Non-interactive Blind Signatures from Lattices

Foteini Baldimtsi  Jiaqi Cheng  Rishab Goyal
George Mason University*  UW-Madison†  UW-Madison‡

Aayush Yadav
George Mason University§

Abstract

Blind signatures enable a receiver to obtain signatures on messages of its choice without revealing any message to the signer. Round-optimal blind signatures are designed as a two-round interactive protocol between a signer and receiver. Coincidentally, the choice of message is not important in many applications, and is routinely set as a random (unstructured) message by a receiver.

With the goal of designing more efficient blind signatures for such applications, Hanzlik (EUROCRYPT '23) introduced a new variant called non-interactive blind signatures (NIBS). These allow a signer to asynchronously generate partial signatures for any recipient such that only the intended recipient can extract a blinded signature for a random message. This bypasses the two-round barrier for traditional blind signatures, yet enables many known applications. Hanzlik provided new practical designs for NIBS from bilinear pairings.

In this work, we investigate efficient NIBS with post-quantum security. We design the first practical NIBS, as well as non-interactive partially blind signatures called tagged NIBS, from lattice-based assumptions. We also propose a new generic paradigm for NIBS from circuit-private leveled homomorphic encryption achieving optimal-sized signatures (i.e., same as any non-blind signature). Finally, we propose new enhanced security properties for NIBS, that could be of practical and theoretical interest.

*Email: foteini@gmu.edu. This research is supported by NSF #2143287.
†Email: jcheng77@wisc.edu.
‡Email: rishab@cs.wisc.edu. Support for this research was provided by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.
§Email: ayadav5@gmu.edu. Work done while visiting UW-Madison. Funded by NSF #2143287.
## Contents

1 Introduction 4

2 Technical Overview 7
   2.1 Defining non-interactive blind signatures 7
   2.2 Additional security properties 8
   2.3 Extending Fischlin’s paradigm to NIBS 9
   2.4 Making Fischlin-based NIBS practical and post-quantum 10
   2.5 Security and the randomized OM-ISIS assumption 11
   2.6 NIBS from circuit-private LHE 16
   2.7 Efficiency comparisons for our NIBS schemes 16

3 Preliminaries 17
   3.1 Lattice preliminaries 17
   3.2 Hardness assumptions 19
   3.3 One-more ISIS assumption 19
   3.4 Randomness extraction 20
   3.5 Non-interactive zero knowledge (NIZK) 20

4 The Randomized One-more ISIS Assumption 21
   4.1 Cryptanalysis of the rOM-ISIS assumption 23
      4.1.1 Recalling attacks on the one-more ISIS assumption 23
      4.1.2 Attacking the rOM-ISIS assumption 24

5 Non-Interactive Blind Signatures 27

6 Lattice-based NIBS 30
   6.1 Construction 31
   6.2 One-more unforgeability 32
   6.3 Receiver blindness 35
   6.4 Efficiency of lattice-based NIBS 39

7 Lattice-based Tagged NIBS 41
   7.1 Tagged NIBS definitions 41
   7.2 Construction 42
   7.3 One-more unforgeability 43
   7.4 Receiver blindness 46

8 Lattice-based NIBS with Nonce Blindness 48
   8.1 Recalling nonce blindness definition 48
   8.2 Construction 49
   8.3 Nonce blindness 51
   8.4 Receiver blindness 55
8.5 One-more unforgeability .................................................. 57

9 NIBS from Circuit Private LHE ........................................ 59
  9.1 Leveled homomorphic encryption .................................. 59
  9.2 Knowledge of secret keys model .................................. 60
  9.3 Construction .......................................................... 61
  9.4 One-more unforgeability ............................................. 63
  9.5 Strong receiver blindness ........................................... 65
  9.6 Strong nonce blindness ............................................. 70

A Preliminaries (continued) .......................................... 79
  A.1 Cryptographic building blocks .................................... 79
     A.1.1 Pseudo Random Functions .................................. 79
     A.1.2 Perfectly binding commitments .............................. 79
     A.1.3 Public key encryption ....................................... 80
     A.1.4 Signature schemes ........................................... 80

B NIBS from General Purpose NIZKs ................................. 81
  B.1 Construction ....................................................... 81
  B.2 One-more unforgeability ......................................... 83
  B.3 Strong receiver blindness ....................................... 84
  B.4 Strong nonce blindness ......................................... 88

C Knowledge of Secret Key Assumption ............................ 90
1 Introduction

A blind signature scheme is a special class of digital signatures that allows a receiver to obtain a signature on a message of the receiver’s choice without revealing this message to the signer. Typically, blind signatures are implemented as an interactive protocol between the signer (holding a secret key sk) and the receiver (holding the message m to be signed and the signer’s public verification key vk). At the end of their interaction, the receiver should obtain a valid signature σ on m. The protocol needs to be correct (i.e., the output signature can be verified using vk and m) and additionally, it needs to satisfy two security properties: blindness (i.e., the signer does not obtain any information with respect to m) and one-more unforgeability (i.e., the receiver cannot create more valid signatures beyond the ones received after interacting with the signer).

Blind signatures were first introduced by Chaum [Cha83] and have been used as a main building block in numerous privacy-preserving applications. One of the most prominent applications, and the one originally considered by Chaum [Cha83] is private electronic payments (e-cash). The idea is that in an e-cash system, a bank (a.k.a. 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. The blindness property guarantees that nobody (including the bank) can trace the spending of user coins, while the unforgeability property guarantees that users cannot forge coins. Beyond e-cash, blind signatures have found applications in e-voting systems [CGT06], anonymous credentials/tokens [PZ13, BL13, FHS15, DGS+18], direct anonymous attestation [BCC04] and coin tumblers for cryptocurrencies [HBG16, HAB+17].

Two-move barrier. The original blind signature scheme by Chaum [Cha83] requires two-moves of interaction: the recipient sends the blinded message to the signer and the signer responds with a signature on the blinded message. The recipient locally un-blinds the signature and outputs the final message/signature pair. It is easy to see that blind signatures are impossible unless the recipient and signer both send at least one message. Thus, two-move blind signatures are considered round-optimal and are particularly interesting for real-world applications. Beyond their optimal communication efficiency, their unforgeability automatically extends to the concurrent setting [Lin08, HKKL07] (i.e. even when the adversary engages in multiple concurrent sessions with the signer). There has been a long line of research on round-optimal blind signatures under different models and assumptions [BNPS02, Bol03, Fis06, GRS+11, SC12, PHS15, FHKS16, Gha17, KNYY21, AKSY22, BLNS23a].

Non-interactive blind signatures: beyond the two-move barrier. In a two-move blind signature scheme, the first message originates from the receiver and typically includes a blinded version of the message to be signed. Interestingly though, in many applications, the signed message is selected randomly and does not come from a specific distribution or has a specific structure. Consider e-cash as an example, where the signed message is a random value denoting the e-coin ID, arbitrarily selected by a user. This observation was recently made explicit by Hanzlik [Han23] who put forward a new notion called non-interactive blind signatures (NIBS).

In a NIBS system, each receiver is associated with a public-secret key pair (pk_R, sk_R) such that any signer can asynchronously create partial signatures psig, called presignatures, given only the
receiver’s public key $pk_R$. The receiver can extract a pair of blinded signature $\sigma$ and message $\mu$ from presignature $psig$ using its corresponding secret key $sk_R$. Completeness states that $\sigma$ is a valid signature for message $\mu$, and can be verified given only signer’s verification key $vk$. The message $\mu$ should be an unpredictable message for the receiver and the signer.

One can naturally extend the desired security properties of (one-more) unforgeability and blindness to the non-interactive setting. As pointed out in [Han23], using some secret input from the receiver (i.e. $sk_R$) during the computation of the final signature is crucial to achieve blindness without incurring two-moves. Additionally, Hanzlik also extended the idea of NIBS to tagged non-interactive blind signatures (TNIBS) to resemble the functionality of partially blind signatures. That is, signatures that allow for the inclusion of some un-blinded metadata, called the ‘tag’, along with the blinded signature. For example, the tag could contain application specific public information such as date, time, purpose, etc.

Non-interactive blind signatures could replace traditional blind signatures in any application where the choice of message is not important, and could be set as a random (unstructured) message. This would lead to protocols with minimal communication complexity. Beyond the case of e-cash, (tagged) NIBS can be used to implement anonymous token systems, à la Privacy Pass [DGS+18], and lottery systems — we refer to [Han23] for a longer discussion on applications. Interestingly, if we consider the PKI model, then a recipient never needs to interact with a signer to send its public key. Thus, a signer can issue and publish presignatures for users without ever interacting with them. This property makes NIBS suitable for many modern applications. As an example, consider cryptocurrency airdrops: a mechanism to gift coins to users (typically used to bootstrap interest in a new coin). A cryptocurrency could distribute coins to registered user public keys by creating and publishing presignatures, and then users could obtain the final signatures (i.e., the actual coins) in a privacy preserving way.

Existing approaches. [Han23] provided a generic template for building (T)NIBS in the random oracle (RO) model from verifiable random functions, digital signatures, and general purpose dual-mode non-interactive witness indistinguishable proofs. They also designed a practically efficient NIBS scheme from signatures on equivalence classes [HS14, FHS19], and a TNIBS from tag-based equivalence class signatures [HS21]. Both schemes crucially rely on the use of bilinear pairings for instantiating the underlying equivalence-class signature. Moreover, their security proofs are carried out in the generic group model.

This work. The current state-of-the-art is that we do not have efficient post-quantum NIBS, even with only conjectured security. This was left as an important open problem in [Han23] and brings us to the central goal of our work:

Can we design efficient (tagged) NIBS from post-quantum assumptions?

We answer the above question in the affirmative, and believe this will lead to further progress in the emerging area of practical (round-optimal) blind signature schemes with post-quantum security (see [LNP22a, AKSY22] and references therein). We summarize our contributions below.

1. We provide the first practical lattice-based NIBS and TNIBS systems. The concrete costs and overhead of our NIBS scheme are similar to that for state-of-the-art interactive (round-optimal)
blind signatures from lattices. We prove security of our systems under a more robust variant of the one-more-ISIS assumption [AKSY22], that we call randomized one-more-ISIS assumption. We do preliminary cryptanalysis of our proposed assumption, and show that all known attacks on the one-more-ISIS assumption [AKSY22] are unsuccessful in breaking our variant. Moreover, we have been unable to find any attack that runs in time faster than needed to break standard ISIS assumption.

2. We propose a new generic paradigm for designing (non-interactive) blind signatures with optimal-sized signatures from any circuit-private leveled homomorphic encryption (LHE) scheme. We show that LHE can upgrade any regular (non-blind) signature into a NIBS without increasing the size of the final signature. While our practically efficient construction uses a different design template, we believe our LHE-based template could serve as an alternative to the famous Fischlin’s paradigm for blind signatures. And, with great ongoing research in the realm of (somewhat/leveled) homomorphic encryption, we believe our proposed template might lead to alternate approaches to efficient (non-interactive) blind signatures in the future.

3. As an additional contribution, we present new stronger blindness properties for NIBS. Our study of stronger blindness definitions is motivated by scenarios in which a blindness attacker might be able to get a hold of some presignatures along with their corresponding message-signature pairs. At a high level, this can be viewed as a CCA-style blindness guarantee, where existing blindness definitions provide a CPA-style blindness guarantee. We also provide new feasibility results for NIBS secure in such stronger corruption models, both from LHE and NIZKs.

Related work. Recently, many new approaches for lattice-based round-optimal blind signatures schemes have been proposed [LNP22a, dK22, AKSY22, BLNS23a]. Lyubashevsky et al. [LNP22a] use one-time signatures, and build blind signatures under standard lattice assumptions (MSIS, MLWE). However, their scheme only supports bounded polynomial number of signatures per public key with each signature being around 150 KB in size. This was improved by del Pino and Katsumata [dK22] who adapted Fischlin’s paradigm [Fis06] to the lattice setting. The resulting scheme allows for an unbounded number of signatures, of around 102 KB each, and is also secure under standard lattice assumptions. Agrawal et al. [AKSY22] significantly improved both the signature, as well as the transcript size, by also leveraging Fischlin’s paradigm, but relying on efficient lattice-based NIZK for linear relations [LNP22b]. Their final signature size is 45 KB and the total transcript is just over 1 KB. The security of their scheme is based on the one-more ISIS assumption [AKSY22]. Most recently, Beullens et al. [BLNS23a] improved the Agrawal et al. scheme to off-loaded some inefficient computation to the receiver’s first message. This has the twin benefit of simultaneously allowing a reduction to standard lattice assumption, as well as reducing the signature size to just 22 KB. However, this came at the cost of the receiver having to prove the validity of cryptographic hash function input-output pair in its first message. This makes the first message significantly less efficient at a few hundred kilobytes. We compare the concrete cost of our NIBS construction with the current state-of-the-art round-optimal blind signatures from lattices in Table 1.
2 Technical Overview

In this section, we provide a high-level overview of our approach and main technical contributions. We start by recalling the notion of non-interactive blind signatures.

2.1 Defining non-interactive blind signatures

In a NIBS system, the Setup algorithm generates the system’s (global) public parameters $pp$ (viewed as CRS in-the-sky). There are two key generation algorithms, $\text{KeyGen}_S \rightarrow (sk, vk)$, $\text{KeyGen}_R \rightarrow (sk_R, pk_R)$, for the signer and receiver, respectively. Given $sk$, the signer runs a (randomized) Issue algorithm for any receiver’s public key $pk_R$ to compute a presignature, $psig$ and nonce. Here nonce roughly denotes the randomness used by the signer. The receiver then runs the Obtain algorithm on the above presignature to compute the final message-signature pair $(\mu, \sigma)$ using secret key $sk_R$.

An essential property of NIBS is reusability, which says that a signer can issue multiple distinct presignatures (leading to multiple distinct message-signature pairs) for the same receiver public key $pk_R$. Hanzlik [Han23] attempted to capture this property by providing a random nonce value as input to the Issue algorithm. Unfortunately, as we explain in Section 5, the existing formulation is insufficient in capturing the desired reusability property. The issue is that, under the current formulation, there can exist trivial NIBS scheme where Issue and Obtain algorithms simply ignore nonce, thus lead to a limited one-use system. To fix this, we formalize reusability as its own property. Informally, it says that any receiver should obtain two distinct message-signature pairs for two distinct presignature-nonce pairs for any given receiver. Fortunately, the bilinear-based NIBS schemes [Han23] already satisfy reusability, but just did not prove/define it formally.

For security, we want NIBS to satisfy one-more unforgeability and blindness. One-more unforgeability for NIBS can be defined as a natural extension of the one-more unforgeability for traditional blind signatures [PS96]: the adversary gets access to a presignature oracle, and after receiving $\ell$ presignatures for recipient public keys specified by the adversary, the adversary must return $\ell + 1$ valid signatures. On the other hand, defining blindness for NIBS is much more nuanced than for traditional blind signatures\(^2\). Intuitively, it is captured by defining unlinkability between presignatures and final signatures. More formally, Hanzlik [Han23] captures it via the following experiment: the adversary receives two recipient public keys $pk_{R_0}, pk_{R_1}$, and outputs two presignatures $psig_0, psig_1$ (one for each). The challenger extracts the final signature-message pair for each output.

---

\(^1\)Our syntax is nearly identical to that proposed in [Han23]. We deviate slightly in the handling of nonce as we discuss in this section, and also later in Section 5.

\(^2\)Essentially, this is because the receiver does not choose the message in the non-interactive setting.
\((\mu_0, \sigma_0), (\mu_1, \sigma_1)\) from these, and the scheme is said to satisfy receiver blindness if the attacker cannot link the final signature-message pairs to the presignatures. Abstractly, receiver blindness could be thought of as an ‘inter-receiver’ blindness property, since it guarantees that a malicious signer cannot figure out the recipient of any final (blinded) signature.

[Han23] also extended the above basic notion to tagged NIBS (TNIBS). The goal was to capture the notion of partially blind signatures [AP96] which allow a signer and recipient to jointly agree on a public metadata/value to be included as part of the signed message. This is captured by allowing such a common metadata/value, called a tag, to be chosen explicitly by the signer, and shared with the receiver along with the presignature. Syntactically, TNIBS is defined identically to NIBS, except Issue, Obtain, and Verify algorithms take tag \(\tau\) as an additional input.

The aforementioned properties of reusability, one-more unforgeability, and receiver blindness, along with the extension to the partially blind setting, suffice for all known applications of (T)NIBS. Thus, in this work, we mostly keep our focus on designing efficient (T)NIBS satisfying these three properties from lattice-based assumptions. However, in the spirit of studying interesting strengthenings of NIBS, we also discuss some additional security properties proposed in [Han23], as well as some CCA-style blindness notions that we introduce in this work.

### 2.2 Additional security properties

Hanzlik [Han23] also proposed a secondary blindness notion, called nonce blindness. This can be viewed as an intra-recipient blindness property, where what we want is that a signer issuing more than one presignature to a specific user should not be able to link the final message-signature pairs to the corresponding presignatures. Essentially, this provides some flavor of “ordering” unlinkability. While it is unclear whether there are any interesting applications where this could be useful in practice\(^3\), we provide lattice-based NIBS constructions with nonce blindness for completeness.

We remark that nonce blindness, as defined in [Han23], only guarantees unlinkability of 2 presignatures with their corresponding message-signature pairs. That is, a NIBS scheme secure under the existing nonce blindness might not satisfy unlinkability when given three (or more) presignatures instead. Later, we prove security of our lattice-based NIBS under the basic nonce blindness definition. However, it can be easily extended to a general \(k\)-nonce blindness version at the cost of slightly larger parameters. Alternatively, one could consider a weaker third-party blindness notion, where one wants any third party passively observing a set of presignatures to not be able to link them to any particular receiver. This could be a desirable property in practice, and it can be achieved efficiently quite simply by additionally encrypting each presignature under (encryption) public key of a receiver, as long as the encryption system is anonymous\(^4\).

Finally, we propose a new stronger security framework for NIBS blindness properties. The motivation is to capture scenarios where an adversary is able to learn some correlation between

---

\(^3\)Consider an extreme scenario, where say an e-cash signer issues presignatures to a single user, and the user wants the signer to not be able to link those presignatures to e-coins (final signatures), say to prevent some sort of timing attack (i.e. the coin used today was issued at a specific time in the past). While nonce blindness could be useful in this scenario, it does not seem relevant to most real-world applications since the signer will typically interact with multiple receivers at the time.

\(^4\)Anonymous public-key encryption says one cannot link ciphertexts and public keys, and is known from a variety of standard assumptions.
presignature-nonce pairs and their corresponding blind signature-message pairs. We formulate this
by providing an adversary oracle access to the receiver’s secret key in the form of Obtain queries.
We call these as strong receiver (resp. nonce) blindness properties and, at a high level, these can
be view as CCA-version of blindness (since adversary gets query access to the secret key like in
IND-CCA). These definitions guarantee blindness to hold even when a malicious signer is able to
bypass blindness of signatures from previous sessions. We prove such stronger forms of blindness for
some of our constructions to indicate their feasibility, and leave the problem of designing practical
NIBS with stronger blindness as an exciting open problem.

2.3 Extending Fischlin’s paradigm to NIBS

In this section, we start describing our main technical ideas. Our starting point is the well-known
Fischlin’s paradigm [Fis06] for constructing traditional blind signatures. At a high level, Fischlin's
scheme assumes a common reference string (CRS), a standard signature scheme $S$, an encryption
scheme $E$, a commitment scheme COM and general NIZK proofs.

The CRS contains an encryption public key $pk_E$, while each signer simply generates its key pair
$(sk, vk)$ by running the setup of the signature scheme. The user computes $c = \text{COM}(m)$ and sends
c to the signer. The signer runs $\text{Sign}(sk, c)$ to produce $\sigma'$ and sends $\sigma'$ to the user. If the signature
verifies, the user computes $ct = \text{Enc}_{pk_E}(c \parallel \sigma')$ and outputs $ct, m$ and a NIZK $\pi$ proving knowledge
of signature $\sigma'$ and message $m$ such that $\sigma'$ is a signature on a commitment of $m$. Intuitively,
the one-more unforgeability follows from binding of the commitment scheme and unforgeability of
the signature scheme, via straight-line extraction enabled by the trapdoor key $sk_E$ corresponding
to $pk_E$. Further, blindness follows from the semantic security of the encryption scheme, zero-
knowledge of the NIZK, and hiding property of the commitment. Consider the following natural
extension of Fischlin’s paradigm to the non-interactive setting.

Adapting Fischlin’s paradigm to NIBS. Given that there is no interaction between the signer
and the user, the receiver’s public key will, in a sense, replace the commitment submitted in the
first move. The challenge now is that the commitment can no longer represent a commitment of a
single message but has to be a succinct commitment of an exponential number of messages at once.
The hope is that signer can sign the commitment in a way such that the signature obliviously binds
to exactly one of those messages at random. This binding to one message from an exponential set
is important for one-more unforgeability as otherwise the receiver could cheat. Lastly, the receiver
can use NIZKs to prove knowledge of the signature and reveals the choice of message that was
obliviously selected.

Thus, to extend the above template, we need to overcome two challenges: (a) how to commit
to an exponential number of messages efficiently, and (b) how can the signer obliviously select one
of those messages. Our approach is to set the receiver’s public key $pk_R$ as a commitment to a
PRF key $K$, and the key $K$ (along with its opening) as the secret key. Note that this implicitly
defines the exponential set of messages as all the outputs of the PRF. Then, during the issuance of
a presignature, the signer can obliviously sample one of the messages by selecting a random input
$r$ and signing it along with $pk_R$. The user obtains a final signature on the message $m = F_K(r)$
and along with $m$ it outputs $\sigma$ to be a NIZK proof of knowledge of a (pre)signature on $pk_R$ and $r$
corresponding to the message \( m \). 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. Blindness properties reduce to the zero-knowledge property of the underlying NIZK and hiding of the commitment scheme.

**Remark 2.1.** Hanzlik [Han23] proposed a similar generic template for NIBS using verifiable random functions and general-purpose dual mode witness indistinguishable proofs. Our construction is notably simpler as it only requires a PRF and general-purpose NIZK. For completeness, we provide the full construction in Appendix B.

### 2.4 Making Fischlin-based NIBS practical and post-quantum

Next, we develop ideas to make the generic Fischlin-based NIBS construction concretely practical. With this goal, we refer to the recent work by Agrawal et al. [AKSY22] on practical round-optimal lattice-based blind signatures. Agrawal et al. suggested that the usage of NIZKs is somewhat unavoidable when designing round-optimal blind signatures from lattices\(^5\). Thus, their main insight was that, by optimizing the NP language for which NIZKs are needed, one could instantiate Fischlin’s paradigm efficiently. In particular, they showed that NIZKs for linear relations, which are already known to be efficiently implementable from lattice assumptions [LNP22b], are sufficient for building practically efficient round-optimal blind signatures.

Our initial attempt is to follow a similar approach. We start by optimizing our NIZK-based template for designing NIBS, similar to how [AKSY22] optimized Fischlin’s paradigm. The challenge here is to remove all inefficient generic cryptographic components currently used in our NIZK relation, and implement them via just linear relations. As a starting point, consider the standard hash-then-sign paradigm. A signer samples lattice trapdoor as \((C, T_C)\) and outputs \( C \) as verification key \( \text{vk} \). Receiver starts by randomly selecting a PRF key \( K \), and sets its public key \( \text{pk}_R = \text{Com}(K; s) \) as a commitment of \( K \) with its secret key \( \text{sk}_R = (K, s) \), for some random string \( s \). Signer then assigns presignatures to receiver: it uniformly samples randomness \( r \) as nonce, and provides a short preimage of \( H(\text{pk}_R || r) \) as the presignature, where \( C : z = H(\text{pk}_R || r) \) and \( \text{psig} = z \), nonce = \( r \). Finally, receiver generates message \( \mu \) as \( F_K(r) \), and signature \( \pi \) as a NIZK proof establishing that \( z = C^{-1}(H(\text{pk}_R || r)), \mu = F_K(r) \) and \( z \) is short.

Such a design essentially instantiates our NIBS template with the lattice-based signature scheme in [GPV08]. Observe that the major source of inefficiency in the NIZK arises from the hash evaluation and the PRF evaluation. These evaluations are extremely heavy and thus not very practical to prove in a NIZK. Our strategy is to entirely exclude both computations from the NIZK relation. Following from the blueprint of [AKSY22], it is possible to remove the hash evaluation (from the NIZK relation) if we fix the receiver’s public key as \( \text{pk}_R = A \cdot x + H(\delta) \) for some randomly sampled \( x \) and hash input \( \delta \), where \( A \) is a random matrix part of the CRS. Evaluations of \( H(\delta) \) could then be performed outside of the NIZK proof system, if the signature contains \( \delta \) in the clear. This removes the need to prove costly hash evaluations using the NIZK. The reason this does not

---

\(^5\)This is due to the fact that all existing practical lattice-based signature schemes do not support simple algebraic homomorphisms in a practically efficient way, unlike signatures from pairings or factoring-based assumptions.
break blindness is that even if the signer learns $\delta$, it still cannot link $\delta$ with $pk_R$ as $x$ contains enough entropy so that $A \cdot x$ statistically hides all information about $H(\delta)$.

However, at this point, the parallels between our design and the interactive blind signature scheme in [AKSY22] start to diminish. This is because, in the interactive setting, a receiver can simply select $\delta$ as a fresh message in each new session. But, in NIBS, we need to create multiple signatures for the same receiver key $pk_R$. And, the issue is $pk_R$ binds to a fixed value for the lifetime of the system. Thus, unlike [AKSY22], we cannot set $\delta$ as the final signed message. This is because this will violate reusability, and make the NIBS scheme only single-use. Therefore, it will not be any more advantageous than a regular two-round blind signature scheme. Basically, this suggests that we cannot simply replace the usage of a PRF within our generic template as easily! Moreover, if we keep on using the PRF to generate a fresh message from each distinct presignature computed by the signer, then the receiver will have to prove it using the NIZK which will be extremely inefficient.

In summary, this means we can no longer use $\delta$ (inside each receiver’s key), or the PRF trick to generate a fresh message from each distinct presignature computed by the signer. To this end, we propose an entirely different approach to generate the final messages from presignatures. Our insight here is that the extracted final messages need not be ‘truly random messages’, but it is enough to have (1) the messages be just uncorrelated amongst different receivers, and (2) be distinct for any two distinct valid presignatures for the same receiver.

Our idea is to set the final message as $A \cdot \begin{bmatrix} x^\perp \\ z^\perp \end{bmatrix}$ instead of $F_K(r)$, where $C \cdot z = pk_R$. Here we write $x^\top$ and $x^\perp$ to be the top and bottom halves of $x$ (respectively), and the same notation applies to $z$. The intuition is that, while the receiver’s public key contains the entire $x$ vector, if we set the parameters appropriately, then we can argue by the leftover hash lemma that $x^\perp$ is statistically hidden from the signer’s view. We point out that it is important in our design that the signer implicitly assigns nonce to be $z^\perp$ (with presignature $z^\top$) where $C \cdot z = pk_R$, instead of sampling $r$ as randomness and $z$ as preimage of $pk_R || r$. This ensures that NIZKs for linear relations are still sufficient as well as makes the design even more optimal. For ensuring this property from preimage sampling, we rely on the Bonsai trick [CHKP10, ABB10b] for lattice trapdoors. Using the standard Bonsai trick, one can generate a preimage $z$ with the above special property, i.e. $z^\perp$ is sampled as a random Gaussian vector with appropriate norm.

The above serves as the core skeleton of our main construction in Section 6 however, as we discuss next, we need to make a few more slight modifications to ensure that we can prove desired unforgeability and blindness security properties for our NIBS scheme.

2.5 Security and the randomized OM-ISIS assumption

Since the lattice-based core of our optimized NIBS scheme shares many similarities with interactive blind signature scheme of [AKSY22], a natural first attempt is to prove unforgeability of our NIBS scheme using the same proof strategy as used in their work. Now, to prove security of their round-optimal blind signature scheme, Agrawal et al. [AKSY22] proposed a new ISIS-like assumption,
called the one-more ISIS assumption (OM-ISIS)\(^6\). Below we briefly recall the assumption\(^7\).

1. The challenger samples a challenge matrix \(A \in \mathbb{Z}_q^{n \times m}\) along with a large set of random target vectors \(T \subset \mathbb{Z}_q^n\). It provides the attacker with \(A\) and \(T\).

2. \(\mathcal{A}\) can make preimage queries for any target vector \(\hat{t} \in \mathbb{Z}_q^n\) to which the challenger replies with a short\(^8\) vector \(\hat{x}\) such that \(A \cdot \hat{x} = \hat{t}\).

3. OM-ISIS assumption says that \(\mathcal{A}\), having made at most \(\ell\) preimage queries, cannot output \(\ell + 1\) distinct vector pairs \(\{(x_j, t_j)\}_{j \in [\ell+1]}\) such that \(A \cdot x_j = t_j\), \(t_j \in T\), \(x_j\) is sufficiently short.

Intuitively, the assumption says that an adversary cannot find good (i.e., reasonably short) preimages for a set of \(\ell + 1\) randomly selected vectors, even when it has access to a preimage sampling algorithm that can generate at most \(\ell\) preimages for adversarially selected target vectors.

Agrawal et al. also performed some preliminary cryptanalysis of their assumption. While they were able to design practical attacks [AKSY22, § 4.5] for certain parameter settings, they suggested that, by carefully selecting the parameters in the above assumption, all known attacks fail. Moreover, they proved that their two-round blind signature scheme can be proven secure under the above assumption with those parameters.

However, the presence of efficient practical attacks on a wide range of parameters on the OM-ISIS assumption suggests that this assumption is not very stable and robust. On a technical level, the assumption appears quite strong since an attacker can ask for short preimages for "any" target vector of its choice. In more detail, an attacker can simply ask for multiple short preimages for the all-zeros vector. Given such preimage vectors, an attacker can combine them to create an approximate lattice trapdoor. As Agrawal et al. discussed, the quality of such a trapdoor computed is worse than the actual trapdoor, thus such a trapdoor would be useless if the \(\ell+1\) preimage vectors that the attacker must compute have to be as short as the preimage vectors it received.

Unfortunately, setting the parameters carefully sidesteps just one limitation of the OM-ISIS assumption, but does not make the assumption truly robust. Simply put, we believe that providing an unrestricted preimage query access to the attacker is too strong. To further illustrate this, consider an even simpler attack that finds short preimage vectors without creating a lattice trapdoor. The attacker just makes two preimage queries (which correspond to preimage queries) — one on vector 0 and other on any non-zero vector \(t \in T\) (\(T\) is the target set). Let \(z_1\) and \(z_2\) be the respective preimage vectors. That is, \(A \cdot z_1 = 0\) and \(A \cdot z_2 = t\). Given \(z_1\) and \(z_2\), an attacker simply output three distinct vectors \(z_2\), \(z_2 - z_1\), and \(z_2 + z_1\) as the preimages for the same target vector \(t \in T\).

Abstractly, the issue is that an adversary can perform simple linear combinations on preimage vectors, and such linear combinations map to appropriate linear combination of the corresponding target vectors. While this does not qualify as an attack on the two-round blind signature scheme of Agrawal et al. [AKSY22] (since any admissible attacker in their system must be generating

---

\(^6\)The “one-more” name inspired from one-more-RSA assumption [BNPS03].

\(^7\)For ease of exposition, we present a simplified assumption here. In the original assumption, the set of target vectors \(T\) can be chosen adaptively by the challenger.

\(^8\)As in typical lattice settings, by short vectors we mean vectors with small norms.
preimages on $\ell + 1$ distinct vectors), this highlights that the OM-ISIS assumption is susceptible to arbitrary linear combination attacks.

Existence of such linear combination attack strategies are a big barrier to designing reusable non-interactive blind signatures. Our initial attempt to build NIBS as a generalization of the Agrawal et al. scheme turns out to be insecure. The attack is basically the same as the one described above. To ensure reusability, we set the final message as $A \cdot x \perp z \perp$ rather than just $\delta$ (where $\delta$ is was used in the receiver’s public key). An attacker simply makes one presignature query for some receiver public key $pk_R$, and one presignature query for the all-zeros vector (as the public key), and combines them linearly to obtain $> 2$ valid presignatures for $pk_R$. One can easily show that the resulting final messages for all these preimage vectors will also be different, thereby constituting an efficient attack on one-more unforgeability of our basic NIBS scheme.

Randomized OM-ISIS assumption. To thwart the aforementioned linear-combination-style attacks on the OM-ISIS assumption, we propose a new variant that we call randomized one-more ISIS assumption $rOM-ISIS$. Our goal here is twofold — (a) we want to turn OM-ISIS assumption into a more robust assumption such that there do not exist any efficient/practical attacks (irrespective of how the parameters are set), (b) we can design a non-interactive (as well as two-round) blind signature scheme which can be proven under the new assumption.

Our strategy is to prevent an attacker from learning preimages on arbitrary target vectors of its choice. This was the central property that was exploited by Agrawal et al. [AKSY22] in their practical attacks/cryptanalytic efforts. To this end, we make the challenger “re-randomize” each target vector (independently for each query) before computing its preimage. In turn, this takes away the attacker’s prior advantage from obtaining distinct short preimages for the same target vector (or any target of its choice more generally). Moreover, by carefully selecting how the per-query re-randomization happens, we can also avoid all known affine attacks that we discussed. Below we summarize our new randomized one-more ISIS assumption $rOM-ISIS$.

1. The challenger samples a challenge matrix $A \in \mathbb{Z}_q^{n \times m}$ and a randomization matrix $B \in \mathbb{Z}_q^{n \times m}$ along with a large set of random target vectors $T \subset \mathbb{Z}_q^n$. It provides the attacker with $A$, $B$ and the vector set $T$.

2. $A$ can make preimage queries for any target vector $t \in \mathbb{Z}_q^n$ such that the challenger replies with a short vector $\hat{x}$ and a $\pm 1$ vector $\hat{y} \in \{\pm 1\}^m$ such that $A \cdot \hat{x} + B \cdot \hat{y} = t$.

3. $rOM-ISIS$ assumption says that $A$ cannot output $\ell+1$ distinct vector tuples $\{(x_j, y_j, t_j)\}_{j \in [\ell+1]}$ such that $A \cdot x_j + B \cdot y_j = t_j$, $t_j \in T$, $x_j$ is sufficiently short, $y_j$ is a $\pm 1$ vector, and $A$ made at most $\ell$ preimage queries.

Intuitively, the attacker now cannot truly select the preimage vector arbitrarily since the challenger randomizes the actual target vector as $(t - B \cdot y)$, where $y$ is a random $\pm 1$ vector. Since the attacker receives the vector $\hat{y}$ used for randomization, it is unclear whether we can reduce it to the standard ISIS assumption.\footnote{Interestingly, one can easily show by a simple application of the leftover hash lemma [HILL99, DRS04, DORS08], that if the attacker does not receive $\hat{y}$, then this is as hard as the standard ISIS assumption.} However, our preliminary cryptanalysis (cf. § 4.1) shows that it is
more robust when compared with the OM-ISIS assumption. We believe that this new formulation could serve as a better lattice analogue of the one-more RSA assumption [BNPS03]. For example, we can also prove that a mild adaptation of the Agrawal et al. [AKSY22] two-round blind signature scheme is still secure under rOM-ISIS assumption, and now we do not have to set the parameters as carefully to avoid simple attacks. Later, in Section 4, we describe the assumption in full detail and also provide some preliminary cryptanalysis.

Remark 2.2. Concurrently to our development of the rOM-ISIS assumption, Bootle et al. [BLNS23b] proposed the ISIS$_f$ assumption. Under this assumption, the adversary is given access to a preimage oracle that outputs a randomly chosen $\hat{y} \in D$ ($D$ some domain) and a short vector $\hat{x} \in \mathbb{Z}_q^n$ such that for (public) matrix $A \in \mathbb{Z}_q^{n \times m}$ sampled from some distribution and (public) function $f : D \rightarrow \mathbb{Z}_q^n$, $A \cdot \hat{x} = f(\hat{y})$. The assumption then requires that it is hard for an adversary to output a value $y \in D$ and a short vector $x$ ($\neq \hat{x}$) satisfying $A \cdot x = f(y)$. Importantly, this assumption is dependent on the choice of $f$. For instance, if $f$ is the linear map $y \mapsto B \cdot y$ for $B \in \mathbb{Z}_q^{n \times m}$ and $y \leftrightarrow \mathbb{Z}_q^m$, then the ISIS$_f$ is trivially broken by linearly combining the query responses. [BLNS23b] also define an interactive version (that is reducible to the non-interactive ISIS$_f$) where the adversary is allowed to query for preimages under specific targets $\tilde{t} \in \mathbb{Z}_q^l$ and the oracle outputs $(\tilde{y}, \tilde{x})$ such that $A \cdot \tilde{x} = f(\tilde{y}) + C \cdot \tilde{t}$ for matrix $C \in \mathbb{Z}_q^{n \times l}$ of the challenger’s choice. Under this characterization, one might hope to abstractly view the rOM-ISIS assumption as a specialization of interactive-ISIS$_f$ where the function $f$ linearly maps $\tilde{y} \in \{\pm 1\}^m$ to $-B \cdot \tilde{y}$. However, there is a non-negligible probability that outputs under this map can be efficiently linearly combined to give an arbitrary valid solution, so the ISIS$_f$ problem is actually not hard for this function. On the other hand, the rOM-ISIS assumption remains hard as the adversary is also restricted to provide (one-more) forgeries on a fixed set $T$ of target vectors.

Our final NIBS construction. With the above strengthening of the one-more ISIS assumption, we make some slight changes to our core design. Each signer additionally samples a random $\pm 1$ vector $y$, and computes the preimage for the syndrome for $(pk_R - B \cdot y)$, instead of just $pk_R$, i.e., it samples $z$ such that $C \cdot z = pk_R - B \cdot y$. The signer then explicitly sets $z$ as the presignature and $y$ as the nonce. Given this, the receiver creates a NIZK proof $\pi$ stating that, given $A, B, C, w, \delta$, there exist vectors $x, y, z$ such that the following relation holds:

$$C \cdot z + B \cdot y = A \cdot x + H(\delta) \land w = A \cdot \begin{bmatrix} x_1 \\ \vdots \\ z_\perp \end{bmatrix} \land$$

$$y = [y_1 y_2 \cdots]^T, \forall i : y_i \in \{\pm 1\} \land \|x\|, \|z\| \text{ are short.}$$

We describe our NIBS construction in detail in Section 6.1. We prove one-more unforgeability of our NIBS protocol under the rOM-ISIS assumption. Here, we run into a slight technical issue. Namely, when proving one-more unforgeability, the reduction will need to extract the adversary’s $\ell + 1$ forgeries from the NIZK proof, but rewinding $\ell + 1$ times results in an exponential (in $\ell$) soundness loss. Fortunately, one can easily apply the so-called “encryption-to-the-sky” trick [AKSY22, BLNS23a] to get straight-line extraction. More concretely, we modify the proof system to also require a proof of encryption to the witness $(x, y, z)$. Therefore the precise relation that the receiver must prove
also includes $ct = \text{PKE.Enc}(\text{pke.pk}, x||y||z; r)$, where the ciphertext $ct$ and the PKE public key $\text{pke.pk}$ also form part of the instance, and the encryption randomness $r$ is included in the witness. As done in prior works, this can be efficiently proved using a linear relation. Finally, the receiver sets $(w, \delta)$ as its message and $(\pi, ct)$ as the corresponding signature. To verify a signature, one simply runs the verification algorithm for the NIZK.

In the security proof, the reduction now retains access to the PKE secret key. This allows the reduction to extract $x_i, y_i$ and $z_i$ for all of the adversary’s $\ell + 1$ forgeries. If the hash function is modeled as a random oracle, the reduction can simply answer all hash queries for $H(\delta)$ from the challenge set $T$; if it further sets the public matrix $A$ as a matrix-matrix product of the rOM-ISIS challenge matrix $C$, it can break rOM-ISIS. Observe that the matrix $A$, statistically hides the receiver’s secret $x$ (by the leftover hash lemma). In the security proof, we combine this fact with the zero-knowledge property of the NIZK proof system, and the semantic security of the encryption scheme to prove receiver blindness.

**Tagging our NIBS construction.** We can also modify the above construction to be a tagged NIBS scheme as well. Recall that in tagged NIBS, the signer and receiver jointly agree on a value that will be treated as a public part of the signed message. To add such a public value $\tau$ to each blind signature, the signer computes a short preimage $z$ such that $C \cdot z = pk_R - B \cdot y + H(\tau)$ using a secret trapdoor for $C$. It then sends $\tau$ along with the preimage $z$ and nonce $y$ to the receiver. The receiver now includes $\tau$ in the instance of the NIZK relation, and generates an appropriate NIZK proof. The one-more unforgeability of this protocol (described in Section 7) follows by a similar reduction to rOM-ISIS (in the random oracle model). Of course, here we have the additional $H(\tau)$ term, but notice that the one-more unforgeability reduction, both models the hash function to the adversary and chooses the tag $\tau$. Thus the challenger can choose $H(\tau)$ in a way that later allows it to extract a short preimage for its rOM-ISIS challenge using the (one-more) forgery. The argument for receiver blindness follows directly from receiver blindness of the NIBS counterpart.

**Nonce blindness.** As we mentioned earlier, we also provide a NIBS scheme that satisfies basic nonce blindness in section 8. Recall that nonce blindness was defined as the intra-receiver blindness property, where the goal was just to hide the ordering of issues presignatures. To make our NIBS scheme into a nonce-blind NIBS scheme, we make two important technical changes. First, the receiver’s key generation algorithm samples and commits a random bit $\theta$, so $pk_R = A \cdot \begin{bmatrix} x \\ \theta \end{bmatrix} + H(\delta)$. The second change is in Obtain, which now computes the message $w$ as $(1 - 2\theta)(z_\perp - x_\perp)$.

The intuition behind the $(1 - 2\theta)$ term is that it prevents simple affine attacks on nonce blindness that can be carried out by subtracting two messages. Crucially, $w$ does not leak any information about the nonce if $x_\perp$ is chosen such that it “smudges-out” $z_\perp$. As we mentioned earlier, this construction is secure under the basic nonce blindness definition, and we can easily upgrade it to $k$-nonce blindness by sampling $(k - 1)$ smudging vectors as well as commit $(k - 1)$ bits instead of a single $\theta$ bits. At a high level, we are using smudging along with a one-time pad style argument for proving basic nonce blindness, and this can be extended to $k$-nonce blindness by having more smudging terms. We want to remind the reader that we do not consider nonce blindness as an
important property for practical applications of NIBS, and only provide a nonce blind NIBS scheme for completeness.

2.6 NIBS from circuit-private LHE

Finally, we provide a new template for designing NIBS with optimal-sized signatures. Our construction explores a new interesting trade-off that yields optimal-sized signatures while paying in terms of higher setup and computation costs. We believe this could lead to alternate approaches to practical NIBS (and round-optimal interactive blind signatures) in the future. Our main observation here is that rather than using NIZKs to hide the receiver’s secrets, we can leverage the fact that leveled homomorphic encryption (LHE) enables arbitrary homomorphic operations on the receiver’s commitment. Specifically, the receiver commits to a PRF secret key $K$ by homomorphically encrypting it. Using this commitment, a signer issues a presignature as follows: it first homomorphically evaluates the PRF $F_K(\cdot)$ on some randomness $r$ of its choice and then homomorphically evaluates a signature on $F_K(r)$. The resulting ciphertext $\ct$ is therefore the encryption of the signature on $F_K(r)$. Under the mild assumption that the LHE is circuit-private [OPP14], the signer can simply send $\ct$ as the presignature along with an argument of knowledge that $\ct$ was evaluated using the signing key as input to the circuit, and randomness $r$ as the nonce. The receiver sets its message to $F_K(r)$ and obtains the corresponding signature by decrypting $\ct$. Notably, the final signature is just a regular signature and is thus optimal in size. Moreover, beyond optimality, this construction satisfies our stronger notion of blindness: strong receiver (resp. nonce) blindness which we discussed earlier.

At a high level, both strong-receiver/nonce blindness follow from the IND-CPA security of the encryption scheme which ensures that $\ct$ does not reveal anything about $K$, and the pseudorandomness of $F$, which ensures that two messages on different nonces 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 both parties also prove knowledge of their secret keys. That is, our proof is in the knowledge of secret key (KOSK) model [MOR01, Bol03]. As a supplementary contribution, we also prove that any NIBS scheme secure in the KOSK model is also secure in the standard model, assuming existence of NIZKs. Our LHE-based NIBS construction is described in Section 9, and the generic compiler to upgrade security in KOSK model to standard model is provided in Appendix C.

2.7 Efficiency comparisons for our NIBS schemes

As a proof concept, we provide estimates for the various parameter sizes in our scheme. Just as [AKSY22], we instantiate our construction 6.1 with FALCON [FHK+17] for signatures, [LPS10] for Regev-style encryption and [LNP22b] for the NIZK for linear relations. In table 2, we provide the public key, transcript and signature sizes for all our NIBS constructions.
3 Preliminaries

We assume the reader is familiar with the standard cryptographic notions of PRFs, commitment schemes, PKE and digital signatures. For completeness, we include them in Appendix A.

Notation. Let \( \lambda \) denote the security parameter, and PPT denote probabilistic polynomial-time. We denote the set of real numbers by \( \mathbb{R} \) and the integers by \( \mathbb{Z} \). 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 \([0, n] := \{0\} \cup [n]\).

For a vector \( x \) of even length, we write \( x_\top \) and \( x_\bot \) to be the top and bottom halves of \( x \) respectively, i.e. \( x = \begin{bmatrix} x_\top \\ x_\bot \end{bmatrix} \). Similarly, for any matrix \( A = [A_L \mid A_R] \in \mathbb{Z}^{n \times 2m} \), we denote its left and right halves as \( A_L \) and \( A_R \), respectively.

For a vector \( x \), we write its \( \ell_2 \) norm as \( \|x\|_2 \), often dropping the subscript and writing it simply as \( \|x\| \). We write the \( \ell_\infty \) norm of \( x \) as \( \|x\|_\infty \).

3.1 Lattice preliminaries

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

Discrete Gaussians. Let \( \varsigma \) be any positive real number. The Gaussian distribution \( D_\varsigma \) with parameter \( \varsigma \) is defined by the probability distribution function \( \rho_\varsigma(x) = \exp(-\pi\|x\|^2/\varsigma^2) \). For any set \( L \subset \mathbb{R}^m \), define \( \rho_\varsigma(L) = \sum_{x \in L} \rho_\varsigma(x) \). The discrete Gaussian distribution \( D_{L,\varsigma} \) over \( L \) with parameter \( \varsigma \) is defined by the probability distribution function \( \rho_{L,\varsigma}(x) = \rho_\varsigma(x)/\rho_\varsigma(L) \) for all \( x \in L \).

The following lemma (Lemma 4.4 of [MR04, GPV08]) shows that if the parameter \( \varsigma \) of a discrete Gaussian distribution is small, then any vector drawn from this distribution will be short (with high probability).

**Lemma 3.1.** Let \( m, n, q \) be positive integers with \( m > n, q \geq 2 \). Let \( A \in \mathbb{Z}_q^{n \times m} \) be a matrix of dimensions \( n \times m \), \( \varsigma = \tilde{\Omega}(n) \) and \( L = \Lambda_q^n(A) \). Then

\[
\Pr[\|x\| > \sqrt{m} \cdot \varsigma : x \leftarrow D_{L,\varsigma}] \leq \text{negl}(n).
\]

We will also require the following lemma (Lemma 4.4 of [Lyu12]) concerning the minimum-entropy of the discrete Gaussian distribution.

| Construction          | \(|pk_R|\) | \(|\text{psig}|+\) | \(|\sigma|\) | Receiver |
|-----------------------|-----------|-----------------|-----------|----------|
| Lattice-based (6.1)   | 1.6 KB    | 0.96 KB         | 68 KB     | Standard |
| Lattice-based (8.2)   | \(\lambda \cdot 1.6\) KB | \(\lambda \cdot 0.96\) KB | \(\lambda \cdot 68\) KB | Standard |
| Circuit-private LHE (9.3) | poly(\(\lambda\)) | poly(\(\lambda\)) | \(~0.5\) KB | Strong |
| General-purpose NIZKS (B.1) | ~2 KB | ~0.5 KB | poly(\(\lambda\)) | Strong |

**Table 2:** Public key, transcript and signature sizes of our constructions.

---

17
Lemma 3.2. Let $D_{\mathbb{Z}^m,\varsigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ for any $m > 1$, with variance $\varsigma$. Then, for any $\varsigma \geq \frac{3}{\sqrt{2\pi}}$ we have that $H_\infty(D_{\mathbb{Z}^m,\varsigma}) \geq m$.

Lattice trapdoors. Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have certain ‘trapdoors’ that allow efficient solutions to hard lattice problems.

Definition 3.3 ([Ajt96, GPV08]). For lattice parameters $n, m, q$ with $m \geq O(n \log q)$, a trapdoor lattice sampler consists of algorithms TrapGen and SamplePre with the following syntax:

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

- **SamplePre**($A, T_A, u, \varsigma$) → $s$: The presampling algorithm takes as input a matrix $A$, trapdoor $T_A$, a vector $u \in \mathbb{Z}^n_q$ and a parameter $\varsigma \in \mathbb{R}$ (which determines the length of the output vectors). It outputs a vector $s \in \mathbb{Z}^m_q$ such that $A \cdot s = u$ and $\|s\| \leq \sqrt{m} \cdot \varsigma$.

These algorithms must satisfy the following properties:

1. **Well-Distributedness of Matrix.** The following distributions are statistically indistinguishable:

   \[
   \{A : (A, T_A) \xleftarrow{\$} \text{TrapGen}(1^n, 1^m, q)\} \approx_s \{A : A \xleftarrow{\$} \mathbb{Z}^{n \times m}_q\}.
   \]

2. **Preimage Sampling:** For all $(A, T_A) \xleftarrow{\$} \text{TrapGen}(1^n, 1^m, q)$, if $\varsigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:

   \[
   \{s : u \xleftarrow{\$} \mathbb{Z}^n_q, s \xleftarrow{\$} \text{SamplePre}(A, T_A, u, \varsigma)\} \approx_s D_{\mathbb{Z}^m,\varsigma}.
   \]

These properties are satisfied by the gadget-based trapdoor lattice sampler of [MP12] for parameters $m$ such that $m = \Omega(n \cdot \log q)$.

Bonsai lattice trapdoors. In this work, we will rely on the bonsai trick for lattice trapdoors [ABB10a, CHKP10]. Briefly, using the standard bonsai trick, one can sample preimage vectors with a special property that a portion of the preimage vector will be sampled as a random Gaussian vector with appropriate norm, rather than from a pre-defined lattice coset. Throughout the sequel, we will routinely sample matrices of dimensions $n \times 2m$ such as $A \in \mathbb{Z}^{n \times 2m}_q$, where

\[
(A_L, T_{A_L}) \xleftarrow{\$} \text{TrapGen}(1^n, 1^m, q), \quad A_R \xleftarrow{\$} \mathbb{Z}^{n \times m}_q.
\]

To create a preimage $s \in \mathbb{Z}^{2m}_q$, for any vector $u \in \mathbb{Z}^n_q$, such that $A \cdot s = u$, we simply sample $s_\perp \xleftarrow{\$} D_{\mathbb{Z}^m,\varsigma}$, and $s_\perp \xleftarrow{\$} \text{SamplePre}(A_L, T_{A_L}, u - A_R \cdot s_\perp, \varsigma)$.

We refer to the above bonsai-based lattice trapdoors as bLT = (bLT.TrapGen, bLT.SamplePre), where the trapdoor generation and preimage sampling algorithms are defined as above. Clearly, the algorithms for bLT satisfy the standard well-distributedness property, as well as the preimage sampling property. Moreover, it satisfies the following stronger half-preimage well distribution property:
**Half-Preimage Well Distributedness.** For all \((A, T_A) \leftarrow \$\text{bLT.TrapGen}(1^n, 1^{2m}, q)\), every vector \(u \in \mathbb{Z}_q^n\), every \(\varsigma\), the following distributions are identical:

\[
\{s_\perp : s \leftarrow \$\text{SamplePre}(A, T_A, u, \varsigma)\} \equiv D_{\mathbb{Z}_m, \varsigma}.
\]

### 3.2 Hardness assumptions

**Definition 3.4 (SIS).** Let \(q, n, m, \beta\) be functions of security parameter \(\lambda\). An instance of the SIS\(_{q,n,m,\beta}\) problem is a matrix \(C \leftarrow \$\mathbb{Z}_q^{n \times m}\), and a solution to the problem is a vector \(z \in \mathbb{Z}^m\) such that \(\|z\|_2 \leq \beta\) and \(C \cdot z = 0 \pmod{q}\).

**Definition 3.5 (Inhomogeneous-SIS).** Let \(q, n, m, \beta\) be functions of security parameter \(\lambda\). An instance of the ISIS\(_{q,n,m,\beta}\) problem is a matrix \(C \leftarrow \$\mathbb{Z}_q^{n \times m}\) and a vector \(y \leftarrow \$\mathbb{Z}_q^m\), and a solution to the problem is a vector \(z \in \mathbb{Z}^m\) such that \(\|z\|_2 \leq \beta\) and \(C \cdot z = y \pmod{q}\).

For suitably chosen parameters, the ISIS problem is at least as hard as certain worst-case lattice problems [Ajt96, MR04, GPV08].

### 3.3 One-more ISIS assumption

We recall the one-more ISIS assumption (OM-ISIS), as defined by Agrawal et al. [AKSY22].

**Assumption 3.6 (One-more ISIS).** Let \(q, n, m, \varsigma, \beta\) be functions of security parameter \(\lambda\). Consider the following experiment:

1. The challenger uniformly samples a matrix \(A \in \mathbb{Z}_q^{n \times m}\), and sends \(A\) to adversary \(A\).
2. \(A\) adaptively makes queries of the following types to the challenger, in any order.
   - **Syndrome queries.** \(A\) requests for a challenge vector, to which the challenger replies with a uniformly sampled vector \(t \leftarrow \$\mathbb{Z}_q^n\). We denote the set of received vectors by \(S\).
   - **Preimage queries.** \(A\) queries a vector \(b_t \in \mathbb{Z}_q^n\), to which the challenger replies with a short vector \(b_x \in \mathbb{Z}_q^m\) such that \(A \cdot b_x = b_t\) and \(\|b_x\| \leq \varsigma \sqrt{m}\). Let \(\ell\) denote the total number of preimage queries.
3. Finally, \(A\) outputs \(\ell + 1\) pairs of the form \(\{(x_j, t_j)\}_{j \in [\ell+1]}\). And, \(A\) wins if
   \[
   \forall j \in [\ell + 1], \quad A \cdot x_j = t_j, \quad \text{and} \quad \|x_j\| \leq \beta \quad \text{and} \quad t_j \in S.
   \]

The OM-ISIS\(_{q,n,m,\varsigma,\beta}\) assumption states that for every PPT adversary \(A\), the probability that \(A\) wins is \(\text{negl}(\lambda)\).

In words, the assumption states that it is hard to find \(\ell + 1\) short preimages from a set \(S\) of syndromes, even when given up to \(\ell\) inversions of arbitrary syndromes not in \(S\), for any polynomially bounded \(\ell\). Equivalently, it is hard to forge \(\ell + 1\) GPV signatures [GPV08] given up to \(\ell\) inversions of arbitrary syndromes. However, the main reason this is not known to be reducible to standard SIS (unlike security of GPV signatures) is that \(A\) can ask preimage queries on any arbitrary vector of its choice.
3.4 Randomness extraction

The min-entropy of a random variable $X$ is defined as $H_{\infty}(X) \overset{\text{def}}{=} -\log_2(\max_x \Pr[X = x])$. Let $SD(X, Y)$ denote the statistical distance between two random variables $X$ and $Y$. Below we state the Leftover Hash Lemma (LHL) from [HILL99, DRS04, DORS08].

**Theorem 3.7.** Let $H = \{h : X \rightarrow Y\}_{h \in \mathcal{H}}$ be a universal hash family, then for any random variable $W$ taking values in $X$, the following holds

$$SD((h, h(W)), (h, U_Y)) \leq \frac{1}{2} \sqrt{2^{-H_{\infty}(W)} \cdot |Y|},$$

where $U_Y$ denotes uniform distribution over $Y$.

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

**Corollary 3.8.** Let $\ell > m \cdot n \log_q 2 + \omega(\log n)$, $q$ a prime, and $m, n$ are positive integers. Let $R$ be an $k \times m$ matrix chosen as per distribution $\mathcal{R}$, where $k = k(n)$ is polynomial in $n$ and $H_{\infty}(\mathcal{R}) = \ell$. Let $A$ and $B$ 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 $n$.

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

**Lemma 3.9 (Smudging Lemma [AJL+12, Lemma 2.1, paraphrased]).** Let $B_1, B_2$ be two polynomials over the integers and let $D = \{D(\lambda)\}_{\lambda}$ be any $B_1$-bounded distribution family. Let $U = \{U(\lambda)\}_{\lambda}$ and $U(\lambda)$ denote the uniform distribution over integers $[-B_2(\lambda), B_2(\lambda)]$. The family of distributions $D$ and $U$ is statistically indistinguishable, $D + U \approx_s U$, if there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $B_1(\lambda)/B_2(\lambda) \leq \text{negl}(\lambda)$.

3.5 Non-interactive zero knowledge (NIZK)

A NIZK proof system for language $L$ consists of the following polynomial time algorithms:

**Setup($1^\lambda$) $\rightarrow$ crs.** The setup algorithm takes as input the security parameter $\lambda$, and outputs a crs $\text{crs}$.

**Prove($\text{crs}, x, \omega$) $\rightarrow$ $\pi$.** The prover algorithm takes as input a crs, an instance $x \in L$, and a witness $\omega$. It outputs a proof $\pi$.

**Verify($\text{crs}, x, \pi$) $\rightarrow$ 0/1.** The verification algorithm takes as input a crs, an instance $x$, and a proof $\pi$. It outputs a bit to signal whether the proof is valid or not.

A proof system is complete if for every $\lambda \in \mathbb{N}$, $\text{crs} \leftarrow \$ \text{Setup}(1^\lambda)$, any instance $x \in L$ with corresponding witness $\omega$, the following holds

$$\Pr[\text{Verify(\text{crs}, x, \pi)} = 1 \ : \ \pi \leftarrow \$ \text{Prove(\text{crs}, x, \omega)}] = 1.$$
Definition 3.10 (Soundness). A proof system $(\text{Setup}, \text{Prove}, \text{Verify})$ is computationally sound if for every stateful PPT attacker $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[ \text{Verify}(crs, x, \pi) = 1 \land x \notin \mathcal{L} : (x, \pi) \leftarrow A(1^\lambda, crs) \right] \leq \text{negl}(\lambda).$$

Definition 3.11 ((Multi-theorem) Zero-knowledge). A proof system $(\text{Setup}, \text{Prove}, \text{Verify})$ is computationally zero-knowledge if there exists a stateful PPT simulator $S$ such that for every stateful PPT attacker $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[ A(\{\pi_i, b\}_i) = b \land \forall i \in [\ell], \omega_i \text{ is a valid witness for } x_i \in \mathcal{L} : b \leftarrow \{0, 1\}, \crs_0 \leftarrow \text{Setup}(1^\lambda), \crs_1 \leftarrow S(1^\lambda), \{(x_i, \omega_i)\}_{i \in [\ell]} \leftarrow A(1^\lambda, \crs_0), \forall i \in [\ell], \pi_{i,0} \leftarrow \text{Prove}(\crs_0, x_i, \omega_i) \right\} \left\{\pi_{i,1}\right\}_i \leftarrow S(\crs_0, \{x_i\}_i) \right] \leq \frac{1}{2} + \text{negl}(\lambda).$$

Definition 3.12 (Knowledge extractor). A proof system $(\text{Setup}, \text{Prove}, \text{Verify})$ has a knowledge extractor if there exists a PPT extractor $E$ such that for every stateful PPT attacker $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[ \text{Verify}(crs, x, \pi) = 1 \land (\omega = E(\tau, x, \pi) \text{ is not a valid witness for } x \in \mathcal{L}) : \tau \leftarrow \{0, 1\}^\lambda, \crs \leftarrow \text{Setup}(1^\lambda; \tau), (x, \pi) \leftarrow A(1^\lambda, crs) \right] \leq \text{negl}(\lambda),$$

where $\tau$ denotes the randomness used for running the setup algorithm, and we assume (w.l.o.g.) that $|\tau| = \lambda$.

Remark 3.13. In our security proofs below, we make regular use of straight-line extraction. This can be achieved generically using PKE by encrypting the witness under a shared public encryption key and providing the ciphertext along with the proof. Additionally, the zero-knowledge proof itself must contain a proof of correct computation of the ciphertext. Looking ahead to the security proofs, the challenger will possess a secret decryption key corresponding to the public encryption key enabling straight-line extraction.

4 The Randomized One-more ISIS Assumption

In this work, we introduce a new variant of the OM-ISIS assumption with two goals — (i) protect from attackers that can ask for preimage queries on the same target vector more than once, and (ii) allow for re-randomization of the target vectors before answering the preimage query phase to make the assumption more robust. As discussed in [AKSY22, § 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 preimage query. Thus, an attacker can potentially request for preimage 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 $\beta$ is set appropriately (i.e., sufficiently small), then an attacker cannot break the assumption since the approximate trapdoor will not give as short preimage 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 [AKSY22] 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. [BNPS03], 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 OM-ISIS (or, rOM-ISIS for short) assumption.

Our intuition is to draw more inspiration from GPV signatures. Recall that in GPV signatures, a signature is computed as a preimage 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 RO 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 preimage vectors to create a new valid preimage vector for a fresh hash output. A little more abstractly, this means that given preimage vectors $\{t_i = A^{-1}_c(H(\mu_i))\}_i$, it is unclear how to find short coefficients $\alpha_i$ s.t. $\sum \alpha_i t_i = A^{-1}_c(H(\mu^*))$. That is, the hash function prevents from creating a valid preimage 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 generalization of the OM-ISIS assumption.

**Assumption 4.1 (randomized OM-ISIS).** Let $q, n, m, \varsigma, \beta$ be functions of security parameter $\lambda$. Consider the following experiment:

1. The challenger uniformly samples two matrices $A, B \in \mathbb{Z}^{n \times m}_q$, and sends $A, B$ to adversary $A$.
2. $A$ adaptively makes queries of the following types to the challenger, in any order.
   - **Syndrome queries.** $A$ requests for a challenge vector, to which the challenger replies with a uniformly sampled vector $t \leftarrow \mathbb{Z}^n_q$. We denote the set of received vectors by $S$.
   - **Preimage queries.** $A$ queries a vector $t \in \mathbb{Z}^n_q$, to which the challenger replies with a short vector $\hat{x} \in \mathbb{Z}^m_q$ and a $\pm 1$ vector $\hat{y} \in \{\pm 1\}^m$ such that $A \cdot \hat{x} + B \cdot \hat{y} = t$ and $\|\hat{x}\| \leq \varsigma \sqrt{m}$. Let $\ell$ denote the total number of preimage queries.
3. Finally, $A$ outputs $\ell + 1$ tuples of the form $\{(x_j, y_j, t_j)\}_{j \in [\ell + 1]}$ $A$ wins if
   \[ \forall j \in [\ell + 1], \quad A \cdot x_j + B \cdot y_j = t_j, \quad \text{and} \quad \|x_j\| \leq \beta, y_j \in \{\pm 1\}^m \text{ and } t_j \in S. \]

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

22
Intuitively, the new assumption says an attacker does not receive preimages for arbitrary target vectors \( \hat{t} \) that it selects, but for publicly re-randomized vector \( t' = \hat{t} - B \cdot y \), where \( y \) is a random \( \pm 1 \) vector (chosen by the challenger). We even provide the attacker with the vector \( y \). The point is that the ability to re-randomize the target vector, gives the challenger more flexibility in answering preimage queries. Now an attacker can win as long as it creates preimage vectors for any re-randomization of the random syndrome vectors. That is, we allow the attacker to also select any arbitrary \( \pm 1 \) vector and use it to re-randomize each syndrome vector. The only constraint is that the vectors \( y_j \) are \( \pm 1 \) vectors.

Unlike [AKSY22], 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 \( y_j \) must be \( \pm 1 \) vectors prevents the algebraic attacks that were mounted on the OM-ISIS assumption. Additionally, one could view the \( B \cdot y \) term as enforcing a structural property, on the target vectors, that a hash function enforces for GPV signatures. We provide some preliminary cryptanalysis next.

4.1 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. [AKSY22], and show that each of these attacks is in fact harder in the randomized one-more ISIS setting. First, let us recall the two types of attacks presented in [AKSY22].

4.1.1 Recalling attacks on the one-more ISIS assumption

**Lattice-based attack.** The attacker can find a solution to the one-more ISIS problem in polynomial time as follows:

1. The attacker makes \( \Theta(m^2) \) preimage queries for \( t_i = 0 \) to the OM-ISIS oracle.

2. Of the \( \Theta(m^2) \) preimages, \( m \) of them will be linearly independent with non-zero probability [Reg05] with norm \( \Theta(\sqrt{m \cdot \varsigma}) \) [BF11]. Using this fact, one can find \( m \) linearly independent vectors that span the subspace of rank \( n \). These \( m \) vectors can be used to form a basis \( E \) for \( \Lambda_+^1(C) \) such that the corresponding Gram-Schmidt basis \( \tilde{E} \) has norm \( \sqrt{m \cdot \varsigma} \).

3. From here, the attacker simply runs Babai’s Nearest Plane Algorithm on \((E, u)\) where vector \( u \in \mathbb{Z}_q \) is a general solution to the OM-ISIS challenge, \( t \). The output \( v \) of the algorithm is such that \( |u - v| \leq \frac{1}{2} \sqrt{\sum_{i \in [m]} \|\tilde{e}_i\|^2} \) which is at most \( m\varsigma \) with overwhelming probability.

Since \( v \) is a point on the lattice, it follows that \( (u - v) \) is a valid solution to OM-ISIS with norm bound \( \beta = \Theta(m\varsigma) \). A tighter bound of \( \beta = \Theta(\sqrt{mn \log q}) \) is also shown for an exponential time attack leveraging a shortest vector problem solver for the vector \( u \).

**Combinatorial attack.** The attacker can find a solution to the one-more ISIS problem in polynomial time as follows:
1. The attacker constructs a set \( A = \{ a \cdot e_i \mid \forall i \in [n], a \in \mathbb{Z}_q \} \) of size \( nq \), where \( e_i \)'s are the canonical \( n \)-dimensional basis vectors.

2. It then queries the OM-ISIS oracle for a preimage of each vector in \( A \).

3. Since any vector in \( \mathbb{Z}_n^q \) can be expressed as the sum of at most \( n \) vectors in \( A \), a preimage \( z \) for any challenge \( t \) can be found by simply adding up the preimages corresponding to the vectors in \( A \) that sum to \( t \).

Given that with all but negligible probability, the original preimages have norms bounded by \( \sqrt{m} \cdot \varsigma \) (Lemma 3.1), it follows that \( z \) is a valid solution to OM-ISIS with norm bound \( \beta = \Theta(\sqrt{nm} \cdot \varsigma) \) with overwhelming probability. In [AKSY22], a slightly more specialised version of this attack is presented for \( Q \geq nq \) preimage queries and \( \beta = \Theta(\sqrt{m} \cdot \varsigma) \).

Remark 4.2. We also observe that an even faster attack is possible at just a small cost to the norm bound. Specifically, the attacker can instead construct the set \( A = \{ 2^j \cdot e_i \mid \forall i \in [n], j \in [\lceil \log q \rceil] \} \), query each vector in the set as before. The observation is that \( t = t_1 t_2 \cdots t_n \) can be expressed as \( \sum_{i \in [Q]} e_i \cdot \sum_{j \in J_i} 2^j \), where each \( J_i \subseteq [\lceil \log q \rceil] \) is the set of indices where the binary decomposition of \( t_i \) is 1. Thus, by taking the sum of at most \( n \lceil \log q \rceil \) preimages of vectors in \( A \) (and making just as many queries), an attacker can construct a valid solution to OM-ISIS, with norm bound \( \beta = \Theta(\sqrt{nm} \log q \cdot \varsigma) \) with overwhelming probability.

4.1.2 Attacking the rOM-ISIS assumption

We now consider variations of these attack strategies against our randomized one-more ISIS assumption.

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 \( \Lambda_{q}^\perp(C) \).

1. Following the same first step as the lattice attack above, suppose the attacker makes \( Q \) preimage queries for \( t_i = 0 \) to the rOM-ISIS oracle and receives \( \{(z_i, y_i)\}_{i \in [Q]} \) such that for each \( i \in [Q] \), \( C \cdot z_i + B \cdot y_i = 0 \), \( z_i \) is short and \( y_i \in \{\pm 1\}^m \).

Here, we make the following observation:

Observation 4.3. If there exist \( a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q \) such that \( B \cdot \sum_{i \in [Q]} (a_i \cdot y_i) = 0 \), then \( \alpha := \sum_{i \in [Q]} (a_i \cdot z_i) \) is a vector on the lattice \( \Lambda_{q}^\perp(C) \). Further, for \( \gamma \) such that \( |a_i| \leq \gamma \) for all \( i \in [Q] \) we have

(i) \( \alpha \) has norm bounded above by \( Q\gamma \sqrt{m} \cdot \varsigma \) with all but negligible probability; and

(ii) \( \left\| \sum_{i \in [Q]} (a_i \cdot y_i) \right\|_2 \leq Q\gamma \sqrt{m} = \|\alpha\|_2/\varsigma \)
At this stage, we identify two types of potential attackers:

- **Type 1:** This type of attacker is able to find a collection \( a_1, a_2, \ldots, a_Q \in \mathbb{Z}_q \) as defined in Observation 4.3 such that \( |a_i| \leq \gamma \) for all \( a_i \). It can use this to construct a short basis for \( \Lambda_q^\perp(C) \) and proceed as in step 3 of the lattice attack above.

- **Type 2:** This type of 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.

**Claim 4.4.** Assuming the hardness of \( \text{SIS}_{q,n,m,\beta'} \) for \( \beta' = Q\sqrt{m} \), a type 1 attacker exists with negligible probability.

**Proof.** This follows by recalling that \( B \in \mathbb{Z}_q^{n \times m} \) is a random matrix, and observing that \( \sum_{i \in [Q]} (a_i \cdot y_i) \) is a solution to \( \text{SIS}_{q,n,m,\beta'} \) where \( \beta' = \|\alpha\|/\varsigma = Q\sqrt{m} \).

Essentially, to match the norm bound on the lattice basis as in [AKSY22] we require \( \|\alpha\| = \sqrt{m} \cdot \varsigma \), which suggests that finding even one such collection of \( a_i \)'s is at least as hard as solving \( \text{SIS}_{q,n,m,\beta'} \) for \( \beta' = \Theta(\sqrt{n \log q}) \). Next, while it is evident that a type 2 attacker cannot proceed according to the lattice-based attack strategy of [AKSY22], 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 preimage queries. However, any such combination that does not satisfy the type 1 condition will also result in a larger norm on the solutions. We leave further cryptanalysis and discovery of an alternative lattice-based attack strategy as an interesting open problem.

**A more generalized attack.** We can generalize this attack a little further by slightly modifying the first step:

1. Suppose the attacker makes \( Q \) preimage queries for any \( t_i \in \mathbb{Z}_q^m \) to the rOM-ISIS oracle and receives \( \{(z_i, y_i)\}_{i \in [Q]} \) such that for each \( i \in [Q] \), \( C \cdot z_i + B \cdot y_i = t_i \).

Now, one strategy could be to guess \( y_i \) and set the query \( t_i = B \cdot y_i \), 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 \( B \cdot \sum_{i \in [Q]} (a_i \cdot y_i) = \sum_{i \in [Q]} a_i \cdot t_i \) as in the previous case, then too \( \alpha := \sum_{i \in [Q]} (a_i \cdot z_i) \) is a vector on the lattice \( \Lambda_q^\perp(C) \). Indeed, a similar analysis shows that the attack makes \( Q \) queries, solves at least \( m \) instances of \( \text{SIS}_{q,n,m,\beta'} \) for \( \beta' = \|\alpha\|/\varsigma \), and outputs a solution to rOM-ISIS for \( \beta = \Theta(\sqrt{m} \cdot \|\alpha\|) \). 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 notice that extending the previous strategy of [AKSY22] — i.e., solving for a preimage of the challenge by adding the preimages corresponding to the scaled canonical basis that sum to the challenge — no longer works. This is because every query response to a member of the set \( A \) now additionally contains a uniformly chosen vector \( y_i \in \{\pm 1\}^m \). Instead, consider an attacker that does the following:


1. The attacker constructs the set 

\[ A = \{ a \cdot e_i + B \cdot (y'_j - 1) \mid \forall i \in [n], \forall j \in [Q], a \in \mathbb{Z}_q \} \]

where \( e_i \)'s are the canonical \( n \)-dimensional basis vectors, and each \( y'_j \in \{ \pm 1 \}^m \).

2. It then queries the rOM-ISIS oracle for a preimage of each vector in \( A \).

3. Consider any challenge vector \( t \in \mathbb{Z}_q^n \). The attacker finds a collection of \( n \) vectors of the form 

\[ a_i \cdot e_i \] such that they sum to \( n \cdot t \) where each \( a_i \in \mathbb{Z}_q \).

4. Let \( (z_{i,j}, y_{i,j}) \) be the oracle response for query \( a_i \cdot e_i + B \cdot (y'_j - 1) \). For each \( i \in [n] \), the next step is to find any \( j \in Q \) such that \( y'_j = y_{i,j} \). 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 \( Q > 2^m \).

5. Let set \( I \) contain all such pairs \( (i,j) \), and set \( J \) contain the corresponding \( j \). Then, the following holds

\[ C \cdot \sum_{(i,j) \in I} z_{i,j} + B \cdot \sum_{(i,j) \in I} y_{i,j} = \sum_{i\in[n]} a_i e_i + B \cdot \sum_{j\in J} (y'_j - 1). \]

Which upon simplifying gives 

\[ C \cdot \left( \sum_{(i,j) \in I} z_{i,j} \right) / n + B \cdot 1 = t. \]

Recall that with all but negligible probability the original preimages \( z_{i,j} \) have norms bounded by \( \sqrt{m} \cdot \varsigma \) (Lemma 3.1). Thus, \( \left( \sum_{(i,j) \in I} z_{i,j} \right) / n, 1 \) is a valid rOM-ISIS solution for \( \beta = \Theta \left( \sqrt{m/n} \cdot \varsigma \right) \) 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 \( Q = o(2^m) \).

Summary. Our above preliminary cryptanalysis suggests that our new assumption is relatively more robust than the OM-ISIS assumption against the same class of attacks. More precisely,

- For the lattice-based attack strategy, we are able to show a partial reduction to SIS (under certain conditions).

- For the combinatorial attack strategy, we give an exponential time algorithm for \( \beta = \Theta \left( \sqrt{m/n} \cdot \varsigma \right) \). In contrast, the OM-ISIS assumption with has (i) a \( \Omega(nq) \) time algorithm for \( \beta = \Theta \left( \sqrt{m/n} \cdot \varsigma \right) \); and (ii) a \( \Omega(n \log q) \) time algorithm for \( \beta = \Theta(\sqrt{m \log q} \cdot \varsigma) \).

We leave the challenges of further cryptanalysis and formally proving security of the rOM-ISIS assumption as potential future directions.
5 Non-Interactive Blind Signatures

We start by recalling the syntax of non-interactive blind signatures (NIBS) [Han23]. In a NIBS system, a signer issues a random presignature $\text{psig}$ for any receiver $R$ with public key $p_{k_R}$, such that the receiver $R$ can extract a blind signature $\sigma$ for a random message $\mu$ using its secret key $s_{k_R}$. Syntactically, a non-interactive blind signature scheme consists of the following polynomial-time algorithms:

Setup$(1^\lambda) \rightarrow pp$. On input the security parameter $\lambda$, the global setup algorithm outputs a set of public parameters $pp$. All the remaining algorithms take $pp$ as an input, but for notational clarity, we usually omit it as an explicit input.

KeyGen$_S(pp) \rightarrow (s_{k}, v_{k})$. This corresponds to the signer’s key generation algorithm. On input $pp$, it samples a public-secret key pair $(sk, vk)$.

KeyGen$_R(pp) \rightarrow (s_{k_{R}}, p_{k_{R}})$. This corresponds to the receiver’s key generation algorithm. On input $pp$, it samples a public-secret key pair $(sk_{R}, pk_{R})$.

Issue$(sk, pk_{R}) \rightarrow (\text{psig}, \text{nonce})$. This is a randomized algorithm that is run by the signer. It takes as input the signer’s secret key $sk$ as well as a receiver’s public key $pk_{R}$. It then outputs a presignature $\text{psig}$ along with nonce which represents (a portion of) the signer’s random coins.

Obtain$(sk_{R}, vk, (\text{psig}, \text{nonce})) \rightarrow (\mu, \sigma)$. This algorithm corresponds to the receiver’s blind signature extraction algorithm. Given the receiver’s secret key $sk_{R}$ as an input, along with a verification key $vk$ and presignature-nonce pair $(\text{psig}, \text{nonce})$, it outputs a message-signature pair $(\mu, \sigma)$ or aborts (in which case it outputs ⊥).

Verify$(vk, \mu, \sigma) \rightarrow \{0, 1\}$. This is the signature scheme verification algorithm that takes as input a verification key and message-signature pair, and outputs 0/1 to denote whether the message-signature pair is valid or not.

Correctness. A non-interactive blind signature scheme satisfies correctness if for every security parameter $\lambda \in \mathbb{N}$, $pp \leftarrow$ Setup$(1^\lambda)$, $(sk, vk) \leftarrow$ KeyGen$_S(pp)$, $(sk_{R}, pk_{R}) \leftarrow$ KeyGen$_R(pp)$, the following holds:

$$\Pr[\text{Verify}(vk, \text{Obtain}(sk_{R}, vk, \text{Issue}(sk, pk_{R})))) = 1] = 1,$$

where the probability is taken over the random coins of Issue and Obtain.

Remark 5.1 (Comparing with [Han23], and the reusability property). The above formalization of non-interactive blind signatures is nearly identical to the syntax introduced by Hanzlik, except we do not regard nonce as an input supplied to the Issue algorithm but as an output. We essentially simplify the syntax by viewing nonce as a signer’s public (random) coins. The Obtain algorithm receives both $(\text{psig}, \text{nonce})$ (as in [Han23]).

Paraphrasing [Han23], the purpose of nonce is to ensure reusability of a receiver’s public key for obtaining multiple message-signature pairs from a single signer. We emphasize that both formulations are nearly equivalent, and moreover, our formulation helps defining an important
reusability property that is necessary to avoid vacuous solutions. Further, we do not see any advantage in defining nonce as anything other than signer’s public randomness since the goal here is to have the signing process be non-interactive, and treating nonce as an extra input is inconsistent with that goal.

Equivalence of both formalizations. It is straightforward to see that one can alternatively make nonce as an explicit parameter as well in the above syntax. Moreover, a NIBS scheme satisfying the above syntax can be generically translated into satisfying the syntax from [Han23]. The idea is to generate the randomness for our Issue algorithm by evaluating a PRF on the nonce value provided as input in Hanzlik’s version

A vacuous NIBS scheme. Consider a NIBS scheme where the Issue algorithm is deterministic and it always outputs the same nonce value (or, following Hanzlik’s notation, the Issue algorithm ignores the nonce value entirely). Such a NIBS scheme clearly does not satisfy any meaningful notion of reusability, since for each receiver’s public key a signer generates at most one presignature. Unfortunately, this is still a valid NIBS scheme as per existing definitions, and furthermore, it satisfies the nonce blindness property [Han23, Definition 17] vacuously, i.e. even if the adversary outputs \((\text{psig}_0, \text{nonce}_0) \neq (\text{psig}_1, \text{nonce}_1)\) it would still not be able to distinguish for \((\mu_b, \sigma_b)\). The main issue is that the existing definitions do not disallow schemes where the Issue and Obtain algorithms ignore the nonce parameter.

A simple and sound approach to capture reusability. Our proposal is to simply define reusability of NIBS schemes directly. Rather than making the nonce parameter explicit, we define reusability of a NIBS scheme as the property that any receiver can obtain two distinct messages (along with valid signatures) for two randomly generated presignature-nonce pairs (for the same receiver’s public key) with all but negligible probability. Formally:

**Definition 5.2 (Reusability).** A NIBS scheme \(\mathcal{S}\) satisfies the reusability property, if there exists a negligible function \(\negl(\cdot)\) such that for every \(\lambda \in \mathbb{N}\), the following holds:

\[
\Pr \left[ \mu_0 = \mu_1 : \begin{array}{l}
pp \leftarrow \mathcal{S}.\text{Setup}(1^\lambda) \\
(\text{sk}, \text{vk}) \leftarrow \mathcal{S}.\text{KeyGen}_\mathcal{R}(pp), (\text{sk}_R, \text{pk}_R) \leftarrow \mathcal{S}.\text{KeyGen}_\mathcal{R}(pp) \\
\forall b \in \{0, 1\} : (\text{psig}_b, \text{nonce}_b) \leftarrow \mathcal{S}.\text{Issue}(\text{sk}, \text{pk}_R) \\
\forall b \in \{0, 1\} : (\mu_b, \sigma_b) \leftarrow \mathcal{S}.\text{Obtain}(\text{sk}_R, \text{vk}, (\text{psig}_b, \text{nonce}_b)) \end{array} \right] \leq \negl(\lambda).
\]

**Remark 5.3.** Note that we are only defining the final messages are distinct. One could additionally require the intermediate presignature-nonce pairs to be distinct, however that is implied by the one-more unforgeability property. This is due to the fact that if a receiver could create signatures for two different messages from a single presignature, then it would break one-more unforgeability.

Next, we provide the standard notion of one-more unforgeability for blind signatures, specialized for the NIBS setting [Han23].

---

10Concretely, \(\mathcal{S}.\text{Issue}^{\text{Hanzlik}}(\text{sk}, \text{pk}_R, \text{nonce})\) outputs \(\text{psig}\) and \(\text{nonce}^*\) where \((\text{psig}, \text{nonce}') \leftarrow \mathcal{S}.\text{Issue}^{\text{Ours}}(\text{sk}, \text{pk}_R, \text{F}_k(\text{nonce})))\) and \(\text{nonce}^* = (\text{nonce}, \text{nonce}')\).

11We want to remark that in [Han23], in the one-more unforgeability security experiment, the adversary can select nonce during each Issue query. In our definition, the challenger samples nonce since it is treated as randomness of Issue. However, both definitions are equivalent since a signer can use a PRF to generate the actual randomness from an input nonce as described in Remark 5.1.
Definition 5.4 (One-more unforgeability). A NIBS scheme $S$ satisfies one-more unforgeability, if for every stateful admissible PPT adversary $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds:

$$\Pr \left[ \bigwedge_{i \in [\ell+1]} \text{Verify}(vk, \mu_i, \sigma_i) = 1 \land \left( \bigwedge_{i \neq j \in [\ell+1]} \mu_i \neq \mu_j \right) : \begin{aligned} pp &\leftarrow \text{Setup}(1^\lambda) \\
(\sk, vk) &\leftarrow \text{KeyGen}_S(pp) \\
(\mu_i, \sigma_i)_{i=1}^{\ell+1} &\leftarrow A^{O_{sk}(\cdot)}(vk) \end{aligned} \right] \leq \text{negl}(\lambda),$$

where $O_{sk}(\cdot)$ takes as input a receiver’s public key $pk_{R_i}$, and outputs a presignature-nonce pair $(\text{psig}_i, \text{nonce}_i)$ by running $\text{Issue}(\sk, pk_{R_i})$, and $A$ is an admissible adversary iff $A$ makes at most $\ell$ queries to $O_{sk}$.

We now provide the receiver blindness property for NIBS [Han23]. Recall that receiver blindness captures the fact that a malicious signer cannot distinguish who was the receiver of a final signature it sees. This is formalized via an indistinguishability experiment, where the adversary is given as input two distinct receiver keys, outputs two pairs of presignatures and nonces and then is asked to distinguish between the final two signatures that were honestly obtained by its output pairs. One can think of the receiver blindness property as an inter-receiver blindness property. We give our definition in the malicious key model, i.e. the adversary only outputs $vk$. As in [Han23], one could also define a weaker version of blindness by requiring the adversary to also output $\sk$ (honest key model).

Definition 5.5 (Receiver blindness). A NIBS scheme $S$ satisfies receiver blindness, if for every stateful admissible PPT adversary $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds:

$$\Pr \left[ A(\mu_\hat{b}, \sigma_\hat{b}, \mu_{1-\hat{b}}, \sigma_{1-\hat{b}}) = \hat{b} : \begin{aligned} pp &\leftarrow \text{Setup}(1^\lambda), \ \hat{b} \leftarrow \{0, 1\}, \\
\forall b \in \{0, 1\} : (\sk_{R_b}, pk_{R_b}) &\leftarrow \text{KeyGen}_R(pp) \\
(vk, (\text{psig}_b, \text{nonce}_{b})) &\leftarrow A(pk_{R_b}, pk_{R_{1-b}}) \\
\forall b \in \{0, 1\} : (\mu_b, \sigma_b) &\leftarrow \text{Obtain}(\sk_{R_b}, vk, (\text{psig}_b, \text{nonce}_b)) \end{aligned} \right] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where $A$ is an admissible adversary iff $\sigma_0, \sigma_1 \neq \bot$ (i.e., Obtain algorithm does not abort).

In this work, we also propose stronger notions of inter/intra-receiver blindness for NIBS schemes. The existing approaches to capture blindness for NIBS do not allow an adversary to learn any correlation between presignature-nonce pairs and their corresponding blind signature-message pairs. Unfortunately, if a server learns the receiver’s identity for just one blind signature, then existing definitions are insufficient in providing any notion of blindness from such attacks. In order to protect from such advanced attackers that can bypass the blindness property for receivers on some selected blind signatures, we introduce stronger notions of inter/intra-receiver blindness properties that we refer to as strong receiver/(nonce) blindness. We describe strong receiver blindness below.
Definition 5.6 (Strong receiver blindness). A NIBS scheme $S$ satisfies strong receiver blindness, if for every stateful admissible PPT adversary $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds:

$$\Pr \left[ A^{O_{sk_{R_0}}, a_{sk_{R_1}}}^{\cdot, \cdot, \cdot}(\mu_b, \sigma_b, \mu_{1-b}, \sigma_{1-b}) = \hat{b} : \right.$$  

$$\text{pp} \leftarrow \text{Setup}(1^\lambda), \ \hat{b} \leftarrow \{0, 1\}, \ \forall b \in \{0, 1\} : (sk_{R_0}, pk_{R_0}) \leftarrow \text{KeyGen}_R(\text{pp})$$

$$(vk, (psig_0, \text{nonce}_0)) \leftarrow A^{O_{sk_{R_0}}, a_{sk_{R_1}}}^{\cdot, \cdot, \cdot}(pk_{R_0}, pk_{R_1})$$

$$\forall b \in \{0, 1\} : (\mu_b, \sigma_b) \leftarrow \text{Obtain}(sk_{R_0}, vk, (psig_0, \text{nonce}_0))$$

$$\text{where oracle } O_{sk_{R_0}, a_{sk_{R_1}}}, \text{ on the } i\text{-th query } (b^{(i)}, vk^{(i)}, (psig^{(i)}, \text{nonce}^{(i)})), \text{ outputs } \text{Obtain}(sk_{R_0^{(i)}}, vk^{(i)}, (psig^{(i)}, \text{nonce}^{(i)})). \text{ That is, } O_{sk_{R_0}, a_{sk_{R_1}}} \text{ provides } A \text{ oracle access to the Obtain algorithm w.r.t. } sk_{R_0}, sk_{R_1}. \text{ We say that } A \text{ is an admissible adversary iff:}$$

- $\sigma_0, \sigma_1 \not\perp$ (i.e., Obtain algorithm does not abort), and
- $\text{nonce}_0 \neq \text{nonce}^{(i)}$ and $\text{nonce}_1 \neq \text{nonce}^{(i)}$ for all $i$. (That is, $A$ cannot make an Obtain query with nonce value to be either of the challenge nonce values.)

6 Lattice-based NIBS

We begin this section by describing our NIBS scheme that we prove secure under the rOM-ISIS assumption.

Let parameters $n, m, \varsigma, \beta = \varsigma\sqrt{m}$, a prime number $q$ be functions of the security parameter $\lambda$ such that the randomized one-more ISIS instance rOM-ISIS$_{q,n,2m,\varsigma,3\sqrt{2}\beta}$ is hard. These parameters must satisfy the following constraints:

$$n = \text{poly}(\lambda), \ m > n \log q + \lambda, \ \varsigma / m = \Omega(1), \ \beta < m\varsigma \quad (1)$$

Tools required. Our construction relies on a public key encryption scheme PKE = (KeyGen, Enc, Dec), a lattice trapdoor bLT = (TrapGen, SamplePre), and a NIZK proof system NIZK = (Setup, Prove, Verify) for the following language:

<table>
<thead>
<tr>
<th>Language $\mathcal{L}_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong> Each instance $x$ is interpreted matrices $C, A$ and $B$, PKE public key $\text{pke.pk}$, ciphertext $ct$, vector $w$ and random string $\delta$.</td>
</tr>
<tr>
<td><strong>Witness:</strong> Witness $\omega$ consists of a secret vector $x$, nonce vector $y$, presignature vector $z$ and randomness $r$.</td>
</tr>
<tr>
<td><strong>Membership:</strong> $\omega$ is a valid witness for $x$ if the following are satisfied:</td>
</tr>
<tr>
<td>- $|z| \leq \sqrt{2}\beta$ and $|x| \leq \sqrt{2}\beta / m$ and $y \in {\pm 1}^{2m}$.</td>
</tr>
<tr>
<td>- $ct$ is an encryption of $x|y|z$ using randomness $r$, s.t. $ct = \text{PKE.Enc}(\text{pke.pk}, x|y|z; r)$</td>
</tr>
<tr>
<td>- $C \cdot z + B \cdot y = A \cdot x + H(\delta)$ and $w = A_L \cdot x_{\perp} + A_R \cdot z_{\perp}$</td>
</tr>
</tbody>
</table>

30
6.1 Construction

Below we describe our non-interactive blind signature scheme.

Setup($1^\lambda, n, m, q, \varsigma, H$) $\rightarrow$ pp. It samples $A, B \leftarrow \mathbb{Z}^{n \times 2m}$. Next, it runs the key generating algorithm of PKE and generates the public and secret key pair as

$$(pke.pk, pke.sk) \leftarrow \text{PKE.KeyGen}(1^\lambda).$$

Next, it generates nizk.crs $\leftarrow$ NIzk.Setup($1^\lambda$) and outputs:

$$pp := (A, B, pke.pk, nizk.crs).$$

Note that $H$ is the hash function which we model as a random oracle.

KeyGen$_S$(pp) $\rightarrow$ (sk, vk). Runs the setup algorithm of lattice trapdoor and obtains

$$(T_C, C) \leftarrow \text{bLT.TrapGen}(1^\lambda, n, 2m, q).$$

It outputs signer's secret key and verification key as $sk := T_C$ and $vk := C$.

KeyGen$_R$(pp) $\rightarrow$ (sk$_R$, pk$_R$). It samples $x \leftarrow \mathcal{D}_{\mathbb{Z}^m_q}^{\varsigma/m}$ and $\delta \leftarrow \{0, 1\}^\lambda$. Next, it computes

$$t = A \cdot x + H(\delta).$$

It outputs user's secret key and public key as $(sk_R := (x, \delta), pk_R := t)$.

Issue(sk, pk$_R$) $\rightarrow$ (psig, nonce). The issue algorithm samples a random $\pm 1$ vector $y \leftarrow \{\pm 1\}^{2m}$. Next, using the signing key $sk = T_C$ and the receiver’s public key $pk_R = t$, it generates

$$z \leftarrow \text{bLT.SamplePre}(C, T_C, t - B \cdot y, \varsigma),$$

and outputs the presignature $psig := z$, and nonce as $nonce := y$.

Obtain(sk$_R$, vk, psig, nonce) $\rightarrow$ ($\mu$, $\sigma$). It parses $sk_R$ as $(x, \delta)$. Then, assigns $C := vk$, $z := psig$, and $y := nonce$. It checks if $C \cdot z + B \cdot y = A \cdot x + H(\delta)$, $||z|| \leq \sqrt{2}/3$ and $y \in \{\pm 1\}^{2m}$. If any check fails it aborts and outputs $\perp$.

Otherwise, it generates

$$ct \leftarrow \text{PKE.Enc}(pke.pk, x || y || z; r)$$

from uniformly sampled randomness $r \leftarrow \{0, 1\}^\lambda$. The obtain algorithm sets $w$ as $w = A \cdot \begin{bmatrix} x_\perp \\ z_\perp \end{bmatrix}$, where $x_\perp, z_\perp \in \mathbb{Z}_q^m$ and generates NIzk proof

$$\pi \leftarrow \text{NIzk.Prove}(\text{nizk.crs}, x := (C, A, B, pke.pk, ct, w, \delta), \omega := (x, y, z, r)).$$

Finally, it outputs message $\mu := (w, \delta)$ and signature $\sigma := (\pi, ct)$.

Verify(vk, $\mu$, $\sigma$) $\rightarrow$ $\{0, 1\}$. It parses $\mu$ as $(w, \delta)$ and $\sigma$ as $(\pi, ct)$. The verification algorithm accepts and outputs 1 if and only if

$$\text{NIzk.Verify}(\text{nizk.crs}, x := (C, A, B, pke.pk, ct, w, \delta), \pi) = 1.$$

Otherwise, it outputs 0.
Completeness. Observe that by correctness of bLT, \( \nu \) is a presignature on \( A \cdot x + H(\delta) - B \cdot y \), ie., \( C \cdot z = A \cdot x + H(\delta) - B \cdot y \) such that \( \|z\| \leq \sqrt{2} \beta \) and \( y_i \in \{\pm 1\} \) for all \( i \in [2m] \) where \( y_i \) is the \( i \)th element of \( y \). Since, it further holds by construction that \( w = A_L \cdot x_{\perp} + A_R \cdot z_{\perp} \), we have that \( \pi \) is a valid NIZK proof for the language \( L_1 \). It therefore follows from the completeness of the underlying NIZK that Construction 6.1 is complete.

Reusability. Notice that for \( b \in \{0, 1\} \), if a signer issues presignature-nonce pairs \((z_b, y_b)\) to a given receiver with secret vector \( x \) then,

\[
Pr[\mu_0 = \mu_1] = Pr[(w_0, \delta) = (w_1, \delta)] = Pr\left[A \cdot \begin{bmatrix} 0 \\ z_{0,\perp} - z_{1,\perp} \end{bmatrix} = 0\right].
\]

The probability that \( z_{0,\perp} = z_{1,\perp} \) is negligible in \( \lambda \), otherwise conditioned on \( z_{0,\perp} \neq z_{1,\perp} \), we have a short solution for SIS. Thus the overall probability is negligible and our construction satisfies the reusability property.

6.2 One-more unforgeability

Consider first, the one-more unforgeability of this protocol.

**Theorem 6.1.** Assume that NIZK proof system NIZK satisfies soundness, lattice trapdoor bLT satisfies well-distributedness, and the rOM-ISIS assumption holds, then our Construction 6.1 is one-more unforgeable.

**Proof.** Let us define the following hybrids:

**Hybrid_0** This is the actual one-more unforgeability game for NIBS:

1. Challenger samples \( A, B \leftarrow \mathbb{Z}_q^{n \times 2m} \) uniformly at random, \( pke.pk \leftarrow \mathsf{PKE.KeyGen}(1^\lambda), \) nizk.crs \( \leftarrow \) NIZK.Setup\((1^\lambda)\) and sets pp as \((A, B, pke.pk, nizk.crs)\). Next, it samples \((T_C, C) \leftarrow \mathsf{bLT.TrapGen}(1^\lambda, n, 2m, q)\) and sets \( sk := T_C, vk := C \). Challenger sends pp and vk to the adversary.

2. Adversary chooses \( pk_R \) and requests a presignature, to which the challenger replies with \((\text{psig, nonce}) \leftarrow \mathsf{Issue}(sk, pk_R)\). The adversary repeats this step a total of \( \ell \) times.

3. Adversary outputs \( k \) \((k = \ell + 1)\) message and signature pairs \( ((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k)) \). It wins if \( \mu_i \neq \mu_j \) for \( 1 \leq i < j \leq k \), Verify\((vk, \mu_i, \sigma_i) = 1\) for all \( i \in [k] \).

**Hybrid_1** This is the same as Hybrid_0 aside from one key difference that instead of uniformly sampling the matrix \( A \), it is chosen by first sampling a matrix \( R \leftarrow \{0, 1\}^{2m \times 2m} \) and then setting \( A = C \cdot R \).

32
1. Challenger samples $R \leftarrow \{0, 1\}^{2m \times 2m}$ uniformly at random, samples $(T_C, C) \leftarrow \text{bLT.\text{TrapGen}}(1^λ, n, 2m, q)$, and sets $A = C \cdot R$. Challenger then samples $B \leftarrow \mathbb{Z}_q^{n \times 2m}$, $\text{pke.pk} \leftarrow \text{PKE.Setup}(1^λ)$, $\text{nizk.crs} \leftarrow NIZK.\text{Setup}(1^λ)$, and sets $pp$ as $(A, B, \text{pke.pk}, \text{nizk.crs})$. Next, it sets $sk := T_C$, $vk := C$. Challenger sends $pp$ and $vk$ to the adversary.

Let $\text{Adv}_A^0$ denote the security advantage of an adversary $A$ in the one-more unforgeability game in $\text{Hybrid}_j$, then the following must hold:

**Lemma 6.2.** For all PPT adversaries $A$, $|\text{Adv}_A^0 - \text{Adv}_A^1| \leq \text{negl}(λ)$.

**Proof.** Note that $R \leftarrow \{0, 1\}^{2m \times 2m}$, and thus $H_\infty(R) = 4m^2$. As $4m^2 > 2m \cdot n \log_q q + λ$. It follows from Leftover Hash Lemma (Corollary 3.8) that the following distributions are statistically indistinguishable:

$$\{(C', C' \cdot R) : C' \leftarrow \mathbb{Z}_q^{n \times 2m}, R \leftarrow \{0, 1\}^{2m \times 2m}\} \approx_s \{(C', A) : C' \leftarrow \mathbb{Z}_q^{n \times 2m}, A \leftarrow \mathbb{Z}_q^{n \times 2m}\}.$$  

In the above equation, $C'$ is sampled from $\mathbb{Z}_q^{n \times 2m}$ uniformly at random. By Definition 3.3, the Well Distributedness property implies that the statistical distance between $C'$ and $C$ is within $2^{-Ω(λ)}$. Thus,

$$\{(C, C \cdot R) : (T_C, C) \leftarrow \text{bLT.\text{TrapGen}}(1^λ, n, 2m, q), R \leftarrow \{0, 1\}^{2m \times 2m}\} \approx_s$$

$$\{(C, A) : C \leftarrow \mathbb{Z}_q^{n \times 2m}, A \leftarrow \mathbb{Z}_q^{n \times 2m}\}.$$  

□

**Lemma 6.3.** Assuming that NIZK argument is sound and the randomized one-more ISIS assumption holds, adversary $A$ can have at most negligible advantage in $\text{Hybrid}_1$.

**Proof.** This follows from a case by case reduction algorithm. Suppose there exists a PPT attacker $A$ that wins the one-more unforgeability game with non-negligible probability $ε = \epsilon(λ)$. It outputs $k$ message and signature pairs $((μ_1, σ_1), \ldots, (μ_k, σ_k))$, and wins if and only if $μ_i \neq μ_j$ for $1 \leq i < j \leq k$, $\text{Verify}(vk, μ_i, σ_i) = 1$ for $1 \leq i \leq k$, and $k = \ell + 1$. We divide the potential attacker into the following cases:

- **Type 1:** Consider that we parse inputs and signatures $(μ_i, σ_i)$ from $A$’s output as $((w_i, δ_i), (π_i, ct_i))$ for all $i \in [k]$. For type 1 attacker $A$, there exists some index $j \in [k]$ such that $\text{NIZK.\text{Verify}}(x^* := (C, A, B, \text{pke.pk}, ct_j, w_j, δ_j), π_j) = 1$ and $x^*$ is not a valid instance for language $L_1$.

- **Type 2:** Consider $ct_i$ as encryptions of $x_i, y_i$, and $z_i$, such that $\text{PKE.Enc}(\text{pke.pk}, x_i || y_i || z_i ; r) = ct_i$ and $x_j, y_j, z_j \in \mathbb{Z}_q^{2m}$. For type 2 attacker $A$, there exists some $j \in [k]$, such that $C \cdot z_j + B \cdot y_j = A \cdot x_j + H(δ_j)$ and the attacker never queries random oracle $H$ using input $δ_j$.

- **Type 3:** For all $i \in [k]$, $x_i := (C, A, B, \text{pke.pk}, ct_i, w_i, δ_i)$ is a valid instance for language $L_1$, and it makes at least one hash query $H(δ_i)$ using input $δ_i$.

**Claim 6.4.** Assuming that NIZK satisfies soundness property, then type 1 attacker $A$ has at most negligible advantage.
Proof. Assume that type 1 attacker $A$ has non-negligible advantage $\epsilon = \epsilon(\lambda)$, we design a reduction algorithm $B$ that breaks soundness of nizk.

The NIzK soundness challenger starts by sampling and outputting nizk.crs $\leftarrow$ NIzK.Setup$(1^\lambda)$. With nizk.crs, $B$ then generates $T_C, C, A, B, pke.pk$ according to Hybrid 1, and sets pp := $(A, B, pke.pk, nizk.crs)$. It sends pp and vk to $A$. Next, $B$ responses to $A$’s $\ell$ presignature queries. $A$ then outputs $k$ message and signature pairs where there exists some $j$ such that NIzK.Verify$(nizk.crs, x^*, \pi_j) = 1$ and $x^*$ is not a valid instance for $L_1$, as explained above. If type 1 adversary $A$ has non-negligible advantage $\epsilon(\lambda)$, then $B$ univormly randomly samples $j' \leftarrow \{k\}$ and outputs $x' := (C, A, pke.pk, ct_{j'}, w_{j'}, \delta_{j'}, \pi_{j'})$. $B$ breaks NIzK soundness game with a non-negligible probability $\epsilon(\lambda)/k$.

Claim 6.5. Type 2 attacker $A$ has at most negligible advantage.

Proof. Since type 2 attacker $A$ never queries $H(\delta_j)$, value of $H(\delta_j)$ is not defined and is completely random in the view of $A$, as $H$ is used as a random oracle. Such attacker $A$ can have at most negligible advantage because it must hold that $H(\delta_j) = C \cdot z_j + B \cdot y_j - A \cdot x_j$.

Claim 6.6. Assuming that rOM-ISIS assumption holds, type 3 adversary $A$ has at most negligible advantage.

Proof. Assume that type 3 attacker $A$ has non-negligible advantage $\epsilon = \epsilon(\lambda)$, we design a reduction algorithm $B$ that breaks the rOM-ISIS assumption.

The rOM-ISIS challenger starts by sampling and outputting a matrix $C, B \in \mathbb{Z}_{q}^{n \times 2m}$ uniformly at random. Reduction algorithm $B$ then samples $R \leftarrow \{0, 1\}^{2m \times 2m}$ and sets $A$ as $A = C \cdot R$. Next, $B$ generates $(pke.pk, pke.sk)$ and nizk.crs. $B$ outputs pp := $(A, B, pke.pk, nizk.crs)$ and verification key vk := $C$.

Next, $A$ is allowed to make a series of hash queries and presignature queries (in any order). On each fresh hash query that $A$ provides an input $\delta$, $B$ simply makes a syndrome query to the challenger. $B$ forwards challenger’s output to $A$ as the output of $H(\delta)$. Meanwhile, $B$ keeps a recording of all hash queries from $A$. For each presignature query on input $t'$ from $A$, $B$ makes a preimage query to the challenger using the same input $t'$. Then, $B$ forwards challenger’s output $z'$ and $y'$ to $A$ such that $C \cdot z' + B \cdot y' = t'$. $A$ makes a total of $\ell$ such presignature queries. To win the one-more unforgeability game, $A$ outputs $k$ message and signature pairs $((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k))$ such that $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$, $\text{Verify}(pp, vk, \mu_i, \sigma_i) = 1$ for all $i \in [k]$, and $k = \ell + 1$.

For all $i \in [k]$, $B$ parses $\mu_i$ as $(w_i, \delta_i)$ and $\sigma_i$ as $(\pi_i, ct_i)$. $B$ obtains $x_i \in \mathbb{Z}_{q}^{2m}$, $y_i \in \{\pm 1\}^{2m}$, and $z_i \in \mathbb{Z}_{q}^{2m}$ by decrypting $ct_i$, where $x_i || y_i || z_i$ = PKE.Dec$(pke.sk, ct_i)$. Thus for all $i \in [k]$,

$$H(\delta_i) = C \cdot z_i + B \cdot y_i - A \cdot x_i = C \cdot z_i + B \cdot y_i - C \cdot R \cdot x_i = C \cdot (z_i - R \cdot x_i) + B \cdot y_i.$$  

Recall that $\|z_i\| \leq \sqrt{2\beta}$, $\|x_i\| \leq \sqrt{2\beta}/m$ and since $R \in \{0, 1\}^{2m \times 2m}$ is a binary matrix,

$$\|z_i - R \cdot x_i\| \leq 3\sqrt{2\beta}.$$  

Note that $\mu_i \neq \mu_j$, for $1 \leq i < j \leq k$. Then there are two cases:
Case 1: There exist \(1 \leq i < j \leq k\), \(\delta_i = \delta_j\) and \(y_i = y_j\). Since \(H(\delta_i) = C \cdot z_i + B \cdot y_i - A \cdot x_i\) we have:

\[
C \cdot (z_i - z_j) = A \cdot (x_i - x_j)
\]

Further, for \(A = C \cdot R\), this equivalently means

\[
z_i - z_j = R \cdot (x_i - x_j)
\]

We show that this happens with negligible probability.

**Claim 6.7.** The probability \(\Pr [z_i - z_j = R \cdot (x_i - x_j)]\), taken over the adversary’s output distribution and the randomness of \(R\), is negligible.

**Proof.** We have by the leftover hash lemma, that \((C, A)\) is statistically close to \((C, C \cdot R)\). As a result, the distribution of the adversary’s outputs \((x_i, z_i, x_j, z_j)\) where the reduction algorithm \(B\) generates \(A\) uniformly at random is statistically close to the distribution of the adversary’s output when reduction algorithm \(B\) generates \(A := C \cdot R\). Thus, with all but negligible probability, the adversary’s output is statistically independent from the choice of \(R\). So it follows that with all but negligible probability, \(z_i - R \cdot x_i \neq z_j - R \cdot x_j\).

Case 2: \(\delta_i \neq \delta_j\). Then with all but negligible probability, we have \(H(\delta_i) \neq H(\delta_j)\), which implies that \(C \cdot (z_i - R \cdot x_i) + B \cdot y_i \neq C \cdot (z_j - R \cdot x_j) + B \cdot y_j\). Thus, either \(z_i - R \cdot x_i \neq z_j - R \cdot x_j\), or \(y_i \neq y_j\).

Case 3: \(\delta_i = \delta_j\) and \(y_i \neq y_j\). Since \(y_i \neq y_j\), the analysis directly follows in this case.

As a result, \(B\) wins the rOM-ISIS game by sending back \(z_i - R \cdot x_i\), \(y_i\), and \(H(\delta_i)\) for \(i \in [k]\).

Note that any successful one-more unforgeability attacker \(A\) must be one of type 1, 2 and 3. Combining the above arguments, the lemma follows.

This completes the proof of our main theorem.

6.3 Receiver blindness

Next, we show that this protocol satisfies receiver blindness property.

**Theorem 6.8.** Assume that NIZK proof system NIZK satisfies zero knowledge property, and public key encryption scheme PKE is IND-CPA secure, then our construction 6.1 is receiver blind.

**Proof.** We prove the theorem through the following hybrids:
Hybrid$_0$ This corresponds to the real experiment.

1. Challenger samples $A, B \leftarrow Z_{q}^{n \times 2m}$ uniformly at random, $\text{pke.pk} \leftarrow \text{PKE.KeyGen}(1^\lambda)$, and $\text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$. Challenger then sets $\text{pp} := (A, B, \text{pke.pk}, \text{nizk.crs}, H)$ and sends $\text{pp}$ to the adversary.

2. Next, challenger samples $(\text{sk}_{R_0}, \text{pk}_{R_0}) \leftarrow \text{KeyGen}_R(\text{pp})$, for $b \in \{0, 1\}$. The challenger sends $\text{pk}_{R_0}$ and $\text{pk}_{R_1}$ to $A$.

3. The adversary outputs a matrix $C \in Z_{q}^{n \times 2m}$ as the verification key $vk$, and issues two presignatures and nonces $(\text{psig}_0, \text{nonce}_0)$ and $(\text{psig}_1, \text{nonce}_1)$.

4. The challenger then generates message and signature pairs:

   (a) For $b \in \{0, 1\}$, challenger sets $z_b$ as $\text{psig}_b$, $y_b$ as $\text{nonce}_b$, and parses $\text{sk}_{R_b}$ as $(x_b, \delta_b)$. It then checks whether $C \cdot z_b + B \cdot y_b = A \cdot x_b + H(\delta_b)$ and $\|z_b\| \leq \sqrt{2\beta}$ and $y_b \in \{\pm 1\}^{2m}$. If any of the checks fails, challenger outputs ⊥ and aborts.

   (b) Otherwise, for $b \in \{0, 1\}$, challenger continues to generate $ct_b \leftarrow \text{PKE.Enc}(\text{pke.pk}, x_b || y_b || z_b; r_b)$ with freshly sampled randomness $r_0$ and $r_1$.

   (c) For $b \in \{0, 1\}$, it computes $w_b = A \cdot \begin{bmatrix} x_b, \perp \\ z_b, \perp \end{bmatrix}$, sets statement $x_b$ as $(C, A, B, \text{pke.pk}, ct_b, w_b, \delta_b)$, witness $\omega_b$ as $(x_b, y_b, z_b, r_b)$, and generates proof $\pi_b \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x_b, \omega_b)$.

   (d) Next, challenger sets message $\mu_b$ as $(w_b, \delta_b)$ and signature $\sigma_b$ as $(\pi_0, ct_b)$. Finally, challenger samples $b \leftarrow \{0, 1\}$ uniformly at random and sends $(\mu_b, \sigma_b, \mu_1-b, \sigma_1-b)$ to the adversary.

5. Adversary sends a bit $b'$ and wins if $\hat{b} = b'$.

Hybrid$_1$ Instead of honestly generating the NIZK proofs $\pi_0$ and $\pi_1$, the challenger simulates $\pi_0$ and $\pi_1$ without any witness.

4.(c) For $b \in \{0, 1\}$, it computes $w_b = A \cdot \begin{bmatrix} x_b, \perp \\ z_b, \perp \end{bmatrix}$ and sets statement $x_b$ as $(C, A, B, \text{pke.pk}, ct_b, w_b, \delta_b)$.

Without setting any witnesses, challenger generates $\pi_b$ using NIZK simulator.

Hybrid$_2$ This is the same as Hybrid$_1$, except that for $b \in \{0, 1\}$, instead of generating $ct_b \leftarrow \text{PKE.Enc}(\text{pke.pk}, x_b || y_b || z_b; r_b)$, challenger sets $ct_b$ as $\text{PKE.Enc}(\text{pke.pk}, 0; r_b)$.

4.(b) Challenger generates $ct_0 \leftarrow \text{PKE.Enc}(\text{pke.pk}, 0; r_0)$ and $ct_1 \leftarrow \text{PKE.Enc}(\text{pke.pk}, 0; r_1)$ with freshly sampled randomness $r_0$ and $r_1$.

Hybrid$_3$ This the same as Hybrid$_2$, except that the challenger samples $\text{pk}_{R_0}$ and $\text{pk}_{R_1}$ uniformly at random.

2. When generating $\text{pk}_{R_b}$ for $b \in \{0, 1\}$, instead of setting it as $A \cdot x_b + H(\delta_b)$, the challenger sets it as a uniformly sampled vector, such that $\text{pk}_{R_b} \leftarrow \$ Z_{q}^{n}$.
This is the same as Hybrid₃, except that the challenger samples vectors \( w₀, w₁ \) uniformly at random.

4.(c) For \( b \in \{0, 1\} \), it samples \( w_b \) uniformly at random, sets statement \( x_b \) as \((C, A, B, pke.pk, ct_b, w_b, δ_b)\), and generates \( π_b \) using NIzk simulator.

Let \( \text{Adv}_A^j \) denote the security advantage of an adversary \( A \) in the NIBS blindness game in Hybrid₇. Then, the following must hold:

**Lemma 6.9.** Assuming zero-knowledge property of NIzk, it holds that for all PPT adversaries \( A \),

\[
|\text{Adv}_A^0 - \text{Adv}_A^1| \leq \text{negl}(λ).
\]

**Proof.** We set the following intermediate step for the proof. In the intermediate step, everything remains the same as Hybrids 0 and 1, except for the following:

4.(c) For \( b \in \{0, 1\} \), it computes \( w_b = A \cdot \begin{bmatrix} x_b, z_b \end{bmatrix} \) and sets statement \( x_b \) as \((C, A, B, pke.pk, ct_b, w_b, δ_b)\).

Challenger only sets witness \( ω₁ \) as \((x₁, y₁, z₁, r)\) without assigning the witness for statement \( x₀ \). Challenger generates \( π₀ \) using NIzk simulator and \( π₁ ← \text{NIzk.Prove}(\text{nizk}.crs, x₁, ω₁) \).

Define adversary \( A \)'s advantage on the intermediate step as \( \text{Adv}_A^{0.5} \). Suppose there exists a PPT attacker \( A \) such that \( |\text{Adv}_A^0 - \text{Adv}_A^{0.5}| = \epsilon(λ) \), we design a reduction algorithm \( B \) that breaks the zero-knowledge property of NIzk with advantage \( \epsilon(λ) \).

Challenger starts by outputing nizk.crs. Reduction algorithm \( B \) samples \( A, B \in \mathbb{Z}_q^{n \times 2m} \) and pke.pk, and simply outputs pp = \((A, B, pke.pk, \text{nizk}.crs)\). Then sends its presignatures and nonces \((\text{psig}_0, \text{nonce}_0)\) and \((\text{psig}_1, \text{nonce}_1)\). Subsequently, \( B \) simulates the Obtain algorithm: For \( b \in \{0, 1\} \), \( B \) sets \( z_b \) as \( \text{psig}_b \) \( y_b \) as \( \text{nonce}_b \), parses \( sk_{R_b} \) as \((x_b, δ_b)\), and checks if \( C \cdot z_b + B \cdot y_b = A \cdot x_b + H(δ_b) \) and \( y_b ∈ \{±1\}^{2m} \) and \( \|z_b\| ≤ √2β \). If any of these checks fails, \( B \) outputs \( ⊥ \) and aborts. Otherwise, for \( b ∈ \{0, 1\} \), it continues to generate \( ct_b ← \text{PKE.Enc}(\text{pke.pk, x}_b, \|y_b\|, z_b; r_b) \) with freshly sampled randomness \( r_b \). Then computes \( w_b = A \cdot \begin{bmatrix} x_b, z_b \end{bmatrix} \) and sets statement \( x_b \) as \((C, A, B, pke.pk, ct_b, w_b, δ_b)\), witness \( ω_b \) as \((x_b, y_b, z_b, r_b)\). Next, \( B \) queries the NIzk challenger with \((x₀, ω₀)\) and challenger returns proof \( π₀ \). \( B \) generates \( π₁ = \text{NIzk.Prove}(\text{nizk}.crs, x₁, ω₁) \) by itself. Next, \( B \) sets message \( μ_b \) as \((w_b, δ_b)\) and signature \( σ_b \) as \((π_b, ct_b)\) for \( b ∈ \{0, 1\} \). \( B \) then samples \( b ← \{0, 1\} \) uniformly at random. Finally, \( B \) sends \((μ_b, σ_b, μ₁-b, σ₁-b)\) to \( A \). \( A \) outputs its guess \( b' \). If \( b' = b \), \( B \) sends \( 0 \) as its guess (i.e., \( π₀ \) is a simulated proof), otherwise it sends \( 1 \) as its guess (i.e., \( π₀ \) is honestly generated from \( \text{NIzk.Prove}(\text{nizk}.crs, x₀, ω₀) \)).

Note that if the NIzk challenger honestly generates proof the \( π₀ \) using NIzk.Prove (nizk.crs, x₀, ω₀), then \( B \) perfectly simulates the experiment of Hybrid₇ for adversary \( A \). Otherwise it simulates the intermediate step for \( A \). As a result, if \( |\text{Adv}_A^0 - \text{Adv}_A^{0.5}| \) is non-negligible, then \( B \) breaks the zero-knowledge property of NIzk with non-negligible advantage. Similarly, \( |\text{Adv}_A^{0.5} - \text{Adv}_A^1| \) is also negligible, and thus \( |\text{Adv}_A^0 - \text{Adv}_A^1| \leq \text{negl}(λ) \).
Lemma 6.10. If public key encryption scheme PKE is an IND-CPA secure public key encryption scheme, then for all PPT adversaries A, $|\text{Adv}^1_A - \text{Adv}^2_A| \leq \text{negl}(\lambda)$.

Proof. Consider the following intermediate step between Hybrids 1 and 2. In the intermediate step, everything remains the same in Hybrids 1 and 2, except for the following:

4. (b) $B$ generates $ct_0 \leftarrow \text{PKE.Enc}(\text{pke.pk}, 0; r_0)$ and $ct_1 \leftarrow \text{PKE.Enc}(\text{pke.pk}, x_1||y_1||z_1; r_1)$ with freshly sampled randomness $r_0$ and $r_1$.

Define adversary A’s advantage on the intermediate step as $\text{Adv}^{1, 5}_A$. Suppose there exists a PPT attacker A such that $|\text{Adv}^1_A - \text{Adv}^{1, 5}_A| = \epsilon(\lambda)$, we design a reduction algorithm B that breaks the security of PKE with advantage $\epsilon(\lambda)$.

Challenger first samples and outputs pke.pk from PKE.KeyGen. Reduction algorithm B then samples $A, B \leftarrow Z_q^{n \times 2m}$ and nizk.crs $\leftarrow \text{NIKZ.Setup}(1^\lambda)$, and outputs pp as $( A, B, \text{pke.pk}, \text{nizk.crs})$. Next, A outputs its verification key vk. For $b \in \{0, 1\}$, B randomly samples key pairs $(sk_{R_0}, pk_{R_0}) \leftarrow \text{KeyGen}_B(1^\lambda, pp)$, $B$ sends $pk_{R_0}$ and $pk_{R_1}$ to A. Adversary A then sends its presignatures and nonces $(\text{psig}_0, \text{nonce}_0)$ and $(\text{psig}_1, \text{nonce}_1)$. Subsequently, for $b \in \{0, 1\}$, B sets $z_b$ as $\text{psig}_b, y_b$ as nonce$_b$, and parses $sk_{R_b}$ as $(x_b, \delta_b)$. B checks if $C \cdot z_b + B \cdot y_b = A \cdot x_b + H(\delta_b)$ and if $|z_b| \leq \sqrt{2}\beta$ for $b \in \{0, 1\}$. If any of the checks fail, $B$ outputs ⊥ and aborts. Otherwise, it queries the challenger with the all zeros string 0 and the string $x_0||y_0||z_0$. Challenger returns ciphertext $ct_0$ (which is an encryption of either 0 or $x_0||z_0$). Then $B$ generates $ct_1 \leftarrow \text{PKE.Enc}(\text{pke.pk}, x_1||y_1||z_1; r)$ with freshly sampled randomness $r$. It computes $w_b = A \cdot \frac{x_b, \perp}{z_b, \perp}$, sets statement $x_b$ as $(C, A, B, \text{pke.pk}, ct_b, w_b, \delta_b)$, and generates $\pi_b$ as a simulated proof for $b \in \{0, 1\}$. Next, $B$ sets message $\mu_b$ as $(w_b, \delta_b)$ and signature $\sigma_b$ as $(\pi_b, ct_b)$ for $b \in \{0, 1\}$. Finally, $B$ samples $b' \leftarrow \{0, 1\}$ uniformly at random and sends $(\mu_{b'}, \sigma_{b'}, \mu_{1-b'}, \sigma_{1-b'})$ to A. A outputs its guess $b'$. If $b' = b$, $B$ sends 0 as its guess (i.e., $ct_0$ is an encryption of 0), otherwise it sends 1 as its guess (i.e., $ct_0$ is an encryption of string $x_0||z_0$).

Note that if $ct_0$ is an encryption of string $x_0||z_0$, then $B$ perfectly simulates the experiment of Hybrid$_1$ for adversary A. Otherwise it simulates the intermediate step. As a result, if $|\text{Adv}^1_A - \text{Adv}^{1, 5}_A|$ is non-negligible, then $B$ breaks the security of public key encryption scheme PKE with non-negligible advantage. Similarly, $|\text{Adv}^{1, 5}_A - \text{Adv}^2_A| \leq \text{negl}(\lambda)$ and thus $|\text{Adv}_A^1 - \text{Adv}_A^3| \leq \text{negl}(\lambda)$.

Lemma 6.11. For all PPT adversaries A, $|\text{Adv}^3_A - \text{Adv}^3_A| \leq \text{negl}(\lambda)$.

Proof. Since $x_1 \leftarrow \mathcal{D}_{Z_q^{n \times m}}$, we have $H_{\infty}(x_0, \tau), H_{\infty}(x_1, \tau) \geq m$. As $m = n \log q + \lambda$, it follows from Leftover Hash Lemma (Corollary 3.8) that the following distributions are statistically indistinguishable:

$$
\{(A_L, A_L \cdot x_0, \tau, A_L \cdot x_1, \tau) : A_L \leftarrow \mathcal{D}_{Z_q^{n \times m}, x_0, \tau \leftarrow \mathcal{D}_{Z_q^{n \times m}, x_1, \tau \leftarrow \mathcal{D}_{Z_q^{n \times m}}}
\} \approx_s \{(A_L, \rho_0, \rho_1) : A_L \leftarrow \mathcal{D}_{Z_q^{n \times m}, \rho_0 \leftarrow \mathcal{D}_{Z_q^n}, \rho_1 \leftarrow \mathcal{D}_{Z_q^n}}
$$

Thus for $b \in \{0, 1\}$, vector $A_L \cdot x_b + H(\delta_b) = A_L \cdot x_b, \tau + A_R \cdot x_b, \perp + H(\delta_b)$ is statistically indistinguishable from a random vector in $Z_q^n$.

Lemma 6.12. For all PPT adversaries A, $|\text{Adv}^3_A - \text{Adv}^3_A| \leq \text{negl}(\lambda)$.
Proof. Note that $x_{0,\perp}, x_{1,\perp} \leftarrow \$ D_{\mathbb{Z}_q^{n,\perp}}$, we have $H_\infty(x_{0,\perp}), H_\infty(x_{1,\perp}) \geq m$. As $m = n \log q + \lambda$.

Again, it follows from Leftover Hash Lemma (Corollary 3.8) that:

$$\{(A_L, A_L \cdot x_{0,\perp}, A_L \cdot x_{1,\perp}) : A_L \leftarrow \$ \mathbb{Z}_q^{n \times m}, x_{0,\perp} \leftarrow \$ D_{\mathbb{Z}_q^n,\perp}/m, x_{1,\perp} \leftarrow \$ D_{\mathbb{Z}_q^n,\perp}/m\}$$

\[\approx s \{(A_L, \rho_0, \rho_1) : A_L \leftarrow \$ \mathbb{Z}_q^n, \rho_0 \leftarrow \$ D_{\mathbb{Z}_q^n,\perp}/m\} \]

As a result, for $b \in \{0, 1\}$ and any $\mu \in \mathbb{Z}_q^n$, vector $w_b = \mu + A_L \cdot x_{b,\perp}$ is statistically indistinguishable from a random vector in $\mathbb{Z}_q^n$. \qed

Finally, note that any adversary has 0 advantage in Hybrid$_4$. From the above lemmas, it follows that the above NIBS construction satisfies recipient blindness. \qed

### 6.4 Efficiency of lattice-based NIBS

We use the same basic building blocks as Agrawal et al. [AKSY22] to instantiate the protocol in Construction 6.1. For each block, we identify the similarities to their scheme and then provide details on key differences in, both, our specific instantiation as well as the choice of parameters. We appropriately modify the Python script in [AKSY22] to obtain parameter estimates for our scheme.

** Trapdoor generation and preimage sampling.** We instantiate Falcon-512 [PHK+17] over the ring $\mathcal{R}_{512} = \mathbb{Z}[x]/(q_F, x^{512} + 1)$. The Falcon modulus, $q_F$ is chosen to be a prime greater than $2^{12.8}$, that is congruent to 5 (mod 8) in order to simplify the linear relations proven in the zero-knowledge proof later. Although the smallest such prime, 7213, would be the natural choice, it turns out that to satisfy certain bounds on $q_{\text{PKE}}$, a higher prime is needed. We set $q_F = 7349$ and carry over the other Falcon parameters $n_F = 512, m_F = 1024, \varsigma$ and $\beta = 1.1 \cdot \varsigma \cdot \sqrt{m_F}$ from [AKSY22].

An important distinction in our construction is the use of the Bonsai trick [CHKP10] to generate a short preimage for $t - b \cdot y_1 + y_2$, where the receiver’s public key $t$, the public polynomial $b$, and the signer generated $y_1$ and $y_2$ are all polynomials in $\mathcal{R}_{512}$. For NTRU lattices, the signer computes this preimage by sampling polynomials $y_1, y_2, z_1', z_2' \in \mathcal{R}_{512}$ such that $\| [z_1' \mid z_2'] \|$ is small, and the coefficients of $y_1$ and $y_2$ are in $\{\pm 1\}$. It then uses its NTRU secret trapdoor to sample short polynomials $z_1, z_2 \in \mathcal{R}_{512}$ such that $c \cdot z_1 + z_2 = t - b \cdot y_1 - y_2 - c \cdot z_1' - z_2'$. 

The presignature is then $(z_1, z_1', z_2')$ and the nonce is $(y_1, y_2)$. Given this, the Receiver can recover $z_2$. We require that $\| [z_1' \mid z_2'] \|_\infty \leq 2$ so as to extract short preimages in the randomized one-more ISIS game. Using this, we estimate the size of the presignature to be $\approx 1$ KB. We note that the Receiver’s first message (here the public key value $t$) is computed identically to [AKSY22], i.e.,

$$t = a' \cdot x_1 + x_2 + H(\delta),$$

where polynomial $a'$ and hash function $H$ are specified in the protocol’s public parameters, and $x_1, x_2$ are sampled from $\mathcal{R}_{512}$ such that $\| [x_1 \mid x_2] \|_\infty \leq 2$ and $\delta \leftarrow \$ \{0, 1\}^\lambda$. The Receiver’s secret key is $( [x_1 \mid x_2], \delta )$. 

39
IND-CPA secure PKE. The Regev style public key encryption scheme from \cite{LPS10} is instantiated over \( R_{128} = \mathbb{Z}[x]/(q_{\text{PKE}}, x^{128} + 1) \). Here, we chose \( q_{\text{PKE}} = q_{F} \cdot 8933 = 65648617 \) and the integer \( p = 61583 \) under the same constraints as listed in \cite{AKSY22}. In our case, the Receiver must encrypt \((x_1^T \mid x_2^T \mid y_1^T \mid z_1^T \mid z_2^T \mid z_2'^T) \in (R_{128})^7 \cong R_{128}^{28} \). Concretely, we must encrypt four additional polynomials over \cite{AKSY22}, which results in a ciphertext of 14.6 KB. Reducing the size of this ciphertext is left as an open problem.

Zero-knowledge proof. Subject to the instantiation listed above, the receiver must prove knowledge of polynomials \( x_1, x_2, y_1, y_2, z_1, z_2, z'_1, z'_2 \in R_{512} \) satisfying

\[
c_1 \cdot z_1 + z_2 + c_2 \cdot z'_1 + z'_2 + b \cdot y_1 + y_2 - a \cdot x_1 - x_2 = H(\delta)
\]

and the coefficients of \( y_1 \) and \( y_2 \) are in \( \{\pm 1\}^{12} \). The receiver must also prove that part of the message, \( w = a \cdot z_1 + x_2 \). It must further prove that it knows encryption randomness \( s, e_1 \in R_{128}^{512} \) and \( e_2 \in R_{128}^{28} \) such that the ciphertext \( ct \) is an encryption of \((x_1^T \mid x_2^T \mid y_1^T \mid z_1^T \mid z_2^T \mid z'_1^T \mid z'_2^T) \) under pke.pk. Lastly, it needs to prove that the following bounds hold

\[
(i) \quad \|[z_1 \mid z_2]\|_2 \leq \beta \quad \quad (ii) \quad \|[z'_1 \mid z'_2]\|_2 \leq 2 \cdot \sqrt{2 \cdot 512} \\
(iii) \quad \|[x_1 \mid x_2]\|_2 \leq 2 \cdot \sqrt{2 \cdot 512} \quad (iv) \quad \|[s \mid e_1 \mid e_2]\|_2 \leq \tau \cdot \sqrt{(2 \cdot 8 + 28) \cdot 128}
\]

Since all our relations are linear or proving norm bounds, we are able to leverage the efficient zero-knowledge proof framework of \cite{LNP22b}. For specific details concerning the implementation, we direct the reader to confer \cite{LNP22b} and remark only that we must prove just one extra norm bound (and two additional quadratic relations for coefficients of \( y_i \)) over \cite{AKSY22} and the one extra relation pertaining the message \( w \) can be proven essentially for free. The proof can then be further compressed using \cite{DKL18}, in effect cutting low-order bits of the commitment.

The process for selecting the parameters is the same as in \cite{AKSY22} so we only highlight the main differences. We set the modulus \( q_{ZK} = q_{F} \cdot q_{\text{PKE}} \cdot 99901 = 6558362486917 \approx 2^{42.5} \), we chose \( m_2 = 35 \) so that Module-LWE problem underlying the zero-knowledge protocol is hard, and the length of the committed vector over \( R_{128} \) is

\[
m_1 = 10 \cdot 4 + 2 \cdot 8 + 28 + \frac{4}{s.e_1.\ell_2\text{-norms}} + \frac{2}{y_i\text{'s coefficients}} = 90.
\]

The final proof size after the cut is 53.1 KB and the signature is therefore 67.7 KB. We believe that further reduction in size may be possible using the recent optimizations given in \cite{LN22} although we do not attempt to do so here.

Concrete security. We also estimate the concrete security of the given instantiation using the script. Our observations are summarized in Table 3.

\[\text{For each } y_i, \text{ we must prove the linear relation } y_i = 2y'_i - 1 \text{ and the quadratic relation } y'_i \cdot (y'_i - 1) = 0.\]
<table>
<thead>
<tr>
<th>Security Property</th>
<th>Assumption</th>
<th>Bit security (Core-SVP Hardness)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key recovery (Falcon)</td>
<td>NTRU</td>
<td>135 bits</td>
</tr>
<tr>
<td>Unforgeability (Falcon)</td>
<td>MSIS</td>
<td>121 bits</td>
</tr>
<tr>
<td>Hiding for (pk) and (w)</td>
<td>MLWE</td>
<td>122 bits</td>
</tr>
<tr>
<td>Secret key and ciphertext security (Encryption)</td>
<td>MLWE</td>
<td>117 bits</td>
</tr>
<tr>
<td>Zero-knowledge proof</td>
<td>MSIS and MLWE</td>
<td>108 and 120 bits (resp.)</td>
</tr>
</tbody>
</table>

Table 3: Overall security of our protocol.

7 Lattice-based Tagged NIBS

Let us begin this section by recalling the security definitions for TNIBS.

7.1 Tagged NIBS definitions

We also recall the definition for a tagged non-interactive blind signature (TNIBS) scheme. A TNIBS scheme has the same set of algorithms as a regular NIBS scheme, except with the following differences:

Setup, KeyGen\(_S\), KeyGen\(_R\) are defined as for a regular NIBS scheme.

\[
\text{Issue}(sk, pk_R, \tau) \rightarrow (\text{psig}, \text{nonce}). \quad \text{The issue algorithm now takes a tag } \tau \in \mathcal{T} \text{ as an additional input, rest is as before.}
\]

\[
\text{Obtain}(sk_R, pk, (\text{psig}, \text{nonce}), \tau) \rightarrow (\mu, \sigma). \quad \text{The obtain algorithm also takes a tag } \tau \text{ as an additional input, with the rest of the syntax being same as before.}
\]

\[
\text{Verify}(vk, \mu, \tau, \sigma) \rightarrow \{0, 1\}. \quad \text{The verification algorithm also takes a tag } \tau \text{ as an additional input.}
\]

As with a NIBS scheme, tagged NIBS is required to satisfy the following security properties.

**Definition 7.1 (One-more unforgeability).** A TNIBS scheme \(S\) satisfies one-more unforgeability, if for every stateful admissible PPT adversary \(A\), there exists a negligible function \(\text{negl}(\cdot)\) such that for every \(\lambda \in \mathbb{N}\), the following holds:

\[
\Pr \left[ \bigwedge_{i \in [\ell+1]} \text{Verify}(vk, \mu_i, \tau_i, \sigma_i) = 1 \bigwedge_{i \neq j \in [\ell+1]} (\mu_i \neq \mu_j) \quad \text{pp} \leftarrow \$ \text{Setup}(1^\lambda) \right. \left. \text{(sk, vk)} \leftarrow \$ \text{KeyGen}_{S}(\text{pp}) \right. \left. \text{(}\{(\tau_i, \mu_i, \sigma_i)\}_{i=1}^{\ell+1}\right) \leftarrow \$ \text{O}_{sk}(.;\cdot)(vk) \right] \leq \text{negl}(\lambda),
\]

where \(O_{sk}(\cdot, \cdot)\) takes as input a receiver’s public key \(pk_R\), and a tag \(\tau\), and outputs a presignature-nonce pair \((\text{psig}_i, \text{nonce}_i)\) by running Issue\((sk, pk_R, \tau)\), and \(A\) is an admissible adversary iff \(A\) makes at most \(\ell\) queries to \(O_{sk}\).
Definition 7.2 (Receiver blindness). A TNIBS scheme $S$ satisfies receiver blindness, if for every stateful admissible PPT adversary $A$, there exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, the following holds:

$$\text{Pr} \left[ A(\mu_b, \sigma_b, \mu_{1-b}, \sigma_{1-b}) = \hat{b} : \begin{align*} pp &\leftarrow \$ \text{Setup}(1^\lambda), \hat{b} \leftarrow \{0,1\}, \\ \forall b \in \{0,1\} : (sk_{R_b}, pk_{R_b}) &\leftarrow \$ \text{KeyGen}_R(pp) \\ (vk, (\text{psig}_b, \text{nonce}_b), \tau) &\leftarrow \$ \text{Enc}(pk_{R_b}, pk_{R_1}) \\ \forall b \in \{0,1\} : (\mu_b, \sigma_b) &\leftarrow \$ \text{Obtain}(sk_{R_b}, vk, (\text{psig}_b, \text{nonce}_b), \tau) \end{align*} \right] \leq \frac{1}{2} + \text{negl}(\lambda),$$

where $A$ is an admissible adversary iff $\sigma_0, \sigma_1 \not\perp$ (i.e., Obtain algorithm does not abort).

Definition 7.3 (Reusability). A TNIBS scheme $S$ satisfies the reusability property, if there exists a negligible function $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, every tag $\tau \in T$, the following holds:

$$\text{Pr} \left[ \mu_0 = \mu_1 : \begin{align*} pp &\leftarrow \$ \text{Setup}(1^\lambda) \\ (sk, vk) &\leftarrow \$ \text{KeyGen}_S(pp), (sk_{R}, pk_{R}) &\leftarrow \$ \text{KeyGen}_R(pp) \\ \forall b \in \{0,1\} : (\text{psig}_b, \text{nonce}_b) &\leftarrow \$ \text{Issue}(sk, pk_{R}, \tau) \\ \forall b \in \{0,1\} : (\mu_b, \sigma_b) &\leftarrow \$ \text{Obtain}(sk_{R_b}, vk, (\text{psig}_b, \text{nonce}_b), \tau) \end{align*} \right] \leq \text{negl}(\lambda).$$

7.2 Construction

We now upgrade our NIBS scheme to a tagged NIBS. Let parameters $n, m, \zeta, \beta = \zeta \sqrt{m}$ and prime $q$ be such that the randomized one-more ISIS instance $\text{rOM-ISIS}_{q,n,2m,\zeta,(3\sqrt{2}+\epsilon)\beta}$ is hard, where $\epsilon = O(m^{-1/2})$. These parameters must satisfy the constraints in (1).

Tools required. The construction relies on a public key encryption scheme $\text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec})$, a lattice trapdoor $\text{bLT} = (\text{TrapGen}, \text{SamplePre})$, and a NIZK scheme $\text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify})$ for the following language:

**Language $\mathcal{L}_2$**

- **Instance:** Each instance $x$ is interpreted as matrices $C, A$ and $B$, PKE public key $pk$, ciphertext $ct$, vector $w$, random string $\delta$ and the tag $\tau$.
- **Witness:** Witness $\omega$ consists of a secret vector $x$, nonce vector $y$, presignature vector $z$ and randomness $r$.
- **Membership:** $\omega$ is a valid witness for $x$ if the following are satisfied:
  - $\|z\| \leq \sqrt{2}\beta$ and $\|x\| \leq \sqrt{2}\beta/m$ and $y \in \{\pm1\}^{2m}$.
  - $ct = \text{PKE.Enc}(pk, x||y||z; r)$
  - $C \cdot z + B \cdot y = A \cdot x + H_1(\delta) + H_2(\tau)$ and $w = A_L \cdot x_L + A_R \cdot z_L$

Below we describe our tagged non-interactive blind signature scheme. Both, completeness and reusability of the above construction follows from the completeness and reusability of construction 6.1. .
Setup$(1^λ, n, m, q, ς, H_1, H_2) \rightarrow pp$. It is exactly the same as the Setup algorithm for Construction 6.1, except for the following: It takes as input two hash functions $H_1$ and $H_2$, which are used as random oracles in the design. It outputs the public parameters as

$$pp := (A, B, pke.pk, nizk.crs).$$

KeyGen$_S(pp) \rightarrow (sk, vk)$. Signer’s key generator algorithm is exactly the same as KeyGen$_S$ in Construction 6.1.

KeyGen$_R(pp) \rightarrow (sk_R, vk_R)$. It first samples $x \leftarrow D_{\mathbb{Z}_{2^m}^\perp \cdot r}$ and $\delta \leftarrow \{0, 1\}^λ$. Next, it computes

$$t = A \cdot x + H_1(\delta).$$

It outputs user’s secret key and public key as $(sk_R := (x, \delta), pk_R := t)$.

Issue$(sk, pk_R, τ) \rightarrow (psig, nonce)$. Same as the Issue algorithm in Construction 6.1, except for the following: Instead of setting $t$ as $pk_R$, it sets $t = pk_R + H_2(τ)$.

Next, it still sets $z$ as $\text{bLT}.\text{SamplePre}(C, T_C, t - B \cdot y, ς)$. It outputs presignature $psig := z$ and nonce $nonce := y$.

Obtain$(sk_R, vk, psig, nonce, τ) \rightarrow (\mu, σ, τ)$. It parses $sk_R$ as $(x, δ)$. Then, it assigns $C := vk$, $y := nonce$, and $z := psig$. It first checks if $C \cdot z + B \cdot y = A \cdot x + H_1(δ) + H_2(τ)$ and $\|z\| \leq \sqrt{2}β$ and $y \in \{±1\}^{2m}$. If any check fails then it aborts and outputs $⊥$.

Otherwise, it generates $ct \leftarrow \text{PKE}.\text{Enc}(pke.pk, x \parallel y \parallel z \parallel r)$ from uniformly sampled randomness $r \leftarrow \{0, 1\}^λ$. The obtain algorithm sets $w$ as $w = A \cdot \begin{bmatrix} x_\perp \\ z_\perp \end{bmatrix}$, where $x_\perp, z_\perp \in \mathbb{Z}_q^m$ and generates NIZK proof $π$ as $π \leftarrow \text{NIZK}.\text{Prove}(\text{nizk}.\text{crs}, x := (C, A, B, pke.pk, ct, w, δ, τ), ω := (x, y, z, r))$. Finally, it outputs message $μ := (w, δ)$, signature $σ := (π, ct)$, and tag $τ$.

Verify$(vk, μ, τ, σ) \rightarrow \{0, 1\}$. It parses $μ$ as $(w, δ)$ and $σ$ as $(π, ct)$. The verification algorithm accepts and outputs 1 if and only if $\text{NIZK}.\text{Verify}(\text{nizk}.\text{crs}, x := (C, A, B, pke.pk, ct, w, δ, τ), π) = 1$. Otherwise, it outputs 0.

Completeness. Observe that by correctness of bLT.\text{SamplePre}, $z$ is a presignature on $A \cdot x + H_1(δ) + H_2(τ) - B \cdot y$, ie., $C \cdot z = A \cdot x + H_1(δ) + H_2(τ) - B \cdot y$ such that $\|z\| \leq \sqrt{2}β$ and $y_i \in \{±1\}$ for all $i \in [2m]$ where $y_i$ is the $i^{th}$ element of $y$. Since, it further holds by construction that $w = A_L \cdot x_\perp + A_R \cdot z_\perp$, ct = \text{PKE}.\text{Enc}(pke.pk, x \parallel y \parallel z \parallel r)$, and (by Lemma 3.1) $\|x\| \leq \sqrt{2}β/m$, we have that $π$ is a valid NIZK proof for the language $L_2$. It therefore follows from the completeness of the underlying NIZK that Construction 7.2 is complete.

Reusability. The reusability of the tagged NIBS protocol can be shown identically as that of the (un-tagged) NIBS protocol 6.1.

7.3 One-more unforgeability

We first prove the one-more unforgeability of this protocol.
Theorem 7.4. Assume that NIZK proof system NIZK satisfies soundness, lattice trapdoor bLT satisfies well-distributedness, and the rOM-ISIS assumption holds, then our construction 7.2 is one-more unforgeable.

Proof. We present the following hybrids:

Hybrid$_0$ This is the actual one-more unforgeability game for tagged NIBS:

1. Challenger samples $A, B \leftarrow \mathbb{Z}_q^{n \times 2m}$ uniformly at random, $pke.pk \leftarrow$ PKE.KeyGen$(1^\lambda)$, $nizk.crs \leftarrow$ NIZK.Setup$(1^\lambda)$ and sets $pp$ as $(A, B, pke.pk, nizk.crs)$. Next, it samples $(T_C, C) \leftarrow$ bLT.TrapGen$(1^\lambda, n, 2m, q)$ and sets $sk := T_C, vk := C$. Challenger sends $pp$ and $vk$ to the adversary.

2. Adversary chooses $pk_R$ and tag $\tau$, and requests a presignature, to which the challenger replies with $(\text{psig}, \text{nonce}) := \text{Issue}(sk, pk_R, \tau)$. The adversary repeats this step a total of $\ell$ times. In this step, adversary is allowed to make hash queries to $H_1$ and $H_2$, which are considered as random oracle queries in our design.

3. Adversary outputs tag $\tau$ and $k$ message and signature pairs $((\tau_1, \mu_1, \sigma_1), \ldots, (\tau_k, \mu_k, \sigma_k))$. Adversary wins if $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$, $\text{Verify}(vk, \mu_i, \sigma_i, \tau_i) = 1$ for all $i \in [k]$, and $k = \ell + 1$.

Hybrid$_1$ This is the same as Hybrid$_0$ aside from one key difference that instead of uniformly sampling the matrix $A$, it is chosen by first sampling a matrix $R \leftarrow \{0, 1\}^{2m \times 2m}$ and then setting $A = C \cdot R$.

1. Challenger samples $R \leftarrow \{0, 1\}^{2m \times 2m}$ uniformly at random and sets $A = C \cdot R$. Challenger then samples $B \leftarrow \mathbb{Z}_q^{n \times 2m}$, $pke.pk \leftarrow$ PKE.Setup$(1^\lambda)$, and sets $pp$ as $(A, B, pke.pk, nizk.crs)$. Next, it samples $T_C, C \leftarrow$ bLT.TrapGen$(1^\lambda, n, 2m, q)$ and sets $sk := T_C, vk := C$. Challenger sends $pp$ and $vk$ to the adversary.

Hybrid$_2$ This is exactly the same as Hybrid$_1$ except the following: Whenever $A$ makes a fresh hash query to $H_2$, instead of responding with a uniformly random output, challenger instead outputs $C \cdot r$ where $r \leftarrow D_{\mathbb{Z}_q^{2m \times \ell/m}}$.

2. Adversary chooses $pk_R$ and tag $\tau$, and requests a presignature, to which the challenger replies with $(\text{psig}, \text{nonce}) := \text{Issue}(sk, pk_R, \tau)$. The adversary repeats this step a total of $\ell$ times. In this step, adversary is allowed to make hash queries to $H_1$ and $H_2$. When adversary makes fresh hash queries to $H_1$, challenger replies with freshly sampled random vectors in $\mathbb{Z}_q^{2m}$. When adversary makes fresh hash queries to $H_2$, challenger samples $r \leftarrow D_{\mathbb{Z}_q^{2m \times \ell/m}}$ and replies with $C \cdot r$.

Let $\text{Adv}_A^1$ denote the advantage of an adversary $A$ in the TNIBS one-more unforgeability game in Hybrid$_1$. Then, the following must hold:

Lemma 7.5. For any PPT adversary $A$, $|\text{Adv}_A^0 - \text{Adv}_A^1| \leq \text{negl}(\lambda)$
Proof. The proof is omitted as it is identical to the proof of Lemma 6.2.

Lemma 7.6. For any PPT adversary $\mathcal{A}$, $|\text{Adv}^1_{\mathcal{A}} - \text{Adv}^2_{\mathcal{A}}| \leq \text{negl}(\lambda)$

Proof. By leftover hash lemma (Corollary 3.8), the following holds:

$$\{(C', C' \cdot r) : C' \leftarrow \mathbb{Z}_{q}^{n \times 2m}, r \leftarrow \mathbb{D}_{\mathbb{Z}_{q}^{m \times k}}\} \approx_s \{(C', v) : C' \leftarrow \mathbb{Z}_{q}^{n \times 2m}, v \leftarrow \mathbb{Z}_{q}\}$$

By Corollary 3.8, $C'$ is statistically close to $C$ and thus $C' \cdot r$ is indistinguishable from a randomly sampled vector.

Lemma 7.7. Assume that NIZK argument is sound and rOM-ISIS assumption holds, then the one-more-unforgeability adversary $\mathcal{A}$ can have at most negligible advantage in Hybrid$_2$.

Proof. Suppose there exists a PPT attacker $\mathcal{A}$ that wins the one-more-unforgeability game with non-negligible advantage $\epsilon = \epsilon(\lambda)$. It outputs a signature and $k$ message and signature pairs $((\tau_1, \mu_1, \sigma_1), \ldots, (\tau_k, \mu_k, \sigma_k))$, and wins if and only if $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$. Verify $(vk, \mu_i, \tau_i, \sigma_i) = 1$ for $i \in [k]$ and $k = \ell + 1$. Consider that we parse the input and signature pairs $(\mu_i, \sigma_i)$ as $((\omega_i, \delta_i), (\pi_i, ct_i))$ for all $i \in [k]$. Divide the potential attacker into the following types:

- Type 1: This is nearly identical to the type 1 attacker in the proof of Lemma 6.3, except that instance $x^*$ is defined as $x^* := (C, A, B, \text{pke.pk}, ct_j, w_j, \delta_j, \tau_j)$.

- Type 2: This is a natural extension from type 2 attacker in Lemma 6.3. For type 2 attacker, there exists some $j \in [k]$, such that $C \cdot z_j + B \cdot y_j = A \cdot x_j + H_1(\delta_j) + H_2(\tau_j)$, and $\mathcal{A}$ only queries at most one of $H_1(\delta_j)$ and $H_2(\tau_j)$.

- Type 3: Definition of type 3 attacker follows from type 3 attacker in Lemma 6.3. We still require the following: For all $i \in [k]$, $x_i := (C, A, B, \text{pke.pk}, ct_i, w_i, \delta_i, \tau_i)$ is a valid instance for language $L_2$. $\mathcal{A}$ queries $H_1(\delta)$ at least once for each fresh $\delta$ in the message $(w, \delta)$ it outputs. Additionally, it makes at least one hash query to $H_2(\tau)$ for each fresh tag $\tau$.

Claim 7.8. Assuming that NIZK satisfies soundness property, then type 1 attacker has at most negligible advantage.

Proof. Omitted as it is nearly identical to the proof of Claim 6.4.

Claim 7.9. Type 2 attacker $\mathcal{A}$ has at most negligible advantage.

Proof. Following from the proof of Claim 6.5, $H_1(\delta_j)$ and $H_2(\tau_j)$ look random in the view of $\mathcal{A}$. Thus without querying $H_1(\delta_j)$ or $H_2(\tau_j)$, $\mathcal{A}$ has at most negligible advantage in finding both $\delta_j$ and $\tau_j$ such that $H_1(\delta_j) + H_2(\tau_j) = C \cdot z_j + B \cdot y_j - A \cdot x_j$.

Claim 7.10. Assuming that rOM-ISIS assumption holds, then type 3 attacker $\mathcal{A}$ has at most negligible advantage.
Proof. If type 3 attacker $A$ has non-negligible advantage, then we build reduction algorithm $B$ that breaks the rOM-ISIS security game. Here, $B$ is nearly identical to the reduction algorithm in the proof of Claim 6.6, except for the following: When $A$ makes a hash query to $H_1$ using fresh input $\delta$, $B$ makes a syndrome query to the rOM-ISIS challenger. It forwards the output from the challenger to $A$ as the output of $H_1(\delta)$. When $A$ makes a hash query $H_2(\tau_i)$ using fresh input $\tau_i$, $B$ samples a random vector $r_i \leftarrow \mathcal{D}_{\mathbb{Z}^m_{q_i}/\delta/m}$ and sets $H_2(\tau_i)$ as $C \cdot r_i$. $A$ wins the one-more unforgeability game by outputting $((\tau_1, \mu_1, \sigma_1), \ldots, (\tau_k, \mu_k, \sigma_k))$ such that $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$, and Verify$(pp, vk, \mu_i, \tau_i, \sigma_i) = 1$ for all $i \in [k]$.

Consider that for all $i \in [k]$, $B$ decrypts $ct_i$ using $pke$ and obtains $x_i$ and $z_i$ following from the proof of Claim 6.6, we have:

$$H_1(\delta_i) = C \cdot z_i + B \cdot y_i - A \cdot x_i - H_2(\tau_i) = C \cdot (z_i - R \cdot x_i - r_i) + B \cdot y_i.$$ 

Since $\|z_i\| \leq \sqrt{2} \beta$, $\|x_i\| \leq \sqrt{2} \beta/m$, $\|r_i\| \leq \sqrt{2} \beta/m$, and $R \in \{0, 1\}^{2m \times 2m}$,

$$\|z_i - R \cdot x_i - r_i\| \leq 3\sqrt{2} \beta + \sqrt{2} \beta/m.$$ 

Following from the case by case argument of Claim 6.6, $B$ wins the randomized one-more ISIS game with non-negligible advantage by sending back $z_i - R \cdot x_i - r_i$ and $y_i$ for $i \in [k]$. □

Any successful one-more unforgeability attacker $A$ must be one of type 1, 2, 3, 4. Thus the lemma follows. □

Combining the above lemmas, our main theorem follows. □

7.4 Receiver blindness

Theorem 7.11. Assume that NIZK proof system NIZK satisfies zero knowledge property, and public key encryption scheme $PKE$ is IND-CPA secure, then the tagged NIBS construction 7.2 is receiver blind.

Proof. Below are the hybrids:

Hybrid$_0$ This corresponds to the real experiment.

1. Challenger samples $A, B \leftarrow \mathcal{Z}^{n \times 2m}_{q}$ uniformly at random, $pke.pk \leftarrow \mathcal{D}$ $PKE.KeyGen(1^\lambda)$, and $nizk.crs \leftarrow \mathcal{D}$ $NIZK.Setup(1^\lambda)$. Challenger then sets $pp$ as $(A, B, pke.pk, nizk.crs, H_1, H_2)$ and sends $pp$ to the adversary.

2. Next, challenger samples $(sk_{R_b}, pk_{R_b}) \leftarrow KeyGen_{R}(pp)$, for $b \in \{0, 1\}$. The challenger sends $pk_{R_0}$ and $pk_{R_1}$ to $A$.

3. The adversary outputs a matrix $C \in \mathcal{Z}^{n \times 2m}_{q}$ as the verification key $vk$, two presignatures and nonces $(psig_0, nonce_0)$ and $(psig_1, nonce_1)$, and tag $\tau$.

4. The challenger then generates message and signature pairs:
(a) For \( b \in \{0, 1\} \), challenger sets \( z_0 \) as \( \text{psig}_b, y_b \) as nonce\(_b\), and parses \( \text{sk}_{R_b} \) as \( (x_b, \delta_b) \). It then checks whether \( C \cdot z_b + B \cdot y_b = A \cdot x_b + H_1(\delta_b) + H_2(\tau) \) and \( \|z_b\| \leq \sqrt{\beta} \) and \( y_b \in \{\pm 1\}^{2m} \). If any of the checks fails, challenger outputs \( \bot \) and aborts.

(b) Otherwise, for \( b \in \{0, 1\} \), challenger continues to generate \( \text{ct}_b \leftarrow \text{PKE.Enc(pke.pk, x}_b\|y}_b\|z}_b; r_b \) with freshly sampled randomness \( r_0 \) and \( r_1 \).

(c) For \( b \in \{0, 1\} \), it computes \( w_b = A \left[ x_b, z_b \right] \), sets statement \( x_b \) as \( (C, A, B, \text{pke.pk, ct}_b, w_b, \delta_b, \tau) \), witness \( \omega_b \) as \( (x_b, y_b, z_b, r_b) \), and generates proof \( \pi_b \leftarrow \text{NIZK.Prove(nizk.crs, } x_b, \omega_b) \).

(d) Next, challenger samples message \( \mu_b \) as \( (w_b, \delta_b) \) and signature \( \sigma_b \) as \( (\pi_b, \text{ct}_b) \). Finally, challenger samples \( b \leftarrow \{0, 1\} \) uniformly at random and sends \( (\mu_b, \sigma_b, \mu_1 \cdot \delta_b, \sigma_1 \cdot \delta_b) \) to the adversary.

Hybrid\(_1\) Instead of honestly generating the NIZK proofs \( \pi_0 \) and \( \pi_1 \), the challenger simulates \( \pi_0 \) and \( \pi_1 \) without any witness.

4.(c) For \( b \in \{0, 1\} \), it computes \( w_b = A \left[ x_b, z_b \right] \) and sets statement \( x_b \) as \( (C, A, B, \text{pke.pk, ct}_b, w_b, \delta_b) \). Without setting any witnesses, challenger generates \( \pi_b \) using NIZK simulator.

Hybrid\(_2\) For \( b \in \{0, 1\} \), instead of generating \( \text{ct}_b \leftarrow \text{PKE.Enc(pke.pk, x}_b\|y}_b\|z}_b; r_b \), challenger sets \( \text{ct}_b \) as \( \text{PKE.Enc(pke.pk, 0; r}_b) \).

4.(b) Challenger generates \( \text{ct}_0 \leftarrow \text{PKE.Enc(pke.pk, 0; r}_0) \) and \( \text{ct}_1 \leftarrow \text{PKE.Enc(pke.pk, 0; r}_1) \) with freshly sampled randomness \( r_0 \) and \( r_1 \).

Hybrid\(_3\) This the same as Hybrid\(_2\), except that the challenger samples \( \text{pk}_{R_0} \) and \( \text{pk}_{R_1} \) uniformly at random.

2. When generating \( \text{pk}_{R_b} \) for \( b \in \{0, 1\} \), instead of setting it as \( A \cdot x_b + H_1(\delta_b) + H_2(\tau) \), the challenger sets it as a uniformly sampled vector, such that \( \text{pk}_{R_b} \leftarrow \mathbb{Z}_q^n \).

Hybrid\(_4\) This is the same as Hybrid\(_3\), except that the challenger samples vectors \( w_0, w_1 \) uniformly at random.

4.(c) For \( b \in \{0, 1\} \), it samples \( w_b \) uniformly at random, sets statement \( x_b \) as \( (C, A, \text{pke.pk, ct}_b, w_b, \delta_b, \tau) \), and generates \( \pi_b \) using NIZK simulator.

Lemma 7.12. Assuming zero-knowledge property of NIZK, then for all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}^0_{\mathcal{A}} - \text{Adv}^3_{\mathcal{A}}| \leq \text{negl}(\lambda) \).

Proof. (omitted) follows by extending the proof of Lemma 6.9 to Construction 7.2.

Lemma 7.13. If public key encryption scheme PKE is an IND-CPA secure public key encryption scheme, then for all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}^1_{\mathcal{A}} - \text{Adv}^2_{\mathcal{A}}| \leq \text{negl}(\lambda) \).
Proof. (omitted) follows by extending the proof of Lemma 6.10.

Lemma 7.14. For all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}^2_{\mathcal{A}} - \text{Adv}^3_{\mathcal{A}}| \leq \text{negl}(\lambda) \).
Proof. (omitted) follows by extending the proof of Lemma 6.11.

Lemma 7.15. For all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}^3_{\mathcal{A}} - \text{Adv}^4_{\mathcal{A}}| \leq \text{negl}(\lambda) \).
Proof. (omitted) follows by extending the proof of Lemma 6.12.

Since the adversary has 0 advantage in Hybrid, Construction 7.2 satisfies recipient blindness.

8 Lattice-based NIBS with Nonce Blindness

We recall nonce blindness as defined by [Han23], which captures the fact that a malicious signer cannot link final signatures to presignatures even if they all belong to the same receiver (intra-receiver blindness). As discussed previously, this is more of an “ordering” property, i.e. the adversary should not be able to figure out the order with which signatures were obtained from presignatures.

8.1 Recalling nonce blindness definition

Definition 8.1 (Nonce blindness). A NIBS scheme \( S \) satisfies nonce blindness, if for every stateful admissible PPT adversary \( \mathcal{A} \), there exists a negligible function \( \text{negl}(\cdot) \) such that for every \( \lambda \in \mathbb{N} \), the following holds:

\[
\Pr \left[ \mathcal{A}(\mu_b, \sigma_b, \mu_{1-b}, \sigma_{1-b}) = \hat{b} : \right.
\begin{align*}
&\text{pp} \leftarrow \text{Setup}(1^\lambda), \ (sk_R, pk_R) \leftarrow \text{KeyGen}_R(pp) \ \hat{b} \leftarrow \{0, 1\} \\
&\forall b \in \{0, 1\} : \ (vk, (\text{psig}_b, \text{nonce}_b)) \leftarrow \mathcal{A}(pk_R), \\
&\forall b \in \{0, 1\} : \ (\mu_b, \sigma_b) \leftarrow \text{Obtain}(sk_R, vk, (\text{psig}_b, \text{nonce}_b))
\end{align*}
\right] \leq \frac{1}{2} + \text{negl}(\lambda),
\]

where \( \mathcal{A} \) is an admissible adversary if \( \sigma_0, \sigma_1 \neq \perp \) (i.e., Obtain algorithm does not abort).

Similar to its receiver blindness counterpart, nonce blindness too can be strengthened as shown below.

Definition 8.2 (Strong nonce blindness). A NIBS scheme \( S \) satisfies nonce blindness, if for every stateful admissible PPT adversary \( \mathcal{A} \), there exists a negligible function \( \text{negl}(\cdot) \) such that for every \( \lambda \in \mathbb{N} \), the following holds:

\[
\Pr \left[ \mathcal{A}^{O_{sk_R}(\cdot)}(\mu_b, \sigma_b, \mu_{1-b}, \sigma_{1-b}) = \hat{b} : \right.
\begin{align*}
&\text{pp} \leftarrow \text{Setup}(1^\lambda), \ (sk_R, pk_R) \leftarrow \text{KeyGen}_R(pp) \ \
&(vk, (\text{psig}_b, \text{nonce}_b)) \leftarrow \mathcal{A}(pk_R), \ \hat{b} \leftarrow \{0, 1\} \\
&\forall b \in \{0, 1\} : \ (\mu_b, \sigma_b) \leftarrow \text{Obtain}(sk_R, vk, (\text{psig}_b, \text{nonce}_b))
\end{align*}
\right] \leq \frac{1}{2} + \text{negl}(\lambda),
\]
where oracle \( O_{sk_R} \), on the \( i \)-th query \( (vk^{(i)}, (\text{psig}^{(i)}, \text{nonce}^{(i)})) \), outputs \( \text{Obtain}(sk_R, vk^{(i)}, (\text{psig}^{(i)}, \text{nonce}^{(i)})) \). That is, \( O_{sk_R} \) provides \( A \) oracle access to the \( \text{Obtain} \) algorithm w.r.t. \( sk_R \). We say that \( A \) is an admissible adversary iff:

- \( \sigma_0, \sigma_1 \neq \perp \) (i.e., \( \text{Obtain} \) algorithm does not abort), and
- \( \text{nonce}_0 \neq \text{nonce}^{(i)} \) and \( \text{nonce}_1 \neq \text{nonce}^{(i)} \) for all \( i \). (That is, \( A \) cannot make an \( \text{Obtain} \) query with nonce value to be either of the challenge nonce values.)

### 8.2 Construction

The NIBS scheme from Section 6 can be upgraded to satisfy nonce blindness with a slight modification. As hinted at in the technical overview, the main idea is to “smudge-out” the presignature information from the signed message by using a large noise that was committed in the receiver’s verification key. We elaborate on this below.

Let parameters \( n, m, \varsigma, \beta = \varsigma \sqrt{m} \) and prime \( q \) be such that the randomized one-more ISIS instance \( \text{rOM-ISIS}_{q,n,2m,\varsigma,\beta}' \) is hard for \( \beta' = (2^{\lambda/2} + 2\sqrt{2} + 2) \cdot \beta \). These parameters must satisfy the constraints in (1).

#### Tools required.

The construction relies on a secure hash function \( H \) (modeled as a random oracle), a public key encryption scheme \( \text{PKE} = (\text{KeyGen}, \text{Enc}, \text{Dec}) \), a lattice trapdoor \( \text{bLT} = (\text{TrapGen}, \text{SamplePre}) \), and a NIZK scheme \( \text{NIZK} = (\text{Setup}, \text{Prove}, \text{Verify}) \) for the following language:

**Language \( \mathcal{L}_4 \)**

**Instance:** Each instance \( x \) is interpreted as a matrix \( C \) with a secret trapdoor, random matrices \( A \) and \( B \), \( \text{PKE} \) public key \( \text{pke.pk} \), ciphertext \( ct \), vector \( w \) and random string \( \delta \).

**Witness:** Witness \( \omega \) consists of a secret vector \( x \), nonce vector \( y \), presignature vector \( z \), a bit \( \theta \), and randomness \( r \).

**Membership:** \( \omega \) is a valid witness for \( x \) if the following are satisfied:

- \( \|z\| \leq \sqrt{2} \beta \) and \( \|x\| \leq (1 + 2^{(\lambda/2-1)}) \cdot \beta/m \) and \( y \in \{\pm 1\}^{2m} \).
- \( ct \) is an encryption of \( x|y|z|\theta \) using randomness \( r \), s.t. \( ct = \text{PKE.Enc}(\text{pke.pk}, x|y|z|\theta; r) \)
- \( C \cdot z + B \cdot y = A \cdot \begin{bmatrix} x \\ \theta \end{bmatrix} + H(\delta), (1 - 2\theta) \cdot w = z_\perp - x_\perp \), and \( \theta(1 - \theta) = 0 \).

Below we describe our non-interactive blind signature scheme.

\[
\text{Setup}(1^\lambda, n, m, q, \varsigma, H) \rightarrow \text{pp}. \quad \text{Instead of a matrix in } \mathbb{Z}_q^{n \times 2m}, \text{it samples the matrix } A \leftarrow \mathbb{Z}_q^{n \times (2m+1)}.
\]

Then, acting exactly as the \( \text{Setup} \) algorithm for Construction 6.1, it outputs the public parameters as \( \text{pp} := (A, \text{pke.pk}, \text{nizk.crs}, n, m, q, \varsigma, H) \).

\[
\text{KeyGen}_S(\text{pp}) \rightarrow (\text{sk}, \text{vk}). \quad \text{Acting exactly as the } \text{KeyGen}_S \text{ algorithm for Construction 6.1, it outputs signer’s secret key and verification key as } \text{sk} := T_C \text{ and } \text{vk} := C \text{ respectively.}
\]
KeyGen \( R \) \((\text{pp}) \to (\text{sk}_R, \text{pk}_R) \). It samples the two halves of \( x \in \mathbb{Z}_q^{2m} \) as \( x^\top \leftarrow \mathcal{D}_{\mathbb{Z}_q^{2m}}, \) and \( x_\bot \leftarrow \left[ -\frac{2^{(\lambda/2-1)}}{m\sqrt{m}} \beta, \frac{2^{(\lambda/2-1)}}{m\sqrt{m}} \beta \right]^m, \) \( \delta \leftarrow \{0,1\}^\lambda \) as well as a bit \( \theta \leftarrow \{0,1\} \). Using these, where it previously computed \( t \) as \( A \cdot x + H(\delta) \), it computes \( t = A \cdot \left[ x / \theta \right] + H(\delta) \) and outputs user’s secret key and public key as \( \text{sk}_R \leftarrow (x, \delta, \theta) \) and \( \text{pk}_R \leftarrow t \) respectively.

\[ \text{Issue}(\text{sk}, \text{pk}_R) \to (\text{psig}, \text{nonce}) \text{.} \] Identically to the Issue algorithm of Construction 6.1, it generates \( z \leftarrow \text{bLT}\).SamplePre\((C, T_C, t - B \cdot y, \varsigma) \) for receiver’s public key given by \( t \), and outputs the presignature \( \text{psig} := z \), \( \text{nonce} := y \).

\[ \text{Obtain}(\text{sk}_R, \text{vk}, \text{psig}, \text{nonce}) \to (\mu, \sigma) \text{.} \] It parses \( \text{sk}_R \) as \( (x, \theta, \delta) \). Then, for \( C := \text{vk}, \ z := \text{psig}, \) and \( y := \text{nonce} \), it checks if \( C \cdot z + B \cdot y = A \cdot \left[ x / \theta \right] + H(\delta) \) and \( \|z\| \leq \sqrt{2\beta} \) and \( y \in \{\pm 1\}^{2m} \). If any check fails it aborts and outputs \( \bot \).

Otherwise, it creates the ciphertext \( ct \leftarrow \text{PKE}\).Enc\((\text{pke}.\text{pk}, x||y||z||\theta||r) \) using uniformly sampled randomness \( r \leftarrow \{0,1\}^\lambda \). The algorithm then, where in Construction 6.1 it set \( w \) as \( A \cdot \left[ x / \theta \right] \), it now sets \( w = (1 - 2\theta)(z_\bot - x_\bot) \). It generates NIZK proof \( \pi \) as \( \pi \leftarrow \text{NIZK}.\text{Prove} \left( \text{nizk}.\text{crs}, x := (C, A, B, \text{pke}.\text{pk}, ct, w, \delta), \omega := (x, y, z, \theta, r) \right) \). Finally, it outputs message \( \mu := (w, \delta) \) and signature \( \sigma := (\pi, ct) \).

\[ \text{Verify}(\text{vk}, \mu, \sigma) \to \{0,1\} \text{.} \] It parses \( \mu \) as \( (w, \delta) \) and \( \sigma \) as \( (\pi, ct) \). The verification algorithm accepts and outputs \( 1 \) if and only if \( \text{NIZK}.\text{Verify} \left( \text{nizk}.\text{crs}, x := (C, A, B, \text{pke}.\text{pk}, ct, w, \delta), \pi \right) = 1 \). Otherwise, it outputs \( 0 \).

**Efficiency.** We estimate that the resulting proof size for an instantiation similar to the one given in Section 6.4 is about a factor of \( \lambda \) larger. A concrete analysis is left for the full version. We also believe that it may be possible to achieve smudging bounds using a similar technique as applied in [BdMW16] although we do not attempt to do so here, leaving it instead as a possible future direction.

**Fully blind TNIBS.** We remark that a fully blind TNIBS protocol can be obtained from Construction 8.2 in the same way as the receiver blind TNIBS construction 7.2 was obtained from NIBS construction 6.1. Concretely, it requires a second hash function, \( H_2 \) that the signer’s Issue algorithm uses to set \( t = \text{pk}_R + H_2(\tau) \) given tag \( \tau \) as input. The signer then sends the tag along with the presignature and the nonce to the receiver, whose Obtain algorithm must now also include the tag \( \tau \) in the NIZK instance when producing the proof \( \pi \).

**Completeness.** As before, we have from the correctness of \( \text{bLT}.\text{SamplePre} \) that \( z \) is a presignature on \( A \cdot \left[ x / \theta \right] + H(\delta) \) such that \( \|z\| \leq \sqrt{2\beta} \). Furthermore, we have by construction that \( w = (1 - 2\theta)(z_\bot - x_\bot) \), \( ct \) is an encryption of \( (x||y||\theta) \) under public key \( \text{pke}.\text{pk} \) and randomness \( r \), and that
\[ \|x\| \leq (1 + 2^{(\lambda/2 - 1)}) \cdot \beta/m. \] Therefore, \( \pi \) is a valid NIZK proof for the language \( \mathcal{L}_4 \) described above. It follows from the completeness of the underlying NIZK that Construction 8.2 is complete.

**Reusability.** Notice that for \( b \in \{0, 1\} \), if a signer issues presignature-nonce pairs \((z_b, y_b)\) to a given receiver with secret vector \( x \) and secret bit \( \theta \) then,

\[
\Pr[\mu_0 = \mu_1] = \Pr[(w_0, \delta) = (w_1, \delta)] = \Pr[(1 - 2\theta)(z_{0, \perp} - z_{1, \perp}) = 0].
\]

Since, \( \theta \in \{0, 1\} \), this is simply the probability that \( z_{0, \perp} = z_{1, \perp} \), which is negligible in \( \lambda \) by construction. Thus the construction is reusable.

### 8.3 Nonce blindness

We begin by proving nonce blindness of this construction.

**Theorem 8.3.** Assume that NIZK proof system NIZK satisfies zero-knowledge property, and public key encryption scheme PKE is IND-CPA secure, then the NIBS construction 8.2 is nonce blind.

**Proof.** We have the following hybrids:

**Hybrid** 0. This corresponds to the real experiment.

1. Challenger samples \( A \leftarrow \$ \mathbb{Z}_q^{n \times (2m + 1)} \), \( B \leftarrow \$ \mathbb{Z}_q^{n \times 2m} \) uniformly at random, \( \text{pke.pk} \leftarrow \$ \text{PKE.KeyGen}(1^\lambda) \), and \( \text{nizk.crs} \leftarrow \$ \text{NIZK.Setup}(1^\lambda) \). Challenger then sets pp as \((A, \text{pke.pk}, \text{nizk.crs})\) and sends pp to the adversary.

2. Next, challenger samples \((\text{sk}_R, \text{pk}_R) \leftarrow \text{KeyGen}_R(pp)\) and sends \( \text{pk}_R \) to \( \mathcal{A} \).

3. The adversary sends a matrix \( C \in \mathbb{Z}_q^{n \times 2m} \) as the verification key, and issues two presignatures and nonces \((\text{psig}_0, \text{nonce}_0)\) and \((\text{psig}_1, \text{nonce}_1)\).

4. The challenger then generates message and signature pairs as follows:

   (a) For \( b \in \{0, 1\} \), challenger sets \( z_b \) as \( \text{psig}_b \), \( y_b \) as \( \text{nonce}_b \), and parses \( \text{sk}_R \) as \((x, \delta)\). Then it checks whether \( C \cdot z_b + B \cdot y_b = A \cdot \left[ x \right]_{\theta} + H(\delta) \) and \( \|z_b\| \leq \sqrt{2}\beta \) and \( y_b \in \{\pm 1\}^{2m} \).

   If the any of the checks fails, challenger outputs \( \bot \) and aborts.

   (b) Otherwise, for \( b \in \{0, 1\} \), challenger generates \( ct_b \leftarrow \text{PKE.Enc}(\text{pke.pk}, x\|y_b\|z_b\|\theta; r_b) \) with freshly sampled randomness \( r_0 \) and \( r_1 \).

   (c) For \( b \in \{0, 1\} \), it computes \( w_b = (1 - 2\theta)(z_{b, \perp} - x_{\perp}) \), sets statement \( x_b \) as \((C, A, B, \text{pke.pk}, ct_b, w_b, \delta)\), witness \( \omega_b \) as \((x, y_b, z_b, \theta, r_b)\), and generates NIZK proof \( \pi_b \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x_b, \omega_b) \).

   (d) Next, challenger sets message \( \mu_b \) as \((w_b, \delta)\) and signature \( \sigma_b \) as \((\pi_b, ct_b)\). Finally, challenger samples \( \hat{b} \leftarrow \$ \{0, 1\} \) uniformly at random and sends \((\mu_{\hat{b}}, \sigma_{\hat{b}}, \mu_{1-\hat{b}}, \sigma_{1-\hat{b}})\) to the adversary.

5. Adversary sends a bit \( b' \) and wins if \( \hat{b} = b' \).
Hybrid_1 This is the same as Hybrid_0, except that instead of generating the NIZK proofs $\pi_0$ and $\pi_1$, the challenger simulates it without any witnesses.

4.(c) For $b \in \{0, 1\}$, it computes $w_b = (1 - 2\theta)(z_{b, \perp} - x_\perp)$ and sets statement $x_b$ as $(C, A, B, \text{pke.pk}, c_t, w_b, \delta)$. Without setting any witnesses, challenger generates $\pi_b$ using NIZK simulator.

Hybrid_2 This is the same as Hybrid_1, except that for $b \in \{0, 1\}$, instead of generating $c_t \leftarrow \text{PKE.Enc}(\text{pke.pk}, x || y_b || z_b || \theta; r_b)$, challenger sets $c_t$ as $\text{PKE.Enc}(\text{pke.pk}, 0; r_b)$.

4.(b) Challenger generates $c_t \leftarrow \text{PKE.Enc}(\text{pke.pk}, 0; r_0)$ and $c_t' \leftarrow \text{PKE.Enc}(\text{pke.pk}, 0; r_1)$ with freshly sampled randomness $r_0$ and $r_1$.

Hybrid_3 This the same as Hybrid_2, except that the challenger samples $pk_R$ uniformly at random.

2. When generating $pk_R$, instead of setting it as $A \cdot \left[ \begin{array}{c} x \\ \theta \end{array} \right] + H(\delta)$, the challenger sets it as a uniformly sampled vector, such that $pk_R \leftarrow \mathbb{Z}_q^n$.

Let $\text{Adv}_{\Lambda}^j$ denote the security advantage of an adversary $A$ in the NIBS blindness game in Hybrid$^j$.

**Lemma 8.4.** Assuming zero-knowledge property of NIZK, then for all PPT adversaries $A$, $|\text{Adv}_{\Lambda}^0 - \text{Adv}_{\Lambda}^j| \leq \text{negl}(\lambda)$.

**Proof.** We set the following intermediate step for the proof. In the intermediate step, everything remains unchanged in Hybrid_0 and Hybrid_1, except for the following:

4.(c) For $b \in \{0, 1\}$, it computes $w_b = (1 - 2\theta)(z_{b, \perp} - x_\perp)$ and sets statement $x_b$ as $(C, A, B, \text{pke.pk}, c_t, w_b, \delta)$. Challenger only sets witness $\omega_1$ as $(x, y_1, z_1, \theta, r_1)$ without assigning the witness for statement $x_0$. Challenger generates $\pi_0$ using NIZK simulator and $\pi_1 \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x_1, \omega_1)$.

Define adversary $A$’s advantage on the intermediate step as $\text{Adv}_{A}^{0,5}$. Suppose there exists a PPT attacker $A$ such that $|\text{Adv}_A^0 - \text{Adv}_A^{0,5}| = \epsilon(\lambda)$, we design a reduction algorithm $B$ that breaks the zero-knowledge property of NIZK with advantage $\epsilon(\lambda)$.

Challenger starts by outputting nizk.crs. Reduction algorithm $B$ samples $A \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{2m+1}$, $B \in \mathbb{Z}_q^n \times \mathbb{Z}_q^{2m}$ and pke.pk, and simply outputs $pp = (A, B, \text{pke.pk, nizk.crs})$. $A$ then outputs its verification key $vk$. Next, $B$ samples $(sk_R, pk_R) \leftarrow \text{KeyGen}_R(pp)$ and outputs $pk_R$. Attacker $A$ then sends two presignatures and nonces $(\text{psig}_0, \text{nonce}_0)$ and $(\text{psig}_1, \text{nonce}_1)$. Subsequently, $B$ simulates the Obtain algorithm: for $b \in \{0, 1\}$, $B$ sets $z_b$ as $\text{psig}_b$, $y_b$ as $\text{nonce}_b$, parses $sk_R$ as $(x, \delta)$, and checks if $C \cdot z_b + B \cdot y_b = A \cdot \left[ \begin{array}{c} x \\ \theta \end{array} \right] + H(\delta)$ and $\|z_b\| \leq \sqrt{2}\beta$ and $y_b \in \{\pm 1\}^{2m}$. If any of these checks fail, $B$ outputs $\perp$ and aborts. Otherwise, for $b \in \{0, 1\}$, it continues to generate $c_t \leftarrow \text{PKE.Enc}(\text{pke.pk}, x || y_b || z_b || \theta; r_b)$ with freshly sampled randomness $r_b$. $B$ then computes $w_b = (1 - 2\theta)(z_{b, \perp} - x_\perp)$ and sets statement $x_b$ as $(C, A, B, \text{pke.pk}, c_t, w_b, \delta)$, witness $\omega_b$ as $(x, y_b, z_b, \theta, r_b)$. Next, $B$ queries the NIZK challenger with $(x_0, \omega_0)$ and challenger returns proof $\pi_0$. $B$ generates $\pi_1 \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x_1, \omega_1)$ by
itself. Next, $B$ sets message $\mu_b$ as $(w_b, \delta)$ and signature $\sigma_b$ as $(\pi_b, c_t)$ for $b \in \{0, 1\}$. $B$ then samples $\check{b} \leftarrow \{0, 1\}$ uniformly at random. Finally, $B$ sends $(\mu_{\check{b}}, \sigma_{\check{b}}; \mu_{1-\check{b}}, \sigma_{1-\check{b}})$ to $A$. $A$ outputs its guess $\check{b}'$. If $\check{b}' = \check{b}$, $B$ sends $0$ as its guess (i.e., $\pi_0$ is a simulated proof), otherwise it sends $1$ as its guess (i.e., $\pi_0$ is honestly generated from NIZK.Prove(nizk.crs, $x_0, \omega_0$)).

Notice, if the NIZK challenger honestly generates proof $\pi_0$ using NIZK.Prove(nizk.crs, $x_0, \omega_0$), then $B$ perfectly simulates the experiment of Hybrid$_0$ for adversary $A$. Otherwise it simulates the intermediate step for $A$. As a result, if $|Adv^0_A - Adv^{0.5}_A|$ is non-negligible, then $B$ breaks the zero-knowledge property of NIZK with non-negligible advantage. Similarly, $|Adv^{0.5}_A - Adv^{1.5}_A|$ is also negligible, and thus $|Adv^0_A - Adv^{1.5}_A| \leq \text{negl}(\lambda)$.

**Lemma 8.5.** If public key encryption scheme $PKE$ is a secure public key encryption scheme, then for all PPT adversaries $A$, $|Adv^1_A - Adv^{1.5}_A| \leq \text{negl}(\lambda)$.

**Proof.** Consider the following intermediate step for the proof. In the intermediate step, everything remains unchanged in Hybrid$_1$ and Hybrid$_{2}$, except for the following:

4.(b) $B$ generates $ct_0 \leftarrow PKE.Enc(pke.pk, 0; r_0)$ and $ct_1 \leftarrow PKE.Enc(pke.pk, x||y_1||z_1||\theta; r_1)$ with freshly sampled randomness $r_0$ and $r_1$.

Define adversary $A$'s advantage on the intermediate step as $Adv^{1.5}_A$. Suppose there exists a PPT attacker $A$ such that $|Adv^1_A - Adv^{1.5}_A| = \epsilon(\lambda)$, we design a reduction algorithm $B$ that breaks the security of $PKE$ with advantage $\epsilon(\lambda)$.

Challenger first samples and outputs $pke.pk$ from $PKE.KeyGen$. Reduction algorithm $B$ then samples $A \in \mathbb{Z}_q^{n \times (2\lambda + 1)}$ and $\text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$, and outputs $pp$ as $(A, pke.pk, \text{nizk.crs})$. Next, $A$ outputs its verification key $vk$. $B$ randomly samples a key pair $(sk_R, pk_R) \leftarrow \text{KeyGen}_R(1^\lambda, pp)$ and sends $pk_R$ to $A$. Adversary $A$ then sends two presignatures and nonces $(\text{psig}_0, \text{nonce}_0)$ and $(\text{psig}_1, \text{nonce}_1)$. Subsequently, for $b \in \{0, 1\}$, $B$ sets $z_b$ as $\text{psig}_b \cdot y_b$ as nonces, and, after parsing $sk_R$ as $(x, \delta)$, it checks if $C \cdot z_b + B \cdot y_b = A \cdot \frac{x}{\theta} + H(\delta_b)$ and if $\|z_b\| \leq \sqrt{2} \beta$ for $b \in \{0, 1\}$. If any of the checks fails, $B$ outputs $\bot$ and aborts. Otherwise, it queries the challenger with the all zeros string $0$ and string $x||y_0||z_0||\theta$. Challenger returns ciphertext $ct_0$ (which is an encryption of either $0$ or $x||y_0||z_0||\theta$). Then $B$ generates $ct_1 \leftarrow PKE.Enc(pke.pk, x||y_1||z_1||\theta; r_1)$ with freshly sampled randomness $r$. It computes $w_b = (1 - 2\theta)(z_{b,\bot} - x_{,\bot})$, sets statement $x_b$ as $(C, A, B, pke.pk, ct_0, w_b, \delta)$, and generates $\pi_b$ as a simulated proof for each $b \in \{0, 1\}$. Next, $B$ sets message $\mu_b$ as $(w_b, \delta)$ and signature $\sigma_b$ as $(\pi_b, ct_b)$ for $b \in \{0, 1\}$. Finally, $B$ samples $\check{b} \leftarrow \{0, 1\}$ uniformly at random and sends $(\mu_{\check{b}}, \sigma_{\check{b}}, \mu_{1-\check{b}}, \sigma_{1-\check{b}})$ to $A$. $A$ outputs its guess $\check{b}'$. If $\check{b}' = \check{b}$, $B$ sends $0$ as its guess (i.e., $ct_0$ is an encryption of $0$), otherwise it sends $1$ as its guess (i.e., $ct_0$ is an encryption of string $x||y_0||z_0||\theta$).

Note that if $ct_0$ is an encryption of string $x||y_0||z_0||\theta$, then $B$ perfectly simulates the experiment of Hybrid$_1$ for adversary $A$. Otherwise it simulates the intermediate step. As a result, if $|Adv^{1.5}_A - Adv^{2}_A|$ is non-negligible, then $B$ breaks the security of public key encryption scheme $PKE$ with non-negligible advantage. Similarly, $|Adv^{1.5}_A - Adv^{3}_A| \leq \text{negl}(\lambda)$ and thus $|Adv^1_A - Adv^{3}_A| \leq \text{negl}(\lambda)$.

**Lemma 8.6.** For all PPT adversaries $A$, $|Adv^2_A - Adv^{3}_A| \leq \text{negl}(\lambda)$.
Proof. Since $x^\top \leftarrow \mathcal{D}_{Z_q^{m \times n}}$, $H_{\infty} (x_\perp) \geq m > n \log q + \omega(\log n)$, and we have by the Leftover Hash Lemma (Corollary 3.8) that the following distributions are statistically indistinguishable:

$$\{(A_L, A_L \cdot x^\top) : A_L \leftarrow \mathbb{Z}^{n \times m}, x^\top \leftarrow \mathcal{D}_{Z_q^{m \times n}}\}$$

$$\approx_s \{(A_L, \rho) : A_L \leftarrow \mathbb{Z}^{n \times m}, \rho \leftarrow \mathbb{Z}_q^n\}$$

Thus the vector $A_L \cdot [x] / \theta + H(\delta) = A_L \cdot x^\top + A_R \cdot x_\perp + \theta a + H(\delta_b)$ is statistically indistinguishable from a random vector in $\mathbb{Z}_q^n$.

Lemma 8.7. For all PPT adversaries $A$, $\text{Adv}_A^2 \leq 1/2 + \text{negl}(\lambda)$.

Proof. We will argue the statistical indistinguishability. To see this, first observe that in step 4. (a), the challenger always checks for each $b \in \{0, 1\}$ whether $\|z_b\| \leq \sqrt{2} \beta$ and aborts if it is not. So at step 4. (c), it must hold that $\|z_b\| \leq \sqrt{2} \beta$ and consequently every entry in $z_b \in \mathbb{Z}^{2m}$ is in the range $[-\beta, \beta]$. Thus every entry in $z_{b, \perp} \in \mathbb{Z}^m$ is in the range $[-\beta, \beta]$. Recall further that, by construction, the integer vector $x_\perp$ is uniformly sampled over $\left[\frac{2^{\lambda/2}}{m^{\sqrt{m}}} \beta, \frac{2^{\lambda/2}}{m^{\sqrt{m}}} \beta\right]^m$. Therefore,

$$\frac{\beta}{\frac{2^{\lambda/2}}{m^{\sqrt{m}}} \beta} = \frac{m^{\sqrt{m}}}{2^{\lambda/2}}$$

is at most negligible, and the smudging lemma (Lemma 3.9) applies. Then, for $x_\perp \leftarrow \left[\frac{2^{\lambda/2}}{m^{\sqrt{m}}} \beta, \frac{2^{\lambda/2}}{m^{\sqrt{m}}} \beta\right]^m$, $z_{b, \perp}, z_{1-b, \perp} \in [-\beta, \beta]^m, \theta \leftarrow \{0, 1\}$, we have:

$$\{(1 - 2\theta)(z_{0, \perp} - x_\perp), (1 - 2\theta)(z_{1, \perp} - x_\perp), z_{0, \perp}, z_{1, \perp}\}$$

$$\approx_s \{(1 - 2\theta)(\frac{z_{0, \perp} - z_{1, \perp}}{2} - x_\perp), (1 - 2\theta)(\frac{z_{1, \perp} - z_{0, \perp}}{2} - x_\perp), z_{0, \perp}, z_{1, \perp}\}.$$
The above equations imply that $w_0$ is statistically indistinguishable from $w_1$ in Hybrid 3. Meanwhile, by definition of Hybrid 3, $\pi_0$ and $\pi_1$, $ct_0$ and $ct_1$ are also indistinguishable. Thus our lemma follows.

From the lemmas above, it follows that NIBS construction 8.2 satisfies nonce blindness assuming that NIZK satisfies zero-knowledge property and PKE is secure.

8.4 Receiver blindness

Next, we show that this construction also satisfies receiver blindness.

**Theorem 8.8.** Assume that NIZK proof system NIZK satisfies zero-knowledge property and public key encryption scheme PKE is IND-CPA secure, then the NIBS construction 8.2 is receiver blind.

**Proof.** We prove the theorem through the following hybrids:

1. **Hybrid₀** This corresponds to the real experiment.
   
   a. Challenger samples $A \leftarrow \mathcal{Z}_q^{n \times (2^m + 1)}$, $B \leftarrow \mathcal{Z}_q^{n \times 2^m}$ uniformly at random, $\text{pke.pk} \leftarrow \mathcal{Z}_q^{n \times 2^m}$, challenger sets $\text{KeyGen}(1^\lambda)$, and $\text{nizk.crs} \leftarrow \mathcal{Z}_q^{n \times 2^m}$. Challenger then sets $pp$ as $(C, A, B, \text{pke.pk}, \text{nizk.crs})$ and sends $pp$ to the adversary.
   
   b. Next, challenger samples $(sk_{R_b}, pk_{R_b}) \leftarrow \text{KeyGen}_R(pp)$, for $b \in \{0, 1\}$. The challenger sends $pk_{R_0}$ and $pk_{R_1}$ to $A$.
   
   c. The adversary sends a matrix $C \in \mathcal{Z}_q^{n \times 2^m}$ as the verification key, and issues two presignatures $\text{psig}_0$ and $\text{psig}_1$.
   
   d. The challenger then generates message and signature pairs:
      
      (a) For $b \in \{0, 1\}$, challenger sets $z_b$ as $\text{psig}_{b_0}$, $y_b$ as $\text{nonce}_{b_0}$, parses $sk_{R_b}$ as $(x_b, \delta_b)$, and then checks whether $C \cdot z_b + B \cdot y_b = A \cdot \lfloor x_b \rfloor_{\theta_b} + H(\delta_b)$ and $\|z_b\| \leq \sqrt{2}\beta$ and $y_b \in \{\pm 1\}^{2^m}$. If the any of the checks fails, challenger outputs $\bot$ and aborts.
      
      (b) Otherwise, for $b \in \{0, 1\}$, challenger continues to generate $ct_b \leftarrow \text{PKE.Enc}(\text{pke.pk}, x_b || z_b || y_b || \theta_b || r_b)$ with freshly sampled randomness $r_0$ and $r_1$.
      
      (c) For $b \in \{0, 1\}$, it computes $w_b = (1 - 2\theta)(z_b,_{\bot} - x_b,_{\bot})$, sets statement $x_b$ as $(C, A, B, \text{pke.pk}, ct_b, w_b, \delta_b)$, witness $\omega_b$ as $(x_b, y_b, z_b, \theta_b, r_b)$, and generates NIZK proof $\pi_b \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x_b, \omega_b)$.
      
      (d) Next, challenger sets message $\mu_b$ as $(w_b, \delta_b)$ and signature $\sigma_b$ as $(\pi_b, ct_b)$. Finally, challenger samples $\hat{b} \leftarrow \mathcal{Z}_q^{n \times 2^m}$ uniformly at random and sends $(\mu_{\hat{b}}, \sigma_{\hat{b}}, \mu_{\hat{b} \cdot b}, \sigma_{\hat{b} \cdot b})$ to the adversary.
   
   5. **Adversary sends a bit $b\,'$ and wins if $\hat{b} = b\,'$.**
Hybrid$_1$ This is the same as Hybrid$_0$, except that instead of generating the NIZK proofs $\pi_0$ and $\pi_1$, the challenger simulates it without any witnesses.

4.(c) For $b \in \{0, 1\}$, it computes $w_b = (1 - 2\theta)(z_{b, \perp} - x_{b, \perp})$, sets statement $x_b$ as $(C, A, B, \text{pke}.pk, \text{ct}_b, w_b, \delta_b)$. Without setting any witnesses, challenger generates $\pi_b$ using NIZK simulator.

Hybrid$_2$ This is the same as Hybrid$_1$, except that for $b \in \{0, 1\}$, instead of generating $\text{ct}_b \leftarrow \text{PKE.Enc}(\text{pke}.pk, x_b || y_b || z_b || \theta_b; r_b)$, challenger sets $\text{ct}_b$ as $\text{PKE.Enc}(\text{pke}.pk, 0; r_b)$, ie.,

4.(b) Challenger generates $\text{ct}_0 \leftarrow \text{PKE.Enc}(\text{pke}.pk, 0; r_0)$ and $\text{ct}_1 \leftarrow \text{PKE.Enc}(\text{pke}.pk, 0; r_1)$ with freshly sampled randomness $r_0$ and $r_1$.

Hybrid$_3$ This the the same as Hybrid$_2$, except that the challenger samples $\text{pk}_{R_0}$ and $\text{pk}_{R_1}$ uniformly at random.

2. When generating $\text{pk}_{R_b}$ for $b \in \{0, 1\}$, instead of setting it as $A \cdot \begin{bmatrix} x_b \\ \theta_b \end{bmatrix} + \text{H}(\delta_b)$, the challenger sets it as a uniformly sampled vector, such that $\text{pk}_{R_b} \leftarrow \mathbb{Z}_q^n$.

Now, let Adv$_A^j$ denote the security advantage of an adversary $A$ in the NIBS blindness game in Hybrid$_j$. Then, the following must hold:

**Lemma 8.9.** Assuming zero-knowledge property of NIZK, then for all PPT adversaries $A$, $|\text{Adv}_A^0 - \text{Adv}_A^1| \leq \text{negl}(\lambda)$.

*Proof.* (omitted) follows by extending the proof of Lemma 6.9 to Construction 8.2. \hfill \Box

**Lemma 8.10.** If public key encryption scheme PKE is a secure public key encryption scheme, then for all PPT adversaries $A$, $|\text{Adv}_A^3 - \text{Adv}_A^0| \leq \text{negl}(\lambda)$.

*Proof.* (omitted) follows by extending the proof of Lemma 6.10 to Construction 8.2. \hfill \Box

**Lemma 8.11.** For all PPT adversaries $A$, $|\text{Adv}_A^3 - \text{Adv}_A^1| \leq \text{negl}(\lambda)$.

*Proof.* (omitted) follows by the proof of Lemma 8.6 repeated each $b \in \{0, 1\}$. \hfill \Box

Finally, will once more leverage the smudging lemma (Lemma 3.9) to argue that for each $b \in \{0, 1\}$ (the distributions over) $w_b = (1 - 2\theta)(z_{b, \perp} - x_{b, \perp})$ is statistically indistinguishable from uniform. As, in step 4. (a), the challenger always checks for each $b \in \{0, 1\}$ whether $\|z_b\| \leq \sqrt{2}\beta$ and aborts if it is not, it must hold that $\|z_b\| \leq \sqrt{2}\beta$ after that step, and consequently every entry in $z_b \in \mathbb{Z}_q^{2m}$ is in the range $[-\beta, \beta]$. Further each $x_{b, \perp}$ is uniformly sampled over $\begin{bmatrix} -\frac{2^{(\lambda/2 - 1)}}{m \sqrt{m}} \beta, \frac{2^{(\lambda/2 - 1)}}{m \sqrt{m}} \beta \end{bmatrix}$ by construction. Therefore we have that the ratio

$$\frac{\beta}{\frac{2^{(\lambda/2 - 1)}}{m \sqrt{m}} \beta} = \frac{m \sqrt{m}}{2^{(\lambda/2 - 1)}}$$
is negligible in $\lambda$, so that the lemma applies and $z_{b,\perp} - x_{b,\perp}$ is statistically indistinguishable from $x_{b,\perp}$. Since $x_{b,\perp}$ and $\theta_b$ are uniformly chosen, the claim holds.

Therefore, any adversary has negligible advantage under $\text{Hybrid}_3$. From the above lemmas, it follows that NIBS construction 8.2 satisfies receiver blindness assuming that NIZK satisfies zero-knowledge property and PKE is secure.

Lastly we give the following theorem concerning the one-more unforgeability of this construction.

8.5 One-more unforgeability

Theorem 8.12. Assume that NIZK proof system NIZK satisfies soundness, lattice trapdoor $bLT$ satisfies well-distributedness, and the $rOM$-ISIS assumption holds, then our construction 8.2 is one-more unforgeable.

Proof. Consider the following hybrids:

Hybrid$_0$ This is the actual one-more unforgeability game for NIBS:

1. Challenger samples $A \leftarrow \mathbb{Z}_q^{n \times (2^m + 1)}$, $B \leftarrow \mathbb{Z}_q^{n \times 2^m}$ uniformly at random, $\text{pke.pk} \leftarrow \text{PKE.KeyGen}(1^\lambda)$, $\text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$ and sets $\text{pp}$ as $(A, B, \text{pke.pk}, \text{nizk.crs})$. Next, it samples $T_C, C \leftarrow \text{bLT.TrapGen}(1^\lambda, n, 2^m, q)$ and sets $\text{sk} := T_C, \text{vk} := C$. Challenger sends $\text{pp}$ and $\text{vk}$ to the adversary.

2. Adversary chooses $\text{pk}_R$ and requests a presignature, to which the challenger replies with $(\text{psig}, \text{nonce}) := \text{Issue}(\text{sk}, \text{pk}_R)$. The adversary repeats this step a total of $\ell$ times.

3. Adversary outputs $k$ message and signature pairs $((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k))$. Adversary wins if $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$, $\text{Verify}(\text{vk}, \mu_i, \sigma_i) = 1$ for all $i \in [k]$, and $k = \ell + 1$.

Hybrid$_1$ This is the same as Hybrid$_0$ aside from one key difference that instead of uniformly sampling the matrix $A$, it is chosen by first sampling a matrix $R \leftarrow \{0, 1\}^{2^m \times (2^m + 1)}$ and then setting $A = C \cdot R$.

1. Challenger samples $R \leftarrow \{0, 1\}^{2^m \times (2^m + 1)}$ uniformly at random and sets $A = C \cdot R$. Challenger then samples $B \leftarrow \mathbb{Z}_q^{n \times 2^m}$, $\text{pke.pk} \leftarrow \text{PKE.Setup}(1^\lambda)$, $\text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$, and sets $\text{pp}$ as $(A, B, \text{pke.pk}, \text{nizk.crs})$. Next, it samples $T_C, C \leftarrow \text{bLT.TrapGen}(1^\lambda, n, 2^m, q)$ and sets $\text{sk} := T_C, \text{vk} := C$. Challenger sends $\text{pp}$ and $\text{vk}$ to the adversary.

Let $\text{Adv}^j_A$ denote the security advantage of an adversary $A$ in the NIBS one-more unforgeability game in Hybrid$_j$. Then, the following must hold:

Lemma 8.13. For all PPT adversaries $A$, $|\text{Adv}^0_A - \text{Adv}^1_A| \leq \text{negl}(\lambda)$. 57
Proof. Omitted as it is identical to the proof of Lemma 6.2 but with matrices \( R \in \{0,1\}^{2m \times (2m+1)} \) and \( A \in \mathbb{Z}_q^{n \times (2m+1)} \), and the divergence over the distribution of \( R \) scaled accordingly. \(\square\)

**Lemma 8.14.** Assuming that NIZK argument is sound and the rOM-ISIS assumption holds, the one-more-unforgeability adversary \( \mathcal{A} \) can have at most negligible advantage in Hybrid\(_i\).

**Proof.** Suppose there exists a PPT attacker \( \mathcal{A} \) that wins the one-more unforgeability game with non-negligible probability \( \epsilon = \epsilon(\lambda) \). It outputs \( k \) message and signature pairs \((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k)\), and wins if and only if \( \mu_i \neq \mu_j \) for \( 1 \leq i < j \leq k \). Verify\((vk, \mu_i, \sigma_i) = 1 \) for \( 1 \leq i \leq k \), and \( k > \ell \). As before, we parse inputs and signatures \((\mu_i, \sigma_i)\) from \( \mathcal{A} \)’s output as \((w_i, \delta_i), (\pi_i, ct_i)\) for all \( i \in [k] \). Let us divide the potential attacker into the following cases,

- **Type 1:** This is same as the type 1 attacker from Lemma 6.3 extended to NIZK, the NIZK for language \( \mathcal{L}_i \).
- **Type 2:** This is same as the type 2 attacker from Lemma 6.3 except for this attacker, there is at least one tuple of \((x_j, y_j, z_j, \theta_j)\) for some \( j \in [k] \), such that \( C \cdot z_j + B \cdot y_j = A \cdot \begin{bmatrix} x_j \\ \theta_j \end{bmatrix} + H(\delta_j) \) where the attacker never queries the random oracle \( H \) on \( \delta_j \).
- **Type 3:** It is exactly the same as the type 3 attacker from Lemma 6.3.

**Claim 8.15.** Assuming that NIZK satisfies soundness property, then type 1 attacker \( \mathcal{A} \) has at most negligible advantage.

**Proof.** (omitted) follows by extending the proof of Claim 6.4 to Construction 8.2. \(\square\)

**Claim 8.16.** Type 2 attacker \( \mathcal{A} \) has at most negligible advantage.

**Proof.** By the same token as the proof of Claim 6.5, \( H(\delta^*) \) is not defined by the random oracle and is completely random in the view of \( \mathcal{A} \) so that it has at most negligible advantage in finding \( \delta^* \) such that \( H(\delta^*) = C \cdot z_j - A \cdot \begin{bmatrix} x_j \\ \theta_j \end{bmatrix} \). \(\square\)

**Claim 8.17.** Assuming that the rOM-ISIS assumption holds, type 3 adversary \( \mathcal{A} \) has at most negligible advantage.

**Proof.** Assume that type 3 attacker \( \mathcal{A} \) has non-negligible advantage \( \epsilon = \epsilon(\lambda) \), we design a reduction algorithm \( B \) that breaks the randomized one-more ISIS assumption.

The rOM-ISIS challenger starts by sampling and outputting a matrix \( C, B \in \mathbb{Z}_q^{n \times 2m} \) uniformly at random. Reduction algorithm \( B \) then samples \( R \leftarrow \{0,1\}^{2m \times (2m+1)} \) and sets \( A = C \cdot R \). Next, \( B \) generates \((pke.pk, pke.sk)\) and nizk.crs. \( B \) outputs \( pp := (A, pke.pk, nizk.crs) \) and verification key \( vk := C \).

Next, \( \mathcal{A} \) is allowed to make a series of hash queries and presignature queries (in any order). On each fresh hash query that \( \mathcal{A} \) provides an input \( \delta \), \( B \) simply makes a syndrome query to the challenger. \( B \) forwards challenger’s output to \( \mathcal{A} \) as the output of \( H(\delta) \). Meanwhile, \( B \) keeps a recording of all hash queries from \( \mathcal{A} \). For each presignature query on input \( t' \) from \( \mathcal{A} \), \( B \) makes a
Thus, combining the above arguments, the lemma follows. Let

\[ C \]

\[ \text{9.1 Leveled homomorphic encryption} \]

privacy. We begin this section by defining leveled homomorphic encryption and the property of circuit privacy. To win the one-more unforgeability game, \( A \) outputs \( k \) message and signature pairs \(( (\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k) )\) such that \( \mu_i \neq \mu_j \) for \( 1 \leq i < j \leq k \), \( \text{Verify}(pp, vk, \mu_i, \sigma_i) = 1 \) for all \( i \in [k] \), and \( k = \ell + 1 \).

For all \( i \in [k] \), \( B \) parses \( \mu_i \) as \( (w_i, \delta_i) \) and \( \sigma_i \) as \( (\pi_i, ct_i) \). \( B \) obtains \( x_i \in \mathbb{Z}_q^{2m} \), \( y_i \in \{\pm 1\}^{2m} \), \( z_i \in \mathbb{Z}_q^{2m} \), and \( \theta_i \) by decrypting \( ct_i \), where \( x_i || y_i || z_i |\theta_i = \text{PKE.DEC}(\text{pke.sk}, ct_i) \). Thus for all \( i \in [k] \),

\[
H(\delta_i) = C \cdot z_i + B \cdot y_i - A \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_i} = C \cdot \left( z_i - R \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_i} \right) + B \cdot y_i.
\]

Moreover, since \( R \) is a binary matrix, \( \| z_i \| \leq \sqrt{2} \beta \), and \( \| x_i \| \leq (1 + 2^{(\lambda/2 - 1)}) \cdot \beta / m \)

\[
\left\| z_i - R \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_i} \right\| \leq (2^{\lambda/2} + 2\sqrt{2} + 2) \cdot \beta = \beta'
\]

Note that \( \mu_i \neq \mu_j \), for \( 1 \leq i < j \leq k \). There are two cases:

Case 1: For some \( 1 \leq i < j \leq k \), \( \delta_i = \delta_j \). By definition of type 4 attacker, we have \( y_i \neq y_j \).

Case 2: \( \delta_i \neq \delta_j \). Then with all but negligible probability, we have \( H(\delta_i) \neq H(\delta_j) \), which implies that \( C \cdot (z_i - R \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_i}) + B \cdot y_i \neq C \cdot (z_j - R \cdot \begin{bmatrix} x_j \end{bmatrix}_{\theta_j}) + B \cdot y_i \). Thus, either

\[
z_i - R \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_1} \neq z_j - R \cdot \begin{bmatrix} x_j \end{bmatrix}_{\theta_1}, \text{ or } y_i \neq y_j.
\]

As a result, \( B \) wins the rOM-ISIS game by sending back \( z_i - R \cdot \begin{bmatrix} x_i \end{bmatrix}_{\theta_1} \) and \( y_i \) for \( i \in [k] \). \( \square \)

Note that any successful one-more unforgeability attacker \( A \) must be one of type 1, 2, and 3. Thus, combining the above arguments, the lemma follows.

\[ \square \]

This completes the proof for one-more unforgeability of Construction 8.2.

\[ \square \]

9 NIBS from Circuit Private LHE

We begin this section by defining leveled homomorphic encryption and the property of circuit privacy.

9.1 Leveled homomorphic encryption

Let \( C_d \) denote the class of boolean valued circuits of depth \( d \). A leveled homomorphic encryption scheme \( \mathcal{LHE} \) with message space \( \{0, 1\} \) for circuit class \( \{C_d\}_{d \in \mathbb{N}} \) consists of the following polynomial time algorithms:

\[ 59 \]
Setup(1^λ, 1^d) → (sk, ek) The setup algorithm takes as input the security parameter λ, bound on circuit depth d and outputs a secret key sk and evaluation key ek.

Enc(sk, m ∈ {0, 1}) → ct The encryption algorithm takes as input a secret key sk, message m ∈ {0, 1} and outputs a ciphertext ct.

Eval(ek, C ∈ C_d, ct) → ct′ The evaluation algorithm takes as input an evaluation key ek, a circuit C ∈ C_d, a sequence of ciphertexts ct = (ct_1, . . . , ct_ℓ) for some ℓ > 0 and outputs a ciphertext ct′. Here ℓ denotes the input length of C.

Dec(sk, ct) → x The decryption algorithm takes as input a secret key sk and ciphertext ct and outputs x ∈ {0, 1} U {⊥}.

Correctness. The scheme LHE is said to be (perfectly) correct if for all security parameter λ, circuit-depth bound d, (sk, ek) ← Setup(1^λ, 1^d), circuit C ∈ C_d and messages m_1, . . . , m_ℓ ∈ {0, 1}, every ciphertext ct_i ←$ Enc(sk, m_i) where ℓ denotes input length of C, the following holds:

Pr [Dec(sk, Eval(ek, C, (ct_1, . . . , ct_ℓ))) = C(m_1, . . . , m_ℓ)] = 1.

Definition 9.1 (Circuit privacy). An LHE scheme is said to be circuit private if there exists a PPT algorithm Sim such that for every d ∈ N any circuit C ∈ C_d with input length ℓ = poly(λ), and any sequence of message bits m_1, . . . , m_ℓ ∈ {0, 1}, the following holds:

(ek, Eval(ek, C, (ct_1, . . . , ct_ℓ)))_{ct_1, . . . , ct_ℓ} ≈_c (ek, ct, ct_1, . . . , ct_ℓ)

where (sk, ek) ←$ Setup(1^λ, 1^d), ct_i ←$ Enc(sk, m_i) ∀i ∈ [ℓ], ct = Sim(ek, C(m_1, . . . , m_ℓ), ct_1, . . . , ct_ℓ).

Also recall that 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) receiver and verification keys. In this section, we will consider definitions under the knowledge of secret keys (KOSK) model [MOR01, Bol03] as define next.

9.2 Knowledge of secret keys model

In the knowledge of secret keys (KOSK) model for NIBS, we require the attacker to specify the corresponding secret keys along with the respective verification/public keys. That is, the attacker supplies the receiver’s secret key in unforgeability experiment and the signer’s signing key in blindness experiments, respectively. We point out that any NIBS scheme satisfying one-more (strong) unforgeability, (strong) receiver blindness, and (strong) nonce blindness in the KOSK model can be generically upgraded into another NIBS scheme satisfying all the aforementioned properties by using a non-interactive zero-knowledge argument of knowledge (NIZKAoK) scheme.

Definition 9.2 (strong receiver blindness in KOSK). Recall the strong receiver blindness security experiment from Definition 5.6. Consider another security experiment where A additionally provides a secret key sk^{(i)} along with its corresponding vk^{(i)} at the time of each oracle query. Moreover, A provides a secret key sk along with vk when declaring the challenge presignature-nonce pairs.
We say $A$ is admissible if $sk^{(i)}(sk)$ is a valid secret key for $vk^{(i)}(vk)$, respectively. Note that admissibility can be checked efficiently as the secret key $sk^{(i)}$ can be regarded as the random coins of the setup algorithm.

A NIBS scheme $S$ satisfies strong receiver blindness in the KOSK model if no admissible PPT adversary $A$ wins in the above adapted security experiment with non-negligible probability.

**Definition 9.3** (One-more unforgeability in KOSK). A NIBS scheme $S$ satisfies one-more unforgeability in the KOSK model if no admissible PPT adversary $A$ wins in the adapted security experiment similar to in Definition 9.2 with non-negligible probability. Briefly, the adaptation is that the adversary provides a valid secret key associated with each receiver key it outputs.

### 9.3 Construction

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

**Tools required.** The construction relies on a pseudorandom function $F$, a signature scheme $S = (S.Setup, S.Sign, S.Verify)$, a leveled homomorphic encryption scheme $LHE = (LHE.Setup, LHE.Enc, LHE.Eval, LHE.Dec)$ and a NIZKAoK proof system $NIZK = (NIZK.Setup, NIZK.Prove, NIZK.Verify)$ for the following language:

<table>
<thead>
<tr>
<th>Language $L_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong> Each instance $x$ is interpreted as an LHE evaluation key $lhe.ek$, LHE ciphertexts $ct$ and $\hat{ct}$ and randomness $r \in {0, 1}^\lambda$.</td>
</tr>
<tr>
<td><strong>Witness:</strong> Witness $\omega$ consists of a secret signing key $sk$ and randomness $\rho$.</td>
</tr>
<tr>
<td><strong>Membership:</strong> Let $C_{sk,r}$ encode the circuit $S.Sign(sk, F(r))$ for a public pseudorandom function $F$. Then $\omega$ is a valid witness for $x$ if the following is satisfied:</td>
</tr>
<tr>
<td>$\hat{ct}$ is the homomorphic evaluation of the ciphertext $ct$ over the circuit $C_{sk,r}$, ie., $\hat{ct} = LHE.Eval(lhe.ek, C_{sk,r}, ct; \rho)$.</td>
</tr>
</tbody>
</table>

Below we describe our NIBS from circuit private LHE.

**Setup**(1$^\lambda$) → pp. It runs the setup algorithms for NIZK (for language $L_3$):

$$nizk.crs \leftarrow NIZK.Setup(1^\lambda),$$

and outputs $pp = nizk.crs$.

**KeyGen**$_S$(pp, 1$^\lambda$) → (sk, vk). The signer’s setup algorithm runs the setup algorithm for the signature scheme $S$. Namely, it generates keys as $(sk, vk) \leftarrow S.Setup(1^\lambda)$.

**KeyGen**$_R$(pp, 1$^\lambda$) → (sk$_R$, pk$_R$). The receiver’s setup algorithm first samples a LHE key pair (lhe.sk, lhe.ek) $\leftarrow$ LHE.Setup(1$^\lambda$, 1$^d$), and random PRF key $K \leftarrow \{0, 1\}^\lambda$. (Here depth $d$ is defined during the issue algorithm.) Next, it encrypts $K$ as $ct \leftarrow LHE.Enc(lhe.sk, K)$. Finally, it outputs receiver’s secret key and public key as $sk_R := (lhe.sk, K)$ and $pk_R := (lhe.ek, ct)$. 

61
Obtain($sk, pk_R$) $\rightarrow$ (psig, nonce). The issue algorithm first samples a random message $r \leftarrow \{0, 1\}^\lambda$. Let $C_{sk,r}(\cdot)$ be the following circuit (with key $sk$ and message $r$ hardwired) — $C_{sk,r}(K) \stackrel{\text{def}}{=} S\cdot\text{Sign}(sk, F_K(r))$. That is, $C_{sk,r}$ runs the PRF function using the circuit input as the PRF key on message $r$, and then runs the signing algorithm to sign the output of the PRF. Let $d$ denote the depth of the circuit $C_{sk,r}$. (This is the depth we set during the setup of the LHE scheme.)

The algorithm runs the homomorphic evaluation algorithm under uniformly random $\rho \leftarrow \{0, 1\}^\lambda$, and $(lhe.ek, ct) := pk_R$, as follows:

$$
\hat{ct} \leftarrow \text{LHE.Eval}(lhe.ek, C_{sk,r}, ct; \rho),
$$

and creates a NIZK proof $\pi$ for the language $L_3$ as

$$
\pi \leftarrow \$ \text{NIZK.Prove}(nizk.crs, x = (pk_R, \hat{ct}, r), \omega = (sk, \rho))
$$

Finally, it outputs the presignature $\text{psig} := (\hat{ct}, \pi)$ and nonce nonce := $r$.

Obtain($sk_R, vk, psig, nonce$) $\rightarrow$ $(\mu, \sigma)$. The receiver parses $(\hat{ct}, \pi) :=$ psig and runs the NIZK verifier as $\text{NIZK.Verify}(nizk.crs, x = (pk_R, \hat{ct}, nonce), \pi)$ and aborts if it outputs 0. Otherwise it continues by computing the message as $\mu = F_K(r)$, where $(lhe.sk, K) := sk_R, \hat{ct} :=$ psig and nonce := $r$. It then runs LHE decryption algorithm to compute the signature $\sigma \leftarrow \$ \text{LHE.Dec}(lhe.sk, \hat{ct})$, and it runs the signature verification algorithm to check that $\sigma$ is a valid signature for $\mu$ under $vk$. That is, $S\cdot\text{Verify}(vk, \mu, \sigma) = 1$. If the check fails, then it aborts. Otherwise, it outputs $\mu$ and $\sigma$ as the corresponding message-signature pair.

$\text{Verify}(vk, \mu, \sigma) \rightarrow \{0, 1\}$. The verification algorithm runs the signature scheme verifier and outputs whether $S\cdot\text{Verify}(vk, F_K(r), \sigma) = 1$.

Completeness. We have by definition, that circuit $C_{sk,r}(\cdot)$ is a signature on the PRF evaluation of $F$ under key $K$. Thus, by correctness of the signature scheme and the PRF, $C_{sk,r}(K)$, computes the signature on the $F_K(r)$ and, further, by completeness of the NIZK proof system, $\hat{ct}$ is the encryption of the (homomorphically evaluated) signature on $F_K(r)$ under $sk$, for uniformly random $r$. Finally, by correctness of LHE, $\hat{ct}$ correctly decrypts to a signature $\sigma$ such that $S\cdot\text{Verify}(vk, F_K(r), \sigma)$ accepts.

Reusability. Notice that for $b \in \{0, 1\}$, if a signer issues presignature-nonce pairs $((\hat{ct}_b, \pi_b), r_b)$ to a given receiver with a secret PRF key $K$ then,

$$
\text{Pr}[\mu_0 = \mu_1] = \text{Pr}[F_K(r_0) = F_K(r_1)]
$$

The probability that for uniformly sampled values $r_0, r_1 \in \{0, 1\}^\lambda$, we have that $r_0 = r_1$ is negligible in $\lambda$. Additionally, if $r_0 \neq r_1$, then the probability that $F_K(r_0) = F_K(r_1)$ also negligibly small for a secure PRF $F$, and hence also the probability above is overall negligible. So the construction is reusable.

---

This randomness is needed for circuit privacy.
9.4 One-more unforgeability

We first consider the one-more unforgeability of this protocol.

Theorem 9.4. If $\mathcal{LHE}$ is a circuit private LHE and $S$ is a secure signature scheme, then Construction 9.3 is one-more unforgeable NIBS protocol in the KOSK model.

Proof. We define the following hybrids:

Hybrid$_0$ This corresponds to the real one-more unforgeability experiment in the KOSK model.

1. After generating the public parameters $pp$ of the protocol, the challenger runs the setup algorithm for the signature scheme $S$, generates keys as $(sk, vk) \leftarrow S.\text{Setup}(1^\lambda)$ and sends $vk$ to the adversary.

2. In the $i$th query, the adversary chooses a valid $pk_{R_i}$, $sk_{R_i}$ pair and requests a presignature using it. For each such query, the challenger replies with a presignature $\text{psig}_i$ that it computes as follows:
   
   (a) It parses $pk_{R_i}$ as $(lhe.ek_i, ct_i)$ and $sk_{R_i}$ as $(lhe.sk_i, K_i)$.  
   (b) It checks whether $K_i = \text{LHE}.\text{Dec}(lhe.sk_i, ct_i)$. If the check fails, the challenger aborts and continues otherwise.
   (c) It samples a random $r_i \leftarrow \{0, 1\}^\lambda$.
   (d) Then, for circuit $C_{sk,r_i}(K_i) \overset{\text{def}}{=} S.\text{Sign}(sk, F_{K_i}(r_i))$, the challenger runs the homomorphic evaluation algorithm with randomness $\rho_i$ to obtain $\hat{ct}_i$ as
      
      $$\hat{ct}_i \leftarrow \text{LHE}.\text{Eval}(lhe.ek_i, C_{sk,r_i}, ct_i; \rho_i)$$
   (e) It creates a NIZK proof for the language $\mathcal{L}_3$ as
      
      $$\pi_i \leftarrow \text{NIZK}.\text{Prove}(\text{nizk.crs}, x_i = (pk_{R_i}\hat{ct}_i, r_i), \omega_i = (sk, C_{sk,r_i}, \rho_i))$$
   (f) Sets $\text{psig}_i := (\hat{ct}_i, \pi_i)$ and nonce := $r_i$.

   The adversary repeats this step for a total of $\ell$ queries.

3. Adversary outputs $k$ message and signature pairs $((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k))$. Adversary wins if $\mu_i \neq \mu_j$ for $1 \leq i < j \leq k$, Verify$(vk, \mu_i, \sigma_i) = 1$ for all $i \in [k]$, and $k > \ell$.

Hybrid$_i$ For $i \in [\ell]$ each Hybrid$_i$ is the same as Hybrid$_{i-1}$, except that on $i$th presignature query, instead of honestly generating the NIZK proof $\pi_i$, the challenger simulates it without any witness.

2. (e) It sets statement $x_i$ to be $(pk_{R_i}\hat{ct}_i, r_i)$ and, without setting any witness, generates $\pi_i$ using NIZK simulator.

\footnote{Note that this can be done without loss of generality in the KOSK model as this does not affect any change to the adversary's point of view.}
Hybrid_{ℓ+1} For \( i \in [\ell] \) each Hybrid_{ℓ+1} is the same as Hybrid_{ℓ+1−1}, except that on the \( i^{th} \) presignature query, instead of homomorphically evaluating the circuit, it simulates the ciphertext generation using the LHE simulator.

2. (c) Then, for circuit \( C_{sk,ri}(K_i) \overset{\text{def}}{=} \text{S.Sign}(sk, F_{K_i}(r_i)) \), the challenger computes \( \sigma_i \leftarrow \text{S.Sign}(sk, F_{K_i}(r_i)) \). It then runs the LHE simulator as \( \text{ct}_i \leftarrow \text{Sim(lhe.ek}_i, \sigma_i) \).

Let \( \text{Adv}_{A}^{\text{τ}} \) denote the security advantage of an adversary \( A \) in the one-more unforgeability game in Hybrid_{\text{τ}}. Then, the following must hold:

Lemma 9.5. Assuming zero-knowledge property of NIZK, it holds that for all PPT adversaries \( A \),
\[
|\text{Adv}_{A}^{\text{τ}} - \text{Adv}_{A}^{\text{τ}−1}| = \text{negl}(\lambda) \quad \text{for } i \in [\ell].
\]

Proof. For any \( i \in [\ell] \), consider Hybrid_{\text{τ}}, in which, the challenger, responds to the \( i^{th} \) presignature query by simulating the NIZK proof according to the statement \( x_i = (pk_{R_i,ct}, r_i) \). Concretely, let \( A \) be a PPT attacker such that \( |\text{Adv}_{A}^{\text{τ}−1} - \text{Adv}_{A}^{\text{τ}}| \) is non-negligible, we design a reduction algorithm \( B \) that breaks the zero-knowledge property of NIZK with non-negligible advantage.

The NIZK challenger starts by sampling \( b^* \leftarrow \{0, 1\} \). If \( b^* = 0 \), it sets nizk.crs \( \leftarrow \text{NIZK.Setup}(1^\lambda) \). Otherwise it runs the simulator nizk.crs \( \leftarrow \text{S.Setup}(1^\lambda) \). It sends \( \text{ct}_i \) to \( A \), which in turn forwards it to \( A \) as \( \text{pp} \). After generating the public parameters of the protocol, \( B \) runs the setup algorithm for the signature scheme \( S \) to generate the key pair \( (sk, vk) \leftarrow \text{S.Setup}(1^\lambda) \).

It sends \( \text{ct}_i \) to \( A \), who is then allowed to make a series of presignature queries. To answer the \( i^{th} \) query, \( B \) computes \( x_i = (pk_{R_i,ct_i}, r_i), \omega_i = (sk, C_{sk,ri} \text{ as the signer would, and queries the NIZK challenger with } (x_i, \omega_i) \) who replies with a proof \( \pi_i \). \( B \) then sets psig. := \((\text{ct}_i, \pi_i) \) and nonce := \( r_i \), and forwards it to \( A \). The attacker \( A \) outputs \( k \) message signature pairs \( (\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k) \). If \( \mu_{i'} \neq \mu_j \) for \( 1 \leq i' < j \leq k \), Verify(vk, \pi_i, \sigma_i) = 1 \) for \( 1 \leq i' \leq k \), and \( k > \ell \), then \( B \) sends 0 as its guess (i.e., \( \pi_i \) is a simulated proof), otherwise it sends 1 as its guess (i.e., \( \pi_i \) is honestly generated from NIZK.Prove(nizk.crs, x_i, \omega_i)).

Note that if the NIZK challenger honestly generates proof the \( \pi_i \) using NIZK.Prove(nizk.crs, x_i, \omega_i), then \( B \) perfectly simulates the experiment of Hybrid_{\text{τ}−1} for adversary \( A \). Otherwise it simulates the experiment of Hybrid_{\text{τ}}. As a result, if \( |\text{Adv}_{A}^{\text{τ}−1} - \text{Adv}_{A}^{\text{τ}}| \) is non-negligible, then \( B \) breaks the zero-knowledge property of NIZK with non-negligible advantage. Thus the lemma holds for \( i \) which is general, so it holds for all \( i \in [\ell] \). \( \square \)

Lemma 9.6. If \( \text{LHE} \) is a circuit private LHE, then for all PPT adversaries \( A \),
\[
|\text{Adv}_{A}^{\text{ℓ+i}−1} - \text{Adv}_{A}^{\text{ℓ+i}}| = \text{negl}(\lambda) \quad \text{for } j \in [\ell].
\]

Proof. For any \( i \in [\ell] \), consider the hybrid Hybrid_{\text{ℓ+i}}, in which, the challenger, responds to the \( i^{th} \) presignature query by simulating the ciphertext generation using the LHE simulator instead of homomorphically evaluating the circuit. Concretely, for circuit \( C_{sk,ri}(K_i) \overset{\text{def}}{=} \text{S.Sign}(sk, F_{K_i}(r_i)) \), the challenger leverages its knowledge of the secret key to compute \( \sigma_i \leftarrow \text{S.Sign}(sk, F_{K_i}(r_i)) \). It then runs the LHE simulator as \( \text{ct}_i \leftarrow \text{Sim(lhe.ek}_i, \sigma_i) \). By circuit privacy of LHE, we have that for such a PPT algorithm \( \text{Sim} \), in particular, the following holds:
\[
(lhe.ek, \text{LHE.Eval}(lhe.ek, C_{sk,ri}(ct_i), ct_i)) \approx_c (lhe.ek, \text{Sim}(lhe.ek, C_{sk,ri}(K_i), ct_i))
\]

64
It further follows by construction that \( \sigma_i \) is a valid signature on \( F_{K_i}(r_i) \). So, \( C_{sk,r_i}(K_i) = \sigma_i \) and thus, a PPT adversary has negligible advantage in distinguishing \( \hat{c}_i \) produced as the output of LHE.Eval(lhe.ek, \( C_{sk,r_i}, \hat{c}_i \)) and that produced by the LHE Simulator Sim(lhe.ek, \( \sigma_i \)). Thus the lemma holds for \( i \) which is general, so it holds for all \( i \in [\ell] \).

**Lemma 9.7.** Assuming the signature scheme \( S \) is secure under existential unforgeability (Definition A.5), adversary \( A \) can have at most negligible advantage in Hybrid_2\( \ell \).

**Proof.** We reduce to the unforgeability of the underlying signature scheme. Suppose there exists a PPT attacker \( A \) that wins the one-more unforgeability game with non-negligible probability \( \epsilon = \epsilon(\lambda) \). It outputs \( k \) message and signature pairs \( ((\mu_1, \sigma_1), \ldots, (\mu_k, \sigma_k)) \), and wins if and only if \( \mu_i \neq \mu_j \) for \( 1 \leq i < j \leq k \), \( \text{Verify}(vk, \mu_i, \sigma_i) = 1 \) for \( 1 \leq i \leq k \), and \( k > \ell \). Also recall that in the KOSK model, the attacker must output a valid \( pk_{R_i}, sk_R \) such that \( K_i = \text{LHE.E} \text{nc}(lhe.sk_i, ct) \) and \( ct_i = \text{LHE.Enc}(lhe.sk_i, K_i) \). It then follows by correctness of the underlying scheme that any ciphertext \( \hat{c}_i \) must decrypt uniquely under lhe.sk.

Consider, then, a reduction algorithm \( B \) that, given access to \( A \), breaks the security of signature scheme \( S \). To begin, the Unf challenger generates a \( (sk, vk) \) pair using its S.Setup algorithm, and sends \( vk \) to \( B \) who then forwards it to \( A \). For every presignature query it receives for \( R_i \), \( B \) ensures admissibility by itself computing the key-pair \( (pk_{R_i}, sk_{R_i}) \) (recall in KOSK model, the secret key can be viewed as random coins of the setup algorithm), and aborts if \( A \) is inadmissible. Otherwise, after parsing the key-pair as \( lhe.ek_i, ct_i \) and \( lhe.sk_i, K_i \) respectively, it samples a value \( r_i \leftarrow \{0, 1\}^\lambda \) uniformly at random, computes \( \mu_i = F_{K_i}(r_i) \). The reduction algorithm, \( B \), then sends \( \mu_i \) as a signing query to the challenger and receives a signature \( \sigma_i \) in response. It computes \( \hat{c}_i \leftarrow \text{Sim}(lhe.ek, \sigma_i) \) and sends \( (\hat{c}_i, r_i) \) as the \( i^{th} \) presignature to \( A \). After \( \ell \) such queries, \( A \) outputs \( k > \ell \) forgeries \( \{\mu_i^*, \sigma_i^*\}_{i=1}^k \) such that for all \( i \in [k] \), \( \text{Verify}(vk, \mu_i^*, \sigma_i^*) = 1 \) and \( \mu_i^* \) are all different.

Since \( A \) makes \( \ell \) presignature queries, as does the reduction algorithm make \( \ell \) signature queries to the challenger. However, all the \( k > \ell \) messages \( \mu_i^* \) are different. So it follows by a simple pigeonhole argument that at least one of them was never queried by the attacker (and thus by the reduction). Let \( \mu^* \) be such a message, and \( \sigma^* \) its corresponding signature. The reduction algorithm, \( B \), then simply gives \( (\mu^*, \sigma^*) \) as a forgery to the challenger, thus breaking the security of signature scheme \( S \).

This completes the proof for one-more unforgeability of Construction 9.3 in the KOSK model.

**9.5 Strong receiver blindness**

**Theorem 9.8.** If NIZK is a NIZKAoK, \( LH\mathcal{E} \) is a IND-CPA secure encryption scheme and \( F \) is a secure pseudorandom function, then Construction 9.3 is a strong receiver-blind NIBS protocol.

**Proof.** We define the following hybrids:
Hybrid$_0$ This corresponds to the real receiver blindness experiment.

1. The challenger, after generating the public parameters $pp$ of the protocol, runs the receiver key generation algorithm for each $b \in \{0, 1\}$ as follows:

   (a) It samples an LHE key pair $(\text{lhe}.sk_b, \text{lhe}.ek_b) \leftarrow \text{LHE}.$Setup$(1^\lambda, 1^d)$ and random PRF key $K_b \leftarrow \{0, 1\}^\lambda$.

   (b) It encrypts $K_b$ as $ct_b \leftarrow \text{LHE}.$Enc$(\text{lhe}.sk_b, K_b)$, and sets receiver secret and public keys as $sk_{R_b} := (\text{lhe}.sk_b, K_b)$ and $pk_{R_b} := (\text{lhe}.ek_b, ct_b)$.

   It then sends $pk_{R_0}$ and $pk_{R_1}$ to the attacker.

2. At any point in the protocol, the adversary is allowed to make a series of $\kappa = \kappa(\lambda)$ queries for $\kappa$ polynomially bounded in $\lambda$. For any $i \in [\kappa]$, the $i^{th}$ query is explained as follows:

   (a) In the $i^{th}$ query, the adversary chooses a bit $b^{(i)}$, a verification key $vk^{(i)}$, a presignature $\left(\hat{ct}^{(i)}, \pi^{(i)}\right)$ and a nonce $r^{(i)}$.

   (b) The challenger runs the NIZK verifier as $\text{NIZK}.\text{Verify}(\text{nizk.crs}, x^{(i)} = (pk_{R_{b^{(i)}}}, \hat{ct}^{(i)}, r^{(i)}, \pi^{(i)})$ and aborts if it outputs $0$.

   (c) Continuing otherwise, the challenger computes the message as $\mu^{(i)} = F_{K_{b^{(i)}}}(r^{(i)})$.

   (d) It then runs LHE decryption algorithm to compute the signature $\sigma^{(i)} \leftarrow \text{LHE}.$Dec$(\text{lhe}.sk_{b^{(i)}}, \hat{ct}^{(i)})$.

   (e) Finally it calls the signature verification algorithm $\text{S}.\text{Verify}(vk^{(i)}, \mu^{(i)}, \sigma^{(i)})$. If the output is $0$, the check fails and it aborts. It continues otherwise.

   The challenger then sends the message-signature pair $\mu^{(i)}$ and $\sigma^{(i)}$ to the attacker.

3. For attacker’s signature query $(vk, ((\hat{ct}_0, \pi_0), (r_0), ((\hat{ct}_1, \pi_1), (r_1)))$, the challenger computes two message-signature pairs, one for each $b \in \{0, 1\}$, as follows:

   (a) It runs the NIZK verifier as $\text{NIZK}.\text{Verify}(\text{nizk.crs}, x_b = (pk_{R_b}, \hat{ct}_b, r_b), \pi_b)$ and aborts if it outputs $0$.

   (b) It computes the message as $\mu_b = F_{K_b}(r_b)$.

   (c) It then runs LHE decryption algorithm to compute the signature $\sigma_b \leftarrow \text{LHE}.$Dec$(\text{lhe}.sk_b, \hat{ct}_b)$

   (d) Finally it calls the signature verification algorithm $\text{S}.\text{Verify}(vk, \mu_b, \sigma_b)$. If the output is $0$, the check fails and it aborts. It continues otherwise.

4. The challenger samples a bit $\hat{b} \leftarrow \{0, 1\}$ and sends $(\mu_{\hat{b}}, \sigma_{\hat{b}}, \mu_{1 - \hat{b}}, \sigma_{1 - \hat{b}})$ to the attacker.

5. The attacker, who continues to have Obtain oracle access, outputs a bit $b'$.

An admissible adversary $\mathcal{A}$ wins if $b' = \hat{b}$. 

66
Hybrid$_1$ This is the same as Hybrid$_0$, except that the challenger uses a uniformly sampled value \( \tau \leftarrow \{0,1\}^\lambda \) to generate nizk.crs.

1. The challenger samples \( \tau \leftarrow \{0,1\}^\lambda \), generates \( \text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda; \tau) \) and sets \( \text{pp} = \text{nizk.crs} \). It then runs the receiver key generation algorithm for each \( b \in \{0,1\} \) (same as before).

Hybrid$_2$ This is the same as Hybrid$_1$, except that for every Obtain query, instead of running the LHE decryption algorithm to obtain the signature, it runs the NIZK AoK extractor to extract the secret signing key (as part of the witness) and uses the signing algorithm under \( \text{sk}^{(i)} \) to obtain the signature on message \( \mu^{(i)} = F_{K_{b(i)}}(r^{(i)}) \) for all \( i \in [\kappa] \), i.e.,

2. (d) Challenger extracts \( (\text{sk}^{(i)}, \rho^{(i)}) := \mathcal{E}(\tau, x^{(i)} := (\text{pk}_{R_{b(i)}}, \tilde{c}^{(i)}, r^{(i)}, \pi^{(i)})) \), and uses it to compute the signature \( \sigma^{(i)} \leftarrow \text{S.Sign}((\text{sk}^{(i)}, \mu^{(i)})). 

Hybrid$_3$ This is the same as Hybrid$_2$, except instead of running the LHE decryption algorithm to obtain the signature, it runs the NIZK AoK extractor to extract the secret signing key (as part of the witness) and uses the signing algorithm under \( \text{sk} \) to obtain the signature on message \( \mu_b \) for each \( b \in \{0,1\} \).

3. (c) Challenger extracts \( (\text{sk}, \rho_b) := \mathcal{E}(\tau, x_b := (\text{pk}_{R_b}, \tilde{c}_b, r_b, \pi_b)) \) and uses it to compute the signature \( \sigma_b \leftarrow \text{S.Sign}(\text{sk}, \mu_b). 

Hybrid$_4$ This is the same as Hybrid$_3$, except that for each \( b \in \{0,1\} \), instead of generating \( \text{ct}_b \leftarrow \text{LHE.Enc}(\text{lhe.sk}_b, K_b) \), the challenger sets \( \text{ct}_b \) as \( \text{LHE.Enc}(\text{lhe.sk}_b, 0^\lambda) \).

1. (b) It computes \( \text{ct}_b \leftarrow \text{LHE.Enc}(\text{lhe.sk}_b, 0^\lambda) \) and sets receiver secret and public keys as \( \text{sk}_{R_b} := (\text{lhe.sk}_b, K_b) \) and \( \text{pk}_{R_b} := (\text{lhe.ek}_b, \text{ct}_b) \).

Hybrid$_5$ This is the same as Hybrid$_4$, except instead of computing message \( \mu_b \) as the PRF evaluation of \( r_b \) under \( K_b \), the challenger samples it uniformly at random. for each \( b \in \{0,1\} \).

3. (b) It samples message \( \mu_b \leftarrow \{0,1\}^\lambda \) uniformly at random.

Let \( \text{Adv}'_\mathcal{A} \) denote the security advantage of an adversary \( \mathcal{A} \) in the strong receiver blindness game in Hybrid$_j$. Then, the following must hold:

**Lemma 9.9.** For all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}'_\mathcal{A} - \text{Adv}'_\mathcal{A}| = 0 \).

**Proof.** Notice that the only difference between Hybrid$_0$ and Hybrid$_1$ is that the randomness \( \tau \) of the CRS generation algorithm is chosen by the challenger. Since \( \tau \) is chosen uniformly by the challenger, this affects no change from the adversaries perspective. \( \Box \)

**Lemma 9.10.** If NIZK is a NIZKAoK, then for all PPT adversaries \( \mathcal{A} \), \( |\text{Adv}'_\mathcal{A} - \text{Adv}'_\mathcal{A}| = \text{negl}(\lambda) \).
Proof. Suppose for contradiction that \( \exists i \in [\kappa] \) for which the lemma does not hold. Then one of two cases must hold during Hybrid_2 against \( A \):

- Case 1: \( \omega^{(i)} := (sk^{(i)}, \rho^{(i)}) \) is not a valid witness for \( \pi^{(i)} \) with respect to the instance \( x^{(i)} \).
- Case 2: \( \omega^{(i)} \) is a valid witness for \( \pi^{(i)} \) with respect to the instance \( x^{(i)} \).

In the former case we have that \( \omega^{(i)} \) is not a valid witness, but also due to the previous step NIZK Verify(nizk.crs, \( x^{(i)} \), \( \pi^{(i)} \)) = 1 since otherwise the challenger would have aborted the protocol. By definition of knowledge extraction, we know the probability of this event to be negligible. In the latter case it must have been that \( (vk^{(i)}, sk^{(i)}) \) was an invalid signature key pair, in which case S.Verify(\( vk^{(i)} \), \( F_{K_o^{(i)}}(r^{(i)}) \), \( \sigma^{(i)} \)) \( \neq 1 \) and the protocol is aborted. Thus the lemma holds for \( i \), which is general, so it holds for all \( i \in [\kappa] \).

\[ \square \]

Lemma 9.11. If NIZK is a NIZKAoK, for all PPT adversaries \( A \), \(|Adv^{2}_{A} - Adv^{3}_{A}| = negl(\lambda)\).

Proof. Let us consider an intermediate step between Hybrid_2 and Hybrid_3. In the intermediate step, everything remains the same in Hybrid_2, except for the following:

3. (c) Challenger extracts \( (sk, \rho_0) := E(\tau, x_0) := (pk_{R_o}, \tilde{c}_t, r_0, \pi_0) \) and uses it to compute the signature \( \sigma_0 \leftarrow S.Sign(sk, \mu_0) \). It also computes \( \sigma_1 \leftarrow LHE.\text{Dec}(lhe.sk_1, \tilde{c}_t) \).

Define adversary \( A \)'s advantage on the intermediate step as \(|Adv^{2.5}_{A}|\). Proceeding exactly as in the previous proof, suppose for contradiction that the adversary wins with non-negligible probability \( \epsilon = \epsilon(\lambda) \). Then one of two cases must hold in the intermediate experiment against \( A \):

- Case 1: \( \omega_0 := (sk, \rho_0) \) is not a valid witness for \( \pi_0 \) with respect to the instance \( x_0 \).
- Case 2: \( \omega_0 \) is a valid witness for \( \pi_0 \) with respect to the instance \( x_0 \).

In the former case we have that \( \omega_0 \) is not a valid witness, but also due to the previous step NIZK.Verify(nizk.crs, \( x_0 \), \( \pi_0 \)) = 1 since otherwise the challenger would have aborted the protocol. By definition of knowledge extraction, we know the probability of this event to be negligible. In the latter case it must have been that \( (vk, sk) \) was an invalid signature key pair, in which case S.Verify(\( vk \), \( F_{K_o}(r_0) \), \( \sigma_0 \)) \( \neq 1 \) and the protocol is aborted. Therefore, \(|Adv^{2}_{A} - Adv^{2.5}_{A}| = negl(\lambda)\). By a similar argument we can show that \(|Adv^{2.5}_{A} - Adv^{3}_{A}| = negl(\lambda)\) and thus the lemma follows. \[ \square \]

Lemma 9.12. If LHE is an IND-CPA secure encryption scheme, then for all PPT adversaries \( A \), \(|Adv^{3}_{A} - Adv^{4}_{A}| = negl(\lambda)\).

Proof. Let us consider an intermediate step between Hybrid_2 and Hybrid_3. In the intermediate step, everything remains the same in Hybrid_3, except for the following:

1. (b) The challenger computes \( ct_0 \leftarrow LHE.\text{Enc}(lhe.sk_{0, 0^\lambda}) \) and \( ct_1 \leftarrow LHE.\text{Enc}(lhe.sk_1, K_1) \), and sets receiver secret and public keys as \( sk_{R_b} := (lhe.sk_b, K_b) \) and \( pk_{R_b} := (lhe.ek_b, ct_b) \).
Define adversary $A$’s advantage on the intermediate step as $\text{Adv}\_{A}^{3,5}$, and suppose there exists such a PPT attacker $A$, such that $|\text{Adv}_{A}^{3} - \text{Adv}_{A}^{3,5}| = \epsilon(\lambda)$. We design a reduction algorithm $B$ that breaks the security of LHE with advantage $\epsilon(\lambda)$.

The IND-CPA challenger first samples a secret key and evaluation key pair $(\text{lhe}.sk_{0}, \text{lhe}.ek_{0})$ using LHE.Setup and sends $\text{lhe}.ek_{0}$ to the reduction algorithm $B$. The reduction algorithm then generates the public parameters of the protocol $\text{pp}$ and then samples an LHE key pair $(\text{lhe}.sk_{1}, \text{lhe}.ek_{1}) \leftarrow \text{LHE.Setup}(1^{\lambda}, 1^{d})$ and random PRF key $K_{b} \leftarrow \{0, 1\}^{\lambda}$ for each $b \in \{0, 1\}$. It sends $0^{\lambda}$ and $K_{0}$ to the challenger and receives a ciphertext $ct_{0}$ as the encryption of either $0^{\lambda}$ or $K_{0}$. It computes $ct_{1} \leftarrow \text{LHE.Enc}(\text{lhe}.sk_{1}, K_{1})$, and sets receiver secret key as $sk_{R_{0}} := (\text{lhe}.sk_{1}, K_{1})$ and the public keys as $pk_{R_{0}} := (\text{lhe}.ek_{0}, ct_{0})$. The reduction algorithm then forwards $pk_{R_{0}}$ and $pk_{R_{1}}$ to $A$. The rest of the protocol proceeds identically to Hybrid$_{3}$. At the end, $B$ samples $b \leftarrow \{0, 1\}$ uniformly at random and sends $((\mu_{b}, \sigma_{b}), (\mu_{1-b}, \sigma_{1-b}))$ to $A$ who outputs its guess $b^\prime$. If $b^\prime = b$, $B$ sends 0 as its guess (i.e., $ct_{0}$ is an encryption of $0^{\lambda}$), otherwise it sends 1 as its guess (i.e., $ct_{0}$ is an encryption of $K_{0}$).

Note that if $ct_{0}$ is an encryption of string $K_{0}$, then $B$ perfectly simulates the experiment of Hybrid$_{3}$ for adversary $A$. Otherwise it simulates the intermediate step. As a result, if $|\text{Adv}_{A}^{3} - \text{Adv}_{A}^{3,5}|$ is non-negligible, then $B$ breaks the IND-CPA security LHE with non-negligible advantage. Similarly, we have $|\text{Adv}_{A}^{3,5} - \text{Adv}_{A}^{4}| \leq \text{negl}(\lambda)$ and thus $|\text{Adv}_{A}^{3} - \text{Adv}_{A}^{4}| \leq \text{negl}(\lambda)$.

**Lemma 9.13.** If $F$ is a pseudorandom function, then for all PPT adversaries $A$, $|\text{Adv}_{A}^{4} - \text{Adv}_{A}^{5}| = \text{negl}(\lambda)$.

**Proof.** Let us consider an intermediate step between Hybrid$_{3}$ and Hybrid$_{0}$. In the intermediate step, everything remains the same in Hybrid$_{3}$, except for the following:

3. (a) Challenger samples message $\mu_{0} \leftarrow \{0, 1\}^{\lambda}$ uniformly at random, and evaluates $\mu_{1} = F_{K_{1}}(r_{1})$.

Define adversary $A$’s advantage on the intermediate step as $\text{Adv}_{A}^{1,5}$. Suppose there exists such a PPT attacker $A$, such that $|\text{Adv}_{A}^{4} - \text{Adv}_{A}^{1,5}| = \epsilon(\lambda)$. We design a reduction algorithm $B$ that breaks the pseudorandomness property of $F$ with advantage $\epsilon(\lambda)$.

The reduction algorithm $B$ generates the public parameters of the protocol $\text{pp}$ and then samples an LHE key pair $(\text{lhe}.sk_{b}, \text{lhe}.ek_{b}) \leftarrow \text{LHE.Setup}(1^{\lambda}, 1^{d})$ and random PRF key $K_{b} \leftarrow \{0, 1\}^{\lambda}$ for each $b \in \{0, 1\}$. It computes $ct_{b} \leftarrow \text{LHE.Enc}(\text{lhe}.ek_{1}, 0^{\lambda})$, and sets receiver secret and public keys as $sk_{R_{b}} := (\text{lhe}.sk_{b}, K_{b})$ and $pk_{R_{b}} := (\text{lhe}.ek_{b}, ct_{b})$. $B$ then forwards $pk_{R_{0}}$ and $pk_{R_{1}}$ to $A$. For each $i \in [\lambda]$, let $(b^{(i)}, v^{(i)}, k^{(i)}, c^{(i)}, r^{(i)})$ be $A$’s $i^{th}$ Obtain query. If $b^{(i)} = 0$ (resp. 0), $B$ queries the first (resp. second) PRF challenger on $r^{(i)}$, who responds with string $y_{b^{(i)}} \in \{0, 1\}^{\lambda}$, which is either a PRF evaluation of $r^{(i)}$ or a uniformly random string, and sets $\mu^{(i)} = y_{b^{(i)}}$. When $A$ outputs a signature query $(v^{(i)}, k^{(i)})$, $B$ again queries the oracle for $r_{0}$, receives a response $y_{0}$ and sets $\mu_{0} = y_{0}$. It also evaluates $\mu_{1} = F_{K_{1}}(r_{1})$. The rest of the protocol proceeds identically to Hybrid$_{5}$. At the end, $B$ samples $b \leftarrow \{0, 1\}$ uniformly at random and sends $((\mu_{b}, \sigma_{b}), (\mu_{1-b}, \sigma_{1-b}))$ to $A$ who outputs its guess $b^\prime$. If $b^\prime = b$, $B$ sends 0 as its guess (i.e., PRF challenges are pseudorandom), otherwise it sends 1 as its guess (i.e., PRF challenges are truly random).
Recall that we require $A$ to be an admissible adversary, i.e., in particular, $A$ must not have queried $r_0$ or $r_1$ in an Obtain query. Then, if PRF challenges are pseudorandom, then $B$ perfectly simulates the experiment of Hybrid$_4$ for adversary $A$. Otherwise it simulates the intermediate step. As a result, if $|\text{Adv}^4_A - \text{Adv}^{4,5}_A|$ is non-negligible, then $B$ breaks the pseudorandomness property of $F$ with non-negligible advantage. Similarly, we have $|\text{Adv}^{4}_A - \text{Adv}^{5}_A| \leq \text{negl}(\lambda)$ and thus $|\text{Adv}^4_A - \text{Adv}^5_A| \leq \text{negl}(\lambda)$. 

Observe that the adversary has 0 advantage under Hybrid$_5$. From the above lemmas, it follows that Construction 9.3 satisfies strong recipient blindness.

9.6 Strong nonce blindness

Theorem 9.14. If NIZK is a NIZK aOKE, LHE is a IND-CPA secure encryption scheme and $F$ is a secure pseudorandom function, then Construction 9.3 is a strong nonce blind NIBS protocol.

Proof. We define the following hybrids:

Hybrid$_0$ This corresponds to the real nonce blindness experiment.

1. The challenger, after generating the public parameters pp of the protocol, runs the receiver key generation algorithm as follows:
   
   a. It samples an LHE key pair $(\text{lhe.sk}, \text{lhe.ek}) \leftarrow \text{LHE.Setup}(1^\lambda, 1^d)$ and random PRF key $K \leftarrow \{0, 1\}^\lambda$.
   
   b. It encrypts $K$ as $ct \leftarrow \text{LHE.Enc}(\text{lhe.sk}, K)$, and sets receiver secret and public keys as $sk_R := (\text{lhe.sk}, K)$ and $pk_R := (\text{lhe.ek}, ct)$.

   It then sends $pk_R$ to the attacker.

2. At any point in the protocol, adversary is allowed to make a series of $\kappa = \kappa(\lambda)$ queries for $\kappa$ polynomially bounded in $\lambda$. For any $i \in [\kappa]$, the $i^{th}$ query is explained as follows:

   a. In the $i^{th}$ query, the adversary chooses a verification key $vk^{(i)}$, a presignature $(ct^{(i)}, \pi^{(i)})$ and a nonce $r^{(i)}$.
   
   b. The challenger runs the NIZK verifier as $\text{NIZK.Verify}(\text{nizk.crs}, x^{(i)} = (pk_R, ct^{(i)}, r^{(i)}), \pi^{(i)})$ and aborts if it outputs 0.
   
   c. The challenger computes the message as $\mu^{(i)} = F_K(r^{(i)})$.
   
   d. It then runs LHE decryption algorithm to compute the signature $\sigma^{(i)} \leftarrow \text{LHE.Dec}(\text{lhe.sk}, ct^{(i)})$.
   
   e. Finally it calls the signature verification algorithm $\text{S.Verify}(vk^{(i)}, \mu^{(i)}, \sigma^{(i)})$. If the output is 0, the check fails and it aborts. It continues otherwise.

   The challenger then sends the message-signature pair $\mu^{(i)}$ and $\sigma^{(i)}$ to the attacker.

3. For attacker’s signature query $(vk, ((ct_0, \pi_0), r_0), ((ct_1, \pi_1), r_1))$, the challenger computes two message-signature pairs, one for each $b \in \{0, 1\}$, as follows:
(a) It runs the NIZK verifier as NIZK.Verify(nizk.crs, x_b = (pk_R, ct_b, r_b), π_b) and aborts if it outputs 0.

(b) It computes the message as µ_b = F_K(r_b).

(c) It then runs LHE decryption algorithm to compute the signature σ_b ← LHE.Dec(lhe.sk, ct_b)

(d) Finally it calls the signature verification algorithm S.Verify(vk, µ_b, σ_b). If the output is 0, the check fails and it aborts. It continues otherwise.

4. The challenger samples a bit ˆb ← {0, 1} and sends µ_ˆb, σ_ˆb, µ_1–ˆb, σ_1–ˆb to the attacker.

5. The attacker, who continues to have oracle access, outputs a bit b'.

An admissible adversary A wins if b' = ˆb.

Hybrid1 This is the same as Hybrid0, except that the challenger provides a uniformly sampled value τ ← {0, 1}^λ to generate nizk.crs.

1. The challenger samples τ ← {0, 1}^λ, generates nizk.crs ← NIZK.Setup(1^λ; τ) and sets pp = nizk.crs. It then runs the receiver key generation algorithm for each b ∈ {0, 1} (same as before).

Hybrid2 This is the same as Hybrid1, except that for every Obtain query, instead of running the LHE decryption algorithm to obtain the signature, it runs the NIZKAoK extractor to extract the secret signing key (as part of the witness) and uses the signing algorithm under sk(i) to obtain the signature on message µ(i) for all i ∈ [κ], i.e.,

2. (d) Challenger extracts (sk(i), ρ(i)) := E(τ, x(i) := (pk_R, ct(i), r(i), π(i)), and uses it to compute the signature σ(i) ← S.Sign(sk(i), µ(i)).

Hybrid3 This is the same as Hybrid2, except instead of running the LHE decryption algorithm to obtain the signature, it runs the NIZKAoK extractor to extract the secret signing key (as part of the witness) and uses the signing algorithm under sk to obtain the signature on message µ_b for each b ∈ {0, 1}.

3. (c) Challenger extracts (sk, ρ_b) := E(τ, x_b := (pk_R, ct_b, r_b), π_b) and uses it to compute the signature σ_b ← S.Sign(sk, µ_b).

Hybrid4 This is the same as Hybrid3, except that for each b ∈ {0, 1}, instead of generating ct_b ← LHE.Enc(lhe.sk_b, K_b), the challenger sets ct_b as LHE.Enc(lhe.sk_b, 0^λ).

1. (b) It computes ct ← LHE.Enc(lhe.sk, 0^λ) and sets receiver secret and public keys as sk_R := (lhe.sk, K) and pk_R := (lhe.ek, ct).
Hybrid$_5$ This is the same as Hybrid$_4$, except instead of computing message $\mu_b$ as the PRF evaluation of $r_b$ under $K$, the challenger samples it uniformly at random. for each $b \in \{0, 1\}$.

3. (b) It samples message $\mu_b \leftarrow \{0, 1\}^\lambda$ uniformly at random.

Let $\text{Adv}^j_A$ denote the security advantage of an adversary $A$ in the strong receiver blindness game in Hybrid$_j$. Then, the following must hold:

**Lemma 9.15.** For all PPT adversaries $A$, $|\text{Adv}^0_A - \text{Adv}^1_A| = 0$.

*Proof.* (omitted) identical to the proof of Lemma 9.9.

**Lemma 9.16.** If NIZK is a NIZKaoK, then for all PPT adversaries $A$, $|\text{Adv}^1_A - \text{Adv}^2_A| = \text{negl}(\lambda)$.

*Proof.* (omitted) identical to the proof of Lemma 9.10.

**Lemma 9.17.** If NIZK is a NIZKaoK, for all PPT adversaries $A$, $|\text{Adv}^2_A - \text{Adv}^3_A| = \text{negl}(\lambda)$.

*Proof.* (omitted) almost identical to the proof of Lemma 9.11 without needing an intermediate step.

**Lemma 9.18.** If LHE is an IND-CPA secure encryption scheme, then for all PPT adversaries $A$, $|\text{Adv}^3_A - \text{Adv}^4_A| = \text{negl}(\lambda)$.

*Proof.* (omitted) identical to the proof of Lemma 9.12.

**Lemma 9.19.** If $F$ is a pseudorandom function, then for all PPT adversaries $A$, $|\text{Adv}^4_A - \text{Adv}^5_A| = \text{negl}(\lambda)$.

*Proof.* (omitted) identical to the proof of Lemma 9.13.

Observe that the adversary has 0 advantage under Hybrid$_5$. From the above lemmas, it follows that Construction 9.3 satisfies strong nonce blindness.

**Acknowledgements.** We thank Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav for answering questions about the OM-ISIS assumption.

**References**


A Preliminaries (continued)

A.1 Cryptographic building blocks

We also recall a series of standard cryptographic building blocks.

A.1.1 Pseudo Random Functions

A pseudorandom function (PRF) \( F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \) is a keyed function\(^{15}\), that takes a \( \lambda \)-bit key as input, and on input \( x \in \{0, 1\}^\lambda \), it outputs a value \( y = F_K(x) \). *(Throughout the paper, we use \( F_K(x) \) as a shorthand for PRF evaluation.)*

Definition A.1 (Pseudorandomness). A PRF \( F \) is said to be secure if for every stateful PPT adversary \( A \), there exists a negligible function \( \text{negl}(\cdot) \) such that for all \( \lambda \), the following holds:

\[
\Pr[A^{O_{K,b}(1^\lambda)}(1^\lambda) = b : K \leftarrow \{0, 1\}^\lambda, b \leftarrow \{0, 1\}] \leq \frac{1}{2} + \text{negl}(\lambda),
\]

where oracle \( O_{K,b}(\cdot) \) is defined as \( F_K(\cdot) \) if \( b = 0 \), otherwise it is defined as a random function from \( \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \).

A.1.2 Perfectly binding commitments

A commitment scheme \( \text{COM} \) consists of the following algorithms.

\( \text{Setup}(1^\lambda, 1^n) \rightarrow \text{crs} \). The setup algorithm takes as input the security parameter \( \lambda \), message length \( n \), and outputs a crs \( \text{crs} \).

\( \text{Com}(\text{crs}, m; r) \rightarrow c \). The commit algorithm that takes as input the crs \( \text{crs} \), message \( m \in \{0, 1\}^n \), and randomness \( r \in \{0, 1\}^\lambda \). It outputs a commitment \( c \). *(We assume for simplicity that \( r \) is always a \( \lambda \)-bit string. Note that this follows w.l.o.g., and is not an extra assumption.)*

\( \text{Verify}(\text{crs}, m, c, r) \rightarrow 0/1 \). The verification algorithm takes as input the crs \( \text{crs} \), message \( m \), commitment \( c \), and an opening/randomness \( r \). It outputs either 0 or 1 to signal validity of the opening.

Correctness. A commitment scheme \( \text{COM} \) is correct if for every \( \lambda, n \in \mathbb{N} \), crs \( \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^n) \), any message \( m \in \{0, 1\}^n \), the following holds

\[
\Pr[\text{Verify}(\text{crs}, m, c, r) = 1 : r \leftarrow \{0, 1\}^\lambda, c = \text{Com}(\text{crs}, m; r)] = 1.
\]

Perfect binding. A commitment scheme is said to be perfectly binding if no commitment can have valid openings for two different messages.

Definition A.2 (Perfect binding). A commitment scheme \( \text{COM} \) is perfectly binding if for all \( \lambda, n \in \mathbb{N} \), every crs \( \text{crs} \leftarrow \text{Setup}(1^\lambda, 1^n) \), for every \( (c, m_1, r_1, m_2, r_2) \) such that \( m_1 \neq m_2 \), the following holds for at least one \( i \in \{1, 2\} \):

\[
\Pr[\text{Verify}(\text{crs}, m_i, c, r_i) = 1] = 0.
\]

\(^{15}\)For simplicity, we fix the key space, input space, and output space to be \( \lambda \)-bit strings.
Computational hiding. A commitment scheme is said to be computationally hiding if a commitment hides the messages when the openings are hidden.

Definition A.3 (Hiding). A commitment scheme COM is computationally hiding if for every stateful PPT attacker A, there exists a negligible function negl(·) such that for all λ ∈ N, the following holds

\[
\Pr \left[ A(c) = b \land m_0, m_1 \in \{0, 1\}^n : \begin{array}{l}
1^n \leftarrow A(1^n), \ crs \leftarrow \text{Setup}(1^n, 1^n) \\
(m_0, m_1) \leftarrow A(\text{crs}), \ b \leftarrow \{0, 1\} \\
c \leftarrow \text{Com}(\text{crs}, m_b)
\end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda).
\]

A.1.3 Public key encryption

A public key encryption (PKE) scheme PKE consists of the following polynomial-time algorithms.

\[\text{Setup}(1^\lambda) \rightarrow (pk, sk).\] The setup algorithm takes as input the security parameter λ, and outputs a public-secret key pair (pk, sk).

\[\text{Enc}(pk, m) \rightarrow ct.\] The encryption algorithm takes as input a public key pk and a message \(m \in \{0, 1\}^\lambda\), and outputs a ciphertext ct.

\[\text{Dec}(sk, ct) \rightarrow m.\] The decryption algorithm takes as input a secret key sk and a ciphertext ct, and outputs a message m.

The scheme satisfies correctness if for all λ, m ∈ {0, 1}^λ, (pk, sk) ← Setup(1^λ), and every ciphertext ct ← Enc(pk, m), we have that Dec(sk, ct) = m. Further, the standard IND-CPA security notion is defined as follows.

Definition A.4. A public key encryption scheme PKE is IND-CPA secure if for every stateful PPT adversary A, there exists a negligible functions negl(·), such that for every λ ∈ N

\[
\Pr \left[ A(\text{ct}) = b : \begin{array}{l}
(pk, sk) \leftarrow \text{Setup}(1^\lambda); b \leftarrow \{0, 1\} \\
(m_0, m_1) \leftarrow A(pk); ct \leftarrow \text{Enc}(pk, m_b)
\end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda).
\]

A.1.4 Signature schemes

A signature scheme S = (Setup, Sign, Verify) with message space \(\mathcal{M}\) consists of three algorithms, as follows:

\[\text{Setup}(1^\lambda)\] is a randomized algorithm that takes security parameter λ as input and returns a pair of keys (sk, vk), where sk is the signing key and vk is the verification key.

\[\text{Sign}(sk, m)\] is a possibly randomized algorithm that takes as input the signing key sk, and a message m, and returns a signature σ.

\[\text{Verify}(vk, m, \sigma)\] is a deterministic algorithm that takes as input the verification key vk, a message m, and a signature σ, and outputs 1 (accepts) if verification succeeds, and 0 (rejects) otherwise.
A signature scheme satisfies correctness if for all $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, and every signing-verification key pair $(sk, vk) \leftarrow \text{Setup}(1^\lambda)$, every signature $\sigma \leftarrow \text{Sign}(sk, m)$, $\text{Verify}(vk, m, \sigma) = 1$.

**Definition A.5.** A signature scheme $S = (\text{Setup}, \text{Sign}, \text{Verify})$ is a secure signature scheme if for every PPT attacker $A$ there exists a negligible function $\text{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr \left[ \text{Verify}(vk, m^*, \sigma^*) = 1 : (sk, vk) \leftarrow \text{Setup}(1^\lambda) \right] \leq \text{negl}(\lambda),$$

and $A$ should never have queried $m^*$ to Sign oracle.

## B NIBS from General Purpose NIZKs

**Tools required.** The construction relies on a pseudorandom function $F$, a perfectly binding commitment scheme $\text{COM} = (\text{COM.Setup, COM.Com, COM.Verify})$, a standard signature scheme $S = (S\text{.Setup, S\text{.Sign, S\text{.Verify}}})$, and a NIZKAoK proof system $\text{NIZK} = (\text{NIZK.Setup, NIZK.Prove, NIZK.Verify})$ for the following language:

<table>
<thead>
<tr>
<th>Language $L_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong> Each instance $x$ is interpreted as a verification key $vk$, crs for a commitment scheme com.crs, and a message $m$.</td>
</tr>
<tr>
<td><strong>Witness:</strong> Witness $\omega$ consists of a signature $s.\sigma$, randomness $r$, PRF key $K$, and commitment opening $\text{op}$.</td>
</tr>
<tr>
<td><strong>Membership:</strong> $\omega$ is a valid witness for $x$ if the following are satisfied:</td>
</tr>
<tr>
<td>$s.\sigma$ is a valid signature for message $'c</td>
</tr>
<tr>
<td>$m$ is a PRF evaluation of $r$ w.r.t. key $K$. Namely, $m = F_K(r)$.</td>
</tr>
</tbody>
</table>

### B.1 Construction

Below we describe our non-interactive blind signature scheme NIBS.

Setup$(1^\lambda) \rightarrow \text{pp}$. It runs the setup algorithms for $\text{NIZK}$ (for language $L_5$) and COM schemes. Namely, it generates

$$\text{nizk.crs} \leftarrow \text{NIZK.Setup}(1^\lambda), \text{com.crs} \leftarrow \text{COM.Setup}(1^\lambda).$$

It outputs $\text{pp} = (\text{nizk.crs, com.crs})$. 

81
KeyGen_S(pp) → (sk, vk). The signer’s setup algorithm runs the setup algorithm for the signature scheme S. Namely, it generates keys as

\[(sk, vk) \leftarrow S.\text{Setup}(1^\lambda).\]

KeyGen_R(pp) → (sk_R, pk_R). The receiver’s setup algorithm first samples a random PRF key \(K \leftarrow \{0, 1\}^\lambda\) and commitment randomness \(op \leftarrow \{0, 1\}^\lambda\). Next, it computes a commitment as \(c = \text{COM}.\text{Com}(\text{com.crs}, K; op)\). Finally, it outputs receiver’s secret key and public key as \(sk_R := (K, op)\) and \(pk_R := c\).

Issue(sk, pk_R) → (psig, nonce). The issue algorithm samples a random message \(r \leftarrow \{0, 1\}^\lambda\), and creates a signature of message ‘\(pk_R || r\)’. Namely,

\[s.\sigma \leftarrow S.\text{Sign}(sk, pk_R || r).\]

Finally, it outputs the pre-signature as \(\text{psig} := s.\sigma\) and \(\text{nonce} := r\).

Obtain(sk_R, vk, psig, nonce) → (m, \(\sigma\)). The receiver first computes the message as \(m = F_K(r)\), where \((K, op) := (sk_R, s.\sigma := \text{psig}, \text{and } r := \text{nonce})\). It then runs the NIZK prover to create the signature \(\sigma\) as a NIZK proof as follows:

\[\sigma \leftarrow \text{NIZK}.\text{Prove}(nizk.crs, x = (vk, \text{com.crs}, m), \omega = (s.\sigma, r, K, op)).\]

It sets \(c = \text{COM}.\text{Com}(\text{com.crs}, K; op)\). If \(s.\sigma\) is not a valid signature for \(c || r\), then it aborts and outputs \(\perp\). Otherwise, it outputs the above message-signature/proof pair.

Verify(vk, m, \(\sigma\)) → \{0, 1\}. The verification algorithm runs the NIZK verifier and outputs whether \(\text{NIZK}.\text{Verify}(\text{nizk.crs}, x = (vk, \text{com.crs}, m), \sigma) = 1\).

Completeness. Observe that by correctness of the signature scheme, \(S\), \(s.\sigma\) is a valid (pre)signature on \(pk_R || r\). Also, by correctness of the commitment scheme \(\text{COM}\), \(pk_R := c\) is a valid commitment to PRF key \(K\) with respect to some fixed opening \(op\). Similarly, correctness of the PRF \(F\) ensures that \(m = F_K(r)\). Thus, we have that \(\sigma\) is a valid NIZK proof for the relation language \(L_5\) described directly above, and follows from the completeness of the underlying NIZK that Construction B.1 is complete.

Reusability. For \(b \in \{0, 1\}\), if a signer issues presignature-nonce pairs \((s.\sigma_b, r_b)\) to a given receiver with \(pk_R := c\), then

\[\Pr[\mu_0 = \mu_1] = \Pr[c || r_0 = c || r_1] = \Pr[r_0 = r_1] = \text{negl}(\lambda)\]

So the construction is reusable.
B.2 One-more unforgeability

Theorem B.1. Assume that NIZK scheme NIZK is an argument of knowledge, and the signature scheme $S$ is secure, then our NIBS construction B.1 is one-more unforgeable.

Proof. This follows from a case by case reduction algorithm.

- Type 1: Consider that type 1 attacker $A$ outputs $\ell + 1$ message and signature pairs $((m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1}))$. Set $x_i := vk, com.crs, m_i$, for all $i \in [\ell + 1]$. For type 1 attacker $A$, there exists some $j \in [\ell + 1]$, such that NIZK.\,\text{Verify}(nizk.crs, x_j, \sigma_j) = 1$ and $\omega = E(\tau, x_j, \pi_j)$ is not a valid witness for some instance $x_j$ of $\mathcal{L}_5$.

- Type 2: Type 2 attacker is defined as the complementary of type 1 attacker, such that $\omega = E(\tau, x_j, \pi_j)$ is a valid witness for all $j \in [\ell + 1]$.

Claim B.2. Assuming that NIZK has a valid knowledge extractor $E$, type 1 attacker $A$ has at most negligible advantage.

Proof. Assume that type 1 adversary $A$ has non-negligible advantage $\epsilon = \epsilon(\lambda)$, we build a reduction algorithm $B$ that breaks the knowledge extractor of NIZK.

Challenger starts by generating $\tau \leftarrow \{0, 1\}^\lambda$ uniformly at random, and then generates $nizk.crs \leftarrow \text{NIZK.Setup}(1^\lambda, \tau)$. It sends $nizk.crs$ to $B$. $B$ then generates $com.crs \leftarrow \$ \text{COM.Setup}(1^\lambda)$ and outputs $pp := (nizk.crs, com.crs)$. Next, $B$ generates $(sk, vk) \leftarrow \$ \text{KeyGen}_S(pp)$ and outputs $vk$. Next, $A$ makes a series of presignature queries. For each query, $A$ sends a public key $pk_R$ to $B$ and $B$ sends back $(psig, nonce) = \text{Issue}(sk, pk_R)$. Finally, $A$ outputs $\ell + 1$ message and signature pairs $((m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1}))$. Consider that we set $x_i := (vk, com.crs, m_i)$, for all $i \in [\ell + 1]$. Then type 1 attacker $A$ guarantees that there exists some $j \in [\ell + 1]$, such that NIZK.\,\text{Verify}(nizk.crs, x_j, \sigma_j) = 1$ and $\omega = E(\tau, x_j, \pi_j)$ is not a valid witness for $x \in \mathcal{L}_5$. As such, $B$ could break the knowledge extractor of NIZK using the output by $A$. \hfill \Box

Claim B.3. Assuming signature scheme $S$ is secure, type 2 attacker $A$ has at most negligible advantage.

Assume that $A$ has non-negligible advantage $\epsilon = \epsilon(\lambda)$, we build a reduction algorithm $B$ that breaks the security of $S$ with non-negligible advantage.

The challenger starts by generating $(sk, vk) \leftarrow \$ \text{S.Setup}(1^\lambda)$, and it sends $vk$ to reduction algorithm $B$. Next, $B$ samples $com.crs \leftarrow \$ \text{COM.Setup}(1^\lambda)$ and $nizk.crs \leftarrow \$ \text{NIZK.Setup}(1^\lambda)$. $B$ then outputs public parameter $pp := (nizk.crs, com.crs)$ and signer’s verification key $vk$.

Next, $A$ is allowed to make a number of $\ell$ presignature queries, where in each query, $A$ sends $pk_R$ to $B$. $B$ then randomly samples $r \leftarrow \{0, 1\}^\lambda$ and sends $pk_R||r$ to the challenger. Challenger then returns $s, \sigma$ as the signature of $pk_R||r$. $B$ then outputs presignature $psig := s, \sigma, r$ and nonce := $r$.

To win the one-more unforgeability game, $A$ outputs $((m_1, \sigma_1), \ldots, (m_{\ell+1}, \sigma_{\ell+1}))$. For all $i \in [\ell + 1], B$ sets $x_i$ as $(vk, com.crs, m_i)$. Then, $B$ extracts witness $\omega_i$ using NIZK knowledge extractor, such that $\omega_i = E(\tau, x_i, \sigma_i)$. $B$ parses $\omega_i$ as $(s, \sigma_i, r_i, K_i, \text{op}_i)$. Next, $B$ sets $c_i$ as $\text{COM.Com}(com.crs, K_i; \text{op})$. Finally, $B$ breaks the unforgeability property of signature scheme $S$ by outputting message and signature pairs $(c_i||r_i, s, \sigma_i)$, for all $i \in [\ell + 1]$.  

83
We argue that such reduction algorithm $B$, is a valid attacker against $S$ security challenger. Recall that type 2 attacker $A$ guarantees that $m_i = F_{K_i}(r_i)$. For all $1 \leq i < j \leq \ell + 1$, $A$ as a valid attacker guarantees that $m_i \neq m_j$, thus either $K_i \neq K_j$ or $r_i \neq r_j$. If $K_i \neq K_j$, then the perfect binding property of our commitment scheme $\text{COM}$ implies $c_i \neq c_j$. As a result, $c_i | r_i \neq c_j | r_j$. Type 2 attacker also guarantees that for all $i \in [\ell + 1]$, $S.\text{Verify}(vk, c_i | r_i, s.\sigma_i) = 1$, thus $B$ is a successful attacker.

\[ \Box \]

### B.3 Strong receiver blindness

**Theorem B.4.** Assume that general-purpose NIZK protocol $\text{NIZK}$ satisfies zero-knowledge, commitment scheme $\text{COM}$ is computationally hiding, and $F$ is a secure PRF, then our NIBS construction B.1 satisfies strong receiver blindness.

**Proof.** Define the following hybrids:

**Hybrid$_0$** This corresponds to the original experiment:

1. Challenger starts by generating $\text{nizk}.\text{crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$ and $\text{com}.\text{crs} \leftarrow \text{COM.Setup}(1^\lambda)$. It outputs $pp := (\text{nizk}.\text{crs}, \text{com}.\text{crs})$.

2. For $b \in \{0, 1\}$, challenger samples PRF key $K_b \leftarrow \{0, 1\}^\lambda$ and commitment randomness $op_b \leftarrow \{0, 1\}^\lambda$. Next, it computes $c_b = \text{COM.Com}(\text{com}.\text{crs}, K_b; op_b)$. Challenger sets $sk_{R_b}$ as $(K_b, op_b)$, and $pk_{R_b}$ as $c_b$. It outputs $pk_{R_0}$ and $pk_{R_1}$.

3. The adversary is allowed to make a series of $k$ queries for $k$ polynomially bounded in $\lambda$. The $i$-th query is explained as the following ($i \in [k]$):
   
   (a) In the $i$-th query, adversary chooses bit $b^{(i)}$, verification key $vk^{(i)}$, presignature $psig^{(i)}$ and nonce $nonce^{(i)}$. Adversary sends them as the query's input to challenger.
   
   (b) Challenger sets $s.\sigma^{(i)} := psig^{(i)}$ and $r^{(i)} := nonce^{(i)}$. Challenger computes message as $m = F_{K^{(i)}}(r^{(i)})$. It sets $x := (vk^{(i)}, \text{com}.\text{crs}, m)$ and $\omega := (s.\sigma^{(i)}, r^{(i)}, K^{(i)}b^{(i)}, op^{(i)}b^{(i)})$. Challenger sets $c = \text{COM.Com}(\text{com}.\text{crs}, K^{(i)}b^{(i)}; op^{(i)}b^{(i)})$. Next, challenger checks if $s.\sigma^{(i)}$ is a valid signature for message $c | r^{(i)}$. If the check fails, challenger outputs $\perp$ and aborts.
   
   (c) Otherwise, challenger runs $\text{NIZK.prover}$ to generate signature $\sigma \leftarrow \text{NIZK.Prove}(\text{nizk}.\text{crs}, x, \omega)$. It outputs $m$ and $\sigma$.

4. Adversary then does the following:
   
   (a) Adversary chooses and outputs $vk_b$ $(psig_b, nonce_b)$ for $b \in \{0, 1\}$.
   
   (b) For $b \in \{0, 1\}$, Challenger sets $s.\sigma_b := psig_b$ and $r_b := nonce_b$. Challenger computes message as $m_b = F_{K_b}(r_b)$. It sets $x_b := (vk, \text{com}.\text{crs}, m_b)$ and $\omega_b := (s.\sigma_b, r_b, K_b, op_b)$. Challenger sets $c_b = \text{COM.Com}(\text{com}.\text{crs}, K_b; op_b)$. Challenger checks if $s.\sigma_b$ is a valid signature for message $c_b | r_b$, for $b \in \{0, 1\}$. If any of the checks fails, challenger outputs $\perp$ and aborts.
(c) Otherwise, challenger runs NIZK prover to generate signature \( \sigma_b \leftarrow \text{NIZK.Prove(nizk.crs, } x_b, \omega_b) \), for \( b \in \{0, 1\} \). Next, challenger chooses \( \hat{b} \). It outputs \((m_{\hat{b}}, \sigma_{\hat{b}}, m_{1-\hat{b}}, \sigma_{1-\hat{b}})\).

5. Admissible adversary outputs bit \( b' \) and wins if \( b' = \hat{b} \).

Hybrid

Instead of honestly generating NIZK proofs for step 3 and 4, challenger outputs simulated proofs.

1. Challenger starts by generating \( \text{nizk.crs} \leftarrow S(1^\lambda) \).

3.(c) Otherwise, challenger runs NIZK simulator to generate signature \( \sigma \) as a simulated proof for instance \( x \), such that \( \sigma = \text{Sim(nizk.crs, } x) \).

4.(c) Otherwise, challenger runs NIZK simulator to generate signature \( \sigma_b \leftarrow \text{Sim(nizk.crs, } x_b) \), for \( b \in \{0, 1\} \). Next, challenger chooses \( \hat{b} \). It outputs \((m_{\hat{b}}, \sigma_{\hat{b}}, m_{1-\hat{b}}, \sigma_{1-\hat{b}})\).

Hybrid

For \( b \in \{0, 1\} \), instead of setting \( c_b = \text{COM.Com(com.crs, } K_b; \text{op}_b) \), challenger sets \( c_b = \text{COM.Com(com.crs, } 0; \text{op}_b) \).

2. For \( b \in \{0, 1\} \), challenger samples PRF key \( K_b \leftarrow \{0, 1\}^\lambda \) and commitment randomness \( \text{op}_b \leftarrow \{0, 1\}^\lambda \). Next, it computes \( c_b = \text{COM.Com(com.crs, } 0; \text{op}_b) \). Challenger sets \( \text{sk}_{R_b} \) as \((K_b, \text{op}_b)\), and \( \text{pk}_{R_0} \) as \( c_b \). It outputs \( \text{pk}_{R_0} \) and \( \text{pk}_{R_1} \).

3.(b) Challenger sets \( s.\sigma^{(i)} := \text{psig}^{(i)} \) and \( r^{(i)} := \text{nonce}^{(i)} \). Challenger computes message as \( m = F_{K_b^{(i)}}(r^{(i)}) \). It sets \( x := (vk^{(i)}, \text{com.crs}, m) \). Challenger sets \( c = \text{COM.Com(com.crs, } 0; \text{op}_{b^{(i)}}) \). Next, challenger checks if \( s.\sigma^{(i)} \) is a valid signature for message \( c||r^{(i)} \). If the check fails, challenger outputs \( \perp \) and aborts.

4.(b) For \( b \in \{0, 1\} \). Challenger sets \( s.\sigma_b := \text{psig}_b \) and \( r_b := \text{nonce}_b \). Challenger computes message as \( m_b = F_{K_b}(r_b) \). It sets \( x_b := (vk, \text{com.crs}, m_b) \). Challenger sets \( c_b = \text{COM.Com(com.crs, } 0; \text{op}_b) \). Challenger checks if \( s.\sigma_b \) is a valid signature for message \( c_b||r_b \), for \( b \in \{0, 1\} \). If any of the checks fails, challenger outputs \( \perp \) and aborts.

Hybrid

On step 3, 4, instead of setting messages as an output of PRF \( F \), challenger samples messages uniformly at random.

3.(b) Challenger sets \( s.\sigma^{(i)} := \text{psig}^{(i)} \) and \( r^{(i)} := \text{nonce}^{(i)} \). Challenger samples message as \( m \) uniformly at random. It sets \( x := (vk^{(i)}, \text{com.crs}, m) \). Challenger sets \( c = \text{COM.Com(com.crs, } 0; \text{op}_{b^{(i)}}) \). Next, challenger checks if \( s.\sigma^{(i)} \) is a valid signature for message \( c||r^{(i)} \). If the check fails, challenger outputs \( \perp \) and aborts.

4.(b) For \( b \in \{0, 1\} \). Challenger sets \( s.\sigma_b := \text{psig}_b \) and \( r_b := \text{nonce}_b \). Challenger samples message as \( m_b \) uniformly at random. It sets \( x_b := (vk, \text{com.crs}, m_b) \). Challenger sets \( c_b = \text{COM.Com(com.crs, } 0; \text{op}_b) \). Challenger checks if \( s.\sigma_b \) is a valid signature for message \( c_b||r_b \), for \( b \in \{0, 1\} \). If any of the checks fails, challenger outputs \( \perp \) and aborts.
Let $\text{Adv}_\lambda^j$ denote the security advantage of an adversary $A$ in Hybrid$_j$, then the following must hold:

**Lemma B.5.** Assuming that our NIZK scheme NIZK satisfies zero-knowledge property, then for all PPT adversaries $A$, $|\text{Adv}_\lambda^1 - \text{Adv}_\lambda^0| \leq \text{negl}(\lambda)$.

**Proof.** Suppose there exists some PPT adversary $A$ such that $|\text{Adv}_\lambda^1 - \text{Adv}_\lambda^0| = \epsilon(\lambda)$, we design a reduction algorithm $B$ that breaks the security of NIZK with non-negligible advantage.

NIZK challenger starts by sampling $b^* \leftarrow \{0, 1\}$. If $b^* = 0$, it samples $\text{nizk.crs} \leftarrow S(1^\lambda)$. Otherwise, it samples $\text{nizk.crs} \leftarrow S(1^\lambda)$. Challenger sends $\text{nizk.crs} \leftarrow B$. $B$ then samples $\text{com.crs} \leftarrow \text{NIZK.Setup}(1^\lambda)$ and outputs $\text{pp} := (\text{nizk.crs}, \text{com.crs})$. Next, for $b \in \{0, 1\}$, $B$ samples PRF key $K_b \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op}_b \leftarrow \{0, 1\}^\lambda$. $B$ computes $c_b = \text{COM.Com}((\text{com.crs}, K_b; \text{op}_b))$.

Then, $B$ sets $\text{sk}_R_b$ as $(K_b, \text{op}_b)$ and $\text{pk}_R_b$ as $c_b$. $B$ sends $\text{pk}_R_0$ and $\text{pk}_R_1$ to $A$.

$A$ then makes a series of $k$ queries: In the $i$-th query, $A$ outputs $b^{(i)}$, $\text{vk}^{(i)}$, $\text{psig}^{(i)}$, and $\text{nonce}^{(i)}$. $B$ sets $s \cdot \sigma^{(i)} := \text{psig}^{(i)}$ and $r^{(i)} := \text{nonce}^{(i)}$. $B$ computes message as $m = F_{K_b^{(i)}}(r^{(i)})$. It sets $x := (\text{vk}, \text{com.crs}, m)$, $\omega := (s \cdot \sigma^{(i)}, r^{(i)}, K_{b^{(i)}}; \text{op}_{b^{(i)}})$. $B$ also sets $c := \text{COM.Com}((\text{com.crs}, K_{b^{(i)}}; \text{op}_{b^{(i)}}))$.

$B$ then checks if $s \cdot \sigma^{(i)}$ is a valid signature for message $c||r^{(i)}$. If the check fails, $B$ outputs $\bot$ and aborts. Otherwise, $B$ queries NIZK challenger with $(x, \omega)$ and challenger replies with signature $\sigma$. It outputs $m$ and $\sigma$.

Finally, $A$ outputs $\text{vk}$, $(\text{psig}_b, \text{nonce}_b)_b$ for $b \in \{0, 1\}$. $B$ sets $s \cdot \sigma_b := \text{psig}_b$ and $r_b := \text{nonce}_b$. For $b \in \{0, 1\}$, $B$ computes message as $m_b = F_{K_b}(r_b)$. It sets $x_b := (\text{vk}, \text{com.crs}, m_b)$ and $\omega_b := (s \cdot \sigma, r, K_{b}; \text{op}_b)$. $B$ also sets $c_b = \text{COM.Com}((\text{com.crs}, K_{b}; \text{op}_b))$. $B$ checks if $s \cdot \sigma_b$ is a valid signature for $c_b||r_b$, for $b \in \{0, 1\}$. If any of the checks fails, $B$ outputs $\bot$ and aborts. Otherwise, $B$ chooses $b$. Next, it queries NIZK challenger with $(x_0, \omega_0)$ and $(x_1, \omega_1)$. Challenger sends back proofs $\sigma_0$ and $\sigma_1$. $B$ outputs $(m_{b}, \sigma_b, m_{1-b}, \sigma_{1-b})$. $A$ then outputs bit $b'$. If $b' = \hat{b}$, $B$ outputs $0$, indicating that it receives an honestly generated proof from NIZK challenger. Otherwise, $B$ outputs $1$.

Note that if challenger uses a simulator, $B$ then perfectly simulates Hybrid$_1$ for $A$. Otherwise, it simulates Hybrid$_2$. Thus, if $|\text{Adv}_\lambda^1 - \text{Adv}_\lambda^0|$ is non-negligible, $B$ would have non-negligible advantage against the NIZK security challenger.

**Lemma B.6.** Assume that the commitment scheme COM is secure, then for all PPT adversaries $A$, $|\text{Adv}_\lambda^2 - \text{Adv}_\lambda^1| \leq \text{negl}(\lambda)$.

**Proof.** We set the following intermediate step for the proof. In the intermediate step, everything remains the same as Hybrid$_1$ and Hybrid$_2$, except for the following:

1. For $b \in \{0, 1\}$, challenger samples PRF key $K_b \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \{0, 1\}^\lambda$. Next, it computes $c_0 := \text{COM.Com}((\text{com.crs}, 0; \text{op}_0))$, and $c_1 := \text{COM.Com}((\text{com.crs}, K_1; \text{op}_1))$.

   Challenger sets $\text{sk}_R_b$ as $(K_b, \text{op}_b)$, and $\text{pk}_R_b$ as $c_b$. It outputs $\text{pk}_R_0$ and $\text{pk}_R_1$.

2. (b) Challenger sets $s \cdot \sigma^{(i)} := \text{psig}^{(i)}$ and $r^{(i)} := \text{nonce}^{(i)}$. Challenger computes message as $m = F_{K_b^{(i)}}(r^{(i)})$. It sets $x := (\text{vk}^{(i)}, \text{com.crs}, m)$ and $\omega := (s \cdot \sigma^{(i)}, r^{(i)}, K_{b^{(i)}}; \text{op}_{b^{(i)}})$. If $b^{(i)} = 0$, challenger sets $c := \text{COM.Com}((\text{com.crs}, 0; \text{op}_{b^{(i)}}))$. Otherwise, challenger sets $c := \text{COM.Com}((\text{com.crs}, K_{b^{(i)}}; \text{op}_{b^{(i)}}))$. Next, challenger checks if $s \cdot \sigma^{(i)}$ is a valid signature for message $c||r^{(i)}$. If the check fails, challenger outputs $\bot$ and aborts.

86
Define adversary $A$'s advantage on the intermediate step as $\text{Adv}_A^{1,5}$. Suppose there exists a PPT adversary $A$ such that $|\text{Adv}_A^{1,5} - \text{Adv}_A^1| = \epsilon(\lambda)$, we design a reduction algorithm $B$ which breaks the hiding property of commitment scheme.

Reduction algorithm $B$ first outputs $1^n$, where $n$ is the size of PRF key $K$. The commitment challenger starts by setting $\text{com.crs} \leftarrow \text{COM.Setup}(1^\lambda, 1^n)$ and $b^* \leftarrow \{0, 1\}$. $B$ then samples $\text{nizk.crs} \leftarrow S(1^\lambda)$ and outputs $\text{pp} := (\text{nizk.crs}, \text{com.crs})$. Next, for $b \in \{0, 1\}$, $B$ samples PRF key $K_b \leftarrow \{0, 1\}^\lambda$. $B$ then queries the challenger with $K_b$ and 0 and the challenger replies commitment $c_0$. $B$ keeps a recording of the value of $c_0$. Next, $B$ generates $c_1 = \text{COM.Com}((\text{nizk.crs}, K_1; \text{op}_1))$. $B$ sends $(\text{pk}_{R_0} := c_0, \text{pk}_{R_1} := c_1)$ to $A$.

$A$ then makes a series of $k$ queries. In the $i$-th query, $A$ sends $b(i)$, $\text{vk}(i)$, $\text{psig}(i)$, and $\text{nonce}(i)$. $B$ sets $s\cdot\sigma(i) := \text{psig}(i)$ and $r(i) := \text{nonce}(i)$. $B$ computes message as $m = F_{K_b(i)}(r(i))$. It sets $x := (\text{vk}, \text{com.crs}, m)$. If $b(i) = 0$, $B$ checks if $s\cdot\sigma(i)$ is a valid signature for message $c_0||r(i)$ and outputs $\bot$ and aborts if the check fails. Otherwise when $b(i) = 1$, $B$ randomly picks $\text{op}_1$ and sets $c_1 = \text{COM.Com}((\text{com.crs}, K_1; \text{op}_1))$. It checks if $s\cdot\sigma(i)$ is a valid signature for message $c_1||r(i)$, and outputs $\bot$ and aborts if fails. Note that in this step, $c_0$ is stored. If the check does not fail, $B$ generates $\sigma \leftarrow \text{Sim}(\text{nizk.crs}, x)$. It outputs $m$ and $\sigma$.

Finally, $A$ outputs $\text{vk}_b$, $(\text{psig}_b, \text{nonce}_b)_b$ for $b \in \{0, 1\}$. $B$ sets $s\cdot\sigma_b := \text{psig}_b$ and $r_b := \text{nonce}_b$. For $b \in \{0, 1\}$, $B$ computes message as $m_b = F_{K_b}(r_b)$. It sets $x_b := (\text{vk}, \text{com.crs}, m_b)$. $B$ also sets $c_1 = \text{COM.Com}((\text{com.crs}, K_1; \text{op}_1))$. $B$ checks if $s\cdot\sigma_b$ is a valid signature for $c_b||r_b$, for $b \in \{0, 1\}$. If any of the checks fails, $B$ outputs $\bot$ and aborts. Otherwise, $B$ chooses $b$. Next, it generates simulated proofs $\sigma_0$ and $\sigma_1$. $B$ outputs $(m_b, \sigma_b, m_{1-b}, \sigma_{1-b})$. $A$ then outputs bit $b'$. If $b' = b$, $B$ outputs 0, indicating that $c_0$ is a commitment of $K_b$. Otherwise, $B$ outputs 1, indicating that $c_1$ is a commitment of 0. Thus, if $|\text{Adv}_A^{1,5} - \text{Adv}_A^1|$ is non-negligible, $B$ would have non-negligible advantage against the hiding property of the underlying commitment scheme. Following from this proof, we also have $|\text{Adv}_A^2 - \text{Adv}_A^1|$ to be negligible. 

\textbf{Lemma B.7.} Assume that the PRF function $F$ is secure, then for all PPT adversaries $A$, $|\text{Adv}_A^3 - \text{Adv}_A^1| \leq \text{negl}(\lambda)$.

\textbf{Proof.} We define the following intermediate step for the proof. In the intermediate step, everything remains the same as Hybrid$_2$ and Hybrid$_3$, except for the following:

3.(b) Challenger sets $s\cdot\sigma(i) := \text{psig}(i)$ and $r(i) := \text{nonce}(i)$. If $b(i) = 0$, challenger samples message as $m$ uniformly at random. Otherwise, it computes $m = F_{K_b(i)}(r(i))$. It sets $x := (\text{vk}, \text{com.crs}, m)$. Challenger sets $c = \text{COM.Com}((\text{com.crs}, 0; \text{op}_b(i)))$. Next, challenger checks if $s\cdot\sigma(i)$ is a valid signature for message $c||r(i)$. If the check fails, challenger outputs $\bot$ and aborts.

4.(b) For $b \in \{0, 1\}$. Challenger sets $s\cdot\sigma_b := \text{psig}_b$ and $r_b := \text{nonce}_b$. Challenger samples message as $m_0$ uniformly at random, and computes $m_1 = F_{K_1}(R_1)$. It sets $x_b := (\text{vk}, \text{com.crs}, m_b)$. Challenger sets $c_b = \text{COM.Com}((\text{com.crs}, 0; \text{op}_b))$. Challenger checks if $s\cdot\sigma_b$ is a valid signature for message $c_b||r_b$, for $b \in \{0, 1\}$. If any of the checks fails, challenger outputs $\bot$ and aborts.

87
Following from this proof, we also have

Define adversary $A$'s advantage on the intermediate step as $\text{Adv}_A^{2,5}$. Suppose there exists a PPT adversary $A$ such that $|\text{Adv}_A^{2,5} - \text{Adv}_A^2| = \epsilon(\lambda)$, we design a reduction algorithm $B$ which breaks the PRF security.

Challenger starts by sampling a PRF key $K$, and samples a bit $b^* \leftarrow \{0, 1\}$. Reduction algorithm $B$ first sets com.crs $\leftarrow$ COM.Setup($1^\lambda, 1^n$) and nizk.crs $\leftarrow$ $S(1^\lambda)$. $B$ outputs $pp := (\text{nizk.crs}, \text{com.crs})$. Next, for $b \in \{0, 1\}$, $B$ generates $c_b = \text{COM.Com}(\text{com.crs}, 0; \text{op}_b)$ using randomness $\text{op}_b$. $B$ sends $(\text{pk}_{R_0} := c_0, \text{pk}_{R_1} := c_1)$ to $A$.

$A$ then makes a series of $k$ queries. In the $i$-th query, $A$ sends $b^{(i)}, \text{vk}^{(i)}, \text{psig}^{(i)}$, and $\text{nonce}^{(i)}$. $B$ sets $s, \sigma^{(i)} := \text{psig}^{(i)}$ and $\text{nonce}^{(i)}$. If $b^{(i)} = 0$, $B$ sends $r^{(i)}$ to the PRF challenger and challenger returns $m$. Otherwise, $B$ computes message as $m = F_{K_1}(r^{(i)})$. It sets $x := (\text{vk}, \text{com.crs}, m)$. Next, $B$ sets $c = \text{COM.Com}(\text{com.crs}, 0; \text{op}_b)$. It checks if $s, \sigma^{(i)}$ is a valid signature for message $c||r^{(i)}$, and outputs $\perp$ and aborts if fails. Otherwise, $B$ generates $\sigma \leftarrow \text{Sim(nizk.crs, x)}$. It outputs $m$ and $\sigma$.

Finally, $A$ outputs $\text{vk}, (\text{psig}_b, \text{nonce}_b)_b$ for $b \in \{0, 1\}$. $B$ sets $s_b := \text{psig}_b$ and $r_b := \text{nonce}_b$. $B$ queries the PRF challenger with $r_b$ and the challenger replies with $m_0$. $B$ computes message as $m_1 = F_{K_1}(r_1)$. For $b \in \{0, 1\}$, it sets $x_b := (\text{vk}, \text{com.crs}, m_b), c_b = \text{COM.Com}(\text{com.crs}, 0; \text{op}_b)$. $B$ checks if $s, \sigma_b$ is a valid signature for $c_b||r_b$, for $b \in \{0, 1\}$. If any of the checks fails, $B$ outputs $\perp$ and aborts. Otherwise, $B$ chooses $b$. Next, it generates simulated proofs $\sigma_0$ and $\sigma_1$. $B$ outputs $(m_b, \sigma_b, m_{1-b}, \sigma_{1-b})$. $A$ then outputs bit $b'$. If $b' = b$, $B$ outputs $0$, indicating that $m_0$ is the output of the PRF function. Otherwise, $B$ outputs $1$, indicating that $m_0$ is randomly sampled. Thus, if $|\text{Adv}_A^{2,5} - \text{Adv}_A^2|$ is non-negligible, $B$ would have non-negligible advantage against the PRF security. Following from this proof, we also have $|\text{Adv}_A^3 - \text{Adv}_A^2|$ to be negligible.

Note that any adversary has 0 advantage in Hybrid$_3$. Thus the above lemma completes the proof of our main theorem.

**B.4 Strong nonce blindness**

**Theorem B.8.** Assume that general-purpose NIZK protocol NIZK satisfies zero-knowledge, commitment scheme COM is computationally hiding, and $F$ is a secure PRF, then our NIBS construction B.1 satisfies strong receiver blindness.

**Proof.** We present the hybrids as the following:

**Hybrid$_0$** This corresponds to the original experiment:

1. Challenger starts by generating nizk.crs $\leftarrow$ NIZK.Setup($1^\lambda$) and com.crs $\leftarrow$ COM.Setup($1^\lambda$). It outputs $pp := (\text{nizk.crs}, \text{com.crs})$.

2. Challenger samples PRF key $K \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \{0, 1\}^\lambda$. Next, it computes $c = \text{COM.Com(\text{com.crs}, K; \text{op})}$. Challenger sets $\text{sk}_R$ as $(K, \text{op})$, and $\text{pk}_R$ as $c$. It outputs $\text{pk}_R$.

3. The adversary is allowed to make a series of $k$ queries for $k$ polynomially bounded in $\lambda$. The $i$-th query is explained as the following ($i \in [k]$):

   - If $b^{(i)} = 0$, $B$ sends $r^{(i)}$ to the PRF challenger and Challenger returns $m$.
   - Otherwise, $B$ generates message $m = F_{K_1}(r^{(i)})$ and outputs $m$ and $\sigma$.

   The $i$-th query is explained as the following ($i \in [k]$):

   - Challenger is to sample PRF key $K \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \{0, 1\}^\lambda$. Next, it computes $c = \text{COM.Com(\text{com.crs}, K; \text{op})}$. Challenger sets $\text{sk}_R$ as $(K, \text{op})$, and $\text{pk}_R$ as $c$. It outputs $\text{pk}_R$.

   - The adversary is allowed to make a series of $k$ queries for $k$ polynomially bounded in $\lambda$. The $i$-th query is explained as the following ($i \in [k]$):

   - Challenger is to sample PRF key $K \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \{0, 1\}^\lambda$. Next, it computes $c = \text{COM.Com(\text{com.crs}, K; \text{op})}$. Challenger sets $\text{sk}_R$ as $(K, \text{op})$, and $\text{pk}_R$ as $c$. It outputs $\text{pk}_R$.

   - The adversary is allowed to make a series of $k$ queries for $k$ polynomially bounded in $\lambda$. The $i$-th query is explained as the following ($i \in [k]$):

   - Challenger is to sample PRF key $K \leftarrow \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \{0, 1\}^\lambda$. Next, it computes $c = \text{COM.Com(\text{com.crs}, K; \text{op})}$. Challenger sets $\text{sk}_R$ as $(K, \text{op})$, and $\text{pk}_R$ as $c$. It outputs $\text{pk}_R$.
(a) In the $i$-th query, adversary chooses verification key $vk^{(i)}$, presignature $psig^{(i)}$ and nonce $nonce^{(i)}$. Adversary sends them as the query's input to challenger.

(b) Challenger sets $s\cdot\sigma^{(i)} := psig^{(i)}$ and $r^{(i)} := nonce^{(i)}$. Challenger computes message as $m = F_K(r^{(i)})$. It sets $x := (vk^{(i)}, \text{com.crs}, m)$ and $\omega := (s\cdot\sigma^{(i)}, r^{(i)}, K, \text{op})$. Challenger sets $c = \text{COM.Com}(\text{com.crs}, K; \text{op})$. Next, challenger checks if $s\cdot\sigma^{(i)}$ is a valid signature for message $c||r^{(i)}$. If the check fails, challenger outputs $\bot$ and aborts.

(c) Otherwise, challenger runs NIZK prover to generate signature $\sigma \leftarrow \$ \text{NIZK.Prove}(nizk.crs, x, \omega)$. It outputs $m$ and $\sigma$.

4. Adversary then does the following:

(a) Adversary chooses and outputs $vk$, $(psig_b, nonce_b)_b$ for $b \in \{0, 1\}$.

(b) For $b \in \{0, 1\}$. Challenger sets $s\cdot\sigma_b := psig_b$ and $r_b := nonce_b$. Challenger computes message as $m_b = F_K(r_b)$. It sets $x_b := (vk, \text{com.crs}, m_b)$ and $\omega_b := (s\cdot\sigma_b, r, K, \text{op})$. Challenger sets $c_b = \text{COM.Com}(\text{com.crs}, K; \text{op})$. Challenger checks if $s\cdot\sigma_b$ is a valid signature for message $c_b||r_b$, for $b \in \{0, 1\}$. If any of the checks fails, challenger outputs $\bot$ and aborts.

(c) Otherwise, challenger runs NIZK prover to generate signature $\sigma_b \leftarrow \$ \text{NIZK.Prove}(nizk.crs, x_b, \omega_b)$, for $b \in \{0, 1\}$. Next, challenger chooses $\hat{b}$. It outputs $(m_b, \sigma_b, m_{1-b}, \sigma_{1-\hat{b}})$.

5. Admissible adversary outputs bit $b'$ and wins if $b' = \hat{b}$.

Hybrid$_1$ Instead of honestly generating NIZK proofs for step 3 and 4, challenger outputs simulated proofs.

1. Challenger starts by generating $nizk.crs \leftarrow \$ S(1^\lambda)$.

3.(c) Otherwise, challenger runs NIZK simulator to generate signature $\sigma$ as a simulated proof for instance $x$, such that $\sigma = \text{Sim}(nizk.crs, x)$.

4.(c) Otherwise, challenger runs NIZK simulator to generate signature $\sigma_b \leftarrow \$ \text{Sim}(nizk.crs, x_b)$, for $b \in \{0, 1\}$. Next, challenger chooses $\hat{b}$. It outputs $(m_b, \sigma_b, m_{1-b}, \sigma_{1-\hat{b}})$.

Hybrid$_2$ For $b \in \{0, 1\}$, instead of setting $c_b = \text{COM.Com}(\text{com.crs}, K_b; \text{op}_b)$, challenger sets $c_b = \text{COM.Com}(\text{com.crs}, 0; \text{op}_b)$.

2. For $b \in \{0, 1\}$, challenger samples PRF key $K \leftarrow \$ \{0, 1\}^\lambda$ and commitment randomness $\text{op} \leftarrow \$ \{0, 1\}^\lambda$. Next, it computes $c = \text{COM.Com}(\text{com.crs}, 0; \text{op})$. Challenger sets $sk_R$ as $(K, \text{op})$, and $pk_R$ as $c$. It outputs $pk_R$.

3.(b) Challenger sets $s\cdot\sigma^{(i)} := psig^{(i)}$ and $r^{(i)} := nonce^{(i)}$. Challenger computes message as $m = F_K(r^{(i)})$. It sets $x := (vk^{(i)}, \text{com.crs}, m)$. Challenger sets $c = \text{COM.Com}(\text{com.crs}, 0; \text{op})$. Next, challenger checks if $s\cdot\sigma^{(i)}$ is a valid signature for message $c||r^{(i)}$. If the check fails, challenger outputs $\bot$ and aborts.
4.(b) For \( b \in \{0, 1\} \). Challenger sets \( s.\sigma_b := \text{psig}_b \) and \( r_b := \text{nonce}_b \). Challenger computes message as \( m_b = F_K(r_b) \). It sets \( x_b := (vk, \text{com.crs}, m_b) \). Challenger sets \( c = \text{COM}.\text{Com}(\text{com.crs}, 0; \text{op}) \).

Challenger checks if \( s.\sigma_b \) is a valid signature for message \( c||r_b \), for \( b \in \{0, 1\} \). If any of the checks fails, challenger outputs \( \bot \) and aborts.

Hybrid3 On step 3, 4, instead of setting messages as an output of PRF \( F \), challenger samples messages uniformly at random.

3.(b) Challenger sets \( s.\sigma(i) := \text{psig}(i) \) and \( r(i) := \text{nonce}(i) \). Challenger samples message as \( m \) uniformly at random. It sets \( x := (vk, \text{com.crs}, m) \). Challenger sets \( c = \text{COM}.\text{Com}(\text{com.crs}, 0; \text{op}) \).

Next, challenger checks if \( s.\sigma(i) \) is a valid signature for message \( c||r(i) \). If the check fails, challenger outputs \( \bot \) and aborts.

4.(b) For \( b \in \{0, 1\} \). Challenger sets \( s.\sigma_b := \text{psig}_b \) and \( r_b := \text{nonce}_b \). Challenger samples message as \( m_b \) uniformly at random. It sets \( x_b := (vk, \text{com.crs}, m_b) \). Challenger sets \( c = \text{COM}.\text{Com}(\text{com.crs}, 0; \text{op}) \).

Challenger checks if \( s.\sigma_b \) is a valid signature for message \( c||r_b \), for \( b \in \{0, 1\} \). If any of the checks fails, challenger outputs \( \bot \) and aborts.

**Lemma B.9.** Assuming that our NIZK scheme \( \text{NIZK} \) satisfies zero-knowledge property, then for all PPT adversaries \( A \), \(|\text{Adv}^1_A - \text{Adv}^0_A| \leq \text{negl}(\lambda)\).

*Proof.* (omitted) follows by extending the proof of Lemma B.5.

**Lemma B.10.** Assume that the commitment scheme \( \text{COM} \) is secure, then for all PPT adversaries \( A \), \(|\text{Adv}^2_A - \text{Adv}^1_A| \leq \text{negl}(\lambda)\).

*Proof.* (omitted) follows by extending the proof of Lemma B.6.

**Lemma B.11.** Assume that the PRF function \( F \) is secure, then for all PPT adversaries \( A \), \(|\text{Adv}^3_A - \text{Adv}^2_A| \leq \text{negl}(\lambda)\).

*Proof.* (omitted) follows by extending the proof of Lemma B.7.

Since adversary could only have 0 advantage in Hybrid3. Thus the above lemma completes the proof of our main theorem.

**C Knowledge of Secret Key Assumption**

In Section 9, we relied on the KOSK model [MOR01, Bol03] to prove the one-more unforgeability of our PHE-based Construction 9.3. We now give a construction showing how any NIBS scheme that is secure in the KOSK model can be generically upgraded to a fully secure scheme, ie., one without the KOSK assumption using a NIZKAoK. Formally, we show how to build a secure NIBS scheme (without KOSK) given a NIBS scheme, \( \text{NIBS}' = (\text{Setup}', \text{KeyGen}_S', \text{KeyGen}_R', \text{Issue}', \text{Obtain}', \text{Verify}') \) that is secure in the KOSK model.
### Language $L_6$

**Instance**: Each instance $x$ is interpreted as a public key $pk'$ and a bit $b \in \{0, 1\}$.

**Witness**: Witness $\omega$ consists of a secret key $sk'$, and randomness $\rho \in \{0, 1\}^\lambda$.

**Membership**: Let $C_0, C_1$ be two circuits encoding $\text{KeyGen}_S(pp', 1^\lambda; \cdot)$ and $\text{KeyGen}_R(pp', 1^\lambda; \cdot)$ respectively. Then $\omega$ is a valid witness for $x$ if the following is satisfied:

- $(sk', pk') = C_b(\rho)$

---

### Setup

The setup algorithm generates $pp' \leftarrow \text{Setup}'(1^\lambda)$, and then runs the setup algorithms for NIZK (for language $L_6$) to generate $\text{nizk.crs} \leftarrow \text{NIZK.Setup}(pp', 1^\lambda)$ and outputs it as the public parameter $pp$ of the protocol.

**KeyGen**

$\text{KeyGen}_S(pp, 1^\lambda) \rightarrow (sk, vk)$. The signer’s setup algorithm samples a random value $r_S \leftarrow \{0, 1\}^\lambda$ and runs $(sk', pk') \leftarrow \text{KeyGen}_S(pp', 1^\lambda; r_S)$. It creates a NIZK proof $\pi_S \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x := (pk', 0), \omega := (sk', r_S))$ for the language $L_6$. It outputs $sk := sk'$ and $vk := (pk', \pi_S)$.

**KeyGen**

$\text{KeyGen}_R(pp, 1^\lambda) \rightarrow (sk_R, pk_R)$. The receiver’s setup algorithm samples a random value $r_R \leftarrow \{0, 1\}^\lambda$ and runs $(sk', pk') := \text{KeyGen}_R(pp', 1^\lambda; r_R)$. It creates a NIZK proof $\pi_S \leftarrow \text{NIZK.Prove}(\text{nizk.crs}, x := (pk', 1), \omega := (sk', r_R))$ for the language $L_6$. It outputs $sk := sk'$ and $vk := (pk', \pi_R)$.

### Obtain

$\text{Issue}(sk, pk_R) \rightarrow (\text{psig}, \text{nonce})$. The issue algorithm parses the receiver’s public key as $(pk', \pi) := pk_R$ and runs the NIZK verifier $\text{NIZK.Verify}(\text{nizk.crs}, x := (pk', 1), \pi)$. If the verifier outputs 0, the signer aborts. Otherwise it continues to execute the issue algorithm.

**Obtain**

$\text{Obtain}(sk_R, vk, psig, nonce) \rightarrow (\mu, \sigma)$. The obtain algorithm parses the signer’s public key as $(pk', \pi) := vk$ and runs the NIZK verifier $\text{NIZK.Verify}(\text{nizk.crs}, x := (pk', 0), \pi_S)$. If the verifier outputs 0, the receiver aborts. Otherwise it continues to execute the obtain algorithm.

### Verify

$\text{Verify}(vk, \mu, \sigma) \rightarrow \{0, 1\}$. The verification algorithm runs $\text{Verify}'(vk, \mu, \sigma)$ and outputs whatever it outputs.

It is intuitively obvious that if NIBS’ is a secure NIBS in the KOSK model and NIZK is an argument of knowledge, then the above construction $\text{NIBS} = (\text{Setup}, \text{KeyGen}_S, \text{KeyGen}_R, \text{Issue}, \text{Obtain}, \text{Verify})$ is secure in the standard model. Essentially, in the security proof, instead of receiving the adversary’s secret key (as in the KOSK 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 NIBS’.