^{-\mathbf{H}_\infty(W)} \cdot |Y|},$$

where $\mathfrak{U}_Y$ denotes uniform distribution over Y.

We will use the following corollary, which follows from the Leftover Hash Lemma.

Corollary 4.2. *Let $\ell > m \cdot n \log_2 q + \omega(\log n)$, q a prime, and m, n are positive integers. Let $\mathbf{R}$ be a $k \times m$ matrix chosen as per distribution $\mathcal{D}$, where $k = k(n)$ is polynomial in n and $\mathbf{H}_\infty(\mathcal{D}) = \ell$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n \times k}$ and $\mathbb{Z}_q^{n \times m}$, respectively. Then the statistical*

distance between the following distributions is negligible in n.

$$\{(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})\} \approx_s \{(\mathbf{A}, \mathbf{B})\}$$

Lemma 4.3 (Smudging lemma [19, Lemma 2.1])**.** *Let $\mathfrak{B} = \mathfrak{B}(\lambda)$ and $\mathfrak{s} = \mathfrak{s}(\lambda)$ be two positive integers and let $\mathcal{D} = \{\mathcal{D}(\lambda)\}_\lambda$ be any $\mathfrak{B}$-bounded distribution family. Let $\Gamma_\mathfrak{s} = \{\Gamma_\mathfrak{s}(\lambda)\}_\lambda$ denote the zero-centered discrete Gaussian distribution with standard deviation $\mathfrak{s}$. Then the distributions $\mathcal{D} + \Gamma_\mathfrak{s}$ and $\Gamma_\mathfrak{s}$ are statistically indistinguishable if for all $\lambda \in \mathbb{N}$, $\mathfrak{s} \geqslant \mathfrak{B} \cdot 2^{\omega(\lambda)}$.*

4.2.4 Lattice preliminaries

An m-dimensional lattice $\mathfrak{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Given positive integers n, m, q and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{z} \in \mathbb{Z}^m \ : \ \mathbf{A} \cdot \mathbf{z} = \mathbf{0} \mod q\}$. For $\mathbf{t} \in \mathbb{Z}_q^n$, we let $\Lambda_q^\mathbf{t}(\mathbf{A})$ denote the coset $\{\mathbf{z} \in \mathbb{Z}^m \ : \ \mathbf{A} \cdot \mathbf{z} = \mathbf{t} \mod q\}$.

Definition 4.4 (Discrete gaussians.)**.** Let $\mathfrak{s}$ be any positive real number. The Gaussian distribution $\Gamma_\mathfrak{s}$ with parameter $\mathfrak{s}$ is defined by the probability distribution function $\rho_\mathfrak{s}(\mathbf{z}) = \mathsf{exp}(-\pi\|\mathbf{z}\|^2/\mathfrak{s}^2)$. For any set $\mathfrak{L} \subset \mathbb{R}^m$, define $\rho_\mathfrak{s}(\mathfrak{L}) = \sum_{\mathbf{z} \in \mathfrak{L}} \rho_\mathfrak{s}(\mathbf{z})$. The discrete Gaussian distribution $\mathcal{D}_{\mathfrak{L},\mathfrak{s}}$ over $\mathfrak{L}$ with parameter $\mathfrak{s}$ is defined by the probability distribution function $\rho_{\mathfrak{L},\mathfrak{s}}(\mathbf{z}) = \rho_\mathfrak{s}(\mathbf{z})/\rho_\mathfrak{s}(\mathfrak{L})$ for all $\mathbf{z} \in \mathfrak{L}$.

The following lemma (Lemma 4.4 of [94, 137]) shows that if the parameter $\mathfrak{s}$ of a discrete Gaussian distribution is small, then any vector drawn from this distribution will be short (with high probability).

Lemma 4.5. *Let m, n, q be positive integers with $m > n$, $q \geqslant 2$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix of dimensions $n \times m$, $\mathfrak{s} = \tilde{\Omega}(n)$ and $\mathfrak{L} = \Lambda_q^\perp(\mathbf{A})$. Then*

$$\Pr[\|\mathbf{z}\| > \sqrt{m} \cdot \mathfrak{s} : \mathbf{z} \leftarrow\!\$\, \Gamma_{\mathfrak{L},\mathfrak{s}}] \leqslant \mathsf{negl}(n).$$

We will also (implicitly) require the following lemma (Lemma 4.4 of [130]) concerning the minimum-entropy of the discrete Gaussian distribution.

Lemma 4.6. *Let $\Gamma_{\mathbb{Z}^m,\mathfrak{s}}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ for any $m > 1$, with variance $\mathfrak{s}$. Then, for any $\mathfrak{s} \geqslant 3/\sqrt{2\pi}$ we have that $\mathbf{H}_\infty\left(\Gamma_{\mathbb{Z}^m,\mathfrak{s}}\right) \geqslant m$.*

Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have a "trapdoor" that allow efficient solutions to hard lattice problems. Formally,

Definition 4.7 (Lattice trapdoors [7, 94].)**.** For lattice parameters n, m, q with $m \geqslant O(n \log q)$, a trapdoor lattice sampler consists of algorithms $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ with the following syntax:

- $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, \mathbf{T_A})$: The lattice generation algorithm is a randomized algorithm that takes as input the matrix dimensions n, m, modulus q, and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with a trapdoor $\mathbf{T_A}$.

- $\mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{t}, \mathfrak{s}) \rightarrow \mathbf{z}$: The presampling algorithm takes as input a matrix $\mathbf{A}$, trapdoor $\mathbf{T_A}$, a vector $\mathbf{t} \in \mathbb{Z}_q^n$ and a parameter $\mathfrak{s} \in \mathbb{R}$ (which determines the length of the output vectors). It outputs a vector $\mathbf{z} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{z} = \mathbf{t}$ and $\|\mathbf{z}\| \leqslant \sqrt{m} \cdot \mathfrak{s}$.[1]

These algorithms must satisfy the following properties:

(i) **Well-Distributedness of matrix.** The following distributions are statistically indistinguishable:

$$\{\mathbf{A} : (\mathbf{A}, \mathbf{T_A}) \leftarrow\$ \mathsf{TrapGen}(1^n, 1^m, q)\} \approx_s \left\{\mathbf{A} : \mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}\right\} .$$

(ii) **Pre-image sampling.** For all $(\mathbf{A}, \mathbf{T_A}) \leftarrow\$ \mathsf{TrapGen}(1^n, 1^m, q)$, if $\mathfrak{s} = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:

$$\left\{\mathbf{z} : \mathbf{U} \leftarrow\$ \mathbb{Z}_q^n, \mathbf{z} \leftarrow\$ \mathsf{SamplePre}(\mathbf{A}, \mathbf{T_A}, \mathbf{U}, \mathfrak{s})\right\} \approx_s \Gamma_{\mathbb{Z}^m,\mathfrak{s}} .$$

[1] For brevity, we may occasionally use the shorthand $\mathbf{A}_{\mathfrak{s}}^{-1}(\mathbf{t})$ for the $\mathsf{SamplePre}$ algorithm.

4.2.5 Cryptographic hardness assumptions

Definition 4.8 (Learning with errors (LWE))**.** Let $q > 0$ be a function of the security parameter λ and χ be an error distribution over $\mathbb{Z}_q^m$. The (decision) $\mathsf{LWE}_{q,n,mm\chi}$ problem asks to distinguish the following instances

$$\left(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \boldsymbol{\varepsilon} \ : \ \mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow\$ \mathbb{Z}_q^n, \boldsymbol{\varepsilon} \leftarrow\$ \chi\right) \text{ and } \left(\mathbf{A}, \boldsymbol{\rho} \ : \ \mathbf{A} \leftarrow\$ \mathbb{Z}_q^{n\times m}, \boldsymbol{\rho} \leftarrow\$ \mathbb{Z}_q^m\right).$$

Definition 4.9 (Ring-LWE)**.** Let $q > 0$ be a function of the security parameter λ and χ be an error distribution over $\mathcal{R}$. The (decision) $\mathsf{RLWE}_{q,\chi}$ problem asks to distinguish the following instances

$$(a, as + \varepsilon \ : \ a \leftarrow\$ \mathcal{R}_q, s \leftarrow\$ \mathcal{R}_q, \varepsilon \leftarrow\$ \chi) \text{ and } (a, u \ : \ a \leftarrow\$ \mathcal{R}_q, u \leftarrow\$ \mathcal{R}) \ .$$

Definition 4.10 (Module-LWE)**.** Let $q, n, m > 0$ be functions of the security parameter λ and χ be an error distribution over $\mathcal{R}$. The (decision) $\mathsf{MLWE}_{q,n,m,\chi}$ problem asks to distinguish the following instances

$$\left(\mathbf{A}, \mathbf{As} + \boldsymbol{\varepsilon} \ : \ \mathbf{A} \leftarrow\$ \mathcal{R}_q^{n\times m}, \mathbf{s} \leftarrow\$ \mathcal{R}_q^m, \boldsymbol{\varepsilon} \leftarrow\$ \chi^n\right) \text{ and } \left(\mathbf{A}, \mathbf{u} \ : \ \mathbf{A} \leftarrow\$ \mathcal{R}_q^{n\times m}, \mathbf{u} \leftarrow\$ \mathcal{R}^n\right) \ .$$

Definition 4.11 (Module-(I)SIS)**.** Let $q, n, m, \mathfrak{B}$ be functions of security parameter λ. Given a matrix $\mathbf{A} \leftarrow\$ \mathcal{R}_q^{n\times m}$, the (search) $\mathsf{MSIS}_{q,n,m,\mathfrak{B}}$ problem asks to find a vector $\mathbf{s} \in \mathcal{R}^m$ such that $\|\mathbf{s}\|_2 \leqslant \mathfrak{B}$ and $\mathbf{A} \cdot \mathbf{s} = \mathbf{0} \pmod q$. The inhomogeneous version of the problem, $\mathsf{MISIS}_{q,n,m,\mathfrak{B}}$, replaces the zero vector by some target vector $\mathbf{t} \in \mathcal{R}_q^n$.

4.3 The Formal NIBBS Framework

We introduce a new generalization of NIBS called non-interactive batched blind signatures (NIBBS). The key capability of NIBBS is to allow a user to derive an entire batch of message and blind signature pairs from a single pre-signature. As suggested earlier, this drastically reduces communication overhead between signer and user in applications requiring multiple blind signatures.

For simplicity, assume a globally fixed batch size N. A NIBBS scheme consists of the following algorithms. The Setup algorithm generates global public parameters pp, which serve as a common reference string (CRS). Two key generation algorithms produce $(\mathsf{sk}, \mathsf{pk})$ for the signer via $\mathsf{KeyGen}_{\mathsf{S}}$ and $(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$ for the user via $\mathsf{KeyGen}_{\mathsf{U}}$. The signer's randomized Issue algorithm takes the signer's secret key sk, the signing context c, and a user's public key pk_{U}, and outputs a pre-signature $\bar{\sigma}$. Finally, the user's Obtain algorithm uses the pre-signature to derive N message-signature pairs $[\![\mathsf{m}, \sigma]\!] \coloneqq \{(\mathsf{m}_1, \sigma_1), (\mathsf{m}_2, \sigma_2), \ldots, (\mathsf{m}_N, \sigma_N)\}$.

In summary, a non-interactive batched blind signature scheme extends a standard NIBS scheme so that, given a single random pre-signature $\bar{\sigma}$, a user with public key pk_{U} can extract $N \in \mathbb{N}$ blind signatures $\sigma_1, \sigma_2, \ldots, \sigma_N$ for the corresponding random messages $\mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_N$ using its secret key sk_{U}. Formally, a NIBBS consists of the following polynomial-time algorithms:

- $\mathsf{Setup}(1^\lambda, N) \to \mathsf{pp}$. On input the security parameter λ and a batch size N, the global setup algorithm outputs a set of public parameters pp. All algorithms take pp as an input, but we usually omit it as an explicit input for brevity.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$. On input pp, the signer's key generation algorithm samples a public (verification)-secret key pair $(\mathsf{sk}, \mathsf{pk})$.

- $\mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \to (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. On input pp, the user's key generation algorithm samples a public-secret key pair $(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}) \to \bar{\sigma}$. The signer's issuance algorithm takes as input the signer's secret key sk, a user's public key pk_{U} and tbe signing context c. It then outputs a pre-signature $\bar{\sigma}$.

- $\mathsf{Obtain}(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}) \to [\![\mathsf{m}, \sigma]\!]/\perp$. The user's blind signature extraction algorithm takes as input the user's secret key sk_{U}, along with a verification key pk and pre-signature and context pair $(\bar{\sigma}, \mathsf{c})$, and outputs N message-signature pairs $[\![\mathsf{m}, \sigma]\!] \coloneqq \{(\mathsf{m}_1, \sigma_1), (\mathsf{m}_2, \sigma_2), \ldots, (\mathsf{m}_N, \sigma_N)\}$ or aborts and outputs $\perp$.

- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \to 1/0$. The signature verification algorithm that takes as input a verification key pk and message-signature pair (m, σ), and outputs 1 (accept) or 0 (reject).

A NIBBS scheme must satisfy correctness.

Definition 4.12 (Correctness)**.** For every $\lambda, N \in \mathbb{N}$ and $\mathsf{c} \in \{0,1\}^\lambda$, a non-interactive batched blind signature scheme satisfies *correctness* if

$$\Pr\left[\bigwedge_{\mathsf{i}\in[N]} \mathsf{Verify}(\mathsf{pk}, \mathsf{m}_i, \sigma_i) = 1 \ : \begin{array}{r} \mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda, N) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}), \\ (\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \\ \bar{\sigma} \leftarrow\!\$\ \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}, \mathsf{c}) \\ [\![\mathsf{m}, \sigma]\!] \leftarrow\!\$\ \mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \mathsf{c}, \bar{\sigma}) \end{array}\right] = 1\ .$$

We further require the NIBBS scheme to be succinct.

Definition 4.13 (Succinct communication)**.** For $\lambda \in \mathbb{N}$, there exists a polynomial $\mathsf{poly}(\lambda)$ such that $|\bar{\sigma}|, |\mathsf{c}| \leqslant \mathsf{poly}(\lambda)$.

Game $\mathsf{OM\text{-}Unf}_{\mathsf{NIBBS},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp})$
3 : $\{(\mathsf{m}_i, \sigma_i)\}_{i=1}^{N\ell+1} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Issue}}}(\mathsf{pp}, \mathsf{pk})$
4 : **if** $\bigvee_{1\leqslant j<j'\leqslant N\ell+1} (\mathsf{m}_j = \mathsf{m}_{j'})$
$\bigvee_{j\in[N\ell+1]} \mathsf{Verify}(\mathsf{pk}, \mathsf{m}_j, \sigma_j) \neq 1$ **then**
5 : **reject**
6 : **accept**

Oracle $\mathcal{O}_{\mathsf{Issue}}(\mathsf{pk}_\mathsf{U}, \mathsf{c})$

1 : **if** ℓ is uninitialized **then**
2 : $\ell := 0$
3 : $\ell \leftarrow \ell + 1$
4 : **return** $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}, \mathsf{c})$

Figure 4.1: The one-more unforgeability experiment for NIBBS.

In terms of security, NIBBS should satisfy two properties: one-more unforgeability and blindness. One-more unforgeability in the NIBBS context is a natural extension of the same property for NIBS. An adversary who observes ℓ pre-signatures for user public keys of its choice must be unable to produce $(N\ell + 1)$ valid signatures. Formally,

Definition 4.14 (One-more unforgeability)**.** For all $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathsf{OM\text{-}Unf}}_{\mathsf{NI\underline{B}BS},\mathcal{A}} = \Pr\left[\mathsf{OM\text{-}Unf}_{\mathsf{NI\underline{B}BS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 4.1. We say a non-interactive blind signature scheme, $\mathsf{NI\underline{B}BS}$, is *one-more unforgeable* if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

Blindness requires that a final signature cannot be linked to its corresponding pre-signature. Since the user now derives an entire batch of signatures from a single pre-signature, we must ensure unlinkability not only between signatures from different batches, but also between signatures *within* the same batch. We therefore define two distinct security notions: *inter-batch blindness* and *intra-batch blindness*.

For inter-batch blindness, the setting closely mirrors the non-batched case, with one key modification. The adversary has access to the Obtain oracle with respect to two different user secret keys, $\mathsf{sk}_{\mathsf{U}_0}$ and $\mathsf{sk}_{\mathsf{U}_1}$, and outputs a pair of pre-signature and context tuples $(\bar{\sigma}_\omega, \mathsf{c}_\omega)$ for $\omega \in \{0, 1\}$. The challenger then runs the Obtain algorithm on each tuple, producing two batches of message and signature pairs, $[\![\mathsf{m}, \sigma]\!]_b$ and $[\![\mathsf{m}, \sigma]\!]_{1-b}$, for a random bit b. The adversary wins by correctly guessing which batch corresponds to which pre-signature.

Definition 4.15 (Inter-batch blindness)**.** For all $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\top\text{-}\mathsf{Blind}}_{\mathsf{NI\underline{B}BS},\mathcal{A}} = \Pr\left[\top\text{-}\mathsf{Blind}_{\mathsf{NI\underline{B}BS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

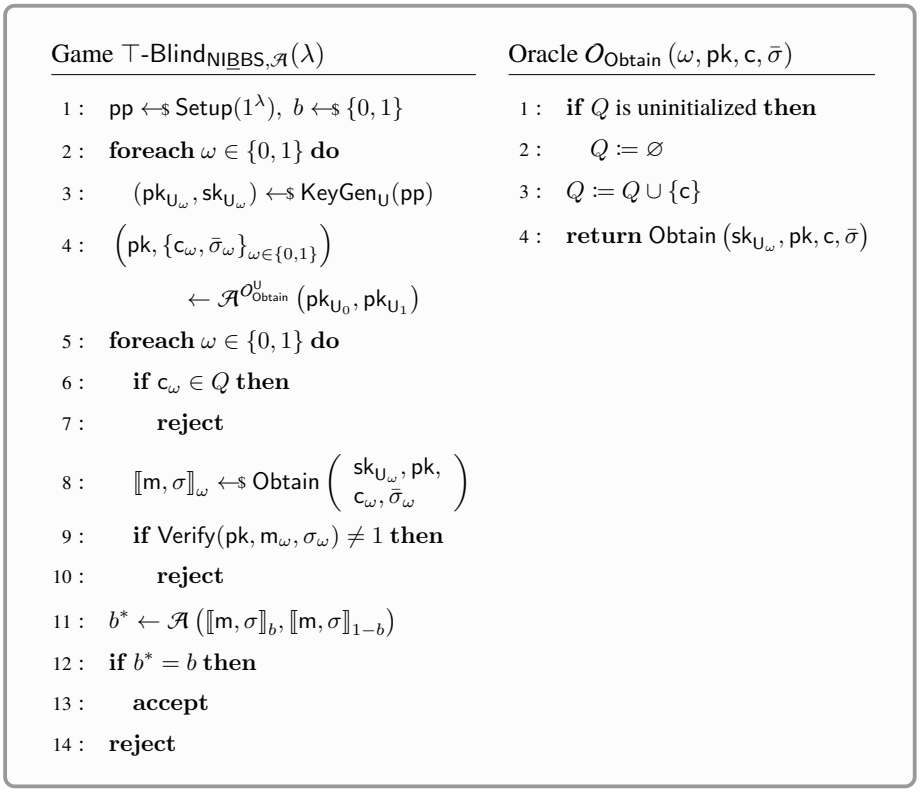

Figure 4.2: The inter-batch blindness experiments for NIBBS.

be the advantage of a *stateful* adversary $\mathcal{A}$ in the experiment defined in Figure 4.2. We say a non-interactive batched blind signature scheme, NIBBS, satisfies *inter-batch blindness* if for any PPT adversary $\mathcal{A}$ each of these advantages is at most $1/2 + \mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\cdot)$.

For intra-batch blindness, the adversary has access to the Obtain oracle with respect to a single user secret key, sk_{U}. The adversary outputs a pair of pre-signature and context tuples $(\bar{\sigma}_\omega, \mathsf{c}_\omega)$ for $\omega \in \{0,1\}$, along with a permutation map μ. The challenger runs the Obtain algorithm on each tuple, generating two batches of N message and signature pairs each. These $2N$ pairs are then presented to the adversary either in their original order if a random bit $b = 0$, or permuted according to μ if $b = 1$. The adversary's goal is to correctly guess b. This definition ensures that an adversary cannot link individual signatures to their originating batches, thereby capturing the intra-batch blindness property.

Definition 4.16 (Intra-batch blindness)**.** For all $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\perp\text{-Blind}}_{\mathsf{NI\underline{B}BS},\mathcal{A}} = \Pr\left[\perp\text{-}\mathsf{Blind}_{\mathsf{NI\underline{B}BS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of a *stateful* adversary $\mathcal{A}$ in the experiment defined in Figure 4.3. We say a non-interactive blind signature scheme, $\mathsf{NI\underline{B}BS}$, satisfies *intra-batch blindness* if for any PPT adversary $\mathcal{A}$ each of these advantages is at most $1/2 + \mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\cdot)$.

Game $\perp\text{-}\mathsf{Blind}_{\mathsf{NI\underline{B}BS},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$,
2 : $b \leftarrow\!\$\ \{0,1\},\ \mu_0 \coloneqq \iota$
3 : $(\mathsf{pk_U}, \mathsf{sk_U}) \leftarrow\!\$\ \mathsf{KeyGen_U}(\mathsf{pp})$
4 : $\left(\mathsf{pk}, \{\mathsf{c}_\omega, \bar{\sigma}_\omega\}_{\omega\in\{0,1\}}, \mu_1\right)$
$\leftarrow \mathcal{A}^{O^{\mathsf{U}}_{\mathsf{Obtain}}}(\mathsf{pk_U})$
5 : **if** μ_1 is not a valid permutation map **then**
6 : **rejectforeach** $\omega \in \{0,1\}$ **do**
7 : **if** $\mathsf{c}_\omega \in Q$ **then**
8 : **reject**
9 : $[\![\mathsf{m}, \sigma]\!]_\omega \leftarrow\!\$\ \mathsf{Obtain}\left(\begin{array}{l}\mathsf{sk}_{\mathsf{U}_\omega}, \mathsf{pk},\\ \mathsf{c}_\omega, \bar{\sigma}_\omega\end{array}\right)$
10 : **if** $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}_\omega, \sigma_\omega) \neq 1$ **then**
11 : **reject**
12 : $b^* \leftarrow \mathcal{A}\left(\left\{\left(\mathsf{m}_{\mu_b(\mathrm{i},\omega)}, \sigma_{\mu_b(\mathrm{i},\omega)}\right)\right\}_{\substack{\mathrm{i}\in[N]\\ \omega\in\{0,1\}}}\right)$
13 : **if** $b^* = b$ **then**
14 : **accept**
15 : **reject**

Oracle $O_{\mathsf{Obtain}}(\mathsf{pk}, \mathsf{c}, \bar{\sigma})$

1 : **if** Q is uninitialized **then**
2 : $Q \coloneqq \varnothing$
3 : $Q \coloneqq Q \cup \{\mathsf{c}\}$
4 : **return** $\mathsf{Obtain}(\mathsf{sk_U}, \mathsf{pk}, \mathsf{c}, \bar{\sigma})$

Figure 4.3: The intra-batch blindness experiments for NIBBS. Here μ_0 and μ_1 are bijective permutation maps in $[N] \times \{0,1\} \to [N] \times \{0,1\}$ with μ_0, in particular, being the identity map ι.

4.4 Randomized One-more ISIS Assumption

We introduce a new variant of the OM-ISIS assumption [6] with two goals: (i) protect from attackers that can ask for pre-image queries on the same target vector more than once, and (ii) allow for re-randomization of the target vectors before answering the pre-image query phase to make the assumption more robust. As discussed in [6, § 4.5], there are polynomial time attacks on the OM-ISIS assumptions for certain parameter regimes. At its core, all cryptanalysis efforts on OM-ISIS exploit the fact that the attacker can submit any target vector of its choice as a pre-image query. Thus, an attacker can potentially request for pre-image queries for short vectors, and use those to create an approximate trapdoor. However, the quality of trapdoor computed this way is much worse than the actual trapdoor, thus if the parameter $\mathfrak{B}$ is set appropriately (i.e., sufficiently small), then an attacker cannot break the assumption since the approximate trapdoor will not give as short pre-image vectors.

While existing cryptanalytic efforts do not succeed in breaking the assumption for the desired parameter regimes, we view the combinatorial and lattice-based attacks provided by [6] as partial evidence of existing assumption not being as robust. Moreover, we believe that while OM-ISIS is a very natural step towards defining lattice-analogue of the one-more-RSA assumption by Bellare, et al. [37], there exists more robust instantiations for a family of one-more assumptions in lattice-based cryptography. To that end, we propose a strengthening of the OM-ISIS assumption that we call as randomized one-more ISIS (rOM-ISIS) assumption.

Our intuition is to draw more inspiration from GPV signatures [94]. Recall that in GPV signatures, a signature is computed as a pre-image of the output of a hash function (modeled as a random oracle). While modeling the hash function as a random oracle enables a reduction to plain ISIS by a standard random oracle programming argument, usage of any specific hash function (such as SHA-3) is not known to enable any efficient attacks. At a high level, the hash function protects the signer from simple algebraic manipulations of multiple pre-image vectors to create a new valid pre-image vector for a fresh hash output. A little more abstractly, this means that given pre-image vectors $\left\{\mathbf{t}_i \leftarrow \mathbf{A}_{\mathfrak{s}}^{-1}(\mathsf{H}(\mathsf{m}_i))\right\}_i$, it is unclear how to find short coefficients α_i such that $\sum \alpha_i \mathbf{t}_i =$

$\mathbf{A}_{\mathfrak{s}}^{-1}(\mathsf{H}(\mathsf{m}^*))$. That is, the hash function prevents from creating a valid pre-image to the hash function which is a short linear combination of other hash values.

Inspired by the intuitive structural guarantee provided by a hash function, we propose the following assumption:

Definition 4.17 (rOM-ISIS)**.** Let $q, n, m, \mathfrak{s}, \mathfrak{B}$ be functions of security parameter λ. Consider the following experiment:

(i) The challenger uniformly samples two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$, and sends $\mathbf{A}, \mathbf{B}$ to adversary $\mathcal{A}$.

(ii) $\mathcal{A}$ adaptively makes queries of the following types to the challenger, in any order.

- **Syndrome queries.** $\mathcal{A}$ requests for a challenge vector, to which the challenger replies with a uniformly sampled vector $\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_q^n$. We denote the set of received vectors by $\mathcal{S}$.
- **Pre-image queries.** $\mathcal{A}$ queries a vector $\hat{\mathbf{t}} \in \mathbb{Z}_q^n$, to which the challenger replies with a short vector $\hat{\mathbf{s}} \in \mathbb{Z}_q^m$ and a ± 1 vector $\hat{\mathbf{r}} \in \mathbb{Z}_2^m$ such that $\mathbf{A} \cdot \hat{\mathbf{s}} + \mathbf{B} \cdot \hat{\mathbf{r}} = \hat{\mathbf{t}}$ and $\|\hat{\mathbf{s}}\| \leqslant \mathfrak{s}\sqrt{m}$. Let Q denote the total number of pre-image queries.

(iii) Finally, $\mathcal{A}$ outputs $Q+1$ tuples of the form $\{(\mathbf{s}_j, \mathbf{r}_j, \mathbf{t}_j)\}_{j \in [Q+1]}$. $\mathcal{A}$ wins if

$$\forall j \in [Q+1],\ \ \mathbf{A} \cdot \mathbf{s}_j + \mathbf{B} \cdot \mathbf{r}_j = \mathbf{t}_j,\ \|\mathbf{s}_j\| \leqslant \mathfrak{B},\ \mathbf{r}_j \in \mathbb{Z}_2^m \text{ and } \mathbf{t}_j \in \mathcal{S}\ \ .$$

The $\mathsf{rOM\text{-}ISIS}_{q,n,m,\mathfrak{s},\mathfrak{B}}$ assumption states that for every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins is $\mathsf{negl}(\lambda)$.

Intuitively, the new assumption says an attacker does not receive pre-images for arbitrary target vectors $\widehat{\mathbf{t}}$ that it selects, but for publicly re-randomized vector $\mathbf{t}' = \widehat{\mathbf{t}} - \mathbf{B} \cdot \mathbf{y}$, where $\mathbf{y}$ is a random ± 1 vector (chosen by the challenger). We even provide the attacker with the vector $\mathbf{y}$. The point is that the ability to re-randomize the target vector, gives the challenger more flexibility in answering pre-image queries. Now an attacker can win as long as it creates pre-image vectors for any re-randomization of the random syndrome vectors. That is, we allow the attacker to also select any

arbitrary ± 1 vector and use it to re-randomize each syndrome vector. The only constraint is that the vectors $\mathbf{y}_j$ are ± 1 vectors.

Unlike [6], we do not make any additional parameter restrictions. This is primarily due to the fact that we have been unable to find any practical attacks for any standard ISIS parameter regimes for the rOM-ISIS assumption. In our perspective, the condition that $\mathbf{y}_j$ must be ± 1 vectors prevents the algebraic attacks that were mounted on the OM-ISIS assumption. Additionally, one could view the $\mathbf{B} \cdot \mathbf{y}$ term as enforcing a structural property, on the target vectors, that a hash function enforces for GPV signatures. We provide some preliminary cryptanalysis next.

4.4.1 Cryptanalysis of the rOM-ISIS assumption

We analyse the robustness of the randomized one-more ISIS assumption. In order to do so, we consider the two categories of attacks presented against the one-more ISIS assumption by Agrawal, et al. [6], and show that each of these attacks is in fact harder in the randomized one-more ISIS setting.

Lattice-based attack strategy

Let us begin by considering the lattice-based attack strategy against the rOM-ISIS assumption. We will argue that under this assumption, it is hard for an attacker to even efficiently *construct* a good basis for $\Lambda_q^{\perp}(\mathbf{C})$. To that end, suppose the attacker makes Q pre-image queries for $\mathbf{t}_i = \mathbf{0}$ to the rOM-ISIS oracle and receives $\{(\mathbf{z}_i, \mathbf{y}_i)\}_{i \in [Q]}$ such that for each $i \in [Q]$, $\mathbf{C} \cdot \mathbf{z}_i + \mathbf{B} \cdot \mathbf{y}_i = \mathbf{0}$, $\mathbf{z}_i$ is short and $\mathbf{y}_i \in \{\pm 1\}^m$.

We observe that if there exist $a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q$ such that $\mathbf{B} \cdot \sum_{i \in [Q]} (a_i \cdot \mathbf{y}_i) = \mathbf{0}$, then $\boldsymbol{\alpha} \coloneqq \sum_{i \in [Q]} (a_i \cdot \mathbf{z}_i)$ is a vector on the lattice $\Lambda_q^{\perp}(\mathbf{C})$. Further, for γ such that $|a_i| \leqslant \gamma$ for all $i \in [Q]$ we have

- $\boldsymbol{\alpha}$ has norm bounded above by $Q\gamma\sqrt{m} \cdot \mathfrak{s}$ with all but negligible probability; and
- $\left\| \sum_{i \in [Q]} (a_i \cdot \mathbf{y}_i) \right\|_2 \leqslant Q\gamma\sqrt{m} = \|\boldsymbol{\alpha}\|_2 / \mathfrak{s}$

At this stage, we identify two types of potential attack scenarios:

- **Case 1.** An attacker is able to find a collection $a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q$ as defined in above, such that $|a_i| \leqslant \gamma$ for all a_i. It can use this to construct a short basis for $\Lambda_q^{\perp}(\mathbf{C})$ and proceed as in step 3 of the lattice attack above.
- **Case 2.** An attacker is unable to find a collection $a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q$ of bounded length, and instead proceeds in some other way.

Lemma 4.18. *Assuming the hardness of* $\mathsf{SIS}_{q,n,m,\mathfrak{B}'}$ *for* $\mathfrak{B}' = Q\gamma\sqrt{m}$*, an attacker of the first type exists with negligible probability.*

Proof. This follows by recalling that $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ is a random matrix, and observing that $\sum_{i \in [Q]}(a_i \cdot \mathbf{y}_i)$ is a solution to $\mathsf{SIS}_{q,n,m,\mathfrak{B}'}$ where $\mathfrak{B}' = \|\boldsymbol{\alpha}\|/\mathfrak{s} = Q\gamma\sqrt{m}$. □

Essentially, to match the OM-ISIS norm bound on the lattice basis as in [6] we require $\|\boldsymbol{\alpha}\| = \sqrt{m} \cdot \mathfrak{s}$, which suggests that finding even one such collection of a_i's is at least as hard as solving $\mathsf{SIS}_{q,n,m,\mathfrak{B}'}$ for $\mathfrak{B}' = \Theta(\sqrt{n \log q})$.

Next, while it is evident that an attacker of the second type cannot proceed according to the lattice-based attack strategy of [6], it is less clear how else the attacker might proceed. In particular, we find that the process of constructing a good basis would ultimately involve taking some combination of the pre-image queries. However, any such combination that does not satisfy the case 1 condition will also result in a larger norm on the solutions. Further cryptanalysis and discovery of an alternative lattice-based attack strategy is left as an interesting open problem.

A more generalized attack. We can generalize this attack a little further: suppose the attacker makes Q pre-image queries for *any* $\mathbf{t}_i \in \mathbb{Z}_q^n$ to the $\mathsf{rOM\text{-}ISIS}$ oracle and receives $\{(\mathbf{z}_i, \mathbf{y}_i)\}_{i \in [Q]}$ such that for each $i \in [Q]$, $\mathbf{C} \cdot \mathbf{z}_i + \mathbf{B} \cdot \mathbf{y}_i = \mathbf{t}_i$.

Now, one strategy could be to guess $\mathbf{y}_i$ and set the query $\mathbf{t}_i = \mathbf{B} \cdot \mathbf{y}_i$, but the probability of guessing correctly is negligible. Alternatively, the attacker can try to find $a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q$ such that $\mathbf{B} \cdot \sum_{i \in [Q]}(a_i \cdot \mathbf{y}_i) = \sum_{i \in [Q]} a_i \cdot \mathbf{t}_i$ as in the previous case, then too $\boldsymbol{\alpha} \coloneqq \sum_{i \in [Q]}(a_i \cdot \mathbf{z}_i)$

is a vector on the lattice $\Lambda_q^{\perp}(\mathbf{C})$. Indeed, a similar analysis shows that the attack makes Q queries, solves at least m instances of $\mathsf{ISIS}_{q,n,m,\mathfrak{B}'}$ for $\mathfrak{B}' = \|\boldsymbol{\alpha}\|/\mathfrak{s}$, and outputs a solution to rOM-ISIS for $\mathfrak{B} = \Theta\left(\sqrt{m} \cdot \|\boldsymbol{\alpha}\|\right)$. Beyond these attacks, we could not find any efficient (sub-exponential time) lattice attacks against our new assumption.

Combinatorial attack strategy

Turning now to the combinatorial attack, we consider an attacker that does the following:

- ▷ The attacker constructs the set $A = \{a \cdot \mathbf{e}_i + \mathbf{B} \cdot (\mathbf{y}'_j - n^{-1}\mathbf{1}) \mid \forall i \in [n],\ \forall j \in [Q], a \in \mathbb{Z}_q\}$ where $\mathbf{e}_i$'s are the canonical n-dimensional basis vectors, and each $\mathbf{y}'_j \in \{\pm 1\}^m$.
- ▷ It then queries the rOM-ISIS oracle for a pre-image of each vector in A.
- ▷ Consider any challenge vector $\mathbf{t} \in \mathbb{Z}_q^n$. The attacker finds a collection of n vectors of the form $a_i \cdot \mathbf{e}_i$ such that they sum to $\mathbf{t}$ where each $a_i \in \mathbb{Z}_q$.
- ▷ Let $(\mathbf{z}_{i,j}, \mathbf{y}_{i,j})$ be the oracle response for query $a_i \cdot \mathbf{e}_i + \mathbf{B} \cdot (\mathbf{y}'_j - n^{-1}\mathbf{1})$. For each $i \in [n]$, the next step is to find any $j \in Q$ such that $\mathbf{y}'_j = \mathbf{y}_{i,j}$. For every i, the probability that such a j exists is $1 - (1 - 2^{-m})^Q \approx 1 - e^{-Q/2^m}$, which is bounded away from zero for $Q > 2^m$.
- ▷ Let set I contain all such pairs (i, j), and set J contain the corresponding j. Then, the following holds

$$\mathbf{C} \cdot \sum_{(i,j)\in I} \mathbf{z}_{i,j} + \mathbf{B} \cdot \sum_{(i,j)\in I} \mathbf{y}_{i,j} = \sum_{i\in[n]} a_i \mathbf{e}_i + \mathbf{B} \cdot \sum_{j\in J} (\mathbf{y}'_j - n^{-1}\mathbf{1}) \quad .$$

  Which upon simplifying gives $\mathbf{C} \cdot \left(\sum_{(i,j)\in I} \mathbf{z}_{i,j}\right) + \mathbf{B} \cdot \mathbf{1} = \mathbf{t}$.

Recall that with all but negligible probability the original pre-images $\mathbf{z}_{i,j}$ have norms bounded by $\sqrt{m} \cdot \mathfrak{s}$ (Lemma 4.5). Thus, $\left(\left(\sum_{(i,j)\in I} \mathbf{z}_{i,j}\right), \mathbf{1}\right)$ is a valid rOM-ISIS solution for $\mathfrak{B} = \Theta\left(n\mathfrak{s}\sqrt{m}\right)$

with overwhelming probability. Notably, this attack is computationally intractable for any PPT adversary making polynomially many queries, as the probability of finding set I (and J) goes to zero when $Q = o(2^m)$.

4.5 Lattice-based NIBBS from rOM-ISIS

The starting point for our design is the generic template for NIBS given in Section 3.2. As it turns out, this template can be quite naturally generalized to support *batching*. In particular, instead of evaluating the PRF on context c, the user evaluates it on $\mathsf{c}\|\mathsf{i}$ for each $\mathsf{i} \in [N]$, thus obtaining N distinct messages, and their corresponding signatures. In order to guarantee unforgeability, the statement for the i^{th} batch must additionally require that $\mathsf{i} \leqslant N$, with i as part of the witness.

The next step is to instantiate the generic approach. Let the vector $\mathbf{k}$ of small norm be the user's secret PRF key. Then, for a random vector $\mathbf{u}$, and public matrix $\mathbf{C}$, the user's public key pk_{U} is given by $\mathbf{t} \coloneqq \mathbf{C} \cdot \mathbf{k} + \mathsf{H}(\mathbf{u})$. Given $\mathbf{t}$, the signer with public verification key matrices $\mathbf{A}$ and $\mathbf{B}$ can now create a pre-signature by (pre-)signing $\mathbf{t}$ along with the context vector $\mathbf{c}$ by using trapdoor lattice sampling [94] (see also [6, 24, 41]) to sample a short preimage $\mathbf{s}$ with respect to $\mathbf{A}$ such that $\mathbf{A} \cdot \mathbf{s} + \mathbf{B} \cdot \mathbf{c} = \mathbf{t}$ and send, both, pre-signature $\mathbf{s}$ and context $\mathbf{c}$ to the user.

On receiving the pre-signature and context pair, the user can compute N random messages as the PRF evaluation $\mathsf{m}_{\mathsf{i}} \coloneqq \mathrm{F}_{\mathbf{k}}\left(\mathbf{c}\|\vec{\mathsf{i}}\right)$ (where $\vec{\mathsf{i}}$ is the binary representation of $\mathsf{i} \in [N]$), and the corresponding signatures as the NIZK proof π_{i} stating that, given $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and m_{i}, there exist vectors $\mathbf{k}, \mathbf{u}, \mathbf{s}$ and $\mathbf{c}$, and integer i such that $\mathbf{A} \cdot \mathbf{s} + \mathbf{B} \cdot \mathbf{c} = \mathbf{C} \cdot \mathbf{k} + \mathsf{H}(\mathbf{u})$, $\mathsf{m}_{\mathsf{i}} = \mathrm{F}_{\mathbf{k}}\left(\mathbf{c}\|\vec{\mathsf{i}}\right)$, $\mathsf{i} \in [N]$ and $\|\mathbf{k}\|, \|(\mathbf{s}, \mathbf{c})\|$ are short.

A technical limitation for such Fischlin-style approaches is that the NIZK must be straight-line extractable. Fortunately, we can utilize the folklore "encryption-to-the-sky" trick and have the user encrypt its secrets using public key encryption. In particular, this allows a challenger, holding the secret decryption key sk_{E}, to extract all the secrets in a straight-line manner. Let ψ_{i} be the encryption of the witness to the relation under public key pk_{E}, sampled during setup, and let ρ_{i} denote the encryption randomness. Then, each NIZK must also prove correct computation of ψ_{i} where ψ_{i} and

pk_{E} are part of the instance, and ρ_{i} of the witness. Importantly, the final signature must now also include ψ_{i}.

Efforts to construct practically efficient (non-)interactive blind signature schemes from hard lattices assumptions [6, 24, 41] have largely relied on the use of the NIZK proof system of Lyubashevsky, et al. [132] which produces very short proofs for affine relations. It is therefore crucial in these works that the entire NIZK relation is also composed of affine equations (and norm bounds). Thus, in order to build practical lattice-based $\mathsf{NI\underline{B}BS}$, we would need to be able to specify the lattice-instantiation explicitly in linear terms. However, for the security of the protocol we need to be able to prove that the final (public) message m_{i} was computed as a PRF evaluation of $(\mathbf{c}\|\vec{\mathsf{i}})$ with key $\mathbf{k}$ where, in particular, $\mathbf{c}$ and $\mathbf{k}$ are hidden as part of the witness of the NIZK (otherwise, blindness can be trivially broken).

This underpins the core hurdle in instantiating the generic template for $\mathsf{NI\underline{B}BS}$ via affine relations in order to apply the [132] framework. Namely, that all popular approaches for designing secure PRFs from lattice-based assumptions require very high degree computations in the inputs and PRF keys [31, 32, 46]. This technical tension suggests that any natural attempt in instantiating the generic template using practical lattice-based NIZKs will ultimately fail. It is also in part why in [24] we could only design lattice-based NIBS with *very weak security* [101]. In the work, we simply omitted the PRF computation entirely and replaced it with a random linear function. We proved that as long as blindness is only desired against at most two signatures, then their NIBS are secure. Moreover, we suggested that one could easily break blindness of their lattice-based NIBS protocol if an attacker learns as few as three blind signatures.

We remark that for many existing lattice-based PRFs [31, 32, 46], their evaluation can be segmented into $\sim \lambda$ incremental checks of degree-1 (or 2) such that by linking all $O(\lambda)$ checks together, it can be proven using the [132] framework that the PRF was correctly computed (this approach was used in [12, 14]). Unfortunately, the family of lattice-based PRFs involve rounding down the final computation (modulo q), to a lower modulus p such that q/p is superpolynomial in the security parameter; an artifact of the reduction from the learning with rounding (LWR) assumption, on which

they rely, to the learning with errors (LWE) assumption. Consequently, the modulus for the NIZK proof system, which itself has to be greater than q for soundness, is rather large in practice.

Given, these challenges inherent in instantiating the protocol, we make two crucial design choices intended to make our final approach practical. Firstly, we instantiate our scheme with the Crypto Dark Matter PRF candidate [45] which can be computed by depth-3 arithmetic circuit. Secondly, we use the lattice-based general purpose SNARK LaBRADOR [42] which provides compact proofs for rank-1 constraint systems (R1CS) to avoid any dependence on [132].[2]

Remark 4.1 (Dynamic batch sizes)**.** While our above exposition considers a fixed batch size N, our approach is much more general. In particular, we can design $\mathsf{NI\underline{B}BS}$ with truly dynamic batch sizes. For instance, consider that each user–when it contacts a signer–provides a (user-specific) batch size N to the signer. Now to handle such adaptive batch sizes, we make a small adjustment to the pre-signature computation. Instead of creating a (standard) digital signature for just the user's public key $\mathsf{pk}_{\mathsf{U}} := \mathbf{t}$, it can create the signature for $\mathsf{pk}_{\mathsf{U}} \| N$. Now, during the final blind signature generation, the user treats N as part of the NP witness. This readily gives us $\mathsf{NI\underline{B}BS}$ with fully dynamic batch size. In the main body, we consider static batch sizes for ease of exposition, however our ideas can be easily generalized as explained here.

Construction 4.1. Let $N \in \mathbb{N}$ be the batch size, and let parameters $n, m, n', m', \mathfrak{s}, \mathfrak{B} = \mathfrak{s}\sqrt{m}, \mathfrak{B}_N = (\mathfrak{s} + n'm_N)\sqrt{m}$, and positive integer q be functions of the security parameter λ such that the randomized one-more ISIS instance $\mathsf{rOM\text{-}ISIS}_{q,n,m,\mathfrak{s},\mathfrak{B}_N}$ is hard. These parameters must satisfy the following constraints:

$$n = \mathsf{poly}(\lambda),\ n' > n,\ m > n \log q + \Theta(\lambda),\ n'm_N > n \log q + \Theta(\lambda)\ . \tag{4.1}$$

Our construction relies on a hash function $\mathsf{H} : \mathbb{Z}_2^\lambda \to \mathbb{Z}_q^n$ (modeled as a random oracle), a lattice trapdoor $\mathcal{T} = (\mathcal{T}.\mathsf{TrapGen}, \mathcal{T}.\mathsf{SamplePre})$, a public key encryption scheme $\mathrm{E} = (\mathrm{E}.\mathsf{KeyGen}, \mathrm{E}.\mathsf{Enc}, \mathrm{E}.\mathsf{Dec})$ and a NIZK proof system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for relation $\mathcal{R}^{\mathsf{NI\underline{B}BS}}$, defined as follows:

[2] LaBRADOR can be made zero-knowledge with a small overhead (cf. [42]).

Language $\mathcal{L}^{\mathsf{NIBBS}}_{\mathsf{rOM\text{-}ISIS}}$

Instance: Each instance $\mathbb{x}$ is interpreted as matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{Y}_0$ and $\mathbf{Y}_1$, a PKE public key pk_{E}, a message vector m and ciphertext ψ.
Witness: Witness $\mathbb{w}$ consists of secret vectors $\mathbf{k}, \mathbf{u}, \mathbf{s}, \mathbf{c}$ and $\mathbf{i}$, and PKE randomness ρ.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\mathbf{A} \cdot \mathbf{s} + \mathbf{B} \cdot \mathbf{c} = \mathbf{C} \cdot \mathbf{k} + \mathsf{H}(\mathbf{u})$$

$$\wedge\, \mathsf{m} = \mathbf{Y}_1 \cdot \left(\mathsf{Vec}^{-1}(\mathbf{k}) \cdot \mathbf{G}^{-1} \left(\mathbf{Y}_0 \cdot \begin{bmatrix} \mathbf{c} \\ \mathbf{i} \end{bmatrix} \mod 3 \right) \mod 2 \right) \mod 3$$

$$\wedge\, \psi = \mathrm{E}.\mathsf{Enc}\left(\mathsf{pk}_{\mathrm{E}}, \mathbf{k} \| \mathbf{u} \| \mathbf{s} \| \mathbf{c} \| \mathbf{i}\, ;\, \rho\right) \wedge \|\mathbf{s}\| \leqslant \mathfrak{B} \wedge \mathbf{k} \in \mathbb{Z}_2^{n' m_N} \wedge \mathbf{c} \in \mathbb{Z}_2^{m} \wedge \mathbf{i} \in \mathbb{Z}_2^{m_N - m'}$$

Concretely, the NIBS construction is defined by the following algorithms:

- $\mathsf{Setup}(1^\lambda, N) \to \mathsf{pp}$. The public parameter setup algorithm computes $n, m, n', m', \mathfrak{s}, q$ from the security parameter λ. It then sets $m_N \coloneqq m' + \lceil \log_2(N) \rceil$, and samples matrices $\mathbf{Y}_0 \leftarrow\!\$\ \mathbb{Z}_3^{\frac{m_N}{2} \times (m + m_N - m')}$, $\mathbf{Y}_1 \leftarrow\!\$\ \mathbb{Z}_3^{n \times n'}$ and $\mathbf{C} \leftarrow\!\$\ \mathbb{Z}_q^{n \times n' m_N}$. It then runs the key generation algorithm of E and generates the corresponding public and secret key pair as

  $$(\mathsf{pk}_{\mathrm{E}}, \mathsf{sk}_{\mathrm{E}}) \leftarrow\!\$\ \mathrm{E}.\mathsf{KeyGen}(1^\lambda)\ .$$

  Next, it generates $\mathsf{crs}_\Pi \leftarrow\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$ and outputs

  $$\mathsf{pp} \coloneqq \left(1^\lambda, n, m, n', m_N, \mathfrak{s}, q, N, \mathsf{pk}_{\mathrm{E}}, \mathsf{crs}_\Pi, \mathbf{Y}_0, \mathbf{Y}_1, \mathbf{C}\right)\ .$$

  The public parameters are fixed once and for all.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$. The signer's key generation algorithm runs the lattice trapdoor setup to obtain

  $$(\mathbf{T}_{\mathbf{A}}, \mathbf{A}) \leftarrow\!\$\ \mathcal{T}.\mathsf{TrapGen}(1^\lambda, n, m, q)\ .$$

It also samples a matrix $\mathbf{B} \leftarrow\$ \mathbb{Z}_q^{n \times m}$, then outputs the signer's secret signing key $\mathsf{sk} \coloneqq \mathbf{T_A}$ and verification key $\mathsf{vk} \coloneqq (\mathbf{A}, \mathbf{B})$.

- $\mathsf{KeyGen_U}(\mathsf{pp}) \to (\mathsf{sk_U}, \mathsf{pk_U})$. The user's key generation algorithm samples the PRF key $\mathbf{K} \leftarrow\$ \mathbb{Z}_2^{n' \times m_N}$ and a random vector $\mathbf{u} \leftarrow\$ \mathbb{Z}_2^{\lambda}$. It then vectorizes $\mathbf{K}$ as $\mathbf{k} \coloneqq \mathsf{Vec}(\mathbf{K})$, and computes

$$\mathbf{t} \coloneqq \mathbf{C} \cdot \mathbf{k} + \mathsf{H}(\mathbf{u}) \; .$$

  Finally, it sets the user's secret key as $\mathsf{sk_U} \coloneqq (\mathbf{K}, \mathbf{u})$ and the public key as $\mathsf{pk_U} \coloneqq \mathbf{t}$, and outputs them.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk_U}, \mathsf{c}) \to \bar{\sigma}$. Given the user's public key $\mathsf{pk_U} =: \mathbf{t}$, the signer's pre-signature issuance algorithm uses the secret signing key $\mathsf{sk} =: \mathbf{T_A}$ and the context vector $\mathsf{c} =: \mathbf{c} \in \mathbb{Z}_2^m$ to compute

$$\mathbf{s} \leftarrow\$ \mathcal{T}.\mathsf{SamplePre}\left(\mathbf{A}, \mathbf{T_A}, \mathbf{t} - \mathbf{B} \cdot \mathbf{c}, \mathfrak{s}\right) ,$$

  and outputs the pre-signature, $\bar{\sigma} \coloneqq \mathbf{s}$.

- $\mathsf{Obtain}(\mathsf{sk_U}, \mathsf{vk}, \bar{\sigma}, \mathsf{c}) \to [\![\mathsf{m}, \sigma]\!]$. The user's signature obtainment algorithm first parses $\mathsf{sk_U}$ as $(\mathbf{K}, \mathbf{u})$ and then, for $\mathsf{vk} =: (\mathbf{A}, \mathbf{B})$, $\bar{\sigma} =: \mathbf{s}$, and $\mathsf{c} =: \mathbf{c}$, it checks if

$$\mathbf{A} \cdot \mathbf{s} + \mathbf{B} \cdot \mathbf{c} \stackrel{?}{=} \mathbf{C} \cdot \mathsf{Vec}(\mathbf{K}) + \mathsf{H}(\mathbf{u}) \;\wedge\; \|\mathbf{s}\| \stackrel{?}{\leqslant} \mathfrak{B} \;\wedge\; \mathbf{c} \stackrel{?}{\in} \mathbb{Z}_2^m \; .$$

  If any check fails it aborts and outputs $\perp$. Otherwise, for each $\mathsf{i} \in [N]$, it computes the message $\mathsf{m_i}$ as

$$\mathsf{m_i} \coloneqq \mathbf{Y}_1 \cdot \left(\mathbf{K} \cdot \mathbf{G}^{-1} \left(\mathbf{Y}_0 \cdot \begin{bmatrix} \mathbf{c} \\ \mathbf{g}^{-1}(\mathsf{i}) \end{bmatrix} \mod 3 \right) \mod 2 \right) \mod 3 \, ,$$

where, $\mathbf{g}^{-1} : \mathbb{Z}_N \to \mathbb{Z}_2^{\log_2(N)}$ ($\log_2(N) = m_N - m'$) is the binary decomposition gadget.

With this, it now generates the corresponding ciphertext as

$$\psi_{\mathsf{i}} \leftarrow \mathrm{E.Enc}\left(\mathsf{pk}_{\mathrm{E}}, \mathbf{k}\|\mathbf{u}\|\mathbf{s}\|\mathbf{c}\|\mathbf{g}^{-1}(\mathsf{i}) \; ; \; \rho_{\mathsf{i}}\right)$$

from uniformly sampled randomness ρ_{i}. It then generates the NIZK proof

$$\pi_{\mathsf{i}} \leftarrow\!\!\$\; \Pi.\mathsf{Prove}\left(\begin{array}{c} \mathsf{crs}_{\Pi}, \mathbb{x}_{\mathsf{i}} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{Y}_0, \mathbf{Y}_1, \mathsf{pk}_{\mathrm{E}}, \mathsf{m}_{\mathsf{i}}, \psi_{\mathsf{i}}) \, , \\ \mathbb{w}_{\mathsf{i}} \coloneqq \left(\mathbf{k}, \mathbf{u}, \mathbf{s}, \mathbf{c}, \mathbf{g}^{-1}(\mathsf{i}), \rho_{\mathsf{i}}\right) \end{array}\right) ,$$

Finally, it outputs $[\![\mathsf{m}, \sigma]\!] \coloneqq \{(\mathsf{m}_1, \sigma_1 \coloneqq (\psi_1, \pi_1)), \ldots, (\mathsf{m}_N, \sigma_N \coloneqq (\psi_N, \pi_N))\}$.

- $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma) \to 1/0$. The signature verification algorithm parses σ as (ψ, π) and outputs 1 (accept) if

$$\Pi.\mathsf{Verify}\left(\mathsf{crs}_{\Pi}, \mathbb{x} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{Y}_0, \mathbf{Y}_1, \mathsf{pk}_{\mathrm{E}}, \mathsf{m}, \psi) \, , \pi\right) = 1 \, ,$$

and 0 (reject) otherwise.

We now state the main security theorem for our construction and sketch a high-level proof. The detailed proofs are provided in the full version of [25].

Theorem 4.19 (Security)**.** *Let $N \in \mathbb{N}$ be the batch size, and let parameters $n, m, n', m', \mathfrak{s}, \mathfrak{B} = \mathfrak{s}\sqrt{m}, \mathfrak{B}_N = (\mathfrak{s} + n'm_N)\sqrt{m}$, and positive integer q be functions of the security parameter λ satisfying (4.1) such that the randomized one-more ISIS instance $\mathsf{rOM\text{-}ISIS}_{q,n,m,\mathfrak{s},\mathfrak{B}_N}$ is hard. Construction 4.1 is:*

(i) One-more unforgeable (Def. 4.14) assuming that the NIZK proof system Π is sound.

(ii) Inter- and intra-blind (Def. 4.15, 4.16) assuming that the NIZK proof system Π *is a (multi-theorem) zero-knowledge, the PKE scheme* E *is IND-CPA secure, and the* F *is a secure pseudorandom function.*

Pf. sketch. The proof of one-more unforgeability follows via a reduction to the rOM-ISIS assumption. The reduction sets $\mathbf{C} \coloneqq \mathbf{A}_\top \cdot \mathbf{R}$ for $\mathbf{R}$ a uniformly chosen binary matrix. The fact that this change is indistinguishable to an adversary can be shown via the leftover hash lemma (Corollary 4.2). Then, whenever the adversary makes a query for the hash of some value $\mathbf{u}$, the reduction queries the rOM-ISIS oracle for a syndrome $\mathbf{h}$ and programs the random oracle such that $\mathsf{H}(\mathbf{u}) \coloneqq \mathbf{h}$. When the adversary outputs its (one-more) forgeries, the reduction extracts $(\mathbf{k}_j, \mathbf{u}_j, \mathbf{s}_j, \mathbf{r}_j)$ from each proof π_j. With high probability, the adversary must have queried each $\mathbf{u}_j$ to the random oracle such that $\mathbf{h}_j = \mathsf{H}(\mathbf{u}_j)$ so that each $\mathbf{h}_j$ is in the set $\mathcal{S}$. So that $\mathbf{h}_j = \mathbf{A} \cdot (\mathbf{s}_j - \mathbf{R} \cdot \mathbf{k}_i) + \mathbf{B} \cdot \mathbf{r}_j$. Consequently, at least one of $(\mathbf{s}_i - \mathbf{R} \cdot \mathbf{k}_i)$ is a valid rOM-ISIS solution.

To prove inter-batch blindness, we will have the challenger simulate its NIZK proof, and then compute the encryptions ψ_i's as encryptions of $\mathbf{0}$ for each user. These changes can be shown to be indistinguishable to an adversary by the zero-knowledge of the NIZK proof system, and IND-CPA security of the public key encryption scheme respectively. Next, leveraging the leftover hash lemma (Corollary 4.2), we can have the challenger sample each user public key $\mathbf{t}$ uniformly. Notice, at this stage, that the adversary learns no information about the user and blindness follows. The proof of intra-batch blindness is essentially similar to that of inter-batch blindness. □

4.5.1 Concrete efficiency

In this section, we propose concrete parameters to instantiate our NIBBS protocol and provide rough estimates for the size of the final signature. In choosing our parameters, we aim for 128-bit classical security.

Preimage sampling paramters. We instantiate the trapdoor generation and preimage sampling procedures using NTRU lattices over the Falcon-512 ring $\mathcal{R}_\mathsf{F} \coloneqq \mathcal{R}_{q_\mathsf{F}, d_\mathsf{F}}$, where $q_\mathsf{F} = 12289$ and $d_\mathsf{F} = 512$. . The other parameters are computed as prescribed in [152, §2.6]). Concretely,

$$\mathfrak{s} = \frac{1.17}{\pi}\sqrt{q_{\mathsf{F}} \cdot \log\left(4d_{\mathsf{F}} \cdot \left(1 + \sqrt{\lambda \cdot 2^{64}}\right)/2\right)} \approx 2^{7.37} \text{ and } \mathfrak{B} = 1.1 \cdot \mathfrak{s}\sqrt{2d_{\mathsf{F}}}\,.$$

PRF parameters. We carry over the parameters from [11] for our desired security level so that $n = 82$ and $n', m_N = 256$ (so that $m' = m_N - \lceil \log_2(B) \rceil$), except we set $m = 1024$ for sufficient rOM-ISIS hardness. Note that, since the final message is in $\mathbb{Z}_3^n$, the final message space is sufficiently large ($2^{\Theta(\lambda)}$).

user keys. Let $\mathsf{Coeff} : \mathbb{Z}[X]/(X^d + 1) \to \mathbb{Z}^d$ be the canonical coefficient map, and $\mathsf{Coeff}^{-1} : \mathbb{Z}^d \to \mathbb{Z}[X]/(X^d + 1)$ be its inverse. Then, the PRF key $\mathbf{K} \in \mathbb{Z}_2^{n' \times m_N}$ can be interpreted as $n'm_N/d_{\mathsf{F}} = 128$ polynomials $(k_1, \ldots, k_{128}) = \mathsf{Coeff}^{-1}(\mathsf{Vec}(\mathbf{K}))$. For $u \leftarrow\!\!\$\ \{0,1\}^{256}$, the user's public key $t \in \mathcal{R}_{\mathsf{F}}$ is then given as

$$t \coloneqq \sum_{j=1}^{128} c_j \cdot k_j + \mathsf{H}(u) \mod q_{\mathsf{F}}\,,$$

where $\mathsf{H} : \{0,1\}^{256} \to \mathcal{R}_{\mathsf{F}}$ is a secure hash function such as SHA-3-256 and $c_i \in \mathcal{R}_{\mathsf{F}}$ are sampled during protocol setup.

Pre-signature issuance. Given a user public key $t \in \mathcal{R}_{\mathsf{F}}$, the signer first samples the nonce vector $\mathbf{r} \leftarrow\!\!\$\ \mathbb{Z}_2^m$, and then decomposes it into $m/d = 2$ polynomials $(r_1, r_2) = \mathsf{Coeff}^{-1}(\mathbf{r})$. The signer then uses its NTRU secret trapdoor to sample short polynomials $s_1, s_2 \in \mathcal{R}_{\mathsf{F}}$ with respect to $a \in \mathcal{R}_{\mathsf{F}}$ such that

$$a \cdot s_1 + s_2 + b \cdot r_1 + r_2 = t \mod q_{\mathsf{F}}\,,$$

for $b \in \mathcal{R}_{\mathsf{F}}$ a part of the signer's verification key. The presignature is then s_1 and the nonce is $\mathbf{r}$ (the user can recover s_2 on its own). An important change, following the template of [6] is that we use rejection sampling to keep the ℓ_∞ norm of (s_1, s_2) to be at most $\lceil 4.15 \cdot \mathfrak{s} \rceil = 688$ for compatibility

with the encryption scheme. Using, this bound, we get that the size of the signer's message is at most 0.62 KB for the presignature and an additional 0.12 KB for the nonce.

Blind signature generation. The user encrypts 133 polynomials in $\mathcal{R}_{\mathsf{F}}$—$\{k_j\}_{j\in[128]}, r_1, r_2, s_1$, along with u and $\mathfrak{i}$ (as constant coefficients) using the Regev style public key encryption scheme from [134] over $\mathcal{R}_{q_{\mathrm{E}},d_{\mathsf{F}}}$. Thus, the rank of the plaintext space is 133, which is the main source of inefficiency in our final signature, as the required PKE modulus is $q_{\mathrm{E}} = 210596593 \approx 2^{27.6}$. Consequently, the ciphertext size is 231.53 KB, and can be brought down to around 214.78 KB by dropping lower order bits [159].

Unfortunately, the Falcon parameters do not satisfy the necessary conditions for the use of the modular Johnson-Lindenstrauss projections [42, Lemma 2.2] (also see [1]). So, we must lift the signature verification equation to a larger modulus ring $\mathcal{R}_{q_{\mathsf{L}},d_{\mathsf{F}}}$ for modulus $q_{\mathsf{L}} > q_{\mathsf{F}}$. To simplify our final estimates, we set it to be as close as possible to the original LaBRADOR modulus 2^{32}. Now, we can lift the signature verification check directly to $\mathcal{R}$ (modulo nothing) so that

$$a \cdot s_1 + s_2 + b \cdot r_1 + r_2 + q_{\mathsf{F}}\delta = \sum_{j=1}^{128} c_j \cdot k_j + \mathsf{H}(u) \in \mathcal{R}\,, \tag{4.2}$$

where the modular wraparound $\delta \in \mathcal{R}_{q_{\mathsf{L}},d_{\mathsf{F}}}$ must now be made part of the witness. For appropriately bounded δ, if (4.2) holds over $\mathcal{R}$, it must hold over $\mathcal{R}_{q_{\mathsf{L}},d_{\mathsf{F}}}$.

Further, to be compatible with the degree of the polynomial-ring in LaBRADOR, i.e., $d_{\mathsf{L}} = 64$, we use a reduced degree polynomial-ring. Note that the constraints over the larger ring $\mathcal{R}_{\mathsf{F}}$ can be reformulated as constraints over any sub-ring of degree d/c for some constant c. Specifically, the encryption scheme is instantiated over $\mathcal{R}_{q_{\mathsf{F}},d_{\mathsf{L}}}$ where $(\mathcal{R}^{d_{\mathsf{F}}/d_{\mathsf{L}}}_{q_{\mathsf{F}},d_{\mathsf{L}}} =)\ \mathcal{R}^{8}_{q_{\mathsf{F}},d_{\mathsf{L}}} \cong \mathcal{R}_{\mathsf{F}}$ and this bijection preserves the norms [133].

The user must prove knowledge of the polynomials satisfying the relation $\mathcal{R}^{\mathsf{NI\underline{B}BS}}$ with respect to the instantiation discussed in this section. In total, we approximate that the overall relation involves $\approx 2^{19.5}$ R1CS constraints (Table 4.3), which gives a proof size of 58 KB using LaBRADOR (cf. [42,

§ 7.1]). However, this only accounts for the SNARK without the additional zero-knowledge property. The LaBRADOR paper gives limited guidance on how to make the protocol zero-knowledge, but the main idea is to deploy a NIZK proof system at an intermediate level of the LaBRADOR recursion (so that the witness is already quite small). This involves some overhead required for blinding the outer commitments up to the NIZK level, masking the Johnson-Lindenstrauss projection and some additional garbage polynomials. In particular, for a binary-R1CS with 2^{20} constraints, using the lattice-based NIZK proof system for linear relations [132], the overhead is about 11 KB (cf. [42, § 6]). Since a single R1CS constraint can be expressed in no more than $\log_2^2(q_{\mathsf{L}}) \approx 2^{10}$ constraints, and given the slow order of growth of the LaBRADOR proofs, a very rough estimate puts the overhead for 2^{20} (general) R1CS constraints at about 36 KB, although with the right instantiation, we expect the actual overhead to be a few kilobytes lower. Nevertheless, with full zero-knowledge, the signature size is $58 + 36 + 214.78 = 308.78$ KB.

Component	**#Constraints**
Hash computation (SHA-3-256)	27000
Signature verification [152]	68609
PRF computation [45]	286000
PKE computation [134]	273408
ℓ_2-norm bounds	73019
Binary coefficient checks	66560
Total	**794596**

Table 4.3: Approximate R1CS constraint count.

We sketch out our calculations of the number of constraints in the following section. These estimates are somewhat conservative, and further optimizations and reductions in the proof size may be possible. In particular, using SNARK-friendly hash functions [13] can also significantly reduce the number of constraints, leading to modest savings in the proof size.

Remark 4.2 (Instantiation of lattice-based NIBS). Our scheme can also readily be used as a standard (non-batched) NIBS. The main distinction is the absence of the index $\mathfrak{i}$ which allows for a few minor optimizations. In particular, the signer can now precompute $\mathbf{G}^{-1}\left(\mathbf{Y}_0 \cdot \mathbf{r} \mod 3\right)$ and send

the resulting $\tilde{\mathbf{r}} \in \mathbb{Z}_3^{256}$. This increases the size of the signer's message by just 0.06 KB. On the other hand, it reduces the user's computation time, as well as the final signature size by 2.6 KB (1.6 KB ciphertext and 1 KB proof) to 306.18 KB after compression. Furthermore, and as we show in the next section, one can entirely do away with the hash computation if one is willing to rely on the recently proposed ISIS_f assumption [48].

4.5.2 Estimating R1CS constraint count

In this section, we sketch out our computation of the number of R1CS constraints shown in Table 4.3.

Hash computation. Standard SHA-3-256 circuits require extensive bit-level constraints. According to zk-SNARK benchmarks, SHA-3-256 for a short input (like 128 bits) costs $\sim 27,000$ constraints.[3]

Signature verification. Recall that the relation we are interested in is

$$a \cdot s_1 + s_2 + b \cdot r_1 + r_2 + q_{\mathsf{F}}\delta = \sum_{j=1}^{128} c_j \cdot k_j + \mathsf{H}(u) \ ,$$

where $a, b, c_1, \ldots, c_{128}$ are public polynomials and $s_1, s_2, r_1, r_2, d, k_1, \ldots, k_{128}, u$ are secret witness polynomials. Now, for each multiplication of public and secret polynomials, we need one R1CS constraint per coefficient. Thus, for $a \cdot s_1$, s_2, $b \cdot r_1$, r_2, and $q_F \cdot d$, each contributes 512 constraints, resulting in a total of 2,560 constraints for the left-hand side. On the right-hand side, each term $c_j \cdot k_j$ requires 512 constraints for the multiplication of their coefficients, resulting in 65536 constraints for the entire sum. Additionally, u, being a secret polynomial, contributes 512 constraints. Together, the right-hand side contributes 66048 constraints. Finally, we need one additional R1CS constraint to enforce the equality between the left-hand side and the right-hand side of the equation. Adding all of these constraints together, we get a total of 68609 R1CS constraints.

[3] https://tinyurl.com/eth-research-sha-const

PRF computation. To compute the R1CS constraints for the given check, we will break down the individual operations. First, we compute $\mathbf{Y}_0 \cdot \begin{bmatrix} \mathbf{r} \\ \mathbf{g}^{-1}(\mathfrak{i}) \end{bmatrix}$, where $\mathbf{Y}_0 \in \mathbb{Z}_3^{128\times(1024+\kappa_N)}$ for $\kappa_N = \lceil \log_2(N) \rceil$ is a public matrix and $\begin{bmatrix} \mathbf{r} \\ \mathbf{g}^{-1}(\mathfrak{i}) \end{bmatrix} \in \mathbb{Z}_2^{1024+\kappa_N}$ is part of the witness. The multiplication requires $(1024 + \kappa_N)$ operations, resulting in as many constraints. Next, the inverse gadget decomposition $\mathbf{G}^{-1}$ maps the resulting vector from $\mathbb{Z}_3^{128} \to \mathbb{Z}_2^{256}$, by expanding each of the 128 entries of the input vector into its (2-bit) binary representation. Thus, each of the 128 elements introduces two binary variables, resulting in a total of 256 constraints for the $\mathbf{G}^{-1}$ operation. We then apply the vectorization operation Vec^{-1} to $\mathsf{Coeff}(k_1, \ldots, k_{128})$. The Vec^{-1} operation transforms a vector in $\mathbb{Z}_2^{65536}$ into a matrix in $\mathbb{Z}_2^{256\times256}$. This transformation involves matrix multiplication and the Kronecker product of identity matrices, which essentially reshapes the vector into a matrix. Since the result is a 256×256 matrix, and each element in the matrix corresponds to a constraint in the R1CS, the operation introduces 65536 constraints. The resulting matrix and vector are then multiplied to give a new vector in $\mathbb{Z}_2^{256}$. For each element of the resulting vector, the matrix-vector multiplication involves a dot product of a row of the 256×256 matrix with the 256-dimensional vector. Each of these dot products requires 256 scalar multiplications, followed by a summation. Since we have a total of 256 elements in the resulting vector, the operations for this matrix-vector multiplication contributes to $256 \cdot 256 = 65536$ constraints. Finally, we multiply the result with $\mathbf{Y}_1 \in \mathbb{Z}_3^{82\times256}$. The total number of operations for this matrix-vector multiplication is $82 \cdot 256 = 20992$ and as many constraints. Summing all of these contributions, the total number of R1CS constraints is $283392 + 128 \cdot \kappa_N$. For $N \leqslant 2^{20}$, this value is bounded from above by 286000.

PKE computation. The proof of computation of the encryption of $\mathbf{m} \in \mathcal{R}_{q_\mathrm{E},d_\mathsf{F}}^{133}$ according to [134] requires proving

$$\psi_1 = a_1 \cdot s + e_1 \text{ and } \boldsymbol{\psi}_2 = \mathbf{a}_2 \cdot s + \mathbf{e}_2 + p \cdot \mathbf{m} \ ,$$

for ciphertexts $(\psi_1, \boldsymbol{\psi}_2) \in \mathcal{R}_{q_\mathrm{E},d_\mathsf{F}} \times \mathcal{R}^{133}_{q_\mathrm{E},d_\mathsf{F}}$, public polynomial $a_1 \in \mathcal{R}_{q_\mathrm{E},d_\mathsf{F}}$ and polynomial vector $\mathbf{a}_2 \in \mathcal{R}^{133}_{q_\mathrm{E},d_\mathsf{F}}$ and witness $(s, e_1, \mathbf{e}_2, \mathbf{m}) \in \mathcal{R}_{q_\mathrm{E},d_\mathsf{F}} \times \mathcal{R}_{q_\mathrm{E},d_\mathsf{F}} \times \mathcal{R}^{133}_{q_\mathrm{E},d_\mathsf{F}} \mathcal{R}^{133}_{q_\mathrm{E},d_\mathsf{F}}$.

To enforce the first relation, we need to compute multiply a public polynomial with a secret polynomial. This requires 512 R1CS constraints, one per coefficient; and the addition of a secret polynomial to this product involves another 512 constraints. For the second relation, we notice that each polynomial-vector multiplication requires 512 constraints for each of the 133 entries in the vector, leading to $133 \cdot 512 = 68096$ constraints, and the scalar multiplication also contributes 68096 constraints. Finally, adding up the three vectors contributes $2 \cdot 68096 = 136192$ constraints. The total number of R1CS constraints required to enforce both equations is therefore $1024 + 4 \cdot 68096 = 273408$.

ℓ_2-norm bound. For any polynomial, computing the square of each coefficient requires 512 multiplication constraints, summing these squares adds 511 constraints, and enforcing that the resulting sum is less than or equal to some (public) bound requires approximately 64 constraints via bit-decomposition and range checking. Therefore, each polynomial contributes roughly $512 + 511 + 64 = 1087$ R1CS constraints, and since we need to perform this check for polynomials s_1, s_2, δ, s and e_1, and polynomial vector $\mathbf{e}_2$, the total number of constraints is $4 \cdot 1087 + 133 \cdot 512 + 511 + 64 = 73019$.

Binary coefficient checks. Checking that the coefficients of polynomials $k_1, k_2, \ldots, k_{128}, r_1$ and r_2 are binary, can be done by ensuring that for each coefficient x, the equation $x \cdot (1 - x) = 0$ holds, which requires one R1CS constraint per coefficient. Since each of the polynomials has 512 coefficients, the total number of R1CS constraints is $512 \cdot 130 = 66560$.

4.6 Lattice-based NIBBS from ISIS_f

A minor drawback of Construction 4.1 is that it requires proving the computation of the hash function in the NIZK proof. Aside from the computational overhead, this additionally necessitates a somewhat strong assumption, since the proof of unforgeability also requires modeling the hash function as a random oracle. While not unusual in practice (see for example [41]), this approach may be unsatisfactory for provable security. In this section, we give an alternate construction that is secure under the recently proposed ISIS_f assumption [48] which does not require this stronger assumption concerning the random oracle. We begin by recalling the ISIS_f assumption [48]:

Definition 4.20 (ISIS_f [48])**.** Let $q, n, m, \mathfrak{s}, \mathfrak{B}$ be functions of security parameter λ. Also let $f : \mathcal{D} \to \mathbb{Z}_q^n$ be a function on some input domain $\mathcal{D}$ and for $\mathbf{A} \leftarrow\!\$\ \mathbb{Z}_q^{n\times m}$, let $\mathcal{O}_{\mathbf{A},f}$ be an oracle that chooses a random input $\mathsf{c} \in \mathcal{D}$ and outputs $(\mathbf{s}, \mathsf{c})$ for $\mathbf{s} \in \mathbb{Z}_q^m$ such that $\mathbf{s} \leftarrow\!\$\ \mathbf{A}_{\mathfrak{s}}^{-1}(f(\mathsf{c}))$. Given $(\mathbf{A}, f)$ and access to $\mathcal{O}_{\mathbf{A},f}$, the ISIS_f problem asks to find $(\mathbf{s}', \mathsf{c}') \in \mathbb{Z}_q^m \times \mathcal{D}$, such that $\mathbf{A}\cdot\mathbf{s}' = f(\mathsf{c}')$, $\mathbf{s}' \neq \mathbf{s}$ and $\|\mathbf{s}'\| \leqslant \mathfrak{B}$.

Construction 4.2. Let $N \in \mathbb{N}$ be the batch size, and let parameters $n, m, n', m', \mathfrak{s}, \mathfrak{B} = \mathfrak{s}\sqrt{m}$, and positive integer q be functions of the security parameter λ satisfying the constraints in (4.1). By $f : [2^m] \to \mathbb{Z}_q^n$ we denote a linear map characterizing the ISIS_f. We give a $\mathsf{NI\underline{B}BS}$ construction that is secure under the ISIS_f assumption.

Our construction relies on a lattice trapdoor $\mathcal{T} = (\mathcal{T}.\mathsf{TrapGen}, \mathcal{T}.\mathsf{SamplePre})$, a public key encryption scheme $\mathrm{E} = (\mathrm{E}.\mathsf{KeyGen}, \mathrm{E}.\mathsf{Enc}, \mathrm{E}.\mathsf{Dec})$ and NIZK proof systems $\Pi_1 = (\Pi_1.\mathsf{Setup}, \Pi_1.\mathsf{Prove}, \Pi_1.\mathsf{Verify})$, $\Pi_2 = (\Pi_2.\mathsf{Setup}, \Pi_2.\mathsf{Prove}, \Pi_2.\mathsf{Verify})$ respectively for languages $\mathcal{L}_{\mathsf{ISIS}_f}^{\mathsf{NI\underline{B}BS}_1}$ and $\mathcal{L}_{\mathsf{ISIS}_f}^{\mathsf{NI\underline{B}BS}_2}$ as defined.

Concretely, the NIBS construction is defined by the following algorithms:

- $\mathsf{Setup}(1^\lambda, N) \to \mathsf{pp}$. The public parameter setup algorithm computes $n, m, n', m', \mathfrak{s}, q$ from the security parameter λ as well as the linear map f. It then sets $m_N := m' + \lceil \log_2(N) \rceil$, and samples matrices $\mathbf{Y}_0 \leftarrow\!\$\ \mathbb{Z}_3^{\frac{m_N}{2}\times(m+m_N-m')}$, $\mathbf{Y}_1 \leftarrow\!\$\ \mathbb{Z}_3^{n\times n'}$ and $\mathbf{C}_0, \mathbf{C}_1 \leftarrow\!\$\ \mathbb{Z}_q^{n\times n' m_N}$. It

Language $\mathcal{L}^{\mathsf{NIBBS}_1}_{\mathsf{ISIS}_f}$

Instance: Each instance $\mathbb{x}$ is interpreted as polynomial matrices $\mathbf{C}_0$ and $\mathbf{C}_1$, an encryption public key pk_{E}, a polynomial vector $\mathbf{t}$ and a ciphertext ϕ.
Witness: Witness $\mathbb{w}$ consists of secret polynomial vectors $\mathbf{k}$ and $\mathbf{u}$, and the encryption randomness ν.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\phi = \mathrm{E.Enc}(\mathsf{pk}_{\mathrm{E}}, \mathbf{k} \| \mathbf{u}; \nu) \ \wedge \ \mathbf{t} = \mathbf{C}_0 \cdot \mathbf{k} + \mathbf{C}_1 \cdot \mathbf{u} \ \wedge \ \mathbf{k}, \mathbf{u} \in \mathbb{Z}_2^{n' m_N}$$

Language $\mathcal{L}^{\mathsf{NIBBS}_2}_{\mathsf{ISIS}_f}$

Instance: Each instance $\mathbb{x}$ is interpreted as matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}_0, \mathbf{C}_1, \mathbf{Y}_0$ and $\mathbf{Y}_1$, a linear map f, a PKE public key pk_{E}, a message vector m and ciphertext ψ.
Witness: Witness $\mathbb{w}$ consists of secret vectors $\mathbf{k}, \mathbf{u}, \mathbf{s}$ and $\mathbf{i}$, integer c and PKE randomness ρ.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\mathbf{A} \cdot \mathbf{s} = \mathbf{C}_0 \cdot \mathbf{k} + \mathbf{C}_1 \cdot \mathbf{u} + f(\mathsf{c}) \wedge$$

$$\mathsf{m} = \mathbf{Y}_1 \cdot \left(\mathsf{Vec}^{-1}(\mathbf{k}) \cdot \mathbf{G}^{-1} \left(\mathbf{Y}_0 \cdot \begin{bmatrix} f(\mathsf{c}) \\ \mathbf{i} \end{bmatrix} \mod 3 \right) \mod 2 \right) \mod 3$$

$$\psi = \mathrm{E.Enc}\left(\mathsf{pk}_{\mathrm{E}}, \mathbf{k} \| \mathbf{u} \| \mathbf{s} \| \mathbf{i} \| \mathsf{c} \ ; \ \rho\right) \ \wedge \ \|\mathbf{s}\| \leqslant \mathfrak{B} \ \wedge \ \mathsf{c} \in [2^m] \ \wedge \ \mathbf{k}, \mathbf{u} \in \mathbb{Z}_2^{n' m_N} \ \wedge \ \mathbf{i} \in \mathbb{Z}_2^{m_N - m'}$$

then runs the key generation algorithm of E and generates the corresponding public and secret key pair as

$$(\mathsf{pk}_{\mathrm{E}}, \mathsf{sk}_{\mathrm{E}}) \leftarrow\!\!\$ \ \mathrm{E.KeyGen}(1^\lambda) \ .$$

Next, it generates $\mathsf{crs}_{\Pi_1} \leftarrow\!\!\$ \ \Pi_1.\mathsf{Setup}(1^\lambda)$ and $\mathsf{crs}_{\Pi_2} \leftarrow\!\!\$ \ \Pi_2.\mathsf{Setup}(1^\lambda)$ and outputs

$$\mathsf{pp} \coloneqq \left(1^\lambda, n, m, n', m_N, \mathfrak{s}, q, N, f, \mathsf{pk}_{\mathrm{E}}, \mathsf{crs}_{\Pi_1}, \mathsf{crs}_{\Pi_2}, \mathbf{Y}_0, \mathbf{Y}_1, \mathbf{C}_0, \mathbf{C}_1\right) \ .$$

The public parameters are fixed once and for all.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$. The signer's key generation algorithm runs the lattice trapdoor setup to obtain

$$(\mathbf{T}_{\mathbf{A}}, \mathbf{A}) \leftarrow\!\$\, \mathcal{T}.\mathsf{TrapGen}(1^{\lambda}, n, m, q) \; .$$

It also samples a matrix $\mathbf{B} \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}$, then outputs the signer's secret signing key $\mathsf{sk} \coloneqq \mathbf{T}_{\mathbf{A}}$ and verification key $\mathsf{vk} \coloneqq (\mathbf{A}, \mathbf{B})$.

- $\mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \to (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. The user's key generation algorithm samples the PRF key $\mathbf{K} \leftarrow\!\$\, \mathbb{Z}_2^{n' \times m_N}$ and a random vector $\mathbf{u} \leftarrow\!\$\, \mathbb{Z}_2^{n' m_N}$. It then vectorizes $\mathbf{K}$ as $\mathbf{k} \coloneqq \mathsf{Vec}(\mathbf{K})$, and computes an Ajtai commitment

$$\mathbf{t} \coloneqq \mathbf{C}_0 \cdot \mathbf{k} + \mathbf{C}_1 \cdot \mathbf{u} \; .$$

Next, it generates a ciphertext,

$$\phi \leftarrow \mathrm{E}.\mathsf{Enc}\left(\mathsf{pk}_{\mathrm{E}}, \mathbf{k} \| \mathbf{u} \; ; \; \nu\right)$$

from uniformly sampled randomness ν and the NIZK proof

$$\tau \leftarrow\!\$\, \Pi_1.\mathsf{Prove}\left(\mathsf{crs}_{\Pi_1}, \mathbb{x} \coloneqq (\mathbf{C}_0, \mathbf{C}_1, \mathsf{pk}_{\mathrm{E}}, \mathbf{t}, \phi), \mathbb{w} \coloneqq (\mathbf{k}, \mathbf{u}, \nu)\right) \; ,$$

It then sets the user's secret key as $\mathsf{sk}_{\mathsf{U}} \coloneqq (\mathbf{K}, \mathbf{u})$, the public key as $\mathsf{pk}_{\mathsf{U}} \coloneqq (\mathbf{t}, \phi, \tau)$ and outputs them.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}) \to \bar{\sigma}$. Given the user's public key $\mathsf{pk}_{\mathsf{U}} =: (\mathbf{t}, \phi, \tau)$, the signer's pre-signature issuance algorithm first checks if

$$\Pi_1.\mathsf{Verify}(\mathsf{crs}_{\Pi_1}, \mathbb{x} \coloneqq (\mathbf{C}_0, \mathbf{C}_1, \mathsf{pk}_{\mathrm{E}}, \mathbf{t}, \phi), \tau) \stackrel{?}{=} 1 \; ,$$

otherwise it aborts and outputs $\perp$. It then uses the secret signing key $\mathsf{sk} =: \mathbf{T}_{\mathbf{A}}$ and the context $\mathsf{c} \in [2^m]$ it computes

$$\mathbf{s} \leftarrow\!\$\, \mathcal{T}.\mathsf{SamplePre}\left(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{t} + f(\mathsf{c}), \mathfrak{s}\right),$$

and outputs the pre-signature, $\bar{\sigma} \coloneqq \mathbf{s}$.

- $\mathsf{Obtain}(\mathsf{sk}_{\mathsf{U}}, \mathsf{vk}, \bar{\sigma}, \mathsf{c}) \to [\![\mathsf{m}, \sigma]\!]$. The user's signature obtainment algorithm first parses sk_{U} as $(\mathbf{K}, \mathbf{u})$ and then, for $\mathsf{vk} =: (\mathbf{A}, \mathbf{B})$, $\bar{\sigma} =: \mathbf{s}$ it checks if

$$\mathbf{A} \cdot \mathbf{s} \stackrel{?}{=} \mathbf{C}_0 \cdot \mathsf{Vec}(\mathbf{K}) + \mathbf{C}_1 \cdot \mathbf{u} + f(\mathsf{c}) \ \wedge \ \|\mathbf{s}\| \stackrel{?}{\leqslant} \mathfrak{B} \ \wedge \ \mathsf{c} \stackrel{?}{\in} [2^m].$$

If any check fails it aborts and outputs $\perp$. Otherwise, for each $\mathsf{i} \in [N]$, it computes the message m_{i} as

$$\mathsf{m}_{\mathsf{i}} \coloneqq \mathbf{Y}_1 \cdot \left(\mathbf{K} \cdot \mathbf{G}^{-1} \left(\mathbf{Y}_0 \cdot \begin{bmatrix} f(\mathsf{c}) \\ \mathbf{g}^{-1}(\mathsf{i}) \end{bmatrix} \mod 3 \right) \mod 2 \right) \mod 3,$$

where, $\mathbf{g}^{-1} : \mathbb{Z}_N \to \mathbb{Z}_2^{\log_2(N)}$ ($\log_2(N) = m_N - m'$) is the binary decomposition gadget. With this, it now generates the corresponding ciphertext as

$$\psi_{\mathsf{i}} \leftarrow \mathrm{E}.\mathsf{Enc}\left(\mathsf{pk}_{\mathrm{E}}, \mathbf{k} \| \mathbf{u} \| \mathbf{s} \| \mathbf{g}^{-1}(\mathsf{i}) \| \mathsf{c}\ ;\ \rho_{\mathsf{i}}\right)$$

from uniformly sampled randomness ρ_{i}. It then generates the NIZK proof

$$\pi_{\mathsf{i}} \leftarrow\!\$\, \Pi_2.\mathsf{Prove}\left(\begin{array}{c} \mathsf{crs}_{\Pi_2}, \mathbb{x}_{\mathsf{i}} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{C}_0, \mathbf{C}_1, \mathbf{Y}_0, \mathbf{Y}_1, f, \mathsf{pk}_{\mathrm{E}}, \mathsf{m}_{\mathsf{i}}, \psi_{\mathsf{i}}), \\ \mathbb{w}_{\mathsf{i}} \coloneqq \left(\mathbf{k}, \mathbf{u}, \mathbf{s}, \mathbf{g}^{-1}(\mathsf{i}), \mathsf{c}, \rho_{\mathsf{i}}\right) \end{array} \right),$$

Finally, it outputs $[\![\mathsf{m}, \sigma]\!] \coloneqq \{(\mathsf{m}_1, \sigma_1 \coloneqq (\psi_1, \pi_1)), \ldots, (\mathsf{m}_N, \sigma_N \coloneqq (\psi_N, \pi_N))\}$.

- $\mathsf{Verify}(\mathsf{vk}, \mathsf{m}, \sigma) \to 1/0$. The signature verification algorithm parses σ as (ψ, π) and outputs 1 (accept) if

$$\Pi_2.\mathsf{Verify}\left(\mathsf{crs}_{\Pi_2}, \mathbb{x} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{C}_0, \mathbf{C}_1, \mathbf{Y}_0, \mathbf{Y}_1, f, \mathsf{pk}_{\mathrm{E}}, \mathsf{m}_{\mathrm{i}}, \psi_{\mathrm{i}}), \pi\right) = 1 \ ,$$

and 0 (reject) otherwise.

We now state the main security theorem for our construction. The detailed proofs are provided in the full version of [25].

Theorem 4.21 (Security)**.** *Let $N \in \mathbb{N}$ be the batch size, and let parameters $n, m, n', m', \mathfrak{s}, \mathfrak{B} = \mathfrak{s}\sqrt{m}$, and positive integer q be functions of the security parameter λ such that (4.1) is satisfied, and the ISIS_f and $\mathsf{SIS}_{q,n,2n'm_N,\sqrt{2n'm_N}}$ problems are hard for an appropriately chosen function $f : [2^m] \to \mathbb{Z}_q^n$. Construction 4.2 is:*

(i) One-more unforgeable (Def. 4.14) assuming that the NIZK proof systems Π_1 and Π_2 are sound.

(ii) Inter- and intra-blind (Def. 4.15, 4.16) assuming that the NIZK proof systems Π_1 and Π_2 are (multi-theorem) zero-knowledge, the PKE scheme E is IND-CPA secure, and the F is a secure pseudorandom function.

Chapter 5: Non-interactive Threshold Blind Signatures

In privacy-sensitive applications such as digital cash, ticketing, and electronic voting, centralized issuance introduces a single point of failure. If a signer is compromised, unauthorized signatures may be issued, compromising system integrity. If the signer experiences outages or denial-of-service attacks, issuance ceases and recipients are blocked. These vulnerabilities arise from concentrating issuer responsibilities on a single authority.

A natural approach to mitigate these risks is to employ *threshold* issuance, distributing the signer's operations across multiple parties. In a threshold blind signature scheme, the signing key is secret-shared among n parties through distributed key generation, and recipients can interact with any t of them to obtain a valid signature. Unlike standard threshold signatures, the signers signers must learn nothing about the message being signed, and the resulting signature must retain unlinkability to the issuance transcript. These schemes are appealing in principle, yet relatively few constructions exist [67, 124, 166]. This scarcity likely reflects the difficulty of simultaneously (i) binding partial signatures to a common signing instance, (ii) providing robustness against misbehaving parties, and (iii) preserving message privacy and unlinkability across concurrent sessions.

While non-interactive blind signatures address the overhead of interaction and threshold schemes address the risk of compromise, no existing construction achieves both properties simultaneously. A *non-interactive threshold blind signature* would permit any t issuers to contribute pre-signature shares that aggregate non-interactively into a valid signature while preserving blindness and unforgeability. This primitive is highly desirable as it combines the efficiency of non-interactivity with the robustness and trust distribution of threshold cryptography.

In this chapter, we show that this goal is achievable. We construct the first threshold non-interactive blind signature scheme with non-interactive pre-signature aggregation. We begin by providing a formal treatment of this primitive, defining the security notions of blindness and one-more

This chapter is based on [27].

unforgeability, and characterizing two hierarchical levels of one-more unforgeability in the threshold setting.

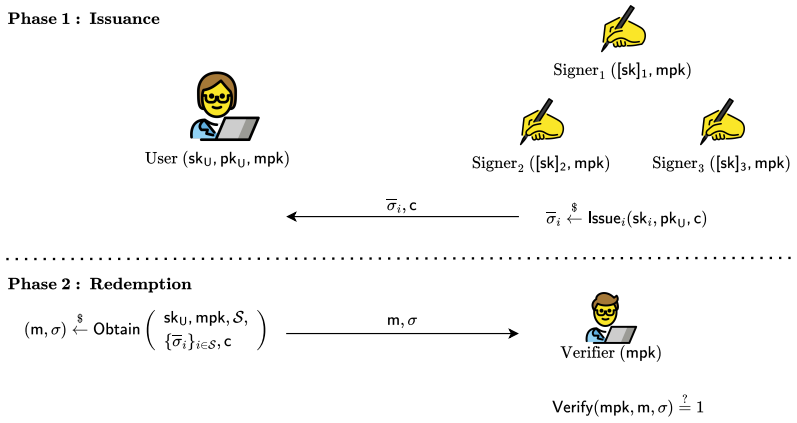

Figure 5.1: A simplified illustration of the non-interactive threshold blind signature protocol. The user engages in issuance with a set of $\mathcal{S}$ signers. If $|\mathcal{S}| \geqslant t$, the user can obtain the blind signature and the corresponding random message.

Following this, we give a design for a practical non-interactive blind signature construction from Pointcheval-Sanders signatures that achieves the smallest-known pre-signature and signature sizes with improved issuance efficiency. We then explain how to exploit the special structure of this NIBS construction to obtain the first full threshold non-interactive blind signature scheme supporting non-interactive aggregation of pre-signatures. The blindness of our scheme reduces to the q-decisional Diffie-Hellman inverse problem, while one-more unforgeability reduces to a threshold variant of the Pointcheval-Sanders assumption, which we show reduces to the one-more co-Discrete Logarithm problem in the algebraic group model. Our scheme tolerates adaptive corruptions of up to $(t-1)$ issuers.

We provide performance estimates for the BLS12-381 curve showing issuance time of 0.6 ms and signature generation in under 5 ms for thresholds up to 5 (with roughly linear growth in t), and 4 ms for verification. On this curve, receiver public keys are approximately 112 bytes, pre-signatures 96 bytes, and final signatures 192 bytes. We also present several extensions: (i) *tagged*

non-interactive threshold blind signatures, enabling inclusion of non-blinded auxiliary data; (ii) non-interactive threshold blind signatures with *verifiable pre-signatures*, allowing recipients to validate pre-signature shares before aggregation; and (iii) *batched* non-interactive threshold blind signatures, enabling a bounded number of final signatures to derive from a single valid pre-signature.

5.1 Prior Work on Threshold Issuance of Blind Signatures

We focus our related work discussion below on threshold and non-interactive blind signatures and summarize some comparisons in Table 5.1.

Threshold blind signatures. The literature on threshold blind signatures is relatively sparse. [166] built on [43] to construct a two-move BLS-style scheme. Kuchta and Manulis [124] proposed a two-move protocol from bilinear pairings, though their first message is large due to two required NIZK proofs. Crites, et al. [67] introduced a five-move protocol in pairing-free groups, obtained by compiling the [163] blind signature scheme and achieving efficient verification. In a recent work, Lehmann, et al. [125] provided the first systematic study of unforgeability notions for threshold blind signatures, showing separations between existing schemes and the strongest definitions, and extending the [67, 166] constructions to achieve stronger guarantees. Coconut [162] is a multi-round protocol for threshold anonymous credentials. Our work has some structural similarities with Coconut as both schemes use PS signatures [148] to build a credential scheme. However the main challenge in our design stems from the non-interactive nature of our issuance.

Other work on NIBS. Hanzlik [101] introduced non-interactive blind signatures, giving a practical construction from signatures on equivalence classes [88, 100], proven secure in the generic group model. Hanzlik, et al. [103] later gave a generic garbled-circuit-based construction supporting pre-signature issuance on RSA keys. Our work [25] (Chapter 4) gave the first practical NIBS from (non-standard) hard lattice assumptions and further enabled derivation of a batch of blind signatures (and random messages) from a single pre-signature.

[1] Satisfies the weakest notion of one-more unforgeability for threshold blind signatures (OM-Unf-0).
[2] Pairing free.

	# moves	Threshold	User key size $\lvert\mathsf{pk}_\mathsf{U}\rvert$	Communication $\mathsf{U} \to \mathsf{S}$	Communication $\mathsf{S} \to \mathsf{U}$	Signature size $\lvert\sigma\rvert$
[166][1]	2	✓	–	$1\mathbb{G} + 1\mathbb{F}$	$1\mathbb{G}$	$1\mathbb{G}$
[67][2]	5	✓	–	$O(t\mathbb{F})$	$2\mathbb{G} + 4\mathbb{F}$	$1\mathbb{G} + 2\mathbb{F}$
[101]	1	✗	$1\mathbb{G}$	–	$2\mathbb{G} + 1\tilde{\mathbb{G}}$	$2\mathbb{G} + 1\tilde{\mathbb{G}}$
Const. 5.1	1	✓	$1\mathbb{G} + 2\mathbb{F}$	–	$2\mathbb{G}$	$2\mathbb{G} + 3\mathbb{F}$

Table 5.1: Comparison of group-based non-interactive and/or threshold blind signature schemes. $\mathsf{U} \to \mathsf{S}$ denotes the communication from the user to the signer (similarly $\mathsf{S} \to \mathsf{U}$). $\mathbb{G}$, $\tilde{\mathbb{G}}$ denote the base groups, $\mathbb{F}$ the finite field elements and t is the threshold. Note that for type-3 pairing groups, the size of $\tilde{\mathbb{G}}$ in bits is at least 4 times bigger than the elements of $\mathbb{F}$.

5.2 Preliminaries

In this section, we give the preliminaries necessary for this chapter. Many of these foundations will be useful in subsequent chapters as well.

Notation. With slight abuse of notation, we use $[x]_i$ to denote the i^{th} secret share of x. The distinction from $[n] = \{1, 2, \ldots, n\}$ will always be clear from context.

5.2.1 Bilinear pairings

Definition 5.1 (Pairing friendly groups)**.** Let $\mathbb{G}$ and $\tilde{\mathbb{G}}$ be two groups of the same prime order p with the generators g and $\tilde{g}$, respectively. Also, let $\mathbb{Z}_p$ be the field of order p. Then, $\mathbb{G}$ and $\tilde{\mathbb{G}}$ are *pairing friendly* if there exists an efficiently computable bilinear pairing, $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \hat{\mathbb{G}}$, satisfying the following properties:

(i) **Bilinearity.** $\forall h \in \mathbb{G}, \tilde{h} \in \tilde{\mathbb{G}},\ x, y \in \mathbb{Z}_p$,

$$e(h^x, \tilde{h})^y = e(h, \tilde{h}^y)^x = e(h, \tilde{h})^{xy} .$$

(ii) **Non-degeneracy.** $e(g, \tilde{g}) \neq 1_{\hat{\mathbb{G}}}$.

Bilinear pairings can be of a few types depending on whether there is an efficiently computable homomorphism from $\mathbb{G}$ to $\tilde{\mathbb{G}}$ in both directions (Type 1), only one direction (Type 2), or in neither

direction (Type 3). Type 3 pairings are the most efficient setting for a relevant security parameter and they are commonly deployed.

5.2.2 Algebraic group model

Definition 5.2 (Algebraic Group Model (AGM) [89])**.** For every group element that an adversary outputs as the challenge or as part of a query, it must output its *representation* in terms of the group elements it has previously been provided by the challenger. For instance, when an adversary, having seen group elements $R_0, R_1, \ldots, R_n$, outputs a group element P, it must also output a vector $(a_0, a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ such that $P = \prod_{i \in [0,n]} R_i^{a_i}$.

Further, we call an adversary satisfying these conditions, an *algebraic* adversary.

5.2.3 Cryptographic hardness assumptions

Definition 5.3 (Co-discrete logarithm problem)**.** Let $(\mathbb{G}, \tilde{\mathbb{G}})$ be pairing friendly groups of order p with a bilinear pairing $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \hat{\mathbb{G}}$. For $x \leftarrow\!\$\ \mathbb{Z}_p$ the co-discrete logarithm problem asks that given $\left(p, g, \tilde{g}, X \coloneqq g^x, \tilde{X} \coloneqq \tilde{g}^x\right)$, find and output x.

Definition 5.4 (One-more co-discrete logarithm problem)**.** Let $(\mathbb{G}, \tilde{\mathbb{G}})$ be pairing friendly groups of order p with a bilinear pairing $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \hat{\mathbb{G}}$. For $x \leftarrow\!\$\ \mathbb{Z}_p$ the one-more co-discrete logarithm problem asks that given $(p, g, \tilde{g})$, oracle $\mathcal{O}$ that outputs tuples $\left(g^{t_i}, \tilde{g}^{t_i}\right)$ and oracle $\mathcal{O}_{\mathsf{dlog}}$ that on input $Y \in \mathbb{G}$ (respectively $\tilde{Y} \in \tilde{\mathbb{G}}$) outputs y such that $Y = g^y$ (respectively $\tilde{Y} = \tilde{g}^y$), find and output distinct $t_1, \ldots, t_{\ell+1}$, where for each $i \in [\ell + 1]$ there exists an output of oracle $\mathcal{O}$ such that $(g^{t_i}, \tilde{g}^{t_i})$, while making at most ℓ queries to $\mathcal{O}_{\mathsf{dlog}}$.

Definition 5.5 (Strong decisional Diffie-Hellman problem (sDDH) [147])**.** Let $\mathbb{G}$ be a group of order p. For $x, y \leftarrow\!\$\ \mathbb{Z}_p$, and $g \in \mathbb{G}$, the decisional Diffie-Hellman problem asks to distinguish, for a randomly chosen $T \in \mathbb{G}$, the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where:

$$\mathcal{D}_0 \coloneqq \left(p, g, g^x, g^{1/x}, g^y, g^{xy}\right) \qquad \mathcal{D}_1 \coloneqq \left(p, g, g^x, g^{1/x}, g^y, T\right).$$

Definition 5.6 (Decisional q-Diffie-Hellman problem (qDDH) [47])**.** Let $\mathbb{G}$ be a group of order p. For $x \leftarrow\!\$\, \mathbb{Z}_p$, and $g \in \mathbb{G}$, the decisional q-Diffie-Hellman inverse problem asks to distinguish, for a randomly chosen $T \in \mathbb{G}$, the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where:

$$\mathcal{D}_0 \coloneqq \left(p, g, g^x, g^{x^2}, \ldots, g^{x^q}, g^{1/x}\right) \qquad \mathcal{D}_1 \coloneqq \left(p, g, g^x, g^{x^2}, \ldots, g^{x^q}, T\right) .$$

Definition 5.7 (PS problem [148, Assumption 2])**.** Let $(\mathbb{G}, \tilde{\mathbb{G}})$ be pairing friendly groups of order p with a bilinear pairing $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \hat{\mathbb{G}}$. For $x, y \leftarrow\!\$\, \mathbb{Z}_p$ the PS problem asks that given $\left(p, g, \tilde{g}, e, \tilde{X} \coloneqq \tilde{g}^x, \tilde{Y} \coloneqq \tilde{g}^y\right)$ and an oracle $\mathcal{O}$ that on input $\mu \in \mathbb{Z}_p$ outputs $(h, h^{x+y\mu})$, find a $\left(\mu^*, h^*, (h^*)^{x+y\mu^*}\right)$ triple such that μ^* is never queried to $\mathcal{O}$ and $h^* \neq 1_{\mathbb{G}}$.

Notably, the PS problem is reducible to the hardness of the co-discrete logarithm problem, as we explain next.

Lemma 5.8. *In the algebraic group model, the PS assumption holds under the co-discrete logarithm assumption.*

Sketch. Complete proofs of the security of the PS assumption in the algebraic group model can be found in [119] and [35]. Here, we sketch the proof idea. Because the adversary is algebraic, it must output a representation for the final forgery (μ^*, h^*, H^*). Thus, we can define $a_x, a_y, a_{\mathsf{R}}, b_x, b_y, b_{\mathsf{R}}$, such that:

$$h^* = g^{a_x \cdot x + a_y \cdot y + a_{\mathsf{R}}} \quad \text{and} \quad H^* = g^{b_x \cdot x + b_y \cdot y + b_{\mathsf{R}}},$$

whereby the validity of the output we know that

$$(a_x \cdot x + a_y \cdot y + a_{\mathsf{R}}) \cdot (x + \mu^* \cdot y) = b_x \cdot x + b_y \cdot y + b_{\mathsf{R}}.$$

The proof then focuses on three distinct cases, where the reduction picks different targets for a co-discrete logarithm instance. It either tries to use the adversary to compute x, where $\tilde{X}$ is set to the

co-discrete logarithm instance. Alternatively, it targets y and appropriately sets $\tilde{Y}$. The last target, and a way for an adversary to break the PS assumption, is to compute the discrete logarithm of h_i to base g, where $(h_i, h_i^{x+y\cdot\mu_i})$ is a response from oracle $\mathcal{O}$. However, note that the reduction can use this to solve the co-discrete logarithm by setting h_i. □

5.2.4 Threshold secret sharing

A (t, n)-threshold secret sharing scheme over $\mathcal{X}$ consists of a tuple of polynomial-time algorithms $(\mathsf{Share}, \mathsf{Combine})$ defined as follows:

- $\mathsf{Share}_{t,n}(x) \to [x]_{i\in[n]}$. Given an element $x \in \mathcal{X}$, the probabilistic secret share generation algorithm outputs a collection of n elements $\{[x]_i \in \mathcal{X} \ : \ \forall i \in [n]\}$ denoted $[x]_{i\in[n]}$.
- $\mathsf{Combine}_{t,n}\left(\mathcal{S}, [x]_{i\in\mathcal{S}}\right) \to x/\perp$. Given a set $\mathcal{S} \subset [n]$ of indices, and a collection of $|\mathcal{S}|$ shares $[x]_i \in \mathcal{X} \ : \ \forall i \in \mathcal{S}$, the deterministic share recombination algorithm outputs x or $\perp$.

Furthermore, a (t, n)-threshold secret sharing scheme must satisfy the following properties:

(i) **Correctness.** For every $1 \leqslant t \leqslant n \in \mathbb{N}$, and any subset $\mathcal{S} \subset [n]$ such that $|\mathcal{S}| = t$,

$$\Pr\left[\mathsf{Combine}_{t,n}\left(\mathcal{S}, [x]_{i\in\mathcal{S}}\right) = x \ : \ [x]_{i\in[n]} \leftarrow \mathsf{Share}_{t,n}(x)\right] = 1\ .$$

(ii) **Robustness.** For every $t \leqslant n \in \mathbb{N}$, and any probabilistic adversary $\mathcal{A}$

$$\Pr\left[\begin{array}{l|r} \mathsf{Combine}_{t,n}\left(\mathcal{S}, [x]_{i\in\mathcal{S}}\right) = \perp & [x]_{i\in[n]} \leftarrow \mathsf{Share}_{t,n}(x) \\ \wedge\ |\mathcal{S}| < t & \mathcal{S} \leftarrow \mathcal{A}\left([x]_{i\in[n]}\right) \end{array}\right] = 1\ .$$

Shamir's secret sharing. Let p be a prime power with $p > n$ and let $\mathbb{Z}_p$ denote the finite field of order p. For integers $1 \leqslant t \leqslant n$ and a secret $s \in \mathbb{Z}_p$ the (t, n)-Shamir secret sharing scheme samples $a_1, \ldots, a_{t-1} \leftarrow\!\!\$\ \mathbb{Z}_p$ uniformly and defines the random polynomial $f(x) = s + \sum_{i=1}^{k-1} a_i x^i \in \mathbb{Z}_p[x]$

so that we can define the i^{th} share $[s]_i := f(i)$. Importantly, any set of t shares determines f uniquely by polynomial interpolation, and in particular the secret is recovered as the constant term $s = f(0)$ via the Lagrange reconstruction formula $s = f(0) = \sum_{i=1}^{t} y_i \mathrm{L}_i$, where $\mathrm{L}_i := \prod_{k \in [t] \setminus \{i\}} \frac{k}{k-i}$ is the i^{th} Lagrange coefficient.

5.3 The Formal NITBS Framework

In this section we formalize non-interactive threshold blind signatures (NITBS) and give the precise security notions tailored to the non-interactive setting. In interactive protocols "non-interactivity" is sometimes taken to mean a single round of communication (user $\rightarrow$ signers $\rightarrow$ user) [125, 145]. Here we mean *strict* non-interactivity, i.e., each signer produces a single pre-signature share that the user finalizes locally. This single-message structure simplifies the security model compared to recent interactive treatments (e.g. [125]).

Due to the inherent non-interactivity of our scheme, we are able to avoid all session bookkeeping requiring no rounds, no session identifiers and no session state. In particular, this lets us state a simple and intuitive one-more unforgeability notion wherein an efficient adversary can adaptively corrupt any subset of signers (learning their secret keys) and may request pre-signature shares from honest signers. As a result, we can model scenarios where an adversary, having at least t pre-signatures per $(\mathsf{pk}_{\mathsf{U}}, \mathsf{c})$ pair by making pre-signature queries and by corrupting up to $(t-1)$ signers, cannot come up with more than the expected number of forgeries. This notion corresponds closely to the one-more unforgeability notion $\mathsf{OM\text{-}Unf\text{-}1}$ from [125].

We also describe a weaker one-more unforgeability notion. Informally, in this weak model each time the adversary makes a pre-signature query on some $(\mathsf{pk}_{\mathsf{U}}, \mathsf{c})$ pair, the challenger returns the pre-signature shares of all non-corrupted signers (i.e., exactly what an honest user would obtain). We therefore treat such a query as if the adversary already possesses a usable pre-signature on $(\mathsf{pk}_{\mathsf{U}}, \mathsf{c})$; consequently the adversary's count of legitimately obtained pre-signatures ℓ increases immediately, and the one-more condition requires that the adversary cannot output more than ℓ valid finalized

signatures[3]. The contrast with the previous stronger notion is important: in the strong model a pre-signature query does not automatically increase ℓ unless the adversary actually obtains enough signer shares (including shares obtained via corruption). The weak notion therefore corresponds roughly to the OM-Unf-0 variant of one-more unforgeability in [125], while the strong notion corresponds to their OM-Unf-1.

It is worth noting that these two definitions completely capture all possible issuing-oracle behaviors in our framework, since we do not need to model sessions or maintain state across protocol runs. Importantly, as our threshold blind signature construction is non-interactive, the four separate one-more unforgeability variants introduced in [125] for interactive threshold blind signatures are not required. For the corruption model we adopt the strongest (adaptive) notion; nonetheless, the definitions can be rephrased to accommodate an adversary that corrupts signers selectively. More details are given below.

A non-interactive threshold blind signature scheme over message space $\mathcal{M}$ is a tuple of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen_S}, \mathsf{KeyGen_U}, \mathsf{Issue}, \mathsf{Obtain})$ defined as follows:

- $\mathsf{Setup}\left(1^\lambda\right) \to \mathsf{pp}$. Given the security parameter λ, the probabilistic setup algorithm generates the collection of public parameters pp, which is an implicit input to the other algorithms.

- $\mathsf{KeyGen_S}\left(\mathsf{pp}, t, n\right) \to \left(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i\in[n]}\right)$. Given the public parameters pp, the minimum threshold weight t and the number of signers n, the probabilistic (signer) key generation algorithm outputs a master public key $\mathsf{mpk} \in \mathcal{PK}$ and a set of secret signing key shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in \mathcal{SK}^n$. We assume without loss of generality that mpk implicitly contains t and n.

- $\mathsf{KeyGen_U}(\mathsf{pp}) \to (\mathsf{sk_U}, \mathsf{pk_U})$. Given the public parameters pp, the probabilistic user key generation algorithm outputs a $\mathsf{pk_U} \in \mathcal{PK}_\mathsf{U}$ and a secret key $\mathsf{sk_U} \in \mathcal{SK}_\mathsf{U}$.

[3]The weaker notion could, for instance, be useful for blockchain applications where the signatures may be posted on-chain at once.

- $\mathsf{Issue}\left(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}\right) \to \bar{\sigma}$. Given a signing key share $\mathsf{sk} \in \mathcal{SK}$, a user public key $\mathsf{pk}_{\mathsf{U}} \in \mathcal{PK}_{\mathsf{U}}$ and context $\mathsf{c} \in \{0,1\}^{\lambda}$, the probabilistic issuance algorithm outputs a pre-signature share $\bar{\sigma} \in \overline{\Sigma}$.

- $\mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}}, \mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i \in \mathcal{S}}\right) \to (\mathsf{m}, \sigma)\,/(\perp, \perp)$. Given the user secret key $\mathsf{sk}_{\mathsf{U}} \in \mathcal{SK}_{\mathsf{U}}$, the master public key $\mathsf{mpk} \in \mathcal{PK}$, a set $\mathcal{S} \subseteq [n]$ of indices, context $\mathsf{c} \in \{0,1\}^{\lambda}$, and a collection of pre-signature shares $\bar{\sigma}_i \in \overline{\Sigma}$ for every $i \in \mathcal{S}$, the probabilistic obtain algorithm outputs a message and signature pair $(\mathsf{m}, \sigma) \in \mathcal{M} \times \Sigma$ or $(\perp, \perp)$.

- $\mathsf{Verify}\left(\mathsf{mpk}, \mathsf{m}, \sigma\right) \to 1/0$. Given the master public key $\mathsf{mpk} \in \mathcal{PK}$, a message $\mathsf{m} \in \mathcal{M}$ and a signature $\sigma \in \Sigma$, the deterministic signature verification algorithm outputs 1 (accept) or 0 (reject).

It must further satisfy the following properties:

Definition 5.9 (Correctness)**.** For all $\lambda, t, n \in \mathbb{N}$, such that $t \leqslant n$, all $\mathsf{c} \in \{0,1\}^{\lambda}$, and all $\mathcal{S} \subseteq [n]$ satisfying $|\mathcal{S}| \geqslant t$, a non-interactive threshold blind signature scheme NITBS satisfies *correctness* if

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{mpk}, \mathsf{m}, \sigma) = 1 \ : \\ \end{array} \begin{array}{r} \mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}\left(1^{\lambda}\right) \\ \left(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i \in [n]}\right) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{S}}\left(\mathsf{pp}, t, n\right) \\ \left(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}}\right) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \\ \forall i \in \mathcal{S} \ : \ \bar{\sigma}_i \leftarrow\!\$\ \mathsf{Issue}\left(\mathsf{sk}_i, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}\right) \\ (\mathsf{m}, \sigma) \leftarrow\!\$\ \mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}}, \mathsf{mpk}, \mathcal{S}, \{\mathsf{c}_i, \bar{\sigma}_i\}_{i \in \mathcal{S}}\right) \end{array}\right] = 1\,.$$

Definition 5.10 (Efficiency)**.** For all $\lambda, n \in \mathbb{N}$, a non-interactive threshold blind signature scheme, NITBS, is *efficient* if $|\mathsf{m}, \sigma| = O_{\lambda}(1)$, i.e., the final message and signature sizes are independent of n.

Definition 5.11 (One-more unforgeability)**.** For all $\lambda, n, t \in \mathbb{N}$ with $t \leqslant n$, let

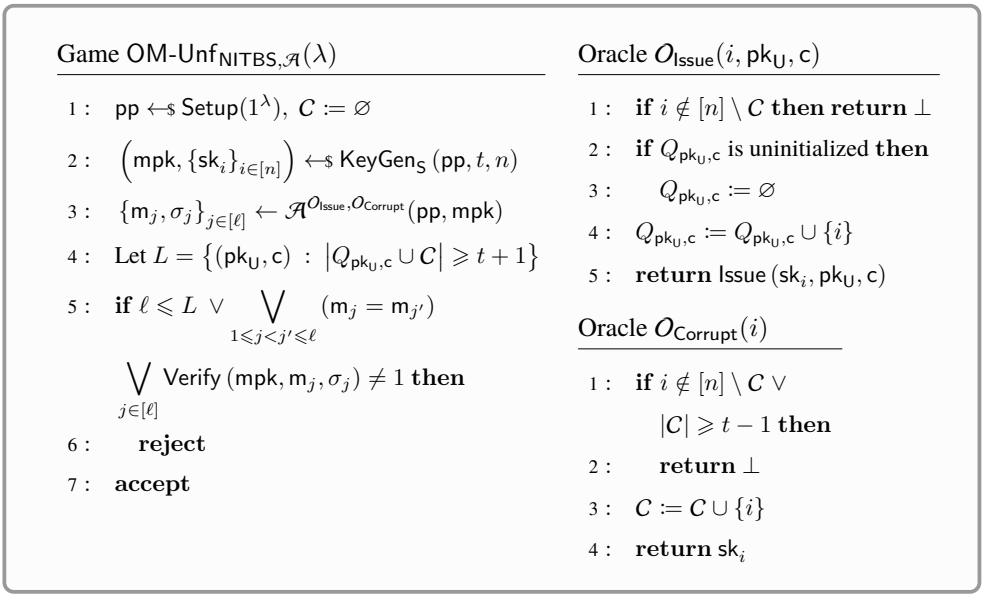

Figure 5.2: The one-more unforgeability experiment for NITBS.

$$\mathsf{Adv}^{\mathsf{OM\text{-}Unf}}_{\mathsf{NITBS},\mathcal{A}} = \Pr\left[\mathsf{OM\text{-}Unf}_{\mathsf{NITBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 5.2. We say a non-interactive threshold blind signature scheme, NITBS, is *one-more unforgeable* if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

Definition 5.12 (Blindness)**.** For all $\lambda, n, t \in \mathbb{N}$ with $t \leqslant n$, let

$$\mathsf{Adv}^{\mathsf{U\text{-}Blind}}_{\mathsf{NITBS},\mathcal{A}} = \Pr\left[\mathsf{U\text{-}Blind}_{\mathsf{NITBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of a *stateful* adversary $\mathcal{A}$ in the experiment defined in Figure 5.3 (left), and

$$\mathsf{Adv}^{\mathsf{C\text{-}Blind}}_{\mathsf{NITBS},\mathcal{A}} = \Pr\left[\mathsf{C\text{-}Blind}_{\mathsf{NITBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

Game U-Blind$_{\mathsf{NITBS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda),\ b \leftarrow\!\$\ \{0,1\}$
foreach $\omega \in \{0,1\}$ **do**
  $(\mathsf{pk}_{\mathsf{U}_\omega}, \mathsf{sk}_{\mathsf{U}_\omega}) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp})$
$\left(\mathsf{mpk}, \left\{ \begin{array}{l} \mathsf{c}_\omega, \\ \{\bar{\sigma}_{\omega,i}\}_{i\in\mathcal{S}_\omega} \end{array} \right\}_{\omega\in\{0,1\}}\right)$
$\leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{U}}_{\mathsf{Obtain}}}\left(\mathsf{pk}_{\mathsf{U}_0}, \mathsf{pk}_{\mathsf{U}_1}\right)$
foreach $\omega \in \{0,1\}$ **do**
  if $\mathsf{c}_\omega \in Q$ **then**
    reject
  $(\mathsf{m}_\omega, \sigma_\omega) \leftarrow\!\$\ \mathsf{Obtain}\left(\begin{array}{l} \mathsf{sk}_{\mathsf{U}_\omega}, \mathsf{mpk}, \mathcal{S}_\omega, \\ \mathsf{c}_\omega, \{\bar{\sigma}_{\omega,i}\}_{i\in\mathcal{S}_\omega} \end{array} \right)$
  if $\mathsf{Verify}(\mathsf{mpk}, \mathsf{m}_\omega, \sigma_\omega) \neq 1$ **then**
    reject
$b^* \leftarrow \mathcal{A}\left(\mathsf{m}_b, \sigma_b, \mathsf{m}_{1-b}, \sigma_{1-b}\right)$
if $b^* = b$ **then**
  accept
reject

Oracle $\mathcal{O}^{\mathsf{U}}_{\mathsf{Obtain}}\left(\omega, \mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i\in\mathcal{S}}\right)$

if Q is uninitialized **then**
  $Q \coloneqq \varnothing$
$Q \coloneqq Q \cup \{\mathsf{c}\}$
return $\mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}_\omega}, \mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i\in\mathcal{S}}\right)$

Game C-Blind$_{\mathsf{NITBS},\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda),\ b \leftarrow\!\$\ \{0,1\}$
$(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}}) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp})$
$\left(\mathsf{mpk}, \left\{ \begin{array}{l} \mathsf{c}_\omega, \\ \{\bar{\sigma}_{\omega,i}\}_{i\in\mathcal{S}_\omega} \end{array} \right\}_{\omega\in\{0,1\}}\right)$
$\leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{C}}_{\mathsf{Obtain}}}(\mathsf{pk}_{\mathsf{U}})$
foreach $\omega \in \{0,1\}$ **do**
  if $\mathsf{c}_\omega \in Q$ **then**
    reject
  $(\mathsf{m}_\omega, \sigma_\omega) \leftarrow\!\$\ \mathsf{Obtain}\left(\begin{array}{l} \mathsf{sk}_{\mathsf{U}}, \mathsf{mpk}, \mathcal{S}_\omega, \\ \mathsf{c}_\omega, \{\bar{\sigma}_{\omega,i}\}_{i\in\mathcal{S}_\omega} \end{array} \right)$
  if $\mathsf{Verify}(\mathsf{mpk}, \mathsf{m}_\omega, \sigma_\omega) \neq 1$ **then**
    reject
$b^* \leftarrow \mathcal{A}\left(\mathsf{m}_b, \sigma_b, \mathsf{m}_{1-b}, \sigma_{1-b}\right)$
if $b^* = b$ **then**
  accept
reject

Oracle $\mathcal{O}^{\mathsf{C}}_{\mathsf{Obtain}}\left(\mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i\in\mathcal{S}}\right)$

if Q is uninitialized **then**
  $Q \coloneqq \varnothing$
$Q \coloneqq Q \cup \{\mathsf{c}\}$
return $\mathsf{Obtain}\left(\mathsf{sk}_{\mathsf{U}}, \mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i\in\mathcal{S}}\right)$

Figure 5.3: The user (left) and context (right) blindness experiments for NITBS.

be its advantage in the experiment defined in Figure 5.3 (right). We say a non-interactive threshold blind signature scheme, NITBS, satisfies *blindness* if for any PPT adversary $\mathcal{A}$ each of these advantages is at most $1/2 + \mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\cdot)$.

For completeness, we also describe the weaker notion of one-more unforgeability, which roughly corresponds to the notion of OM-Unf-0 in threshold (interactive) blind-signatures.

Definition 5.13 (Weak one-more unforgeability)**.** For all $\lambda, n, t \in \mathbb{N}$ with $t \leqslant n$

$$\mathsf{Adv}^{\mathsf{wOM\text{-}Unf}}_{\mathsf{NITBS},\mathcal{A}} = \Pr\left[\mathsf{wOM\text{-}Unf}_{\mathsf{NITBS},\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

denotes the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 5.4. We say that a non-interactive threshold blind signature scheme, NITBS, is *weakly one-more unforgeable* if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

Game $\mathsf{wOM\text{-}Unf}_{\mathsf{NITBS},\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda),\ C \coloneqq \varnothing$
2 : $\left(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i\in[n]}\right) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{S}}\left(\mathsf{pp}, t, n\right)$
3 : $\{\mathsf{m}_j, \sigma_j\}_{j\in[\ell]} \leftarrow \mathcal{A}^{O_{\mathsf{Issue}}, O_{\mathsf{Corrupt}}}(\mathsf{pp}, \mathsf{mpk})$
4 : **if** $\ell \leqslant L \ \vee \bigvee_{1\leqslant j<j'\leqslant\ell} (\mathsf{m}_j = \mathsf{m}_{j'})$
$\bigvee_{j\in\ell} \mathsf{Verify}\left(\mathsf{mpk}, \mathsf{m}_j, \sigma_j\right) \neq 1$ **then**
5 : **reject**
6 : **accept**

Oracle $O_{\mathsf{Issue}}(\mathsf{pk}_{\mathsf{U}}, \mathsf{c})$

1 : **if** L is uninitialized **then**
2 : $L \coloneqq 0$
3 : $L \coloneqq L + 1$
4 : **foreach** $i \in [n] \setminus C$ **do**
5 : $\bar{\sigma}_i \leftarrow\!\$\ \mathsf{Issue}\left(\mathsf{sk}_i, \mathsf{pk}_{\mathsf{U}}, \mathsf{c}\right)$
6 : **return** $\{\bar{\sigma}_i\}_{i\in[n]\setminus C}$

Figure 5.4: The weak one-more unforgeability experiment for NITBS.

5.4 NITBS from Pointcheval-Sanders Assumption

This section introduces our main result, a non-interactive threshold blind signatures construction from PS signatures. Our starting point is the non-interactive blind signature scheme that we construct using the Pointcheval-Sanders signature scheme [148]. We choose PS signatures specifically for their efficiency and, critically, their amenability to the thresholding template we employ below. Other signature schemes such as BBS [44] and BBS+ [20, 53] share similar properties, and we propose their exploration for candidate NITBS constructions as future work.

NIBS from PS signatures. Recall that in the PS signature scheme, given generators $g \in \mathbb{G}$ and $\tilde{g} \in \tilde{\mathbb{G}}$ of two pairing-friendly groups, a signature on a message vector $(\mu_1, \ldots, \mu_k) \in \mathbb{F}^k$ is $\Psi := \left(\psi_1 = h, \psi_2 = h^{x+\sum_{i=1}^{k} y_i \mu_i}\right)$ where $(x, y_1, y_2, \ldots, y_k) \in \mathbb{F}^{k+1}$ is the secret signing key sk. The corresponding public verification key is $\mathsf{pk} := \left(\tilde{X} = \tilde{g}^x, \tilde{Y}_1 = \tilde{g}^{y_1}, \tilde{Y}_2 = \tilde{g}^{y_2}, \ldots, \tilde{Y}_k = \tilde{g}^{y_k}\right) \in \tilde{\mathbb{G}}^{k+1}$. Verification succeeds if $e(\psi_2, \tilde{g}) \stackrel{?}{=} e\left(\psi_1, \tilde{X} \cdot \prod_{i=1}^{k} \tilde{Y}_i^{\mu_i}\right)$. The PS problem (Def. 5.7) requires that, given $g, \tilde{g}, \tilde{g}^x, \tilde{g}^y$ and an oracle that outputs $(h, h^{x+y\mu})$ on input $\mu \in \mathbb{F}$ (with h uniform over $\mathbb{G}$), it is hard to find $(\mu^*, h^*, (h^*)^{x+y\mu^*})$ for any fresh $\mu^* \in \mathbb{F}$ and nontrivial h^*.

To construct a NIBS scheme, a natural approach is to have the signer with key-pair $\mathsf{pk} := (\tilde{X}, \tilde{Y}_1, \tilde{Y}_2)$ and $\mathsf{sk} := (x, y_1, y_2)$ sign the tuple $(\mathsf{c}, \mathsf{pk}_\mathsf{U})$ and send it as the pre-signature $\bar{\sigma}$, which the user then re-randomizes. However, pk_U is a group element and cannot be signed directly. We resolve this by setting $\mathsf{sk}_\mathsf{U} := \alpha$ and $\mathsf{pk}_\mathsf{U} := g^\alpha$. The signer instead computes $h = g^r$ and $H = \left(g^{x+y_1\mathsf{c}} \cdot \mathsf{pk}_\mathsf{U}^{y_2}\right)^r$ for $r \leftarrow\!\!\$\ \mathbb{F}$. Since $H = (g^r)^{x+y_1\mathsf{c}+y_2\alpha}$, we have $\bar{\sigma} := (h, H)$ as a valid PS signature on $(\mathsf{c}, \mathsf{sk}_\mathsf{U})$.

The user re-randomizes $\bar{\sigma}$ to obtain a fresh signature Ψ unlinkable to the pre-signature. However, the message cannot be sent in the clear. Instead, the user computes $\mathsf{m} := g^{1/(\mathsf{c}+\alpha)}$ and constructs a NIZK proof that it knows c and α such that Ψ is a valid PS signature under pk on the message (c, α) and $\mathsf{m} = g^{1/(\mathsf{c}+\alpha)}$. The final blind signature consists of the re-randomized PS signature Ψ along with this proof π. To optimize efficiency, we design π as a Σ-protocol based on the Schnorr identification scheme [157].

Blindness follows from the zero-knowledge property of π, indistinguishability of Ψ from uniform under the Strong Decisional Diffie-Hellman problem [147], and pseudorandomness of m under the hardness of the Decisional q-Diffie Hellman Inverse problem [77]. Unforgeability reduces to the hardness of the PS problem, with two key technical obstacles. First, we cannot sign c directly. Instead, we replace it with a programmable value from the PS challenger, so the signer signs the message $(\kappa := \mathsf{H}(\mathsf{pk}_\mathsf{U}, \mathsf{c}), \mathsf{sk}_\mathsf{U})$, where we model H as a random oracle whose outputs are tied to the PS oracle. Second, a naive forking argument for extracting the NIZK proof would incur soundness

loss exponential in ℓ, the number of pre-signature queries. Instead, we achieve online extractability by working in the AGM [89], which preserves the efficiency of the Schnorr protocol.

Remark 5.1. This scheme offers faster issuance, and smaller pre-signatures and final signatures than the previous state of the art NIBS scheme due to Hanzlik [101]. Concretely, issuance requires 3 exponentiations in $\mathbb{G}$, 1 multiplication in $\mathbb{G}$, and 1 H computation into $\mathbb{F}$, compared to 5 exponentiations in $\mathbb{G}$ and 2 multiplications in $\mathbb{G}$ in [101]. The pre-signature size here is just 2 elements in $\mathbb{G}$, which is roughly half the pre-signature size in [101], where it additionally includes 1 element in $\tilde{\mathbb{G}}$. The same is true for the final signature.

Thresholdizing PS-NIBS. Given the NIBS scheme from PS signatures, we attempt to apply the standard thresholdizing approach for some threshold t by distributing t-out-of-n linear secret shares of $\mathsf{sk} := (x, y_1, y_2)$ among n signers.[4] Let $[x]_i$ denote the i^{th} share of x (and similarly for $[y_1]_i$ and $[y_2]_i$). Each signer issues a pre-signature $\bar{\sigma}_i$ over a shared context c, which must be fixed across all signers to prevent users from mixing pre-signatures from different contexts. For example, signers could compute $\mathsf{c} := \mathsf{H}(\mathsf{pk_U} \| \texttt{cur\_epoch})$, where the epoch duration is fixed at some reasonable interval to ensure all signers issue their signatures within that window.

Now, a natural question is whether the user can simply aggregate t or more pre-signature shares and prove validity. However, doing so with Schnorr proofs would yield proof sizes scaling with $t = O(n)$, defeating the efficiency gains. Instead, the user must locally aggregate the t or more pre-signature shares into a single valid PS signature. The first obstacle is that each signer signs over a distinct base h_i, making it unclear how to combine the resulting signature components. We address this by having all signers compute a common base $h := \mathsf{H}_{\mathbb{G}}(\mathsf{pk_U} \| \mathsf{c})$ that depends only on the user's key and issuance context. Let $\kappa := \mathsf{H}_{\mathbb{Z}}(\mathsf{pk_U}, \mathsf{c})$.

The second obstacle arises because signers cannot compute the full signature component, as no individual signer knows h^{α}. Instead, the i^{th} signer factors the pre-signature into two parts: $A_i := h^{[x]_i + [y_1]_i \kappa} \cdot g^{r_i}$ and $B_i := (\mathsf{pk_U})^{-r_i} \cdot h^{[y_2]_i}$, sending both to the user. The user can then compute a

[4] We designed the issuance computation to decompose into additive shares that independent signers can produce non-interactively while still yielding a valid blind signature upon aggregation. This property does not appear in prior NIBS schemes, and the standard secret-sharing recipe does not apply directly to existing designs [24, 25, 101, 103].

valid PS signature as $(h, A_i \cdot B_i^{\alpha}) = (h, h^{[x]_i + [y_1]_i \kappa + [y_2]_i \alpha})$ with the common base h. Note that the user's public key must now be $\mathsf{pk}_{\mathsf{U}} := g^{1/\alpha}$ to preserve the property that the PS signature is over α. The randomness r_i introduced by each signer is essential for the unforgeability of the scheme.

Any user with at least t valid pre-signature shares can use the share recomposition algorithm of the linear threshold secret-sharing scheme to obtain the final PS signature $(h, h^{x+y_1\kappa+y_2\alpha})$. The protocol then proceeds as before, with the user proving knowledge of κ and α such that the re-randomized PS signature Ψ is valid under pk on the message (κ, α) and $\mathsf{m} := g^{1/(\kappa+\alpha)}$. This aggregation step does not affect the issuance times or the pre-signature and final signature sizes compared to the non-threshold NIBS construction.

Construction 5.1. Our construction requires hash functions $\mathsf{H}_{\mathbb{G}} : \{0,1\}^{\lambda} \to \mathbb{G}$ and $\mathsf{H}_{\mathbb{Z}} : \{0,1\}^{\lambda} \to \mathbb{Z}_p^*$. We will also make use of three non-interactive proof systems $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$, for languages $\mathcal{L}_{\mathsf{U}}^{\mathsf{NITBS}}$, $\mathcal{L}_{\mathsf{PS}}^{\mathsf{NITBS}}$ as defined. The language will be evident from the statement used in $\Pi.\mathsf{Prove}$ and $\Pi.\mathsf{Verify}$. Moreover, we will use different common reference strings $\mathsf{crs}_{\mathsf{U}}$ and $\mathsf{crs}_{\mathsf{PS}}$ to distinguish those two different proof systems.

Language $\mathcal{L}_{\mathsf{U}}^{\mathsf{NITBS}}$
Instance: Each instance $\mathbb{x}$ is interpreted as group elements $(\phi_1, \phi_2) \in \mathbb{G}^2$.
Witness: Witness $\mathbb{w}$ consists of scalar $\alpha \in \mathbb{Z}_p^*$.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\phi_2^{\alpha} = \phi_1$$

Concretely, the NITBS construction is defined by the following algorithms:

- $\mathsf{Setup}\left(1^{\lambda}\right) \to \mathsf{pp}$. The setup algorithm generates the common reference strings $\mathsf{crs}_{\mathsf{U}}$ and $\mathsf{crs}_{\mathsf{PS}}$ for the proof systems for languages $\mathcal{L}_{\mathsf{U}}^{\mathsf{NITBS}}$ and $\mathcal{L}_{\mathsf{PS}}^{\mathsf{NITBS}}$, respectively.[5]

[5] It is worth noting that, depending on the instantiation of the proof systems, the setup algorithm can be transparent, i.e., no trusted party may be needed to run this setup algorithm.

Language $\mathcal{L}_{\mathsf{PS}}^{\mathsf{NITBS}}$

Instance: Each instance $\mathbb{x}$ is interpreted as the issuer's public key $\mathsf{pk} = (\tilde{X}, \tilde{Y}_1, \tilde{Y}_2) \in \tilde{\mathbb{G}}^3$, a group element $\mathsf{m} \in \tilde{\mathbb{G}}$ and a PS signature $\Psi = (\psi_1, \psi_2) \in \mathbb{G}^2$.
Witness: Witness $\mathbb{w}$ consists of scalars $(\kappa, \alpha) \in (\mathbb{Z}_p^*)^2$.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$e(\psi_1, \tilde{X}) \cdot e(\psi_1, \tilde{Y}_1)^{\kappa} \cdot e(\psi_1, \tilde{Y}_2)^{\alpha} = e(\psi_2, \tilde{g})$$

▷ Ψ is a valid PS signature for the message (κ, α)

$$\wedge\ \mathsf{m}^{\kappa+\alpha} = g_2$$

▷ and m is a valid NIBS message constructed from (κ, α)

It then sets the public parameters $\mathsf{pp} := (e, p, \mathbb{G}, \tilde{\mathbb{G}}, \hat{\mathbb{G}}, g, g_1, g_2, \tilde{g}, \tilde{g_1}, \tilde{g_2}, \mathsf{crs}_{\mathsf{U}}, \mathsf{crs}_{\mathsf{PS}})$, where $e : \mathbb{G} \times \tilde{\mathbb{G}} \to \hat{\mathbb{G}}$ is a bilinear pairing of type 3, g, g_1, g_2 are generator of $\mathbb{G}$ and $\tilde{g}$ is a generator of $\tilde{\mathbb{G}}$, all of order p.

- $\mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp}, t, n) \to \left(\mathsf{mpk}, \{\mathsf{sk}_i\}_{i\in[n]}\right)$. Given the public parameters pp, the minimum threshold weight t and the number of signers n, the signing key generation algorithm samples a random $(x, y_1, y_2) \leftarrow\!\$\ (\mathbb{Z}_p^*)^3$ and computes the master public key $\mathsf{mpk} := (Y_1 := g^{y_1}, Y_2 := g^{y_2}, \tilde{X} := \tilde{g}^x, \tilde{Y_1} := \tilde{g}^{y_1}, \tilde{Y_2} := \tilde{g}^{y_2})$. It then computes (t, n)-threshold secret shares and assigns the i^{th} secret key share to the i^{th} signer as $\mathsf{sk}_i := ([x]_i, [y_1]_i, [y_2]_i) \leftarrow \mathsf{Share}_{t,n}((x, y_1, y_2))$. Finally, it outputs mpk and $[\mathsf{sk}]_1, [\mathsf{sk}]_2, \ldots, [\mathsf{sk}]_n$.

- $\mathsf{KeyGen}_{\mathsf{U}}(\mathsf{pp}) \to (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. Given the public parameters pp, the user's key generation algorithm samples a random value $\mathsf{sk}_{\mathsf{U}} \leftarrow\!\$\ \mathbb{Z}_p^*$ as the secret signing key. It sets the public key $\mathsf{pk}_{\mathsf{U}} := (g_1^{1/\mathsf{sk}_{\mathsf{U}}}, \pi_{\mathsf{U}})$, where π_{U} is a proof of possession of the secret key computed as

$$\pi_{\mathsf{U}} \leftarrow\!\$\ \Pi_{\mathsf{U}}.\mathsf{Prove}\left(\mathsf{crs}_{\mathsf{U}}, \mathbb{x} := (g_1, g_1^{1/\alpha}), \mathbb{w} := (\alpha)\right) ,$$

and outputs sk_{U} and pk_{U}.

- $\mathsf{Issue}\left([\mathsf{sk}]_i, \mathsf{pk}_\mathsf{U}, \mathsf{c}\right) \rightarrow \bar{\sigma}$. Given a signing key share $[\mathsf{sk}]_i$ parsed as $([x]_i, [y_1]_i, [y_2]_i)$, a user public key pk_U parsed as $(\mathsf{pk}'_\mathsf{U}, \pi_\mathsf{U})$ and context c, the i^{th} verifies the user's public key by running the NIZK verification as $\Pi_\mathsf{U}.\mathsf{Verify}(\mathsf{crs}_\mathsf{U}, \mathbb{x} := (g_1, \mathsf{pk}'_\mathsf{U}), \pi_\mathsf{U})$. If the verification fails, the signer aborts the issuance and continues otherwise. It then picks a random $r \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and computes the generator $h := \mathsf{H}_\mathbb{G}(\mathsf{pk}_\mathsf{U}, \mathsf{c})$ using the user's public key pk_U and context c. It then computes

$$A_i := h^{[x]_i + [y_1]_i \kappa} \cdot g_1^r \text{ and } B_i := (\mathsf{pk}'_\mathsf{U})^{-r} \cdot h^{[y_2]_i} \ ,$$

where $\kappa := \mathsf{H}_\mathbb{Z}(\mathsf{pk}_\mathsf{U}, \mathsf{c})$. Finally it outputs a pre-signature share $\bar{\sigma}_i := (A_i, B_i)$.

- $\mathsf{Obtain}\left(\mathsf{sk}_\mathsf{U}, \mathsf{mpk}, \mathcal{S}, \mathsf{c}, \{\bar{\sigma}_i\}_{i \in \mathcal{S}}\right) \rightarrow (\mathsf{m}, \sigma)$. Given the user secret key $\alpha =: \mathsf{sk}_\mathsf{U}$, the master public key mpk parsed as $(Y_1, Y_2, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2)$, a set $\mathcal{S} \subseteq [n]$ of indices, context c, and a collection of pre-signature shares $\bar{\sigma}_i$ for every $i \in \mathcal{S}$, the user's obtain algorithm first computes $h := \mathsf{H}_\mathbb{G}(\mathsf{pk}_\mathsf{U}, \mathsf{c})$ and $\kappa := \mathsf{H}_\mathbb{Z}(\mathsf{pk}_\mathsf{U}, \mathsf{c})$. Then, using the Lagrange coefficients L_i for each issuer in $\mathcal{S}$ (which it can pre-compute), it sets the PS signature

$$\Psi := \left(h^\rho, \prod_{i \in \mathcal{S}} (A_i \cdot B_i^\alpha)^{\rho \Lambda_i} \right) \ ,$$

for $\rho \leftarrow\!\!\$\ \mathbb{Z}_p$. In the final step, it computes the message for the blind signature as $\mathsf{m} := g_2^{1/(\alpha+\kappa)}$ and computes proof

$$\pi_\mathsf{PS} \leftarrow\!\!\$\ \Pi_\mathsf{PS}.\mathsf{Prove}\left(\mathsf{crs}_\mathsf{PS}, \mathbb{x} := (\mathsf{mpk}, \mathsf{m}, \Psi), \mathbb{w} := (\kappa, \alpha)\right) \ ,$$

and outputs the message m and the signature $\sigma := (\Psi, \pi_\mathsf{PS})$.

- $\mathsf{Verify}\,(\mathsf{mpk}, \mathsf{m}, \sigma) \rightarrow 1/0$. Given the master public key, a message m and a signature $(\Psi, \pi_{\mathsf{PS}}) =: \sigma$, the signature verification algorithm returns the output of NIZK verification $\Pi_{\mathsf{PS}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{PS}}, \mathbb{x} := (\mathsf{mpk}, \mathsf{m}, \Psi), \pi_{\mathsf{PS}})$.

The unforgeability of our construction relies on the (t, n)-threshold PS problem that we define next.

Definition 5.14 ((t, n)-threshold PS Problem)**.** Let $(\mathbb{G}, \tilde{\mathbb{G}})$ be pairing friendly groups of order p with a bilinear pairing $e : \mathbb{G} \times \tilde{\mathbb{G}} \rightarrow \hat{\mathbb{G}}$. For a uniformly random chosen polynomials $f_x(X)$ and $f_y(X)$ of degree t over $\mathbb{Z}_p[X]$, the (t, n)-threshold PS problem asks that given access to the oracles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_{\mathsf{Corrupt}}$ and $\left(p, g, \tilde{g}, e, \tilde{X} := \tilde{g}^{f_x(0)}, \tilde{Y} := \tilde{g}^{f_y(0)}, \left\{\tilde{X}_i := \tilde{g}^{f_x(i)}, \tilde{Y}_i := \tilde{g}^{f_y(i)}\right\}_{i \in [n]}\right)$, find a $\left(\mu^*, h^*, (h^*)^{f_x(0)+f_y(0)\mu^*}\right)$ triple such that $|C|+|\mathcal{S}_{\mu^*}| \leqslant (t-1)$ and $h \neq 1_{\mathbb{G}}$. On input $\mu \in \mathbb{Z}_p$ oracle $\mathcal{O}_1$ outputs $h \in \mathbb{G}$. On input of the same $\mu \in \mathbb{Z}_p$ and $i \in [n] \setminus C$, oracle $\mathcal{O}_2$ outputs $h^{f_x(i)+f_y(i)\mu}$. On any repeated queries, both oracles replay the same response as before and additionally, oracle $\mathcal{O}_2$ adds i to a set $\mathcal{S}_\mu$. Finally, the corruption oracle $\mathcal{O}_{\mathsf{Corrupt}}$ on input index $i \in [n]$, adds i to set C and returns $(f_x(i), f_y(i))$.

In fact, the (t, n)-threshold PS problem can be reduced to the one-more co-discrete logarithm assumption in the AGM.

Claim 5.15. *In the algebraic group model, the (t, n)-threshold PS assumption (Def. 5.14) holds under the one-more co-discrete logarithm assumption (Def. 5.4).*

Pf. sketch. We sketch a reduction to the one-more co-discrete log problem. The reduction obtains t tuples $(T_i, \tilde{T}_i) = (g^{t_i}, \tilde{g}^{t_i})$ from the challenger and constructs the instance by embedding these values as the first t points of the polynomial $f_y(X)$ (i.e., $\tilde{Y}_i = \tilde{T}_i$ for $i \in [t-1]$ and $\tilde{Y} = \tilde{T}_0$). The remaining evaluation points of f_y are computed via Lagrange interpolation in the exponent.

When the (t, n)-threshold PS adversary queries the corruption oracle, the reduction uses the discrete logarithm oracle to recover the corrupt shares. When the adversary outputs a solution, it provides algebraic representations as a linear combination of known group elements. The key

observation is that after at most $(t-1)$ corruptions, the reduction knows $(t-1)$ distinct points of f_y, leaving a single unknown degree of freedom ξ. By interpolating the known points, the reduction can express $f_y(X) = q_y(X) + \xi \cdot u_y(X)$ where q_y and u_y are known. The reduction then extracts ξ using the same algebraic group model technique as the standard PS assumption, allowing it to recover f_y and output a valid solution to the one-more co-discrete logarithm problem using at most t oracle queries. □

We now state the main security theorem for our construction. The detailed proofs are provided in the full version of [27].

Theorem 5.16 (Security)**.** *Construction 5.1 is:*

(i) One-more unforgeable in the random oracle model (Def. 5.11) assuming the (t, n)-threshold PS problem is hard, and Π proof systems Π_{U} and Π_{PS} are online knowledge extractable, is one-more unforgeable.

(ii) Blind (Def. 5.12) assuming the sDDH *(Def. 5.5) and* qDDH *(Def. 5.6) problems are hard in $\mathbb{G}$ and proof systems Π_{U} and Π_{PS} satisfy zero-knowledge.*

Pf. sketch. To prove one-more unforgeability, we give a reduction to the (t, n)-threshold PS problem. The reduction receives threshold PS parameters from a challenger and simulates the NITBS scheme for the adversary by answering the random oracle queries consistently; and for each issuance request, it uses the knowledge extractor for Π_{U} to learn the user's secret, forwards the appropriately transformed query to its own PS oracle, and returns the response to the adversary. Suppose now that the adversary succeeds by producing $(\ell + 1)$ valid signatures on distinct messages. The reduction extracts the witness behind each signature using the online knowledge extractor for Π_{PS}. Now, since the messages are distinct and valid, the extracted witnesses must satisfy a structural property. Specifically, the values $(\kappa_i + z\alpha_i)$ cannot all be equal. This means that among the $(\ell+1)$ forged messages, at least one was queried to the PS oracle fewer than t times (essentially by a simple pigeonhole argument). For this message, the reduction has extracted a valid PS signature obtained without the threshold number of oracle queries.

Intuitively, both user and context blindness follows from the zero-knowledge of π, indistinguishability of Ψ from uniform (under the Strong Decisional Diffie-Hellman problem) and pseudorandomness of m (under the hardness of Decisional q-Diffie Hellman Inverse problem). The idea is to simulate all zero-knowledge and then replace the messages in the challenged signatures with random group elements. Finally, all the challenged values given to the adversary are either simulated without the user's secret key or are random group elements. □

5.4.1 Instantiating the NIZK proof system

We will now show how to efficiently instantiate the NIZK proof systems for languages $\mathcal{L}_{\mathsf{U}}^{\mathsf{NITBS}}$ and $\mathcal{L}_{\mathsf{PS}}^{\mathsf{NITBS}}$ and show that our instantiations satisfy the necessary properties.

Proof for language $\mathcal{L}_{\mathsf{U}}^{\mathsf{NITBS}}$

Users can use a proof of knowledge of α, where $\mathsf{pk}'_{\mathsf{U}} := g_1^{1/\alpha}$. This standard proof of discrete logarithm knowledge can be instantiated as follows. Compute $T_{\mathsf{U}} := (\mathsf{pk}'_{\mathsf{U}})^{\alpha'}$, for some random $\alpha' \leftarrow\!\!\$\ \mathbb{Z}_p^*$, and challenge $c_{\mathsf{U}} := \mathsf{H}(g_1, \mathsf{pk}'_{\mathsf{U}}, T_{\mathsf{U}})$. Set proof $\pi_{\mathsf{U}} = (c_{\mathsf{U}}, s_{\mathsf{U}})$, where $s_{\mathsf{U}} := \alpha' - c_{\mathsf{U}} \cdot \alpha \pmod p$. To verify the proof, the verifier computes $T'_{\mathsf{U}} = g_1^{c_{\mathsf{U}}} \cdot (\mathsf{pk}'_{\mathsf{U}})^{s_{\mathsf{U}}}$ and returns 1 iff $\mathsf{H}(g_1, \mathsf{pk}'_{\mathsf{U}}, T'_{\mathsf{U}}) = c_{\mathsf{U}}$.

Lemma 5.17. *Proof system Π_{U} is online knowledge extractable in the algebraic group model.*

Proof. We give an extractor $\mathcal{E}_{\mathsf{U}}$ that, given an instance $(\phi_1, \phi_2) \in \mathbb{G}^2$ and a proof π_{U}, is able to extract $\alpha \in \mathbb{Z}_p^*$ satisfying $\phi_2 = \phi_1^{\alpha}$. Observe that the adversary outputs π_{U} as part of its query to the O_{Issue} oracle, and that every such query has the form $(\mathsf{pk}_{\mathsf{U}}, \mathsf{c})$ where $\mathsf{pk}_{\mathsf{U}} =: (\mathsf{pk}'_{\mathsf{U}}, \pi_{\mathsf{U}}) \in \mathbb{G} \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$. In particular, for the k^{th} such query, the algebraic adversary must also output a representation $\Big(a_0, a_1, a_2, \{a_{3,i}\}_{i\in[|Q|]}, \{a_{4,j},\ a_{5,j}\}_{j\in[k-1]}\Big)$ such that

$$\mathsf{pk}'_{\mathsf{U}_k} = g^{a_0} \cdot g_1^{a_1} \cdot g_2^{a_2} \cdot \prod_{i\in[|Q|]} h_i^{a_{3,i}} \cdot \prod_{j\in[k-1]} \left(A_j^{a_{4,j}} \cdot B_j^{a_{5,j}}\right) ,$$

where Q is the set of all query and response pairs to $\mathsf{H}_{\mathbb{G}}$. Then, clearly for any such $\mathsf{pk}'_{\mathsf{U}_k}$, we have by soundness of the proof system that

$$g_1^{1/\alpha_k} = g^{a_0} \cdot g_1^{a_1} \cdot g_2^{a_2} \cdot \prod_{i \in [|Q|]} h_i^{a_{3,i}} \cdot \prod_{j \in [k-1]} \left(A_j^{a_{4,j}} \cdot B_j^{a_{5,j}} \right) .$$

Expanding each $A_j = h_j^{x + y_1 \kappa_j}$ and $B_j = \mathsf{pk}'_{\mathsf{U}_j} \cdot h_j^{y_2} = g_1^{1/\alpha_j} \cdot h_j^{y_2}$, (for indeterminates x, y_1 and y_2), we can re-write the right-hand side so that

$$g_1^{1/\alpha_k} = g^{a_0} \cdot g_1^{a_1 + \sum_{j \in [k-1]} (a_{4,j}/\alpha_j)} \cdot g_2^{a_2} \cdot \prod_{i \in [|Q|]} h_i^{a_{3,i}} \cdot \prod_{j \in [k-1]} h_j^{a_{4,j} x + a_{4,j} y_1 \kappa_j + a_{5,j} y_2} . \tag{5.1}$$

Now, let each $h_i = g^{\gamma_i}$ for some indeterminate $\gamma_i \in \mathbb{Z}_p$. Then, passing to the discrete-log base g and re-arranging gives

$$\beta_1 \cdot \left(1/\alpha_k - a_1 - \sum_{j \in [k-1]} (a_{4,j}/\alpha_j) \right) - \beta_2 \cdot a_2 - \sum_{i \in [|Q|]} \gamma_i \cdot a_{3,i}$$

$$+ \sum_{j \in [k-1]} \gamma_j \cdot (a_{4,j} x + a_{4,j} y_1 \kappa_j + a_{5,j} y_2 \alpha_) = a_0 \pmod{p} .$$

For uniform choices of β_1, β_2 and all γ_i's, except with probability at most $1/p$, this equation holds only when

$$\forall i \in [|Q|],\ j \in [k-1] \ :\ a_2 = a_{3,i} = a_{4,j} = a_{5,j} = 0 \ \text{ and } \ 1/\alpha_k = a_1 .$$

An additional caveat completes the argument. Specifically, the extractor may not control the actual values of the γ_i when the bases h_i are supplied by an external challenger. The argument above

therefore only guarantees vanishing exponents against uniform γ_i. However, except with some negligible probability $\nu(\lambda)$, the h_i's are computationally indistinguishable from uniform[6]. Combining the above, $\mathcal{E}_{\mathsf{U}}$ fails to extract the witness for π_{U} with probability at most $\frac{1}{p} + \nu(\lambda) + \mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{A}}(\lambda)$. $\square$

Proof for language $\mathcal{L}^{\mathsf{NITBS}}_{\mathsf{PS}}$

The proof for language $\mathcal{L}^{\mathsf{NITBS}}_{\mathsf{PS}}$ is comprised of a proof of knowledge of (κ, α) such that $\Psi = (\psi_1, \psi_2)$ is a valid PS-signature on (κ, α) and $\mathsf{m} = g_2^{\kappa+\alpha}$. In order to instantiate this proof, we first recall that the verification check for a PS-signature Ψ on (κ, α) is

$$\underbrace{e(\psi_1, \tilde{Y}_1)}_{E_1}{}^{\kappa} \cdot \underbrace{e(\psi_1, \tilde{Y}_2)}_{E_2}{}^{\alpha} = \underbrace{e(\psi_1, \tilde{X}^{-1}) \cdot e(\psi_2, \tilde{g})}_{E_3} .$$

Now in order to compute this proof, the prover first samples $(\kappa', \alpha') \leftarrow\!\$\ (\mathbb{Z}_p^*)^2$, and then computes commitments $T_1 \coloneqq E_1^{\kappa'} \cdot E_2^{\alpha'}$ and $T_2 \coloneqq \mathsf{m}^{\kappa'+\alpha'}$. Next, it computes the challenge $c_{\mathsf{PS}} = \mathsf{H}(g, g_2, \tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2, \psi_1, \psi_2, E_1, E_2, E_3, T_1, T_2)$ and sends the proof $\pi_{\mathsf{PS}} \coloneqq (c_{\mathsf{PS}}, s_\kappa, s_\alpha)$ where $s_\kappa \coloneqq \kappa' - c_{\mathsf{PS}} \cdot \kappa \pmod p$ (similarly, s_α). To verify the proof, the verifier first computes E_1, E_2 and E_3 as defined, then sets $T_1 \coloneqq E_1^{s_\kappa} \cdot E_2^{s_\alpha} \cdot E_3^{-c_{\mathsf{PS}}}$ and $T_2 \coloneqq \mathsf{m}^{s_\kappa + s_\alpha} \cdot g_2^{-c_{\mathsf{PS}}}$ and finally outputs 1 iff $\mathsf{H}(g, g_2, \tilde{g}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2, \psi_1, \psi_2, E_1, E_2, E_3, T_1, T_2) = c_{\mathsf{PS}}$.

Summarizing all the above, the proof π_{U} consists of 2 elements $(c_{\mathsf{U}}, s_{\mathsf{U}})$ in $\mathbb{Z}_p^*$; and the proof π_{PS} consists of the randomized PS signature (ψ_1, ψ_2) and the proof $\pi_{\mathsf{PS}} \coloneqq (c_{\mathsf{PS}}, s_\kappa, s_\alpha)$. Therefore, π_{PS} consists of 2 elements in $\mathbb{G}$ and 3 elements in $\mathbb{Z}_p^*$.

Lemma 5.18. *Proof system Π_{PS} is online knowledge extractable in the algebraic group model.*

Proof. We give an extractor $\mathcal{E}_{\mathsf{PS}}$ that, given an instance $(\mathsf{m}, \psi_1, \psi_2, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2) \in \mathbb{G}^3 \times \tilde{\mathbb{G}}^3$ and a proof π_{PS}, is able to extract scalars $(\kappa, \alpha) \in (\mathbb{Z}_p^*)^2$ satisfying $e(\psi_1, \tilde{X}) \cdot e(\psi_1, \tilde{Y}_1)^{\kappa} \cdot e(\psi_1, \tilde{Y}_2)^{\alpha} = e(\psi_2, \tilde{g}_2)$ and $\mathsf{m} = g_2^{1/(\kappa+\alpha)}$.

[6] A similar argument was made in [2] to prove simulation extractability of Schnorr proof system for password-authenticated key exchange, where the crux of the argument fell on showing that the challenge bases were indistinguishable from uniformly random bases.

Recall that the adversary outputs π_{PS} as part of its (one-more) forgery, and that every such forgery consists of group elements $(\mathsf{m}, \psi_1, \psi_2)$ along with their representations in their respective bases. We can rewrite this representation similarly to (5.1) as

$$\mathsf{m} = g^{a_0} \cdot g_1^{a_1 + \sum_{j \in [\ell]} (a_{4,j}/\alpha_j)} \cdot g_2^{a_2} \cdot \prod_{i \in [|Q|]} h_i^{a_{3,i}} \cdot \prod_{j \in [\ell]} h_j^{a_{4,j} x + a_{4,j} y_1 \kappa_j + a_{5,j} y_2} \ ,$$

the honest relation for this commitment asserts that $\mathsf{m} = g_2^{1/(\kappa+\alpha)}$ for some witness pair (κ, α). We may thus equate the two group elements and rearrange to obtain

$$g^{a_0} \cdot g_1^{a_1 + \sum_{j \in [\ell]} (a_{4,j}/\alpha_j)} \cdot g_2^{a_2 - \frac{1}{\kappa+\alpha}} \cdot \prod_{i \in [|Q|]} h_i^{a_{3,i}} \cdot \prod_{j \in [\ell]} h_j^{a_{4,j} x + a_{4,j} y_1 \kappa_j + a_{5,j} y_2} = 1 \ .$$

We can show identically as for the case of π_{U} that with probability at most $\frac{1}{p} + \nu(\lambda) + \mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{A}}(\lambda)$, the exponents other than a_2 are all zero, which allows $\mathcal{E}_{\mathsf{PS}}$ to obtain

$$\kappa + \alpha = a_2^{-1} \ . \tag{5.2}$$

Next, given the representation $\psi_1 = g^{a_0} \cdot g_1^{a_1} \cdot g_2^{a_2} \cdot \prod_{i \in [|Q|]} h_i^{a_{3,i}} \cdot \prod_{j \in [\ell]} \left(A_j^{a_{4,j}} \cdot B_j^{a_{5,j}} \right)$, we express it so that $\psi_1 = \prod_i R_i^{a'_i}$, where every R_i enumerates the individual $\mathbb{G}$ bases appearing in the system (i.e., $R_0 = g, R_1 = g_1, \ldots$).

Observe that for any proof π_{PS}, there must exist $E_1 \in \hat{\mathbb{G}}$ such that

$$E_1 = e(\psi_1, \tilde{X} \cdot \tilde{Y_1}^{\kappa} \cdot \tilde{Y_2}^{\alpha}) = e(\psi_1, \tilde{g})^{x + y_1 \kappa + y_2 \alpha} = e(\psi_1, \tilde{g})^{\tau} \ ,$$

where $\tau \coloneqq x + y_1 \kappa + y_2 \alpha$. Moreover, with probability $1 - 1/p$, the algebraic adversary must have queried the random oracle for E_1 to generate the challenge c_{PS}. Thus, it must have also output a representation with respect to pairings over known group elements. Thus, we can write $E_1 = \prod_{i,j} e(S_i, \tilde{S}_j)^{b_{i,j}}$, where each S_i and $\tilde{S}_j$ themselves have known representations in their respective

group bases. By bilinearity and the AGM-expansions of the $\mathbb{G}$ and $\tilde{\mathbb{G}}$ bases, we may re-write E_1, by splitting and recombining the pairings so that each resulting individual pairing has a distinct R_i as the $\mathbb{G}$ term. Let $E_1 = \prod_{i,j} e(R_i, \tilde{R}_j)^{b'_{i,j}}$ be the resulting representation. Then we have that

$$\prod_{i,j} e(R_i, \tilde{R}_j)^{b'_{i,j}} = E_1 = e(\psi_1, \tilde{g})^{\tau} = e\left(\prod_i R_i^{a'_i}, \tilde{g}\right)^{\tau} = \prod_i e(R_i, \tilde{g})^{a'_i \tau} .$$

Rearranging,

$$\prod_{i,j} e(R_i, \tilde{R}_j)^{c_{i,j}} = 1 \pmod{p} , \tag{5.3}$$

where $c_{i,j} = b'_{i,j} - a'_i \tau \cdot \mathbf{1}_{\{j=k\}}$. Now, let $\hat{g} = e(g, \tilde{g})$, then by bilinearity, we have for all i, j that

$$e(R_i, \tilde{R}_j) = e(g, \tilde{g}_0)^{\beta_i \gamma_j} = \hat{g}^{\beta_i \gamma_j} ,$$

for some indeterminate $\beta_i, \gamma_j \in \mathbb{Z}_p$. Passing Equation (5.3) to the discrete-log base $\hat{g}$ under this substitution then gives

$$\sum_{i,j} \beta_i \gamma_j c_{i,j} = \sum_i \beta_i \left(\sum_j c_{i,j} \gamma_j\right) = 0 .$$

Viewing the above as a bilinear polynomial in indeterminates $(\beta_i)_i$ and $(\gamma_j)_j$, we observe that one of the following two cases must be true:

- If the coefficient $c_{i,j}$ equals zero for all i, j, then necessarily $b'_{i,j} = a'_i \tau$ when $j = k$ and $b'_{i,j} = 0$ for $j \neq k$, and extraction succeeds deterministically by picking any i with $a'_i \neq 0$ and computing

$$\tau = b'_{i,k}/a'_i \,. \tag{5.4}$$

In this case, solving (5.2) and (5.4) gives us (κ, α).

- Otherwise we have a nonzero bilinear polynomial and by a simple application of the Schwartz–Zippel lemma, we get that the probability (over uniformly random choices of the indeterminates β_i and γ_j) that this nonzero polynomial evaluates to zero is at most $2/p$.

Therefore, in the latter failure case the equality $E_1 = e(\psi_1, \tilde{g})^\tau$ can hold only with probability at most $2/p$ over the uniform choice of $(\beta_i)_i$ and $(\gamma_j)_j$. Combining all of the above, we get that the probability that the extractor fails to extract is at most $\frac{1}{p} + \frac{2}{p} + \nu(\lambda) + \mathsf{Adv}^{\mathsf{Snd}}_{\mathcal{A}}(\lambda)$. □

We also note that the zero-knowledge properties of Π_{U} and Π_{PS} follow directly from that of the Schnorr proof system.

5.4.2 Concrete efficiency

	# of operations	**Threshold**				
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
Issue	$6\times$	—— 0.59 ms ——				
Obtain	$4e + (5 + 2t)\times$	3.837 ms	4.036 ms	4.235 ms	4.435 ms	4.634 ms
Verify	$4e + 5\times$	—— 3.638 ms ——				

Table 5.2: The number of operations and the estimated time taken by each step of our non-interactive threshold blind signature scheme for threshold t. $\times$ refers to scalar multiplication in base group $\mathbb{G}$, $\tilde{\times}$ to the same operation in second base group $\tilde{\mathbb{G}}$, and e refers to pairing operations. A dash through a row indicates that times are the same across all values of t.

We provide some micro-benchmarks for the Issue, Obtain and Verify algorithms for our protocol in Table 5.2 to demonstrate its feasibility. All estimates are over the `blstrs` implementation of BLS12-381 curve, and were obtained using `zka.lc` [80] for an AWS EC2 m5.large instance (comparable to a mid-range home computer). On the same curve, the receiver's public key is around 112

bytes, the presignature size is 96 bytes and the final signature is around 192 bytes for a 256-bit scalar field.

Part II

Anonymous Tokens with a Private Metadata Bit

Chapter 6: Non-interactive Anonymous Tokens with a Private Metadata Bit

The Internet faces a persistent bot problem. Malicious traffic including scrapers, fraud bots, and credential stuffing attacks consume enormous server resources and pose genuine security threats. The predominant defense mechanism against such abuse is a CAPTCHA. However, these detection systems are notoriously unreliable. They flag legitimate users with high frequency, particularly those seeking privacy through VPNs, Tor networks, shared proxies, or privacy-enhancing browser extensions; so users who take reasonable steps to protect their privacy are systematically blocked from accessing basic web resources unless they repeatedly solve CAPTCHAs. This creates a powerful disincentive to anonymous browsing and fundamentally undermines privacy on the web [118].

A more principled solution is to establish trust in honest users upfront, without requiring them to repeatedly prove their authenticity. Anonymous tokens offer a concrete instantiation of this idea. In this setting, users first solve a CAPTCHA with a trusted authority (the signer) and receive an *anonymous* token in exchange. Later, the user can present this token to any service provider (the verifier) to bypass CAPTCHA challenges entirely. In effect, the signer certifies the token's validity without learning which service the user accessed, and the service accepts the token without learning the user's identity. This protocol should satisfy unforgeability, preventing users from manufacturing tokens as well as unlinkability, preventing both signer and service from linking a user across sessions.

Recent work has explored the introduction of private metadata to these tokens. While public metadata enables services to embed labels like geographical information or expiration times, private metadata takes a different approach. The signer encodes a hidden bit within the token that the user cannot observe. At redemption, the signer can extract and inspect this bit without the user's knowledge. This enables powerful applications. For instance, a signer detecting suspicious behavior can mark a token with a secret flag. Since users remain unaware of the flag, developing tools to

This chapter is based on [26].

evade detection becomes substantially harder. Most proposals supporting private metadata adopt the private verification model, where the signer participates in token redemption to recover the hidden information.

In this chapter, we formalize non-interactive anonymous token (NIAT) schemes with a private metadata bit. Eliminating the interactive exchange between user and signer during token issuance allows the tokens to be precomputed and distributed asynchronously. This change removes a fundamental bottleneck in deployment by allowing the signing services to compute tokens during off-peak hours, and operate more smoothly under load compared to existing solutions that require online signing. We also propose an efficient construction from structure preserving signatures on equivalence classes (SPS-EQ) [88] based on pairing-friendly groups and prove it secure under standard cryptographic assumptions. We also provide an implementation and validate the practical performance of our scheme experimentally.

Mostly for theoretical interest, we also sketch a candidate NIAT construction with full unlinkability using leveled homomorphic encryption (LHE) [144] inspired by the corresponding non-interactive blind signature construction from Chapter 3 that gives optimal size tokens at the cost of a large setup.

Use case: improved CDN access

A major advantage of our main construction is the elimination of client and signer interaction during token issuance. This enables offline pre-computation, leading to substantial savings when a signing server handles large request volumes.

Consider the canonical application of anonymous tokens in privacy-preserving CDN access, which inspired the Privacy Pass protocol [68]. Current CDN deployments operate as follows: when a client requests content, an attestation server evaluates the request and presents a CAPTCHA if the origin appears untrustworthy. The client solves the puzzle and returns the solution. The attestation server verifies correctness and either forwards the request to the origin web server for dynamic content or serves cached static pages. This approach effectively mitigates bot traffic and DDoS attacks, and it has proven durable in practice.

Unfortunately, this solution sacrifices privacy entirely. Users cannot browse anonymously without revealing their identity and access patterns to the attestation server. Privacy Pass addressed this gap by allowing clients to include blinded tokens with their CAPTCHA solutions. Upon verification, the signer signs these tokens on-the-fly and returns them to the client. Later, when accessing protected web servers, the client redeems tokens to bypass CAPTCHAs while remaining unlinkable to the signer. The protocol imposes a limit of approximately 30 tokens per session to prevent hoarding and redistribute attacks.

The fundamental limitation is that online signing is expensive. When a single signer serves millions of clients simultaneously, the cost of signing tokens on-demand becomes a bottleneck. This concern motivated fundamental shifts in protocol design, most famously in DNSSEC.

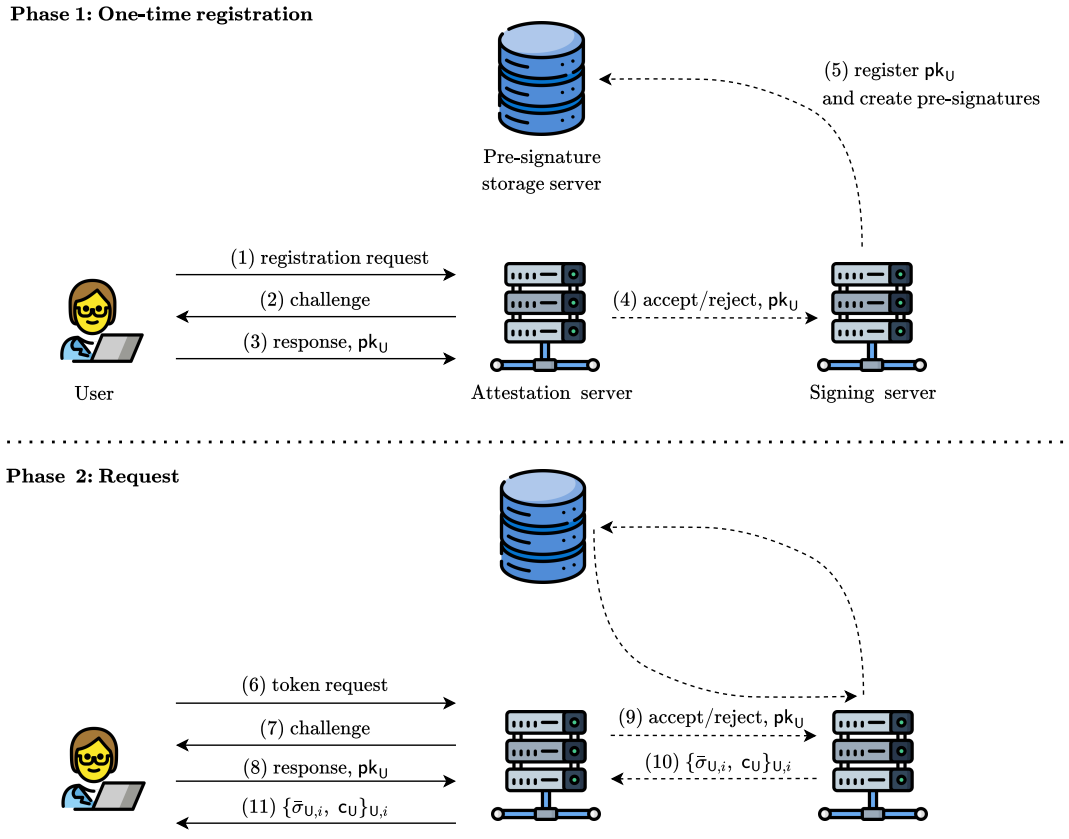

Figure 6.1: An illustration of NIAT issuance for CDNs with offline signing.

Our NIAT construction circumvents this constraint by eliminating online signing entirely (Fig. 6.1). We propose a two-phase deployment model. During registration, clients prove their legitimacy by solving CAPTCHAs and submit their public key to the signer. The signer verifies the client once, then offline pre-computes and stores a batch of pre-signatures. This one-time interaction per client is minimal. Subsequently, when a client requests tokens during the request phase, they obtain pre-signed tokens immediately without waiting for online computation, ensuring a ready supply for all registered clients. The amortized cost of issuance then depends only on client authentication and network latency, not on the volume of tokens requested.

6.1 Prior Work on Anonymous Tokens

Privacy Pass [68] introduced the idea of one-time use anonymous tokens as a method to prevent DDoS attacks while enhancing privacy and ensuring a seamless user experience for those who access web servers protected by Cloudflare infrastructure.

In the simplest setting, Privacy Pass is an interactive, two-move, client-initiated protocol which relies on Verifiable Oblivious Pseudorandom Functions (VOPRFs), and is secure in the random oracle model. The recent partially oblivious PRFs [164] may be used in place of VOPRFs for a more efficient and flexible instantiation of Privacy Pass. One disadvantage of the Privacy Pass protocol is that if a client is deemed malicious, they will not be issued tokens and the request would be dropped. This kind of feedback may inform malicious actors of their detection, which could be leveraged to refine their methods to circumvent bot-detection mechanisms.

To address this problem, Kreuter, et al. proposed the idea of anonymous tokens with private metadata bit [123] in order to propagate trust signal from the issuers to the verifiers in a private way. They formalized the security properties of this primitive, and proposed a construction called PMBTokens which are essentially an extension of the Privacy Pass protocol to support the private metadata bit. Like Privacy Pass, PMBTokens are also based on VOPRFs and are secure in the random oracle model. Also like Privacy Pass, PMBTokens do not allow any parties other than the parties holding the secret signing keys to verify the validity of tokens or to extract the private metadata bits in the case of PMBTokens. On the other hand, publicly verifiable anonymous tokens with private

metadata bit [39] allow for public token verification under the issuer's public key, while the private metadata bit remains hidden from users and is only extractable with the issuer's secret key. Publicly verifiable anonymous tokens have also been instantiated from RSA blind signatures [15], however they do not support private metadata.

Chase, et al. [60], proposed ATHM which is an anonymous token protocol for the setting where the issuer and verifier are the same entity. ATHM is based on a symmetric key primitive, namely algebraic MACs, and is secure in the generic group model. In addition, they also revisited the definition of anonymous tokens and merged the two notions of token validity in [123]. Specifically, in the new definition, only tokens from which we can extract the embedded metadata bit successfully are considered valid tokens, and the private verification algorithm $\mathsf{AT.Verify}$ was removed for redundancy. ATHM can also be extended to support public metadata in tokens.

Anonymous Counting Tokens (ACTs) [40] are another variant of anonymous tokens where clients are not able to redeem more than one token per message with the same verifier. This security property can be achieved through two different approaches: either by allowing the issuer to detect repeated token issuance requests for the same message from the same client during issuance, or by allowing the verifier, during redemption, to detect that two different tokens for the same message were issued to the same client.

6.2 Preliminaries

In this section, we give the preliminaries necessary for this chapter.

6.2.1 Cryptographic hardness assumptions

Definition 6.1 (Inverse decisional Diffie-Hellman problem (InvDDH))**.** Let $\mathbb{G}$ be a group of order p. For $x \leftarrow\!\!\$\ \mathbb{Z}_p$ and $g \in \mathbb{G}$, the inverse decisional Diffie-Hellman problem asks to distinguish, for a randomly chosen $T \in \mathbb{G}$, the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$, where:

$$\mathcal{D}_0 \coloneqq \left(p, g, g^x, g^{1/x}\right) \qquad \mathcal{D}_1 \coloneqq (p, g, g^x, T)\ .$$

6.2.2 Structure-preserving signatures on equivalence classes

Structure-preserving signatures on equivalence classes (SPS-EQ) [88] are used to sign equivalence classes $[\mathbf{m}]$ of message vectors $\mathbf{m} \in (\mathbb{G}^*)^\ell$ for $\ell > 1$, with equivalence relation $\mathbf{m}, \mathbf{n} \in \mathbb{G}^\ell : \mathbf{m} \sim_{\mathcal{R}} \mathbf{n} \Leftrightarrow \exists s \in \mathbb{Z}_p : \mathbf{m} = \mathbf{n}^s$. Let us now recall the definition for SPS-EQ [88], adapting the notation for multiplicative groups as in [101].

An SPS-EQ scheme EQ is a tuple of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{VerKey}, \mathsf{Sign}, \mathsf{ChRep}, \mathsf{Verify})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda, 1^\ell) \to (\mathsf{sk}_{\mathsf{EQ}}, \mathsf{pk}_{\mathsf{EQ}})$. Given the security parameter 1^λ, and the length of message vectors ℓ the probabilistic key generation algorithm it outputs a key pair $(\mathsf{pk}_{\mathsf{EQ}}, \mathsf{sk}_{\mathsf{EQ}})$.
- $\mathsf{Sign}(\mathsf{sk}_{\mathsf{EQ}}, \mathbf{m}) \to \boldsymbol{\sigma}_{\mathsf{EQ}}$. Given the secret key $\mathsf{sk}_{\mathsf{EQ}}$ and a representative $\mathbf{m} \in (\mathbb{G}^*)^\ell$ the probabilistic signing algorithm outputs a signature $\boldsymbol{\sigma}_{\mathsf{EQ}}$ for the equivalence class $[\mathbf{m}]$.
- $\mathsf{ChRep}(\mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}, \mu) \to \boldsymbol{\sigma}'_{\mathsf{EQ}}$. Given the representative $\mathbf{m}$, a signature $\boldsymbol{\sigma}_{\mathsf{EQ}}$ and a scalar $\mu \in \mathbb{Z}_p$ the representative changing algorithm returns an updated message-signature pair $(\mathbf{m}', \boldsymbol{\sigma}'_{\mathsf{EQ}})$ where the new representative is $\mathbf{m}' = \mathbf{m}^\mu$ and $\boldsymbol{\sigma}'_{\mathsf{EQ}}$ is the corresponding updated signature.
- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}) \to 1/0$. Given the public key $\mathsf{pk}_{\mathsf{EQ}}$, a representative $\mathbf{m}$, and a signature $\boldsymbol{\sigma}_{\mathsf{EQ}}$ the deterministic signature verification algorithm outputs 1 (accept) or 0 (reject).

An SPS-EQ scheme must satisfy the following properties:

Definition 6.2 (Correctness)**.** For every $\lambda \in \mathbb{N}$, $\ell > 1$, $\mathbf{m} \in (\mathbb{G}^*)^\ell$, and all scalars $\mu \in \mathbb{Z}_p$ an SPS-EQ scheme satisfies *correctness* if

$$\Pr\left[\begin{array}{l|r} \mathsf{Verify}(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}) \wedge & (\mathsf{sk}_{\mathsf{EQ}}, \mathsf{pk}_{\mathsf{EQ}}) \leftarrow\!\!\$\ \mathsf{KeyGen}(\mathsf{pp}_{\mathsf{EQ}}) \\ \mathsf{Verify}\left(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}^\mu, \mathsf{ChRep}(\boldsymbol{\sigma}_{\mathsf{EQ}}, \mu)\right) & \boldsymbol{\sigma}_{\mathsf{EQ}} \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}_{\mathsf{EQ}}, \mathbf{m}) \end{array}\right]$$

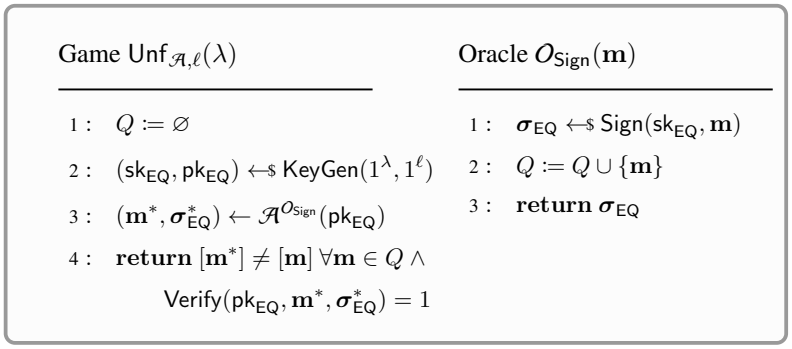

Figure 6.2: The existential unforgeability under chosen message attack experiment for SPS-EQ.

Definition 6.3 (Existential unforgeability under chosen message attack)**.** Let

$$\mathsf{Adv}^{\mathsf{Unf}}_{\mathcal{A},\ell} = \Pr\left[\mathsf{Unf}_{\mathcal{A},\ell}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an PPT adversary $\mathcal{A}$ in the Unf experiment defined in Figure 6.2. An SPS-EQ scheme over $(\mathbb{G}^*)^\ell$ is *existentially unforgeable under adaptive chosen-message* attacks if for all $\ell > 1$, and any PPT adversary $\mathcal{A}$, this advantage is at most $\mathsf{negl}(\lambda)$, for some negligible function $\mathsf{negl}(\cdot)$.

Definition 6.4 (Perfect signature adaptation under malicious keys)**.** For every $\ell > 1$ and all tuples $(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}, \mu)$ satisfying

$$\mathbf{m} \in \mathbb{G}^*\ \wedge \mathsf{Verify}(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}) = 1\ \wedge \mu \in \mathbb{Z}_p$$

an SPS-EQ scheme on $(\mathbb{G}^*)^\ell$ *perfectly adapts signatures under malicious keys* if the output of $\mathsf{ChRep}(\boldsymbol{\sigma}_{\mathsf{EQ}}, \mu)$ is a uniformly random element in the space of signatures, conditioned on Verify $(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}^\mu, \boldsymbol{\sigma}'_{\mathsf{EQ}}) = 1$.

The FHS19 SPS-EQ. In this work, we will use the pairing-based construction of SPS-EQ due to Fuchsbauer, Hanser, and Slamanig [88]. Specifically, a signature on a message $\mathbf{m} = (\mathsf{m}, \mathsf{m}_2, \ldots, \mathsf{m}_n) \in$

$(\mathbb{G}^*)^\ell$ is a triple $(Z, Y, \tilde{Y}) \in \mathbb{G} \times \mathbb{G}, \times\tilde{\mathbb{G}}$. The secret key of the signer is a tuple of ℓ elements in $\mathbb{Z}_p^*$, and the public key is an an ℓ-tuple from $\tilde{\mathbb{G}}^*$. Importantly, the signature can be adapted to another signature on message $\mathbf{m}^\mu$ without the secret key of the signer. We explain the full construction below:

- $\mathsf{KeyGen}(1^\lambda, 1^\ell) \to (\mathsf{sk}_{\mathsf{EQ}}, \mathsf{pk}_{\mathsf{EQ}})$. Given the security parameter 1^λ, and the length of message vectors ℓ the key generation algorithm samples a random vector $\mathbf{x} \leftarrow\!\!\$\ (\mathbb{Z}_p^*)^\ell$ and sets it as the secret key $\mathsf{sk}_{\mathsf{EQ}}$. It then sets the corresponding public key $\mathsf{pk}_{\mathsf{EQ}}$ as $\tilde{g}^{\mathbf{x}} = (\tilde{g}^{x_1}, \tilde{g}^{x_2}, \ldots, \tilde{g}^{x_\ell})$ and outputs both $\mathsf{sk}_{\mathsf{EQ}}$ and $\mathsf{pk}_{\mathsf{EQ}}$.

- $\mathsf{Sign}(\mathsf{sk}_{\mathsf{EQ}}, \mathbf{m}) \to \boldsymbol{\sigma}_{\mathsf{EQ}}$. Given the secret key $\mathsf{sk}_{\mathsf{EQ}} \coloneqq (x_1, x_2, \ldots, x_\ell)$ and a representative $\mathbf{m} \coloneqq (\mathsf{m}, \mathsf{m}_2, \ldots, \mathsf{m}_\ell) \in (\mathbb{G}^*)^\ell$ the signing algorithm samples a random scalar $\upsilon \leftarrow\!\!\$\ \mathbb{Z}_p$. It then computes

$$Z \coloneqq \left(\prod_{i=1}^{\ell} \mathsf{m}_i^{x_i}\right)^{\upsilon}, \quad Y \coloneqq g^{1/\upsilon} \quad \text{and} \quad \tilde{Y} \coloneqq \tilde{g}^{1/\upsilon}.$$

  Finally it returns the signature $\boldsymbol{\sigma}_{\mathsf{EQ}} \coloneqq (Z, Y, \tilde{Y})$.

- $\mathsf{ChRep}(\mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}, \mu) \to \boldsymbol{\sigma}'_{\mathsf{EQ}}$. Given the representative $\mathbf{m} \coloneqq (\mathsf{m}, \mathsf{m}_2, \ldots, \mathsf{m}_\ell) \in (\mathbb{G}^*)^\ell$, a signature $\boldsymbol{\sigma}_{\mathsf{EQ}} \coloneqq (Z, Y, \tilde{Y})$ and a scalar $\mu \in \mathbb{Z}_p$ the representative changing algorithm first runs the signature verification algorithm as $\mathsf{Verify}(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}})$ and outputs $\perp$ and aborts if it rejects. Otherwise, it samples a random scalar $\xi \leftarrow\!\!\$\ \mathbb{Z}_p^*$ and returns the updated signature $\boldsymbol{\sigma}'_{\mathsf{EQ}} \coloneqq (Z^{\xi \cdot \mu}, Y^{1/\xi}, \tilde{Y}^{1/\xi})$.

- $\mathsf{Verify}(\mathsf{pk}_{\mathsf{EQ}}, \mathbf{m}, \boldsymbol{\sigma}_{\mathsf{EQ}}) \to 1/0$. Given the public key $\mathsf{pk}_{\mathsf{EQ}} \coloneqq (\tilde{g}^{x_1}, \tilde{g}^{x_2}, \ldots, \tilde{g}^{x_\ell})$, a representative $\mathbf{m} \coloneqq (\mathsf{m}, \mathsf{m}_2, \ldots, \mathsf{m}_\ell)$, and a signature $\boldsymbol{\sigma}_{\mathsf{EQ}} \coloneqq (Z, Y, \tilde{Y})$ returns the output of

$$\prod_{i=1}^{\ell} \mathsf{e}(\mathsf{m}_i, \tilde{g}^{x_i}) = \mathsf{e}(Z, \tilde{Y}) \ \wedge \ \mathsf{e}(Y, \tilde{g}) = \mathsf{e}(g, \tilde{Y}) \ .$$

6.3 The Formal NIAT Framework

We now formally define a non-interactive anonymous tokens (NIAT). As discussed in the introduction, our definitions are inspired by existing definitions of anonymous tokens with private metadata bit (ATPM) [60, 123] properly adapted to capture the non-interactive properties of non-interactive blind signatures (NIBS) [24, 101].

A non-interactive anonymous tokens scheme consists of a tuple of polynomial time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}_\mathsf{S}, \mathsf{KeyGen}_\mathsf{U}, \mathsf{Issue}, \mathsf{Obtain}, \mathsf{Verify}, \mathsf{ReadBit})$ defined as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. The setup algorithm takes a security parameter λ as input and outputs a set of common public parameters pp. All of the remaining algorithms take pp as an input, but for notational clarity, we usually omit it as an explicit input.

- $\mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$. The signer's key generation algorithm takes public parameters pp as input, and outputs a public key pk, a secret signing key sk, and an extraction key ek.

- $\mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \to (\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})$. The user's key generation algorithm takes pp as input, and outputs a public-secret key pair $(\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})$.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{ek}, \mathsf{pk}_\mathsf{U}, b, \tau) \to (\bar{\sigma}, \mathsf{c})$. The signer runs the probabilistic algorithm Issue which takes as input the signer's secret signing key sk, the extraction key ek, the user's public key pk_U, a private metadata bit $b \in \{0, 1\}$, and public metadata τ. It then outputs a pre-signature $\bar{\sigma}$, and a random c.

- $\mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}, \tau) \to (t, \sigma)$ or $\perp$. The probabilistic Obtain algorithm is run by the user to obtain the final token. It takes as input the user's secret key sk_U, the signer's public key pk, a pre-signature $\bar{\sigma}$, and context c. It outputs a token (t, σ) if the algorithm runs successfully, or $\perp$ otherwise.

- $\mathsf{Verify}(\mathsf{pk}, t, \sigma, \tau) \to 0/1$. The deterministic public verification algorithm takes as input the signer's public key pk, the token (t, σ) and the public metadata τ, and outputs 0 or 1.
- $\mathsf{ReadBit}(\mathsf{ek}, t) \to b \in \{0,1\}$ or $\perp$. On input the signer's extraction key and a tag t, the deterministic ReadBit algorithm outputs a bit $b \in \{0,1\}$ or $\perp$.[1]

The reader will observe that our interface has two algorithms that could signal whether or not a token is valid. Namely, the Verify and ReadBit algorithms. However, as pointed out by [60], having two different notions of validity can potentially open up new attack vectors against an anonymous token protocol. For instance, a malicious user could forge tokens that yield a valid bit, but fail verification or vice-versa. This can be concerning in situations where distinct parties are performing ReadBit and Verify. To address this problem, we will insist that the correctness of the scheme hold only when the outcomes of ReadBit and Verify are consistent. In practice, this means that the ReadBit algorithm should only be run on tags whose corresponding tokens pass Verify. This unifies our two notions of validity and facilitates public verification, which is necessary for applications where token verification is outsourced to a standalone server.

It must further satisfy the following properties:

Definition 6.5 (Correctness)**.** For every security parameter $\lambda \in \mathbb{N}$, for every $b \in \{0,1\}$ and public metadata $\tau \in \{0,1\}^\lambda$, a non-interactive anonymous tokens scheme satisfies *correctness* if

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{pk}, t, \sigma, \tau) = 1 \wedge \\ \mathsf{ReadBit}(\mathsf{pk}, \mathsf{ek}, t, \sigma, \tau) = b \end{array} : \begin{array}{l} \mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}) \\ (\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \\ (\bar{\sigma}, \mathsf{c}) \leftarrow\!\$\ \mathsf{Issue}(\mathsf{sk}, \mathsf{ek}, \mathsf{pk}_\mathsf{U}, b, \tau) \\ (t, \sigma) \leftarrow\!\$\ \mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}, \tau) \end{array}\right] = 1$$

[1] Note that ReadBit is a deterministic function of the signer's secret extraction key and the tag. Additionally, as discussed in the introduction, ek could be potentially delegated to "premium" verifiers. This delegation process is orthogonal to the core scheme.

Given that our scheme is non-interactive, we formalize the notion of *reusability* for NIAT based on a similar notion from the NIBS literature [24] (also see Appendix A). Reusability allows us to capture the meaningful property that a signer should be able to issue multiple tokens to the same user's public key pk_U. In other words, a signer can *reuse* a user's public key pk_U to issue tokens non-interactively.

We note that this property is not needed in the ATPM definition given by Chase et al. [60] because their scheme is interactive. More precisely, in an ATPM scheme, whenever a user initiates a token issuance session, the user sends to the signer a query which suffices to uniquely identify that session. This is not possible in the non-interactive setting, as the only information that the signer knows about a user is its public key pk_U.

Definition 6.6 (Reusability)**.** A non-interactive anonymous token scheme is reusable if for every security parameter $\lambda \in \mathbb{N}$, metadata bit $b \in \{0,1\}$ and public metadata $\tau \in \mathcal{M}$ the following probability is negligible

$$\Pr\left[\begin{array}{l|r} & \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ & (\mathsf{pk},\mathsf{sk},\mathsf{ek}) \leftarrow\!\!\$\ \mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}) \\ \mathsf{c}_0 = \mathsf{c}_1 \ \vee\ t_0 = t_1\ : & (\mathsf{pk}_\mathsf{U},\mathsf{sk}_\mathsf{U}) \leftarrow\!\!\$\ \mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \\ & \forall i \in \{0,1\} : (\bar{\sigma}_i, \mathsf{c}_i) \leftarrow\!\!\$\ \mathsf{Issue}\left(\mathsf{sk},\mathsf{ek},\mathsf{pk}_\mathsf{U}, b, \tau\right) \\ & \forall i \in \{0,1\} : (t_i, \sigma_i) \leftarrow\!\!\$\ \mathsf{Obtain}\left(\mathsf{sk}_\mathsf{U},\mathsf{pk},\bar{\sigma}_i,\mathsf{c}_i,\tau\right) \end{array}\right]$$

We now define one-more unforgeability for NIAT. Our adversary is given access to an issuance oracle $\mathcal{O}_{\mathsf{Issue}}$ that allows it to request pre-signatures (and the corresponding contexts) for any user public key pk_U, metadata bit $b \in \{0,1\}$, and public metadata τ of its choice. It is also given access to an oracle $\mathcal{O}_{\mathsf{ReadBit}}$ for the bit extraction algorithm. We consider three types of NIAT forgeries:

- **Type 1 (Token forgery).** The adversary breaks unforgeability if, after receiving ℓ pre-signatures from $\mathcal{O}_{\mathsf{Issue}}$, it outputs more than ℓ valid tokens (t, σ) under the same public metadata τ^*. This

corresponds to the standard "one-more" forgery notion for blind-signature-based constructions.

- **Type 2 (Public-validity consistency).** The adversary also breaks unforgeability if it outputs a token (t, σ) such that $\mathsf{Verify}(\mathsf{pk}, t, \sigma, \tau^*) = 1$ but $\mathsf{ReadBit}(\mathsf{ek}, t) = \perp$ on any public metadata τ^* of its choice. This guarantees, that public verification and private bit extraction define compatible notions of validity, i.e., any token accepted by Verify must have a well-defined embedded metadata bit. This property is not needed in [60], but is important in publicly verifiable settings, where verification may be performed by parties that do not hold the extraction key.

- **Type 3 (Metadata bit integrity).** Finally, the adversary breaks unforgeability if it violates the integrity of the embedded metadata bit. Concretely, let ctr_0 and ctr_1 denote the number of issuance queries made by the adversary for bits 0 and 1, respectively, under public metadata τ^*. The adversary wins if it outputs a set of valid tokens for which the number of tokens extracting to some bit $b \in \{0, 1\}$ exceeds ctr_b. In other words, the adversary cannot produce more valid tokens carrying bit b than the number of times it queried that specific bit during issuance.

As with the existing definition of [60], we stress that we do not aim to rule out 'one-for-one' bit switching, where an adversary transforms a token issued under bit 0 into a token under bit 1 and vice-versa, preserving the *total* number of tokens per bit. Such transformations do not violate the above counting condition. However, in practice such a transformation does not give the adversary any additional power as it does not increase the number of usable tokens of any type, nor does it enable amplification of trust signals or bypass of system policies. In particular, any meaningful attack would require a *net increase* in tokens of a given bit, which is precisely what our definition rules out.

Unlike the definition of [60], where the adversary commits to a single target bit and all forgeries must correspond to that bit, our definition allows the adversary to produce tokens under *both* bits

without prior declaration. This results in a more flexible formulation that avoids restricting the adversary's output to a single metadata value, while capturing the same core security requirement of preventing a net increase in valid tokens for any given bit.

Definition 6.7 (One-more unforgeability)**.** Let

$$\mathsf{Adv}^{\mathsf{OM\text{-}Unf}}_{\mathcal{A}} = \Pr\left[\mathsf{OM\text{-}Unf}_{\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 6.3. We say a NIAT scheme NIAT is one-more unforgeable if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

Game $\mathsf{OM\text{-}Unf}_{\mathcal{A}}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow\!\$\ \mathsf{KeyGen}_{\mathsf{S}}(\mathsf{pp})$
3 : $\tau^*, \{(t_i, \sigma_i)\}_{i\in[N]} \leftarrow \mathcal{A}^{O_{\mathsf{Issue}}, O_{\mathsf{Read}}}(\mathsf{pp}, \mathsf{pk})$
// Type 1: token forgery
4 : **if** $\#\{t_i\}_{i\in[N]} > \ell \wedge \bigwedge_{i\in[N]} \mathsf{Verify}(\mathsf{pk}, t_i, \sigma_i, \tau^*) = 1$ **then accept**
5 : $\mathsf{ctr}^*_{0,\tau^*} \coloneqq 0, \mathsf{ctr}^*_{1,\tau^*} \coloneqq 0$
6 : **foreach** $i \in [N]$ **do**
7 : **if** $\mathsf{Verify}(\mathsf{pk}, t_i, \sigma_i, \tau^*) = 1$ **then**
8 : $b' \leftarrow \mathsf{ReadBit}(\mathsf{ek}, t_i)$
// Type 2: public-validity consistency
9 : **if** $b' = \bot$ **then accept**
10 : $\mathsf{ctr}^*_{b',\tau^*} \leftarrow \mathsf{ctr}^*_{b',\tau^*} + 1$
// Type 3: metadata bit integrity
11 : **if** $\mathsf{ctr}^*_{0,\tau^*} > \mathsf{ctr}_{0,\tau^*} \vee \mathsf{ctr}^*_{1,\tau^*} > \mathsf{ctr}_{1,\tau^*}$ **then accept**
12 : **reject**

Oracle $O_{\mathsf{Issue}}(\mathsf{pk}_{\mathsf{U}}, b, \tau)$

1 : **if** ℓ, $\mathsf{ctr}_{0,\tau}$, $\mathsf{ctr}_{1,\tau}$ are uninitialized **then** $\ell \coloneqq 0, \mathsf{ctr}_{0,\tau} \coloneqq 0, \mathsf{ctr}_{1,\tau} \coloneqq 0$
2 : $(\bar{\sigma}, \mathsf{c}) \leftarrow\!\$\ \mathsf{Issue}(\mathsf{sk}, \mathsf{ek}, \mathsf{pk}_{\mathsf{U}}, b, \tau)$
3 : $\ell \leftarrow \ell + 1$
4 : $\mathsf{ctr}_{b,\tau} \leftarrow \mathsf{ctr}_{b,\tau} + 1$
5 : **return** $(\bar{\sigma}, \mathsf{c})$

Oracle $O_{\mathsf{Read}}(t, \sigma, \tau)$

1 : **return** $\mathsf{ReadBit}(\mathsf{ek}, t)$

Figure 6.3: The one-more unforgeability experiment for NIAT.

Next, we formalize NIAT unlinkability. The overall intuition here remains the same as in the ATPM definitions of [60], except that the adversary is now equipped with a GenUser oracle O_{GenUser} that issues new public keys, and an Obtain oracle O_{Obtain} that returns the tokens on chosen $(\bar{\sigma}, \mathsf{c}, \tau)$ triples.

Definition 6.8 (κ-unlinkability)**.** Let

$$\mathsf{Adv}^{\mathsf{UNLINK}}_{\mathcal{A},n} = \Pr\left[\mathsf{UNLINK}_{\mathcal{A},n}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ for parameter $n \in \mathbb{N}$ in the experiment defined in Figure 6.4. We say a NIAT scheme NIAT is κ-unlinkable if for any PPT adversary $\mathcal{A}$ and $n \in \mathbb{N}$, this advantage at most $\kappa/n + \mathsf{negl}(\lambda)$.[2]

Game $\mathsf{UNLINK}_{\mathcal{A},n}(\lambda)$

1 : $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(1^\lambda)$
2 : $(\mathsf{pk}, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\mathsf{pp})$
3 : $(Q^*, \tau, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{O_{\mathsf{GenUser}}, O_{\mathsf{Obtain}}}(\mathsf{state}_1)$
4 : **foreach** $(\mathsf{pk}_\mathsf{U}, \bar{\sigma}, \mathsf{c}) \in Q^*$
5 : **if** $(\mathsf{pk}_\mathsf{U}, \cdot) \notin Q_{\mathsf{usr}}$
$\vee\ (\mathsf{pk}_\mathsf{U}, \cdot, \mathsf{c}) \in Q_{\mathsf{obt}}$ **then reject**
6 : **if** $|Q^*| \neq n$ **then reject**
7 : **foreach** $i \in [n]$ **do**
8 : $\mathsf{tok}_i \leftarrow\!\$\ \mathsf{Obtain}(\mathsf{sk}_{\mathsf{U}_i}, \mathsf{pk}, \bar{\sigma}_i, \mathsf{c}_i, \tau)$
9 : **if** $\mathsf{tok}_i = \bot$ **then reject**
10 : $\hat{i} \leftarrow\!\$\ [n]$
11 : Pick a random permutation $\varphi : [n] \to [n]$
12 : $i^* \leftarrow \mathcal{A}_3\left(\mathsf{state}_2, \mathsf{tok}_{\hat{i}}, \left(\mathsf{tok}_{\varphi(i)}\right)_{i \in [n] \setminus \{\varphi^{-1}(\hat{i})\}}\right)$
13 : **if** $i^* = \hat{i}$ **then accept**

Oracle $O_{\mathsf{GenUser}}(\mathsf{pp})$

1 : **if** Q_{usr} is uninitialized **then** $Q_{\mathsf{usr}} \coloneqq \varnothing$
2 : $(\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U}) \leftarrow\!\$\ \mathsf{KeyGen}_\mathsf{U}(\mathsf{pp})$
3 : $Q_{\mathsf{usr}} \coloneqq Q_{\mathsf{usr}} \cup \{(\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})\}$
4 : **return** pk_U

Oracle $O_{\mathsf{Obtain}}(\mathsf{pk}_\mathsf{U}, \bar{\sigma}, \mathsf{c}, \tau)$

1 : **if** Q_{obt} is uninitialized **then** $Q_{\mathsf{obt}} \coloneqq \varnothing$
2 : **if** $(\mathsf{pk}_\mathsf{U}, \cdot) \notin Q_{\mathsf{usr}}$ **then return** $\bot$
3 : Retrieve sk_U from Q_{usr}
4 : $\mathsf{tok} \leftarrow\!\$\ \mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}, \tau)$
5 : **if** $\mathsf{tok} \neq \bot$ **then**
6 : $Q_{\mathsf{obt}} \coloneqq Q_{\mathsf{obt}} \cup \{(\mathsf{pk}_\mathsf{U}, \bar{\sigma}, \mathsf{c})\}$
7 : **return** tok

Figure 6.4: The unlinkability experiment for NIAT.

[2] Intuitively, the probability κ/n quantifies the best strategy for an honest signer that has κ choices for each private metadata b_i. In the case that the private metadata is a bit, ATPM (and thus also NIAT) schemes generally focus on the case where $\kappa = 2$.

Concretely, the parameter n is the number of pairwise distinct $(\mathsf{pk}_{\mathsf{U}}, \bar{\sigma}, \mathsf{c})$ triples in the challenge set Q^*. Notice that we do not enforce any other constraints on the values inside Q^*. This allows for cases where some public keys can repeat. In particular, when $n = 2$, $\kappa = 1$, and $\mathsf{pk}_{\mathsf{U}_1} \neq \mathsf{pk}_{\mathsf{U}_2}$ (resp. $\mathsf{pk}_{\mathsf{U}_1} = \mathsf{pk}_{\mathsf{U}_2}$), we can view this experiment as being analogous to the stronger versions of NIBS user (resp. context) blindness [24]. Thus we view our unlinkability definition as a generalization of the NIBS blindness definitions.

Remark 6.1. An important requirement for unlinkability is that all $(\bar{\sigma}, \mathsf{c})$ pairs in Q^* must be under the same public metadata, otherwise the property is trivially broken. This also holds for the definition given in [60].

We then define the equivalent notion of privacy for the metadata bit in a NIAT.

Definition 6.9 (Privacy of metadata bit)**.** Let

$$\mathsf{Adv}^{\mathsf{PMB}}_{\mathcal{A}} = |\Pr\left[\mathsf{PMB}_{\mathcal{A},1}(\lambda) = \mathbf{accept}\right] - \Pr\left[\mathsf{PMB}_{\mathcal{A},0}(\lambda) = \mathbf{accept}\right]|$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 6.5. We say a NIAT scheme NIAT preserves privacy of the metadata bit if for any PPT adversary $\mathcal{A}$, this advantage is negligible.

6.4 NIAT from Structure-preserving Signatures on Equivalence Classes

We now present our NIAT construction from structure-preserving signatures on equivalence classes. At a high level, in order to build NIAT we will extend the NIBS construction of [101] to additionally embed the private metadata bit into the signer's pre-signature. This is a non-trivial exercise as the bit must also be extractable from the final token, under the signer's secret key. We resolve this by hiding the bit inside the exponent of a random group element and sending the result as part of the pre-signature. Thus, when a user sends the re-randomized signature, the part corresponding to the hidden bit can be checked in the exponent using the signer's secret key.

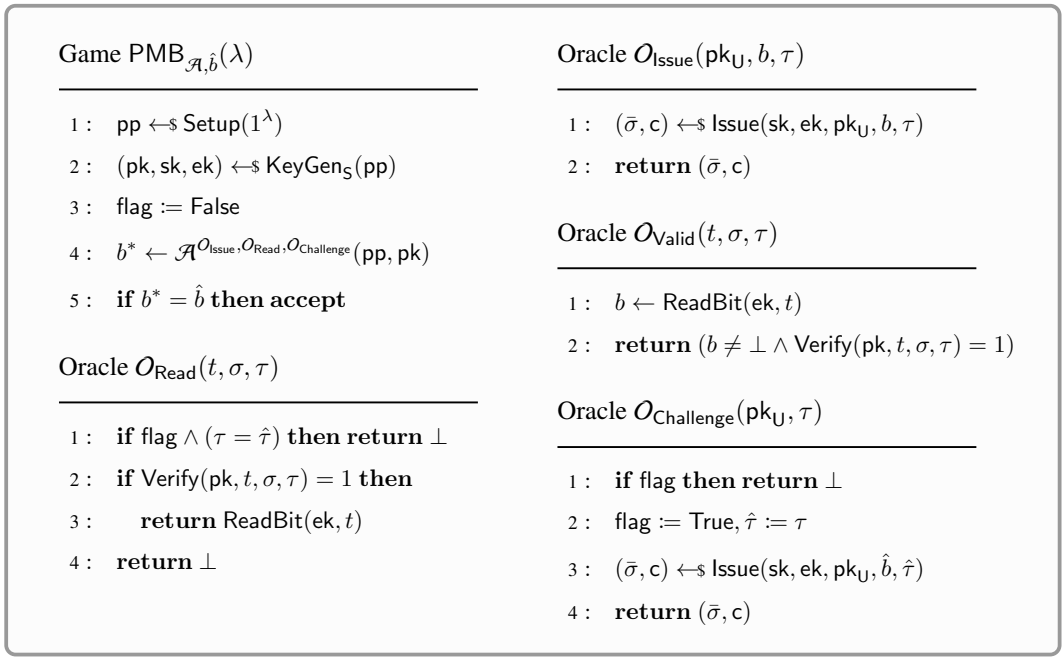

Figure 6.5: The privacy of metadata bit experiment for NIAT.

In all that follows, we will ignore the public metadata τ for simplicity, instead noting that (and as we show in the full version of this article, all our constructions are extendable to include τ in a manner similar to [101]).

Construction 6.1. Our construction requires a hash function $\mathsf{H} : \{0,1\}^\lambda \to \mathbb{G}$ modeled as a random oracle, an SPS-EQ scheme $\mathsf{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.KeyGen}, \mathsf{EQ.Sign}, \mathsf{EQ.ChRep}, \mathsf{EQ.Verify})$, and a NIZK $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}}$ as described.

System setup. This algorithm sets up the generators for the bilinear groups $\mathbb{G}$ and $\tilde{\mathbb{G}}$ of prime order p. It also runs the NIZK setup to generate the crs for the language $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}}$.

Key generation. The signer's key generation algorithm samples a random scalar vector $\mathbf{x}$ and runs the SPS-EQ setup algorithm. The user's key generation algorithm samples a random value as the secret signing and extraction keys, and sets the public key with respect to this secret.

Language $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}}$

Instance: Each instance $\mathbb{x}$ is interpreted as a collection of group elements pk and elements R, S.

Witness: Witness $\mathbb{w}$ consists of an integer vectors $\mathsf{ek} := (x_1, x_2)$, $\mathsf{sk} := (y_1, y_2, y_3)$ and a bit b.

Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$b \in \{0,1\} \ \wedge \ S = R^{(1-b)x_1 + bx_2} \ \wedge$$
$$\mathsf{pk} = (g^{x_1}, g^{x_2}, \tilde{g}^{y_1}, \tilde{g}^{y_2}, g^{y_3})$$

Pre-signature issuance and token generation. We show the pre-signature issuance and token generation protocols in detail in Figure 6.7. To issue a pre-signature, the signer creates an equivalence class signature on $(\mathsf{pk}_{\mathsf{U}}, \mathsf{H}(\mathsf{c}), S)$, where group element $S \in \mathbb{G}$ embeds the metadata bit b. The pre-signature $\bar{\sigma}$ includes the signature $\overline{\boldsymbol{\sigma}}$ and the bit embedding S, and the c is the uniformly chosen hash input c. We must additionally include a NIZK proof π in $\bar{\sigma}$, proving $b \in \{0,1\}$, the knowledge of secret keys corresponding to the public keys and the correct computation of the bit embedding. In order to obtain a redeemable token, the user first verifies the NIZK proof and the equivalence class signature. It can then transform the representation of $\overline{\boldsymbol{\sigma}}$ to the equivalence class signature on $(g, \mathsf{H}(\mathsf{c})^{\alpha^{-1}}, S^{\alpha^{-1}})$. The final token is the tag $\mathbf{t} := \left(\mathsf{H}(\mathsf{c})^{\alpha^{-1}}, S^{\alpha^{-1}}\right)$ along with the transformed signature $\boldsymbol{\sigma}$. We discuss the protocol for validation of this token next.

Public verification (token redemption). The public verification algorithm consists of verifying the equivalence class signature $\boldsymbol{\sigma}$.

Bit extraction. The bit extraction proceeds by first calling the public verification algorithm. Then, the signer simply checks which $b \in \{0,1\}$ satisfies $t_1^{x_{1+b}} = t_2$ and returns that bit if one is found, otherwise it returns an error.

We now state the main security theorem for our construction. The detailed proofs are provided in the full version of [26].

Theorem 6.10 (Security)**.** *Construction 6.1:*

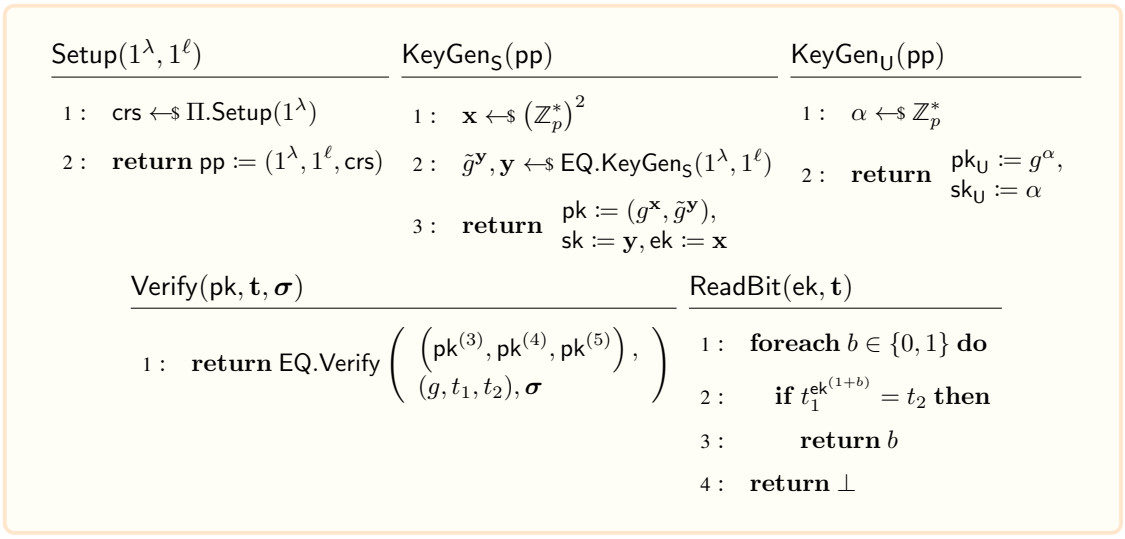

Figure 6.6: The system setup, key generation, verification and bit extraction algorithms for NIAT from SPS-EQ.

(i) One-more unforgeable in the random oracle model (Def. 6.7) assuming the NIZK proof system Π *satisfies zero knowledge and the SPS-EQ scheme* EQ *is existentially unforgeable under adaptively chosen-message attacks.*

(ii) κ*-unlinkable (Def. 6.8) in the random oracle and honest-key models for* $\kappa = 2$*, assuming the NIZK proof system* Π *is an argument of knowledge, the* InvDDH *assumption (Def. 6.1) holds in* $\mathbb{G}$ *and the SPS-EQ scheme* EQ *perfectly adapts signatures under a malicious signer.*

(iii) Private-metadata bit secure (Def. 6.9) in the random oracle model assuming the NIZK proof system Π *satisfies zero-knowledge, that* DDH *assumption holds and the SPS-EQ scheme* EQ *is existentially unforgeable under adaptively chosen-message attacks.*

Pf. sketch. To prove one-more unforgeability, we give a reduction to the existential unforgeability of SPS-EQ. Suppose the adversary succeeds by producing N valid signatures on distinct equivalence classes. Since the adversary made at most ℓ issuance queries, it must have obtained signatures on at most ℓ distinct classes. By the pigeonhole principle, at least one of the N forgeries must be a signature on a fresh equivalence class that was never queried to the EQ oracle during issuance. The reduction cannot determine which of the N forgeries is on this fresh class, so it guesses uniformly

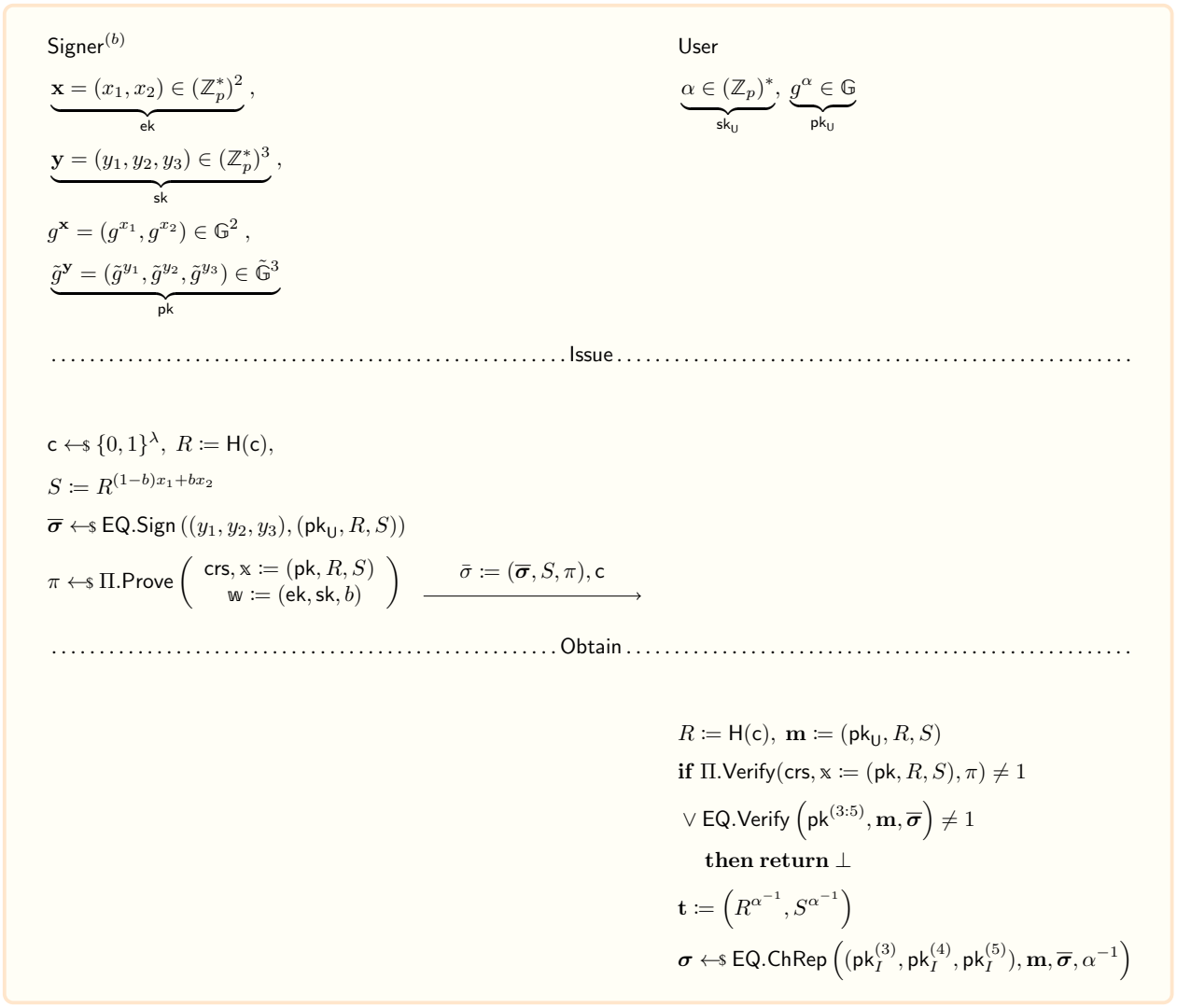

Figure 6.7: The pre-signature issuance and token obtain algorithms for NIAT from SPS-EQ. $\mathsf{pk}^{(3:5)}$ is notational shorthand for $(\mathsf{pk}^{(3)}, \mathsf{pk}^{(4)}, \mathsf{pk}^{(5)})$.

at random. If the guess is correct, it outputs a valid signature on an unqueried class, breaking EQ unforgeability. By zero knowledge of the NIZK proof system, the reduction can simulate all proofs π without knowledge of the witness, making the simulated scheme indistinguishable to the adversary.

For κ-unlinkability, we first program the random oracle so that $H(\mathsf{c}) = g_1^{\nu_\mathsf{c}}$ for random ν_c. This is indistinguishable to the adversary. Next, using the perfect adaptation property of SPS-EQ under malicious keys, we can compute the obtained signatures directly using the signer's secret key

rather than relying on the adaptation guarantee. By soundness of the NIZK proof, the adversary cannot forge a valid commitment S containing a bit different from the one it encoded, so we can extract the bit directly from S using the signer's key. The crucial step is to replace $t_1 = H(\mathsf{c})^{1/\mathsf{sk}_\mathsf{U}}$ with $t_1 = g_1^\rho$ for a uniformly random ρ. By the InvDDH assumption in $\mathbb{G}$, this replacement is indistinguishable. Intuitively, the adversary cannot distinguish between the two because computing $H(\mathsf{c})^{1/\mathsf{sk}_\mathsf{U}}$ versus sampling a random group element are indistinguishable when sk_U is hidden. After these transformations, each obtained token is completely independent of its pre-signature and context. The adversary's best strategy now is to observe the hidden bits embedded in the n tokens and guess which token corresponds to a challenge redemption. Since the tokens are independent of their derivation, the adversary can do no better than randomly guessing among all tokens with the same embedded bit, succeeding with probability at most $2/n + \mathsf{negl}(\lambda)$.

Metadata bit privacy is relatively straightforward. As with inforgeability, we program the random oracle and simulate all proofs. The key stepm however, is replacing the bit-dependent challenge S with a uniformly random group element. By the DDH assumption, this replacement is indistinguishable. Since S can be made independent of the bit, both bit values are equally likely, and the adversary's advantage is negligible. Lastly, we can use existential unforgeability of SPS-EQ to handle the edge case where the adversary can forge signatures. □

6.4.1 Instantiating the NIZK proof system

We will now show how to efficiently instantiate the NIZK proof systems for languages $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}}$ which consists of a proof of knowledge of discrete log of the elements of the signer's public key pk corresponding to its secret key sk, along with the proof that $S = R^{(1-b)x_1+bx_2}$ for x_1, x_2 in sk.

The NIZK proof proceeds as follows. The NIZK prover algorithm takes as input a common reference string crs, the statement $\mathbb{x} \coloneqq (S, T_0, T_1, U, V, W)$, where $T_0 = \mathsf{pk}^{(1)} = g_1^{x_1}, T_1 = \mathsf{pk}^{(2)} = g_1^{x_2}, U = \mathsf{pk}^{(3)} = g_2^{y_1}, V = \mathsf{pk}^{(4)} = g_2^{y_2}$ and $W = \mathsf{pk}^{(5)} = g_2^{y_3}$, and the witness $\mathbb{w} = (\mathsf{sk} \coloneqq (x_1, x_2, y_1, y_2, y_3), b)$. It does the following depending on the bit b:

- If $b = 0$, the signer (acting as the prover) samples integers $z_0, z_u, z_v, z_w c_1, a_1$, and computes commitments $\tilde{S}_0 \coloneqq R^{z_0}, \tilde{S}_1 \coloneqq R^{a_1} \cdot S^{-c_1}, \tilde{T}_0 \coloneqq g_1^{z_0}, \tilde{T}_1 \coloneqq g_1^{a_1} \cdot T_1^{-c_1}, \tilde{U} \coloneqq g_2^{z_u}, \tilde{V} \coloneqq g_2^{z_v}$ and $\tilde{W} \coloneqq g_2^{z_w}$. It then computes the challenge $c \coloneqq \mathsf{H}(g_1, g_2, \mathsf{pk}_\mathsf{U}, S, T_0, T_1, U, V, , W, \tilde{S}_0, \tilde{S}_1, \tilde{T}_0, \tilde{T}_1, \tilde{U}, \tilde{V}, \tilde{W})$, followed by $a_u \coloneqq z_u + cy_1, a_v \coloneqq z_v + cy_2, a_w \coloneqq z_w + cy_3, c_0 \coloneqq c - c_1, a_0 \coloneqq z_0 + c_0 x_1$.

- If $b = 1$, the signer samples integers $z_1, z_u, z_v, z_w, c_0, a_0$, and computes commitments $\tilde{S}_0 \coloneqq R^{a_0} \cdot S^{-c_0}, \tilde{S}_1 \coloneqq R^{z_1}, \tilde{T}_0 \coloneqq g_1^{a_0} \cdot T_0^{-c_0}, \tilde{T}_1 \coloneqq g_1^{z_1}, \tilde{U} \coloneqq g_2^{z_u}, \tilde{V} \coloneqq g_2^{z_v}$ and $\tilde{W} \coloneqq g_2^{z_w}$. It then computes the challenge $c \coloneqq \mathsf{H}(g_1, g_2, \mathsf{pk}_\mathsf{U}, S, T_0, T_1, U, V, W, \tilde{S}_0, \tilde{S}_1, \tilde{T}_0, \tilde{T}_1, \tilde{U}, \tilde{V}, \tilde{W})$, followed by $a_u \coloneqq z_u + cy_1, a_v \coloneqq z_v + cy_2, a_w \coloneqq z_w + cy_3, c_1 \coloneqq c - c_0, a_1 \coloneqq z_1 + c_1 x_2$.

Finally, it outputs the proof $\pi \coloneqq (c_0, c_1, a_u, a_v, a_w, a_0, a_1)$. We now use a small modification to the Fiat-Shamir transformation similar to [60] for efficient verification. Specifically, the user (acting as the verifier) parses $\pi =: (c_0, c_1, a_u, a_v, a_w, a_0, a_1)$ and computes $c = c_0 + c_1, \tilde{S}_0 \coloneqq R^{a_0} \cdot S^{-c_0}, \tilde{S}_1 \coloneqq R^{a_1} \cdot S^{-c_1}, \tilde{T}_0 \coloneqq g_1^{a_0} \cdot T_0^{-c_0}, \tilde{T}_1 \coloneqq g_1^{a_1} \cdot T_1^{-c_1}, \tilde{U} \coloneqq g_2^{a_u} \cdot U^{-c}, \tilde{V} \coloneqq g_2^{a_v} \cdot V^{-c}$ and $\tilde{W} \coloneqq g_2^{a_w} \cdot W^{-c}$. Finally, the user checks that $c \coloneqq \mathsf{H}(g_1, g_2, \mathsf{pk}_\mathsf{U}, S, T_0, T_1, U, V, W, \tilde{S}_0, \tilde{S}_1, \tilde{T}_0, \tilde{T}_1, \tilde{U}, \tilde{V}, \tilde{W})$.

6.4.2 Batch verification

Let us now explain the joint verification check that reduces the overall number of pairing computations required for batch verifications. Firstly notice that for both, the user and the verifier, the pairing computation for $\mathsf{e}(\mathsf{pk}_\mathsf{U}, \mathsf{pk}^{(3)})$ and $\mathsf{e}(g_1, \mathsf{pk}^{(3)})$ respectively, can be precomputed. This already reduces the number of pairing computations per pre-signature/token by one. For an issuing authority with key pair $(\mathsf{sk}, \mathsf{pk})$, let n be the number of pre-signatures issued to some user with public key pk_U, and n' be the number of tokens redeemed. Then the batch verification check for the user is given by

	# Moves	Public Verification	Private Metadata Bit	Issue	Obtain	Readbit	Token size $\lvert t\rvert + \lvert\sigma\rvert$
Const. 6.1*	1	✓	✓	$16\times$	$1\mathsf{e} + 19\times$	$3\mathsf{e} + 1\times$	$5\mathbb{G}$
[39]	3	✓	✓	$7\times$	$11\times$	$6\times$	$3\mathbb{G} + 4\mathbb{Z}$
[60]	2	✗	✓	$11\times$	$17\times$	$1\times$	$2\mathbb{G} + 1\mathbb{Z}$
[123]	2	✗	✓	$12\times$	$15\times$	$4\times$	$2\mathbb{G} + 1\mathbb{Z}$
[68]	2	✗	✗	$7\times$	$2\times$	$1\times$	$1\mathbb{G}$
[161]	2	✓	✗	$3\times$	$2\mathsf{e} + 1\times$	$2\mathsf{e}$	$1\mathbb{G} + 1\mathbb{Z}$

Table 6.1: Comparison of anonymous token schemes. $\times$ represents group exponentiation and e represents pairing operation. $\mathbb{G}$ and $\mathbb{Z}$ stand for group elements and integers respectively. An asterisk $(*)$ indicates the verification cost is amortized.

$$\prod_{i=1}^{n} \mathsf{e}\left(Z, Y_{2,i}\right) \stackrel{?}{=} \mathsf{e}\left(\mathsf{pk}_{\mathsf{U}}, \mathsf{pk}^{(3)}\right)^{n} \cdot \mathsf{e}\left(\prod_{i=1}^{n} R, \mathsf{pk}^{(4)}\right) \cdot \mathsf{e}\left(\prod_{i=1}^{n} S, \mathsf{pk}^{(5)}\right) \tag{6.1}$$

$$\wedge\, \mathsf{e}\left(\prod_{i=1}^{n} Y_{1,i}, g_2\right) \stackrel{?}{=} \mathsf{e}\left(g_1, \prod_{i=1}^{n} Y_{2,i}\right) \tag{6.2}$$

Similarly, the batch verification check for the verifier is given by replacing n by n' in (6.2) and linearly computing the other pairing check. So that instead of $5n$ (resp. $5n'$) pairings, the user (resp. the verifier) performs $n + 4$ (resp. $3n' + 2$) pairings.[3]

6.4.3 Concrete efficiency

We implemented our main NIAT Construction 6.1 in C++ using the `mcl` library [138].[4] For our implementation, we chose the pairing-friendly BLS12-381 curve [34]. All our experiments were done on a laptop equipped with an Apple M3 Pro chip and the reported numbers are the average of 1000 executions.

[3] At verification, we are cautious to not batch first part of the pairing check similar to (6.1) as the user does not produce any ZK proof for its computations.

[4] Source code available at: `https://github.com/aayux/mcl`.

Storage and communication costs. An element in $\mathbb{G}$ occupies 48 bytes in its compressed representation, and similarly, an element in $\tilde{\mathbb{G}}$ occupies 96 bytes. The message from the signer to the user consists of 3 elements in $\mathbb{G}$, 1 element in $\tilde{\mathbb{G}}$, 6 integers for the proof, and a context $\mathsf{c} \in \{0,1\}^{\lambda}$; and a token consists of 4 elements in $\mathbb{G}$ and 1 element in $\tilde{\mathbb{G}}$, we followed the `minSig` approach (signatures in $\mathbb{G}$ and signing verification keys in $\tilde{\mathbb{G}}$) to minimize the token size. With this, we estimate that in terms of communication costs, a pre-signature issuance (to the user) needs around 464 bytes to be transferred over the wire, and the final token size is around 288 bytes. Therefore, in order to design a system that incorporates the offline signing protocol shown in Figure 6.1 where the server supports m users, and each user will be issued n pre-signatures in a batch, the required total server storage would be calculated as $m \cdot (48 + 464n)$ bytes. We provide an estimation of storage needs for up to 10^8 users in Figure 6.8 (left) for batch size $n = 30, 60$ and 100. We further note that in order to prevent double spending, the system must anyway keep track of all redeemed tokens, which will require an additional storage capacity comparable to our initial storage estimation.

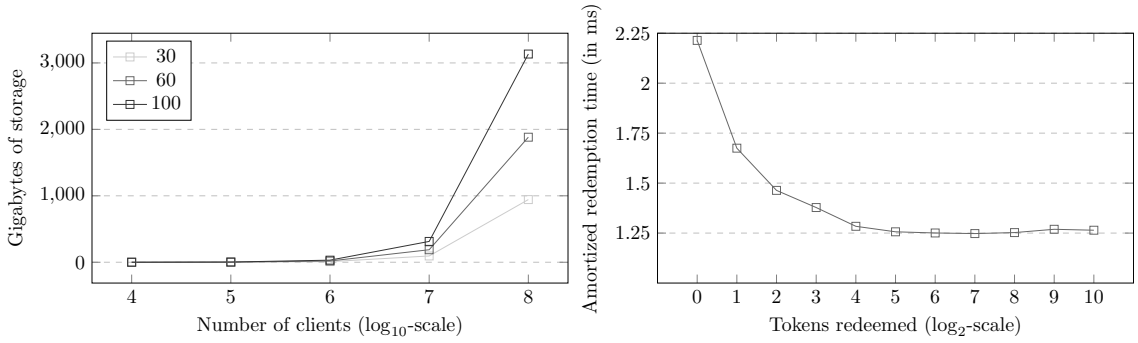

Figure 6.8: Storage estimates for an offline signing server storing batches of 30, 60 and 100 pre-signatures per user (left); and Signer's amortized redemption cost per token (right). Roughly, our protocol requires 13.6 KB to store 30 tokens per user.

Computational cost. Due to the non-interactive nature of our protocol, we are able to cut down the first message from a user to the signer. Consequently, the signer does not need to maintain an open

connection with a user for token issuance. To issue a token, the signer performs 16 exponentiations (1 for bit embedding, 6 for signature generation and 9 for the proof computation) which can be done in an *offline* manner as part of a precomputation. After receiving the signer's message, the user at the other end computes 5 pairing operations (plus one precomputed pairing of $\mathsf{e}(\mathsf{pk}_{\mathsf{U}}, \mathsf{pk}^{(3)})$) to verify the SPS-EQ signature, performs 14 exponentiations to verify the proof, and 5 more to obtain the final token. Finally, at redemption, the verifier performs 5 pairing operations (plus one precomputed pairing of $\mathsf{e}(g_1, \mathsf{pk}^{(3)})$) to verify the SPS-EQ signature, and then extracts the embedded bit with 1 exponentiation. We summarize our results (after amortization) in Table 6.1 and present them as part of a comparison with related works.

Experimental benchmarks. From our experiments we found that an exponentiation in $\mathbb{G}$ took approximately 0.047 ms, an exponentiation in $\tilde{\mathbb{G}}$ took 0.088 ms, and a pairing operation took 0.445 ms on average. We further found that (i) the signer took 0.84 ms to issue a token, (ii) the user verified the $\bar{\sigma}$ in 1.73 ms and the disjunctive NIZK proof in 0.69 ms, and finalized the token in another 0.34 ms, and (iii) during bit extraction, the signer took 1.64 ms to verify a token and additional 0.06 ms to extract the bit. It is worth noting that since our scheme is non-interactive, there is, as such, no issuance 'session' and these costs may incur asynchronously. Our results are summarized in Table 6.2. We remark that although the σ verification step is a required check before bit extraction during token redemption, these costs are reported separately because our verification algorithm can be done publicly and independently of the bit extraction computation.

	Operation	**Time** (in ms)
Signer	Issue	0.94
	Read bit (σ verification)	1.98
	Read bit (extraction)	0.06
User	Obtain ($\bar{\sigma}$ verification)	2.07
	Obtain (proof verification)	0.83
	Obtain (finalization)	0.34

Table 6.2: Benchmarks for our implementation of Construction 6.1.

Offline pre-computation. We further observe that based on the issuance times (Table 6.2), a laptop computer can pre-compute slightly over 30 million tokens overnight at 1064 users per second, thus serving more than a million users without any latency. For comparison, an interactive system like PrivacyPass handles also about 1000 users per second at peak (even without a private bit) but with a 32-core server [68]. Indeed, the throughput of our scheme will be significantly higher for a commercial server computer, virtually removing the online signing bottleneck during peak hours. This highlights an enormous advantage gained by our non-interactive system over interactive alternatives.

Batched verification. We also ran benchmark tests for a batched version of signature verification as explained above. In this batched version the user (resp. the signer) is able to perform $n + 4$ (resp. $2n + 2$) pairings instead of $4n$, for a batch of n pre-signatures (resp. tokens). As shown in Table 6.1, we calculated the amortized cost of our token generation to be approximately equivalent to 1 pairing (and 17 exponentiations).

Batch size	1	30	60	100
Time (in ms)	2.07	0.28	0.25	0.24

Table 6.3: User's amortized runtimes for batch verification.

In Table 6.3 we show that actual costs amortized over $30, 60$ and 100 pre-signatures based on experimental evaluations averaged over 1000 runs. These numbers were chosen based on PrivacyPass. A similar calculation gives the amortized cost of our token redemption (verification) to be approximately equivalent to 2 pairings, (plus 1 exponentiation for bit extraction). In Figure 6.8 (right) we show the actual amortized costs over up to 2^{12} token redemptions.

6.5 Double-spend Identification

The construction presented in the previous section requires online verification to prevent to ensure that the same token is not redeemed twice, a problem known as 'double-spending'. While this aligns with prior work on anonymous tokens with private metadata bit, the online verification requirement

becomes problematic in practice. In scenarios involving multiple independent verifiers, maintaining a synchronized record of all spent tokens across verifiers is operationally challenging.

In this section, we extend NIAT to support for double-spending *identification*. This capability enables offline verification, allowing tokens to be presented without immediate uniqueness checks. When a token is redeemed twice, the double-spending can be identified after the fact, permitting post-event detection and accountability.

Moreover, detecting double-spend is only a partial solution as the misbehaving user goes unpenalized. Private metadata bit and double-spending identification reinforce each other in both-directions. Consider a central server recording spent tokens with periodic resets. A signer can leverage information from detected double-spending to adaptively assign the private bit in future issuances, for example by marking users with frequent or suspicious spending patterns. Conversely, the private bit can inform verification practices. Authorized verifiers with access to the bit may apply stricter checks or consult the spent-token log before granting access when a token carries a low-trust indicator.

Finally, while one-more unforgeability ensures that a user cannot derive multiple tokens from a single pre-signature issuance, NIAT does not inherently prevent a user from redeeming the same token multiple times. One might initially worry that if a signer detects double-spending through token tracking, unlinkability would prevent identifying the offending user. However, this is not the case—unlinkability is defined with respect to a *fixed* user across distinct contexts, so a NIAT scheme can publicly identify a user attempting to redeem the same token multiple times while remaining consistent with our unlinkability definition.

6.5.1 Extending the NIAT framework for double-spend identification

In order to enable detection of douvle-spending users, we must introduce additional machinery to our NIAT framework. We begin, first, by highlighting the main syntactical changes to our definition. A non-interactive anonymous tokens scheme consists of a tuple of polynomial time algorithms (Setup, $\mathsf{KeyGen_S}$, $\mathsf{KeyGen_U}$, Issue, Obtain, Verify, $\mathsf{ReadBit}$) (values in gray are required only for NIAT with double-spend protection) defined as follows:

- $\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pp}, \mathsf{aux})$. The setup algorithm takes a security parameter λ as input and outputs a set of common public parameters pp and some auxiliary data aux.

- $\mathsf{KeyGen}_\mathsf{S}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$. Same as before.

- $\mathsf{KeyGen}_\mathsf{U}(\mathsf{pp}) \rightarrow (\mathsf{pk}_\mathsf{U}, \mathsf{sk}_\mathsf{U})$. Same as before.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{ek}, \mathsf{pk}_\mathsf{U}, b, \tau, \mathsf{aux}) \rightarrow (\bar{\sigma}, \mathsf{c}, \mathsf{aux})$. The signer runs the probabilistic algorithm Issue which takes as input the signer's secret signing key sk, the extraction key ek, the user's public key pk_U, a private metadata bit $b \in \{0, 1\}$, public metadata τ and some auxiliary data aux. It then outputs a pre-signature $\bar{\sigma}$, a random c and the updated auxiliary data aux.

- $\mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}, \tau, \mathsf{id}) \rightarrow (t, \sigma)$ or $\perp$. The probabilistic Obtain algorithm is run by the user to obtain the final token. It takes as input the user's secret key sk_U, the signer's public key pk, a pre-signature $\bar{\sigma}$, context c and and a verifier identifier id. It outputs a token (t, σ) if the algorithm runs successfully, or $\perp$ otherwise.

- $\mathsf{Verify}(\mathsf{pk}, t, \sigma, \tau) \rightarrow 0/1$. The deterministic public verification algorithm takes as input the signer's public key pk, the token (t, σ) and the public metadata τ, and outputs 0 or 1.

- $\mathsf{ReadBit}(\mathsf{ek}, t) \rightarrow b \in \{0, 1\}$ or $\perp$. On input the signer's extraction key and a tag t, the deterministic ReadBit algorithm outputs a bit $b \in \{0, 1\}$ or $\perp$.

- $\mathsf{DSIden}(\mathsf{pk}, t, \sigma, \sigma', \mathsf{aux}) \rightarrow (\mathsf{pk}_\mathsf{U}, \Pi)$ or $(\perp, \perp)$. The double-spend identification algorithm takes as input the signer's public key pk, a tag t, two different presentations (σ, σ') with respect to t,[5] and some auxiliary data aux, and outputs the embedded user public key and a proof of guilt $(\mathsf{pk}_\mathsf{U}, \Pi)$ or $(\perp, \perp)$.

- $\mathsf{DSVer}(\mathsf{pk}_\mathsf{U}, \Pi, \mathsf{aux}) \rightarrow 1/0$. The double-spend verification algorithm takes a user public key pk_U, a proof of guilt Π and the auxiliary data aux, and outputs a bit 1 (accept) or 0 (reject).

[5] We consider schemes where the tag t is unique during the execution of the Obtain algorithm (i.e. each tag t is uniquely determined by $(\bar{\sigma}, \mathsf{c})$), while the σ part can be prepared during presentation and differ for every possible verifier.

A NIAT scheme with double spend identification must satisfy correctness, reusability, one-more unforgeability, κ-unlinkability and privacy of the metadata bit. Of these properties, κ-unlinkability requires some care, the rest of them are the definitions are the same as defined in Section 6.3 with aux added in appropriate places.

Unlinkability for double-spend identification. Notice in the syntax above that in a NIAT scheme with double-spend identification, the verifier identifier id appears as an input to the Obtain algorithm. In particular, pre-signatures are generated by the signer with respect to $(\mathsf{pk}_{\mathsf{U}}, b, \tau)$ without knowledge of id, while the identifier is chosen later by the user at token derivation time. Thus, tokens are not issued toward specific verifier identifiers, but are instantiated for a particular identifier only at presentation time. This asymmetry is fundamental to, both, the functionality and security of double-spend identification. The role of the verifier id is to enable double-spend *detection* when the same token is used multiple times, while also ensuring that different presentations of the same token produce unlinkable views. Thus, in the real protocol, the signer does not observe id during issuance, but may learn it later when tokens are presented (e.g., via verifier-side forwarding).

A NIAT scheme with double-spend identification must satisfy correctness, reusability, one-more unforgeability, κ-unlinkability, and privacy of the metadata bit. Among these properties, κ-unlinkability requires special attention due to the double-spend identification mechanism. The remaining definitions are unchanged from Section 6.3, with aux added in appropriate places.

Unlinkability for double-spend identification. In a NIAT scheme with double-spend identification, the verifier identifier id is provided as input to the Obtain algorithm. The signer generates pre-signatures with respect to $(\mathsf{pk}_{\mathsf{U}}, b, \tau)$ without knowledge of id. The user chooses id later, at token derivation time. Consequently, tokens are not issued toward specific verifiers but are bound to a particular identifier only when presented. This asymmetry is fundamental to both the functionality and security of double-spend identification.

The verifier identifier serves two purposes. It enables double-spending detection when the same token appears multiple times, and it ensures that distinct presentations of the same token produce

unlinkable views. In the actual protocol, the signer does not learn id during issuance but may observe it later when tokens are presented, such as through verifier-side reporting.

This distinction is critical for correctly modeling unlinkability in the double-spend identification setting. If the adversary, controlling the signer in the unlinkability game, could freely choose all identifiers for challenge tokens, this would not accurately capture the intended protocol. Instead, it would permit artificial attacks where the adversary breaks unlinkability by comparing the identifier of a challenge token to its chosen identifier list. To avoid this, our unlinkability experiment fixes a common identifier id across all challenge tokens. This prevents artificial attacks while preserving the desired privacy guarantees. Specifically, the ability to link tokens of the same user across different identifiers would give the adversary a way to recover the hidden permutation in our challenge. Formally,

Definition 6.11 (κ-unlinkability)**.** Let

$$\mathsf{Adv}^{\mathsf{UNLINK}}_{\mathcal{A},n} = \Pr\left[\mathsf{UNLINK}_{\mathcal{A},n}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ for parameter $n \in \mathbb{N}$ in the experiment defined in Figure 6.9. We say a NIAT scheme with double-spend identification NIAT^+ is κ-unlinkable if for any PPT adversary $\mathcal{A}$ and $n \in \mathbb{N}$, this advantage at most $\kappa/n + \mathsf{negl}(\lambda)$.

NIAT with double-spend identification must satisfy two additional properties that we call double-spending identification and exculpability. Formally, for oracles defined as above,

Definition 6.12 (Double-spending identification)**.** Let

$$\mathsf{Adv}^{\mathsf{Iden}}_{\mathcal{A}} = \Pr\left[\mathsf{Iden}_{\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 6.10. We say a NIAT scheme NIAT^+ achieves double-spending identification if for any PPT adversary $\mathcal{A}$, this advantage is negligible.

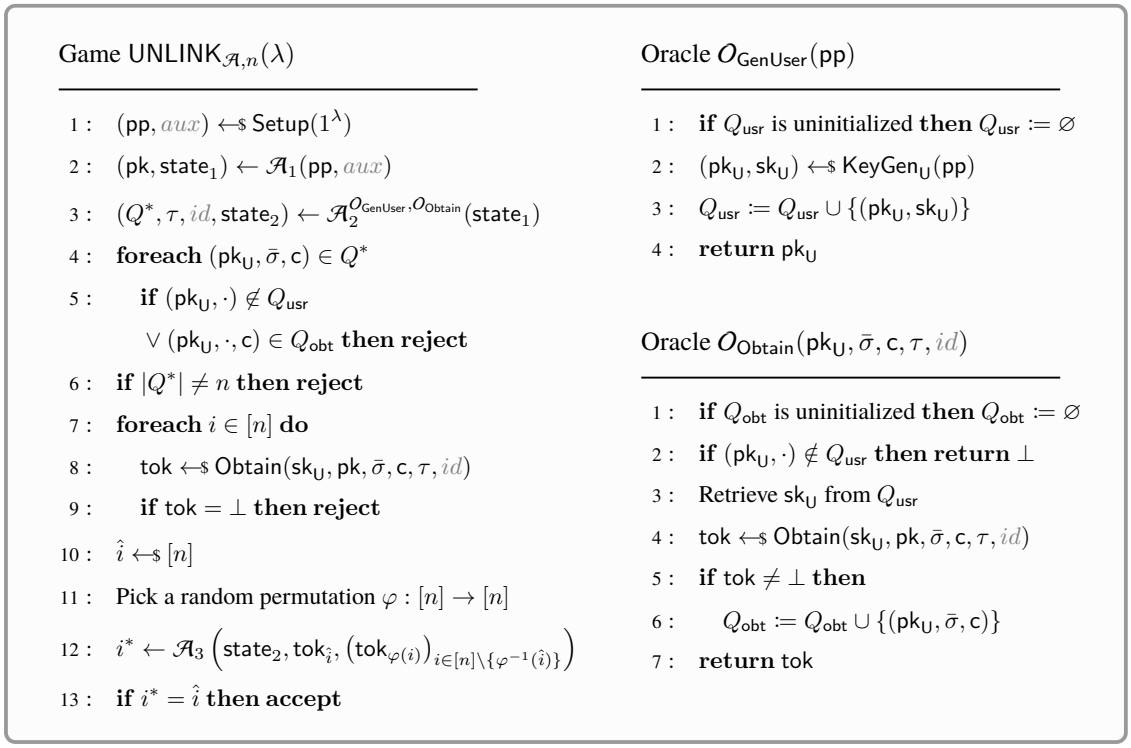

Figure 6.9: The unlinkability experiment for NIAT with double-spend identification.

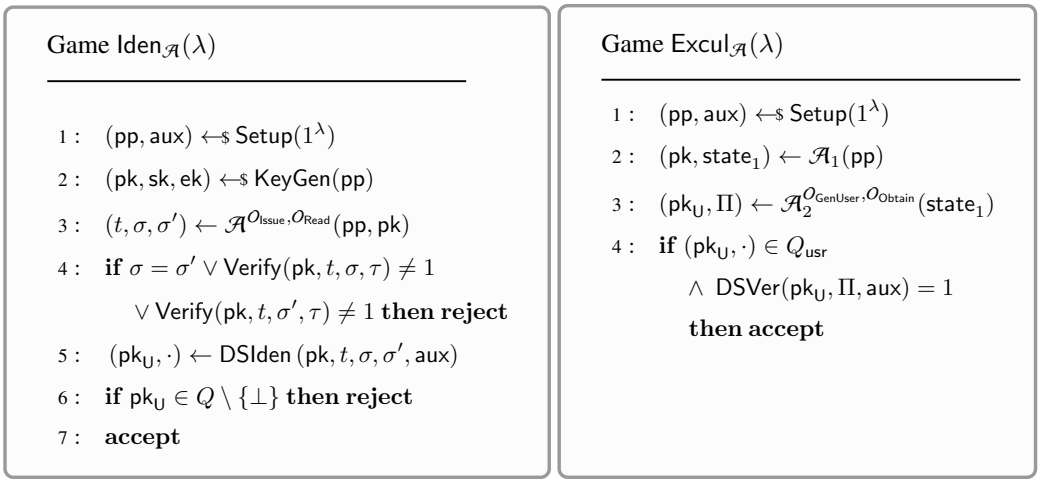

Figure 6.10: (Left) the double-spending identification experiment for NIAT. Oracles $\mathcal{O}_{\mathsf{Issue}}$ and $\mathcal{O}_{\mathsf{Read}}$ are as defined in Figure 6.3, except $\mathcal{O}_{\mathsf{Issue}}$ which stores all pk_U queried by the adversary in a list Q. (Right) the double-spend exculpability experiment for NIAT. Oracles $\mathcal{O}_{\mathsf{GenUser}}$ and $\mathcal{O}_{\mathsf{Obtain}}$ are as defined in Figure 6.4, except $\mathcal{O}_{\mathsf{Obtain}}$ behaves as an honest user that accepts a given $(\bar{\sigma}, \mathsf{c})$ pair exactly once.

Roughly, double-spending identification guarantees that no adversary is able to present the same token twice without having their public key revealed.

Definition 6.13 (Double-spending exculpability)**.** Let

$$\mathsf{Adv}^{\mathsf{Excul}}_{\mathcal{A}} = \Pr\left[\mathsf{Excul}_{\mathcal{A}}(\lambda) = \mathbf{accept}\right]$$

be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ in the experiment defined in Figure 6.10. We say a NIAT scheme NIAT achieves double-spending exculpability if for any PPT adversary $\mathcal{A}$, this advantage is negligible.

This property is essentially guaranteeing that a signer cannot wrongly accuse an honest user of double-spending.

6.5.2 Constructing NIAT with double-spend identification

We are now ready to explain the modification to Construction 6.1 that facilitates protection against double-spending by allowing a verifier to identify the offending user's public key. The core property that we are able to exploit is the fact that the pre-signature already embeds the user's public key pk_{U}, a feature unique to our non-interactive paradigm. To avoid redundancy, we omit algorithms that remain identical or trivially extendable from Construction 6.1.

Construction 6.2. Our modified construction requires hash functions $\mathsf{H}_{\mathbb{G}} : \{0,1\}^{\lambda} \to \mathbb{G}$, $\mathsf{H}_{\mathbb{Z}} : \{0,1\}^{\lambda} \to \mathbb{Z}_p$, $\mathsf{H}_{\mathbb{G}\to\mathbb{G}} : \mathbb{G} \to \mathbb{G}$, an SPS-EQ scheme $\mathsf{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Sign}, \mathsf{EQ.ChRep}, \mathsf{EQ.Verify})$, $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for languages $\mathcal{L}^{\mathsf{NIAT}_1}_{\mathsf{EQ}} \equiv \mathcal{L}^{\mathsf{NIAT}}_{\mathsf{EQ}}$ defined previously, and $\mathcal{L}^{\mathsf{NIAT}_2}_{\mathsf{EQ}}$ as defined here.

Language $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}_2}$

Instance: Each instance $\mathbb{x}$ is interpreted as group elements t_1, ϕ_1 and ϕ_2 and an integer id.
Witness: Witness $\mathbb{w}$ consists of integers sk_{U} and $\hat{\mathsf{c}}$.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\phi_2 = g^{\mathsf{id}\cdot(\mathsf{sk}_{\mathsf{U}}+\hat{\mathsf{c}})} \cdot \phi_1^{\mathsf{sk}_{\mathsf{U}}+\hat{\mathsf{c}}} \ \wedge \ t_1 = g^{(\mathsf{sk}_{\mathsf{U}}+\hat{\mathsf{c}})^{-1}}$$

Pre-signature issuance and token generation. In Figure 6.12, we show the pre-signature issuance and token generation protocols in detail, and highlight the differences from the previous construction. The main distinction in issuance is the inclusion of the integer $\hat{\mathsf{c}}$ computed as $\mathsf{H}_{\mathbb{Z}}(\mathsf{c})$ and the signature, which is now computed over $\mathbf{m} := (g, \mathsf{pk}_{\mathsf{U}} \cdot g^{\hat{\mathsf{c}}}, R, S)$. As we will see, this is the key to our double-spend identification mechanism.

To obtain a token, a user performs the same steps as before, except that instead of $\mathbf{m}^{\alpha^{-1}}$, it now computes the transformed signature over $\mathbf{m}^{(\alpha+\hat{\mathsf{c}})^{-1}}$ with the corresponding tag $\mathbf{t} := \left(g^{(\alpha+\hat{\mathsf{c}})^{-1}}, R^{(\alpha+\hat{\mathsf{c}})^{-1}}, S^{(\alpha+\hat{\mathsf{c}})^{-1}}\right)$.

Now, if a user intends to redeem its token with some verifier with public identifier id, as we show in Figure 6.12, the user's Obtain algorithm must additionally compute an ElGamal ciphertext $\boldsymbol{\phi}$ over id along with a proof π_2 of the well-formedness of the ciphertext (language $\mathcal{L}_{\mathsf{EQ}}^{\mathsf{NIAT}_2}$). Therefore, the final signature is the equivalence class signature $\mathbf{s}$, the ciphertext $\boldsymbol{\phi}$ and the proof π_2.

Public verification. The public verification algorithm verifies the equivalence class signature $\mathbf{s}$ and the NIZK proof π_2.

Double-spend identification. Suppose that a user attempts to double-spend a token (i.e., one created under the same pre-signature and context pair) with another verifier with public identifier id'. In order to do so, it must create a fresh encryption $\boldsymbol{\phi}'$ over id' and the corresponding proof π_2' with the same $\mathbf{s}$ and $\mathbf{t}$. When a system detects that a double-spend of $(\mathbf{t}, \mathbf{s})$ has occurred, it can compute

$\mathsf{DSIden}(\mathsf{pk}_\mathsf{S}, \mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{aux})$

1 : **if** $\mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathbf{t}, \boldsymbol{\sigma}) \neq 1 \vee$
$\mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathbf{t}, \boldsymbol{\sigma}') \neq 1$
$\vee\, \boldsymbol{\sigma} = \boldsymbol{\sigma}'$ **then return** $(\perp, \perp)$
2 : $\boldsymbol{\sigma} =: (\mathbf{s}, \boldsymbol{\phi}, \pi_2, \mathsf{id})\,,\ \boldsymbol{\sigma}' =: (\mathbf{s}', \boldsymbol{\phi}', \pi_2', \mathsf{id}')$
3 : $h \coloneqq \left(\phi_2{}^{-1} \cdot \phi_2'\right)^{(\mathsf{id}'-\mathsf{id})^{-1}}$
4 : **foreach** $(\mathsf{pk}_\mathsf{U}, \mathsf{c}) \in \mathsf{aux}$ **do**
5 : **if** $h = \mathsf{pk}_\mathsf{U} \cdot g^{\mathsf{H}_\mathbb{Z}(\mathsf{c})}$ **then**
6 : **return** $\mathsf{pk}_\mathsf{U}, \Pi \coloneqq (\mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{c})$
7 : **return** $(\perp, \perp)$

$\mathsf{DSVer}(\mathsf{pk}_\mathsf{U}, \Pi, \mathsf{aux})$

1 : $\Pi =: (\mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{c})$
2 : **if** $\mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathbf{t}, \boldsymbol{\sigma}) \neq 1 \vee$
$\mathsf{Verify}(\mathsf{pk}_\mathsf{S}, \mathbf{t}, \boldsymbol{\sigma}') \neq 1$
then return 0
3 : $\boldsymbol{\sigma} =: (\mathbf{s}, \boldsymbol{\phi}, \pi_2, \mathsf{id})\,,\ \boldsymbol{\sigma}' =: (\mathbf{s}', \boldsymbol{\phi}', \pi_2', \mathsf{id}')$
4 : **return** $(\mathsf{pk}_\mathsf{U}, \mathsf{c}) \in \mathsf{aux}$
$\wedge\ \left(\phi_2{}^{-1} \cdot \phi_2'\right)^{(\mathsf{id}-\mathsf{id}')^{-1}} = \mathsf{pk}_\mathsf{U} \cdot g^{\mathsf{H}_\mathbb{Z}(\mathsf{c})}$

Figure 6.11: The double-spend identification and verification algorithms for NIAT from SPS-EQ.

$\left(\phi_2{}^{-1} \cdot \phi_2'\right)^{(\mathsf{id}'-\mathsf{id})^{-1}} = g^{\alpha+\hat{\mathsf{c}}}$ and then obtain the offending user's public key by checking against the available contexts. Note that we have assumed distinct id's for ease of exposition, however, this is just as easily accomplished for the same verifier by hashing a timestamp into the identifier.

Notably, this step can be performed publicly by anyone and leads to a strong incentive against double-spending.[6] A simple optimization that offers significant asymptotic advantage over the linear search on aux is to instead store the value $\mathsf{pk}_\mathsf{U} \cdot g^{\mathsf{H}_\mathbb{Z}(\mathsf{c})}$ value itself, which allows the verifier running DSIden to perform efficient lookup over aux.

Double-spend verification. The double-spend verification algorithm essentially checks the identification algorithm's work. It ensures that the accused pk_U and the corresponding c are a valid pair in the auxiliary data, and confirms that the double-spend identification equation holds.

Theorem 6.14 (Security)**.** *Construction 6.1:*

[6] One could potentially alter the protocol to instead reveal the sk_U, but we consider this to be an extremely punitive measure given that DSIden is a public algorithm.

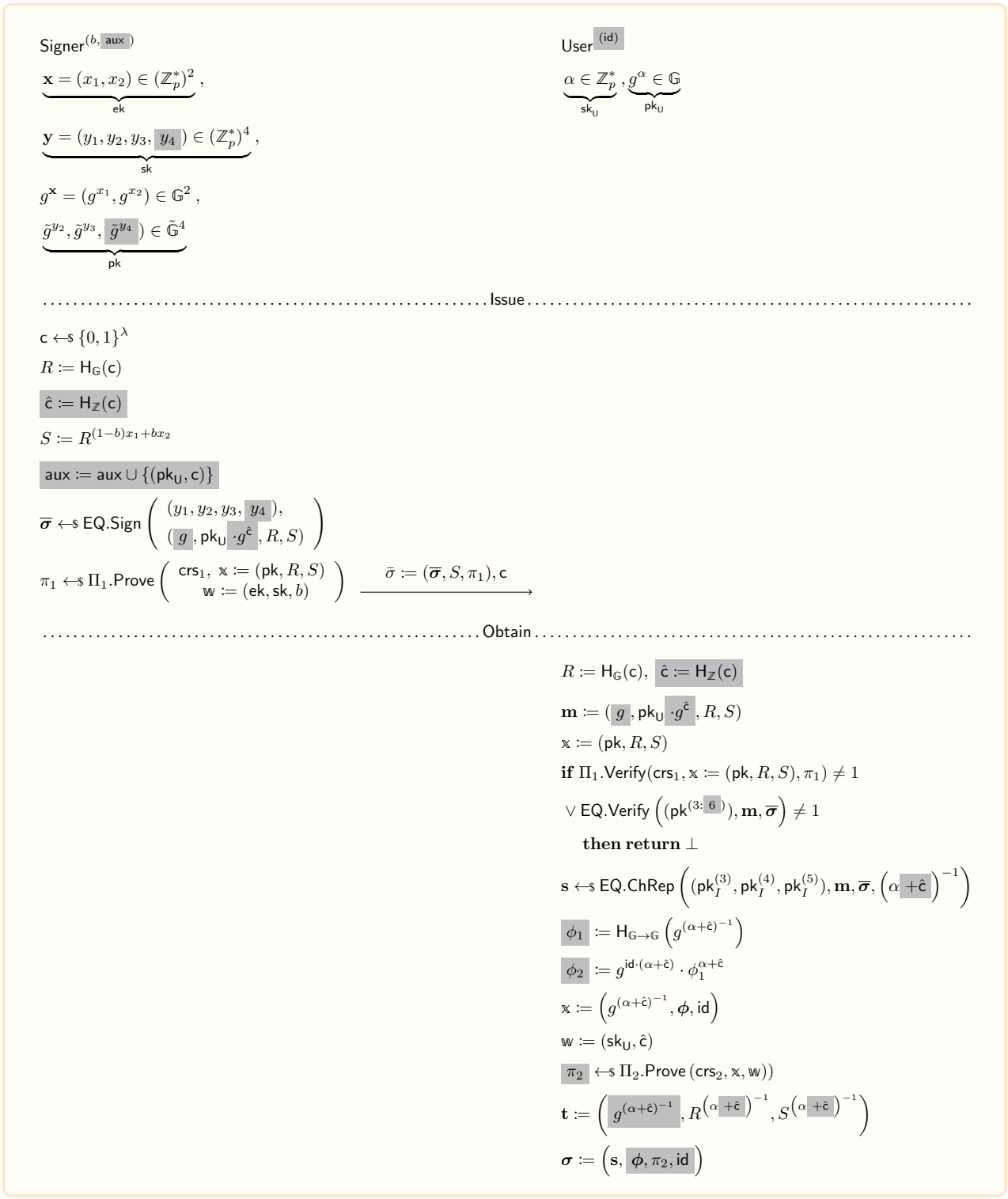

Figure 6.12: The pre-signature issuance and token obtain algorithms for NIAT with double spend identification.

(i) One-more unforgeable in the random oracle model (Def. 6.7) assuming the NIZK proof system Π_1 satisfies zero knowledge, the SPS-EQ scheme EQ is existentially unforgeable under adaptively chosen-message attacks and the NIZK proof system Π_2 is sound.

(ii) κ-unlinkable (Def. 6.8) in the random oracle and honest-key models for $\kappa = 2$, assuming the NIZK proof system Π_1 is an argument of knowledge, the DDH and qDDH assumption (Def. 5.6) holds in $\mathbb{G}$ and the SPS-EQ scheme EQ perfectly adapts signatures under a malicious signer.

(iii) Private-metadata bit secure (Def. 6.9) in the random oracle model assuming the NIZK proof system Π satisfies zero-knowledge, that DDH assumption holds and the SPS-EQ scheme EQ is existentially unforgeable under adaptively chosen-message attacks.

(iv) Double-spend identifiable in the random oracle model assuming the NIZK proof system Π_2 is sound.

(v) Double-spend exculpable assuming the NIZK proof system Π_2 is sound and the discrete log assumption holds in $\mathbb{G}$.

Pf. sketch. The one-more unforgeability and metadata bit privacy can be shown similarly to that for Construction 6.1. The approach for proving κ-unlinkability is also similar, however we now need to deal with several additional elements in the final token. Notice that due to zero-knowledge of Π_2, the proof π_2 is simulatable. Next, we can replace $\mathbf{t} := \left(g^{(\alpha+\hat{\mathsf{c}})^{-1}}, R^{(\alpha+\hat{\mathsf{c}})^{-1}}, S^{(\alpha+\hat{\mathsf{c}})^{-1}}\right)$ by $\left(g^{\rho}, (g^{\varepsilon})^{\rho}, \left(g^{\varepsilon\cdot((1-b)x_1+bx_2)}\right)^{\rho}\right)$ where we might hope to again use InvDDH to argue indistinguishability, but this does not fully work here. Instead, we make the observation that each $g^{(\alpha+\hat{\mathsf{c}})^{-1}}$ is the output of the Dodis-Yampolskiy PRF [77] which is adaptively secure under the qDDH assumption [47] for $\mathsf{poly}(\lambda)$ sized domains. This, in particular means that we must ensure that the range of $\mathsf{H}_{\mathbb{Z}}(\mathsf{c})$ is of size $\mathsf{poly}(\lambda)$. However, we remark that this does not influence security of our scheme in any way, (although resuability will now hold only with high, and not all but negligible, probability).

Lastly, viewing ϕ as ElGamal encryption of $g^{\mathsf{id}\cdot(\alpha+\hat{\mathsf{c}})}$ (times some extra randomness), we replace it with the encryption of a random value and argue indistinguishability by IND-CPA security of the

encryption scheme (which, in turn, follows from DDH). At a high level, we first set each $\mathsf{H}_{\mathbb{G}\to\mathbb{G}}(\mathsf{c}) \coloneqq g^{\nu_r}$. Then, for a query of the form $(\cdot, \mathsf{c}, \mathsf{id})$, we create the encryption $\boldsymbol{\phi}$ as $\phi_1 \coloneqq g^{\nu_r}$, and $\phi_2 \coloneqq \mu \cdot \phi_1^{\alpha}$ for message $\mu \coloneqq \mathsf{pk}_{\mathsf{U}}^{\mathsf{id}} \cdot g^{(\mathsf{id}+\nu_r)\cdot\hat{\mathsf{c}}}$. Next, we can replace $\phi_1^{\alpha} = g^{\alpha\cdot\nu_r}$ with g^{γ} for $\gamma \leftarrow\!\$\ \mathbb{Z}_p$, so that $\phi_2 = \mu \cdot g^{\gamma}$ is indistinguishable from uniform. Consequently, the final token is now independent of $\mathsf{pk}_{\mathsf{U}}, \bar{\sigma}$ and c as desired.

Double-spend identifiability follows from the soundness of Π_2. If the adversary produces two tokens with a common $\mathbf{t}$, then by soundness it must know $(\mathsf{sk}_{\mathsf{U}}, \hat{\mathsf{c}})$ and $(\mathsf{sk}'_{\mathsf{U}}, \hat{\mathsf{c}}')$ such that $t_1 = g^{(\mathsf{sk}_{\mathsf{U}}+\hat{\mathsf{c}})^{-1}}$ and $t_1 = g^{\left(\mathsf{sk}'_{\mathsf{U}}+\hat{\mathsf{c}}'\right)^{-1}}$, along with the corresponding values for ϕ_2 and ϕ'_2. Since $\mathbf{t}$ is shared between both tokens, we have $(\mathsf{sk}_{\mathsf{U}} + \hat{\mathsf{c}}) = \left(\mathsf{sk}'_{\mathsf{U}} + \hat{\mathsf{c}}'\right) \pmod{p}$. Computing $\left({\phi_2}^{-1} \cdot \phi'_2\right)^{(\mathsf{id}-\mathsf{id}')^{-1}}$ then yields $g^{\mathsf{sk}_{\mathsf{U}}+\hat{\mathsf{c}}} = g^{\mathsf{sk}'_{\mathsf{U}}+\hat{\mathsf{c}}'}$. By the structure of the protocol, at least one of $(g^{\mathsf{sk}_{\mathsf{U}}}, \mathsf{c})$ and $(g^{\mathsf{sk}'_{\mathsf{U}}}, \mathsf{c}')$ must appear in aux with $\hat{\mathsf{c}} =: \mathsf{H}_{\mathbb{Z}}(\mathsf{c})$ (and similarly for $\hat{\mathsf{c}}'$). Consequently, with all but negligible probability, $\mathsf{DSIden}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}') \neq \perp$.

In order to prove double-spend exculpability, suppose an adversary accuses some (honest) client pk_{U} and provides a proof of guilt $\Pi \coloneqq (\mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{c})$ where $\boldsymbol{\sigma} \coloneqq (\mathsf{s}, \boldsymbol{\phi}, \pi_2, \mathsf{id})$ (similarly $\boldsymbol{\sigma}'$) such that $\left(\phi_2^{-1} \cdot \phi'_2\right)^{(\mathsf{id}-\mathsf{id}')^{-1}} = \mathsf{pk}_{\mathsf{U}} \cdot g^{\mathsf{H}_{\mathbb{Z}}(\mathsf{c})}$ and, both, the proof and signature verify. Having queried O_{Issue} at most once per $(\bar{\sigma}, \mathsf{c})$ pair (without loss of generality suppose the query corresponds to the token $(\mathbf{t}, \boldsymbol{\sigma})$) then, such an adversary must have to create a satisfying proof π'_2. However, in order to do so, it either learns sk_{U}—in which case, we can reduce to the hardness of discrete log—or it is able to create a proof without sk_{U}, and we can reduce to the soundness of Π_2. □

6.5.3 Concrete efficiency

Relative to the base scheme, token issuance incurs approximately 0.15 ms additional cost due to 2 extra exponentiations (1 for the equivalence class signature and 1 for the proof) and requires about 2 additional integers in the proof. For amortized token generation, the overhead is roughly 0.6 ms, arising from 8 extra exponentiations (2 for proof verification, 1 for the tag, 2 for the proof, and 3 for encryption). The token size grows by approximately 3 additional $\mathbb{G}$ elements (1 in the tag and 2 in

the ciphertext) and 4 additional integers (3 from the client's proof and 1 for the verifier identifier). Amortized redemption requires an additional 0.4 ms, consisting of roughly 1 extra pairing and 4 exponentiations for proof verification. The cost of bit extraction remains unchanged. Double-spend identification itself demands 5 pairings plus a lookup in aux.

6.6 NIAT from Leveled Homomorphic Encryption

To close out this chapter, we provide a candidate construction for NIAT from circuit-private LHE. Our construction is inspired by the NIBS construction from Section 3.3. The idea now is that to hide a bit b the signer must now sample its own PRF key K for an *trapdoor permutation* P and homomorphically evaluate $\mathrm{P}(\mathsf{pk}_{\mathrm{P}}, \mathrm{F}_K(\mathsf{c})\|b)$ and also evaluate a signature on the same circuit it. It then sends over the evaluations along with the randomness c and a proof of well-formedness to the user, who decrypts the evaluated ciphertexts to obtain the message and the signature. To extract a bit at redemption, the signer simply inverts the tag using sk_{P} and extracts the last bit. As with the [24] scheme, our construction also yields optimal size tokens at the cost of a large setup. We give the formal construction next.

Language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIAT}}$

Instance: Each instance $\mathbb{x}$ is interpreted as signer public key pk, LHE ciphertexts ψ, $\bar{\psi}_1$ and $\bar{\psi}_2$, and a context string c.

Witness: Witness $\mathbb{w}$ consists of a signer secret key $\mathsf{sk} := (\mathsf{sk}_\Sigma, \mathsf{sk}_{\mathrm{P}})$, the private metadata bit b and the LHE randomness ρ_1 and ρ_2.

Membership: Let C_1 encode the circuit $\mathrm{P}(\mathsf{pk}_{\mathrm{P}}, \mathrm{F}_\circ(\mathsf{c})\|b)$ for public pseudorandom function F and trapdoor permutation P. Similarly, let C_2 encode the circuit $\mathsf{Sign}\,(\mathsf{sk}_\Sigma, \mathrm{P}(\mathsf{pk}_{\mathrm{P}}, \mathrm{F}_\circ(\mathsf{c})\|b))$. Then $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$b \in \{0,1\} \ \wedge \ \bar{\psi}_1 = \mathrm{HE}.\mathsf{Eval}(\mathsf{pk}, \mathsf{C}_1, \psi; \rho_1) \ \wedge \ \bar{\psi}_2 = \mathrm{HE}.\mathsf{Eval}(\mathsf{pk}, \mathsf{C}_2, \psi; \rho_2)$$

Construction 6.3 (candidate)**.** Our construction requires a keyed pseudorandom function $\mathrm{F} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$, a trapdoor permutation P, a signature scheme $\Sigma = (\Sigma.\mathsf{KeyGen}, \Sigma.\mathsf{Sign}, \Sigma.\mathsf{Verify})$, an LHE scheme $\mathrm{HE} = (\mathrm{HE}.\mathsf{KeyGen}, \mathrm{HE}.\mathsf{Enc}, \mathrm{HE}.\mathsf{Eval}, \mathrm{HE}.\mathsf{Dec})$, and a NIZKAoK system $\Pi = (\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for the following language:

System setup. The setup algorithm runs the NIZK setup algorithm for the language $\mathcal{L}_{\mathrm{HE}}^{\mathsf{NIAT}}$ to generate the crs.

Key generation. The signer's key generation algorithm runs the setup for signature scheme and that for the trapdoor permutation. The user's key generation algorithm runs the LHE key generation algorithm, and additionally samples a PRF key which it encrypts under the LHE encryption algorithm.

$\mathsf{Setup}(1^\lambda, 1^\ell)$

1 : $\mathsf{crs} \leftarrow\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$
2 : **return** $\mathsf{pp} := (1^\lambda, 1^\ell, \mathsf{crs})$

$\mathsf{KeyGen_S}(\mathsf{pp})$

1 : $\mathsf{pk}_\Sigma, \mathsf{sk}_\Sigma \leftarrow\!\$\ \Sigma.\mathsf{Setup}(\mathsf{pp})$
2 : $(\mathsf{pk}_\mathrm{P}, \mathsf{sk}_\mathrm{P}) \leftarrow\!\$\ \mathrm{P}.\mathsf{Setup}(\mathsf{pp})$
3 : **return** $\mathsf{pk} := (\mathsf{pk}_\Sigma, \mathsf{pk}_\mathrm{P})$, $\mathsf{sk} := \mathsf{sk}_\Sigma, \mathsf{ek} := \mathsf{sk}_\mathrm{P}$

$\mathsf{KeyGen_U}(\mathsf{pp})$

1 : $\mathsf{ek}_{\mathrm{HE}}, \mathsf{sk}_{\mathrm{HE}} \leftarrow\!\$\ \mathrm{HE}.\mathsf{KeyGen}(\mathsf{pp})$
2 : $K \leftarrow\!\$\ \{0,1\}^\lambda$
3 : $\psi \leftarrow\!\$\ \mathrm{HE}.\mathsf{Enc}(\mathsf{sk}_{\mathrm{HE}}, K)$
4 : **return** $\mathsf{pk} := (\mathsf{ek}_{\mathrm{HE}}, \psi)$, $\mathsf{sk} := (\mathsf{sk}_{\mathrm{HE}}, K)$

$\mathsf{Verify}(\mathsf{pk}, t, \sigma)$

1 : **return** $\Sigma.\mathsf{Verify}(\mathsf{sk}_\Sigma, t, \sigma)$

$\mathsf{ReadBit}(\mathsf{ek}, t, \sigma)$

1 : **if** $\mathsf{Verify}(\mathsf{pk}, t, \sigma) \neq 1$ **then return** $\perp$
2 : **if** $\mathrm{P}^{-1}(\mathsf{ek}, t) \neq \perp$ **then**
3 : $\quad f' \| b := \mathrm{P}^{-1}(\mathsf{ek}, t)$
4 : $\quad$ **return** b
5 : **return** $\perp$

Figure 6.13: The system setup, key generation, verification and bit extraction algorithms for NIAT from LHE.

Pre-signature issuance and token generation. We show the pre-signature issuance and token generation protocols in detail in Figure 6.14. To issue a pre-signature, the signer samples the random context c and then creates two circuits $\mathsf{C}_1 := \mathrm{P}(\mathsf{pk}_\mathrm{P}, \mathrm{F}_\circ(\mathsf{c})\|b)$ and $\mathsf{C}_2 := \Sigma.\mathsf{Sign}\left(\mathsf{sk}_\Sigma, \mathrm{P}(\mathsf{pk}_\mathrm{P}, \mathrm{F}_\circ(\mathsf{c})\|b)\right)$. Intuitively, the first circuit is the trapdoor permutation evaluation under the signer's secret, of a string comprised of the secret bit and the user's PRF evaluated on the context c; and the second circuit is a signature evaluation on the first circuit's output. The signer then sends the LHE evaluation of both circuits for the input ciphertext ψ, along with the context. We must additionally include a NIZK proof π in $\bar{\sigma}$, proving both $b \in \{0, 1\}$ as well as the well-formedness of the pre-signature. In order to redeem a token, the user simply decrypts the two ciphertexts to obtain both the message (tag) and the signature. If the signature is valid under the tag, and the NIZK proof verifies, it outputs the (t, σ) tuple as its token.

Token redemption. Validation of the token upon redemption proceeds by first verifying the signature σ. Then, the signer inverts the tag and extracts the last bit.

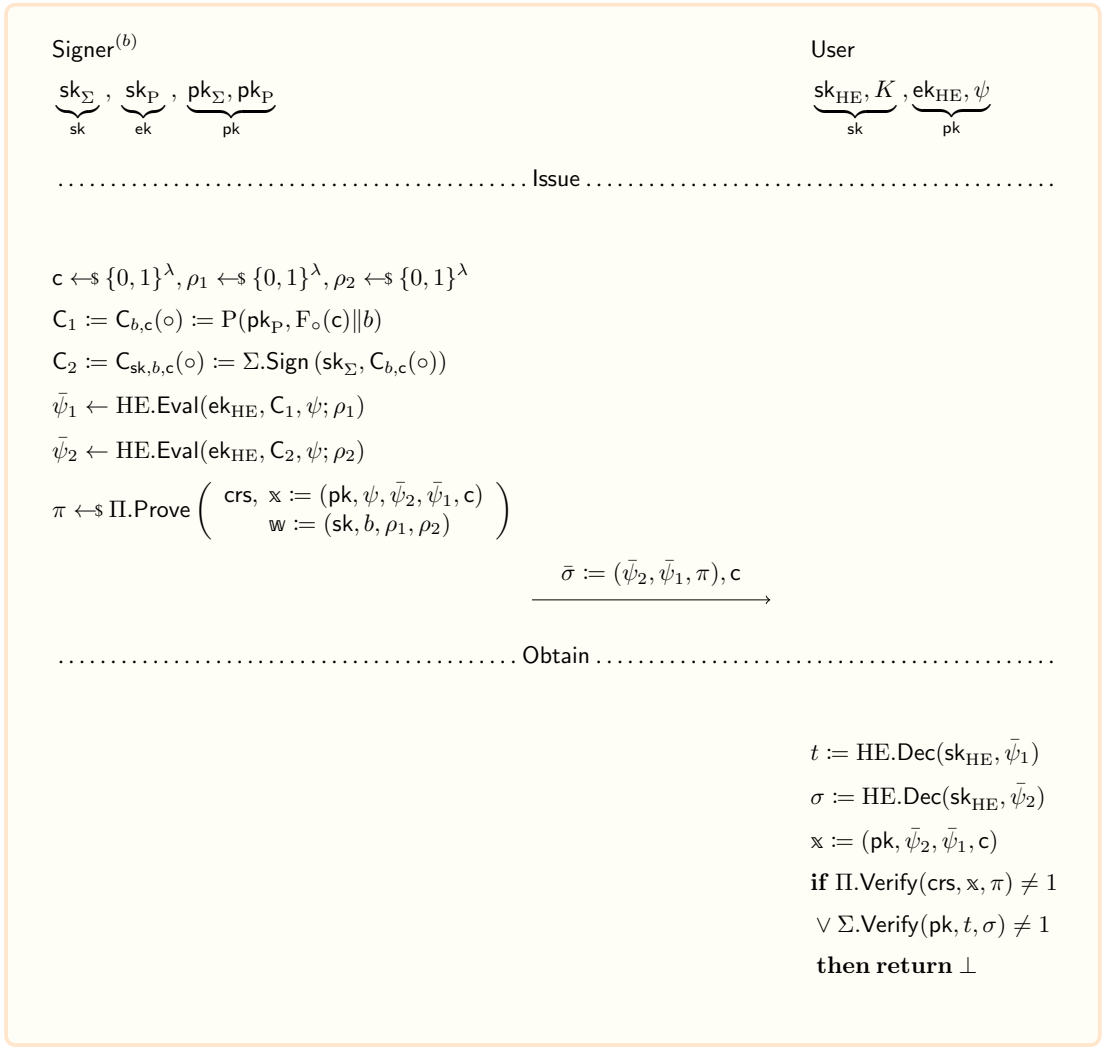

Figure 6.14: The pre-signature issuance and token obtain algorithms for NIAT from LHE.

Chapter 7: Lattice-based Anonymous Tokens with a Private Metadata Bit

As we saw in the previous chapter, several constructions of anonymous tokens with private metadata bits (ATPM) have appeared in the literature [26, 39, 60, 123, 143, 161]. These schemes support various features such as public and private verifiability, additional public metadata, non-interactive issuance, and built-in rate-limit enforcement. They are constructed using diverse cryptographic tools including verifiable oblivious pseudorandom functions, algebraic MACs, and blind signatures. Despite this progress, none of the existing designs achieve security against quantum adversaries, making post-quantum ATPM constructions an open problem. The urgency is underscored by NIST's post-quantum cryptography transition roadmap and the National Security Memorandum 10, which target 2035 for phasing out pre-quantum public-key algorithms [140].

In this chapter, we present the first ATPM construction from lattice assumptions. Our scheme is based on blind signatures and supports public token verifiability. We observe that our construction applies more broadly to upgrade any round-optimal Fischlin-type blind signature scheme [84] to an anonymous token scheme with private metadata bit. This provides a partial answer to an open problem posed by Amjad, Yeo, and Yung [15] on extending blind signatures to support private metadata bit. We also present Atlantis (Anonymous tokens from lattices with NTRU and ISIS), a practical instantiation using FALCON [152] and the efficient NIZK for linear relations from [132]. Our scheme achieves 70 KB transcripts and 129 KB token size under the hardness of standard LWE and one-more ISIS assumptions [6].

The reader will observe that unlike the previous chapters, the result in this chapter focuses on an interactive protocol for pre-signature issuance. Nevertheless, we hope that our techniques can serve as a way forward for building efficient lattice-based NIAT protocols.

This chapter is based on [30].

7.1 Prior Work on Lattice-based Anonymous Tokens

In addition to the prior works on lattice-based blind signatures discussed in Chapter 4, other recent work has explored lattice-based variants of anonymous tokens. Parno, et al. [151] construct a post-quantum secure anonymous credential scheme using a STARK-friendly Dilithium signature protocol [78] and instantiate an anonymous token scheme, though without private metadata bit support. More recently, Yan et al. [168] proposed an ATPM scheme with private verification under the hashed-MLWE assumption [41].

Lattice-based VOPRFs. Albrecht, et al. developed practical verifiable oblivious PRFs (VOPRFs) from structured lattice assumptions [12, 14]. While these techniques plausibly extend to ATPM constructions, they would only yield privately verifiable tokens, similar to other VOPRF-based schemes. We focus instead on the stronger goal of *public* verifiability and therefore pursue a blind signature based approach.

7.2 Preliminaries

In this section, we give the preliminaries necessary for this chapter.

7.2.1 Anonymous tokens with a private metadata bit

We recall the ATPM interface defined by Chase, et al. [60]. Formally, it is comprised of polynomial time algorithms Setup, KeyGen, $\mathsf{ReadBit}$ and Verify, and a polynomial time interactive token issuance protocol between User and Signer.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. The setup algorithm takes as input the security parameter λ and outputs the protocol's public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$. The signer's key generation algorithm the public parameters pp as input and generates the public key pk and secret key sk of the signer.

- $\langle \mathsf{User}(\mathsf{pk}, \mu), \mathsf{Signer}(\mathsf{sk}, b, \mu) \rangle \rightarrow (t, \sigma)$ or $\perp$. This is the interactive token issuance protocol between User and Signer where the user's input is the signer's public key pk and some public metadata $\mu \in \mathcal{M}$, and the signer's input is its secret key sk, the hidden metadata bit $b \in \{0, 1\}$ and the public metadata μ. It can be broken down into three algorithms.
  - $\mathsf{Query}(\mathsf{pk}, \mu) \rightarrow (\mathsf{query}, \mathsf{state})$. User initiates the protocol by sending query to the signer.
  - $\mathsf{Issue}(\mathsf{sk}, b, \mu, \mathsf{query}) \rightarrow \mathsf{resp}$. Signer replies with resp.
  - $\mathsf{Obtain}(\mathsf{state}, \mathsf{resp}) \rightarrow (t, \sigma)$. User locally computes the token from the signer's response.
- $\mathsf{ReadBit}(\mathsf{sk}, t, \sigma, \mu) \rightarrow b \in \{0, 1\}$ or $\perp$. The signer's bit extraction algorithm takes as input the signer's secret key sk, a token (t, σ), the public metadata μ and outputs $b \in \{0, 1\}$ or $\perp$.
- $\mathsf{Verify}(\mathsf{pk}, t, \sigma, \mu) \rightarrow 1/0$. The public verification algorithm takes as input the signer's public key pk, a token (t, σ), the public metadata μ and outputs 1 (accept) or 0 (reject).

Correctness of an ATPM scheme holds if for any given execution of the interactive protocol, the output of the Signer's ReadBit algorithm is the same as the embedded metadata bit b. As with NIAT, an ATPM scheme must satisfy one-more unforgeability, unlinkability, and privacy of the metadata bit.

One-more unforgeability ensures that for a signer running ℓ times, on some fixed (b, μ) pair, an adversary should not be able to produce more than ℓ tokens with pairwise distinct tags.

Definition 7.1 (One-more unforgeability)**.** Let

$$\mathsf{Adv}^{\mathsf{OM\text{-}Unf}}_{\mathsf{ATPM},\mathcal{A}} = \Pr[\mathsf{OM\text{-}Unf}_{\mathsf{ATPM},\mathcal{A}}(\lambda) = \mathbf{accept}]$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 7.1. We say an ATPM scheme ATPM is *one-more unforgeable* if for any PPT adversary $\mathcal{A}$ this advantage is negligible.

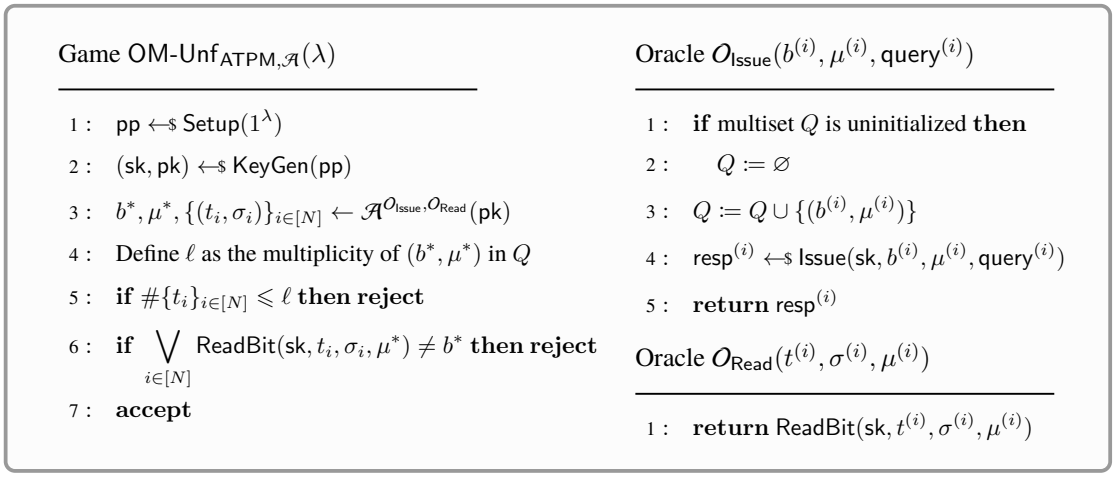

Figure 7.1: The one-more unforgeability experiment for ATPM.

Unlinkability ensures that a malicious signer can not link tokens to any given user, and more generally to the corresponding issuing sessions.

Definition 7.2 (κ-unlinkability)**.** Let

$$\mathsf{Adv}^{\mathsf{UNLINK}}_{\mathsf{ATPM},\mathcal{A},n} = \Pr[\mathsf{UNLINK}_{\mathsf{ATPM},\mathcal{A},n}(\lambda) = \mathbf{accept}$$

be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ for parameter $n \in \mathbb{N}$ in the experiment defined in Figure 7.2. We say an ATPM scheme ATPM is κ-unlinkable if for any PPT adversary $\mathcal{A}$ and $n \in \mathbb{N}$, this advantage at most $\kappa/n + \mathsf{negl}(\lambda)$.

Finally, privacy of the metadata bit ensures that an adversary can do no better than guess the hidden metadata bit with trivial probability.

Definition 7.3 (Privacy of metadata bit)**.** Let

$$\mathsf{Adv}^{\mathsf{PMB}}_{\mathcal{A}} = \left| \Pr[\mathsf{PMB}_{\mathsf{ATPM},\mathcal{A},1}(\lambda) = \mathbf{accept}] - \Pr[\mathsf{PMB}_{\mathsf{ATPM},\mathcal{A},0}(\lambda) = \mathbf{accept}] \right|$$

be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 7.3. We say an ATPM scheme ATPM preserves privacy of the metadata bit if for any PPT adversary $\mathcal{A}$, this advantage negligible.

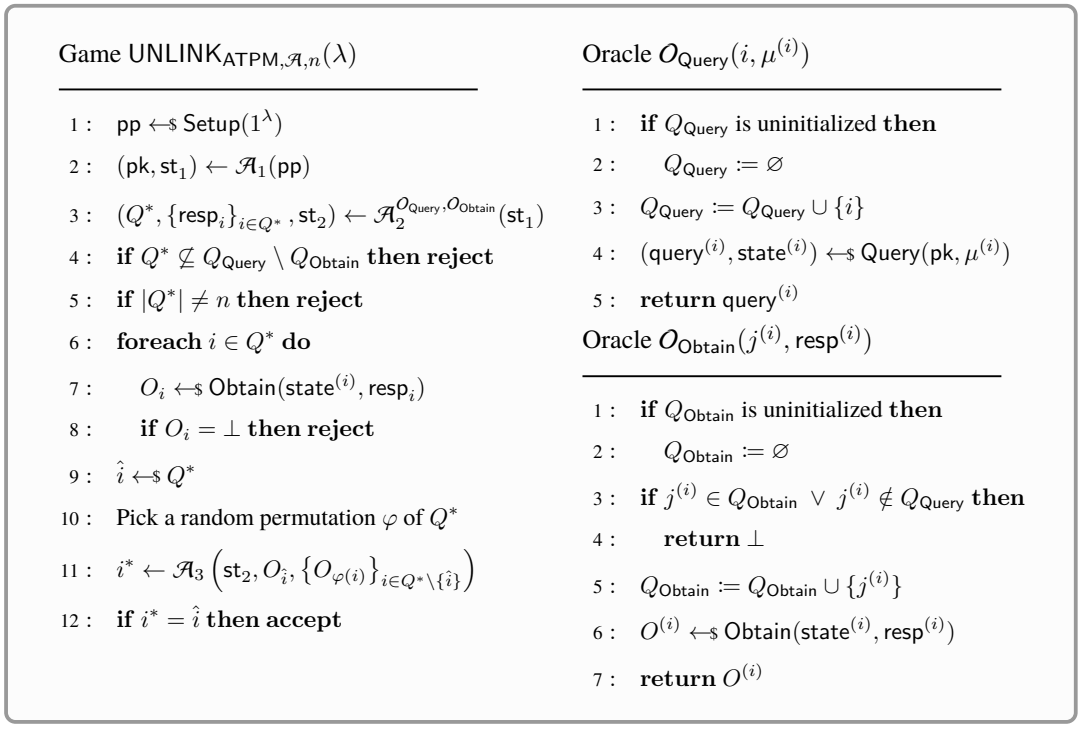

Figure 7.2: The unlinkability experiment for ATPM.

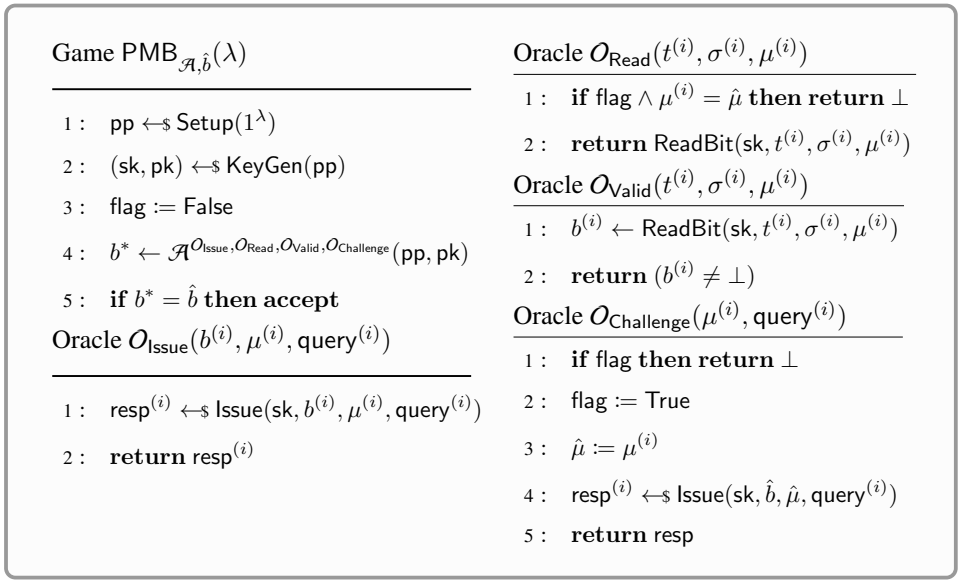

Figure 7.3: The privacy of metadata bit experiment for ATPM.

7.2.2 Linearly homomorphic public key encryption

The following is a linearly homorphic PKE scheme LinHE that is IND-CPA secure under LWE [153]:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$. The setup algorithm takes as input the security parameter λ and computes the integers $n, m, \mathfrak{s}$ and q as the functions of the security parameter. It then samples a matrix $\mathbf{B} \leftarrow\!\!\$\ \mathbb{Z}_q^{n\times m}$, and vectors $\mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n$ and $\boldsymbol{\varepsilon} \leftarrow\!\!\$\ \Gamma_{\mathbb{Z}^m,\mathfrak{s}}$, and outputs a public-secret key pair $(\mathsf{pk} \coloneqq (\mathbf{B}, \mathbf{s}^\top\mathbf{B} + \boldsymbol{\varepsilon}), \mathsf{sk} \coloneqq \mathbf{s})$.

- $\mathsf{Enc}(\mathsf{pk}, b) \to \mathbf{x}$. The encryption algorithm takes as input a public key pk, interpreted as a matrix $\mathbf{B} \in \mathbb{Z}_q^{n\times m}$ and a vector $\mathbf{c} \in \mathbb{Z}_q^m$, and a message $b \in \{0,1\}$. It samples a random vector $\mathbf{r} \leftarrow\!\!\$\ \{0,1\}^m$, and outputs a ciphertext $\mathbf{x}$ computed as

$$\mathbf{x} \coloneqq \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{bmatrix} \cdot \mathbf{r} + b \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix}.$$

- $\mathsf{Dec}(\mathsf{sk}, \mathbf{x}) \to b$. The decryption algorithm takes as input a secret key sk interpreted as a vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a ciphertext $\mathbf{x} \in \mathbb{Z}_q^{n+1}$. It outputs 0 if $\left|\begin{bmatrix} -\mathbf{s}^\top & 1 \end{bmatrix} \cdot \mathbf{x}\right| \leqslant q/4$, and 1 otherwise.

If $m > n\log(q)$ and $m\mathfrak{B} \ll q/4$ for $\mathfrak{B} = \mathfrak{s}\sqrt{m}$, then the scheme is secure under the $\mathsf{LWE}_{n,m,q,\mathfrak{s}}$ assumption. More importantly, for our purposes, notice that if $m\mathfrak{B} \ll q/8$ then $\mathrm{LinHE}.\mathsf{Enc}(\mathsf{pk}, b_0) + \mathrm{LinHE}.\mathsf{Enc}(\mathsf{pk}, b_1)$ is equal to

$$\begin{bmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{bmatrix} \cdot \mathbf{r}_0 + b_0 \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{bmatrix} \cdot \mathbf{r}_1 + b_1 \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^\top \end{bmatrix} \cdot (\mathbf{r}_0 + \mathbf{r}_1) + (b_0 + b_1) \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix}$$

which is $x \coloneqq \mathrm{LinHE.Enc}(\mathsf{pk}, b_0 \oplus b_1)$, when $m\mathfrak{B} \ll q/8$ for suitable choice of q, so that

$$\left|\begin{bmatrix} -\mathbf{s}^\top & 1 \end{bmatrix} \cdot \hat{\mathbf{x}}\right| = \left|\boldsymbol{\varepsilon}^\top \cdot (\mathbf{r}_0 + \mathbf{r}_1) + (b_0 \oplus b_1) \cdot \lfloor q/2 \rfloor\right| \leqslant |2m\mathfrak{B} + (b_0 \oplus b_1) \cdot \lfloor q/2 \rfloor|$$

is at most $q/4$ if and only if $b_0 = b_1$.

7.2.3 Cryptographic hardness assumptions

Definition 7.4 (OM-ISIS [6])**.** Let $q, n, m, \mathfrak{s}, \mathfrak{B}$ be functions of security parameter λ. Consider the following experiment:

(i) The challenger uniformly samples a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and sends $\mathbf{A}$ to adversary $\mathcal{A}$.

(ii) $\mathcal{A}$ adaptively makes queries of the following types to the challenger, in any order.

- **Syndrome queries.** $\mathcal{A}$ requests for a challenge vector, to which the challenger replies with a uniformly sampled vector $\mathbf{t} \leftarrow\!\!\$\ \mathbb{Z}_q^n$. We denote the set of received vectors by $\mathcal{S}$.
- **Pre-image queries.** $\mathcal{A}$ queries a vector $\hat{\mathbf{t}} \in \mathbb{Z}_q^n$, to which the challenger replies with a short vector $\hat{\mathbf{z}} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \hat{\mathbf{z}} = \hat{\mathbf{t}}$ and $\|\hat{\mathbf{z}}\| \leqslant \mathfrak{s}\sqrt{m}$. Let ℓ denote the total number of preimage queries.

(iii) Finally, $\mathcal{A}$ outputs $\ell + 1$ tuples of the form $\{(\mathbf{z}_i, \mathbf{t}_i)\}_{j \in [\ell+1]}$. $\mathcal{A}$ wins if

$$\forall i \neq j \in [\ell + 1] \ : \ \mathbf{A} \cdot \mathbf{z}_i = \mathbf{t}_i, \ \|\mathbf{z}_i\| \leqslant \mathfrak{B}, \ \mathbf{t}_i \neq \mathbf{t}_j \text{ and } \mathbf{t}_i \in \mathcal{S} \ .$$

The $\mathsf{OM\text{-}ISIS}_{q,n,m,\mathfrak{s},\mathfrak{B}}$ assumption states that for every PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins is $\mathsf{negl}(\lambda)$.

7.3 ATPM from OM-ISIS

In this section, we give our lattice-based ATPM construction. The idea here is similar to the commit-sign-prove approach for lattice blind signatures [6, 41]. The non-trivial part of our construction is facilitating the extraction of the embedded metadata bit.

To motivate our construction, note that an anonymous token with public verifiability and without a private metadata bit is simply a blind signature. Our first attempt is to generically upgrade a blind signature scheme to embed a private metadata bit and then explain the difficulties.

ATPM from blind signatures. The natural approach would have the issuer encrypt the bit b into a ciphertext $\hat{\phi}$ (with a proof of correctness), allowing later decryption via a secret key only the issuer holds. However, this naive approach immediately fails: including $\hat{\phi}$ in the final signature in the clear allows the issuer to break unlinkability by recognizing the same ciphertext across token redemptions.

To preserve unlinkability, the client should homomorphically transform $\hat{\phi}$ into an unlinkable ciphertext while preserving information about b that the issuer can extract upon decryption. Our key insight is that this can be achieved using a specific primitive, which we explain next.

Our core technical contribution is observing that upgrading any round-optimal blind signature to an ATPM scheme requires an IND-CPA secure linearly homomorphic PKE scheme. Such a scheme satisfies the crucial property that for $b_0, b_1 \in \{0,1\}$,

$$\mathrm{LinHE.Enc}(\mathsf{pk}, b_0) + \mathrm{LinHE.Enc}(\mathsf{pk}, b_1) = \mathrm{LinHE.Enc}(\mathsf{pk}, b_0 \oplus b_1) \ .$$

The issuer computes $\hat{\phi} \leftarrow \mathrm{LinHE.Enc}(\mathsf{pk}, b)$ and signs it along with the blinded message, sending the issuer sends $\hat{\phi}$, the presignature $\hat{\sigma}$, and a proof $\hat{\pi}$ of ciphertext validity. After verification, the client computes two ciphertexts for each $b' \in \{0,1\}$ as

$$\phi_{b'} \leftarrow \hat{\phi} + \mathrm{LinHE.Enc}(\mathsf{pk}, b') \ ,$$

and proves correctness of each. By the linearity property, $\phi_{b'} = \mathrm{LinHE.Enc}(\mathsf{pk}, b \oplus b')$. The issuer recovers b from the final signature by checking which $b' \in \{0,1\}$ satisfies $\mathrm{LinHE.Dec}(\mathsf{sk}, \phi_{b'}) = 0$. Privacy of b is guaranteed by encryption security.

An issue with unlinkability. This approach unfortunately lacks automatic unlinkability as the client cannot trivially adapt the signature $\hat{\sigma}$ from covering the blinded message and $\hat{\phi}$ to covering the original message and $\phi_{b'}$. While equivalence class signatures might overcome this barrier, the resulting scheme would be non-generic. Instead, we observe that this difficulty vanishes entirely when starting from Fischlin's paradigm.

ATPM from Fischlin-type blind signature. Fischlin's paradigm generically combines a commitment scheme C, a signature scheme Σ, a PKE scheme, and a NIZK proof system. Given a CRS containing the NIZK reference string and PKE public key pk, a client obtains a blind signature on a message m as follows: create a commitment com to m, send it to the issuer who signs com and returns $\hat{\sigma}$. After verifying, the client computes a NIZK proof π demonstrating knowledge of $\hat{\sigma}$ and com such that $\hat{\sigma}$ validly signs the commitment of m. The proof serves as the blind signature.

This avoids the unlinkability issue since the final signature is a zero-knowledge proof, and the client simply hides the original ciphertext $\hat{\phi}$ within a fresh ciphertext. By zero-knowledge, unlinkability holds for any IND-CPA secure linearly homomorphic PKE.

We build our lattice-based ATPM on the efficient round-optimal blind signature of Agrawal et al. [6], which instantiates Fischlin's paradigm. To obtain a blind signature on message m, the client creates a commitment $\mathbf{v} = \mathbf{C} \cdot \mathbf{u} + \mathsf{H}(\mathsf{m})$ for public matrix $\mathbf{C}$ and short uniform vector $\mathbf{u}$, sending $\mathbf{v}$ to the issuer. The issuer uses its secret trapdoor to sample a short preimage $\mathbf{z}$ such that $\mathbf{A} \cdot \mathbf{z} = \mathbf{v}$ for its public key matrix $\mathbf{A}$. After verification, the client computes a straight-line extractable NIZK proof π demonstrating knowledge of short vectors $\mathbf{u}$ and $\mathbf{z}$ satisfying $\mathbf{A} \cdot \mathbf{z} = \mathbf{C} \cdot \mathbf{u} + \mathsf{H}(\mathsf{m})$. This proof becomes the blind signature. Security relies on the one-more ISIS assumption in the random oracle model.

To add the private bit, the issuer encrypts b as $\mathbf{x} \leftarrow \mathrm{LinHE.Enc}(\mathsf{pk}_{\mathrm{LinHE}}, b; r)$ and signs both $[\mathbf{v}\ \mathbf{x}]^\top$, sending $\mathbf{x}$, $\mathbf{z}$, and a proof $\hat{\pi}$ that $b \in \{0,1\}$ and $\mathbf{x}$ is a valid encryption of b. The client then computes $\mathbf{w}_{\hat{b}} = \mathbf{x} + \mathbf{c}_{\hat{b}}$ where $\mathbf{c}_{\hat{b}} \leftarrow \mathrm{LinHE.Enc}(\mathsf{pk}_{\mathrm{LinHE}}, \hat{b}; \nu_{\hat{b}})$ for each $\hat{b} \in \{0,1\}$. It sends $\mathbf{w}_0$, $\mathbf{w}_1$, and a NIZK proof π demonstrating knowledge of $\mathbf{u}, \mathbf{x}, \mathbf{z}, \nu_0, \nu_1$ such that

$$\mathbf{A} \cdot \mathbf{z} = \begin{bmatrix} \mathbf{C} \cdot \mathbf{u} + \mathsf{H}(\mathsf{m}) \\ \mathbf{x} \end{bmatrix}\ ,\ \mathbf{w}_0 = \mathbf{x} + \mathrm{LinHE.Enc}(\mathsf{pk}_{\mathrm{LinHE}}, 0; \nu_0)\ ,$$

$$\mathbf{w}_1 = \mathbf{x} + \mathrm{LinHE.Enc}(\mathsf{pk}_{\mathrm{LinHE}}, 1; \nu_1)\ ,\ \text{and } \|\mathbf{u}\|, \|\mathbf{z}\| \text{ are short .}$$

Public verification is performed by checking the NIZK proof. Bit extraction proceeds by determining which $\hat{b} \in \{0,1\}$ yields $\mathrm{LinHE.Dec}(\mathsf{sk}, \mathbf{w}_{\hat{b}}) = 0$.

We remark that while our idea of using linearly homomorphic PKE applies generally, the lattice instantiation introduces some technical issues. In particular, the issuer holding the secret decryption key $\mathsf{sk}_{\mathrm{LinHE}}$ can additionally recover the noise. This allows a malicious issuer to choose its own encryption randomness r in a way that it can skew the distribution of the recovered noise and break anonymity of the client. Interestingly, this is generally not the case for group-based linearly homomorphic PKE such as Paillier, because the randomness is (in a very loose sense) "wiped" during decryption, so the issuer does not get the additional distributional data as in the lattice case. To address this, some care is needed in how the encryption randomness ν_0 and ν_1 are chosen by the client. More concretely,

Construction 7.1. Let system parameters $n, m, m', \mathfrak{s}_{\boldsymbol{\varepsilon}}, \mathfrak{B}_{\boldsymbol{\varepsilon}} = \mathfrak{s}_{\boldsymbol{\varepsilon}}\sqrt{m}, \mathfrak{s}_{\mathbf{z}}, \mathfrak{B}_{\mathbf{z}} = \mathfrak{s}_{\mathbf{z}}\sqrt{m'}, \mathfrak{s}_{\boldsymbol{\nu}}, \mathfrak{B}_{\boldsymbol{\nu}} = \mathfrak{s}_{\boldsymbol{\nu}}\sqrt{m}$ and a prime number q be functions of the security parameter λ such that the one-more ISIS instance $\mathsf{OM\text{-}ISIS}_{q,n,m',\mathfrak{s}_{\mathbf{z}},2\mathfrak{B}_{\mathbf{z}}}$ and the LWE instance $\mathsf{LWE}_{q,\mathfrak{s}_{\boldsymbol{\varepsilon}}}$ are hard. These parameters must satisfy the following constraints:

$$n = \mathsf{poly}(\lambda),\ m, m' > n \log q + \lambda,\ \frac{\mathfrak{s}_{\mathbf{z}}}{2m'} = \Omega(1),\ m\mathfrak{B}_{\boldsymbol{\varepsilon}} \ll q/8,\ \mathfrak{s}_{\boldsymbol{\nu}} = O(2^{\lambda})\ . \tag{7.1}$$

Our construction requires a hash function $\mathsf{H} : \{0,1\}^{\lambda} \to \mathbb{Z}_q^n$, a lattice trapdoor $\mathcal{T} = (\mathsf{TrapGen}, \mathsf{SamplePre})$, a NIZK $\Pi_1 = (\Pi_1.\mathsf{Setup}, \Pi_1.\mathsf{Prove}, \Pi_1.\mathsf{Verify})$ and a NIZK argument of knowledge (NIZKAoK) with straight-line extraction $\Pi_2 = (\Pi_2.\mathsf{Setup}, \Pi_2.\mathsf{Prove}, \Pi_2.\mathsf{Verify})$ respectively for languages $\mathcal{L}_1^{\mathsf{ATPM}}$ and $\mathcal{L}_2^{\mathsf{ATPM}}$ as defined,

Language $\mathcal{L}_1^{\mathsf{ATPM}}$

Instance: Each instance $\mathbb{x}$ is interpreted as matrices $\mathbf{A}$ and $\mathbf{B}$, and vectors $\mathbf{c}$ and $\mathbf{x}$.
Witness: Witness $\mathbb{w}$ consists of vectors $\mathbf{s}$ and $\mathbf{r}$, and a bit b.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$b \in \{0,1\} \ \wedge\ \mathbf{x} = \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^{\top} \end{bmatrix} \cdot \mathbf{r} + b \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix}$$

Language $\mathcal{L}_2^{\mathsf{ATPM}}$

Instance: Each instance $\mathbb{x}$ is interpreted as matrices $\mathbf{A}$ and $\mathbf{B}$, vectors $\mathbf{c}, \mathbf{w}_0$ and $\mathbf{w}_1$, and a string t.
Witness: Witness $\mathbb{w}$ consists of vectors $\mathbf{u}, \mathbf{x}, \mathbf{z}, \boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$.
Membership: $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if

$$\mathbf{A}_{\top} \cdot \mathbf{z} - \mathbf{C} \cdot \mathbf{u} = \mathsf{H}(t) \ \wedge\ \mathbf{A}_{\perp} \cdot \mathbf{z} = \mathbf{x} \ \wedge\ \|\mathbf{u}\| \leqslant \frac{\mathfrak{B}_{\mathbf{z}}}{2m'} \ \wedge\ \|\mathbf{z}\| \leqslant \mathfrak{B}_{\mathbf{z}}$$
$$\wedge\ \mathbf{w}_0 = \mathbf{x} + \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^{\top} \end{bmatrix} \cdot \boldsymbol{\nu}_0 \ \wedge\ \mathbf{w}_1 = \mathbf{x} + \begin{bmatrix} \mathbf{B} \\ \mathbf{c}^{\top} \end{bmatrix} \cdot \boldsymbol{\nu}_1 + \begin{bmatrix} \mathbf{0}^n \\ \lfloor q/2 \rfloor \end{bmatrix} \ \wedge\ \|\boldsymbol{\nu}_0\|, \|\boldsymbol{\nu}_1\| \leqslant \mathfrak{B}_{\boldsymbol{\nu}}$$

$\mathsf{Setup}(1^\lambda)$

1 : $\mathbf{C} \leftarrow\!\$\ \mathbb{Z}_q^{n\times m'}$
2 : $\mathsf{crs}_1 \leftarrow\!\$\ \Pi_1.\mathsf{Setup}(1^\lambda)$
3 : $\mathsf{crs}_2 \leftarrow\!\$\ \Pi_2.\mathsf{Setup}(1^\lambda)$
4 : **return** $\mathsf{pp} \coloneqq (\mathbf{C}, \mathsf{crs}_1, \mathsf{crs}_2)$

$\mathsf{KeyGen}(1^\lambda)$

1 : $(\mathbf{A}, \mathbf{T_A}) \leftarrow\!\$\ \mathsf{TrapGen}(2n+1, m', q, \mathfrak{s}_\mathbf{z})$
2 : $\mathbf{B} \leftarrow\!\$\ \mathbb{Z}_q^{n\times m},\ \mathbf{s} \leftarrow\!\$\ \mathbb{Z}_q^n,\ \boldsymbol{\varepsilon} \leftarrow\!\$\ \Gamma_{\mathbb{Z}^m, \mathfrak{s}_{\boldsymbol{\varepsilon}}}$
3 : **return** $\mathsf{pk} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{s}^\top\mathbf{B} + \boldsymbol{\varepsilon}),\ \mathsf{sk} \coloneqq (\mathbf{T_A}, \mathbf{s})$

$\mathsf{Verify}(\mathsf{pk} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{c}), t, \sigma \coloneqq (\mathbf{w}_0, \mathbf{w}_1, \pi_2))$

1 : $\mathbb{x} \coloneqq (\mathbf{A}, \mathbf{B}, \mathbf{c}, \mathbf{w}_0, \mathbf{w}_1, t)$
2 : **return** $\Pi_2.\mathsf{Verify}(\mathsf{crs}_2, \mathbb{x}, \pi_2)$

$\mathsf{ReadBit}(\mathsf{sk} \coloneqq (\mathbf{T_A}, \mathbf{s}), t, \sigma \coloneqq (\mathbf{w}_0, \mathbf{w}_1, \pi_2))$

1 : **if** $\mathsf{Verify}(\mathsf{pk}, t, \sigma) \neq 1$ **then**
2 : **return** $\perp$
3 : **foreach** $\hat{b} \in \{0,1\}$ **do**
4 : **if** $\left|\begin{bmatrix}-\mathbf{s}^\top & 1\end{bmatrix} \cdot \mathbf{w}_{\hat{b}}\right| \leqslant q/4$ **then**
5 : **return** $\hat{b}$
6 : **return** $\perp$

Figure 7.4: The system setup, key generation, verification and bit extraction algorithms for ATPM from OM-ISIS.

System setup. Given the system parameters $\lambda, n, m, m', q, \mathfrak{s}_\mathbf{z}, \mathfrak{s}_{\boldsymbol{\varepsilon}}$ as well as the hash function H modeled as a random oracle, the setup algorithm samples a random matrix $\mathbf{C} \in \mathbb{Z}_q^{n\times m'}$. It also runs the NIZK setup algorithms for the relations $\mathcal{L}_1^{\mathsf{ATPM}}$ and $\mathcal{L}_2^{\mathsf{ATPM}}$ to generate the corresponding crs_1 and crs_2.

Key generation. The issuer's key generation algorithm uses trapdoor sampling to create a random matrix and its trapdoor $\mathbf{A}, \mathbf{T_A}$ as the public and secret keys respectively. It also samples a uniform public matrix $\mathbf{B}$ and additionally outputs $\mathbf{B}$ and $\mathbf{s}^\top\mathbf{B} + \boldsymbol{\varepsilon}$ as part of the public key, for uniformly sampled $\mathbf{s}$ and $\boldsymbol{\varepsilon}$ sampled from a discrete Gaussian with bound $\mathfrak{B}_{\boldsymbol{\varepsilon}}$ such that $m\mathfrak{B}_{\boldsymbol{\varepsilon}} \ll q/8$.

Presignature issuance and token generation. We show the presignature issuance and token generation protocols in detail in Figure 7.5. Given some query $\mathbf{v}$, the issuer embeds the metadata bit b inside the vector $\mathbf{x}$, which in turn can be viewed as (LWE-based) encryption of the b under the public key $(\mathbf{B}, \mathbf{c})$. For a suitable choice of the modulus q, the shown encryption scheme is linearly

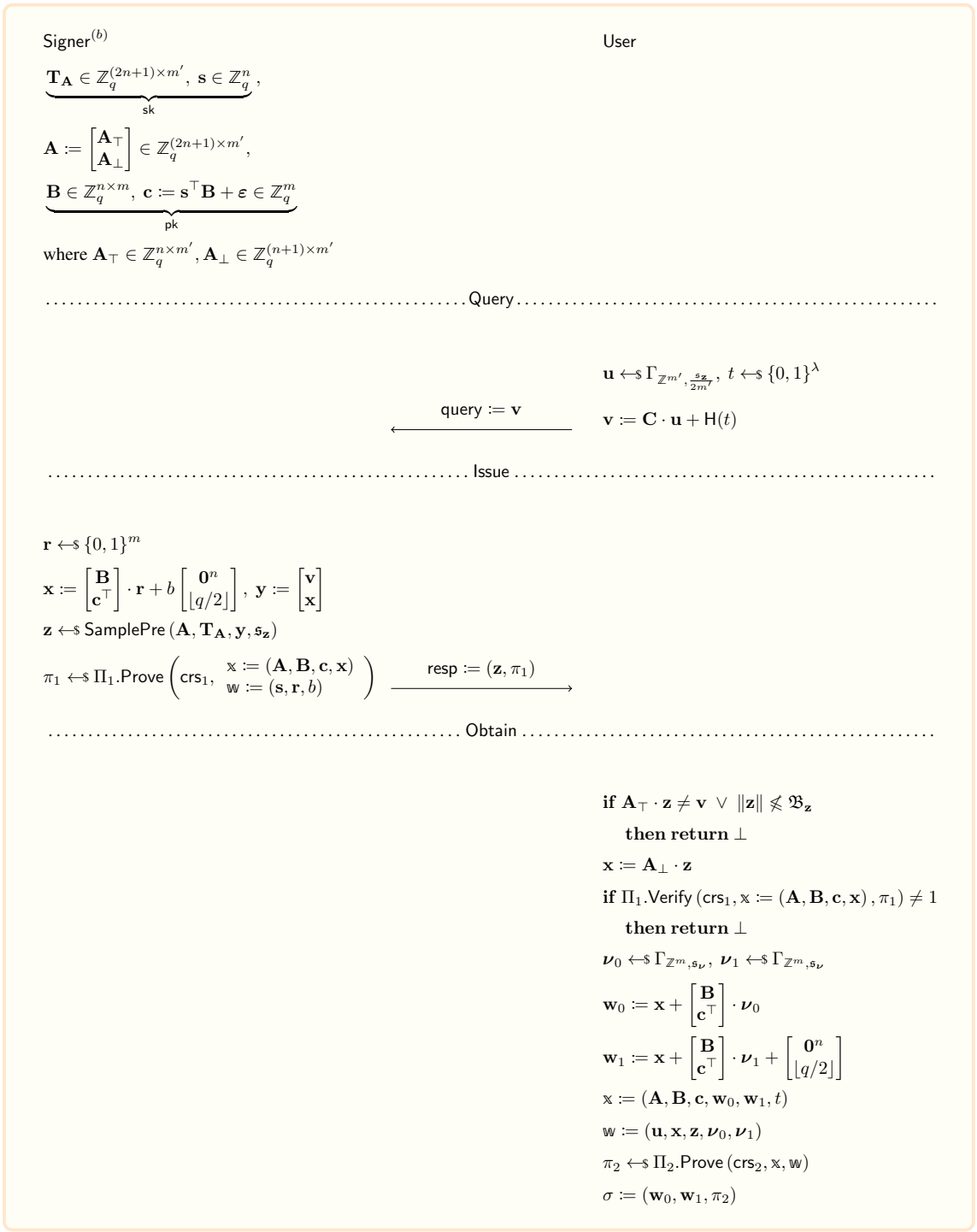

Figure 7.5: The pre-signature issuance and token obtain algorithms for ATPM from OM-ISIS.

homomorphic in the sense we explained previously. The issuer then samples a preimage $\mathbf{z}$ on input $\mathbf{y} \coloneqq \begin{bmatrix} \mathbf{v} \\ \\ \mathbf{x} \end{bmatrix}$ using the lattice trapdoor $\mathbf{T}_{\mathbf{A}}$. The full issuance includes $\mathbf{z}$ and a NIZK proof π_1 proving $b \in \{0,1\}$ and the well-formedness of the vector $\mathbf{x}$.

In order to create the token, the client first verifies signature, as well as the NIZK proof. This is then followed by the client computing $\mathbf{w}_0 \coloneqq \mathbf{x} + \overline{\mathbf{w}}_0$ (similarly $\mathbf{w}_1$), where $\overline{\mathbf{w}}_0$ (resp. $\overline{\mathbf{w}}_1$) is essentially the encryption of 0 (resp. 1) under the public key $(\mathbf{B}, \mathbf{c})$ and randomness $\boldsymbol{\nu}_0$ (resp. $\boldsymbol{\nu}_1$). Notably, instead of sampling $\boldsymbol{\nu}_{\hat{b}}$ for each $\hat{b} \in \{0,1\}$ from the same distribution as $\mathbf{r}$, the client samples them from a discrete Gaussian with variance $\mathfrak{s}_{\boldsymbol{\nu}} = O(2^{\lambda})$. Looking ahead to the security proof, this allows us to leverage the smudging lemma (Lemma 4.3) so that the distribution over $(\mathbf{r} + \boldsymbol{\nu}_{\hat{b}})$ is statistically indistinguishable from $\boldsymbol{\nu}_{\hat{b}}$, thereby preventing a malicious signer (who can decrypt $\mathbf{w}_{\hat{b}}$ to learn $\boldsymbol{\varepsilon}^{\top} \cdot (\mathbf{r} + \boldsymbol{\nu}_{\hat{b}})$) from breaking unlinkability by skewing $(\mathbf{r} + \boldsymbol{\nu}_{\hat{b}})$ with its choice of $\mathbf{r}$.

Now, due to the linear homomorphism of the encryption scheme, $\mathbf{w}_0$ (resp. $\mathbf{w}_1$) is an encryption of $b \oplus 0$ (resp. $b \oplus 1$). Finally, the client creates a NIZK proof π_2 for proving knowledge of a short $\mathbf{z}$ as a signature on $\mathbf{z}$ as well as the well-formedness of $\mathbf{w}_0, \mathbf{w}_1$. The final token is the tag t and the signature comprised of the ciphertexts $\mathbf{w}_0$ and $\mathbf{w}_1$ along with the NIZK proof π_2.

Public verification (token redemption). The public verification algorithm simply verifies the NIZK proof π_2.

Bit extraction. Extraction of the private metadata bit proceeds by first calling the verification algorithm. Then the issuer essentially decrypts both $\mathbf{w}_0$ and $\mathbf{w}_1$, and outputs the bit corresponding to whichever of the two is 0.

We now state the main security theorem for our construction. The detailed proofs are provided in the full version of [30].

Theorem 7.5. *Let system parameters* $n, m, m', \mathfrak{s}_{\boldsymbol{\varepsilon}}, \mathfrak{B}_{\boldsymbol{\varepsilon}} = \mathfrak{s}_{\boldsymbol{\varepsilon}}\sqrt{m}, \mathfrak{s}_{\mathbf{z}}, \mathfrak{B}_{\mathbf{z}} = \mathfrak{s}_{\mathbf{z}}\sqrt{m'}, \mathfrak{s}_{\boldsymbol{\nu}}, \mathfrak{B}_{\boldsymbol{\nu}} = \mathfrak{s}_{\boldsymbol{\nu}}\sqrt{m}$ *and a prime number* q *be functions of the security parameter* λ *satisfying the constraints defined in (7.1), then Construction 7.1 is:*

(i) one-more unforgeable (in the random oracle model) assuming Π_1 *satisfies zero-knowledge,* Π_2 *are straight-line extractable and the* $\mathsf{OM\text{-}ISIS}_{q,n,m',\mathfrak{s}_{\mathbf{z}},2\mathfrak{B}_{\mathbf{z}}}$ *problem is hard;*

(ii) κ*-unlinkable for* $\kappa = 2$*, assuming* Π_2 *satisfies zero-knowledge; and*

(iii) has private metadata bit assuming Π_1 *satisfies zero-knowledge,* Π_2 *is straight-line extractable, and the* $\mathsf{OM\text{-}ISIS}_{q,n,m',\mathfrak{s}_{\mathbf{z}},2\mathfrak{B}_{\mathbf{z}}}$ *and* $\mathsf{LWE}_{q,n,m,\mathfrak{s}_{\boldsymbol{\varepsilon}}}$ *problems are hard.*

Pf. sketch. The proof of one-more unforegeability of the construction essentially follows via a reduction to the OM-ISIS assumption. At a high-level, the reduction sets $\mathbf{C} \coloneqq \mathbf{A}_\top \cdot \mathbf{R}$ for $\mathbf{R}$ chosen uniformly from $\{0,1\}^{m' \times m'}$. We can use the leftover hash lemma (Corollary 4.2) to argue that this change is statistically indistinguishable to an adversary. Then, whenever the adversary makes a query for the hash of some value t, the reduction programs the random oracle such that $\mathsf{H}(t) \coloneqq \mathbf{t}$ where $\mathbf{t}$ is the response of a *syndrome query* to the OM-ISIS challenger. Then, for each Issue query $\mathbf{v}$, the reduction makes a *preimage query* on $\mathbf{v}$ to the OM-ISIS challenger and forwards the corresponding preimage, $\mathbf{z}$ along with a simulated proof π_1. Finally, when the adversary outputs its (one-more) forgeries, the reduction can extract $(\mathbf{u}_i, \mathbf{z}_i)$ from each proof $\pi_{2,i}$. The reduction then computes $\mathbf{z}'_i \coloneqq \mathbf{z}_i - \mathbf{R} \cdot \mathbf{x}_i$ and $\mathbf{t}_i \coloneqq \mathbf{A}_\top \cdot \mathbf{z}_i - \mathbf{C} \cdot \mathbf{x}_i$ and outputs $\{(\mathbf{z}'_i, \mathbf{t}_i)\}_i$ as the OM-ISIS solution. By choosing $\mathbf{u}$ from a suitable distribution, we can ensure that $\|\mathbf{z}'_i\| \leqslant 2\mathfrak{B}_{\mathbf{z}}$, as desired.

To prove κ-unlinkability of the construction, the idea is to make the final output values independent of the query as well as the responses. In order to accomplish this, we will firstly have the challenger simulate all its proofs. Following this, we can make a strictly information theoretic argument to reason that any query $\mathbf{v}$ is indistinguishable from uniformly random. Moreover, the corresponding final challenge output, $O_i \coloneqq (t_i, \mathbf{w}_{0,i}, \mathbf{w}_{1,i}, \pi_{2,i})$ is also independent of the response $(\mathbf{z}_i, \pi_{1,i})$ by additional application of the smudging lemma (Lemma 4.3). It follows that the only meaningful information the adversary can learn in this case is the private bit embedded in $\mathbf{w}_{0,i}$ and $\mathbf{w}_{1,i}$. So the

best adversarial strategy is to create some n_b responses with bit b for each bit ($n_0 + n_1 = n$), read the bit $b_{\hat{i}}$ of $O_{\hat{i}}$ and output a random value i^* from the set of all $n_{b_{\hat{i}}}$ indices where the embedded bit was equal to $b_{\hat{i}}$. The probability that the adversary wins is

$$\sum_{b\in\{0,1\}} \Pr\left[b_{i^*} = b_{\hat{i}}\right] \cdot \Pr\left[i^* = \hat{i}\right] = \sum_{b\in\{0,1\}} \frac{n_b}{n} \cdot \frac{1}{n_b} = \frac{2}{n} \ .$$

Lastly, we can leverage the IND-CPA security (from LWE) of our linearly homomorphic encryption scheme for privacy of the metadata bit of our construction. We also need to be able to carefully answer any Read or Valid queries, which can be done by extracting $\mathbf{x}$ from any π_2 and looking up the corresponding bit $\hat{b}$ stored during issuance. Importantly, if no such entry is found, we can once again reduce the security to OM-ISIS (as in one-more unforgeability) since the adversary must have created a valid token without the issuer. □

7.3.1 Concrete efficiency

In this section, we explain our instantiation of our ATPM construction and provide concrete estimates under suitable parameter choices.

Following the efficient blind signature scheme due to Agrawal, et al. [6], our scheme is instantiated with SHA-3-256 as the hash function, Falcon-512 [152] for pre-image sampling and the efficient NIZK for general linear relations (and norm relations) due to [132]. The LNP22 proof system, however, does not provide a straight-line extraction which is crucial for our purpose. As we previously remarked, straight-line witness extraction from a NIZK can be generically achieved using an IND-CPA secure PKE by encrypting the witness, and then additionally proving correct computation of the ciphertext inside the NIZK. This construction is secure if the secret key is discarded after setup, and the public key is given as part of the public parameters. To extract a witness, a challenger runs the setup algorithm and withholds the secret key. Zero-knowledge follows from that of the NIZK and IND-CPA security of the PKE. Concretely, we use the PKE scheme in [134] (similar to CRYSTALS-Kyber).

Now, in order to evaluate the concrete costs of our (expanded) construction, we instantiate it in a manner identical to [6]. Recall that the modulus for the zero-knowledge proof should factor into primes congruent to $5 \mod 8$ so that the polynomial $x^{128}+1$ generating the quotient for the ring used in the zero-knowledge proof splits exactly into two factors (cf. [132, Lemma 2.6]). Further, by choosing the moduli for the signature scheme and PKE as multiples of the zero-knowledge modulus we can simplify the relation.

Note that the base modulus q must be chosen to be sufficiently large so as to account for the smudging lemma. In particular, for around 80-bits of hardness (i.e., roughly 2^{-40} statistical distance between the "good" and "bad" distributions), we set q to be the smallest prime greater than 2^{50} that is $5 \mod 8$. For brevity, we will summarize the salient aspects of the instantiation below.

- Query :

  Client's commitment. The client samples $t \in \{0,1\}^{256}$ and $u_1, u_2 \in \mathcal{R}_{q,256}$ such that $\|(u_1,\, u_2)\|_\infty = 2^{22}$, and then for public polynomial $c \in \mathcal{R}_{q,256}$ computes the commitment $v \coloneqq c \cdot u_1 + u_2 + \mathsf{H}(t)$.

  Consequently, the size of the client's query is 1.5 KB.

- Issue :

  Trapdoor generation and preimage sampling. We instantiate Falcon-512 [152] over the ring $\mathcal{R}_{q,512}$,

$$\mathfrak{s}_z = \frac{1.17}{\pi}\sqrt{q \cdot \log\left(4 \cdot 512 \cdot \left(1+\sqrt{\lambda \cdot 2^{64}}\right)/2\right)} \text{ and } \mathfrak{B}_z = 1.1 \cdot \mathfrak{s}_{\mathbf{z}}\sqrt{2 \cdot 512}\,.$$

  Let $x \in \mathcal{R}_q^{256}$ be the encryption of the private metadata bit b. The signer runs the preimage sampling algorithm over $v\|x \in \mathcal{R}_{q,512}$ with respect to its public verification key polynomial $a_1 \in \mathcal{R}_{q,512}$, to obtain $z_1, z_2 \in \mathcal{R}_{q,512}$ such that $a_1 \cdot z_1 + z_2 = v\|x$.

Issuer's ZK proof. The client must prove the relation for $\mathcal{L}_1$. We set the modulus $q_{\mathsf{ZK}}^{(1)} = q \cdot q_1^{(1)} \approx 2^{118}$, and choose the Module-LWE parameter $\hat{m}_2 = 58$ and Module-SIS parameter $\hat{n} = 14$ for soundness of the zero-knowledge proof, and carry over the other parameters from [132]. The length of the committed vector over $\mathcal{R}_{q_{\mathsf{ZK}}^{(1)},128}$ is $\hat{m}_1 = 4$. The final proof size after compression is 64.5 KB.

The issuer sends z_1, x (the client can compute z_2) as well as the proof. Given this, the estimated size of the issuer's response is around $4.6 + 64.5 = 69.1$ KB.

- Obtain :

Client's ciphertext. The client computes w_0, w_1 using randomness ν_0 and ν_1 sampled according to $\Gamma_{\mathcal{R}_q,\mathfrak{s}_\nu}$, where $\mathfrak{s}_\nu \approx 2^{40}$ for roughly 2^{-40} statistical distance. For straight-line extraction, the client must encrypt the 6 witness polynomials $u_1, u_2, x, z_1, \nu_0, \nu_1 \in \mathcal{R}_{q,512}$ (z_2 can be computed using the instance and the extracted values). The PKE scheme is instantiated over $\mathcal{R}_{q_{\mathrm{E}}^{(2)},128}$ for $q_{\mathrm{E}}^{(2)} = q \cdot q_1^{(2)} \approx 2^{62}$. The reduced degree polynomial-ring allows the protocol to remain compatible with the degree of the polynomial-ring in the zero-knowledge proof system of [132]. Since $\mathcal{R}_{q,512} \subset \mathcal{R}_{q_{\mathrm{E}}^{(0)},512} \cong \mathcal{R}^4_{q_{\mathrm{E}}^{(0)},128}$, the rank of the plaintext space is $6 \cdot 4 = 24$. Choosing the infinity norm bound $\tau = 3$ on the encryption randomness $(s, e_1, \mathbf{e}_2)$, and the Module-LWE rank to be 19 for sufficient hardness results in a final ciphertext size of approximately 42.2 KB.

Client's ZK proof. The client must prove the relation for $\mathcal{L}_2$. We set the modulus $q_{\mathsf{ZK}}^{(2)} = q \cdot q_1^{(2)} \cdot q_2^{(2)} \approx 2^{79}$, and choose the Module-LWE parameter $\hat{m}_2 = 46$ and Module-SIS parameter $\hat{n} = 12$ for soundness of the zero-knowledge proof, and carry over the other parameters from [132]. The length of the committed vector over $\mathcal{R}_{q_{\mathsf{ZK}}^{(2)},128}$ is 69. The resulting proof can then be further compressed using [78], essentially by cutting low-order bits of the commitment. The final proof size after compression is 83.9 KB.

Thus, the transcript size is 70.6 KB, and the final token size is 129.1 KB, including 32 bytes for the message and 3 KB for w_0 and w_1.

Concrete security. In Table 7.1, we summarize the concrete security of the given instantiation estimated using the Python script provided by [6][1].

Security Property	**Assumption**	**Bit security (Core-SVP Hardness)**
Hiding for commitment	MLWE	106 bits
Unforgeability (Falcon)	MSIS	121 bits
Ciphertext security (Obtain)	MLWE	105 bits
NIZK proof (Issue)	MSIS and MLWE	101 and 102 bits (resp.)
NIZK proof (Obtain)	MSIS and MLWE	100 and 104 bits (resp.)
One-more Unforgeability	rOM-ISIS	140 bits

Table 7.1: Concrete security of token generation.

[1] Our estimation code can be found at https://github.com/aayux/atlantis-estimates. In it, we also provide parameters for 128-bit hardness of smudging and find that the resulting transcript size is 102.7 KB and the final token size is around 176.3 KB.

Chapter 8: Conclusion and Future Work

The transition from interactive to non-interactive protocols represents a fundamental shift in how we can deploy tokens at scale. By recognizing that blindness is achievable without interaction when the message is sampled by the user rather than the signer, we enable a new class of practical applications. In this dissertation we demonstrated that non-interactivity greatly simplifies protocol design, reducing latency, and facilitating asynchronous pre-computation in systems that serve billions of requests daily.

The contributions contained within this work establish a rigorous formal foundation for non-interactive blind signatures and extend the general set of design techniques to address challenges of modern systems. The security framework developed in Chapter 3 places NIBS on a theoretically sound footing, moving beyond the insufficient guarantees of prior work and providing the confidence necessary for real-world deployment. Further, the generic constructions from circuit-private leveled homomorphic encryption and general-purpose NIZKs offer practitioners a way forward towards building concrete protocols.

In Chapter 4 we propose non-interactive batched blind signatures to address two pressing challenges: resistance to quantum adversaries and achieving efficient communication for batch issuance of signatures. By leveraging recent advancements in lattice-based cryptography, we show that practical post-quantum solutions are within reach.

The introduction of threshold issuance in Chapter 5 further extends the applicability of non-interactive protocols to low-trust environments by distributing the secret signing key. The construction based on the Pointcheval-Sanders signature scheme also advances the state of the art for NIBS, achieving lower communication overhead and faster issuance time than prior works.

Our work on non-interactive anonymous tokens demonstrates how our formalizations and techniques from NIBS propagate to solve real-world privacy problems. The NIAT scheme presented in Chapter 6 combines non-interactivity with private metadata bits, enabling signers to embed hidden

trust signals in tokens while maintaining user anonymity. This secret flag allows signers to tag suspicious users without alerting them to their detection. In addition, we propose a critical extension for identifying double-spenders. This unique capability of NIAT is made possible by the presence of user public keys in our non-interactive setting. With rigorous software implementation and benchmarking, we also demonstrate practical feasibility of our scheme.

Finally, in Chapter 7 we present Atlantis, a lattice-based scheme that extends anonymous tokens with private metadata to the post-quantum setting, addressing the pressing need for quantum-safe deployments.

Looking ahead, several natural research directions emerge. The constructions presented in this work are either proven secure in non-standard models, or rely on structured hardness assumptions. Therefore, an obvious open problem here is to build protocols under weaker or more standard assumptions. In the same vein, it is also of practical interest to design protocols that issue signatures under existing user public keys (which are typically ECDSA or EdDSA keys) rather than requiring users to generate and store NIBS/NIAT specific keys. Even more generally, efficient protocols under pairing-free groups would be of interest. In the post-quantum setting, giving more efficient protocols for these primitives remains an exciting challenge. Here, a possible avenue is to consider other families of post-quantum secure hardness assumptions such as those based on isogenies of supersingular elliptic curves. Another more concrete direction in this area is to replace the PKE used in the lattice-based design with the Katsumata transform, which can also be used to facilitate straight-line extraction without significantly impacting the signature size.

The blind signature protocols proposed in this work are descendants of Chaum's foundational insight and the cypherpunk conviction that cryptography is a tool for human liberation. The techniques in this work allow signatures to be distributed at scale to billions of Internet users without requiring them to surrender their privacy. It is my hope that in doing so, we have strengthened the technical foundations upon which privacy-preserving systems are built.

Intentionally left blank.

Afterword

Chaum's blind signatures have surpassed their original purpose, evolving into a general cryptographic tool in their own right, with rich applications to and beyond the Web. The focus of my work has been on a narrow specialization of this tool, namely non-interactive blind signatures to which I have offered a strictly technical treatment in this dissertation. However, I feel it necessary to acknowledge that this work, like all cryptographic work, carries its own internal politics [154]; and the struggle between its (just-)use and abuse of technology mirrors my own internal struggle, a vastly complex beast that deserves deeper investigation that I intend to undertake as I stake out my own research path.

The cypherpunk movement presciently predicted many of the encroachments of large organizations into private lives that are commonplace today. Their concrete contributions are undeniable. David Chaum pioneered anonymous communication networks, blind signatures, and the first cryptographic electronic cash. Whitfield Diffie and Martin Hellman invented public-key cryptography, eliminating the need for physical key exchange. Through *Bernstein v. United States*, Daniel Bernstein and the Electronic Frontier Foundation secured the legal right to publish cryptographic source code, a foundation upon which much of modern digital privacy systems rely [165].

At the same time the assumption that cryptography and decentralization alone can solve these problems overlooks how institutional power shapes technological use. The past decade has provided stark reminders. Edward Snowden exposed the NSA's global mass-surveillance infrastructure [97, 98]. Cambridge Analytica's manipulation of electoral data revealed how personal information could be weaponized at scale [52]. Various state-sponsored interference campaigns exploit social media vulnerabilities, systematically distorting public discourse [107]. These incidents have have cast a pall on the techno-optimism of the 1990s.

The erosion of these technological ideals is further evidenced by the renewed attacks on individual privacy with 'Chat Control' regulations [83], age-verification laws [66], and VPN restrictions [10], all framed as necessary to protect children. Yet this framing obscures a dangerous substitution; trading fundamental privacy rights for performative security that protects neither children nor adults [18].

The narrative of privacy and accountability as irreconcilable binary forces is not new. During the so called 'Crypto Wars', the Clinton Administration endorsed the development and standardization of the infamous Clipper Chip—which would have allowed law-enforcement to decrypt private communication using escrowed keys—using the same facile 'cops and robbers' argument [73, 127]. In many ways, we have come full circle.

How, then, should we think about cryptography's role today? Perhaps it is worth looking outside the field. I have come to believe that a research program built around communities with the greatest need for privacy-enhancing technologies offers a different way forward. This could mean human-centered work that emphasizes understanding threat models through the lived experiences of communities facing these threats. Some such groups include domestic violence survivors, political asylees, journalists under surveillance, and protesters facing state opression. But even before designing better frameworks and tools, the work begins with teaching people how to use existing ones.

Of course, this kind of research program is difficult to sustain. The incentive structures we navigate within the academy pull us away from such work, even when we recognize its importance. These are hard problems to which I don't have the answers, but I think it is important to acknowledge this tension without giving in to cynicism. Indeed, the work of re-framing cryptography as a community focused activity was already begun by Seny Kamara in his talk titled 'Crypto for the People' [114],[1] and was recently made concrete by Leah Rosenbloom [155] alongside a novel concept of 'Abolition Cryptography' [156]. These works suggest to me that it is possible to hold this tension and still do meaningful cryptographic work.

[1] Incidentally, the talk was what inspired me to consider cryptography as an area of research when applying to doctoral programs. It seems that I too have come a full circle.

One of the things I love about working in the field of cryptography is that our threat models are constantly evolving as new social and technological paradigms emerge. This means that there will always be new challenges to address and more work to be done.

Part III

Appendix

Appendix A: Further Remarks on Context as Input or Output

As previously discussed, the signing context c can either be given as input to or returned as the output of the Issue algorithm. Both approaches make inherent assumptions about the nature of the system. The former approach (Hanzlik's [101]) considers the signing context to be a random nonce supplied through some external source of randomness, ensuring reusability of the protocol by default. The latter approach (BCGY's [24]) allows one to define the reusability of a NIBS scheme as a formal correctness property. Both approaches have their benefits and drawbacks.

Hanzlik's approach makes sense for systems where the signing context is often shared across signers, such as in the case of non-interactive threshold blind signatures. However, without a meaningful notion of reusability, this interpretation allows one to define a vacuous scheme where the signer ignores the c during issuance and consequently a critical robustness property is left as a design choice.

BCGY's approach circumvents this issue, since with the context being the output of the signer's Issue algorithm, one can formulate a suitable model for reusability for NIBS. Conversely, because the signing context is now determined by the signers, it becomes challenging to co-ordinate signing contexts for multiple signers within the same context.

Interestingly, we can show that the two models are actually equivalent. For the forward direction, let $\mathsf{NIBS}^{(\mathsf{B})} = (\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{B})}, \mathsf{KeyGen}_{\mathsf{R}}^{(\mathsf{B})}, \mathsf{Issue}^{(\mathsf{B})}, \mathsf{Obtain}^{(\mathsf{B})}, \mathsf{Verify}^{(\mathsf{B})})$ be a NIBS scheme according to the BCGY convention. Then, given a pseudorandom function F, we build a NIBS scheme $\mathsf{NIBS}^{(\mathsf{H})} = (\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{H})}, \mathsf{KeyGen}_{\mathsf{R}}^{(\mathsf{H})}, \mathsf{Issue}^{(\mathsf{H})}, \mathsf{Obtain}^{(\mathsf{H})}, \mathsf{Verify}^{(\mathsf{H})})$ according to Hanzlik's convention.

- $\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{H})}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$. The signer's key generation algorithm samples the key-pair $(\mathsf{sk}^{(\mathsf{B})}, \mathsf{pk}^{(\mathsf{B})}) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{B})}(1^\lambda)$ and also samples a PRF key $K \leftarrow\!\!\$\ \{0,1\}^\lambda$. It sets $\mathsf{sk} \coloneqq (\mathsf{sk}^{(\mathsf{B})}, K)$ and $\mathsf{pk} \coloneqq \mathsf{pk}^{(\mathsf{B})}$.

- $\mathsf{KeyGen}_{\mathsf{R}}^{(\mathsf{H})}(1^\lambda) \to (\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{R}})$. The receiver's key generation algorithm samples the key-pair $(\mathsf{sk}_{\mathsf{R}}^{(\mathsf{B})}, \mathsf{pk}_{\mathsf{R}}^{(\mathsf{B})}) \leftarrow \mathsf{KeyGen}_{\mathsf{R}}^{(\mathsf{B})}(1^\lambda)$ and sets $\mathsf{sk}_{\mathsf{R}} \coloneqq \mathsf{sk}_{\mathsf{R}}^{(\mathsf{B})}$ and $\mathsf{pk}_{\mathsf{R}} \coloneqq \mathsf{pk}_{\mathsf{R}}^{(\mathsf{B})}$.

- $\mathsf{Issue}^{(\mathsf{H})}(\mathsf{sk}, \mathsf{pk}_{\mathsf{R}}, \mathsf{c}) \to \bar{\sigma}$. The issue algorithm parses sk as $(\mathsf{sk}^{(\mathsf{B})}, K)$. It then computes $\rho \leftarrow \mathrm{F}_K(\mathsf{c}, \mathsf{pk}_{\mathsf{R}})$ and then generates $\left(\bar{\sigma}^{(\mathsf{B})}, \mathsf{c}^{(\mathsf{B})}\right) \leftarrow \mathsf{Issue}^{(\mathsf{B})}(\mathsf{sk}^{(\mathsf{B})}, \mathsf{pk}_{\mathsf{R}}; \rho)$ and outputs $\bar{\sigma} \coloneqq (\bar{\sigma}^{(\mathsf{B})}, \mathsf{c}^{(\mathsf{B})})$.

- $\mathsf{Obtain}^{(\mathsf{H})}(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}) \to (\mathsf{m}, \sigma)$. The obtain algorithm parses $\bar{\sigma}$ as $(\bar{\sigma}^{(\mathsf{B})}, \mathsf{c}^{(\mathsf{B})})$. It then generates $(\mathsf{m}, \sigma) \leftarrow\!\!\$\ \mathsf{Obtain}^{(\mathsf{B})}(\mathsf{sk}_{\mathsf{R}}, \mathsf{pk}, \bar{\sigma}^{(\mathsf{B})}, \mathsf{c}^{(\mathsf{B})})$ and outputs it.

- $\mathsf{Verify}^{(\mathsf{H})}(\mathsf{pk}, \mathsf{m}, \sigma) \to 1/0$. The verification algorithm runs $\mathsf{Verify}^{(\mathsf{B})}(\mathsf{pk}, \mathsf{m}, \sigma)$ and returns its output.

Both user and context blindness follow from that of the underlying NIBS scheme. We now sketch out the proof for one-more unforgeability. More precisely, given an attacker against the one-more unforgeability of $\mathsf{NIBS}^{(\mathsf{H})}$ we must build a reduction that breaks the one-more unforgeability of $\mathsf{NIBS}^{(\mathsf{B})}$. The essential aspect of the reduction is that it must simulate the oracle $O_{\mathsf{Issue}}^{(\mathsf{H})}$ to the attacker, who can choose both the pk_{U} and c parts of the input. In order to do so, the reduction must internally maintain a state of all queries $(\mathsf{pk}_{\mathsf{U}_i}, \mathsf{c}_i)$ made by the attacker along with the response $\bar{\sigma}_i$. Then, on each oracle query it first checks if it was previously queried and if so it returns the corresponding $\bar{\sigma}_i$. Otherwise, it in turn queries the BCGY one-more unforgeability challenger for $\mathsf{pk}_{\mathsf{U}_i}$ and returns the response $(\bar{\sigma}_i^{(\mathsf{B})}, \mathsf{c}_i^{(\mathsf{B})})$ as the $\bar{\sigma}_i$ to the attacker (also storing it with the query). Since for every unique query $(\mathsf{pk}_{\mathsf{U}_i}, \mathsf{c}_i)$, the output of $\mathrm{F}_K(\mathsf{c}_i, \mathsf{pk}_{\mathsf{U}_i})$ is indistinguishable from uniform, the reduction is able to succesfully simulate $O_{\mathsf{Issue}}^{(\mathsf{H})}$ to the adversary. Finally, the reduction simply outputs the attacker's forgery as its own.

Now, in the other direction, let $\mathsf{NIBS}^{(\mathsf{H})} = (\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{H})}, \mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{H})}, \mathsf{Issue}^{(\mathsf{H})}, \mathsf{Obtain}^{(\mathsf{H})}, \mathsf{Verify}^{(\mathsf{H})})$ be a NIBS scheme according to Hanzlik's convention. We show how to build a NIBS scheme

$\mathsf{NIBS}^{(\mathsf{B})} = (\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{B})}, \mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{B})}, \mathsf{Issue}^{(\mathsf{B})}, \mathsf{Obtain}^{(\mathsf{B})}, \mathsf{Verify}^{(\mathsf{B})})$ according to the BCGY convention.

- $\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{B})}(1^{\lambda}) \rightarrow (\mathsf{sk}, \mathsf{pk})$. The signer's key generation algorithm samples the key-pair $(\mathsf{sk}^{(\mathsf{H})}, \mathsf{pk}^{(\mathsf{H})}) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{H})}(1^{\lambda})$ and sets $\mathsf{sk} \coloneqq \mathsf{sk}^{(\mathsf{H})}$ and $\mathsf{pk} \coloneqq \mathsf{pk}^{(\mathsf{H})}$.

- $\mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{B})}(1^{\lambda}) \rightarrow (\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}})$. The user's key generation algorithm samples the key-pair $(\mathsf{sk}_{\mathsf{U}}^{(\mathsf{H})}, \mathsf{pk}_{\mathsf{U}}^{(\mathsf{H})}) \leftarrow \mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{H})}(1^{\lambda})$ and sets $\mathsf{sk}_{\mathsf{U}} \coloneqq \mathsf{sk}_{\mathsf{U}}^{(\mathsf{H})}$ and $\mathsf{pk}_{\mathsf{U}} \coloneqq \mathsf{pk}_{\mathsf{U}}^{(\mathsf{H})}$.

- $\mathsf{Issue}^{(\mathsf{B})}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}) \rightarrow (\bar{\sigma}, \mathsf{c})$. The issue algorithm samples c uniformly from the appropriate domain and then generates $\bar{\sigma}^{(\mathsf{H})} \leftarrow\!\!\$\ \mathsf{Issue}^{(\mathsf{H})}(\mathsf{sk}, \mathsf{pk}_{\mathsf{U}}, \mathsf{c})$. It then sets $\bar{\sigma} \coloneqq \bar{\sigma}^{(\mathsf{H})}$ and $\mathsf{c} \coloneqq \mathsf{c}^{(\mathsf{H})}$, and outputs them.

- $\mathsf{Obtain}^{(\mathsf{B})}(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, (\bar{\sigma}, \mathsf{c})) \rightarrow (\mathsf{m}, \sigma)$. The obtain algorithm generates $(\mathsf{m}, \sigma) \leftarrow\!\!\$\ \mathsf{Obtain}^{(\mathsf{H})}($ $\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}, (\bar{\sigma}, \mathsf{c})$ and outputs it.

- $\mathsf{Verify}^{(\mathsf{B})}(\mathsf{pk}, \mathsf{m}, \sigma) \rightarrow 1/0$. The verification algorithm runs $\mathsf{Verify}^{(\mathsf{H})}(\mathsf{pk}, \mathsf{m}, \sigma)$ and returns its output.

Given an attacker against the one-more unforgeability of $\mathsf{NIBS}^{(\mathsf{B})}$ we sketch out a reduction that breaks the one-more unforgeability of $\mathsf{NIBS}^{(\mathsf{H})}$. This turns out to be even more straightforward. In order to simulate the oracle $\mathcal{O}_{\mathsf{Issue}}^{(\mathsf{B})}$ to the attacker, who chooses the input $\bar{\sigma}_i$, the reduction simply samples a context c_i uniformly, forwards the query $(\mathsf{pk}_{\mathsf{U}_i}, \mathsf{c}_i)$ to Hanzlik's one-more unforgeability challenger. It returns the challenger's oracle response $\bar{\sigma}_i^{(\mathsf{H})}$ as the corresponding $\bar{\sigma}_i$ along with c_i to the attacker. At the end the reduction outputs the attacker's forgery as its own.

Appendix B: Honest-key Model

In the honest-key model for NIBS, we require the attacker to specify the corresponding secret keys along with the respective public keys. That is, the attacker supplies the user's secret key in unforgeability experiment and the signer's signing key in blindness experiments, respectively.

Definition B.1 (User blindness in the honest-key model)**.** Recall the user blindness security experiment from Def. 3.3. Consider another security experiment where $\mathcal{A}$ additionally provides a secret key $\mathsf{sk}^{(i)}$ along with its corresponding $\mathsf{pk}^{(i)}$ at the time of each oracle query. Moreover, $\mathcal{A}$ provides a secret key sk along with pk when declaring the challenge pre-signature and context pairs. We say $\mathcal{A}$ is admissible if $\mathsf{sk}^{(i)}$ (sk) is a valid secret key for $\mathsf{pk}^{(i)}$ (pk, respectively). Note that admissibility can be checked efficiently as the secret key $\mathsf{sk}^{(i)}$ can be regarded as the random coins of the setup algorithm.

A NIBS scheme NIBS satisfies user blindness *in the honest-key model* if no admissible PPT adversary $\mathcal{A}$ wins in the above adapted security experiment with non-negligible probability.

Definition B.2 (Context blindness in the honest-key model)**.** A NIBS scheme NIBS satisfies context blindness *in the honest-key model* if no admissible PPT adversary $\mathcal{A}$ wins in the adapted security experiment similar to the original context blindness experiment (Def. 3.3) with non-negligible probability. Briefly, the adaptation is that the adversary provides a valid secret key associated with each verification key it outputs.

Definition B.3 (One-more unforgeability in the honest-key model)**.** A NIBS scheme NIBS satisfies one-more unforgeability *in the honest-key model* if no admissible PPT adversary $\mathcal{A}$ wins in the adapted security experiment similar to the original one-more unforgeability experiment (Def. 3.2) with non-negligible probability. Briefly, the adaptation is that the adversary provides a valid secret key associated with each user key it outputs.

We claim that any NIBS scheme that is secure in the honest-key model can be generically upgraded to a fully secure scheme, i.e., one *without* honest-keys using a NIZKAoK. We show this formally

by building a secure NIBS scheme (without honest-keys) given a NIBS scheme that is secure in the honest-key model. Formally,

Construction B.1. Our construction requires an honest-key secure NIBS scheme given by algorithms $\mathsf{NIBS_h} = (\mathsf{NIBS_h.Setup}, \mathsf{NIBS_h.KeyGen_S}, \mathsf{NIBS_h.KeyGen_U}, \mathsf{NIBS_h.Issue}, \mathsf{NIBS_h.Obtain}, \mathsf{NIBS_h.Verify})$ and a NIZKAoK for the language $\mathcal{L}_{\mathsf{h}}$ as described.

Language $\mathcal{L}_{\mathsf{h}}$

Instance: Each instance $\mathbb{x}$ is interpreted as a public key $\mathsf{pk}^{(\mathsf{h})}$ and a bit $b \in \{0,1\}$.

Witness: Witness $\mathbb{w}$ consists of a secret key $\mathsf{sk}^{(\mathsf{h})}$, and randomness $\rho \in \{0,1\}^\lambda$.

Membership: Let $\mathsf{C}_0, \mathsf{C}_1$ be two circuits encoding $\mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{h})}(\mathsf{pp}^{(\mathsf{h})}, 1^\lambda; \cdot)$ and $\mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{h})}(\mathsf{pp}^{(\mathsf{h})}, 1^\lambda; \cdot)$ respectively. Then $\mathbb{w}$ is a valid witness for $\mathbb{x}$ if:

$$(\mathsf{sk}^{(\mathsf{h})}, \mathsf{pk}^{(\mathsf{h})}) = \mathsf{C}_b(\rho)$$

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. The setup algorithm generates $\mathsf{pp}^{(\mathsf{h})} \leftarrow\!\!\$\ \mathsf{Setup}^{(\mathsf{h})}(1^\lambda)$, and then runs the setup algorithms for Π (for language $\mathcal{L}^{(\mathsf{h})}$) to generate $\mathsf{crs} \leftarrow \Pi.\mathsf{Setup}(\mathsf{pp}^{(\mathsf{h})}, 1^\lambda)$ and outputs it as the public parameter pp of the protocol.

- $\mathsf{KeyGen_S}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$. The signer's setup algorithm samples a random value $r_{\mathsf{S}} \leftarrow\!\!\$\ \{0,1\}^\lambda$ and runs $(\mathsf{sk}^{(\mathsf{h})}, \mathsf{pk}^{(\mathsf{h})}) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}^{(\mathsf{h})}(\mathsf{pp}^{(\mathsf{h})}, 1^\lambda; r_{\mathsf{S}})$. It creates a NIZK proof $\pi_{\mathsf{S}} \leftarrow\!\!\$\ \Pi.\mathsf{Prove}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}^{(\mathsf{h})}, 0), \mathbb{w} := (\mathsf{sk}^{(\mathsf{h})}, r_{\mathsf{S}}))$ for the language $\mathcal{L}^{(\mathsf{h})}$. It outputs $\mathsf{sk} := \mathsf{sk}^{(\mathsf{h})}$ and $\mathsf{pk} := (\mathsf{pk}^{(\mathsf{h})}, \pi_{\mathsf{S}})$.

- $\mathsf{KeyGen_U}(\mathsf{pp}) \to (\mathsf{sk_U}, \mathsf{pk_U})$. The user's setup algorithm samples a random value $r_{\mathsf{U}} \leftarrow\!\!\$\ \{0,1\}^\lambda$ and runs $(\mathsf{sk}^{(\mathsf{h})}, \mathsf{pk}^{(\mathsf{h})}) = \mathsf{KeyGen}_{\mathsf{U}}^{(\mathsf{h})}(\mathsf{pp}^{(\mathsf{h})}, 1^\lambda; r_{\mathsf{U}})$. It creates a NIZK proof $\pi_{\mathsf{S}} \leftarrow\!\!\$\ \Pi.\mathsf{Prove}(\mathsf{crs}, \mathbb{x} := (\mathsf{pk}^{(\mathsf{h})}, 1), \mathbb{w} := (\mathsf{sk}^{(\mathsf{h})}, r_{\mathsf{U}}))$ for the language $\mathcal{L}^{(\mathsf{h})}$. It outputs $\mathsf{sk} := \mathsf{sk}^{(\mathsf{h})}$ and $\mathsf{pk} := (\mathsf{pk}^{(\mathsf{h})}, \pi_{\mathsf{U}})$.

- $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_\mathsf{U}, \mathsf{c}) \to \bar{\sigma}$. The issue algorithm parses the user's public key as $(\mathsf{pk}^{(\mathsf{h})}, \pi) \coloneqq \mathsf{pk}_\mathsf{U}$ and runs the NIZK verifier $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathbb{x} \coloneqq (\mathsf{pk}^{(\mathsf{h})}, 1), \pi)$. If the verifier outputs 0, the signer aborts. Otherwise it continues to execute the issue algorithm.

- $\mathsf{Obtain}(\mathsf{sk}_\mathsf{U}, \mathsf{pk}, \bar{\sigma}, \mathsf{c}) \to (\mathsf{m}, \sigma)$. The obtain algorithm parses the signer's public key as $(\mathsf{pk}^{(\mathsf{h})}, \pi) \coloneqq \mathsf{pk}$ and runs the NIZK verifier $\Pi.\mathsf{Verify}(\mathsf{crs}, \mathbb{x} \coloneqq (\mathsf{pk}^{(\mathsf{h})}, 0), \pi_\mathsf{S})$. If the verifier outputs 0, the user aborts. Otherwise it continues to execute the obtain algorithm.

- $\mathsf{Verify}(\mathsf{pk}, \mathsf{m}, \sigma) \to 1/0$. The verification algorithm runs $\mathsf{Verify}^{(\mathsf{h})}(\mathsf{pk}, \mathsf{m}, \sigma)$ and outputs whatever it outputs.

It is intuitively obvious that if $\mathsf{NIBS}^{(\mathsf{h})}$ is a secure NIBS in the honest-key model and Π is an argument of knowledge, then the above construction $\mathsf{NIBS} = (\mathsf{NIBS.Setup}, \mathsf{NIBS.KeyGen}_\mathsf{S}, \mathsf{NIBS.KeyGen}_\mathsf{U}, \mathsf{NIBS.Issue}, \mathsf{NIBS.Obtain}, \mathsf{NIBS.Verify})$ is secure in the standard model. Essentially, in the security proof, instead of receiving the adversary's secret key (as in the honest-key model), the challenger must now run the NIZK extractor to obtain the adversary's secret. The rest of the proof would then proceed identically to that of $\mathsf{NIBS}^{(\mathsf{h})}$.

Bibliography

[1] Aardal, M. A., Aranha, D. F., Boudgoust, K., Kolby, S., and Takahashi, A. (2024). Aggregating falcon signatures with LaBRADOR. In Reyzin, L. and Stebila, D., editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 71–106, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[2] Abdalla, M., Benhamouda, F., and MacKenzie, P. (2015). Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587, San Jose, CA, USA. IEEE Computer Society Press.

[3] Abe, M. (2001). A secure three-move blind signature scheme for polynomially many signatures. In Pfitzmann, B., editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151, Innsbruck, Austria. Springer Berlin Heidelberg, Germany.

[4] Afshar, A., Mohassel, P., Pinkas, B., and Riva, B. (2014). Non-interactive secure computation based on cut-and-choose. In Nguyen, P. Q. and Oswald, E., editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 387–404, Copenhagen, Denmark. Springer Berlin Heidelberg, Germany.

[5] Agarwal, A., Babel, K., Das, S., Juels, A., Rindal, P., and Yadav, A. (2026). Cryptocurrency-backed trustless anonymous tokens and their applications. Cryptology ePrint Archive, Paper 2026/1074.

[6] Agrawal, S., Kirshanova, E., Stehlé, D., and Yadav, A. (2022). Practical, round-optimal lattice-based blind signatures. In Yin, H., Stavrou, A., Cremers, C., and Shi, E., editors, *ACM CCS*

2022: 29th Conference on Computer and Communications Security, pages 39–53, Los Angeles, CA, USA. ACM Press.

[7] Ajtai, M. (1999). Determinism versus non-determinism for linear time RAMs (extended abstract). In *31st Annual ACM Symposium on Theory of Computing*, pages 632–641, Atlanta, GA, USA. ACM Press.

[8] Akamai Technologies (2024). Bots Compose 42% of Overall Web Traffic; Nearly Two-Thirds Are Malicious, Reports Akamai.

[9] Akamai Technologies (2025). AI and LLM Bot Management Has Become a Business-Critical Issue: Do It Right.

[10] Alajaji, R. (2026). Utah's new law targeting VPNs goes into effect may 6th. Updated: 2026-05-11.

[11] Albrecht, M. R., Davidson, A., Deo, A., and Gardham, D. (2024). Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. In Joye, M. and Leander, G., editors, *Advances in Cryptology – EUROCRYPT 2024, Part VI*, volume 14656 of *Lecture Notes in Computer Science*, pages 447–476, Zurich, Switzerland. Springer, Cham, Switzerland.

[12] Albrecht, M. R., Davidson, A., Deo, A., and Smart, N. P. (2021). Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In Garay, J., editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 261–289, Virtual Event. Springer, Cham, Switzerland.

[13] Albrecht, M. R., Grassi, L., Rechberger, C., Roy, A., and Tiessen, T. (2016). MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Cheon, J. H. and Takagi, T., editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, Hanoi, Vietnam. Springer Berlin Heidelberg, Germany.

[14] Albrecht, M. R. and Gür, K. D. (2024). Verifiable oblivious pseudorandom functions from lattices: Practical-ish and thresholdisable. In Chung, K.-M. and Sasaki, Y., editors, *Advances in Cryptology – ASIACRYPT 2024, Part IV*, volume 15487 of *Lecture Notes in Computer Science*, pages 205–237, Kolkata, India. Springer, Singapore, Singapore.

[15] Amjad, G., Yeo, K., and Yung, M. (2023). RSA blind signatures with public metadata. Cryptology ePrint Archive, Report 2023/1199.

[16] Apple (2021). iCloud private relay overview. https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf. Accessed: 2026-05-09.

[17] Apple (2025). icloud private relay overview. https://security.apple.com/documentation/private-cloud-compute. Accessed: 2026-05-09.

[18] Aranha, D. F. and Melissaris, N. (2025). What is cryptography hiding from itself? Cryptology ePrint Archive, Report 2025/1951.

[19] Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., and Wichs, D. (2012). Multiparty computation with low communication, computation and interaction via threshold FHE. In Pointcheval, D. and Johansson, T., editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK. Springer Berlin Heidelberg, Germany.

[20] Au, M. H., Susilo, W., and Mu, Y. (2006). Constant-size dynamic k-TAA. In De Prisco, R. and Yung, M., editors, *SCN 06: 5th International Conference on Security in Communication Networks*, volume 4116 of *Lecture Notes in Computer Science*, pages 111–125, Maiori, Italy. Springer Berlin Heidelberg, Germany.

[21] Au, M. H., Susilo, W., and Mu, Y. (2007a). Practical compact e-cash. In Pieprzyk, J., Ghodosi, H., and Dawson, E., editors, *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 431–445, Townsville, Australia. Springer Berlin Heidelberg, Germany.

[22] Au, M. H., Wu, Q., Susilo, W., and Mu, Y. (2007b). Compact e-cash from bounded accumulator. In Abe, M., editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages 178–195, San Francisco, CA, USA. Springer Berlin Heidelberg, Germany.

[23] Badrinarayanan, S., Jain, A., Ostrovsky, R., and Visconti, I. (2018). Non-interactive secure computation from one-way functions. In Peyrin, T. and Galbraith, S., editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 118–138, Brisbane, Queensland, Australia. Springer, Cham, Switzerland.

[24] Baldimtsi, F., Cheng, J., Goyal, R., and Yadav, A. (2024). Non-interactive blind signatures: Post-quantum and stronger security. In Chung, K.-M. and Sasaki, Y., editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 70–104, Kolkata, India. Springer, Singapore, Singapore.

[25] Baldimtsi, F., Goyal, R., and Yadav, A. (2026a). Batched & non-interactive blind signatures from lattices. In Bai, S. and Persichetti, E., editors, *PKC 2026: 29th International Conference on Theory and Practice of Public Key Cryptography*, Lecture Notes in Computer Science, West Palm Beach, USA. Springer, Cham, Switzerland.

[26] Baldimtsi, F., Hanzlik, L., Nguyen, Q., and Yadav, A. (2026b). Non-interactive anonymous tokens with private metadata bit. In Smaragdakis, G., Liang, K., Cortier, V., and Lin, Z., editors, *ACM CCS 2026: 33rd Conference on Computer and Communications Security (to appear)*, The Hague, The Netherlands. ACM Press.

[27] Baldimtsi, F., Hanzlik, L., and Yadav, A. (2026c). Non-interactive blind signatures with threshold issuance. Cryptology ePrint Archive, Paper 2026/400.

[28] Baldimtsi, F. and Lysyanskaya, A. (2013a). Anonymous credentials light. In Sadeghi, A.-R., Gligor, V. D., and Yung, M., editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 1087–1098, Berlin, Germany. ACM Press.

[29] Baldimtsi, F. and Lysyanskaya, A. (2013b). On the security of one-witness blind signature schemes. In Sako, K. and Sarkar, P., editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 82–99, Bengalore, India. Springer Berlin Heidelberg, Germany.

[30] Baldimtsi, F. and Yadav, A. (2026). Atlantis: Lattice-based anonymous tokens with private metadata bit. In Galdi, C. and Siniscalch, L., editors, *SCN 25: 15th International Conference on Security in Communication Networks (to appear)*, Lecture Notes in Computer Science, Amalfi, Italy. Springer, Cham, Switzerland.

[31] Banerjee, A. and Peikert, C. (2014). New and improved key-homomorphic pseudorandom functions. In Garay, J. A. and Gennaro, R., editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[32] Banerjee, A., Peikert, C., and Rosen, A. (2012). Pseudorandom functions and lattices. In Pointcheval, D. and Johansson, T., editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK. Springer Berlin Heidelberg, Germany.

[33] Barak, B., Brakerski, Z., Komargodski, I., and Kothari, P. K. (2018). Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In Nielsen, J. B. and Rijmen, V., editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 649–679, Tel Aviv, Israel. Springer, Cham, Switzerland.

[34] Barreto, P. S. L. M., Lynn, B., and Scott, M. (2003). Constructing elliptic curves with prescribed embedding degrees. In Cimato, S., Galdi, C., and Persiano, G., editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267, Amalfi, Italy. Springer Berlin Heidelberg, Germany.

[35] Bauer, B., Fuchsbauer, G., and Loss, J. (2020). A classification of computational assumptions in the algebraic group model. In Micciancio, D. and Ristenpart, T., editors, *Advances in*

Cryptology – CRYPTO 2020, Part II, volume 12171 of *Lecture Notes in Computer Science*, pages 121–151, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[36] Bellare, M., Namprempre, C., Pointcheval, D., and Semanko, M. (2002). The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In Syverson, P. F., editor, *FC 2001: 5th International Conference on Financial Cryptography*, volume 2339 of *Lecture Notes in Computer Science*, pages 319–338, Grand Cayman, British West Indies. Springer Berlin Heidelberg, Germany.

[37] Bellare, M., Namprempre, C., Pointcheval, D., and Semanko, M. (2003). The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215.

[38] Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., and Raykova, M. (2021). On the (in)security of ROS. In Canteaut, A. and Standaert, F.-X., editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53, Zagreb, Croatia. Springer, Cham, Switzerland.

[39] Benhamouda, F., Lepoint, T., Orrù, M., and Raykova, M. (2022). Publicly verifiable anonymous tokens with private metadata bit. Cryptology ePrint Archive, Report 2022/004.

[40] Benhamouda, F., Raykova, M., and Seth, K. (2023). Anonymous counting tokens. In Guo, J. and Steinfeld, R., editors, *Advances in Cryptology – ASIACRYPT 2023, Part II*, volume 14439 of *Lecture Notes in Computer Science*, pages 245–278, Guangzhou, China. Springer, Singapore, Singapore.

[41] Beullens, W., Lyubashevsky, V., Nguyen, N. K., and Seiler, G. (2023). Lattice-based blind signatures: Short, efficient, and round-optimal. In Meng, W., Jensen, C. D., Cremers, C., and Kirda, E., editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 16–29, Copenhagen, Denmark. ACM Press.

[42] Beullens, W. and Seiler, G. (2023). LaBRADOR: Compact proofs for R1CS from module-SIS. In Handschuh, H. and Lysyanskaya, A., editors, *Advances in Cryptology – CRYPTO 2023, Part V*,

volume 14085 of *Lecture Notes in Computer Science*, pages 518–548, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[43] Boldyreva, A. (2003). Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Desmedt, Y., editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA. Springer Berlin Heidelberg, Germany.

[44] Boneh, D., Boyen, X., and Shacham, H. (2004). Short group signatures. In Franklin, M., editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[45] Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., and Wu, D. J. (2018). Exploring crypto dark matter: New simple PRF candidates and their applications. In Beimel, A. and Dziembowski, S., editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, Panaji, India. Springer, Cham, Switzerland.

[46] Boneh, D., Lewi, K., Montgomery, H. W., and Raghunathan, A. (2013). Key homomorphic PRFs and their applications. In Canetti, R. and Garay, J. A., editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[47] Boneh, D., Montgomery, H. W., and Raghunathan, A. (2010). Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In Al-Shaer, E., Keromytis, A. D., and Shmatikov, V., editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 131–140, Chicago, Illinois, USA. ACM Press.

[48] Bootle, J., Lyubashevsky, V., Nguyen, N. K., and Sorniotti, A. (2023). A framework for practical anonymous credentials from lattices. In Handschuh, H. and Lysyanskaya, A., editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 384–417, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[49] Bresson, E., Monnerat, J., and Vergnaud, D. (2008). Separation results on the "one-more" computational problems. In Malkin, T., editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 71–87, San Francisco, CA, USA. Springer Berlin Heidelberg, Germany.

[50] Brickell, E. F., Camenisch, J., and Chen, L. (2004). Direct anonymous attestation. In Atluri, V., Pfitzmann, B., and McDaniel, P., editors, *ACM CCS 2004: 11th Conference on Computer and Communications Security*, pages 132–145, Washington, DC, USA. ACM Press.

[51] Brown, D. R. L. (2007). Irreducibility to the one-more evaluation problems: More may be less. Cryptology ePrint Archive, Report 2007/435.

[52] Cadwalladr, C. (2018). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*.

[53] Camenisch, J., Drijvers, M., and Lehmann, A. (2016). Anonymous attestation using the strong Diffie Hellman assumption revisited. Cryptology ePrint Archive, Report 2016/663.

[54] Camenisch, J., Hohenberger, S., and Lysyanskaya, A. (2005a). Compact e-cash. In Cramer, R., editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321, Aarhus, Denmark. Springer Berlin Heidelberg, Germany.

[55] Camenisch, J., Koprowski, M., and Warinschi, B. (2005b). Efficient blind signatures without random oracles. In Blundo, C. and Cimato, S., editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148, Amalfi, Italy. Springer Berlin Heidelberg, Germany.

[56] Canard, S., Gaud, M., and Traoré, J. (2006). Defeating malicious servers in a blind signatures based voting system. In Di Crescenzo, G. and Rubin, A., editors, *FC 2006: 10th International Conference on Financial Cryptography and Data Security*, volume 4107 of *Lecture Notes in Computer Science*, pages 148–153, Anguilla, British West Indies. Springer Berlin Heidelberg, Germany.

[57] Celi, S., Davidson, A., Valdez, S., and Wood, C. A. (2024). Privacy Pass Issuance Protocols. RFC 9578, IETF. Proposed Standard RFC.

[58] Chairattana-Apirom, R., Tessaro, S., and Zhu, C. (2024). Pairing-free blind signatures from CDH assumptions. In Reyzin, L. and Stebila, D., editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 174–209, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[59] Chase, M., Dodis, Y., Ishai, Y., Kraschewski, D., Liu, T., Ostrovsky, R., and Vaikuntanathan, V. (2019). Reusable non-interactive secure computation. In Boldyreva, A. and Micciancio, D., editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 462–488, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[60] Chase, M., Durak, F. B., and Vaudenay, S. (2023). Anonymous tokens with stronger metadata bit hiding from algebraic MACs. In Handschuh, H. and Lysyanskaya, A., editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 418–449, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[61] Chaum, D. (1982). Blind signatures for untraceable payments. In Chaum, D., Rivest, R. L., and Sherman, A. T., editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA. Plenum Press, New York, USA.

[62] Chaum, D. (1983). Blind signature system. In Chaum, D., editor, *Advances in Cryptology – CRYPTO'83*, page 153, Santa Barbara, CA, USA. Plenum Press, New York, USA.

[63] Chaum, D. (1985). Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044.

[64] Chaum, D., Fiat, A., and Naor, M. (1990). Untraceable electronic cash. In Goldwasser, S., editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327, Santa Barbara, CA, USA. Springer, New York, USA.

[65] Cheon, J. H., Cho, W., Kim, J. H., and Kim, J. (2022). Adventures in crypto dark matter: attacks, fixes and analysis for weak pseudorandom functions. *Designs, Codes and Cryptography*, 90(8):1735–1760.

[66] Colorado General Assembly (2026). Senate bill 26-051. Colorado legislature website.

[67] Crites, E. C., Komlo, C., Maller, M., Tessaro, S., and Zhu, C. (2023). Snowblind: A threshold blind signature in pairing-free groups. In Handschuh, H. and Lysyanskaya, A., editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 710–742, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[68] Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., and Valsorda, F. (2018). Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018(3):164–180.

[69] Davidson, A., Iyengar, J., and Wood, C. A. (2024). The Privacy Pass Architecture. RFC 9576, IETF. Informational RFC.

[70] del Pino, R. and Katsumata, S. (2022). A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Dodis, Y. and Shrimpton, T., editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[71] Desmedt, Y. (1988). Society and group oriented cryptography: A new concept. In Pomerance, C., editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[72] Desmedt, Y. and Frankel, Y. (1990). Threshold cryptosystems. In Brassard, G., editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA. Springer, New York, USA.

[73] Diffie, W. and Landau, S. (1997). *Privacy on the Line: The Politics of Wiretapping and Encryption*. MIT Press.

[74] Dinur, I., Goldfeder, S., Halevi, T., Ishai, Y., Kelkar, M., Sharma, V., and Zaverucha, G. (2021). MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In Malkin, T. and Peikert, C., editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 517–547, Virtual Event. Springer, Cham, Switzerland.

[75] Dodis, Y., Ostrovsky, R., Reyzin, L., and Smith, A. (2003). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Cryptology ePrint Archive, Report 2003/235.

[76] Dodis, Y., Reyzin, L., and Smith, A. (2004). Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Cachin, C. and Camenisch, J., editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland. Springer Berlin Heidelberg, Germany.

[77] Dodis, Y. and Yampolskiy, A. (2005). A verifiable random function with short proofs and keys. In Vaudenay, S., editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 416–431, Les Diablerets, Switzerland. Springer Berlin Heidelberg, Germany.

[78] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., and Stehlé, D. (2018). CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268.

[79] Dutton, S. (2020). Getting started with trust tokens. https://web.dev/articles/trust-tokens. Accessed: 2026-05-11.

[80] Ernstberger, J., Chaliasos, S., Kadianakis, G., Steinhorst, S., Jovanovic, P., Gervais, A., Livshits, B., and Orrù, M. (2024). zk-bench: A toolset for comparative evaluation and performance benchmarking of SNARKs. In Galdi, C. and Phan, D. H., editors, *SCN 24: 14th International Conference on Security in Communication Networks, Part I*, volume 14973 of *Lecture Notes in Computer Science*, pages 46–72, Amalfi, Italy. Springer, Cham, Switzerland.

[81] Esgin, M. F., Kuchta, V., Sakzad, A., Steinfeld, R., Zhang, Z., Sun, S., and Chu, S. (2021). Practical post-quantum few-time verifiable random function with applications to algorand. In Borisov, N. and Díaz, C., editors, *FC 2021: 25th International Conference on Financial Cryptography and Data Security, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 560–578, Virtual Event. Springer Berlin Heidelberg, Germany.

[82] Esgin, M. F., Steinfeld, R., Liu, D., and Ruj, S. (2023). Efficient hybrid exact/relaxed lattice proofs and applications to rounding and VRFs. In Handschuh, H. and Lysyanskaya, A., editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 484–517, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[83] European Parliament and Council of the European Union (2021). Regulation (EU) 2021/1232 of the European Parliament and of the Council of 14 July 2021 on a temporary derogation from certain provisions of Directive 2002/58/EC.

[84] Fischlin, M. (2006). Round-optimal composable blind signatures in the common reference string model. In Dwork, C., editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[85] Fischlin, M. and Schröder, D. (2010). On the impossibility of three-move blind signature schemes. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215, French Riviera. Springer Berlin Heidelberg, Germany.

[86] Fuchsbauer, G., Hanser, C., Kamath, C., and Slamanig, D. (2016). Practical round-optimal blind signatures in the standard model from weaker assumptions. In Zikas, V. and De Prisco, R., editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 391–408, Amalfi, Italy. Springer, Cham, Switzerland.

[87] Fuchsbauer, G., Hanser, C., and Slamanig, D. (2015). Practical round-optimal blind signatures in the standard model. In Gennaro, R. and Robshaw, M. J. B., editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 233–253, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[88] Fuchsbauer, G., Hanser, C., and Slamanig, D. (2019). Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546.

[89] Fuchsbauer, G., Kiltz, E., and Loss, J. (2018). The algebraic group model and its applications. In Shacham, H. and Boldyreva, A., editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[90] Fuchsbauer, G., Plouviez, A., and Seurin, Y. (2020). Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model. In Canteaut, A. and Ishai, Y., editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95, Zagreb, Croatia. Springer, Cham, Switzerland.

[91] Ganguly, A., Karmakar, A., Kundu, S., Pal, D., and Sarkar, S. (2025). Revisiting lattice-based non-interactive blind signature. Cryptology ePrint Archive, Report 2025/1848.

[92] Garg, S. and Gupta, D. (2014). Efficient round optimal blind signatures. In Nguyen, P. Q. and Oswald, E., editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 477–495, Copenhagen, Denmark. Springer Berlin Heidelberg, Germany.

[93] Garg, S., Rao, V., Sahai, A., Schröder, D., and Unruh, D. (2011). Round optimal blind signatures. In Rogaway, P., editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 630–648, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[94] Gentry, C., Peikert, C., and Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In Ladner, R. E. and Dwork, C., editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada. ACM Press.

[95] Ghadafi, E. (2017). Efficient round-optimal blind signatures in the standard model. In Kiayias, A., editor, *FC 2017: 21st International Conference on Financial Cryptography and Data Security*, volume 10322 of *Lecture Notes in Computer Science*, pages 455–473, Sliema, Malta. Springer, Cham, Switzerland.

[96] Google (2024). Private state tokens. https://developers.google.com/privacy-sandbox/protections/private-state-tokens. Accessed: 2026-05-11.

[97] Greenwald, G. (2013). Verizon ordered to hand over phone records of millions of Americans to NSA. *The Guardian*.

[98] Greenwald, G. and MacAskill, E. (2013). NSA collecting phone records of millions of Americans daily. *The Guardian*.

[99] Groth, J. and Sahai, A. (2008). Efficient non-interactive proof systems for bilinear groups. In Smart, N. P., editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey. Springer Berlin Heidelberg, Germany.

[100] Hanser, C. and Slamanig, D. (2014). Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Sarkar, P. and Iwata, T., editors, *Advances in Cryptology – ASIACRYPT 2014, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511, Kaoshiung, Taiwan, R.O.C. Springer Berlin Heidelberg, Germany.

[101] Hanzlik, L. (2023). Non-interactive blind signatures for random messages. In Hazay, C. and Stam, M., editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 722–752, Lyon, France. Springer, Cham, Switzerland.

[102] Hanzlik, L., Loss, J., and Wagner, B. (2023). Rai-choo! Evolving blind signatures to the next level. In Hazay, C. and Stam, M., editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*,

volume 14008 of *Lecture Notes in Computer Science*, pages 753–783, Lyon, France. Springer, Cham, Switzerland.

[103] Hanzlik, L., Paracucchi, E., and Zanotto, R. (2025). Non-interactive blind signatures from RSA assumption and more. In Fehr, S. and Fouque, P.-A., editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 365–394, Madrid, Spain. Springer, Cham, Switzerland.

[104] Hazay, C., Katz, J., Koo, C.-Y., and Lindell, Y. (2007). Concurrently-secure blind signatures without random oracles or setup assumptions. In Vadhan, S. P., editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341, Amsterdam, The Netherlands. Springer Berlin Heidelberg, Germany.

[105] Heilman, E., Alshenibr, L., Baldimtsi, F., Scafuro, A., and Goldberg, S. (2017). TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. In *ISOC Network and Distributed System Security Symposium – NDSS 2017*, San Diego, CA, USA. The Internet Society.

[106] Heilman, E., Baldimtsi, F., and Goldberg, S. (2016). Blindly signed contracts: Anonymous on-blockchain and off-blockchain Bitcoin transactions. In Clark, J., Meiklejohn, S., Ryan, P. Y. A., Wallach, D. S., Brenner, M., and Rohloff, K., editors, *FC 2016 Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 43–60, Christ Church, Barbados. Springer Berlin Heidelberg, Germany.

[107] House of Commons Digital, Culture, Media and Sport Committee (2019). Disinformation and 'fake news': Final Report. Accessed: 2026-05-16.

[108] Hughes, E. (1993). A cypherpunk's manifesto. Accessed: 2026-05-09.

[109] IETF (2024). Privacy pass (privacypass). https://datatracker.ietf.org/wg/privacypass/about/. Accessed: 2026-05-11.

[110] Ishai, Y., Kushilevitz, E., Ostrovsky, R., Prabhakaran, M., and Sahai, A. (2011). Efficient non-interactive secure computation. In Paterson, K. G., editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425, Tallinn, Estonia. Springer Berlin Heidelberg, Germany.

[111] Jeudy, C. and Sanders, O. (2024). Improved lattice blind signatures from recycled entropy. Cryptology ePrint Archive, Report 2024/1289.

[112] Jeudy, C. and Sanders, O. (2025). Improved lattice blind signatures from recycled entropy. In Kalai, Y. T. and Kamara, S. F., editors, *Advances in Cryptology – CRYPTO 2025, Part I*, volume 16000 of *Lecture Notes in Computer Science*, pages 477–513, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[113] Juels, A., Luby, M., and Ostrovsky, R. (1997). Security of blind digital signatures (extended abstract). In Kaliski, Jr., B. S., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[114] Kamara, S. (2020). Accessed: 2026-05-20.

[115] Kastner, J., Loss, J., and Xu, J. (2022). On pairing-free blind signature schemes in the algebraic group model. In Hanaoka, G., Shikata, J., and Watanabe, Y., editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 468–497, Virtual Event. Springer, Cham, Switzerland.

[116] Katsumata, S., Nishimaki, R., Yamada, S., and Yamakawa, T. (2021). Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In Canteaut, A. and Standaert, F.-X., editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 404–434, Zagreb, Croatia. Springer, Cham, Switzerland.

[117] Katz, J., Schröder, D., and Yerukhimovich, A. (2011). Impossibility of blind signatures from one-way permutations. In Ishai, Y., editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 615–629, Providence, RI, USA. Springer Berlin Heidelberg, Germany.

[118] Khattak, S., Fifield, D., Afroz, S., Javed, M., Sundaresan, S., McCoy, D., Paxson, V., and Murdoch, S. J. (2016). Do you see what I see? Differential treatment of anonymous users. In *ISOC Network and Distributed System Security Symposium – NDSS 2016*, San Diego, CA, USA. The Internet Society.

[119] Kim, H., Sanders, O., Abdalla, M., and Park, J. H. (2023). Practical dynamic group signatures without knowledge extractors. *Designs, Codes and Cryptography*, 91(3):853–893.

[120] Kim, J., Kim, K., and Lee, C. (2001). An efficient and provably secure threshold blind signature. In *Proceedings of the 4th International Conference Seoul on Information Security and Cryptology*, ICISC '01, page 318–327, Berlin, Heidelberg. Springer-Verlag.

[121] Klooß, M. and Reichle, M. (2025). Blind signatures from proofs of inequality. In Kalai, Y. T. and Kamara, S. F., editors, *Advances in Cryptology – CRYPTO 2025, Part VI*, volume 16005 of *Lecture Notes in Computer Science*, pages 157–189, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[122] Klooß, M., Reichle, M., and Wagner, B. (2024). Practical blind signatures in pairing-free groups. In Chung, K.-M. and Sasaki, Y., editors, *Advances in Cryptology – ASIACRYPT 2024, Part I*, volume 15484 of *Lecture Notes in Computer Science*, pages 363–395, Kolkata, India. Springer, Singapore, Singapore.

[123] Kreuter, B., Lepoint, T., Orrù, M., and Raykova, M. (2020). Anonymous tokens with private metadata bit. In Micciancio, D. and Ristenpart, T., editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 308–336, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[124] Kuchta, V. and Manulis, M. (2015). Rerandomizable threshold blind signatures. In Yung, M., Zhu, L., and Yang, Y., editors, *Trusted Systems*, pages 70–89, Cham. Springer International Publishing.

[125] Lehmann, A., Nazarian, P., and Özbay, C. (2025). Stronger security for threshold blind signatures. In Fehr, S. and Fouque, P.-A., editors, *Advances in Cryptology – EUROCRYPT 2025, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 335–364, Madrid, Spain. Springer, Cham, Switzerland.

[126] Lessig, L. (1999). *Code and Other Laws of Cyberspace*. Basic Books, New York, NY.

[127] Levy, S. (1994). Battle of the Clipper Chip.

[128] Lindell, Y. (2003). Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692, San Diego, CA, USA. ACM Press.

[129] Lindell, Y. (2008). Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21(2):200–249.

[130] Lyubashevsky, V. (2012). Lattice signatures without trapdoors. In Pointcheval, D. and Johansson, T., editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK. Springer Berlin Heidelberg, Germany.

[131] Lyubashevsky, V., Nguyen, N. K., and Plançon, M. (2022a). Efficient lattice-based blind signatures via gaussian one-time signatures. In Hanaoka, G., Shikata, J., and Watanabe, Y., editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 498–527, Virtual Event. Springer, Cham, Switzerland.

[132] Lyubashevsky, V., Nguyen, N. K., and Plançon, M. (2022b). Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Dodis, Y. and Shrimpton, T., editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 71–101, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[133] Lyubashevsky, V., Nguyen, N. K., Plançon, M., and Seiler, G. (2021). Shorter lattice-based group signatures via "almost free" encryption and other optimizations. In Tibouchi, M. and Wang, H., editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 218–248, Singapore. Springer, Cham, Switzerland.

[134] Lyubashevsky, V., Palacio, A., and Segev, G. (2010). Public-key cryptographic primitives provably as secure as subset sum. In Micciancio, D., editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 382–400, Zurich, Switzerland. Springer Berlin Heidelberg, Germany.

[135] Meunier, T., Rubin, C. D., and Faz-Hernández, A. (2024). Privacy pass: upgrading to the latest protocol version. https://blog.cloudflare.com/privacy-pass-standard. Accessed: 2026-05-11.

[136] Micali, S., Ohta, K., and Reyzin, L. (2001). Accountable-subgroup multisignatures: Extended abstract. In Reiter, M. K. and Samarati, P., editors, *ACM CCS 2001: 8th Conference on Computer and Communications Security*, pages 245–254, Philadelphia, PA, USA. ACM Press.

[137] Micciancio, D. and Regev, O. (2004). Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381, Rome, Italy. IEEE Computer Society Press.

[138] Mitsunari, S. (2024). MCL. https://github.com/herumi/mcl.

[139] Mohassel, P. and Rosulek, M. (2017). Non-interactive secure 2PC in the offline/online and batch settings. In Coron, J.-S. and Nielsen, J. B., editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 425–455, Paris, France. Springer, Cham, Switzerland.

[140] Moody, D., Perlner, R., Regenscheid, A., Robinson, A., and Cooper, D. (2024). Transition to Post-Quantum Cryptography Standards. Technical Report NIST IR 8547 (Initial Public Draft), National Institute of Standards and Technology, Gaithersburg, MD.

[141] Mukherjee, P. and Wichs, D. (2016). Two round multiparty computation via multi-key FHE. In Fischlin, M. and Coron, J.-S., editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 735–763, Vienna, Austria. Springer Berlin Heidelberg, Germany.

[142] Ohkubo, M. and Abe, M. (2003). Security of some three-move blind signature schemes reconsidered. In *The 2003 Symposium on Cryptography and Information Security.*

[143] Orrù, M., Tessaro, S., Zaverucha, G., and Zhu, C. (2024). Oblivious issuance of proofs. In Reyzin, L. and Stebila, D., editors, *Advances in Cryptology – CRYPTO 2024, Part IX*, volume 14928 of *Lecture Notes in Computer Science*, pages 254–287, Santa Barbara, CA, USA. Springer, Cham, Switzerland.

[144] Ostrovsky, R., Paskin-Cherniavsky, A., and Paskin-Cherniavsky, B. (2014). Maliciously circuit-private FHE. In Garay, J. A. and Gennaro, R., editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA. Springer Berlin Heidelberg, Germany.

[145] Pass, R. (2011). Limits of provable security from standard assumptions. In Fortnow, L. and Vadhan, S. P., editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 109–118, San Jose, CA, USA. ACM Press.

[146] Pauly, T., Valdez, S., and Wood, C. A. (2024). The Privacy Pass HTTP Authentication Scheme. RFC 9577, IETF. Standards Track RFC.

[147] Pfitzmann, B. and Sadeghi, A.-R. (2000). Anonymous fingerprinting with direct nonrepudiation. In Okamoto, T., editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 401–414, Kyoto, Japan. Springer Berlin Heidelberg, Germany.

[148] Pointcheval, D. and Sanders, O. (2016). Short randomizable signatures. In Sako, K., editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 111–126, San Francisco, CA, USA. Springer, Cham, Switzerland.

[149] Pointcheval, D. and Stern, J. (1996). Provably secure blind signature schemes. In Kim, K. and Matsumoto, T., editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265, Kyongju, Korea. Springer Berlin Heidelberg, Germany.

[150] Pointcheval, D. and Stern, J. (2000). Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396.

[151] Policharla, G.-V., Westerbaan, B., Faz-Hernández, A., and Wood, C. A. (2023). Post-quantum privacy pass via post-quantum anonymous credentials. Cryptology ePrint Archive, Report 2023/414.

[152] Prest, T., Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. (2022). FALCON. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[153] Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. In Gabow, H. N. and Fagin, R., editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA. ACM Press.

[154] Rogaway, P. (2015). The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162.

[155] Rosenbloom, L. N. (2023a). Cryptography and collective power. In Aly, A. and Tibouchi, M., editors, *Progress in Cryptology - LATINCRYPT 2023: 8th International Conference on Cryptology and Information Security in Latin America*, volume 14168 of *Lecture Notes in Computer Science*, pages 3–39, Quito, Ecuador. Springer, Cham, Switzerland.

[156] Rosenbloom, L. N. (2023b). A living framework for abolitionist teaching in computer science. In *Proceedings of the ACM Conference on Global Computing Education Vol 1*, CompEd 2023, page 133–139, New York, NY, USA. Association for Computing Machinery.

[157] Schnorr, C.-P. (1990). Efficient identification and signatures for smart cards. In Brassard, G., editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA. Springer, New York, USA.

[158] Schröder, D. and Unruh, D. (2012). Security of blind signatures revisited. In Fischlin, M., Buchmann, J., and Manulis, M., editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 662–679, Darmstadt, Germany. Springer Berlin Heidelberg, Germany.

[159] Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Seiler, G., Stehlé, D., and Ding, J. (2022). CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[160] Seo, J. H. and Cheon, J. H. (2012). Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Cramer, R., editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 133–150, Taormina, Sicily, Italy. Springer Berlin Heidelberg, Germany.

[161] Silde, T. and Strand, M. (2022). Anonymous tokens with public metadata and applications to private contact tracing. In Eyal, I. and Garay, J. A., editors, *FC 2022: 26th International Conference on Financial Cryptography and Data Security*, volume 13411 of *Lecture Notes in Computer Science*, pages 179–199, Grenada. Springer, Cham, Switzerland.

[162] Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., and Danezis, G. (2019). Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *ISOC Network and Distributed System Security Symposium – NDSS 2019*, San Diego, CA, USA. The Internet Society.

[163] Tessaro, S. and Zhu, C. (2022). Short pairing-free blind signatures with exponential security. In Dunkelman, O. and Dziembowski, S., editors, *Advances in Cryptology – EUROCRYPT 2022,*

Part II, volume 13276 of *Lecture Notes in Computer Science*, pages 782–811, Trondheim, Norway. Springer, Cham, Switzerland.

[164] Tyagi, N., Celi, S., Ristenpart, T., Sullivan, N., Tessaro, S., and Wood, C. A. (2022). A fast and simple partially oblivious PRF, with applications. In Dunkelman, O. and Dziembowski, S., editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 674–705, Trondheim, Norway. Springer, Cham, Switzerland.

[165] U.S. Court of Appeals for the Ninth Circuit (1999). Bernstein v. United States Department of Justice. Docket: 97-16686. Opinion by Judge Betty B. Fletcher. Argued December 8, 1997, San Francisco, California.

[166] Vo, D. L., Zhang, F., and Kim, K. (2003). A new threshold blind signature scheme from pairings. In *The 2003 Symposium on Cryptography and Information Security, SCIS 2003, Hamamatsu, Japan, Jan 26-29, 2003 The Institute of Electronics, Information and Communication Engineers*.

[167] Wilander, J. (2021). Introducing private click measurement, pcm. https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/. Accessed: 2025-05-11.

[168] Yan, Y., Chow, S. S. M., Ng, L. K. L., Wong, H. W. H., Zhao, Y., and Wang, B. (2025). Batch anonymous MAC tokens from lattices. In Niederhagen, R. and Saarinen, M.-J. O., editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Part I*, pages 349–384, Taipei, Taiwan. Springer, Cham, Switzerland.

[169] Zhang, H., Chen, X., and Huang, Q. (2025). Lattice-based non-interactive blind signature schemes in the random oracle model. In Liu, J. K., Chen, L., Sun, S.-F., and Liu, X., editors, *Provable and Practical Security*, pages 289–308, Singapore. Springer Nature Singapore.

Biography

Aayush Yadav received a Bachelor's degree in Computer Engineering from the University of Pune in 2017 and a Master's degree in Computer Science from Rutgers University–Camden in 2021. He was a PhD Research Intern at Mysten Labs, Category Labs, and the Input Output Group during the summers of 2024–2026 respectively. From 2023 through 2026, Aayush was also a long-term visiting fellow at the University of Wisconsin, Madison. He will be returning to Rutgers University–Camden as an Assistant Professor of Computer Science.