Non-Interactive Blind Signatures from RSA Assumption and More

Lucjan Hanzlik[1][0009−0006−1941−693X], Eugenio Paracucchi[1,2][0009−0004−5919−226X], and Riccardo Zanotto[1,2][0009−0001−2086−8398]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
{firstname.surname}@cispa.de
[2] Saarland University, Saarbrücken, Germany

Abstract. Blind signatures have received increased attention from researchers and practitioners. They allow users to obtain a signature under a message without revealing it to the signer. One of the most popular applications of blind signatures is to use them as one-time tokens, where the issuing is not linkable to the redeeming phase, and the signature under a random identifier forms a valid token. This concept is the backbone of the Privacy Pass system, which uses it to identify honest but anonymous users and protect content delivery networks from botnets.
Non-interactive blind signatures for random messages were introduced by Hanzlik (Eurocrypt'23). They allow a signer to create a pre-signature with respect to a particular public key, while the corresponding secret key can later be used to finalize the signature. This non-interaction allows for more applications than in the case of blind signatures. In particular, the author suggested using regular PKI keys as the recipient public key, allowing for a distribution of one-time tokens to users outside the system, e.g., to public keys of GitHub users, similar to airdropping of cryptocurrencies. Unfortunately, despite introducing this concept, the paper fails to provide schemes that work with keys used in the wild.
We solve this open problem. We introduce a generic construction of non-interactive blind signatures that relies on Yao's garbled circuit techniques and provide particular improvements to this generic setting. We replace oblivious transfer with their non-interactive variant and show how to construct them so that the recipient's public key, encoding the OT choice, is a standard RSA public key (e, N). To improve the efficiency of the garbling, we show how to garble the signing algorithm of the pairing-based Pointcheval-Sanders (PS) signatures and the RSA-based signature scheme with efficient protocols by Camenisch and Lysyanskaya. Our technique also apply to the well-known BBS signatures. All our improvements are of independent interest and are central to our contribution.

1 Introduction

Recently, blind signatures have received increased attention from researchers and practitioners. This cryptographic primitive allows a user to obtain a signature

under a message of her choosing from the signer using an interactive protocol. The critical property of blind signatures called *blindness*, ensures that the signer does not learn anything about the user's message. On the other hand, the user cannot create a valid signature without the signer, which forms the so-called *unforgeability* notion.

In his seminal work, Chaum [12] introduced the main application of blind signatures, i.e., electronic cash. The scenario proposed by Chaum includes a user, a merchant, and a bank. The bank plays the role of the signer in the blind signature scheme. To obtain an electronic coin, the user picks an identifier uniformly at random, which she uses as the message in the blind signature issuing process. The bank gives a signature under the user's message, which allows the user to create a valid e-coin formed by the unique identifier and a signature under it. The user can now send this coin to the merchant, who can use the bank's public key to check the coin's validity. Later, the coin can be redeemed for real currency by sending it to the bank. The bank keeps a list of used identifiers to prevent double-spending. Chaum provided more advanced protection mechanisms against double-spending, but this base scenario is already very practical.

In particular, this concept is one of the backbones of the Privacy Pass system [15], which is currently discussed as part of the IETF standard [11]. One of the main problems tackled by this system is the problem of CAPTCHA challenges. Content delivery networks (CDN) apply different heuristics and policies to incoming communication to protect against bot networks and DDoS attacks. Those algorithms usually heavily challenge honest users who use anonymous networks like VPN and TOR, which diminishes the usability of the internet for those clients. Privacy Pass solves this problem using so-called tokens (or tickets), which correspond to the e-coin in Chaum's banking scenario. Clients receive those tokens from an issuer and can later redeem them when communicating with the CDN instead of solving the CAPTCHA. From the user's perspective, no interaction is required; the whole redeeming process is done via the user's browser (i.e., browser extension[3]). Privacy Pass supports using the RSA-PSS blind signature scheme [6] as the underlying algorithm. Lysyanskaya showed the security of this blind signature scheme in this context [23]. Alternatively, Privacy Pass can be instantiated as follows. Suppose the issuer is also the party redeeming the token. In that case, the issuer can rely on an oblivious pseudorandom function instead, which can also be seen as blind signatures with a designated verifier.

Hanzlik [21] observed that the identifier picked by the user in e-cash-like applications (e.g., Privacy Pass) is a uniformly random value and does not follow any particular distribution. This observation gave birth to a new primitive called non-interactive blind signatures for random messages (NIBS). The signer can use a public key corresponding to the user and a nonce to create a pre-signature. The user can later finalize the pre-signature to a complete signature. As part of this process, she receives a random identifier under which the full signature is valid. In other words, in non-interactive blind signatures, the user cannot pick the signed message, but it is an output of the finalization process. The advantage

[3] https://privacypass.github.io

is that no interaction between the user and the signer is required. The latter only needs the former's public key. The message is an output of a pseudorandom function that takes as input the user's secret key and a nonce picked by the signer. Thanks to that, the final message-signature pair does not leak who the recipient of the pre-signature was but also which nonce was used to create it, making the scheme blind. Formally, the former is called *recipient blindness*, and the latter property is called *nonce blindness*.

One of the exciting applications of NIBS that cannot be used with standard blind signatures is the concept of airdropping. Hanzlik proposes instantiating NIBS so that the user's public key corresponds to a long-term public key for other schemes (e.g., keys used for PKI primitives or GitHub public keys). Airdropping of cryptocurrencies is a known technique to distribute to users who can but are not required to redeem the currency. Boneh et al. [31] formally defined this concept, which also preserves the privacy of the recipients. NIBS can be used to airdrop e-coins (e.g., Privacy Pass tokens), allowing issuers to create pre-signatures, e.g., for a selected group of GitHub public keys, and distribute them. Thanks to the blindness property, the issuer cannot distinguish which of the chosen users redeemed the e-coin and which did not.

The main scheme proposed in [21] supports standard public keys in the discrete logarithm setting g^x but requires the public key to be a point on an elliptic curve that supports pairings, e.g., the BLS12-381 curve [30]. Unfortunately, due to the specifics of the scheme construction, it cannot be used with public keys for the BLS signature scheme [8]. In other words, the composition of the BLS and NIBS schemes would be insecure, which, as shown in the paper, is different for, e.g., Schnorr and ECDSA signatures. However, in practice, we use more efficient curves with those schemes that do not support pairings. Thus, on the one hand, composing BLS and NIBS is insecure, while on the other hand, all publicly available keys for Schnorr and ECDSA cannot be used. The same problem is with the generic scheme proposed in [21], which requires that the user's public key is a key for a verifiable random function [24]. Consequently, the airdropping application cannot be used with any of the schemes introduced in [21]. Therefore, in this paper, we consider the following research question.

Can we construct non-interactive signature schemes composable with public keys and schemes in the wild? In particular, can we airdrop Privacy Pass-like tokens to users with a standard RSA key (e, N)*?*

1.1 Our Contribution

We positively answer this question and show how we can securely construct non-interactive blind signatures while the signer can use users' standard PKI RSA public keys. Our construction leverages Yao's garbled circuit idea but relies on non-interactive oblivious transfer [4]. Instead of using this construction naively, we show several improvements that are of independent interest and allow us to achieve the abovementioned result.

- Firstly, we show how to construct a non-interactive oblivious transfer scheme where we do not require a special structure of the recipient's public key (like is the case in [4]). The scheme works with standard PKI RSA public keys (e, N), and security relies on a variant of the quadratic residuosity problem and the random oracle model. Prior schemes were not able to leverage existing PKI as the recipient keys. Using this scheme as part of our non-interactive blind signature scheme allows the signer to provide the user with the input labels for the garbled circuit. Additionally, we introduce a new property that allows an extraction algorithm to output the recipient's choice in the OT protocol, which we leverage in later proofs.
- Secondly, we show how to garble particular signature schemes' signing procedures in a way that the secret key of the signer does not leak. We show how our idea works efficiently for Pointcheval-Sanders (PS) signatures [27] and the RSA-based signature scheme with efficient protocols [9]. Interestingly, we can extend our techniques to other schemes, including BBS [7,10] and signatures where the part of the signature dependent on the message can be represented as $\alpha \cdot h^m$ for group element h and α independent of m.
- We show how to generically construct non-interactive blind signatures from non-interactive oblivious transfer and garbled circuits. The proposed construction is only secure against honest-but-curious signers, and therefore, we extend the formal definitions of NIBS to accommodate such adversaries. The scheme security can be elevated by the signer providing a zero-knowledge proof of honest behavior.
- Our final contribution is a fully secure construction of non-interactive blind signatures. The generic construction inspired the scheme, but instead of using black-box access to the underlying primitives, we leverage their internal workings. The scheme can be used to airdrop PS signatures to users with standard RSA public keys (e, N) at a cost of approximately 20 MB for 80-bit security.

1.2 Our Techniques

In this subsection, we provide a high-level overview of our techniques and leave the details for later. We start with the following observation. To construct non-interactive blind signatures, we need some way for the signer to send data obliviously to the user. A natural candidate for this is oblivious transfer (OT), which allows the user to pick one out of many messages sent by the signer (e.g., choose one of two messages based on a bit). The security of OT guarantees that the signer is oblivious to the user's choice. OT protocols are inherently interactive and require the recipient to send messages to the sender. Fortunately, Bellare and Micali [4] introduced the notion of non-interactive oblivious transfer (NIOT). The idea is to replace the communication with the recipient's public key. The sender uses the public key to encode the messages it wants and does not need further communication with the receiver. In other words, a NIOT can be seen as a two-move OT, where the first message can be reused and is set to be the recipient's public key.

Scheme	λ	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{pk}_R\rvert$	$\lvert\mathsf{psig}\rvert$	$\lvert\mathsf{sig}\rvert$	Blindness
$\mathsf{Generic}_4$	80	2292 [bit]	2048 [bit]	$\approx$ 3247 [kB]	764 [bit]	Honest
$\mathsf{Generic}_5$	80	8192 [bit]	2048 [bit]	$\approx$ 3283 [kB]	$\lvert\pi\rvert$ [bit]	Honest
$\mathsf{Generic}_4$	128	2292 [bit]	3072 [bit]	$\approx$ 12402 [kB]	764 [bit]	Honest
$\mathsf{Generic}_5$	128	12288 [bit]	3072 [bit]	$\approx$ 12486 [kB]	$\lvert\pi\rvert$ [bit]	Honest
NIBPS	80	2292 [bit]	2048 [bit]	$\approx$ 20484 [kB]	764 [bit]	Malicious
NIBPS	128	2292 [bit]	3072 [bit]	$\approx$ 49071 [kB]	764 [bit]	Malicious

Table 1. Concrete efficiency of our generic construction and the NIBPS (Scheme 9). We will use $\mathsf{Generic}_a$ to denote Scheme 6 instantiated with NIOT Scheme 3 (batched version as per Remark 5) and garbled circuit Scheme a. We use the BLS12-381 curve for all schemes to instantiate the groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ independent of the security parameters λ. For BLS12-381 we have: p with 381-bit, q with 255-bit representation, elements of $\mathbb{G}_1$ with 382-bit and elements of $\mathbb{G}_2$ with 764-bit representation. We assume an RSA modulus of size 2048-bit for 80-bit security and 3072-bit for 128-bit security. Additionally, we assume that message size is $\ell_m = \lambda$ for the generic scheme and π denotes the bit size of the proof of knowledge of a signature from [9], which depends on the actual instantiation.

Generic Solution. Using NIOT, the signer can send one out of many messages to the user. The first approach would be for the signer to sign all possible messages and use the non-interactive oblivious transfer to send it to the user. Unfortunately, there are two problems with this approach. Firstly, the message space would have to be polynomial in size. Otherwise, the signer would have to sign exponentially many messages. Secondly, the scheme would not be blind and usable since the user would always pick a signature under the same message (i.e., the recipient's secret key). Note that the user's choice in the OT protocol is fixed by its public key.

Our approach instead is to leverage Yao's garbled circuits [32], which work well with OT protocols and are one of the ways to implement multi-party computation (MPC). Garbled circuits allow a garbler (in our scenario, the signer) to generate an encoded version of a circuit $f(\cdot)$ that can be shared with the recipient. To evaluate the encoded circuit $F(\cdot)$, the recipient must ask the signer for an encoding of its input sk_R with encoding bit choices $\bar{\mathsf{sk}}_{R,1}, \ldots, \bar{\mathsf{sk}}_{R,\ell}$. Both parties run the OT protocols ℓ times, where for each run, the sender provides encodings for both bits at a given position $i \in [\ell]$, while the user only learns a so-called label for its bits $\bar{\mathsf{sk}}_{R,i}$. All labels for each bit then form the encoding X of the input sk_R. The recipient can now evaluate the garbled circuit F and receive the output $f(\mathsf{sk}_R)$.

The second approach to construct non-interactive blind signatures is to leverage Yao's techniques. We first replace the OT with a non-interactive OT (NIOT), leading to the secret input of the recipient being now the user's private key sk_R. The next step is to take a standard signature scheme with the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ and set the circuit to $f_{\mathsf{sk}} = \mathsf{Sign}(\mathsf{sk}, \mathsf{sk}_R)$. The pre-signature is the garbled circuit and the NIOT protocol for encoding the secret key sk_R. Unfor-

tunately, this would lead to a NIBS scheme without the reusability property, i.e., a recipient can only receive one signature. To solve the problem, we observe that in our NIOT scheme, we do not commit to only one secret key but can commit to many, and the key choice can be picked pseudorandomly based on a nonce nonce. We therefore evaluate the circuit $f_{\mathsf{sk},\mathsf{nonce}} = \mathsf{Sign}(\mathsf{sk}, \mathsf{sk}_{R,\mathsf{nonce}})$. The recipient uses the secret key $\mathsf{sk}_{R,\mathsf{nonce}}$ to execute the NIOT protocol and runs the garbled circuit using the received encoding. As a result, the user receives a signature under the message $\mathsf{sk}_{R,\mathsf{nonce}}$), which we will treat as a one-time use key. To redeem/finalize the NIBS, the recipient must make sure that the evaluated function $f_{\mathsf{sk},\mathsf{nonce}}$ cannot be used to link the pre-signature to the final signature. We will show that the recipient can use either the re-randomization properties of the signature scheme or an efficient protocol to show knowledge of the signature.

There are three problems with the above approach. First of all, the efficiency of garbling and execution of the garbled circuit depends heavily on the garbled circuit $f(\cdot)$. Garbling the signing process usually requires a circuit for modular arithmetic and more (e.g., elliptic curve operations). Secondly, a malicious signer can easily break the blindness property by providing malformed inputs to the OT transfer protocol. Consequently, the recipient can only reconstruct a valid signature in certain situations, e.g., if $\bar{\mathsf{sk}}_{R,i} = 1$ and not in case $\bar{\mathsf{sk}}_{R,i} = 0$, which gives the malicious signer an oracle on the bits of the user's secret. These attacks are known as selective-failure attacks in the multi-party-computation literature [22,29]. Fortunately, this problem, as we will see later, can be easily solved by an honest signer definition and later a folklore transformation from an honest signer to a malicious signer using zero-knowledge proofs. In other words, we first will assume an honest signer and later have it prove that it behaved honestly using ZK proofs. To do that, we introduce two new notions called *honest recipient blindness* and *honest nonce blindness*. We assume that instead of outputting the pre-signature, the adversary must output its secret signing key and the random coins used to allow the challenger to generate the pre-signatures in an explainable and honest way. The third problem is that Bellare and Micali only provide NIOT schemes, where the receiver's public key is in a particular form to work with the NIOT scheme. However, the main goal of this paper is to provide a NIBS scheme with airdropping that supports public keys existing in the wild. Finally, once we directly apply the abovementioned improvements to the generic construction, we end up with a NIBS scheme that is efficient and allows using a regular RSA public key as the recipient's key pair.

Non-interactive Oblivious Transfer for RSA Public Keys. We will now describe how we construct a 1-out-of-2 non-interactive oblivious transfer using a standard RSA public key for the RSA signature and encryption scheme. Such a key is of the form (e, N), whereas for the private key d, we have $e \cdot d \equiv 1 \pmod{\phi(N)}$. Unfortunately, this key cannot be used with the Bellare and Micali non-interactive oblivious transfer scheme since it lacks the elements β_0 and β_1, encoding the user's choice for the OT (i.e., message m_0 or m_1). The main problem is that the user generates the key (e, N) without the oblivious transfer protocol in mind since this key is used for other primitives.

What we want to achieve is that this key must somehow encode the user's choice in a way that the sender cannot distinguish. A potential hard problem is whether an element in $\mathbb{Z}_N$ with Jacobi symbol 1 is a quadratic residue or a quadratic non-residue. In particular, we can associate the choice m_0 when x is a quadratic residue, while m_1 when x is not a square. Given such a random element x, the sender cannot distinguish, by QR, which of the two messages, m_0 or m_1, is picked by the receiver. This random element can be computed as the random oracle $\mathsf{H}_N(N)$ output. Note that this produces a single bit choice but can easily be extended to more bits and have λ independent bit choices by using $\mathsf{H}_N(N, i)$ for $i \in [\lambda]$. Such a string of bits can form the recipient's secret key sk_R. Note that this idea can be extended to many secret strings by using an additional nonce in the computation $\mathsf{H}_N(N, \mathsf{nonce}, i)$.

Given this generic idea, we will now show how to utilize it. The main problem is how to encrypt the messages m_0 and m_1 and how the receiver can decrypt its choice. We turn our attention to the Goldwasser-Micali cryptosystem. Given a quadratic non-residue x and modulus N (public key (x, N)), one can encrypt a bit b as follows. The ciphertext is $\mathsf{ct} = y^2 \cdot x^b \mod N$ for y chosen from the group of units modulo N. To decrypt, the recipient checks if ct is a quadratic residue (bit $b = 0$) or a quadratic non-residue (bit $b = 1$). The scheme can be easily extended to a larger message string by repeating the encryption process for the other message bits and constructing a bigger ciphertext. In our discussion, we will only consider the Goldwasser-Micali cryptosystem for strings of bits.

Unfortunately, this cryptosystem itself does not solve our problem. We want to achieve that the message the recipient can decrypt should depend on the element $\mathsf{H}_N(N)$. We solve this using ideas from dual-mode non-interactive zero-knowledge [20]. We notice that depending on the element x, the Goldwasser-Micali cryptosystem acts as an encryption scheme or a hiding commitment scheme. If x is a quadratic non-residue, we have the standard cryptosystem. In contrast, in the opposite case, the ciphertext ct is always a square, independent of the encrypted bit b, providing a hiding property even if the recipient knows the secret key. The first approach would be for the sender to use $\mathsf{H}_N(0, N)$ as the x parameter for the Golwasser-Micali cryptosystem to encrypt m_0 and $\mathsf{H}_N(1, N)$ to encrypt m_1. Unfortunately, this could lead to situations where the recipient can decrypt both m_0 and m_1 or neither. We must use the element $\mathsf{H}_N(N)$ as a switch between m_0 and m_1. We will show how to do that below.

To accommodate for all cases of modulus N, we first observe that the Cocks IBE scheme [13] can be used as a counterpart to the Goldwasser-Micali cryptosystem, i.e., while GM requires quadratic non-residues to work, the Cocks IBE scheme uses quadratic residues as a part of the public key. We notice that in case N is squarefree, the same argument as in the case of GM can be made for Cocks IBE, i.e., if the public key is a quadratic non-residue, then ciphertext is a perfectly hiding commitment. To construct a NIOT, we then use the Goldwasser-Micali with $\mathsf{H}_N(N)$ to encrypt message m_0 and the Cocks IBE scheme to encrypt m_1 using the same element $\mathsf{H}_N(N)$. To prove that N is squarefree, we use the

techniques from [18] in a non-interactive manner, similar to, in some sense, a witness encryption scheme.

A related approach was considered in [25]. However, adapting [25] for NIOT presents challenges, particularly in ensuring security against malicious receivers and maintaining compatibility with existing keys. The scheme relies on an honestly generated modulus, which a malicious receiver could manipulate to decrypt both messages. Mitigating this would require additional NIZK proofs, conflicting with our design goals. Instead, we leverage the duality between the Cocks and Goldwasser-Micali schemes, offering a novel approach that overcomes these limitations.

Efficient Garbling of Signing Functions. Constructing OPRFs from OT is well-known, but extending this concept to the non-interactive setting introduces new challenges, particularly in efficiently handling signature schemes. Our starting point are Pointcheval-Sanders (PS) signatures [27] in the discrete logarithm setting. The scheme looks as follows. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing, where all groups are of order q and are generated by respectively $g_1, g_2, e(g_1, g_2)$. Moreover, let $X = g_2^x$ and $Y = g_2^y$ be part of the public key. A signature under message m is then of the form $(\sigma_1, \sigma_2) = (h, h^{x+y\cdot m})$, where $m_1 \dots m_\ell$ is the bit encoding of the message m and h is a random element from $\mathbb{G}_1$. Using the pairing function e, one can easily verify the signature by checking that $e(\sigma_1, X \cdot Y^m) = e(\sigma_2, g_2)$. Naively garbling such a scheme would require the garbling party to implement all the operations for groups $\mathbb{G}_1$, which would lead to severe inefficiency. Fortunately, a different approach exists for the garbling instead of taking a circuit for the group operations. The σ_2 part of the signature can be rewritten as $h^x \cdot \prod_{i=1}^{\ell} (h^y)^{2^{i-1}\cdot m_i}$, where now it is a product of h^x, and elements $(h^y)^{2^{i-1}\cdot m_i}$, which depend on the bits of the message and h^y. In other words, we can write down a valid message for a signature as $s_0 \cdot \prod_{i=1}^{\ell} s_i$, where $s_0 = h^x$ and

$$s_i = \begin{cases} 1_{\mathbb{G}_1} & \text{for } m_i = 0 \\ (h^y)^{2^{i-1}} & \text{for } m_i = 1 \end{cases}$$

With this, our approach should already be visible. We have a constant share s_0 and shares s_i that depend on the message. The idea is for each $i \in [\ell]$ to run the oblivious transfer protocol on decryption keys for either $1_{\mathbb{G}_1}$ (bit 0) or $(h^y)^{2^{i-1}}$ (bit 1), where the sender randomly chooses $h \leftarrow\!\!\$\, \mathbb{G}_1$. Unfortunately, this garbling would not be secure since an adversary can easily flip bits of a message from 1 to 0 by not including the factor $(h^y)^{2^{i-1}}$ in the computation of σ_2. What we need to ensure is that the only operation the recipient can do with $s_0, s_1, \dots, s_\ell$ is to compute the product $\sigma_2 = s_0 \cdot \prod_{i=1}^{\ell} s_i$. The folklore technique from multi-party computation to achieve this is to use a multiplicative secret sharing of $1_{\mathbb{G}_1}$ on top. In other words, let $\prod_{i=0}^{\ell} a_i = 1_{\mathbb{G}_1}$ be a simple multiplicative secret sharing of the neutral element in $\mathbb{G}_1$. The idea is now to use $s'_0 = s_0 \cdot a_0$ and

$$s'_i = \begin{cases} a_i & \text{for } m_i = 0 \\ (h^y)^{2^{i-1}} \cdot a_i & \text{for } m_i = 1 \end{cases}$$

The garbled circuit is composed of the value s'_0, σ_1 and the ciphertexts for plaintexts a_i and $(h^y)^{2^{i-1}} \cdot a_i$. We use authenticated encryption to ensure the evaluator gets the correct share s'_i since we permute the ciphertext order to provide privacy. The corresponding keys form the encoding vector, i.e., in case the recipient chooses message m in the OT protocol, it receives keys $k_1^{m_1}, \ldots, k_\ell^{m_\ell}$ and can use them to find and decrypt the correct ciphertext. Security follows by the simulator encoding $s'_0 = \sigma_2 \cdot a_0$ and $s'_i = a_i$ for all $i \in [\ell]$, which is indistinguishable based on the secret sharing. One attractive property of PS signatures is that the evaluator can later re-randomize the signature by computing $(\sigma'_1, \sigma'_2) = (\sigma_1^r, \sigma_2^r)$ for some $r \in \mathbb{Z}_q$. We use this property in the NIBS construction mentioned above to ensure that pre-signatures (containing the garbled circuit) are independent of the final NIBS signature, which is in the form of the PS signature.

The idea works nicely with PS signatures since the message is in the exponent. Applying the same technique to standard RSA signatures is not possible using this technique. However, we notice the RSA-based signature scheme from [9] not only uses the RSA assumption for security but also provides a similar property to PS re-randomization, i.e., where the recipient can efficiently prove knowledge of a signature (e, s, v) in zero-knowledge. The scheme itself requires the signer to compute prime e, number s, and value v such that $v^e = a^m \cdot b^s \cdot c \mod N$, where N is an RSA modulus constructed from safe primes and a, b, c are quadratic residues that are part of the public key. We will show how to garble the computation of v since it is the only part of the signature that depends on the message. Notice that similar to the h element in PS signatures, the signer picks e, s as part of the signing process, which can lead to breaking the blindness of the NIBS scheme. However, we already mentioned that the scheme provides efficient ZK proofs of signature knowledge that can solve this problem.

To garble the scheme from [9], we observe that v can be computed as $\alpha \cdot \beta$, where $\alpha = (b^s \cdot c)^{1/e} \mod N$ and $\beta = \prod_{i=1}^{\ell} (a^{1/e})^{2^{i-1} \cdot m_i}$. Thus, we can apply the same strategy as in the case of PS signatures, i.e., compute $s'_0 = \alpha \cdot a_0$ and

$$s'_i = \begin{cases} a_i \text{ for } m_i = 0 \\ (a^{1/e})^{2^{i-1}} \cdot a_i \text{ for } m_i = 1 \end{cases}$$

One thing we notice here is that all computations are performed in the group of quadratic residues, so instead of picking a secret sharing of 1, we pick $a_0, \ldots, a_\ell$ form a multiplicative secret sharing of 4 and later divide the result by 4. This ensures that all $a_0, \ldots, a_\ell$ are in QR_N.

Interestingly, our techniques are not limited to the above scheme and can be applied to any signature scheme where we can pinpoint a part of the signature that depends on the message m. For such a part, we have $\alpha \cdot h^m$ for some h and α independent of m. In particular, the well-known BBS signature [7,10] falls into this category. Given public key $X = g_2^x$ and public $h \in \mathbb{G}_1$, the signature is (A, e), where $A = (g_1 \cdot h^m)^{1/(x+e)}$. It is easy to see that we can garble the computation of A similarly to PS signatures, i.e., A can be rewritten as $A = s_0 \cdot \prod_{i=1}^{\ell} s_i$, where $s_0 = g_1^{1/(x+e)}$ and $s_i = (h^{1/(x+e)})^{2^{i-1} \cdot m_i}$. The BBS signature scheme also supports efficient protocols and can be used in our generic construction.

Efficient Construction. The main reason why our generic construction is only secure against honest-but-curious adversaries is that it is susceptible to selective-failure attacks [22], where the adversary can encode false NIOT labels or payloads so that the Obtain algorithm aborts with $\perp$ for some messages while works for other. Such an attack can be used to identify a bit of one of the messages from the final signatures, leading to an attack against both recipient and nonce blindness.

Our first idea is to construct the scheme so that the final signature is not under m (where bits of m correspond to the inputs to the garbled circuits) but under $\mathsf{H}(m)$ for a random oracle H. Assuming m is long enough, even if the adversary learns up to λ bits using the selective failure attacks, the final message would still be uniformly random. Unfortunately, the problem is that constructing a garbled circuit for a hash function would be inefficient and have unknown security implications since we would have to treat a circuit representation of the hash function as a random oracle.

To solve the problem, we construct a concrete scheme directly by utilizing the building blocks in a non-black-box manner. The idea is to replace the random oracle H above with an algebraic universal hash function. Such functions are known to be constructible in fields. We, therefore, fix the underlying signature scheme to PS signatures that work over groups with prime order q (the same idea works for BBS). We also increase the input from one message of size ℓ_q, where ℓ_q is the bit-size of q, to $2 \cdot \ell_q$. Now given inputs x_1, x_2 both of size ℓ_q bits, the final PS will be under message $m = \alpha_1 \cdot x_1 + \beta_1 + \alpha_2 \cdot x_2 + \beta_2$, where $\alpha_1, \alpha_2, \beta_1, \beta_2$ will be outputs of a random oracle to ensure independence with x_1, x_2. The other parts of the NIBS scheme are the same as in the generic construction, i.e., the garbled signing values are encrypted using AE, where the corresponding keys are sent to the recipient using the ideas from our NIOT constructions.

The proof of blindness follows through the leftover hash lemma. Since the function we use to construct the message m is a universal hash function, it follows from the LHL that m is indistinguishable from random for any adversary. We, therefore, can make the final signature disjoint of the issuing process. The only problem we encounter is that to apply the LHL, we must ensure that x_1 and x_2 are unpredictable to even an unbounded adversary. Unfortunately, this is false since x_1 and x_2 can be computed by factorizing the recipient's public key N. To solve this problem, we have to make the choices of the NIOT independent of the actual values encoded in N. To do so, we identify the positions where the adversary "cheats" by using random oracle queries and decrypting all values. Once we identify those positions, we replace the other ones with random bit choices instead of using the corresponding values encoded in N. This change will be indistinguishable for an adversary under the quadratic residuosity problem. Note that this step will be similar to known techniques of applying the LHL, where a function is first replaced by its lossy version before using the LHL. In the end, at least $2 \cdot \ell_q - \lambda$ bit of the input x_1, x_2 are unpredictable to the adversary, where it can learn up to λ bits through the selective failure attack, and for those bits, we still rely on the encoding in N. Assuming $\ell_q \approx 2 \cdot \lambda$, the final message m will be uniformly random for the adversary.

The construction is possible only because we treat the building blocks in a non-black-box manner, since in the generic construction, the selective failure attack can be applied on the NIOT part or the garbled circuit part, while here, we can consider both simultaneously. In the end, with only a cost of 2 times for input bits, we can construct a NIBS scheme that supports the standard blindness definitions. We provide more details on the construction in Section 7.

2 Preliminaries

2.1 Basic Primitives, Notions and Assumptions.

We denote by $y \leftarrow \mathcal{A}(x;r)$ the execution of algorithm $\mathcal{A}$ on input x, optional random coins r and with output y. By $r \leftarrow\!\$\ S$ we mean that r is chosen uniformly at random over the set S. We will use $1_{\mathbb{G}}$ to denote the identity element in group $\mathbb{G}$ and $[n]$ to denote the set $\{1,\ldots,n\}$. Throughout the paper we will use the multiplicative notation and by $\mathcal{A}^{\mathcal{O}}$ we denote an algorithm $\mathcal{A}$ that has access to oracle $\mathcal{O}$. For a positive integer N, we will denote by $\mathbb{Z}_N$ the ring of integers modulo N and with $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \mid \gcd(x,N)=1\}$ its units. For integers n, m we will denote the Jacobi symbol by $\left(\frac{n}{m}\right)$. We define $\mathcal{J}_N = \{x \in \mathbb{Z}_N^* \mid \left(\frac{x}{N}\right) = 1\}$ and $\mathrm{QR}_N = \{x \in \mathbb{Z}_N^* \mid \exists\ y \text{ s.t. } x = y^2\}$. We have the following inclusions: $\mathrm{QR}_N \subseteq \mathcal{J}_N \subseteq \mathbb{Z}_N^*$. An integer N is said *squarefree* if for all prime $p \mid N$ we have $p^2 \nmid N$.

Definition 1 (Guessing Probability). *Let $\mathcal{X}$ be a finite set and let $D_{\mathcal{X}}$ be some distribution on $\mathcal{X}$. Then, we define the guessing probability of $D_{\mathcal{X}}$ to be $\gamma = \max_{x^* \in \mathcal{X}} \Pr_{x \leftarrow\!\$ D_{\mathcal{X}}}[x^* = x]$.*

Definition 2 (Universal Hash Function). *A hash function $\mathsf{H} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is universal if for every two distinct elements $x, x' \in \mathcal{X}$, we have*

$$\Pr_{\mathsf{pk} \leftarrow\!\$ \mathcal{K}}[\mathsf{H}(\mathsf{pk}, x) = \mathsf{H}(\mathsf{pk}, x')] \leq \frac{1}{|\mathcal{Y}|}.$$

Definition 3 (Simplified Leftover Hash Lemma). *Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets, and let $\mathsf{H} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a universal family of hash functions. Let $D_{\mathcal{X}}$ be some distribution on $\mathcal{X}$ with guessing probability at most γ. Then, for any (unbounded) adversary $\mathcal{A}$, we have:*

$$|\Pr[\mathcal{A}(\mathsf{pk}, \mathsf{H}(\mathsf{pk}, x)) = 1 | \mathsf{pk} \leftarrow\!\$\ \mathcal{K}, x \leftarrow\!\$\ D_{\mathcal{X}}] - \Pr[\mathcal{A}(\mathsf{pk}, y) = 1 | \mathsf{pk} \leftarrow\!\$\ \mathcal{K}, y \leftarrow\!\$\ \mathcal{Y}]| \leq \gamma \cdot |\mathcal{Y}|.$$

Definition 4 (Quadratic Residuosity Assumption (QR)). *Let p, q be distinct prime numbers and $N = pq$. Given $x \in \mathcal{J}_N$ it is hard to decide if $x \in \mathrm{QR}_N$.*

Definition 5 (Extended Quadratic Residuosity Assumption (EQR)). *Let p, q be distinct prime numbers and $N = pq$. Given $x, y \in \mathcal{J}_N$, it is hard to decide whether $x \in \mathrm{QR}_N$ or not, given that $y \notin \mathrm{QR}_N$. We will use $\mathbf{Adv}_{\mathcal{A},\mathsf{EQR}}$ as the advantage of an adversary $\mathcal{A}$ in distinguishing if $x \in \mathrm{QR}_N$ or not. Note that for Blum modulus N, this assumption is equivalent to the standard QR assumption.*

Bilinear Groups. Let us consider cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ of prime order q. Let g_1, g_2 be generators of respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a *bilinear map* (pairing) if it is efficiently computable and the following holds: 1) Bilinearity: $\forall (S, T) \in \mathbb{G}_1 \times \mathbb{G}_2$, $\forall a, b \in \mathbb{Z}_p$, we have $e(S^a, T^b) = e(S, T)^{a \cdot b}$, 2) Non-degeneracy: $e(g_1, g_2) \neq 1$ is a generator of group $\mathbb{G}_T$.

Definition 6 (Assumption 1 [27]). *For $X_1 = g_1^x$, $Y_1 = g_1^x$ and $X_2 = g_2^x$, $Y_2 = g_2^y$, where x and y are random scalars in $\mathbb{Z}_q$, we define the oracle $\mathcal{O}$ on input $m \in \mathbb{Z}_q$ that chooses a random $h \in \mathbb{G}_1$ and outputs the pair $P = (h, h^{x+m \cdot y})$ Given $(g_1, g_2, Y_1, X_2, Y_2)$ and unlimited access to this oracle, no adversary $\mathcal{A}$ can efficiently generate such a pair, with $h \neq 1_{\mathbb{G}_1}$, for a new scalar m^*, not asked to $\mathcal{O}$. We will use $\mathbf{Adv}_{\mathsf{Ass}}(\mathcal{A})$ as the advantage of $\mathcal{A}$ in breaking this assumption.*

Authenticated Encryption [5]. In this paper, we will use authenticated encryption $\mathsf{AE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, a variant of symmetric encryption that also confidentially ensures the integrity of the encryption mechanism.

Non-intereactive Zero-Knowledge We will also use non-interactive zero-knowledge proof systems for languages $\mathcal{L}_{\mathcal{R}}$ defined by the $\{\mathsf{NP}\}$ relation $\mathcal{R}$. A *non-intractive zero-knowledge proof* (NIZK) $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ consists of three PPT algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$. We will require the system to have classical notions of zero-knowledge (there exists a simulator algorithm creating valid proofs without a witness) and extractable (there exists an extractor algorithm that, given a valid proof, is able to output a valid witness for the statement).

2.2 Signature Schemes

Definition 7. *A signature scheme* SIG *consists of three PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *with the following syntax.*

$\mathsf{KeyGen}(\lambda)$**:** *On input a security parameter λ, it outputs a public and secret signing key* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$**:** *On input a key* sk *and a message m, it outputs a signature σ.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** *On input a public key* pk*, a message m and a signature σ, it outputs either 0 or 1.*

We require the following properties of a signature scheme.

Correctness: *For every security parameter $\lambda \in \mathbb{N}$ and every message m given that* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SIG.KeyGen}(\lambda)$, $\mathsf{sig} \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$ *it holds that*

$$\mathsf{SIG.Verify}(\mathsf{pk}, m, \mathsf{sig}) = 1.$$

Existential Unforgeability under Chosen Message Attacks: *Every PPT adversary $\mathcal{A}$ has at most negligible advantage in the following experiment.*

$\underline{\mathsf{EUF\text{-}CMA}_{\mathcal{A},\mathsf{SIG}}(\lambda)}$

$Q := \emptyset$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SIG.KeyGen}(\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot)}(\mathsf{pk})$
return $m^* \neq m \quad \forall m \in Q \wedge$
$\mathsf{SIG.Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$

$\underline{\mathcal{O}_1(\mathsf{sk}, m)}$

$\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$
$Q := Q \cup \{m\}$
return σ

The advantage of $\mathcal{A}$ *is defined by* $\mathbf{Adv}_{\mathsf{SIG}}(\mathcal{A}) = \Pr[\mathsf{EUF\text{-}CMA}_{\mathcal{A},\mathsf{SIG}}(\lambda) = 1]$.

Remark 1. We call a signature scheme deterministic if for all secret keys sk and messages m the signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ always outputs the same messages. The folklore way of constructing deterministic signatures is to derandomize the signing algorithm using randomness derived deterministically from the secret key sk and the message m (e.g., using randomness $\mathsf{PRF}(\mathsf{sk}, m)$). A special case of deterministic signatures is unique signatures (e.g., full-domain hash RSA scheme), where exactly one valid signature exists for a given message.

Additionally, we will require that SIG is randomizable or possesses efficient proof of possession. We will use $\mathsf{sig}' \leftarrow \mathsf{Rand}(\mathsf{sig})$ as the randomization/proving algorithm and require that randomized signatures are unlinkable to pre-randomized signatures even if the adversary provides the signature. We will denote the adversary's advantage as $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}$. Note that for schemes with efficient proofs of signature knowledge, the proof is easily distinguishable from the signature. However, we assume that the final signature is the proof itself, simplifying our considerations for the unforgeability of the signature scheme, i.e., we omit the signature extraction from the proof.

2.3 Goldwasser-Micali and Cocks Cryptosystems.

We recall the well-known Goldwasser-Micali public-key encryption [19] for which we will use the notation $\mathsf{GM} = (\mathsf{GM.KeyGen}, \mathsf{GM.Enc}, \mathsf{GM.Dec})$. We define a GM ciphertext as $\mathsf{ct} \leftarrow y^2 x^b \mod N$, where N is an RSA modulus and x is a non-residue modulo N. The bit b can be computed by checking if ct is a non-residue or a residue modulo N. We observe that if x is picked as a square modulo N, then both bits are likely to appear equivalently. In fact, if x is a square then the value $y^2 x^b$ is also a square modulo N, regardless of the bit b. We will use this observation to construct the *non-interactive oblivious transfer* presented in Section 3. In this paper, we will consider the message space of GM to be $\{0,1\}^\lambda$ and not a single bit. Such a scheme can be achieved by encrypting each bit separately and using λ basic ciphertexts. The Cocks cryptosystems [14] $\mathsf{Cocks} = (\mathsf{Cocks.KeyGen}, \mathsf{Cocks.Enc}, \mathsf{Cocks.Dec})$ is dual to the GM encryption scheme, and it works when x is a residue. The public key is (N, x) with $x = u^2$. To encrypt a bit $b = 0$ (resp. $b = 1$), sample a t in $\mathbb{Z}_N^*$ with Jacobi symbol 1 (resp. -1) and send the ciphertext $\mathsf{ct} = t + x/t$. The bit can be recomputed from ct since $\left(\frac{\mathsf{ct}+2u}{N}\right) = \left(\frac{t}{N}\right)$. See Scheme 2 for a formal definition of the scheme. We will show in Section 3 that Cocks provides similar hiding properties when x is a non-residue modulo N and N is squarefree.

$\underline{\mathsf{GM.KeyGen}(\lambda)}$**:** Generate two distinct prime numbers $p(\lambda)$ and $q(\lambda)$, set $N = pq$, and compute some non-residue $x \in \mathcal{J}_N \setminus \mathrm{QR}_N$. Return $(\mathsf{pk} = (N, x), \mathsf{sk} = (p, q))$.

$\underline{\mathsf{GM.Enc}(m, \mathsf{pk} = (N, x))}$**:** On input a message $m \in \{0,1\}^\lambda$ and a public key $\mathsf{pk} = (N, x)$, for each bit m_i of m, sample $y_i \leftarrow\!\!\$\ \mathbb{Z}_N^*$ and compute $c_i = y_i^2 x^{m_i} \pmod N$. Return the ciphertext $c = (c_1, ..., c_\lambda)$.

$\underline{\mathsf{GM.Dec}(c, \mathsf{sk} = (p, q))}$**:** On input a ciphertext $c = (c_1, \ldots, c_\lambda)$ and a secret key $\mathsf{sk} = (p, q)$, for each $i \in [\lambda]$, if $c_i \in \mathrm{QR}_N$ set $m_i = 0$, otherwise set $m_i = 1$. Return a message $m = (m_1, \ldots, m_\lambda)$.

Scheme 1: The Goldwasser-Micali encryption scheme.

$\underline{\mathsf{Cocks.KeyGen}(\lambda)}$**:** Generate two distinct prime numbers $p(\lambda)$ and $q(\lambda)$, set $N = pq$, and compute some quadratic residue $x \in \mathcal{J}_N \setminus \mathrm{QR}_N$. Return $(\mathsf{pk} = (N, x), \mathsf{sk} = (p, q, u))$ where u is a square root of x modulo N.

$\underline{\mathsf{Cocks.Enc}(m, \mathsf{pk} = (N, x))}$**:** On input a message $m \in \{0,1\}^\lambda$ and a public key $\mathsf{pk} = (N, x)$, for each bit m_i of m, sample $t_i \leftarrow\!\!\$\ \mathbb{Z}_N^*$ such that

$$\left(\frac{t_i}{N}\right) = \begin{cases} 1 & \text{if } b = 0 \\ -1 & \text{if } b = 1 \end{cases}$$

and compute $c_i = t_i + x/t_i \pmod N$. Return the ciphertext $c = (c_1, ..., c_\lambda)$.

$\underline{\mathsf{Cocks.Dec}(c, \mathsf{sk} = (p, q))}$**:** On input a ciphertext $c = (c_1, \ldots, c_\lambda)$ and a secret key $\mathsf{sk} = (p, q)$, for each $i \in [\lambda]$, compute $\alpha_i = c_i + 2u$. If $\left(\frac{\alpha_i}{N}\right) = 1$ set $m_i = 0$, otherwise set $m_i = 1$. Return a message $m = (m_1, \ldots, m_\lambda)$.

Scheme 2: The Cocks encryption scheme.

2.4 Garbled Circuits.

We will now recall the notion of garbled circuits put forward by Bellare, Hoang, and Rogaway [3]. *Garbling schemes* $G = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ are used to evaluate functions $f : \{0,1\}^n \longrightarrow \{0,1\}^m$. In particular, for any functions f, expressed as a string, and λ, the probabilistic algorithm Gb returns a triple of strings $(F, e, d) \leftarrow \mathsf{Gb}(\lambda, f)$.

- For any input string $x \in \{0,1\}^n$ the *encoding function*, $\mathsf{En}(e, x)$ defines a *garbled input* $X = \mathsf{En}(e, x)$.
- The *garbled function*, $\mathsf{Ev}(F, X)$, maps each garbled input X to a *garbled output* $Y = \mathsf{Ev}(F, X)$.
- The *decoding function*, $\mathsf{De}(d, Y)$, maps a garbled output Y to a *final output* $y = \mathsf{De}(d, Y)$.

For any input x and function f, the algorithm $\mathsf{ev}(f, x)$ evaluates f in x. We require the scheme G to be *correct* in the sense that $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = \mathsf{ev}(f, x)$. We refer to [3] for the definitions of the security notions of *privacy*, *obliviousness* and *authenticity*.

Finally, we say that a garbled scheme $G = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is a *projective scheme* if for all $f, x, x' \in \{0,1\}^n$, λ, and $i \in [n]$, when (F, e, d) is an output of $\mathsf{Gb}(\lambda, f)$, $X \leftarrow \mathsf{En}(e, x)$ and $X' \leftarrow \mathsf{En}(e, x')$, then $X = (X_1, \ldots, X_n)$ and $X' = (X'_1, \ldots, X'_n)$ are n vectors, $|X_i| = |X'_i|$, and $X_i = X'_i$ if x and x' have the same ith bit. We also expect that a corresponding output vector with similar projective properties exists.

2.5 Non-interactive Blind Signatures.

Definition 8. *A non-interactive blind signature* NIBS *scheme consists of the following PPT algorithms.*

$\mathsf{KeyGen}(\lambda)$**:** *On input security parameter λ, outputs a key pair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{RKeyGen}(\lambda)$**:** *On input security parameter λ, outputs a key pair* $(\mathsf{sk}_R, \mathsf{pk}_R)$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})$**:** *On input a secret key* sk*, user public key* pk_R *and nonce* nonce*, outputs a pre-signature* psig.

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})$**:** *On input a user secret key* sk_R*, signer's public key* pk*, pre-signature* psig *and nonce* nonce*, outputs a message-signature pair* (m, sig) *or* $\perp$.

$\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig}))$**:** *On input a public key* pk*, a message-signature pair* (m, σ) *deterministically outputs a bit* $b \in \{0,1\}$.

From a NIBS scheme, we require correctness, reusability, one-more unforgeability, recipient, and nonce blindness. Correctness works as expected, i.e., for all messages and secret keys, we can finalize an honestly generated pre-signature to a valid signature. Reusability, introduced in [1], ensures that any receiver can obtain, with high probability, two distinct messages (along with valid signatures) from pre-signature under different nonces.

Definition 9 (Reusability). *A* NIBS *scheme satisfies the reusability property, if there exists a negligible function $\epsilon(\lambda)$ such that for every λ, the following holds:*

$$\Pr\left[\begin{array}{c|c} \mathsf{nonce}_0 \neq \mathsf{nonce}_1 & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda), (\mathsf{sk}_R, \mathsf{pk}_R) \leftarrow \mathsf{RKeyGen}(\lambda), \\ \wedge & \forall b \in \{0,1\} : \mathsf{psig}_b \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce}_b), \\ m_0 = m_1 & (m_b, \sigma_b) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}_b, \mathsf{nonce}_b) \end{array}\right] \leq \epsilon(\lambda).$$

Definition 10 (One-More Unforgeability). *A* NIBS *scheme is one-more unforgeable, if for all PPT adversaries $\mathcal{A}$, their advantage defined as* $\mathbf{Adv}^{\mathsf{OM\text{-}UNF}}_{\mathcal{A},\mathsf{NIBS}} = \Pr[\mathsf{OM\text{-}UNF}_{\mathcal{A},\mathsf{NIBS}}(\lambda) = 1]$ *is negligible, where the experiment* $\mathsf{OM\text{-}UNF}_{\mathcal{A},\mathsf{NIBS}}$ *is defined as follows.*

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

$(\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$

$((m_1,\mathsf{sig}_1),\ldots,(m_\ell,\mathsf{sig}_\ell)) \leftarrow \mathcal{A}^{\mathcal{O}_1(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$

return $m_i \neq m_j$ *for* $1 \leq i < j \leq \ell \wedge$
$\mathsf{Verify}(\mathsf{pk}, m_i, \mathsf{sig}_i) = 1$ *for* $1 \leq i \leq \ell \wedge$
$k < \ell$

$\underline{\mathcal{O}_1(\mathsf{sk},\mathsf{pk}_R,\mathsf{nonce})}$

if k *not initialized* **then**
$k := 0$
$\mathsf{psig} \leftarrow \mathsf{Issue}(\mathsf{sk},\mathsf{pk}_R,\mathsf{nonce})$
$k := k+1$
return psig

Recipient blindness ensures that final signatures do not leak to the original recipient of the pre-signature. In contrast, nonce blindness ensures that two signatures to the same recipient but for different nonces are unlinkable.

Definition 11 (Recipient Blindness). *A* NIBS *scheme is recipeint blind, if for all PPT adversaries $\mathcal{A}$, their advantage defined as*

$$\mathbf{Adv}^{\mathsf{RBnd}}_{\mathcal{A},\mathsf{NIBS}} = |\Pr[\mathsf{RBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda) = 1] - 1/2|$$

is negligible, where the experiment $\mathsf{RBnd}_{\mathcal{A},\mathsf{NIBS}}$ *is defined below.*

Definition 12 (Nonce Blindness). *A* NIBS *scheme is nonce blind, if for all PPT adversaries $\mathcal{A}$, their advantage defined as*

$$\mathbf{Adv}^{\mathsf{NBnd}}_{\mathcal{A},\mathsf{NIBS}} = |\Pr[\mathsf{NBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda) = 1] - 1/2|$$

is negligible, where the experiment $\mathsf{NBnd}_{\mathcal{A},\mathsf{NIBS}}$ *is defined below.*

$\underline{\mathsf{RBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda)}$

$(\mathsf{sk}_{R_0},\mathsf{pk}_{R_0}) \leftarrow \mathsf{RKeyGen}(\lambda)$
$(\mathsf{sk}_{R_1},\mathsf{pk}_{R_1}) \leftarrow \mathsf{RKeyGen}(\lambda)$
$\mathcal{O} := \mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(\cdot,\cdot,\cdot)$
$(\mathsf{psig}_0,\mathsf{nonce}_0,\mathsf{psig}_1,\mathsf{nonce}_1,\mathsf{pk}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_{R_0},\mathsf{pk}_{R_1})$
$(m_0,\mathsf{sig}_0) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_0},\mathsf{pk},\mathsf{psig}_0,\mathsf{nonce}_0)$
$(m_1,\mathsf{sig}_1) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_1},\mathsf{pk},\mathsf{psig}_1,\mathsf{nonce}_1)$
if $\mathsf{sig}_0 = \perp$ *or* $\mathsf{sig}_1 = \perp$ **then**
$(m_0,\mathsf{sig}_0) := \perp; (m_1,\mathsf{sig}_1) := \perp$
$b \leftarrow\!\!\$\ \{0,1\}$
$\hat{b} \leftarrow \mathcal{A}((m_b,\mathsf{sig}_b),(m_{1-b},\mathsf{sig}_{1-b}))$
return $b = \hat{b}$

$\underline{\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(b,\mathsf{pk},(\mathsf{psig},\mathsf{nonce}))}$

$(m,\mathsf{sig}) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_b},\mathsf{pk},\mathsf{psig},\mathsf{nonce})$
return (m,sig)

$\underline{\mathsf{NBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda)}$

$(\mathsf{sk}_R,\mathsf{pk}_R) \leftarrow \mathsf{RKeyGen}(\lambda)$
$\mathcal{O} := \mathcal{O}_{\mathsf{sk}_R}(\cdot,\cdot)$
$(\mathsf{psig}_0,\mathsf{nonce}_0,\mathsf{psig}_1,\mathsf{nonce}_1,\mathsf{pk}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_R)$
$(m_0,\mathsf{sig}_0) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R,\mathsf{pk},\mathsf{psig}_0,\mathsf{nonce}_0)$
$(m_1,\mathsf{sig}_1) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R,\mathsf{pk},\mathsf{psig}_1,\mathsf{nonce}_1)$
if $\mathsf{sig}_0 = \perp$ *or* $\mathsf{sig}_1 = \perp$ **then**
$(m_0,\mathsf{sig}_0) := \perp; (m_1,\mathsf{sig}_1) := \perp$
$b \leftarrow\!\!\$\ \{0,1\}$
$\hat{b} \leftarrow \mathcal{A}((m_b,\mathsf{sig}_b),(m_{1-b},\mathsf{sig}_{1-b}))$
return $b = \hat{b} \ \wedge\ \mathsf{nonce}_0 \neq \mathsf{nonce}_1$

$\underline{\mathcal{O}_{\mathsf{sk}_R}(\mathsf{pk},\mathsf{psig},\mathsf{nonce})}$

$(m,\mathsf{sig}) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R,\mathsf{pk},\mathsf{psig},\mathsf{nonce})$
return (m,sig)

It is worth noting that despite using the original syntax from [21] we use the stronger blindness notion defined in [1], which assumes that the adversary is additionally receiving access to obtain oracles for the challenged recipients.

2.6 Non-interactive Oblivious Transfer.

An *Oblivious Transfer* (OT), introduced by Rabin [28] and later extended to the 1-out-of-2 setting by Even, Goldreich and Lempel [16], is an interactive protocol between a *sender* and a *receiver* (also called recipient). The sender holds two messages m_0 and m_1 and wants to transfer exactly one to the receiver. On the other hand, the receiver does not want the sender to know which one she received.

In some applications, interaction is a crucial part, and this leads us to the notion of *Non-interactive Oblivious Transfer* (NIOT) [4]. We require the sender to obliviously transfer a message without the receiver having to act. Usually, the recipient's choice is "encoded" in its public key pk. Here, we will consider NIOT schemes, where the public key encodes many decisions at once, and they can be chosen using an optional context parameter cnt (set to 0 by default. More formally.

Definition 13 (Non-interactive Oblivious Transfer). *A non-interactive oblivious transfer* $\mathsf{NIOT} = (\mathsf{KeyGen}, \mathsf{S}, \mathsf{R})$ *consists of the following PPT algorithms:*

$\mathsf{KeyGen}(\lambda)$**:** *On input the security parameter* λ*, outputs a public key* pk *and a secret key* sk*.*

$\mathsf{S}(m_0, m_1, \mathsf{pk}, \mathsf{cnt})$**:** *On input two messages* $m_0, m_1 \in \mathcal{M}$*, a public key* pk *and optional context* $\mathsf{cnt} \in \{0,1\}^*$*, outputs a ciphertext* ct*.*

$\mathsf{R}(\mathsf{ct}, \mathsf{sk}, \mathsf{cnt})$**:** *On input a ciphertext* ct*, a secret key* sk *and optional context* nonce*, outputs a bit* $b \in \{0,1\}$ *and a message* m*. We will use* $\bar{\mathsf{sk}}_{\mathsf{cnt}}$ *to denote the bit choice for context* cnt*.*

Correctness. *For any key pairs* $(\mathsf{pk}, \mathsf{sk})$ *outputted by* KeyGen *and for any context* cnt*, there exists a bit* b *such that for any messages* $m_0, m_1 \in \mathcal{M}$*:*

$$\mathsf{R}(\mathsf{S}(m_0, m_1, \mathsf{pk}, \mathsf{crs}), \mathsf{sk}, \mathsf{cnt}) = (b, m_b).$$

We will denote the bit choice b*, which depends on the secret key and the context, as* $\bar{\mathsf{sk}}_{\mathsf{cnt}}$*.*

Remark 2. The cnt parameter can be used to generate λ independent bit choices that we can treat as the recipient's secret key. Moreover, it can be used to encode many of such keys.

Remark 3. According to our definition, a NIOT is a two-move oblivious transfer where the first message, from the receiver to the sender, can be reused and is set to be the receiver's public key.

$\underline{\mathsf{RcvSec}_{\mathcal{A},\mathsf{NIOT}}(\lambda)}$

$Q = \emptyset;\ (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{KeyGen}(\lambda)$

$(\mathsf{ct},\mathsf{cnt},\hat{b}) \leftarrow \mathcal{A}^{\mathcal{O}(\mathsf{sk},\cdot,\cdot)}(\mathsf{pk})$

$(b,\cdot) \leftarrow \mathsf{R}(\mathsf{ct},\mathsf{sk},\mathsf{cnt})$

return $b = \hat{b} \ \wedge\ \mathsf{cnt} \notin Q$

$\underline{\mathcal{O}(\mathsf{sk},\mathsf{ct},\mathsf{cnt})}$

$Q = Q \ \cup\ \{\mathsf{cnt}\}$

$(b,m) \leftarrow \mathsf{R}(\mathsf{ct},\mathsf{sk},\mathsf{cnt})$

return (b,m)

$\underline{\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}}(\lambda)}$

$(\mathsf{pk},m_0,m_1,\mathsf{cnt}) \leftarrow \mathcal{A}(\lambda)$

$b_0 \leftarrow\!\$\ \{0,1\}, b_1 \leftarrow\!\$\ \{0,1\}$

$m_{-1} \leftarrow\!\$\ \mathcal{M}$

if $b_0 = 1$ **then** $m_0 = m_{-1}$

if $b_1 = 1$ **then** $m_1 = m_{-1}$

$\mathsf{ct} \leftarrow \mathsf{S}(m_0,m_1,\mathsf{pk},\mathsf{cnt})$

$\hat{b} \leftarrow \mathcal{A}(\mathsf{ct})$

return $b_0 \oplus b_1 = \hat{b}$

Fig. 1. Experiments for Non-Interactive Oblivious Transfer

We have two security notions: one for the sender and the other for the receiver. Using simulation-based definition is the standard way of defining those notions for plain oblivious transfer protocols. However, in the non-interactive case, we can define them by using game-based notions, and we provide definitions that will be sufficient for our applications. The receiver's choice depends on the public key, so we describe an experiment where the adversary is given the public key and must decide which of the two messages will be the receiver's output. For sender security, we provide an experiment where the adversary outputs the receiver's public key, two messages, and a context, and is then provided a NIOT ciphertext where one of the messages (for the bit that the receiver will not be able to receive) is either the one specified by the adversary or a random one. We define the experiments more formally below.

Definition 14 (Receiver Security). *A* NIOT *scheme is receiver secure if for all PPT adversaries* $\mathcal{A}$*, their advantage* $\mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}} = |\Pr[\mathsf{RcvSec}_{\mathcal{A},\mathsf{NIOT}}(\lambda) = 1] - 1/2|$ *is negligible, where the experiment* $\mathsf{RcvSec}_{\mathcal{A},\mathsf{NIOT}}$ *is defined in Figure 1.*

Definition 15 (Sender Security). *A* NIOT *scheme is sender secure if for all PPT adversaries* $\mathcal{A}$*, their advantage* $\mathbf{Adv}^{\mathsf{SndSec}}_{\mathcal{A},\mathsf{NIOT}} = |\Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}}(\lambda) = 1] - 1/2|$ *is negligible, where the experiment* $\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}}$ *is defined in Figure 1.*

In our construction, we will require another property of non-interactive oblivious transfer.

Definition 16 (Extractability). *Consider a PPT adversary* $\mathcal{A}$ *such that:*

- *outputs a* NIOT *public key,*
- *it then takes as input context* cnt *and a ciphertext* $\mathsf{ct} \leftarrow \mathsf{NIOT.S}(m_0,m_1,\mathsf{pk},\mathsf{cnt})$ *for random messages* m_0, m_1*,*
- *finally,* $\mathcal{A}$ *outputs a message* m^* *such that* $m^* = m_b$ *for some bit* b*.*

$\mathsf{IND\text{-}GM}_{\mathcal{A}}(\lambda)$	$\mathsf{IND\text{-}Cocks}_{\mathcal{A}}(\lambda)$
$(N, x, m_0, m_1) \leftarrow \mathcal{A}(\lambda)$	$(N, x, m_0, m_1) \leftarrow \mathcal{A}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$
$c^* \leftarrow \mathsf{GM.Enc}(m_b, (N, x))$	$c^* \leftarrow \mathsf{Cocks.Enc}(m_b, (N, x))$
$\hat{b} \leftarrow \mathcal{A}(c^*)$	$\hat{b} \leftarrow \mathcal{A}(c^*)$
return $b = \hat{b} \ \wedge\ x \in \mathrm{QR}_N$	**return** $b = \hat{b} \ \wedge\ x \in \mathcal{J}_N \setminus \mathrm{QR}_N$ $\wedge\ N$ squarefree

Fig. 2. Experiments for GM and Cocks indistinguishability

Suppose such an adversary $\mathcal{A}$ exists. In that case, there exists a PPT extractor $\mathsf{Extract}_{\mathsf{NIOT}}$ that takes the same input (and potentially randomness) as $\mathcal{A}$ and outputs the same message m^* and the NIOT public key pk together with the corresponding secret key sk. We will use $\mathbf{Adv}^{\mathsf{Extract}}_{\mathcal{A},\mathsf{NIOT}}$ to denote the extractor's probability of failing to output sk. For a clear definition, we omit auxiliary information the adversary $\mathcal{A}$ might take as input or generate as output.

3 NI Oblivious Transfer for Standard RSA Keys

This section presents one of our main contributions, a non-interactive OT built on the Goldwasser-Micali and Cocks encryption schemes. Before giving the complete description, we formalize some properties needed for our construction.

3.1 Two indistinguishability properties

We have noticed in the preliminaries that both Goldwasser-Micali and Cocks schemes satisfy some hiding properties. Consider the experiments defined in Figure 2, then we have the following proposition.

Proposition 1. *For any adversary $\mathcal{A}$ (possibly unbounded) we have:*

1. $\Pr[\mathsf{IND\text{-}GM}_{\mathcal{A}}(\lambda) = 1] = \frac{1}{2}$;
2. $\Pr[\mathsf{IND\text{-}Cocks}_{\mathcal{A}}(\lambda) = 1] = \frac{1}{2}$.

Proof. We will show this result for the special case N squarefree (and will be sufficient for our results). It is enough to show the proposition for bit messages.

1. Let $N = p_1 \cdots p_k$ be a squarefree integer and $x \in \mathbb{Z}_N^*$ a quadratic residue. Let b be a bit. We show that $c \leftarrow \mathsf{GM.Enc}(b, (N, x))$ is a randomly distributed square. Let $u \in \mathrm{QR}_N$ be a quadratic residue, we have that

$$\begin{aligned}\Pr[c = u \mid c \leftarrow \mathsf{GM.Enc}(b, (N, x))] &= \Pr[y^2 x^b = u \mid y \leftarrow\!\!\$\ \mathbb{Z}_N^*)] \\ &= \Pr[y^2 = u x^{-b} \mid y \leftarrow\!\!\$\ \mathbb{Z}_N^*)] \\ &= \frac{2^k}{|\mathbb{Z}_N^*|} = \frac{1}{|\,\mathrm{QR}_N\,|}.\end{aligned}$$

To prove the second statement we need the following lemma:

Lemma 1. *Let N be an odd, squarefree, composite integer, and let $a \in \mathcal{J}_N \backslash \mathrm{QR}_N$ be a nonsquare with Jacobi symbol one. For $t \in \mathbb{Z}_N^*$, define $s = t + a/t$ and denote by X the set of solution in $\mathbb{Z}_N$ of the equation*

$$x + a/x = s. \tag{1}$$

Then, half of the elements of X have Jacobi symbol 1 and half have symbol -1.

Proof. Let $N = p_1 \cdots p_k$ be the factorization of N where p_i are distinct odd primes. By construction, t is a solution of 1 and hence a/t is also. Therefore, any solution in $\mathbb{Z}_N$ of our equation can be computed by solving the following 2^k systems of congruences:

$$\begin{cases} x \equiv x_1 \pmod{p_1} \\ x \equiv x_2 \pmod{p_2} \\ \quad \vdots \\ x \equiv x_k \pmod{p_k} \end{cases} \tag{2}$$

where $x_1, \ldots, x_k$ vary in the set $\{t, a/t\}$. Since a is not a square there exists at least one prime p_i such that $\left(\frac{a}{p_i}\right) = -1$. Up to reordering the prime factors of N we have:

$$\left(\frac{a}{p_1}\right) = \cdots = \left(\frac{a}{p_h}\right) = -1, \text{ and } \left(\frac{a}{p_{h+1}}\right) = \cdots = \left(\frac{a}{p_k}\right) = 1.$$

It follows that a solution x of 1 has Jacobi symbol equals to $\left(\frac{t}{N}\right)$ if and only if there are an odd number of a/t among $x_1, \ldots, x_h$ in 2. The thesis follows by noticing that in any set the number of subsets with odd cardinality is the same as the number of those with even cardinality.

2. Let $N = p_1 \cdots p_k$ be a squarefree integer and $x \in \mathcal{J}_N \setminus \mathbb{Z}_N^*$ be a non square. Let $s \leftarrow \mathsf{Cocks.Enc}(b, (N, x))$ be an encryption of a bit b. Consider the Cocks equation (in t) $t + x/t = s$. By the previous lemma, exactly half of the solutions of that equation have Jacobi symbol 1 (and exactly half have symbol -1). It follows that, with the same probability, s can be the encryption of $b = 0$ or the encryption of $b = 1$.

3.2 Non-interactive oblivious transfer for RSA modulus

A user holds a key pair $\mathsf{sk} = (p, q), \mathsf{pk} = N = pq$ for some distinct primes p and q. To obliviously send two messages m_0 and m_1 to the user, the sender will choose an element $x \in \mathcal{J}_N$ and encrypt m_0 with Goldwasser-Micali and m_1 with Cocks, both using the public key (N, x). If x is a square modulo N then the receiver will be able to decrypt m_1, otherwise it will decrypt m_0.

Following this plain approach, we encounter some difficulties. Consider the following scenarios:

1. A malicious sender can choose x to control the choice bit of the user. For example by always choosing a square. We cope with this by defining x as the output $\mathsf{H}_N(N, \mathsf{cnt})$ of a random oracle H_N.
2. What happens if the user's public key N is not a valid RSA modulus? If x is a square modulo N, no matter the factorization of N, the GM encryption statically hides the message m_0. On the other hand, if x is not a square and N is not square-free, then the user can decrypt both ciphertexts.

Thanks to Proposition 1, if N is square-free and x is a non-square, then Cocks statistically hides m_1. Therefore, we can tackle with 2 by "forcing" N to be square-free. Using a similar technique to [17], instead of sending the "plain" ciphertexts, the sender will encrypt them using a CPA-secure symmetric cipher and provide some hints to reconstruct the encryption key. The user can recover the encryption key only if N is squarefree. The full description is given in Scheme 3.

Let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_N : \{0,1\}^* \to \mathcal{J}_N$ be two hash functions modeled as a random oracle, and let $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure symmetric cipher. We define the scheme $\mathsf{NIOT} = (\mathsf{KeyGen}, \mathsf{S}, \mathsf{R})$ as follows

$\underline{\mathsf{KeyGen}(\lambda)}$**:** Generate two distinct prime numbers $p(\lambda)$ and $q(\lambda)$, set $N = pq$ and return $(\mathsf{pk} = N, \mathsf{sk} = (p, q))$.

$\underline{\mathsf{S}(m_0, m_1, \mathsf{pk} = N, \mathsf{cnt})}$**.** On input two messages $m_0, m_1 \in \{0,1\}^\lambda$, a public key N and a context cnt:

1. compute $x \leftarrow \mathsf{H}_N(N, \mathsf{cnt})$, for $i \in [\lambda]$ sample random $a_i, a'_i \leftarrow\!\!\$\ \mathbb{Z}_N$ and derive a key $k \leftarrow \mathsf{H}(a_1, a'_1, \ldots, a_\lambda, a'_\lambda)$;
2. compute $y_i \leftarrow (a_i)^N \mod N$, $s_i \leftarrow (a'_i)^2 \mod N$ and $h_i = \mathsf{H}(a'_i)$ for $i \in [\lambda]$,
3. compute $c_0 \leftarrow \mathsf{GM.Enc}(m_0, (N, x))$ and $c_1 \leftarrow \mathsf{Cocks.Enc}(m_1, (N, x))$;
4. compute $\mathsf{ct}_0 \leftarrow \mathsf{Enc}_k(c_0)$ and $\mathsf{ct}_1 = \mathsf{Enc}_k(c_1)$;
5. return $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$.

$\underline{\mathsf{R}(\mathsf{ct}, \mathsf{pk} = N, \mathsf{sk} = (p, q), \mathsf{cnt})}$**.** On input a ciphertext ct, a public key $\mathsf{pk} = N$, the corresponding secret key $\mathsf{sk} = (p, q)$ and a context cnt, parse $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$. Recover $x = \mathsf{H}_N(N, \mathsf{cnt})$ and $a'_i = s_i^{1/2} \mod N$, such that $h_i = \mathsf{H}(a'_i)$. Compute $k = \mathsf{H}(y_1^{1/N}, a'_1, \ldots, y_\lambda^{1/N}, a'_\lambda)$. Depending on x, perform the following:

- if $x \in \mathrm{QR}_N$, compute $c_1 = \mathsf{Dec}_k(\mathsf{ct}_1)$ and output $(\mathsf{Cocks.Dec}(c_1, (N, x), (p, q)), 1)$;
- If $x \notin \mathrm{QR}_N$, compute $c_0 = \mathsf{Dec}_k(\mathsf{ct}_0)$ then output $(\mathsf{GM.Dec}(c_0, (N, x), (p, q)), 0)$.

Scheme 3: The description of NIOT

Theorem 1. *The non-interactive oblivious transfer defined in Scheme 3 is correct and extractable.*

Proof. **Correctness.** Let $(\mathsf{pk}, \mathsf{sk}) = (N, (p, q))$ be a key pair generated with KeyGen and let cnt be a context. Let $x = \mathsf{H}_N(N, \mathsf{cnt})$, then the choice bit is $b = 1$ if $x \in \mathrm{QR}_N$ and $b = 0$ otherwise. It is easy to check that for any messages m_0, m_1:

$$\mathsf{R}(\mathsf{S}(m_0, m_1, \mathsf{pk}, \mathsf{cnt}), \mathsf{sk}, \mathsf{cnt}) = (b, m_b).$$

Extractability. To decrypt any of the messages m_0 or m_1, the adversary needs to query the oracle H with the square roots of the values s_i for all $i \in [\lambda]$. There is a $1 - 1/2^{\lambda}$ probability that the adversary queries a square root $a_i'' \neq \pm a_i'$ mod N for some $i \in [\lambda]$. The extractor $\mathsf{Extract}_{\mathsf{NIOT}}$ can be constructed using an adversary $\mathcal{A}$ as a subroutine and waiting for such a query. Once $\mathcal{A}$ makes such a query, it uses the square roots a_i'' and a_i' to factor N. Note that if $\mathcal{A}$ "fails" while querying a_i'' (i.e., the check $\mathsf{H}(a_i'') \neq h_i$), it will compute a different square root until it computes and queries a_i'. Thus, by looking at the random oracle calls of $\mathcal{A}$, the extractor $\mathsf{Extract}_{\mathsf{NIOT}}$ learns both square roots. We do not assume that the extractor can program the random oracle (it only handles $\mathcal{A}$'s calls).

Theorem 2. *The non-interactive oblivious transfer defined in Scheme 3 satisfies receiver security (definition 14).*

Proof. We can turn an efficient adversary against $\mathsf{RcvSec}_{\mathcal{A},\mathsf{NIOT}}$ into an efficient distinguisher for EQR. Let $\mathcal{A}$ be an adversary against $\mathsf{RcvSec}_{\mathcal{A},\mathsf{NIOT}}$. Assume that $\mathcal{A}$ makes at most $q_s = \mathsf{poly}(\lambda)$ queries at the random oracle. We can also assume that for any output $(\mathsf{ct}, \mathsf{cnt}, \hat{b})$ of $\mathcal{A}$ (on input $\mathsf{pk} = N, \lambda$), the adversary asked for $\mathsf{H}_N(N, \mathsf{cnt})$ at some point of its execution.

We construct the following distinguisher D against EQR. On input N, $x \in \mathcal{J}_N$ and $y \in \mathcal{J}_N \setminus \mathrm{QR}_N$ do:

1. Set $Q = \emptyset$ and $\mathsf{pk} = N$. Run $\mathcal{A}$ on input (pk, λ).
2. Respond to the random oracle queries of $\mathcal{A}$ in the following way. Pick a random $I \in [q_s]$ and program the oracle such that the output of the I-th query is x. For the other queries do the following: sample a bit $c \leftarrow \{0, 1\}$ and an element $z \leftarrow \mathbb{Z}_N^*$, if $c = 0$ return z^2 otherwise otput $z^2 y$. In particular, if $c = 0$ then the output is a square and if $c = 1$ is a non-square.
3. Respond to the queries to $\mathcal{O}$ of $\mathcal{A}$ according to the RO queries.
4. Receive $(\mathsf{ct}, \mathsf{cnt}, \hat{b})$ from $\mathcal{A}$. If cnt corresponds to the context of the I-th query output $\hat{b}$, otherwise repeat form step 1.

Let us denote by P the following quantity

$$P = |\Pr[D(N, x) = 1 \mid x \in \mathrm{QR}_N] - \Pr[D(N, x) = 1 \mid x \notin \mathrm{QR}_N]|.$$

By the construction of D, it follows that

$$\mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}} = \mathsf{poly}(\lambda) \cdot P.$$

By extended quadratic residuosity, there exists a negligible function $\epsilon'(\lambda)$ such $P = \epsilon'(\lambda)$. We therefore have

$$\mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}} = \epsilon(\lambda)$$

where $\epsilon(\lambda) = \mathsf{poly}(\lambda)\epsilon'(\lambda)$ is a negligible function.

Theorem 3. *The non-interactive oblivious transfer defined in Scheme 3 satisfies sender security (definition 15).*

Proof. Let $\mathcal{A}$ be an adversary against $\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}}$. Consider the event

$$E = \text{“The modulus returned by } \mathcal{A} \text{ is squarefree”}.$$

By the law of total probability we have:

$$\begin{aligned}\Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1] = \Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E]\Pr[E] + \\ + \Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E^c]\Pr[E^c].\end{aligned}$$

Event E: case squarefree. From $\mathcal{A}$ we construct an adversary $\mathcal{A}^*$ against $\mathsf{IND\text{-}Cocks}$ or $\mathsf{IND\text{-}GM}$.

1. Run $\mathcal{A}$ and get $(m_0, m_1, N, \mathsf{cnt})$. Compute the factorization of N and the value $x = \mathsf{H}_N(N, \mathsf{cnt})$.
2. If $x \in \mathrm{QR}_N$ (resp. $x \notin \mathrm{QR}_N$) abort the experiment $\mathsf{IND\text{-}Cocks}$ (resp. $\mathsf{IND\text{-}GM}$).
3. $\mathcal{A}^*$ gives $\mathsf{pk} = (N, x)$ and m_0, m_1 to the challenger and receives a ciphertext $c^* = \mathsf{Cocks.Enc}(m_b, (N, x))$ (resp. $c^* = \mathsf{GM.Enc}(m_b, (N, x))$) where b is the challenge bit.
4. For each $i \in [\lambda]$ sample elements $a_i, a_i' \leftarrow \mathbb{Z}_N^*$, set $y_i \leftarrow a_i^N \mod N$, $s_i \leftarrow (a_i')^2 \mod N$ and $h_i = \mathsf{H}(a_i')$.
5. Compute the key $k = \mathsf{H}(a_1, a_1', \ldots, a_\lambda, a_\lambda')$. Define ct_0 and c_1 as follows:

$$\mathsf{ct}_0 = \mathsf{Enc}_k(\mathsf{GM.Enc}(m_0, (N, x))) \text{ (resp. } \mathsf{ct}_0 = \mathsf{Enc}_k(c^*)),$$

$$\mathsf{ct}_1 = \mathsf{Enc}_k(c^*) \text{ (resp. } \mathsf{ct}_1 = \mathsf{Enc}_k(\mathsf{Cocks.Enc}(m_1, (N, x)))).$$

   Send to $\mathcal{A}$ the ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$.
6. Receive a bit $\hat{b}$ from $\mathcal{A}$ and outputs $\hat{b}$.

By proposition 1, we have

$$\Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E] = \frac{1}{2}.$$

Event E: case non squarefree. We use $\mathcal{A}$ to construct an adversry against $\mathsf{IND\text{-}CPA}_{\Pi}$. Consider the following adversary $\mathcal{A}^*$:

1. Run $\mathcal{A}$ and receive $(m_0', m_1', N, \mathsf{cnt})$. Define $x = \mathsf{H}_N(N, \mathsf{cnt})$.
2. Sample a bit $c \leftarrow\!\!\$\ \{0, 1\}$
3. Define the messages

$$m_0 = \begin{cases} \mathsf{GM.Enc}(m_0', (N, x)) & \text{if } c = 0 \\ \mathsf{Cocks.Enc}(m_0', (N, x)) & \text{if } c = 1 \end{cases}$$

   and

$$m_1 = \begin{cases} \mathsf{GM.Enc}(m_1', (N, x)) & \text{if } c = 0 \\ \mathsf{Cocks.Enc}(m_1', (N, x)) & \text{if } c = 1 \end{cases}$$

Also, define

$$m = \begin{cases} \mathsf{Cocks.Enc}(m'_1, (N, x)) & \text{if } c = 0 \\ \mathsf{GM.Enc}(m'_0, (N, x)) & \text{if } c = 1 \end{cases}$$

Ask the challenger for $d = \mathsf{Enc}(k, m)$. Send m_0 and m_1 to the challenger and receive $c^* = \mathsf{Enc}(k, m_b)$ where b is the challenge bit and k is the challenger's key.

4. For each $i \in [\lambda]$ sample elements $a_i, a'_i \leftarrow \mathbb{Z}_N^*$ and set $y_i \leftarrow a_i^N \mod N$, $s_i \leftarrow (a'_i)^2 \mod N$ and $h_i = \mathsf{H}(a'_i)$.
5. If $c = 0$ send $(c^*, d, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$ to $\mathcal{A}$, otherwise send $(d, c^*, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$.
6. Receive a bit $\hat{b}$ from $\mathcal{A}$ and output $\hat{b}$ if $c = 0$ and $1 - \hat{b}$ otherwise.

Let us analyze the advantage. Consider the following event

$A =$"There exists elements a_i s.t. $k = \mathsf{H}(a_1, a'_1, \ldots, a_\lambda, a'_\lambda)$ and $a_i^N = y_i$".

Since N is not squarefree we have that $D = \gcd(\varphi(N), N) > 1$. Then, for all i, the number of solutions to the equation $x^N = y_i$ is greater or equal to D. Therefore, $\Pr[A^c]$ is negligible. Hence, we have the following:

$$\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1] = \Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1 \mid A] + \epsilon'(\lambda)$$

for a negligible function $\epsilon'(\lambda)$. Furthermore, we have

$$\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1 \mid A] = \frac{1}{2}\left(\Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1] + \frac{1}{2}\right)$$

and hence

$$\Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E^c] = 2\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1] - \frac{1}{2} - \epsilon'(\lambda)$$

In conclusion:

$$\begin{aligned} \Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1] &= \Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E]\Pr[E] + \\ &+ \Pr[\mathsf{SndSec}_{\mathcal{A},\mathsf{NIOT}} = 1 \mid E^c]\Pr[E^c] \\ &= \frac{1}{2}\Pr[E] + \\ &+ \left(2\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1] - \frac{1}{2} - \epsilon'(\lambda)\right)\Pr[E^c] \\ &= \frac{1}{2} + 2\Pr[E^c]\left(\Pr[\mathsf{IND\text{-}CPA}_{\mathcal{A}^*,\Pi}(\lambda) = 1] - \frac{1}{2}\right) + \Pr[E^c]\epsilon'(\lambda). \end{aligned}$$

Since Π is a secure CPA symmetric cipher, then the advantage of $\mathcal{A}$ is negligible.

Remark 4 (Optimized version). In case the extractability of Scheme 3 is not needed, then it can be optimized by removing all the elements s_i and h_i from ct. Moreover, if the sender checks whether the smallest primes up to prime 257 divide the modulus N before computing ct, then the sender can only sample $\lambda/8$ elements a_i instead of λ.

Remark 5 (Batching). Scheme 3 is a 1 out of 2 but allows for efficiency improvements if the sender wants to batch execute t such protocols. To batch, it can reuse the values $y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda$ from one execution for the other ones. In particular, the ciphertext for the batch schemes can be set to $\mathsf{ct} = (\mathsf{ct}_0^{(1)}, \mathsf{ct}_1^{(1)}, \ldots, \mathsf{ct}_0^{(t)}, \mathsf{ct}_1^{(t)}, y_1, s_1, h_1, \ldots, y_\lambda, s_\lambda, h_\lambda)$, which saves $2 \cdot \lambda \cdot (t-1)$ elements from $\mathbb{Z}_N$ and $\lambda \cdot (t-1)$ elements from $\{0,1\}^\lambda$.

4 Garbled Circuits for Signing Algorithms

We now show how to garble specific scheme's signing procedures efficiently. We present our techniques on two example schemes: the Pointcheval-Sanders (PS) [27] and the RSA-based signature scheme with efficient protocols [9].

4.1 Garbling Pointcheval-Sanders Signatures.

We start this section with a way to garble the signing algorithm for PS signatures [27], where the evaluator's input is the message to sign, and the garbled function includes the signing key. In particular, let $X = g_2^x$ and $Y = g_2^y$ be the public key elements. A Pointcheval-Sanders signature under message m with bit encoding $m_1, \ldots, m_\ell$ is of the form (σ_1, σ_2), where $\sigma_1 = h \in \mathbb{G}_1$ and $\sigma_2 = h^{x+y \cdot m}$ To garble the signing process, the garbler first randomly picks encryption keys $k_1^0, k_1^1, \ldots, k_\ell^0, k_\ell^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and generates a secret sharing of $1_{\mathbb{G}_1}$, i.e., this can be realized by first picking $a_1, \ldots, a_\ell \leftarrow\!\!\$\ \mathbb{G}_1$ and then computing $a_0 = (\prod_{i=1}^{\ell} a_i)^{-1}$. In the next step, the garbler chooses a random $h \leftarrow\!\!\$\ \mathbb{G}_1$ and sets $\sigma_1 = h$ directly. The second element of the signature σ_2 is the part we garble. For each $i \in [\ell]$ the garbler computes "shares" $s_i^0 = a_i$ (for the case that bit $m_i = 0$) and $s_i^1 = a_i \cdot (h^y)^{2^{i-1}}$ (if the bit is $m_i = 1$). It then computes ciphertexts $\mathsf{ct}_i^t \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^0), s_i^0)$ and $\mathsf{ct}_i^{1-t} \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^1), s_i^1)$, where H_{AE} is a random oracle that outputs keys from the keyspace of the authenticated encryption scheme AE. We use the random bit t to permute and hide the ciphertexts. To allow the evaluator to properly decode the share having only one of the keys k_i^0, k_i^1, we use the properties of authenticated encryption, i.e., in case the evaluator picks the wrong ciphertext the decryption will output $\perp$. The garbler then constructs the garbled circuit as $F = (\{\mathsf{ct}_i^0, \mathsf{ct}_i^1\}_{i \in [\ell]}, \sigma_1)$. The encoding key e contains all the keys $\{k_i^0, k_i^1\}_{i \in \ell}$, which can be used as the encoding vector in the En algorithm. Thanks to this vector, the scheme is projective, and the evaluator can choose the keys corresponding to its message using an oblivious transfer protocol. Using the keys k_i and the ciphertext ct_i for $i \in [\ell]$, the evaluator can reconstruct the shares s_i. It then computes $s = \prod_{i=1}^{\ell} s_i$ and the evaluator can output the garbled output $Y = (\sigma_1, s)$. Finally, using the decoding information $d = (a_0 \cdot h^x)$, the evaluator finalizes the process by computing $\sigma_1 = s \cdot d$ and outputting (σ_1, σ_2). More details are provided in Scheme 4.

Remark 6. The scheme can be easily adapted to compute a message under $\alpha \cdot m + \beta$. To do so, we replace $s_i^1 = a_i \cdot (h^y)^{2^{i-1}}$ with $s_i^1 = a_i \cdot (h^{y \cdot \alpha})^{2^{i-1}}$ and $d = (a_0 \cdot h^x)$

Let $\mathsf{pk} = (X = g_2^x, Y = g_2^y)$ be the public key for the Pointcheval-Sanders scheme for message space $q \subseteq \{0,1\}^\ell$ (i.e., elements from $\mathbb{Z}_q$ can be encoded in ℓ bits). Let $\mathsf{sk} = (x, y)$ be the corresponding secret key. Finally, let AE define an authenticated encryption scheme, where H_{AE} is a random oracle to the keyspace of AE. We define the function we garble $f_{\mathsf{sk}}(m) = \mathsf{Sign}_{\mathsf{PS}}(\mathsf{sk}, m)$.

$\underline{\mathsf{Gb}(\lambda, f)}$**:** pick a random $h \leftarrow\!\!\$\ \mathbb{G}_1$, pick random keys $k_1^0, k_1^1, \ldots, k_\ell^0, k_\ell^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$. Generate a multiplicative secret sharing of $1_{\mathbb{G}_1}$ by first picking $a_1, \ldots, a_\ell \leftarrow\!\!\$\ \mathbb{G}_1$ and then computing $a_0 = 1_{\mathbb{G}_1} \cdot (\prod_{i=1}^{\ell} a_i)^{-1}$.
For each $i \in [\ell]$:
- set $s_i^0 = a_i$, $s_i^1 = a_i \cdot (h^y)^{2^{i-1}}$ and sample bit t,
- compute ciphertexts

$$\mathsf{ct}_i^t \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^0), s_i^0) \qquad \mathsf{ct}_i^{1-t} \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^1), s_i^1)$$

Set and output $F = (\{\mathsf{ct}_i^0, \mathsf{ct}_i^1\}_{i\in\ell}, h)$, $d = (a_0 \cdot h^x)$, and $e = (\{k_i^0, k_i^1\}_{i\in[\ell]})$.
$\underline{\mathsf{En}(e, x)}$**:** Let $x_1, \ldots, x_\ell$ be the bit encoding of x. Return $X = (k_i^{x_i})_{i\in\ell}$. e can also be used as the encoding vector, and the scheme can be used as a projective scheme.
$\underline{\mathsf{Ev}(F, X)}$**:** parse X as $k_1, \ldots, k_\ell$ and F as $(\{\mathsf{ct}_i^0, \mathsf{ct}_i^1\}_{i\in\ell}, \sigma_1)$. For each $i \in [\ell]$ compute the value $s_{i,t} \leftarrow \mathsf{AE.Dec}(\mathsf{H}_{\mathsf{AE}}(k_i), \mathsf{ct}_i^t)$ for $t \in \{0,1\}$ and set $s_i = s_{i,0}$ if $s_{i,0} \neq \perp$ or $s_i = s_{i,1}$, otherwise. Abort if $s_{i,0} = \perp = s_{i,1}$. Output $Y = (\sigma_1, s = \prod_{i=1}^{\ell} s_i)$.
$\underline{\mathsf{De}(d, Y)}$**:** parse Y as (σ_1, s). Compute $\sigma_2 = s \cdot d$. Output $y = (\sigma_1, \sigma_2)$.

Scheme 4: Garbled Pointcheval-Sanders Signatures

with $d = (a_0 \cdot h^x \cdot (h^y)^\beta)$. We also remark that the scheme can be extended to two messages m_1, m_2 and with a final message under $m = (\alpha_1 \cdot m_1 + \beta_1) + (\alpha_2 \cdot m_2 + \beta_2)$, we will later use this in our fully secure NIBS construction.

Remark 7 (Rand *for PS Signatures).* PS signatures can be randomized using $\mathsf{Rand}((\sigma_1, \sigma_2))$ that outputs (σ_1^r, σ_2^r) for some $r \leftarrow\!\!\$\ \mathbb{Z}_q$. It is easy to see that randomized signatures are indistinguishable from freshly signed signatures. We have $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}} = 0$.

Theorem 4. *Scheme 4 is simulation private in the random oracle model.*

Proof. We show this proof by constructing a simulator algorithm S that takes as input a valid signature $y = (\sigma_1, \sigma_2)$ and outputs a garbled circuit F, encoding X and decoding d. The simulator first picks random keys $k_1, \ldots, k_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda$ and $k'_1, \ldots, k'_\ell \leftarrow\!\!\$\ \{0,1\}^\lambda$. S picks $a_0, a_1, \ldots, a_\lambda$ such that $\prod_{i=0}^{\lambda} a_i = 1_{\mathbb{G}_1}$. Then for each $i \in [\lambda]$ it: 1) sets $s_i = a_i$ and picks random bit $t \leftarrow \{0,1\}$, 2) computes ciphertext $c_i^t \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i), s_i)$, 3) computes ciphertext $c_i^{1-t} \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k'_i), 0)$. The simulator sets $F = (\{\mathsf{ct}_i^0, \mathsf{ct}_i^1, \}_{i\in[\ell]}, \sigma_1)$, $X = (\{k_i\}_{i\in[\ell]})$ and $d = \sigma_2 \cdot a_0$. (F, X, d) is a valid output generated by S for y.

4.2 Garbling RSA Signatures.

We showed how to garble PS signatures efficiently. We will explore a similar idea for the RSA-based signature scheme with efficient protocols [9]. The scheme works over an RSA modulus $N = pq$ for safe primes p and q and length ℓ_N. The signer's public key consists of the modulus n and quadratic residues $a, b, c \in \mathrm{QR}_N$, while the secret key is one of the factors p. To sign a ℓ_m bit long message m, the signer picks prime $e > 2^{\ell_m+1}$, random number s of length $\ell_N + \ell_m + \lambda$ and computes v such that $v^e = a^m \cdot b^s \cdot c \mod N$. The signature is $\sigma = (e, s, v)$. The interesting part of this signature scheme is that an efficient proof of signature knowledge exists for σ. Therefore, this section will show how to garble the signing function when e and s are specified before signing. To do so, we notice that given e and s, we can write the computation of $v = \alpha \cdot \beta \mod N$, where $\alpha = (b^s \cdot c)^{1/e}$ and $\beta = (a^{1/e})^m$. It is easy to see that only β depends on the input to the garbled circuit (i.e., the message). Since β is similar to the $(h^y)^m$ component in the PS signatures, we can apply the same techniques to achieve the garbling. We provide the full description in Scheme 5.

Let $\mathsf{pk} = (N = pq, a, b, c)$ be the public key for the signature scheme [9] for message space $\{0,1\}^{\ell_m}$, let ℓ_N corresponds to the bit size of N and $\mathsf{sk} = p$ be the corresponding secret key. Finally, let AE define an authenticated encryption scheme, where H_{AE} is a random oracle to the keyspace of AE.

$\underline{\mathsf{Gb}(\lambda, f)}$**:** Choose random elements $a_1, \ldots, a_{\ell_m} \leftarrow\!\!\$\ \mathrm{QR}_N$, keys keys $k_1^0, k_1^1, \ldots, k_{\ell_m}^0, k_{\ell_m}^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$, prime number $e_\sigma > 2^{\ell_m+1}$ and s_σ of bit length $\ell_N + \ell_m + \lambda$. Compute $a_0 = 4/\prod_{i=1}^{\ell_m} a_i \mod N$.
For each $i \in [\ell_m]$:
- set $s_i^0 = a_i$, $s_i^1 = a_i \cdot (a^{1/e_\sigma})^{2^{i-1}}$ and sample bit t,
- compute ciphertexts

$$\mathsf{ct}_i^t \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^0), s_i^0) \qquad \mathsf{ct}_i^{1-t} \leftarrow \mathsf{AE.Enc}(\mathsf{H}_{\mathsf{AE}}(k_i^1), s_i^1)$$

Set and output $F = (\{\mathsf{ct}_i^0, \mathsf{ct}_i^1\}_{i\in\ell_m}, e_\sigma, s_\sigma)$, $d = (a_0 \cdot (b^{s_\sigma} \cdot c)^{1/e_\sigma})$, and $e = (\{k_i^0, k_i^1\}_{i\in[\ell_m]})$.

$\underline{\mathsf{En}(e_G, x)}$**:** Let $x_1, \ldots, x_{\ell_m}$ be the bit encoding of x. Return $X = (k_i^{x_i})_{i\in\ell_m}$. e can also be used as the encoding vector, and the scheme can be used as a projective scheme.

$\underline{\mathsf{Ev}(F_G, X)}$**:** parse X as $k_1, \ldots, k_{\ell_m}$ and F as $(\{\mathsf{ct}_i^0, \mathsf{ct}_i^1\}_{i\in\ell}, e_\sigma, s_\sigma)$. For each $i \in [\ell_m]$ compute the value $s_{i,t} \leftarrow \mathsf{AE.Dec}(\mathsf{H}_{\mathsf{AE}}(k_i), \mathsf{ct}_i^t)$ for $t \in \{0,1\}$ and set $s_i = s_{i,0}$ if $s_{i,0} \neq \perp$ or $s_i = s_{i,1}$, otherwise. Abort if $s_{i,0} = \perp = s_{i,1}$. Compute the output $s = (\prod_{i=1}^{\ell_m} s_i)/4 \mod N$. Output $Y = (e_\sigma, s_\sigma, s)$.

$\underline{\mathsf{De}(d_G, Y)}$**:** Compute $v_\sigma = s \cdot d \mod N$. Set $\sigma = (e_\sigma, s_\sigma, v_\sigma)$ and $y = \sigma$.

Scheme 5: Garbled RSA-based Signatures with Efficient Protocol [9]

$\mathsf{HRBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda)$

$(\mathsf{sk}_{R_0}, \mathsf{pk}_{R_0}) \leftarrow \mathsf{RKeyGen}(\lambda)$
$(\mathsf{sk}_{R_1}, \mathsf{pk}_{R_1}) \leftarrow \mathsf{RKeyGen}(\lambda)$
$\mathcal{O} := \mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(\cdot,\cdot,\cdot)$
$((r_i, \mathsf{nonce}_i)_{i=0,1}, (\mathsf{sk}, \mathsf{pk})) \leftarrow \mathcal{A}^{\mathcal{O}}((\mathsf{pk}_{R_i})_{i=0,1})$
$\mathsf{psig}_0 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{R_0}, \mathsf{nonce}_0; r_0)$
$\mathsf{psig}_1 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{R_1}, \mathsf{nonce}_1; r_1)$
$(m_0, \mathsf{sig}_0) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_0}, \mathsf{pk}, \mathsf{psig}_0, \mathsf{nonce}_0)$
$(m_1, \mathsf{sig}_1) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_1}, \mathsf{pk}, \mathsf{psig}_1, \mathsf{nonce}_1)$
if $\mathsf{sig}_0 = \bot$ or $\mathsf{sig}_1 = \bot$ **then**
  $(m_0, \mathsf{sig}_0) := \bot; (m_1, \mathsf{sig}_1) := \bot$
$b \leftarrow\!\!\$\ \{0,1\}$
$\hat{b} \leftarrow \mathcal{A}((m_b, \mathsf{sig}_b), (m_{1-b}, \mathsf{sig}_{1-b}))$
return $b = \hat{b}$

$\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(b, \mathsf{pk}, (\mathsf{psig}, \mathsf{nonce}))$

$(m, \mathsf{sig}) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_b}, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})$
return (m, sig)

$\mathsf{HNBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda)$

$(\mathsf{sk}_R, \mathsf{pk}_R) \leftarrow \mathsf{RKeyGen}(\lambda)$
$\mathcal{O} := \mathcal{O}_{\mathsf{sk}_R}(\cdot,\cdot)$
$(r_0, \mathsf{nonce}_0, r_1, \mathsf{nonce}_1, (\mathsf{sk}, \mathsf{pk})) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}_R)$
$\mathsf{psig}_0 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce}_0; r_0)$
$\mathsf{psig}_1 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce}_1; r_1)$
$(m_0, \mathsf{sig}_0) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}_0, \mathsf{nonce}_0)$
$(m_1, \mathsf{sig}_1) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}_1, \mathsf{nonce}_1)$
if $\mathsf{sig}_0 = \bot$ or $\mathsf{sig}_1 = \bot$ **then**
  $(m_0, \mathsf{sig}_0) := \bot; (m_1, \mathsf{sig}_1) := \bot$
$b \leftarrow\!\!\$\ \{0,1\}$
$\hat{b} \leftarrow \mathcal{A}((m_b, \mathsf{sig}_b), (m_{1-b}, \mathsf{sig}_{1-b}))$
return $b = \hat{b} \ \wedge\ \mathsf{nonce}_0 \neq \mathsf{nonce}_1$

$\mathcal{O}_{\mathsf{sk}_R}(\mathsf{pk}, \mathsf{psig}, \mathsf{nonce})$

$(m, \mathsf{sig}) \leftarrow \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})$
return (m, sig)

Fig. 3. HRBnd and HNBnd Experiments.

*Remark 8 (*Rand *for CL Signatures).* The Camenisch Lysyanskaya signatures [9] allow for an efficient proof of knowledge of a signature. We will use this proof in the Rand algorithm, where the randomized signature $\mathsf{sig}' \leftarrow \mathsf{Rand}(\mathsf{sig})$ is the proof of knowledge of (e, s, A).

Theorem 5. *Scheme 5 is simulation private in the random oracle model.*

Proof. The proof follows the same idea as the proof of Theorem 4, hence we omit it.

5 NIBS from NI Oblivious Transfer and Garbled Circuits

In this section, we discuss one of our contributions. We show how to construct non-interactive blind signatures securely from non-interactive oblivious transfer and garbled circuits. As in the case of multi-party computation based on Yao's garbling circuits, we only achieve security against an honest-but-curious adversary. Therefore, we modify the original notions of recipient and nonce blindness [21,2] and propose their honest variants. We prove the security of our construction under these new definitions. A folklore solution to achieve malicious security

is to employ a zero-knowledge proof of knowledge proving the consistency of computation and knowledge of the secret key. This basic approach can also be applied in our construction. For completeness, we provide a more formal description of this well-known approach with proofs in the extended version.

5.1 Honest Blind Non-Interactive Blind Signatures.

In our new notions, we assume that the pre-signatures are generated honestly by the experiment but using the secret key and random coins provided by the adversary. In the original notions, the adversary is not required to output the private key, and instead of giving the random coins to generate the pre-signatures, it provides them directly. In the new notions, the adversary can still pick the pre-signatures but must be able to "explain" how they were generated. In particular, we assume it outputs the random coins used in the Issue algorithm to create the pre-signature of its choosing. More formally.

Definition 17 (Honest Recipient Blindness). *A* NIBS *scheme is recipeint blind against an honest-but-curious adversary, if for all PPT adversaries* $\mathcal{A}$*, their advantage defined as* $\mathbf{Adv}^{\mathsf{HRBnd}}_{\mathcal{A},\mathsf{NIBS}} = |\Pr[\mathsf{HRBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda) = 1] - 1/2|$ *is negligible, where the experiment* $\mathsf{HRBnd}_{\mathcal{A},\mathsf{NIBS}}$ *is defined in Fig. 3.*

Definition 18 (Honest Nonce Blindness). *A* NIBS *scheme is nonce blind against an honest-but-curious adversary, if for all PPT adversaries* $\mathcal{A}$*, their advantage defined as* $\mathbf{Adv}^{\mathsf{HNBnd}}_{\mathcal{A},\mathsf{NIBS}} = |\Pr[\mathsf{HNBnd}_{\mathcal{A},\mathsf{NIBS}}(\lambda) = 1] - 1/2|$ *is negligible, where the experiment* $\mathsf{HNBnd}_{\mathcal{A},\mathsf{NIBS}}$ *is defined in Fig. 3.*

5.2 Generic NIBS Scheme.

We will now define the generic construction of non-interactive blind signatures that we construct using the techniques of Yao's garbled circuits. The idea is for the pre-signature psig to include a garbled circuit F of the function $f_{\mathsf{sk},\mathsf{nonce}}(\mathsf{sk}_R) = \mathsf{SIG.Sign}(\mathsf{sk}, \bar{\mathsf{sk}}_{R,\mathsf{nonce}}))$. In other words, the signer garbles the signing algorithm of a signatures scheme SIG, for message $\bar{\mathsf{sk}}_{R,\mathsf{nonce}}$. This key corresponds to the user's multiple and independent choices in the oblivious transfer protocol. The signer gives the user the correct input labels for the string $\bar{\mathsf{sk}}_{R,\mathsf{nonce}}$. However, we rely on a non-interactive version instead of an interactive oblivious transfer protocol, so we use a counter and the signer's nonce nonce to generate different choice bits that correspond to the bit string $\bar{\mathsf{sk}}_{R,\mathsf{nonce}} = \bar{\mathsf{sk}}_{1,\mathsf{nonce}} \ldots \bar{\mathsf{sk}}_{\ell_m,\mathsf{nonce}}$. In other words, assume that the input to the garbled circuit is ℓ_m bits, give a NIOT public key pk, the sender will use the context $\mathsf{cnt} = i, \mathsf{nonce}$ for $i \in [\ell_m]$ to send labels X_i^0, X_i^1 for the garbled circuit. Note that in the end, the user will receive the label $X_i^{\bar{\mathsf{sk}}_{i,\mathsf{nonce}}}$ for this input position. The user can obtain the final signature sig by running the NIOT protocol for each receiving the encoded input X, and then by evaluating the circuit F, she receives the signature sig. Before outputting, the user verifies the message-signature pair (m, sig). Note, that before outputting the signature sig, the user randomizes it using the Rand algorithm. More details are given in Scheme 6.

Let $\mathsf{NIOT} = (\mathsf{KeyGen}, \mathsf{S}, \mathsf{R})$ be a non-interactive oblivious transfer scheme and let $G = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ be a projective garbled circuit scheme. Moreover, let $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Rand})$ be a signature scheme with message space $\mathcal{M} \subseteq \{0,1\}^{\ell_m}$. Finally, define the function: $f_{\mathsf{sk},\mathsf{nonce}}(\mathsf{sk}_R) = \mathsf{SIG.Sign}(\mathsf{sk}, \bar{\mathsf{sk}}_{R,\mathsf{nonce}})$.

$\mathsf{KeyGen}(\lambda)$**:** return $\mathsf{SIG.KeyGen}(\lambda)$.

$\mathsf{RKeyGen}(\lambda)$**:** generate keys $(\mathsf{sk}_R, \mathsf{pk}_R) \leftarrow \mathsf{NIOT.KeyGen}(\lambda)$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})$**:** garble the function $f_{\mathsf{sk},\mathsf{nonce}}$ by running $(F, e, d) \leftarrow G.\mathsf{Gb}(\lambda, f_{\mathsf{sk},\mathsf{nonce}})$. Compute the projective encoding vector $(X_1^0, X_1^1, \ldots, X_{\ell_m}^0, X_{\ell_m}^1)$ using $G.\mathsf{En}(e, \cdot)$. Run the NIOT scheme on $(X_1^0, X_1^1, \ldots, X_{\ell_m}^0, X_{\ell_m}^1)$ i.e., for each $i \in [\ell_m]$ compute $\mathsf{cnt}_i = (i, \mathsf{nonce})$ and execute $\mathsf{ot}_i \leftarrow \mathsf{NIOT.S}((X_i^0, X_i^1), \mathsf{pk}_R, \mathsf{cnt}_i)$. Output pre-signature

$$\mathsf{psig} = \left(F, d, \{\mathsf{ot}_i\}_{i \in [\ell_m]}\right).$$

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})$**:** parse psig as $\left(F, d, \{\mathsf{ot}_i\}_{i \in [\ell_m]}\right)$. For each $i \in [\ell_m]$ compute $\mathsf{cnt}_i = (i, \mathsf{nonce})$ and execute $(m_i, X_i) \leftarrow \mathsf{NIOT.R}(\mathsf{ot}_i, \mathsf{sk}_R, \mathsf{cnt}_i)$ for each $i \in [\ell_m]$ and combine them into vector X. Evaluate the garbled circuit $Y \leftarrow G.\mathsf{Ev}(F, X)$ and compute the output $\mathsf{sig} \leftarrow G.\mathsf{De}(d, Y)$. Compute $m = m_1 \ldots m_{\ell_m}$ and output $\perp$ if $\mathsf{SIG.Verify}(\mathsf{pk}, m, \mathsf{sig}) = 0$. Otherwise, randomize signature $\mathsf{sig}' \leftarrow \mathsf{Rand}(\mathsf{sig})$. Output (m, sig').

$\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig}))$**:** output $\mathsf{SIG.Verify}(\mathsf{pk}, m, \mathsf{sig})$.

Scheme 6: NIBS Scheme from NI Oblivious Transfer and Garbled Circuits

Theorem 6. *Scheme 6 is reusable (Definition 9) assuming the the non-interactive oblivious transfer scheme is receiver secure and correct.*

Proof. Receiver privacy ensures that the bit choices for different contexts cnt are computationally indistinguishable. Otherwise, the oracle in the definition would allow the adversary to break the property easily. It follows that the probability that ℓ_m bits match is negligible since we assume that $\ell_m(\lambda)$. In other words, by using different nonces nonce_0 and nonce_1, the labels the recipient will receive from the NIOT scheme will, with high probability, differ on at least one bit. Thus, $m_0 = m_1$ will only happen with negligible probability, proving the theorem.

Theorem 7. *Scheme 6 is one-more unforgeable (Definition 10) assuming the garbled circuits scheme G is simulation private, the non-interactive oblivious transfer scheme is extractable and sender secure.*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the one-more unforgeability of NIBS. Assume that $\mathcal{A}$ makes at most $q_s = \mathsf{poly}(\lambda)$ signing queries.

Game G_0: The original one-more unforgeability experiment $\mathrm{G}_0 := \mathsf{OM\text{-}UNF}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G_1}$: For each query $(\mathsf{pk}_R, \mathsf{nonce})$ made by $\mathcal{A}$ to the signing oracle, where $(\mathsf{pk}_R, \cdot) \notin \mathcal{KEY}$ we proceed as follows. We run extractor $\mathsf{Extract}_{\mathsf{NIOT}}$ to extract sk_R corresponding to pk_R and keep $(\mathsf{pk}_R, \mathsf{sk}_R)$ in a list $\mathcal{KEY}$. The experiment aborts in case extraction fails.

Game $\mathbf{G_2}$: For each query $(\mathsf{pk}_R, \mathsf{nonce})$ made by $\mathcal{A}$ to the signing oracle, we proceed as follows. We first find the choice $\bar{\mathsf{sk}}_{R,\mathsf{nonce}}$ using the secret key sk_R, where $(\mathsf{pk}_R, \mathsf{sk}_R) \in \mathcal{KEY}$. Now instead of using (X_i^0, X_i^1) for ot_i, we sample a random X_i^b where $b = 1 - \bar{\mathsf{sk}}_{R,i}$ and $\bar{\mathsf{sk}}_{R,\mathsf{nonce}} = \bar{\mathsf{sk}}_{R,1} \ldots \bar{\mathsf{sk}}_{R,\ell_m}$

Game $\mathbf{G_3}$: For each query $(\mathsf{pk}_R, \mathsf{nonce})$ made by $\mathcal{A}$ to the signing oracle, we proceed as follows. We set message $m = \bar{\mathsf{sk}}_{R,\mathsf{nonce}}$, where $(\mathsf{pk}_R, \mathsf{sk}_R) \in \mathcal{KEY}$. We compute the signature $\sigma \leftarrow \mathsf{SIG.Sign}(\mathsf{sk}, m)$ and use the privacy simulator $\mathcal{S}$ for the garbling circuit scheme instead of computing the garbling directly. Note that the simulator outputs an encoding X and not an encoding vector (for the projective scheme), but thanks to the change in G_2, we can encode X in the NIOT without requiring the full encoding vector.

We will show the indistinguishability of the hybrids via a sequence of claims.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_0 and G_1 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_1(\mathcal{A}) = 1] - \Pr[\mathrm{G}_0(\mathcal{A}) = 1]| \leq q_s \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathsf{Extract},\mathsf{NIOT}}.$$

At most q_s, signing queries are made by $\mathcal{A}$, where each can be made with a distinct key pk_R. The probability of the extractor algorithm failing is $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Extract},\mathsf{NIOT}}$, so the claim follows.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_1 and G_2 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_2(\mathcal{A}) = 1] - \Pr[\mathrm{G}_1(\mathcal{A}) = 1]| \leq q_s \cdot \ell_m \cdot \mathbf{Adv}^{\mathsf{SndSec}}_{\mathcal{A},\mathsf{NIOT}}$$

The claim follows via hybrids by applying the sender security notion to each of the adversary's signing queries and each of ℓ_m executions of the NIOT protocol.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_2 and G_3 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_3(\mathcal{A}) = 1] - \Pr[\mathrm{G}_2(\mathcal{A}) = 1]| \leq q_s \cdot \mathbf{Adv}^{\mathcal{A}}_{\mathsf{prv},\mathcal{G}}$$

The claim follows directly by applying the privacy notion of the garbling circuit to each of the adversary's signing queries.

Claim. We now show that an adversary $\mathcal{A}$ against the one-more unforgeability of NIBS in G_3 can be used to break the existential unforgeability of the signature scheme SIG. In other words, we claim that the adversary's $\mathcal{A}$ advantage in hybrid G_3 is:

$$\Pr[\mathrm{G}_3(\mathcal{A})] = \mathbf{Adv}_{\mathsf{SIG}}(\mathcal{A}).$$

To prove this claim, we construct a reduction algorithm $\mathcal{R}$ that uses $\mathcal{A}$ and breaks the existential unforgeability of SIG. Given the public key pk from the challenger, the reduction provides the public key to $\mathcal{A}$. For each query of the adversary to the signing oracle, we proceed as in G_3, except instead of signing the message m, the reduction uses its signing oracle to receive σ. Finally, the adversary output ℓ signatures while the reduction only made $k < \ell$ queries to its signing oracle. Therefore, amongst one of the message-signature pair outputs by the adversary, there must be one message that was not queried by $\mathcal{R}$ to its signing oracle.

Theorem 8. *Scheme 6 is recipient blind against a honest-but-curious adversary (Definition 17) assuming the non-interactive oblivious transfer scheme is receiver secure and correct.*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the honest recipient blindness of NIBS.

Game $\mathbf{G}_0$: The original honest recipient blindness experiment $\mathrm{G}_0 := \mathbf{Adv}^{\mathsf{HRBnd}}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G}_1$: Intead of running the Issue and Obtain algorithms to compute (m_0, sig_0) and (m_1, sig_1), we compute them directly. First, we compute $m_0 = \bar{\mathsf{sk}}_{R_0,\mathsf{nonce}_0}$, $m_1 = \bar{\mathsf{sk}}_{R_1,\mathsf{nonce}_1}$ and then use the secret key sk given by adversary to compute $\mathsf{sig}_0 = \mathsf{SIG.Sign}(\mathsf{sk}, m_0)$ and $\mathsf{sig}_1 = \mathsf{SIG.Sign}(\mathsf{sk}, m_1)$. In case, any of the pre-signatures cannot be finalized to a valid signature, we set $\mathsf{sig}_0 = \perp$ and $\mathsf{sig}_1 = \perp$.

Game $\mathbf{G}_{2,i}$: In each sub-hybrid $i \in [\ell_m]$, we replace bit $\bar{\mathsf{sk}}_{R_0,i}$ with a uniformly random one and compute m_0 using the changed key. Note the in hybrid G_{2,ℓ_m} the key $\bar{\mathsf{sk}}_{R_0,\mathsf{nonce}_0}$ is random and independent of the public key pk_{R_0}.

Game $\mathbf{G}_{3,i}$: Similar as above but now we target bits of the key $\bar{\mathsf{sk}}_{R_1,\mathsf{nonce}}$.

We will show the indistinguishability of the hybrids via a sequence of claims.

Claim. We claim that hybrids G_0 and G_1 are computationally indistinguishable, i.e.,

$$|\Pr[\mathrm{G}_1(\mathcal{A}) = 1] - \Pr[\mathrm{G}_0(\mathcal{A}) = 1]| \leq 2 \cdot \mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}$$

Since the adversary is honest-but-curious, it follows that the only way for the adversary to distinguish this change is by using sig_0 and sig_1. However, the claim follows since the final signatures are randomized using Rand, and we assume that the adversary only has an advantage $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}$ in distinguishing randomized and "fresh" signatures.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_1 and G_{2,ℓ_m} only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{2,\ell_m}(\mathcal{A}) = 1] - \Pr[\mathrm{G}_1(\mathcal{A}) = 1]| \leq \ell_m \cdot \mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}}.$$

We show this claim by constructing a reduction $\mathcal{R}$ that is using an adversary $\mathcal{A}$ distinguishing between hybrids $\mathrm{G}_{2,i}$ and $\mathrm{G}_{2,i+1}$ can be used to break the receiver

security of the NIOT scheme. Assume that $\mathcal{A}$ is playing the recipient blindness experiment but additionally outputs bit d such that $d = 0$ if it is in $\mathrm{G}_{2,i}$ and $d = 1$ if it is in $\mathrm{G}_{2,i+1}$. The reduction plays the recipient security adversary. It receives the public key $\mathsf{pk}_{\mathsf{NIOT}}$ from the challenger and sets it as the public key pk_{R_0}. Since the reduction has access to the oracle $\mathcal{O}_{\mathsf{sk}}(\mathsf{ct}, \mathsf{cnt})$ it can easily simulate all oracles queries of $\mathcal{A}$ to the $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(b, \mathsf{pk}, (\mathsf{psig}, \mathsf{nonce}))$ for calls with $b = 0$. Since all calls to $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ use distinct nonce values, it follows that the calls the reduction makes to $\mathcal{O}_{\mathsf{sk}}(\mathsf{ct}, \mathsf{cnt})$ will contain different context cnt than the challenged one, which makes it possible for us to use it. In other words, in some sense, the reduction has access to the secret key $\mathsf{sk}_{\mathsf{NIOT}}$ for all nonces except the challenged one, which we know by the limitations on $\mathcal{A}$'s access to $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ will never happen.

The i first bits of m_0 are randomly sampled by $\mathcal{R}$ as per the definition of $\mathrm{G}_{2,i}$. For the $i+1$-th bit, the reduction computes the context $\mathsf{cnt} = (i+1, \mathsf{nonce})$ and sets $\mathsf{ct} = \mathsf{ot}_{i+1}$. It then picks a random bit $\bar{b}$ as sets the $i+1$-th bit of m_0 to $\bar{b}$. It behaves according to the protocol for the rest of the bits and uses the $\mathcal{O}_{\mathsf{sk}}$ oracle provided by the NIOT challenger. Note that it will never ask the oracle for the context cnt. At the end of the experiment, the adversary $\mathcal{A}$ outputs the bit d distinguishing between $\mathrm{G}_{2,i}$ and $\mathrm{G}_{2,i+1}$. If $d = 0$, the reduction communicates $(\mathsf{ct}, \mathsf{cnt}, \bar{b})$ to the recipient privacy challenger of the NIOT scheme. Otherwise, it sends $(\mathsf{ct}, \mathsf{cnt}, 1 - \bar{b})$. Note that if $\bar{b}$ is the correct bit in the NIOT, then the reduction perfectly simulated $\mathrm{G}_{2,i}$ and otherwise, if $1 - \bar{b}$ is the correct bit then we are in $\mathrm{G}_{2,i+1}$, which the distinguisher $\mathcal{A}$ can identify.

The full claim follows by a hybrid argument over all ℓ_m instances of the NIOT protocol.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_{2,ℓ_m} and G_{3,ℓ_m} only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{3,\ell_m}(\mathcal{A}) = 1] - \Pr[\mathrm{G}_{2,\ell_m}(\mathcal{A}) = 1]| \leq \ell_m \cdot \mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}}.$$

We can use the same arguments as above.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrid G_{3,ℓ_m} is $1/2$, i.e.:

$$\Pr[\mathrm{G}_{3,\ell_m}(\mathcal{A})] = 1/2.$$

The claim follows since signatures sig_0 and sig_1 are fresh signatures under random messages independent of the bit b. The only thing adversary $\mathcal{A}$ can do is guess bit b.

Theorem 9. *Scheme 6 is nonce blind against a honest-but-curious adversary (Definition 18) assuming the non-interactive oblivious transfer scheme is receiver secure and correct, and the* PRF *is pseudorandom.*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the honest nonce blindness of NIBS.

Game $\mathbf{G_0}$: The original honest blindness experiment $\mathrm{G}_0 := \mathbf{Adv}^{\mathsf{HNBnd}}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G_1}$: Intead of running the Issue and Obtain algorithms to compute (m_0, sig_0) and (m_1, sig_1), we compute them directly. First, we compute $m_0 = \bar{\mathsf{sk}}_{R,\mathsf{nonce}_0}$, $m_1 = \bar{\mathsf{sk}}_{R,\mathsf{nonce}_1}$ and then use the secret key sk given by adversary to compute $\mathsf{sig}_0 = \mathsf{SIG.Sign}(\mathsf{sk}, m_0)$ and $\mathsf{sig}_1 = \mathsf{SIG.Sign}(\mathsf{sk}, m_1)$. In case, any of the pre-signatures cannot be finalized to a valid signature, we set $\mathsf{sig}_0 = \bot$ and $\mathsf{sig}_1 = \bot$.

Game $\mathbf{G_{2,i}}$: In each sub-hybrid $i \in [\ell_m]$, we replace the bit $\bar{\mathsf{sk}}_{R,i}$ with a uniformly random one and compute m_0 using the changed key. Note the in hybrid G_{2,ℓ_m} the key $\bar{\mathsf{sk}}_{R,\mathsf{nonce}}$ is random and independent of the public key pk_R.

We will show the indistinguishability of the hybrids via a sequence of claims.

Claim. We claim that hybrids G_0 and G_1 are indistinguishable, i.e.,

$$|\Pr[\mathrm{G}_1(\mathcal{A}) = 1] - \Pr[\mathrm{G}_0(\mathcal{A}) = 1]| \leq 2 \cdot \mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}.$$

Since the adversary is honest-but-curious, it follows that the only way for the adversary to distinguish this change is by using sig_0 and sig_1. However, the claim follows since the final signatures are randomized using Rand, and we assume that the adversary only has an advantage $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}$ in distinguishing randomized and "fresh" signatures.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_1 and G_{2,ℓ_m} only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{2,\ell_m}(\mathcal{A}) = 1] - \Pr[\mathrm{G}_1(\mathcal{A}) = 1]| \leq \ell_m \cdot \mathbf{Adv}^{\mathsf{RcvSec}}_{\mathcal{A},\mathsf{NIOT}}.$$

The claim follows similarly like in the case of honest recipient blindness above.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrid G_{2,ℓ_m} is $1/2$, i.e.:

$$\Pr[\mathrm{G}_{2,\ell_m}(\mathcal{A})] = 1/2.$$

The claim follows since signatures sig_0 and sig_1 are fresh signatures under random messages independent of the bit b. The only thing adversary $\mathcal{A}$ can do is guess bit b.

6 Generic Transformation from Honest to Malicious NIBS

In this section we present a generic transformation, using non-interactive zero-knowledge proofs, to turn any semi-honest NIBS into a maliciously secure one.

Theorem 10 (Recipient Blindness). *Scheme 7 is recipient blind (Definition 11) assuming the used NIBS' scheme is recipient blind against an honest-but-curious adversary (Definition 17) and Π is a proof of knowledge.*

Let $\mathsf{NIBS}' = (\mathsf{KeyGen}', \mathsf{RKeyGen}', \mathsf{Issue}', \mathsf{Obtain}', \mathsf{Verify}')$ be a non-interactive blind signature secure against a honest-but-curious adversary. Moreover, let $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ be a non-interactive zero-knowledge proof of knowledge for the language $\mathcal{L}_{\mathcal{R}}$, where we define the relation $\mathcal{R}$ as:

$$((\mathsf{psig}, \mathsf{nonce}, \mathsf{pk}_R, \mathsf{pk}), (r, \mathsf{sk})) \in \mathcal{R} \iff \mathsf{psig} \leftarrow \mathsf{Issue}'(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce}; r) \ \wedge \ (\mathsf{sk}, \mathsf{pk}) \text{ is a valid keypair}$$

$\underline{\mathsf{KeyGen}(\lambda)}$**:** return $\mathsf{KeyGen}'(\lambda)$.

$\underline{\mathsf{RKeyGen}(\lambda)}$**:** return $\mathsf{RKeyGen}'(\lambda)$.

$\underline{\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})}$**:** choose random r generate pre-signature $\mathsf{psig}' \leftarrow \mathsf{Issue}'(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce}; r)$ and compute proof $\pi \leftarrow \Pi.\mathsf{Prove}((\mathsf{psig}', \mathsf{nonce}, \mathsf{pk}_R, \mathsf{pk}), (r, \mathsf{sk}))$. Return $\mathsf{psig} = (\mathsf{psig}', \pi)$.

$\underline{\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})}$**:** parse psig as (psig', π). Return $\perp$ if $\Pi.\mathsf{Verify}((\mathsf{psig}', \mathsf{nonce}, \mathsf{pk}_R, \mathsf{pk}), \pi) = 0$. Otherwise, return $\mathsf{Obtain}'(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}', \mathsf{nonce})$.

$\underline{\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig}))}$**:** output $\mathsf{Verify}'(\mathsf{pk}, (m, \mathsf{sig}))$.

Scheme 7: Generic Transformation from Honest to Malicious NIBS

Proof (Sketch). The proof follows by constructing a reduction algorithm $\mathcal{R}$ that takes as input an adversary $\mathcal{A}$ against the recipient blindness property of Scheme 7. The reduction plays the role of the adversary in the honest recipient blindness experiment for the scheme NIBS'. At the beginning of which, it receives public keys $(\mathsf{pk}_{R_0}, \mathsf{pk}_{R_1})$, which it provides to the adversary $\mathcal{A}$. In return, it receives $(\mathsf{psig}_0 = (\mathsf{psig}'_0, \pi_0), \mathsf{nonce}_0, \mathsf{psig}_1 = (\mathsf{psig}'_1, \pi_1), \mathsf{nonce}_1, \mathsf{pk})$. The reduction then uses the extraction algorithm to extract r_0, r_1 and sk such that $\mathsf{psig}'_0 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{R_0}, \mathsf{nonce}_0, \mathsf{sk}; r_0)$, $\mathsf{psig}'_1 \leftarrow \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{R_1}, \mathsf{nonce}_1, \mathsf{sk}; r_1)$ and aborts in case of failure. Note that due to the properties of Π, the reduction algorithm will abort with only a negligible probability. The reduction sends $(r_0, \mathsf{nonce}_0, r_1, \mathsf{nonce}_1, (\mathsf{sk}, \mathsf{pk}))$ to its challenger. It then receives $((m_b, \mathsf{sig}_b), (m_{1-b}, \mathsf{sig}_{1-b}))$ from the challenger and forward it to $\mathcal{A}$. Finally, the adversary outputs bit b, which is also the output of the reduction $\mathcal{R}$ to its challenger. It is easy to see that if $\mathcal{A}$ breaks the recipient blindness of Scheme 7, then R breaks the recipient blindness of the NIBS' scheme

Theorem 11 (Nonce Blindness). *Scheme 7 is nonce blind (Definition 12) assuming the used* NIBS' *scheme is nonce blind against an honest-but-curious adversary (Definition 18) and* Π *is a proof of knowledge.*

Proof (Sketch). The proof follows a similar strategy as the one for recipient blindness.

Theorem 12 (One-more Unforgeability). *Scheme 7 is one-more unforgeable assuming the used* NIBS' *scheme is one-more unforgeable and* Π *is zero-knowledge.*

Proof (Sketch). We will show this proof by constructing a reduction algorithm $\mathcal{R}$ that takes as input an adversary $\mathcal{A}$ against the one-more unforgeability of Scheme 7. The reduction plays the role of the adversary in the one-more unforgeability experiment for the NIBS' scheme. Initially, the reduction receives the public key pk from the challenger and sends it directly to $\mathcal{A}$. Then, for any signing query $(\mathsf{pk}_R, \mathsf{nonce})$ made by adversary $\mathcal{A}$, the reduction asks its oracle on the same input receiving pre-signature psig'. It then uses the zero-knowledge simulator for the proof system Π to simulate proof π. It returns $\mathsf{psig} = (\mathsf{psig}', \pi)$ as the adversary's query. Finally, $\mathcal{A}$ returns ℓ message-signature pairs that are also the output of the reduction.

7 Efficient NIBS Scheme

This section focuses on constructing a non-interactive blind signature scheme NIBPS that is secure against malicious adversaries. However, we do not opt for a naive approach, using the generic construction with zero-knowledge proof of honest behavior by the signer. Nevertheless, we will build the scheme in this section by directly instantiating the generic construction and using the building blocks in a non-black-box way.

We observe that the only strategy of an adversary to break the blindness property is to use selective-abort attacks, as we already mentioned. The intuition here is that the final message corresponds to the bit choices in the NIOT scheme, which are hidden in the NIOT protocol. The idea is to make the final message and signature computationally unlinkable to the pre-signature (i.e., the recipient's NIOT choice). We achieve this with universal hashing and the leftover hash lemma, i.e., even if the malicious signer learns some bits of the input, the final message will still be computationally close to uniform for the adversary.

To achieve this we need to rely on remark 6 and use a simple universal hash function $\mathcal{H}_{\mathsf{uni}}(x_1, x_2) = \alpha_1 \cdot x_1 + \beta_1 + \alpha_2 \cdot x_2 + \beta_2$ over the field $\mathbb{F}_q$. The function combines two simple universal functions to increase the input size of the recipient and, consequently, the entropy for the LHL. The parameters for the function are computed using a random oracle $\mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, i) \in \mathbb{F}_q$ for $i \in \{0, 1, 2, 3\}$.

Since we require the function $\mathcal{H}_{\mathsf{uni}}$ to work over a finite field, the underlying signature we pick is the PS signature. Therefore, the signer's public key is $\mathsf{pk} = (X = g_2^x, Y = g_2^y, V_1 = \mathsf{H}_{\mathbb{G}_1}(X)^x, V_2 = \mathsf{H}_{\mathbb{G}_1}(Y)^y)$ and the secret key $\mathsf{sk} = (x, y)$, as in PS signatures. We add the components V_1 and V_2 as proof of possession of the secret key and also a way for the reduction to obtain g_1^x and g_1^y, which can be used as a secret key for the PS scheme. Note we can get an equivalent PS signatures by computing $\sigma_1 = g_1^h$ for some $h \leftarrow\!\!\$\, \mathbb{Z}_q$ and setting $\sigma_2 = ((g_1^x) \cdot (g_1^y)^m)^h$. The recipient's public key pk_R is a standard RSA modulus $N = pq$, and the corresponding secret key is $\mathsf{sk}_R = (p, q)$.

The pre-signature is created in a similar way to the generic construction. The signer first picks random keys $k_1^0, k_1^1, \ldots, k_\kappa^0, k_\kappa^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$ for each potential input of the recipient, where we assume that κ is the bit-size corresponding to the bit-size of two elements from $\mathbb{F}_q$ (the order of the group used in the PS signature). We now directly use the idea from the NIOT Scheme 3, i.e., for each input index $i \in [\kappa]$ we compute the element $x_i = \mathsf{H}_N(N, \mathsf{nonce}, i) \in \mathbb{Z}_N$ and then use the Goldwasser-Micali encryption scheme to encrypt choice bit 0, i.e., the key k_i^0, as $n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i))$. The opposite choice and key k_i^1 is encrypted using the Cocks scheme as $n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i))$. In both cases, to compute n_i^0 and n_i^1 uses random coins from a random oracle query based on respective keys k_i^0 and k_i^1. Thanks to this, the recipient can verify the validity of the ciphertexts n_i^0 and n_i^1.

To ensure that the recipient's public key is well-formed (i.e., N is squarefree), we require the recipient to compute values $o_1, \ldots, o_\lambda \in \mathbb{Z}_N$ while provided $o_i^N \bmod N$ in the pre-signature (as mentioned in the discussion of Scheme 3). Those values are needed to compute that AE secret keys $K_i^0 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^0)$ and $K_i^1 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^1)$ for each of the input bits of the recipient $i \in [\kappa]$. The signer uses the keys (K_i^0, K_i^1) to encrypt the garbled circuit values for the PS signatures, i.e., for each $i \in [\kappa]$, the signer computes the ciphertexts for the garbled circuit payloads. The pre-signature then contains the element o_i^N, the AE ciphertexts $(\mathsf{ct}_i^0, \mathsf{ct}_i^1)$ containing the garbling circuit payloads, the Goldwasser-Micali and Cocks ciphertexts (n_i^0, n_i^1), the random value $\sigma_1 = h \in \mathbb{G}_1$ from the PS signature and the value $s_0 = a_0 \cdot h^x \cdot (h^y)^{\beta_1+\beta_2}$.

To construct the final signature from the pre-signature psig, the recipient first computes the values o_i using the inverse $1/N \bmod \phi(N)$. Note that this inverse only exists if N is coprime to $\phi(N)$, which only exists if N is squarefree. The recipient then uses the factorization of N to identify for each $I \in [\kappa]$ if the i-th bit of its choice for the nonce nonce corresponds to 0 or 1. It decrypts n_i^0 using the Goldwasser-Micali encryption scheme in the former case and sets $m_i = 0$. Otherwise, it uses the Cocks scheme and sets $m_i = 1$. In the end, it receive the keys $k_1, \ldots, k_\kappa$ and computes the corresponding AE key $K_i = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i)$. It then for each $i \in [\kappa]$ decrypts both ct_i^0 and ct_i^1. Note that due to the permutation bit t, the recipient can only identify the correct ciphertext by decrypting it and checking if the result is not $\perp$. In case both ciphertexts decrypt to $\perp$, the recipient aborts. Otherwise, it has values $s_1, \ldots, s_\kappa$. It sets σ_1 and computes $\sigma_2 = s_0 \cdot \prod_{i=1}^{\kappa} s_i$. Before constructing the final output, the PS signature is re-randomize (similar to Rand in the generic construction) and set the final signature to $\mathsf{sig} = (\sigma_1^r, \sigma_2^r)$ for some $r \in \mathbb{Z}_q$. As the last part, the recipient computes the final message $m = \alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2 \bmod q$, where $l_1 = m_1 \ldots m_{\ell_q}$ and $l_2 = m_{\ell_q+1} \ldots m_\kappa$. The recipient, of course, aborts in case the pair (m, sig) is invalid, i.e., $\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig})) = 0$.

Theorem 13. NIBPS *is reusable (Definition 9).*

Proof. The proof follows from the collision resistance of the hash function H_N. The other proofs assume this function is a random oracle, so collision resistance

follows. We now see that for two different nonces nonce_0 and nonce_1, the input bits will differ, and so by the collision resistance of the universal function will the messages m_0 and m_1.

Theorem 14. *In the random oracle model,* NIBPS *is one-more unforgeable (Definition 10) under Assumption 1 (Definition 6) and assuming* AE *is indistinguishable under chosen plaintext attacks, the Goldwasser-Micali and the Cocks cryptosystems are indistinguishable.*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the one-more unforgeability of NIBS. Assume that $\mathcal{A}$ makes at most $Q_s = \mathsf{poly}(\lambda)$ signing queries and at most Q_{AE} queries to the H_{AE} random oracle.

Game $\mathbf{G}_0$: The original one-more unforgeability experiment $\mathrm{G}_0 := \mathsf{OM\text{-}UNF}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G}_1$: We abort the experiment if there are collisions in the random oracle H_{AE}.

Game $\mathbf{G}_2$: We now modify the experiment significantly. The challenger initializes empty lists L_{EQR} and Fail_N. Then for any query $\mathsf{H}_N(N, \mathsf{nonce}, i)$ it first checks if $(\mathsf{pk}_R, \cdot) \in L_{\mathsf{EQR}}$. If not, it proceeds as follows. The challenger wants to learn a quadratic non-residue modulo N. Therefore, it creates a copy of the adversary $\mathcal{A}_C$, and it then waits for a $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$ for which it replies according to the experiment while retaining the $o_1, \ldots, o_\lambda$ and nonce values. It proceeds the execution of the experiment with $\mathcal{A}_C$ until it makes a query to the random oracle for $\mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^0)$ for any $i \in [\kappa]$. It then stores $y_N = \mathsf{H}_N(N, \mathsf{nonce}, i)$ together with $\mathsf{pk}_R = N$ in list L_{EQR}, i.e., the challenger adds (N, y_N) to L_{EQR}. After that, the reduction destroys the copy and uses the original adversary. If there is no such query, the challenger tries again with another copy and different random coins. If after λ times, the copy does not make such a query, the challenger adds pk_R to Fail_N. In such a case, the reduction replies $\mathsf{H}_N(N, \mathsf{nonce}, i)$ with a randomly sampled output.

Game $\mathbf{G}_3$: We now modify the previous experiment. For each $\mathsf{H}_N(N, \mathsf{nonce}, i)$ where $(N, x_N, y_N) \in L_{\mathsf{EQR}}$, we randomly sample a bit $b_{N,\mathsf{nonce},i} \leftarrow\!\!\$\ \{0,1\}$, compute $x_N = u^2 \mod N$ for a random $u \leftarrow\!\!\$\ \mathbb{Z}_N^*$ and set the output for $\mathsf{H}_N(N, \mathsf{nonce}, i)$ as x_N if $b_{N,\mathsf{nonce},i} \leftarrow\!\!\$\ \{0,1\} = 1$ and $x_N \cdot y_N \mod N$ otherwise. If the oracle H_N is defined for all $i \in [\kappa]$ (i.e., $\mathsf{H}_N(N, \mathsf{nonce}, j)$ is not defined for some $j \in [\kappa]$), the challenger does the above for all $i \in [\kappa]$. It keeps all values $b_{N,\mathsf{nonce},i}$.

Game $\mathbf{G}_4$: Again, we will change the experiment a bit. For each query $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$ of the adversary, the challenger sets the following strings $l_1 = b_{N,\mathsf{nonce},1} \ldots b_{N,\mathsf{nonce},\ell_q}$, $l_2 = b_{N,\mathsf{nonce},\ell_q+1} \ldots b_{N,\mathsf{nonce},\kappa}$ and computes message

$$m_{N,\mathsf{nonce}} = (\alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2) \mod q,$$

where $\alpha_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 0)$, $\beta_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 1)$ and $\alpha_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 2)$, $\beta_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 2)$.

Game $\mathbf{G}_5$: We now change how the challenger computes the values V_1 and V_2 that are part of the public key pk. Instead of computing them directly as respectively $(\mathsf{H}_{\mathbb{G}_1}(X))^x$ and $(\mathsf{H}_{\mathbb{G}_1}(Y))^y$). It computes V_2 as $(g_1^y)^{r_1}$ for some $r_1 \leftarrow\!\!\$\ \mathbb{Z}_q$ and then programs the random oracle $\mathsf{H}_{\mathbb{G}_1}$ to output $g_1^{r_1}$ for Y. To compute the value V_1, the challenger computes a PS signature under message 0, i.e., (h, h^x) and then sets $V_1 = (h^x)^{r_2}$ and programs the random oracle $\mathsf{H}_{\mathbb{G}_1}$ to output h^{r_2} for a query X.

Game $\mathbf{G}_6$: We now introduce an abort event for the experiment. The challenger aborts the experiment if for a query $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$ the adversary asks the random oracle for $\mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^0)$ and $\mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^1)$ for some $i \in [\kappa]$. Otherwise, it keeps a list $L_{\mathsf{nonce},\mathsf{AE}}$ with entries (i, b) where the adversary makes a $\mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^{1-b})$ query.

Game $\mathbf{G}_7$: We modify the experiment in a way that the challenger for all queries $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$, and all $(i, b) \in L_{\mathsf{nonce},\mathsf{AE}}$ replaces the ciphertext $\mathsf{AE.Enc}(K_i^b, s_i^b)$ with $\mathsf{AE.Enc}(K_i^b, 0)$. We will show the indistinguishability of the hybrids via a sequence of claims.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_0 and G_1 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_1(\mathcal{A}) = 1] - \Pr[\mathrm{G}_0(\mathcal{A}) = 1]| \leq \frac{Q_{\mathsf{AE}}^2}{2^\lambda}$$

The claim follows from the number of collisions that can occur for Q_{AE} queries.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_1 and G_2 is the same, i.e.,:

$$\Pr[\mathrm{G}_2(\mathcal{A}) = 1] = \Pr[\mathrm{G}_1(\mathcal{A}) = 1].$$

The claim follows since our change in G_2 is just conceptual.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_2 and G_3 is the same, i.e.,:

$$\Pr[\mathrm{G}_3(\mathcal{A}) = 1] = \Pr[\mathrm{G}_2(\mathcal{A}) = 1].$$

The claim follows since our change in G_3 is just conceptual..

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_3 and G_4 is the same, i.e.,:

$$\Pr[\mathrm{G}_4(\mathcal{A}) = 1] = \Pr[\mathrm{G}_3(\mathcal{A}) = 1].$$

The claim follows since our change in G_4 is just conceptual.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_4 and G_5 is the same, i.e.,:

$$\Pr[\mathrm{G}_5(\mathcal{A}) = 1] = \Pr[\mathrm{G}_4(\mathcal{A}) = 1].$$

The claim follows since our change in G_5 is just conceptual since the elements we program the random oracle with are uniformly random.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_5 and G_6 only differs by a negligible factor, i.e.:

$$|\Pr[G_6(\mathcal{A}) = 1] - \Pr[G_5(\mathcal{A}) = 1]| \leq \frac{1}{2^\lambda}$$

The claim follows from the following observation. The only way for an adversary to get both keys k_i^0 and k_i^1 without guessing them is to decrypt both ciphertext n_i^0 and n_i^1. However, for squarefree N, we know it is impossible since at least one n_i^b hides the messages statistically (this is the base idea behind our NIOT construction, see Proposition 1). Therefore, to get those keys, the public key pk_R is picked so that N is not squarefree. However, since the adversary also computed $o_1, \ldots, o_\lambda$ give $o_1^N, \ldots, o_\lambda^N$ this means that N is squarefree. The probability that given a single o_1^N, the adversary can compute o_1 is $1/p_1$, where p_1 is the smallest factor of N. Setting $p_1 = 2$, the claim follows by applying λ parallel repetitions so we get $(1/p_1)^\lambda \leq (1/2)^\lambda$.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_6 and G_7 only differs by a negligible factor, i.e.:

$$|\Pr[G_7(\mathcal{A}) = 1] - \Pr[G_6(\mathcal{A}) = 1]| \leq Q_s \cdot \kappa \cdot \mathbf{Adv}_{\mathsf{AE},\mathsf{cpa}}(\mathcal{A}).$$

The claim follows since there are at most Q_s signing queries, and for each one, the list $L_{\mathsf{nonce},\mathsf{AE}}$ has at most κ elements. The claim follows by changing each ciphertext at once and constructing a reduction to the CPA security of AE.

Claim. We now show that an adversary $\mathcal{A}$ against the one-more unforgeability of NIBS in G_7 can be used to break Assumption 1 (Definition 6). In other words, we have:

$$\Pr[G_7(\mathcal{A})] \leq \mathbf{Adv}_{\mathsf{Ass}}(\mathcal{A}).$$

To prove this claim, we construct a reduction algorithm $\mathcal{R}$ that uses $\mathcal{A}$ and breaks Assumption 1. The reduction is given $(g_1, g_2, Y_1, X_2, Y_2)$ as input and then uses computes the public key $\mathsf{pk} = (X, Y, V_1, V_2)$ by setting $X = X_2$, $Y = Y_2$ and computing V_1 and V_2 as in G_5. Note that $\mathcal{R}$ can use the oracle provided by the assumption to compute a signature under 0 by querying the oracle with $\mathcal{O}(0 \in \mathbb{Z}_q)$.

At some point the adversary $\mathcal{A}$ will query the signing oracle with $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$. For each query, the reduction learns message $m_{N,\mathsf{nonce}}$ (as per G_4) before it outputs the pre-signature psig for that query. Thus, it asks the oracle $\mathcal{O}$ with $m_{N,\mathsf{nonce}}$ and receives (σ_1, σ_2) for which $e(\sigma_1, X \cdot Y^{m_{N,\mathsf{nonce}}}) = e(\sigma_2, g_2)$. The reduction then samples $a_1, \ldots, a_\kappa \leftarrow\!\!\$\ \mathbb{G}_1$ and computes $a_0 = 1_{\mathbb{G}_1} \cdot (\prod_{i=1}^{\kappa} a_i)^{-1}$. It then computes the shares $s_i = a_i$ for all $i \in [\kappa]$ and sets $s_0 = a_0 \cdot \sigma_2$. The rest of the values are computed as in the experiment.

At the end the adversary outputs ℓ message-signature pairs $(m_1, \mathsf{sig}_1), \ldots, (m_\ell, \mathsf{sig}_\ell)$, where it only makes $\ell - 1$ queries to the oracle $\mathcal{O}_1(\mathsf{sk}, \mathsf{pk}_R = N, \mathsf{nonce})$. Due to

the check in Verify for all $i \in [\kappa]$ we have $m_i \neq 0$. Therefore, there exists a single $j \in [\ell]$ for which $m_j \notin m_{N,\mathsf{nonce}}$ for any pair (N, nonce). The reduction can output (m_j, sig_j) as a valid solution to Assumption 1 since the reduction never queried m_j to the oracle $\mathcal{O}$ provided by the assumption.

Theorem 15. *In the random oracle model,* NIBPS *is recipient blind (Definition 11) assuming the extended quadratic residuosity problem (Definition 5).*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the recipient blindness of NIBS. We assume without loss of generality that the adversary outputs pk for which $e(V_1, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(X), X)$ and $e(V_2, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(Y), Y)$. Note that otherwise, the recipient always aborts and the only strategy for the adversary is to guess the bit b and the proof holds. We also assume that $\mathcal{A}$ makes at most $Q_{\mathsf{AE}}, Q_q, Q_N, Q_R$ queries to the respective random oracles and Q_{NIBS} queries to the $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ oracle.

Game $\mathbf{G}_0$: The original recipient blindness experiment $\mathrm{G}_0 := \mathbf{Adv}^{\mathsf{RBnd}}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G}_1$: We abort the experiment if there are collisions in the random oracles H_{AE}, H_q, H_N or H_R.

Game $\mathbf{G}_{2,0}$: We slightly modify the experiment from G_1. In this experiment, for the presignature-nonce pair $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the challenger decrypts the ciphertexts ct_i^0 and ct_i^1 obtaining all values s_i^0 and s_i^1, where the challenger finds the key by looking through the list of outputs of the random oracle H_{AE}. Note that AE will only accept the correct key. Additionally, the challenger keeps a set $\mathsf{Fail}_{\mathsf{psig}_0}$ with tuples (i, b), meaning that for $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the ciphertext ct_i^b cannot be decrypted, i.e., there exists no key for it in the random oracle H_{AE}. For pairs $(i, b) \notin \mathsf{Fail}_{\mathsf{psig}_0}$, the challenger learn the key k_i^b. For those values, the challenger additionally checks the validity of the ciphertexts n_i^0 and n_i^1, i.e., that $n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0))$ and $n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i), \mathsf{H}_R(i, k_i^1))$. In case for some pair $(i, b) \notin \mathsf{Fail}_{\mathsf{psig}_0}$, any of the ciphertext is wrong, the challenger adds (i, b) to $\mathsf{Fail}_{\mathsf{psig}_0}$.

Game $\mathbf{G}_{2,1}$: Similar to $\mathrm{G}_{2,0}$ but for $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_3$: Similar to $\mathrm{G}_{2,1}$ but for all Q_{NIBS} queries to the oracle $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$.

Game $\mathbf{G}_{4,0}$: We now slightly modify G_3. In this experiment, for the presignature-nonce pair $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the challenger checks the shares s_i^0 and s_i^1. To do so, for each $i \in [\kappa]$ the challenger picks random bits $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_\kappa \leftarrow\!\!\$\ \{0,1\}$, where we ensure that for each of them we have $(j, m_j) \notin \mathsf{Fail}_{\mathsf{psig}_0}$. The challenger then sets $m_i = 0$ and computes message $m = (\alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2)$, where $l_1 = m_1 \ldots m_{\ell_q}$, $l_2 = m_{\ell_q+1} \ldots m_\kappa$. It then uses the shares s_i^0, s_i^1 to compute the signature sig under m (as in Obtain). The challenger repeats the process λ times for different random $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_\kappa \leftarrow\!\!\$\ \{0,1\}$. It adds $(i, 0)$ to $\mathsf{Fail}_{\mathsf{psig}_0}$ if any generated signatures fail verification. It then repeats the process for $m_i = 1$ and all other positions $i \in [\kappa]$, making sure only to pick the values

not already in the set $\mathsf{Fail}_{\mathsf{psig}_0}$. This way, the challenger checks which shares s_i^0 and s_i^1 are valid and can be used to obtain valid signatures.

With this change, the challenger can check if for psig_0 and given input bits $m_1, \ldots, m_\kappa$ the Obtain algorithm should abort even without knowing the secret key $\mathsf{sk}_{R,0}$. To do so, the challenger can check for all $i \in [\kappa]$ the pair $(i, m_i) \notin \mathsf{Fail}_{\mathsf{psig}_0}$. If the check fails for at least one i, then psig_0 fails to produce a valid signature for input bits $m_1, \ldots, m_\kappa$ of the recipient, and the challenger knows that in such a case, it needs to return $\perp$ (as mentioned this can be done without the challenger knowing the recipient secret key sk_{R_0}).

Game $\mathbf{G}_{4,1}$: Similar to $\mathrm{G}_{4,0}$ but for $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_5$: Similar to $\mathrm{G}_{4,1}$ but for all Q_{NIBS} queries to the oracle Q_{NIBS}.

Game $\mathbf{G}_6$: Similar to the previous experiment, but we abort if the size of any of the sets $|\{(i,b) : (i,b) \in \mathsf{Fail}_{\mathsf{psig}_0}\}|$ or $|\{(i,b) : (i,b) \in \mathsf{Fail}_{\mathsf{psig}_1}\}|$ is bigger than λ.

Game $\mathbf{G}_7$: We slightly modify the experiment from G_6. To answer queries to the $\mathsf{H}_{\mathbb{G}_1}(x)$ oracle, the challenger samples a random $r \leftarrow\!\!\$\, \mathbb{Z}_q$ and outputs g_1^r. It then stores (x, r) on list $L_{\mathsf{H}_{\mathbb{G}_1}}$. Given the public key $\mathsf{pk} = (X, Y, V_1, V_2)$ of the adversary the challenger computes $X_1 = V_1^{r_1^{-1}}$ and $Y_1 = V_2^{r_2^{-1}}$, where $(X, r_1) \in L_{\mathsf{H}_{\mathbb{G}_1}}$ and $(Y, r_2) \in L_{\mathsf{H}_{\mathbb{G}_1}}$. Note that we assume that $e(V_1, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(X), X)$ and $e(V_2, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(Y), Y)$, so the elements (X, r_1) and (Y, r_2) exist in $L_{\mathsf{H}_{\mathbb{G}_1}}$ and we have $X_1 = g_1^x$ and $Y_1 = g_1^y$.

Game $\mathbf{G}_8$: We modify experiment G_7 in the way the challenger computes the signatures sig_0 and sig_1 in case $\mathsf{sig}_0 \neq \perp$ and $\mathsf{sig}_1 \neq \perp$. Before providing $((m_b, \mathsf{sig}_b), (m_{1-b}, \mathsf{sig}_{1-b}))$ the challenger uses X_1 and Y_1 to compute fresh signatures as:

$$\mathsf{sig}_0 = (g_1^{h_0}, (X_1)^{h_0} \cdot (Y_1)^{h_0 \cdot m_0}) \qquad \mathsf{sig}_1 = (g_1^{h_1}, (X_1)^{h_1} \cdot (Y_1)^{h_1 \cdot m_1})$$

for $h_0, h_1 \leftarrow\!\!\$\, \mathbb{Z}_q$.

Game $\mathbf{G}_{9,i}$: We now define subgames $i \in [\kappa - \lambda]$, and in $\mathrm{G}_{9,i}$, we change how the challenger executes the Obtain algorithm. For each j, where $(j, \cdot) \notin \mathsf{Fail}_{\mathsf{psig}_0}$ and not already changed, we replace parts of the Obtain algorithm run by challenger for $(\mathsf{psig}_0, \mathsf{nonce}_0)$. Instead of setting s_j to either s_j^0 or s_j^1 (note that the challenger knows both values due to prior changes) based on if $x_j = \mathsf{H}_N(N, \mathsf{nonce}_0, j)$ is a quadratic residue modulo N or not, we sample a uniformly random bit b_j and set $s_j = s_j^{b_j}$. Note that due to this change, there are $[\kappa - \lambda]$ input bits that the challenger picks independently of the recipient's public key.

Game $\mathbf{G}_{10,i}$: We make similar changes as above but for message $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_{11}$: We sample the message m_0 uniformly from $\mathbb{Z}_q$.

Game $\mathbf{G}_{12}$: We sample the message m_1 uniformly from $\mathbb{Z}_q$.
We will show the indistinguishability of the hybrids via a sequence of claims.

Claim. We claim that hybrids G_0 and G_1 are indistinguishable. Let $Q_{\mathsf{AE}}, Q_q, Q_N, Q_R$ denote an upper bound on the number of queries the adverasary makes to the oracles H_{AE}, H_q, H_N, H_R, we then have:

$$|\Pr[\mathrm{G}_1(\mathcal{A}) = 1] - \Pr[\mathrm{G}_0(\mathcal{A}) = 1]| \leq \frac{Q_{\mathsf{AE}}^2}{2^\lambda} + \frac{Q_q^2}{2^{\ell_q}} + \frac{Q_N^2}{2^{|\mathcal{J}_N|}} + \frac{Q_R^2}{2^\lambda}.$$

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_1 and G_5 is the same, i.e., :

$$\Pr[\mathrm{G}_5(\mathcal{A}) = 1] = \Pr[\mathrm{G}_1(\mathcal{A}) = 1].$$

The changes in games $\mathrm{G}_{2,0}$, $\mathrm{G}_{2,1}$, G_3, $\mathrm{G}_{4,0}$, $\mathrm{G}_{4,1}$ and G_5 are just conceptual and do not change $\mathcal{A}$'s advatange.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_6 and G_5 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_6(\mathcal{A}) = 1] - \Pr[\mathrm{G}_5(\mathcal{A}) = 1]| \leq 2 \cdot \frac{1}{2^\lambda}.$$

We observe that if there are more than λ elements in the set $|\{(i,b) : (i,b) \in \mathsf{Fail}_{\mathsf{psig}_0}\}|$ then that means that there are more than λ input bits where the adversary needs to guess the input bits of the recipients exactly. However, this only happens with negligible probability. In particular, if the adversary uses such a strategy, the recipient will always abort the protocol (except with negligible probability), which would lead to a zero advantage of $\mathcal{A}$. Same for the other set. Thus, we can bind the probability of this event as in the claim.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_6 and G_8 is the same, i.e., :

$$\Pr[\mathrm{G}_8(\mathcal{A}) = 1] = \Pr[\mathrm{G}_6(\mathcal{A}) = 1].$$

The changes in game G_7 are just conceptual. Note that the challenger can always compute the values X_1 and Y_1 because of the consistency check of the public key pk in Obtain. On the other hand, the change in G_8 follows from the perfect randomization of PS signatures, i.e., the final randomization with the exponent r in Obtain samples a uniform random signature from the signature space for the given message. In other words, an adversary can notice this change with advantage $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}}$, but we know that $\mathbf{Adv}_{\mathcal{A},\mathsf{Rand}} = 0$ for PS signatures, so the claim follows.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids $\mathrm{G}_{9,\kappa-\lambda}$ and G_8 only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{9,\kappa-\lambda}(\mathcal{A}) = 1] - \Pr[\mathrm{G}_8(\mathcal{A}) = 1]| \leq (\kappa - \lambda) \cdot \mathbf{Adv}_{\mathcal{R}_{\mathsf{EQR}},\mathsf{EQR}}.$$

To show this claim, we will show that the difference for each change, i.e., between $\mathrm{G}_{9,i+1}$ and $\mathrm{G}_{9,i}$ is $\mathbf{Adv}_{\mathcal{A},\mathsf{EQR}}$ and since there are $\kappa - \lambda$ subgames, the claim follows.

To show the difference between $\mathrm{G}_{9,i+1}$ and $\mathrm{G}_{9,i}$ we will construct a reduction $\mathcal{R}_{\mathsf{EQR}}$ that breaks the extended quadratic residuosity assumption (Definition 5) using an adversary $\mathcal{A}$ that is able to distinguish between $\mathrm{G}_{9,i+1}$ and $\mathrm{G}_{9,i}$. The input to the reduction is an instance N, x_N, y_N for the EQR problem, where $x_N, y_N \in \mathcal{J}_N$ and $y \notin \mathrm{QR}_N$. The reduction will use this modulus as the recipient's public key, i.e., it sets $\mathsf{pk}_{R_0} = N$. The problem we now encounter is that the reduction is unable to answer calls to $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ and also perform the computation $(m_0, \mathsf{sig}_0) \leftarrow \mathsf{Obtain}(\mathsf{sk}_{R_0}, \mathsf{pk}, \mathsf{psig}_0, \mathsf{nonce}_0)$. The reason is that it does not know the corresponding secret key sk_{R_0}.

To solve this issue, we first need to solve a different problem, i.e., how to sample random elements from $\mathcal{J}_N \cap \mathrm{QR}_N$ and $\mathcal{J}_N \cap \neg\, \mathrm{QR}_N$. For the former, the reduction can sample a random element $z \leftarrow\!\!\$\, \mathbb{Z}_N$ and then compute $z^2 \mod N$, the result is an element of $\mathcal{J}_N \cap \mathrm{QR}_N$. For the case of $\mathcal{J}_N \cap \neg\, \mathrm{QR}_N$ the reduction can first compute z^2 and then output $z^2 \cdot y_N \mod N$, which is in $\mathcal{J}_N \cap \neg\, \mathrm{QR}_N$ because $y_N \notin \mathrm{QR}_N$.

We will now show how reduction $\mathcal{R}_{EQR}$ handles calls to the oracle $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(0, \mathsf{pk}, (\mathsf{psig}, \mathsf{nonce}))$. For each fresh nonce is first samples κ random bits that will represent the input bits of the recipient $m_1, \ldots, m_\kappa$ for this nonce. It then programs the random oracle H_N for call (N, nonce, j) to a random element from $\mathcal{J}_N \cap \neg\, \mathrm{QR}_N$ if $m_j = 0$ and a random element from $\mathcal{J}_N \cap \mathrm{QR}_N$ if $m_j = 1$. The reduction then uses $\mathsf{Fail}_{\mathsf{psig}}$ (from G_5) to identify if, for the given presignature psig, the expected output for those random bits is a signature or $\perp$. Note that, as already mentioned, this can be done by the reduction without knowing the corresponding recipient secret key sk_{R_0}. If the oracle call is expected to return a valid signature, the reduction computes it as in G_8.

We will now show how the reduction handles the computation of in Obtain. Given nonce_0 it samples $m_1, \ldots, m_\kappa$ at random. For $j \in \{i+1, \ldots, \kappa\}$ the reductions program the oracle for call H_N for call (N, nonce_0, j) to a random element from $\mathcal{J}_N \cap \neg\, \mathrm{QR}_N$ if $m_j = 0$ and a random element from $\mathcal{J}_N \cap \mathrm{QR}_N$ if $m_j = 1$. The calls for $j \in \{1, \ldots, i-1\}$ are sampled directly as random element of $\mathcal{J}_N$ independent of $m_1, \ldots, m_{i-1}$ due to prior changes. For the call (N, nonce_0), the reduction returns the element x_N from the EQR instance. The reduction then uses $\mathsf{Fail}_{\mathsf{psig}_0}$ set to check if psig_0 for $m_1, \ldots, m_\kappa$ aborts or not. In the former case, the reduction outputs $\perp$. Otherwise, it computes the final message $m_0 \in \mathbb{Z}_q$ and the final signature sig_0 as per the prior changes.

It is easy to see that in case $m_i = 0$ and $x_N \in \mathrm{QR}_N$ or $m_1 = 1$ and $x_N \notin QR_N$ the reduction perfectly simulates $\mathrm{G}_{9,i}$. Otherwise if $m_i = 0$ and $x_N \notin \mathrm{QR}_N$ or $m_1 = 1$ and $x_N \in QR_N$ then the reduction simulates $\mathrm{G}_{9,i+1}$. Thus, given an adversary that can distinguish between $\mathrm{G}_{9,i}$ and $\mathrm{G}_{9,i+1}$, the the reduction $\mathcal{R}_{\mathsf{EQR}}$ can solve the EQR assumption.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids $\mathrm{G}_{10,\kappa-\lambda}$ and $\mathrm{G}_{9,\kappa-\lambda}$ only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{10,\kappa-\lambda}(\mathcal{A})=1]-\Pr[\mathrm{G}_{9,\kappa-\lambda}(\mathcal{A})=1]]| \leq (\kappa-\lambda)\cdot \mathbf{Adv}_{\mathcal{R}_{\mathsf{EQR}},\mathsf{EQR}}.$$

The claim follows using the same arguments as above.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_{11} and $\mathrm{G}_{10,\kappa-\lambda}$ only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{11}(\mathcal{A})=1]-\Pr[\mathrm{G}_{10,\kappa-\lambda}(\mathcal{A})=1]]| \leq \frac{1}{2^{\kappa-\lambda-\ell_q}}.$$

The claim follows by the leftover hash lemma (LHL). There are κ bits, where the adversary learns λ bits, i.e., the guessing probability of the input distribution is $\gamma=\frac{1}{2^{\kappa-\lambda}}$. Moreover, the universal hash function's output space is $\mathbb{Z}_q$, so there are around 2^{ℓ_q} elements. By LHL, the distance between the games is $\gamma\cdot 2^{\ell_q}$. The claim follows.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrids G_{12} and G_{11} only differs by a negligible factor, i.e.:

$$|\Pr[\mathrm{G}_{12}(\mathcal{A})=1]-\Pr[\mathrm{G}_{11}(\mathcal{A})=1]]| \leq \frac{1}{2^{\kappa-\lambda-\ell_q}}.$$

The claim follows using the same arguments as above.

Claim. We claim that the adversary's $\mathcal{A}$ advantage in hybrid G_{12} is $1/2$, i.e.:

$$\Pr[\mathrm{G}_{12}(\mathcal{A})]=1/2.$$

The claim follows since signatures sig_0 and sig_1 are fresh signatures under random messages independent of the bit b. The only thing adversary $\mathcal{A}$ can do is guess bit b.

Combining everything together, we receive:

$$\Pr[\mathrm{G}_0(\mathcal{A})] \leq \frac{Q_{\mathsf{AE}}^2}{2^\lambda}+\frac{Q_q^2}{2^{\ell_q}}+\frac{Q_N^2}{2^{|\mathcal{J}_N|}}+\frac{Q_R^2}{2^\lambda}+2\cdot(\kappa-\lambda)\cdot\mathbf{Adv}_{\mathcal{R}_{\mathsf{EQR}},\mathsf{EQR}}+\frac{2}{2^{\kappa-\lambda-\ell_q}}+1/2$$

Theorem 16. *In the random oracle model,* NIBPS *is nonce blind (Definition 12) assuming the extended quadratic residuosity problem (Definition 5).*

Proof. We will prove this theorem by a series of hybrid arguments. Let $\mathcal{A}$ be a PPT-adversary against the recipient blindness of NIBS. We assume without loss of generality that the adversary outputs pk for which $e(V_1,g_2)=e(\mathsf{H}_{\mathbb{G}_1}(X),X)$ and $e(V_2,g_2)=e(\mathsf{H}_{\mathbb{G}_1}(Y),Y)$. Note that otherwise, the recipient always aborts and the only strategy for the adversary is to guess the bit b and the proof holds. We also assume that $\mathcal{A}$ makes at most $Q_{\mathsf{AE}},Q_q,Q_N,Q_R$ queries to the respective random oracles and Q_{NIBS} queries to the $\mathcal{O}_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ oracle.

Game $\mathbf{G}_0$: The original nonce blindness experiment $\mathrm{G}_0 := \mathbf{Adv}^{\mathsf{NBnd}}_{\mathcal{A},\mathsf{NIBS}}$.

Game $\mathbf{G}_1$: We abort the experiment if there are collisions in the random oracles H_{AE}, H_q, H_N or H_R.

Game $\mathbf{G}_{2,0}$: We slightly modify the experiment from G_1. In this experiment, for the presignature-nonce pair $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the challenger decrypts the ciphertexts ct_i^0 and ct_i^1 obtaining all values s_i^0 and s_i^1, where the challenger finds the key by looking through the list of outputs of the random oracle H_{AE}. Note that AE will only accept the correct key. Additionally, the challenger keeps a set $\mathsf{Fail}_{\mathsf{psig}_0}$ with tuples (i, b), meaning that for $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the ciphertext ct_i^b cannot be decrypted, i.e., there exists no key for it in the random oracle H_{AE}. For pairs $(i, b) \notin \mathsf{Fail}_{\mathsf{psig}_0}$, the challenger learn the key k_i^b. For those values, the challenger additionally checks the validity of the ciphertexts n_i^0 and n_i^1, i.e., that $n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0))$ and $n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i), \mathsf{H}_R(i, k_i^1))$. In case for some pair $(i, b) \notin \mathsf{Fail}_{\mathsf{psig}_0}$, any of the ciphertext is wrong, the challenger adds (i, b) to $\mathsf{Fail}_{\mathsf{psig}_0}$.

Game $\mathbf{G}_{2,1}$: Similar to $\mathrm{G}_{2,0}$ but for $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_3$: Similar to $\mathrm{G}_{2,1}$ but for all Q_{NIBS} queries to the oracle Q_{NIBS}.

Game $\mathbf{G}_{4,0}$: We now slightly modify G_3. In this experiment, for the presignature-nonce pair $(\mathsf{psig}_0, \mathsf{nonce}_0)$ the challenger checks the shares s_i^0 and s_i^1. To do so, for each $i \in [\kappa]$ the challenger picks random bits $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_\kappa \leftarrow\!\!\$\ \{0,1\}$, where we ensure that for each of them we have $(j, m_j) \notin \mathsf{Fail}_{\mathsf{psig}_0}$. The challenger then sets $m_i = 0$ and computes message $m = (\alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2)$, where $l_1 = m_1 \ldots m_{\ell_q}$, $l_2 = m_{\ell_q+1} \ldots m_\kappa$. It then uses the shares s_i^0, s_i^1 to compute the signature sig under m (as in Obtain). The challenger repeats the process λ times for different random $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_\kappa \leftarrow\!\!\$\ \{0,1\}$. It adds $(i, 0)$ to $\mathsf{Fail}_{\mathsf{psig}_0}$ if any generated signatures fail verification. It then repeats the process for $m_i = 1$ and all other positions $i \in [\kappa]$, making sure only to pick the values not already in the set $\mathsf{Fail}_{\mathsf{psig}_0}$. This way, the challenger checks which shares s_i^0 and s_i^1 are valid and can be used to obtain valid signatures.

With this change, the challenger can check if for psig_0 and given input bits $m_1, \ldots, m_\kappa$ the Obtain algorithm should abort even without knowing the secret key sk_R. To do so, the challenger can check for all $i \in [\kappa]$ the pair $(i, m_i) \notin \mathsf{Fail}_{\mathsf{psig}_0}$. If the check fails for at least one i, then psig_0 fails to produce a valid signature for input bits $m_1, \ldots, m_\kappa$ of the recipient, and the challenger knows that in such a case, it needs to return $\perp$ (as mentioned this can be done without the challenger knowing the recipient secret key sk_R).

Game $\mathbf{G}_{4,1}$: Similar to $\mathrm{G}_{4,0}$ but for $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_5$: Similar to $\mathrm{G}_{4,1}$ but for all Q_{NIBS} queries to the oracle Q_{NIBS}.

Game $\mathbf{G}_6$: Similar to the previous experiment, but we abort if the size of any

of the sets $|\{(i,b) : (i,b) \in \mathsf{Fail}_{\mathsf{psig}_0}\}|$ or $|\{(i,b) : (i,b) \in \mathsf{Fail}_{\mathsf{psig}_1}\}|$ is bigger than λ.

Game $\mathbf{G}_7$: We slightly modify the experiment from G_6. To answer queries to the $\mathsf{H}_{\mathbb{G}_1}(x)$ oracle, the challenger samples a random $r \leftarrow\!\$\, \mathbb{Z}_q$ and outputs g_1^r. It then stores (x,r) on list $L_{\mathsf{H}_{\mathbb{G}_1}}$. Given the public key $\mathsf{pk} = (X, Y, V_1, V_2)$ of the adversary the challenger computes $X_1 = V_1^{r_1^{-1}}$ and $Y_1 = V_2^{r_2^{-1}}$, where $(X, r_1) \in L_{\mathsf{H}_{\mathbb{G}_1}}$ and $(Y, r_2) \in L_{\mathsf{H}_{\mathbb{G}_1}}$. Note that we assume that $e(V_1, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(X), X)$ and $e(V_2, g_2) = e(\mathsf{H}_{\mathbb{G}_1}(Y), Y)$, so the elements (X, r_1) and (Y, r_2) exist in $L_{\mathsf{H}_{\mathbb{G}_1}}$ and we have $X_1 = g_1^x$ and $Y_1 = g_1^y$.

Game $\mathbf{G}_8$: We modify experiment G_7 in the way the challenger computes the signatures sig_0 and sig_1 in case $\mathsf{sig}_0 \neq \perp$ and $\mathsf{sig}_1 \neq \perp$. Before providing $((m_b, \mathsf{sig}_b), (m_{1-b}, \mathsf{sig}_{1-b}))$ the challenger uses X_1 and Y_1 to compute fresh signatures as:

$$\mathsf{sig}_0 = (g_1^{h_0}, (X_1)^{h_0} \cdot (Y_1)^{h_0 \cdot m_0}) \qquad \mathsf{sig}_1 = (g_1^{h_1}, (X_1)^{h_1} \cdot (Y_1)^{h_1 \cdot m_1})$$

for $h_0, h_1 \leftarrow\!\$\, \mathbb{Z}_q$.

Game $\mathbf{G}_{9,i}$: We now define subgames $i \in [\kappa - \lambda]$, and in $\mathrm{G}_{9,i}$, we change how the challenger executes the Obtain algorithm. For each j, where $(j, \cdot) \notin \mathsf{Fail}_{\mathsf{psig}_0}$ and not already changed, we replace parts of the Obtain algorithm run by challenger for $(\mathsf{psig}_0, \mathsf{nonce}_0)$. Instead of setting s_j to either s_j^0 or s_j^1 (note that the challenger knows both values due to prior changes) based on if $x_j = \mathsf{H}_N(N, \mathsf{nonce}_0, j)$ is a quadratic residue modulo N or not, we sample a uniformly random bit b_j and set $s_j = s_j^{b_j}$. Due to this change, there are $[\kappa - \lambda]$ input bits that the challenger picks independently of the recipient's public key.

Game $\mathbf{G}_{10,i}$: We make similar changes as above but for message $(\mathsf{psig}_1, \mathsf{nonce}_1)$.

Game $\mathbf{G}_{11}$: We sample the message m_0 uniformly from $\mathbb{Z}_q$.

Game $\mathbf{G}_{12}$: We sample the message m_1 uniformly from $\mathbb{Z}_q$.

Claim. We claim that

$$\Pr[\mathrm{G}_{12}(\mathcal{A})] = 1/2.$$

and

$$\Pr[\mathrm{G}_0(\mathcal{A})] \leq \frac{Q_{\mathsf{AE}}^2}{2^\lambda} + \frac{Q_q^2}{2^{\ell_q}} + \frac{Q_N^2}{2^{|\mathcal{J}_N|}} + \frac{Q_R^2}{2^\lambda} + 2 \cdot (\kappa - \lambda) \cdot \mathbf{Adv}^{\mathcal{R}_{\mathsf{EQR}}, \mathsf{EQR}} + \frac{2}{2^{\kappa - \lambda - \ell_q}} + 1/2.$$

The claim follows using the same arguments as in the proof of recipient blindness.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be group of prime order q with an efficient bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1, g_2 be the generators of respective $\mathbb{G}_1$ and $\mathbb{G}_2$. Let AE define an authenticated encryption scheme, where H_{AE} is a random oracle to the keyspace of AE. Finally, ℓ_q be the bit size of q, let $\kappa = 2 \cdot \ell_q$ be a parameter, let $\mathsf{H}_q, \mathsf{H}_{\mathbb{G}_1}, \mathsf{H}_N, \mathsf{H}_R$ be random oracles hashing to respectively $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathcal{J}_N$, $\{0,1\}^\lambda$.

$\underline{\mathsf{KeyGen}(\lambda)}$**:** pick $x, y \leftarrow\!\!\$\ \mathbb{Z}_q$ and return $\mathsf{pk} = (X = g_2^x, Y = g_2^y, V_1 = (\mathsf{H}_{\mathbb{G}_1}(X))^x, V_2 = (\mathsf{H}_{\mathbb{G}_1}(Y))^y)$ and $\mathsf{sk} = (x, y)$.

$\underline{\mathsf{RKeyGen}(\lambda)}$**:** pick prime numbers p and q, compute $\mathsf{pk}_R = N = pq$ and set $\mathsf{sk} = (p, q)$.

$\underline{\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})}$**:** pick random keys $k_1^0, k_1^1, \ldots, k_\kappa^0, k_\kappa^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and elements $o_1, \ldots, o_\lambda \leftarrow\!\!\$\ \mathbb{Z}_N$. For each $i \in [\kappa]$ compute the AE keys $K_i^0 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^0)$ and $K_i^1 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^1)$. Compute $\alpha_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 0)$, $\beta_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 1)$ and $\alpha_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 2)$, $\beta_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 3)$. Generate a multiplicative secret sharing of $1_{\mathbb{G}_1}$ by first picking $a_1, \ldots, a_\kappa \leftarrow\!\!\$\ \mathbb{G}_1$ and then computing $a_0 = 1_{\mathbb{G}_1} \cdot (\prod_{i=1}^{\kappa} a_i)^{-1}$. Pick random element $h \leftarrow\!\!\$\ \mathbb{G}_1$ and for $i \in [\kappa]$ compute:

$$s_i^0 = a_i \qquad\qquad s_i^1 = (h^{y \cdot \alpha_j})^{2^{i-1}} \cdot a_i$$

where $j = 1$ if $i \leq \ell_q$ and $j = 2$ for $i > \ell_q$. Pick a random bit t and for each $i \in [\kappa]$ compute the ciphertexts:

$$\mathsf{ct}_i^t \leftarrow \mathsf{AE.Enc}(K_i^0, s_i^0) \qquad\qquad \mathsf{ct}_i^{1-t} \leftarrow \mathsf{AE.Enc}(K_i^1, s_i^1).$$

For each $i \in [\kappa]$ compute $x_i = \mathsf{H}_N(N, \mathsf{nonce}, i)$ and ciphertexts:

$$n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0)) \qquad n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i); \mathsf{H}_R(i, k_i^1)).$$

Compute $s_0 = a_0 \cdot h^x \cdot (h^y)^{\beta_1 + \beta_2}$. Set presignature as

$$\mathsf{psig} = (\{o_i^N\}_{i \in [\lambda]}, \{(\mathsf{ct}_i^0, \mathsf{ct}_i^1, n_i^0, n_i^1)\}_{i \in [\kappa]}, h, s_0).$$

$\underline{\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})}$**:** compute a_i from $a_i^N \mod N$ using $1/N \mod \phi(N)$ and return $\perp$ if $e(V_1, g_2) \neq e(\mathsf{H}_{\mathbb{G}_1}(X), X)$ or $e(V_2, g_2) \neq e(\mathsf{H}_{\mathbb{G}_1}(Y), Y)$.

For each $i \in [\kappa]$ compute the element $x_i = \mathsf{H}_N(N, \mathsf{nonce}, i)$, set $m_i = 1$ if x_i is a quadratic residue and set $m_i = 0$, otherwise.
If $m_i = 1$ compute key $k_i^1 \leftarrow \mathsf{Cocks.Dec}(n_i^1, (N, x_i), \mathsf{sk}_R)$ and check that $n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i); \mathsf{H}_R(i, k_i^1))$. Otherwise, compute $k_i^0 \leftarrow \mathsf{GM.Dec}(n_i^1, (N, x_i), \mathsf{sk}_R)$ and check that $n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0))$.

For each $i \in [\kappa]$ compute the corresping AE key $K_i = \mathsf{H}_{\mathsf{AE}}(a_1, \ldots, a_\lambda, k_i^{m_i})$. Use key K_i to decrypt ct_i^0 and ct_i^1 for all $i \in [\kappa]$. For each i this will result with one plaintext s_i and one $\perp$. In case for both decryption fail for i with $\perp$, then abort.

Set $\sigma_1 = h$ and $\sigma_2 = s_0 \cdot \prod_{i=1}^{\kappa} s_i$. Pick a random $r \leftarrow\!\!\$\ \mathbb{Z}_q$ and set the final signature as $\mathsf{sig} = (\sigma_1^r, \sigma_2^r)$. Set $l_1 = m_1 \ldots m_{\ell_q}$, $l_2 = m_{\ell_q + 1} \ldots m_\kappa$ and compute $m = (\alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2) \mod q$. Return $\perp$ if $\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig})) = 0$. Return (m, sig).

$\underline{\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig}))}$**:** parse sig as (σ_1, σ_2) and output 1 if $e(\sigma_1, X \cdot Y^m) = e(\sigma_2, g_2)$ and $m \neq 0$.

Scheme 8: Fully-Secure NIBS Construction Based on PS Signatures (NIBPS)

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be group of prime order q with an efficient bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Let g_1, g_2 be the generators of respective $\mathbb{G}_1$ and $\mathbb{G}_2$. Let AE define an authenticated encryption scheme, where H_{AE} is a random oracle to the keyspace of AE. Finally, ℓ_q be the bit size of q, let $\kappa = 2 \cdot \ell_q$ be a parameter, let $\mathsf{H}_q, \mathsf{H}_{\mathbb{G}_1}, \mathsf{H}_N, \mathsf{H}_R$ be random oracles hashing to respectively $\mathbb{Z}_q$, $\mathbb{G}_1$, $\mathcal{J}_N$, $\{0,1\}^\lambda$.

$\underline{\mathsf{KeyGen}(\lambda)}$**:** pick $x, y \leftarrow\!\!\$\ \mathbb{Z}_q$ and return $\mathsf{pk} = (X = g_2^x, Y = g_2^y, V_1 = (\mathsf{H}_{\mathbb{G}_1}(X))^x, V_2 = (\mathsf{H}_{\mathbb{G}_1}(Y))^y)$ and $\mathsf{sk} = (x, y)$.

$\underline{\mathsf{RKeyGen}(\lambda)}$**:** pick prime numbers p and q, compute $\mathsf{pk}_R = N = pq$ and set $\mathsf{sk} = (p, q)$.

$\underline{\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})}$**:** pick random keys $k_1^0, k_1^1, \ldots, k_\kappa^0, k_\kappa^1 \leftarrow\!\!\$\ \{0,1\}^\lambda$ and elements $o_1, \ldots, o_\lambda \leftarrow\!\!\$\ \mathbb{Z}_N$. For each $i \in [\kappa]$ compute the AE keys $K_i^0 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^0)$ and $K_i^1 = \mathsf{H}_{\mathsf{AE}}(o_1, \ldots, o_\lambda, k_i^1)$. Compute $\alpha_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 0)$, $\beta_1 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 1)$ and $\alpha_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 2)$, $\beta_2 = \mathsf{H}_q(\mathsf{pk}_R, \mathsf{nonce}, 3)$. Generate a multiplicative secret sharing of $1_{\mathbb{G}_1}$ by first picking $a_1, \ldots, a_\kappa \leftarrow\!\!\$\ \mathbb{G}_1$ and then computing $a_0 = 1_{\mathbb{G}_1} \cdot (\prod_{i=1}^{\kappa} a_i)^{-1}$. Pick random element $h \leftarrow\!\!\$\ \mathbb{G}_1$ and for $i \in [\kappa]$ compute:

$$s_i^0 = a_i \qquad\qquad s_i^1 = (h^{y \cdot \alpha_j})^{2^{i-1}} \cdot a_i$$

where $j = 1$ if $i \leq \ell_q$ and $j = 2$ for $i > \ell_q$. Pick a random bit t and for each $i \in [\kappa]$ compute the ciphertexts:

$$\mathsf{ct}_i^t \leftarrow \mathsf{AE.Enc}(K_i^0, s_i^0) \qquad\qquad \mathsf{ct}_i^{1-t} \leftarrow \mathsf{AE.Enc}(K_i^1, s_i^1).$$

For each $i \in [\kappa]$ compute $x_i = \mathsf{H}_N(N, \mathsf{nonce}, i)$ and ciphertexts:

$$n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0)) \qquad n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i); \mathsf{H}_R(i, k_i^1)).$$

Compute $s_0 = a_0 \cdot h^x \cdot (h^y)^{\beta_1 + \beta_2}$. Set presignature as

$$\mathsf{psig} = (\{o_i^N\}_{i \in [\lambda]}, \{(\mathsf{ct}_i^0, \mathsf{ct}_i^1, n_i^0, n_i^1)\}_{i \in [\kappa]}, h, s_0).$$

$\underline{\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{pk}, \mathsf{psig}, \mathsf{nonce})}$**:** compute a_i from $a_i^N \mod N$ using $1/N \mod \phi(N)$ and return $\perp$ if $e(V_1, g_2) \neq e(\mathsf{H}_{\mathbb{G}_1}(X), X)$ or $e(V_2, g_2) \neq e(\mathsf{H}_{\mathbb{G}_1}(Y), Y)$.

For each $i \in [\kappa]$ compute the element $x_i = \mathsf{H}_N(N, \mathsf{nonce}, i)$, set $m_i = 1$ if x_i is a quadratic residue and set $m_i = 0$, otherwise.
If $m_i = 1$ compute key $k_i^1 \leftarrow \mathsf{Cocks.Dec}(n_i^1, (N, x_i), \mathsf{sk}_R)$ and check that $n_i^1 \leftarrow \mathsf{Cocks.Enc}(k_i^1, (N, x_i); \mathsf{H}_R(i, k_i^1))$. Otherwise, compute $k_i^0 \leftarrow \mathsf{GM.Dec}(n_i^1, (N, x_i), \mathsf{sk}_R)$ and check that $n_i^0 \leftarrow \mathsf{GM.Enc}(k_i^0, (N, x_i); \mathsf{H}_R(i, k_i^0))$.

For each $i \in [\kappa]$ compute the corresping AE key $K_i = \mathsf{H}_{\mathsf{AE}}(a_1, \ldots, a_\lambda, k_i^{m_i})$. Use key K_i to decrypt ct_i^0 and ct_i^1 for all $i \in [\kappa]$. For each i this will result with one plaintext s_i and one $\perp$. In case for both decryption fail for i with $\perp$, then abort.

Set $\sigma_1 = h$ and $\sigma_2 = s_0 \cdot \prod_{i=1}^{\kappa} s_i$. Pick a random $r \leftarrow\!\!\$\ \mathbb{Z}_q$ and set the final signature as $\mathsf{sig} = (\sigma_1^r, \sigma_2^r)$. Set $l_1 = m_1 \ldots m_{\ell_q}$, $l_2 = m_{\ell_q + 1} \ldots m_\kappa$ and compute $m = (\alpha_1 \cdot l_1 + \beta_1 + \alpha_2 \cdot l_2 + \beta_2) \mod q$. Return $\perp$ if $\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig})) = 0$. Return (m, sig).

$\underline{\mathsf{Verify}(\mathsf{pk}, (m, \mathsf{sig}))}$**:** parse sig as (σ_1, σ_2) and output 1 if $e(\sigma_1, X \cdot Y^m) = e(\sigma_2, g_2)$ and $m \neq 0$.

Scheme 9: Fully-Secure NIBS Construction Based on PS Signatures (NIBPS)

8 Conclusions

In this paper, we solved an interesting open problem of airdropping (i.e., distributing in a non-interactive way) one-time tokens that was not fully solved by prior work. We proposed a generic construction of non-interactive blind signatures based on Yao's garbled circuits and a secure concrete scheme in the malicious setting. One of the critical components we introduced is a non-interactive oblivious transfer protocol that works with existing PKI RSA keys and does not require users to generate keys designated for this primitive. Moreover, we showed how to garble PS signatures, RSA-based CL signatures, and even BBS signatures without naively garbling entire circuits, including modular arithmetic. An interesting problem we leave open is the formalization of our garbling techniques, i.e., identifying some fundamental properties that the above signatures have and which made our technique work. Such a formalization would make finding more examples of signature schemes (e.g., post-quantum secure ones) that could be efficiently garbled using our or similar techniques easier. Another open problem is making the garbling and NIOT more concise and efficient. All proposed contributions can find applications outside of this paper's non-interactive blind scheme scenario.

Designated Verifier. A simple modification of our NIBS scheme based on Yao's circuit allows us also to propose solutions to this scenario. In particular, instead of garbling the function $\mathsf{Sign}(\mathsf{sk}, \bar{\mathsf{sk}}_{R,\mathsf{nonce}}))$ we replace the garbled function to one depending on a pseudo-random function instead $\mathsf{PRF}(\mathsf{sk}, \bar{\mathsf{sk}}_{R,\mathsf{nonce}}))$. We can implement the pseudo-random function using the AES block cipher, leading to a designated verifier NIBS. As shown in prior work, existing techniques allow the garbling of such circuits to be very efficient [26].

Acknowledgements

Lucjan Hanzlik and Eugenio Paracucchi were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 545066907. Riccardo Zanotto is funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Baldimtsi, F., Cheng, J., Goyal, R., Yadav, A.: Non-interactive blind signatures: Post-quantum and stronger security. In: Chung, K.M., Sasaki, Y. (eds.) Advances in Cryptology – ASIACRYPT 2024, Part II. Lecture Notes in Computer Science, vol. 15485, pp. 70–104. Springer, Singapore (Dec 2024). https://doi.org/10.1007/978-981-96-0888-1_3

2. Baldimtsi, F., Cheng, J., Goyal, R., Yadav, A.: Non-interactive blind signatures: Post-quantum and stronger security. Cryptology ePrint Archive, Report 2024/614 (2024), https://eprint.iacr.org/2024/614
3. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012: 19th Conference on Computer and Communications Security. pp. 784–796. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382279
4. Bellare, M., Micali, S.: Non-interactive oblivious transfer and applications. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 547–557. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_48
5. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology – ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer, Berlin, Heidelberg (Dec 2000). https://doi.org/10.1007/3-540-44448-3_41
6. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) Advances in Cryptology – EUROCRYPT'96. Lecture Notes in Computer Science, vol. 1070, pp. 399–416. Springer, Berlin, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_34
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_3
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) Advances in Cryptology – ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 514–532. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
9. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02: 3rd International Conference on Security in Communication Networks. Lecture Notes in Computer Science, vol. 2576, pp. 268–289. Springer, Berlin, Heidelberg (Sep 2003). https://doi.org/10.1007/3-540-36413-7_20
10. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 56–72. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_4
11. Celi, S., Davidson, A., Valdez, S., Wood, C.A.: Privacy Pass Issuance Protocol. Internet-Draft draft-ietf-privacypass-protocol-16, Internet Engineering Task Force (Oct 2023), https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/16/, work in Progress
12. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – CRYPTO'82. pp. 199–203. Plenum Press, New York, USA (1982). https://doi.org/10.1007/978-1-4757-0602-4_18
13. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 2260, pp. 360–363. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45325-3_32

14. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding. pp. 360–363. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
15. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: Bypassing internet challenges anonymously. Proceedings on Privacy Enhancing Technologies **2018**(3), 164–180 (Jul 2018). https://doi.org/10.1515/popets-2018-0026
16. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – CRYPTO'82. pp. 205–210. Plenum Press, New York, USA (1982). https://doi.org/10.1007/978-1-4757-0602-4_19
17. Gennaro, R., Micciancio, D., Rabin, T.: An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In: Gong, L., Reiter, M.K. (eds.) ACM CCS 98: 5th Conference on Computer and Communications Security. pp. 67–72. ACM Press (Nov 1998). https://doi.org/10.1145/288090.288108
18. Goldberg, S., Reyzin, L., Sagga, O., Baldimtsi, F.: Efficient noninteractive certification of RSA moduli and beyond. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 700–727. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34618-8_24
19. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984), https://www.sciencedirect.com/science/article/pii/0022000084900709
20. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer, Berlin, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
21. Hanzlik, L.: Non-interactive blind signatures for random messages. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part V. Lecture Notes in Computer Science, vol. 14008, pp. 722–752. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_25
22. Lindell, Y., Pinkas, B.: An efficient protocol for secure two-party computation in the presence of malicious adversaries. In: Naor, M. (ed.) Advances in Cryptology – EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 52–78. Springer, Berlin, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_4
23. Lysyanskaya, A.: Security analysis of RSA-BSSA. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 13940, pp. 251–280. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31368-4_10
24. Micali, S., Rabin, M.O., Vadhan, S.P.: Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science. pp. 120–130. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814584
25. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. Lecture Notes in Computer Science, vol. 5157, pp. 554–571. Springer, Berlin, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_31

26. Pohle, E., Abidin, A., Preneel, B.: Fast evaluation of S-boxes with garbled circuits. Cryptology ePrint Archive, Report 2022/1278 (2022), https://eprint.iacr.org/2022/1278
27. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) Topics in Cryptology – CT-RSA 2016. Lecture Notes in Computer Science, vol. 9610, pp. 111–126. Springer, Cham (Feb / Mar 2016). https://doi.org/10.1007/978-3-319-29485-8_7
28. Rabin, M.O.: How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187 (2005), https://eprint.iacr.org/2005/187
29. shelat, a., Shen, C.H.: Fast two-party secure computation with minimal assumptions. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013: 20th Conference on Computer and Communications Security. pp. 523–534. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516698
30. Wahby, R.S., Boneh, D.: Fast and simple constant-time hashing to the BLS12-381 elliptic curve. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(4), 154–179 (2019). https://doi.org/10.13154/tches.v2019.i4.154-179, https://tches.iacr.org/index.php/TCHES/article/view/8348
31. Wahby, R.S., Boneh, D., Jeffrey, C., Poon, J.: An airdrop that preserves recipient privacy. In: Bonneau, J., Heninger, N. (eds.) FC 2020: 24th International Conference on Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 12059, pp. 444–463. Springer, Cham (Feb 2020). https://doi.org/10.1007/978-3-030-51280-4_24
32. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th Annual Symposium on Foundations of Computer Science. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25