Universal Ring Signatures in the Standard Model

Pedro Branco[1] Nico Döttling[2] and Stella Wohnig[2,3]

[1] Johns Hopkins University
[2] Helmholtz Center for Information Security (CISPA)
[3] Universität des Saarlandes

Abstract. Ring signatures allow a user to sign messages on behalf of an *ad hoc* set of users - a ring - while hiding her identity. The original motivation for ring signatures was whistleblowing [Rivest et al. ASIACRYPT'01]: a high government employee can anonymously leak sensitive information while certifying that it comes from a reliable source, namely by signing the leak. However, essentially all known ring signature schemes require the members of the ring to publish a structured verification key that is compatible with the scheme. This creates somewhat of a paradox since, if a user does not want to be framed for whistleblowing, they will stay clear of signature schemes that support ring signatures.
In this work, we formalize the concept of universal ring signatures (URS). A URS enables a user to issue a ring signature with respect to a ring of users, independently of the signature schemes they are using. In particular, none of the verification keys in the ring need to come from the same scheme. Thus, in principle, URS presents an effective solution for whistleblowing.
The main goal of this work is to study the feasibility of URS, especially in the standard model (i.e. no random oracles or common reference strings). We present several constructions of URS, offering different trade-offs between assumptions required, the level of security achieved, and the size of signatures:

- Our first construction is based on superpolynomial hardness assumptions of standard primitives. It achieves compact signatures. That means the size of a signature depends only logarithmically on the size of the ring and on the number of signature schemes involved.
- We then proceed to study the feasibility of constructing URS from standard polynomially-hard assumptions only. We construct a non-compact URS from witness encryption and additional standard assumptions.
- Finally, we show how to modify the non-compact construction into a compact one by relying on indistinguishability obfuscation.

Keywords: Ring signatures, whistleblowing

1 Introduction

Ring Signatures. Ring signatures, introduced in [33], allow for a user to create a signature σ for a message m with respect to an ad-hoc group of users R, called

a ring. A ring signature should be: i) unforgeable, meaning that, given a valid signature σ for a ring R, it must have been created by one of the users in R; and ii) anonymous, meaning that it should be infeasible for someone, even if they have access to every signing key corresponding to the verification keys in the ring R, to identify which user created the signature.

Ring signatures have recently found wide-spread application in the context of cryptocurrencies. However in this work we revisit the original motivation of ring signatures: *whistleblowing* [33]. Using a ring signature scheme, a whistleblower in a high government office with access to some classified information can leak this information e.g. to the media, in a way that convinces them that this information comes from a reliable source, namely by signing the leak. At the same time, the identity of the whistleblower remains hidden in the ring of insiders. A critical aspect in this scenario is that *the whistleblower can issue such a signature without the consent of the other parties in the ring.*

Rivest, Shamir and Tauman-Kalai [33] showed that signature schemes with RSA verification keys can be used to issue ring signatures. If RSA signatures were the universally agreed-upon standard for digital signatures, this would be great for whistleblowers! Yet, currently there is a plethora of competing schemes and standards for digital signatures.

Support for ring signatures might however even deter users from adopting some signature scheme: Knowing that a certain signature scheme supports ring signatures, why should loyal government officials even use such a scheme and potentially be framed for being a whistleblower? Furthermore, wouldn't it even be in the interest of a government to mandate their officials to use signature schemes which do not allow to issue ring signatures? Can the kind of whistleblowing envisioned by [33] be prohibited by such measures? Are there effective countermeasures which protect users against being abused as a crowd in which a whistleblower seeks anonymity? Concretely, can we construct signature schemes which protect their users from being involuntarily forced into a ring?

Universal Ring Signatures. Formalizing the idea of a ring signature compatible with *all* digital signature schemes, we define the notion of *Universal Ring Signatures* (URS)[4]. URS allow users to create a ring signature for a ring composed of verification keys $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *independently* of the structure of each vk_i and even the signature schemes which were used to create these keys. In other words, each vk_i can be a verification key from a *(possibly different) signature scheme.*[5] Most importantly, *none of the verification keys is required to be compatible with known ring signature schemes.*

Thus, URS allow users to conceal their identity inside a ring in a *non-cooperative way*: The user can create a signature with respect to a ring of verification keys, even if they were specifically chosen to be incompatible with specific

[4] The term universal ring signatures was also used in [36] to refer to a completely different property of ring signatures.

[5] For example, one of the verification keys can be from an SIS-based signature scheme and another from a group-based signature scheme.

ring signature schemes. This is in stark contrast to standard ring signatures, where the parties *cooperate* by issuing verification keys that are compatible with a ring signature scheme, thus intentionally providing anonymity to one another (which is what happens in a cryptocurrency setting).

A URS provides a way out of the whistleblower problem described above. Equipped with a URS scheme, a whistleblower just needs to somehow specify (implicitly or explicitly) the verification keys of the users in the ring. However, unlike for all known ring signature schemes, these verification keys do not need to obey any particular structure.

Ring Signatures via Non-Interactive Zero-Knowledge Proofs. Non-interactive zero-knowledge (NIZK) proofs [8] are a powerful and quite general tool to make protocols secure against malicious adversaries. In the context of ring signatures, the slightly stronger notion of non-interactive zero-knowledge proofs of knowledge (NIZKPoK) provide a stronger soundness guarantee, in the sense that any (efficient) prover providing a valid proof of some statement x must *know* corresponding witness w of x.

NIZKPoK proofs provide a direct approach to construct ring signatures: For a ring R, a message m and a commit c one provides a proof π which certifies that c commits to a signature σ such that the pair (σ, m) verifies under some verification key vk in the ring R.

This construction does not require that the verification keys in the ring R come from one and the same signature scheme. Thus, NIZKPoK proofs in fact imply universal ring signatures. Yet, NIZK (and thus also NIZKPoK) are known to be impossible in the standard model [23], that is without a common reference string and without making use of the random oracle heuristic [4]. We will later discuss the ramifications of relying on either the random oracle model or the random oracle heuristic in the construction of URS.

1.1 Our Results

The main problem we address in this work is the question of whether universal ring signatures exist in the standard model, and if so under which assumptions.

Before we tackle the problem of constructing universal ring signatures, we first provide definitions that formalize the requirements informally laid out above.

We present three standard model URS construction, offering different tradeoffs between compactness, security and primitives/assumptions needed to construct them. Our schemes are *fully* universal, in the sense that no assumptions on the structure of verification keys are made.

Our first construction is a URS scheme with compact signatures, i.e., the signature size depends only logarithmically on the number of users in the ring and on the number of signature schemes. This scheme relies on superpolynomial hardness of standard assumptions. Specifically, we rely on a superpolynomially secure signature scheme, a (polynomially secure) perfectly binding commitment scheme,

perfectly sound non-interactive witness-indistinguishability (NIWI) proof systems for NP and somewhere perfectly binding (SPB) hashing scheme [3]. All of these primitives can be instantiated using standard hardness assumptions.

We get the following theorem.

Theorem 1 (Informal). *Assuming the existence of perfectly binding commitment schemes, perfectly sound NIWI proof systems for* NP *and SPB hashing schemes (all three with polynomial security), there exists a universal ring signature scheme in the standard model with compact signatures under the condition that the underlying signature schemes are superpolynomially secure.*

While this construction provides the baseline for our investigation, it raises the question whether superpolynomial hardness is necessary to construct standard model universal ring signatures. Compared with 2-move blind signatures, we do know standard model constructions (again, no CRS or RO) from superpolynomial hardness assumptions [20,19], yet we don't know of any such construction from polynomial hardness assumptions and in fact, it is known that no such construction is achievable via a black-box reduction [16]. Thus, it is conceivable that something similar might be the case for universal ring signatures.

Perhaps somewhat surprisingly, our second construction shows that this is not the case for URS: We provide a construction that enjoys a security reduction to polynomial and falsifiable hardness assumptions. Concretely, we rely on the existence of a witness encryption (WE) scheme for NP, a perfectly sound NIWI proof system for NP, an SPB hashing scheme, and a pseudorandom function (PRF). In terms of compactness, the size of the signatures of this scheme depends linearly on the number of users in the ring. Further, this scheme fulfills a slightly relaxed notion of anonymity, which we call t-anonymity, which requires that there need to be at least t honestly generated verification keys in the ring. The standard notion of anonymity corresponds to 2-anonymity.

Theorem 2 (Informal). *Assuming the existence of a WE for* NP*, a perfectly sound NIWI proof system for* NP*, an SPB hashing scheme, and a PRF, there exists a (non-compact) universal ring signature scheme in the standard model with* t*-anonymity, where* t *is a parameter depending on the signature schemes involved.*

For all conceivable purposes, the parameter t here is a small constant. Concretely, t depends on the entropy κ of the honest verification keys involved. Asymptotically, any such key must have entropy at least $\kappa = \omega(\log(\lambda))$. Otherwise, it would be trivially insecure. Our only requirement on t will be that $t \cdot \kappa \geq \lambda$. In terms of concrete parameters, κ would have to be at least 50 bits (or else the underlying scheme would be trivially insecure). Setting $t = 3$ or $t = 4$ will be sufficient for this parameter choice.

This leaves open the question of compactness. Is perhaps any standard model URS necessarily non-compact?

We can also resolve this question negatively, yet under still a (potentially) stronger assumption: We provide a construction of a compact WE scheme from

polynomial hardness assumptions for a special type of languages that we call (t, N) threshold conjunction languages, which together with Theorem 2 will imply a compact URS scheme from polynomial hardness assumptions.

A (t, N) threshold conjunction language is the set of statements $(x_1, \ldots, x_N)$ for which there are at least t valid statements x_i among them. The size of the ciphertexts we receive when encrypting under such a statement is compact in the sense that it only depends logarithmically on N. Our WE construction requires indistinguishability obfuscation ($i\mathcal{O}$), puncturable pseudorandom functions (PPRF) [10], somewhere statistically binding (SSB) hashing schemes [27,30] and (t, N)-linear secret sharing (LSS). We obtain the following theorem.

Theorem 3 (Informal). *Assuming the existence of an $i\mathcal{O}$ for all circuits, a (non-compact) WE for NP, a PPRF, an SSB hashing scheme, and a (t, N)-LSS, there exists a compact WE scheme for (t, N) threshold conjunction languages, when $N - t \in \mathcal{O}(\log N)$.*

Combining the two previous theorems, we obtain our final URS construction. This URS construction achieves compact signatures.

Theorem 4 (Informal). *Assuming the existence of a compact WE for $(N - 1, N)$ threshold conjunction languages, a perfectly sound NIWI proof system for NP, an SPB hashing scheme and a PRF, there exists a compact universal ring signature scheme in the standard model with t-anonymity.*

1.2 Discussion and Interpretation of our Results

Returning to our main motivation, a URS enables whistleblowing since a whistleblower can *force* any honest users into a ring, regardless of which signature scheme they use. In this sense, one can view the process of signing a message using a URS as an *adversarial act*: even if a set of honest users do not want to hide the whistleblower, there are no effective measures on the level of signature schemes which could protect users from being included in an anonymity set.

Bearing this in mind, we interpret our results, which establish the feasibility of URS, as demonstrating the impossibility of designing signature schemes that resist coercion into rings. Needless to say, the rather heavy components involved in our constructions do not lead to practically useful protocols.

Above we briefly discussed that universal ring signatures can be constructed from NIZKPoK proofs and by now there is a plethora of constructions of NIZKPoK proofs from standard assumptions in the common reference string (CRS) model [8,15,25,32,12,28], or alternatively in the random oracle model [4]. If the goal was to construct a practically useful URS to provide support across different, seemingly incompatible but *common* signature schemes, then a protocol relying on succinct NIZKPoK arguments would be preferable. In such a setting, one would expect the users of these schemes to collaborate in the sense that they are willing to provide anonymity to one another, i.e. one could assume that all users trust a common reference string as well as all the signature schemes involved.

Yet, the scenario we are interested in is different, in the sense that the "users" have no reason to trust one another, as they were potentially forced into a ring against their will. In this sense, a universal ring signature scheme in the CRS could give users who have been forced into a ring against their will a means of plausible deniability, e.g. by claiming that they do not trust the CRS that was used to generate a universal ring signature, as the party who generated such a CRS may also forge such a signature.

On the other hand, if we consider URS in the random oracle model, then the unsoundness of the ROM could cause issues. When protocols in the ROM are instantiated, we replace the random oracle with a concrete hash function H. As shown by Goldwasser and Kalai [24], this heuristic can lead to unsound proof systems if the underlying language already depends on this (concrete) hash function H.

This issue also comes up in the context of universal ring signatures, as one of the signature schemes could be chosen *depending on the hash function* H. Somewhat more concretely, assume we wanted to build a signature scheme Σ^* which makes a URS relying on a random oracle unsound, in the sense that if any verification key of Σ^* is used in a ring R, then universal ring signatures can be forged, while Σ^* is still EUF-CMA secure. We could achieve this by taking any EUF-CMA secure signature scheme Σ and modifying it to Σ^* by additionally including into the verification keys vk^* of Σ^* an obfuscated program $\mathcal{O}$ which lets anyone publicly generate URS of rings involving vk^*. Note that this obfuscated program $\mathcal{O}$ needs to *know* a succinct description of the hash function H, but this is feasible as we assume H to be *instantiated*, rather than a random oracle. The same can in fact be argued for any fixed static common reference string CRS, i.e. CRS can be hardwired into $\mathcal{O}$. Note that while for such a scheme the size of the verification key would increase, both generation and verification would remain essentially unchanged.

Looking ahead, if such a transformation from Σ to Σ^* was done starting relative to one of our standard-model secure URS, then Σ^* would be necessarily insecure. But for a URS whose unforgeability rests on the CRS model or the random oracle model, we would generally expect such a Σ^* to be unforgeable (once the CRS or the RO has been instantiated).

The bottom line of this discussion is that it seems hard to argue that URS constructions in the CRS model or the ROM would be robust against signature schemes which undermine the unforgeability of the URS by depending on the concrete CRS which is used or the concrete hash function which instantiates the Fiat-Shamir paradigm.

1.3 Previous Works

Ring signatures have been extensively studied in the last two decades. Constructions in the random oracle model (ROM) include [33,1,9,14,29]. Ring signatures in the CRS model were studied in [35,11], where [11] solves the interesting but orthogonal problem of how to include users in a ring whose public keys are

not posted publicly by using a PKI structure. We can also find standard model constructions for ring signatures in e.g. [5,3,31].

All works presented above assume some form of structure on the verification keys. For example, the work of [33] assumes that ring verification keys are RSA keys or the work of [5] assumes that ring verification keys are composed by a standard verification key and a uniformly random string.

The only exceptions we are aware of are the works [1,22]. In these works, ring signatures that support different signature schemes are presented. However, these works are only *somewhat universal* in the sense that there are signature schemes that are not compatible with their schemes.[6] Moreover, these schemes are only secure in the ROM whereas we work in the standard model. In essence, the focus of these works is different from ours as they sacrifice universality for efficiency. In this work, we take the opposite direction.

A construction of a universal primitive from $i\mathcal{O}$ has previously been given for a notion called signature aggregators in [26]. This allows to combine signatures from different users using arbitrary signature schemes into one signature to succinctly store and verify. The application and techniques used are however different and can not be transferred to ring signatures trivially.

2 Technical Overview

Before presenting our constructions of URS, we briefly recall the ring signature scheme of Backes *et al.* [3]

In the scheme of [3] (which is itself based on [5]), verification keys are composed by $\mathsf{VK}_i = (\mathsf{vk}_i, \mathsf{pk}_i)$ where vk_i is a verification key of a standard signature scheme Sig and pk_i is a public key of a public-key encryption (PKE) scheme that has pseudorandom ciphertexts[7].

To sign a message m with respect to a ring $R = \{\mathsf{VK}_i\}_{i\in[\ell]}$, the signer i first generates a signature $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$ and then encrypts σ it using pk_i, that is, $\mathsf{ct}_0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_i, \sigma)$. The signer then samples $\mathsf{ct}_1 \leftarrow\!\!\$\, \{0,1\}^\lambda$. One crucial point is that, if the underlying PKE has pseudorandom ciphertexts, then we cannot distinguish well-formed ciphertexts from uniformly random strings. In particular, this means that ct_0 contains (computationally) no information about the public key under which it was encrypted.

The signer now proves that either ct_0 or ct_1 encrypts a valid signature under one of the verification keys in the ring using a non-interactive witness-indistinguishable (NIWI) proof system. If off-the-shelf NIWIs were used in this construction, the size of the proof would scale linearly with the size of the ring. This would lead to signatures of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$, where λ is the security parameter. To circumvent this problem, [3] employed a new strategy.

[6] More precisely, the scheme of [1] is compatible with *trapdoor-one-way* and *three-move* signature schemes. The scheme of [22] is compatible with certain sigma protocols. Any scheme outside of these classes is not compatible with their ring signature schemes.

[7] Examples of such PKE schemes exist from the LWE or DDH assumption.

Compact NIWI proofs. The main ingredient to compress the size of the NIWI proof is a somewhere perfectly binding (SPB) hashing scheme. An SPB hashing scheme allows one to hash a database such that the hash perfectly binds to the database item an index i, while the hashing key hides the index i. [27,30,3].

Given $\mathsf{ct}_0, \mathsf{ct}_1$, the signer can now use a NIWI proof system together with a somewhere perfectly binding (SPB) hashing scheme to create a compact proof π that either ct_0 or ct_1 encrypts a valid signature under one of the keys in the ring. The basic idea here is that instead of proving a statement over all verification keys in the ring, it is sufficient to prove a statement about just two SPB hashes. More concretely, to compute the proof π, the signer first generates an SPB pair of hashing key/secret key $(\mathsf{hk}_j, \mathsf{shk}_j) \leftarrow \mathsf{SPB.KeyGen}(1^\lambda, i)$ that binds to position i, for $j \in \{0,1\}$. Then, it hashes R into a digest $h_j \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_j, R)$ for $j \in \{0,1\}$. Finally, the signer proves that there exists an index i such that one of the two statements is true:

1. ct_0 encrypts a valid signature under vk_i and hk_0 binds to i;
2. ct_1 encrypts a valid signature under vk_i and hk_1 binds to i.

The signature is composed by $(\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \pi)$. Thus, by the efficiency requirements of SPB, the signature has size $\mathcal{O}(\log |R| \cdot \mathsf{poly}(\lambda))$.

Finally, to verify that a signature is valid, one just needs to recompute h_j as the hash of R under hk_j, for $j \in \{0,1\}$, and check that π is a valid proof.

Security. Unforgeability and anonymity are roughly argued as follows in [3]. To argue unforgeability, the security of the scheme is reduced to the security of the underlying signature scheme. To do this, the reduction receives a verification key vk_{i^*} from the challenger, creates the remaining verification keys vk_i, for $i \neq i^*$, and also the public keys pk_i for all $i \in [\ell]$. Importantly, the public keys pk_i are created such that the reduction knows the corresponding secret keys.

Upon receiving a (ring signature) forge from the adversary, the reduction proceeds as follows:

1. Decrypt both ct_0 and ct_1, to obtain σ_0 and σ_1, respectively;
2. Check if any of σ_0, σ_1 is a valid signature under vk_{i^*}. If one of them is valid, the reduction outputs it as the forge.

By the perfect correctness of the SPB hashing and perfect soundness of the NIWI, the reduction outputs a valid forge with non-negligible probability.

To prove anonymity, one relies on the witness-indistinguishability of the NIWI and the fact that the underlying PKE has pseudorandom ciphertexts. Concretely, given two honestly generated verification keys vk_{i_0} and vk_{i_1}, build a sequence of hybrids to prove that a signature created under vk_{i_0} is indistinguishable from a signature created under vk_{i_1}. The sequence of hybrids starts by replacing ct_1 with an encryption of a valid signature under vk_{i_1}, and this change goes unnoticed since the PKE has pseudorandom ciphertexts. Next, change the index in the witness used to create the proof π from i_0 to i_1 using the witness-indistinguishability of the NIWI scheme.

2.1 Compact Universal Ring Signatures from Signatures with Superpolynomial Security

The construction of the Backes et al. scheme [3] serves as the starting point of our first construction. Observe that the ring signature verification keys of the Backes et al. scheme have a special format: each verification key VK_i is composed of a standard verification key vk_i and a public key pk_i.

The public key pk_i, which can be chosen by the unforgeability reduction, is what enables this reduction to extract a valid forge. In a URS, however, verification keys are not required to have any particular format. In particular, they are not required to include an independently chosen public key of a PKE. How can we facilitate the extraction of a forge by an unforgeability reduction in the setting of URS?

Commitments instead of Ciphertexts. Our first observation is that the ciphertexts in the scheme of Backes et al. [3] are never decrypted *in the actual scheme*. So, ciphertexts in this scheme actually serve as extractable commitments. Thus, a natural approach is to rely on commitments instead of ciphertexts in this construction. The main reason for using commitments instead of ciphertexts is that we can choose a *keyless* commitment scheme.

Using a commitment scheme, we can build a URS as follows: To sign a message m under a ring of users $R = \{\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell\}$ (where each vk_i is from a possibly different signature scheme), a signer first creates a signature $\sigma \leftarrow \mathsf{Sig}.\mathsf{Sign}_i(\mathsf{sk}_i, m)$ using its signature scheme Sig_i. Then, it commits to $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{CS}.\mathsf{Commit}(1^\lambda, \sigma)$ and to $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{CS}.\mathsf{Commit}(1^\lambda, 0)$ (where γ_b is the opening information). Using SPB and NIWI exactly as before, the signer can create a compact proof π that one of the commitments hides a valid signature under one of the keys in R.

Anonymity follows by essentially the same argument as before, where the hiding property of the underlying commitment is used instead of the ciphertext pseudorandomness of the PKE in [3].

Unforgeability from Superpolynomial Hardness. We now show how the unforgeability reduction can extract a valid forge from the adversary. Assume that the hiding property of the commitment scheme CS holds against *polynomial-time* adversaries but that CS can be extracted in *superpolynomial-time*. We can then use *complexity leveraging* to prove the unforgeability of the scheme, given that the underlying signature schemes are unforgeable against superpolynomial-time adversaries.

Concretely, given a PPT adversary $\mathcal{A}$ that breaks the unforgeability of our URS, we can construct a superpolynomial-time reduction against the unforgeability of one of the Sig_i. The reduction, after receiving a forge $\Sigma^* = (\mathsf{com}_0^*, \mathsf{com}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$ by $\mathcal{A}$, opens both com_0 and com_1 by brute force to recover σ_0^* and σ_1^* respectively. Note that, since CS can be extracted in superpolynomial time, the reduction succeeds in recovering σ_0^* and σ_1^*. Now, as before, the reduction tests if there is a $b \in \{0, 1\}$ such that $1 \leftarrow \mathsf{Sig}.\mathsf{Verify}_i(\mathsf{vk}_i, m, \sigma_b^*)$ and outputs σ_b^* if it is the case.

2.2 Non-Compact Universal Ring Signatures from Witness Encryption

Considering both the construction of Backes et al. [3] and our construction in the last paragraph, the question emerges of how one could possibly *efficiently* extract a signature, even if we cannot shoehorn an extraction trapdoor into the protocol utilizing a CRS or augmenting the verification keys. Somewhat more abstractly:

Is it possible to extract a secret from a protocol when the protocol constraints don't allow us to embed an extraction gadget into the protocol?

Extracting via Witness Encryption. Our way out of this dilemma starts with the observation that by relying on a sufficiently strong tool, namely standard witness encryption (WE) [18], we can repurpose any *sufficiently cryptographic* object as a public key. In our case, these objects will be the verification keys of the honest parties.

Recall that a WE for an NP language $\mathcal{L}$ (with relation $\mathcal{R}$) allows an encrypter to encrypt a message m with respect to a statement x. If $x \in \mathcal{L}$, then a party in possession of a witness w such that $\mathcal{R}(x, w) = 1$ can recover the encrypted m. But, if $x \notin \mathcal{L}$, then indistinguishability of encryptions holds. Currently, we have constructions of WE from indistinguishability obfuscation ($i\mathcal{O}$) [17] or multilinear maps [18], but WE is potentially a weaker assumption than either of these.

To use the security of WE, we need to craft a language $\mathcal{L}$ with distinct true and false statements, such that witnesses of true statements allow for decryption, whereas ciphertexts under false statements hide the encrypted message. Ideally, true and false statements should be indistinguishable. Our design-choice of true and false statements will be informed by the following consideration: Consider two distributions of (honest) verification keys, one where each honest vk is generated using truly random coins, and another one where each honest $\tilde{\mathsf{vk}}$ is generated using (possibly correlated) pseudorandom coins. While these distributions are clearly computationally indistinguishable, under the right circumstances we can also make them *statistically far*, meaning that one of them can serve as a distribution of true statements, while the other one will be the distribution of false statements.

More concretely, let PRG be a pseudorandom generator (PRG). We say that a verification key vk is *malformed* if it is created using random coins coming from a PRG, instead of using truly random coins. That is, for some seed s

$$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda; \mathsf{PRG}(s)).$$

Similarly, a *well-formed* key vk is created using truly random coins.

Now, consider the language $\mathcal{L}$ parameterized by ℓ different verification keys $\{\mathsf{vk}_i\}_{i\in[\ell]}$. The *yes* instances of $\mathcal{L}$ are the instances $\{\mathsf{vk}_i\}_{i\in[\ell]}$ where *all but one of the verification keys are malformed*. In other words, there exist $\{s_i\}_{i\in[\ell]\setminus\{i^*\}}$ with $i^* \in [\ell]$ such that for all $i \in [\ell] \setminus \{i^*\}$

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRG}(s_i)).$$

Looking ahead, the dichotomy between *all but one key are malformed* vs *at least two keys are well-formed* is what will allow us to prove unforgeability and anonymity respectively. In the former case, the statement under which the WE ciphertext is created is true and, thus, we will be able to decrypt it. In the latter case, the statement is false. Therefore, we can use the security of the WE scheme.

At first glance, this approach seems to work. However, there is a caveat: when the reduction wants verification keys to be well-formed, it might accidentally end up creating them malformed. As an example, consider a signature scheme Sig whose verification keys have less min-entropy than the underlying PRG. Say the key generation algorithm $\mathsf{Sig.Gen}(1^\lambda, r)$ only uses the first $\lambda/3$ bits of r whereas the PRG seed has $\lambda/2$ bits of entropy. In other words, the distributions of well-formed keys and malformed keys might not be sufficiently statistically far. Then, there is a non-negligible probability that a key chosen from the well-formed distribution is actually malformed. We could assume that the underlying signature schemes have exponential security (e.g., verification keys have λ bits of min-entropy) but this would to some degree defeat the purpose of URS.

Replacing the PRG by a PRF. The solution for this problem is to use a pseudo-random function (PRF) instead of a PRG to sample malformed keys. Instead of generating malformed keys individually, we now generate them in a correlated fashion: A set of keys $\{\mathsf{vk}_i\}$ is malformed iff a PRF key K exists such that

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRF}(K, i)).$$

Note that now, all malformed keys are correlated via the PRF key. This implies that the distribution of t malformed keys has λ bits of min-entropy because as soon as we choose the PRF key, all malformed keys are fixed. On the other hand, when sampling t well-formed keys independently, the resulting distribution will have $t\kappa$ bits of min-entropy where κ is the min-entropy of each verification key. Setting $t\kappa > \lambda$ we conclude that the distributions of well-formed and malformed keys are statistically far apart.

This fact will allow us to prove t-anonymity by *making the number of honest keys in the ring just large enough.*

Given this, we redefine the language $\mathcal{L}$ in the following way: *yes* instances of $\mathcal{L}$ are the instances $\{\mathsf{vk}_i\}_{i\in[\ell]}$ where *all but one of the verification keys are malformed*. In other words, there exists $K \in \{0,1\}^\lambda$ such that for all $i \in [\ell] \setminus \{i^*\}$

$$(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_i(1^\lambda; \mathsf{PRF}(K, i)).$$

The Scheme. Armed with a WE scheme WE for the language $\mathcal{L}$ described above, we now outline how we can construct a URS scheme.

The scheme is essentially the same as above except that we use the WE scheme for language $\mathcal{L}$ as a drop-in replacement for the commitment scheme.

To sign a message m with respect to the ring R, the signer encrypts a valid signature σ created using its own signing key. Then, it encrypts σ using WE under the statement $x = R$, that is, $\mathsf{ct}_0 \leftarrow \mathsf{WE.Enc}(1^\lambda, x, \sigma)$. Additionally, it

creates the ciphertext $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x, 0)$. Finally, the signer can again use NIWI and SPB to prove compactly that one of the ciphertexts encrypts a valid signature.

We first analyze the size of the signature. Note that, for all known WE schemes, the ciphertext size is proportional to the size of the verification circuit for the language $\mathcal{L}$. Since the statement is of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$, then the ciphertexts output by WE are of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$. This implies that the signature is of size $\mathcal{O}(|R| \cdot \mathsf{poly}(\lambda))$.

Security. We now sketch how we prove the security of the scheme. As mentioned before, we will set all but one key to be malformed in order to prove unforgeability. Whereas in the t-anonymity proof, we set none of the keys to be malformed (recall that t-anonymity requires that the challenge ring as at least t honestly generated verification keys).

To prove unforgeability, we design a reduction that sets all verification keys, but the challenge key vk_{i^*} to be malformed. That is, $\{\mathsf{vk}_i\}_{i \in [\ell] \setminus \{i^*\}}$ are malformedly created using a PRF key K. By the security of the PRF, the adversary is not able to distinguish the case where the verification keys $\{\mathsf{vk}_i\}_{i \in [\ell] \setminus \{i^*\}}$ are well-formed from the case when they are malformed.

The crucial observation now is that the reduction has a valid witness $w = K$ for the statement $x = R$ under which WE ciphertexts are encrypted. This means that, upon receiving a URS forge

$$\Sigma^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$$

by the adversary, the reduction can use w to decrypt both ct_0^* and ct_1^*. An analysis identical as for the previous scheme shows us that, if Σ^* is a valid URS signature, then there is a non-negligible probability that one of ct_0^* and ct_1^* decrypts to a valid signature σ^* under vk_{i^*}.

In the t-anonymity proof, we set *none* of the verification keys to be malformed, from which the adversary chooses t of them, say, $\mathsf{vk}_{i_0}, \ldots, \mathsf{vk}_{i_{t-1}}$. If the parameters of the PRF are chosen properly, then there is a negligible probability that $x \in \mathcal{L}$. As explained above, since all t verification keys are sampled independently, it is unlikely that $t-1$ share correlations via a PRF key K. This is because the distribution of $t-1$ honestly generated keys has much more min-entropy than $t-1$ malformed keys. Thus, there will be *at least two well formed* verification keys in the challenge ring R^* with overwhelming probability. We conclude that WE encryptions of σ are indistinguishable from WE encryptions of 0 by the security of the WE.

Given this, we can easily build a sequence of hybrids in a similar fashion as for the previous schemes. That is, given two honestly generated verification keys $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1}$ and a signature $\Sigma^* = (\mathsf{ct}_0^*, \mathsf{ct}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \pi^*)$ for a message m^* with respect to the ring R^* where $\mathsf{vk}_{i_0}, \mathsf{vk}_{i_1} \in R^*$:

1. We first replace ct_1^* by an encryption of a valid signature σ' under vk_{i_1}. By the security of the underlying WE, this change is undetected by the adversary.
2. We switch witnesses from i_0 to i_1, using the witness-indistinguishability of the NIWI scheme.

2.3 Compact Universal Ring Signatures from Indistinguishability Obfuscation

At first glance, the techniques that we employed in the previous construction seem hopeless in our ultimate goal of building a compact URS from falsifiable hardness assumptions. On the one hand, for all known WE schemes that we know of, the size of the ciphertexts grows with the size of the statement. On the other hand, if we try to reduce the size of the statement of the language $\mathcal{L}$, we immediately run into trouble.

The reason for this is that to be able to extract a valid forge, the reduction needs to set up all verification keys but the challenge one in a *special mode*.[8] If the reduction sets just a few of them in this special mode, anonymity does not hold anymore: An adversary breaking anonymity could just use the same strategy as the unforgeability reduction to extract a signature from the challenge URS signature since, in the anonymity game, all but two verification keys may be adversarially chosen.

Given this state of affairs, it seems implausible (or even impossible!) that we can achieve a compact URS scheme just from WE.

Our final contribution is to build a WE scheme for a special type of NP languages that we call *threshold conjunction languages*. A threshold conjunction language $\mathcal{L}'$ is a language of the form

$$\mathcal{L}' = \{(x_1, \ldots, x_N) : \exists (x_{i_1}, \ldots, x_{i_{N-1}}) \text{ s.t. } x_{i_1} \in \mathcal{L} \wedge \cdots \wedge x_{i_{N-1}} \in \mathcal{L}\}.$$

In other words, given an instance $x = (x_1, \ldots, x_N)$, x is a yes instance of $\mathcal{L}'$ if *all but one* of the x_i are instances of $\mathcal{L}$.

Compact URS from compact WE. Assume for now that we have a *compact* WE scheme for threshold conjunction languages. That is ciphertexts of such a scheme scale only logarithmically with N. Then, plugging this WE scheme into our construction from the previous section immediately yields a compact URS.

Compact witness encryption for threshold conjunction languages. It remains to show how we can obtain such a scheme. For simplicity, we focus on the case where we have N instances $x = (x_1, \ldots, x_N)$ and $x \in \mathcal{L}'$ iff $x_i \in \mathcal{L}$ for all $i \in [N]$. The case where all but one of the statements x_i must be true can be easily obtained by additionally using a secret sharing scheme.

The high-level idea of the construction is as follows: We build an obfuscated circuit $\bar{\mathcal{C}}$ that receives an index $i \in [N]$ and outputs non-compact WE ciphertexts $\mathsf{WE.Enc}(1^\lambda, x_i, r_i)$ for uniform $r_i \leftarrow\!\!\$\, \{0,1\}$.[9] The ciphertext of our new WE scheme for a message $m \in \{0,1\}$ is composed by $\bar{\mathcal{C}}$ and $c = m + \sum r_i$.

If one is in possession of witnesses for all statements x_i, then by the correctness of the underlying non-compact WE scheme, one can recover all r_i. On the

[8] In our case, the special mode is when keys are malformed.

[9] To make the circuit size independent of N, we use a pseudorandom function (PRF) to succinctly describe all the r_i. This PRF has to be puncturable in order to use the *puncturing technique* of [34].

other hand, if one of the statements x_{i^*} is false, then we can build a sequence of hybrids where we replace $\mathsf{WE.Enc}(1^\lambda, x_i, r_i)$ by an encryption of 0 and then replace c by a uniform value.

Although the idea seems to work at first glance, there is a critical issue: The scheme is not compact. The reason for this is that we have to hardwire all the statements in $\bar{\mathcal{C}}$, otherwise how does the circuit know under which statements it must encrypt each r_i? To circumvent this problem we use (again!) a somewhere statistically binding (SSB) hashing scheme in a similar way as [27].[10] That is, the circuit only has a hash value $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, \{x_1, \ldots, x_N\})$ hardwired. Now, when it receives (i, x_i, γ_i), it first checks if γ_i is a valid opening with respect to x_i, h. Since $\{x_1, \ldots, x_N\}$ is public, anyone can compute a valid opening

$$\gamma_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, \{x_1, \ldots, x_N\}, i)$$

for every $i \in [N]$.

Recall that the verification algorithm of an SSB hashing scheme can be implemented in size $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$. Hence, the efficiency requirements are met and the circuit is now of size $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$.

We thus obtain a WE scheme that outputs ciphertexts that depend only logarithmically on N.

How to avoid the exponential security loss of current $i\mathcal{O}$ schemes. We stress that, although the scheme presented above enjoys a polynomial reduction to the underlying cryptographic primitives, current $i\mathcal{O}$ schemes incur a security loss - compared to the underlying hardness assumptions - which is proportional to the size of the domain of the circuit being obfuscated (e.g., [2,7,6]). This implies that the construction presented above suffers from an exponential security loss when we instantiate the $i\mathcal{O}$ scheme by any known construction since the circuit being obfuscated has an exponentially-sized domain.[11]

Intending to avoid this exponential security loss, we present an alternative construction of compact WE for threshold languages where we just obfuscate a program with a polynomial-size domain. Note that, if the domain of the obfuscated program has only polynomial size, then the security reduction from $i\mathcal{O}$ to the underlying hardness assumptions loses only a polynomial factor.

As explained above, the statements cannot be hardwired in the circuit, otherwise, the size of the obfuscated circuit is not compact. To avoid this conundrum, we utilize the $i\mathcal{O}$ for Turing machines (TM) scheme of [21].

We note that, in the scheme of [21], a TM is modeled as a sequence of circuits. The input is written on a tape and the obfuscated TM accesses the input via a laconic oblivious transfer (LOT) [13]. We can consider a second tape which includes the statements $(x_1, \ldots, x_N)$ and from which the TM reads from using a LOT in a similar way as in [21]. Note that since $(x_1, \ldots, x_N)$ is public knowledge, this tape can be created by any party and does not have to be part

[10] This time we use SSB in its *statistically binding* form.

[11] Observe that the obfuscated circuit receives as input an index i, a statement x_i and an SSB proof γ_i.

of the description of the obfuscated TM. Instead, only the LOT hash needs to be hardwired in the TM. The size of the resulting obfuscated TM depends only logarithmically on the size of this tape.

Given this, to encrypt a message m, one obfuscates a TM $\mathcal{M}$ that receives an index $i \in [N]$ as input, retrieves x_i from the public tape and outputs $\mathsf{WE.Enc}(1^\lambda, x_i, r_i)$.[12] A ciphertext is composed by $\bar{\mathcal{M}}$ (which is the result of obfuscating $\mathcal{M}$) and $c = m + \sum r_i$. Decryption works exactly as before.

As mentioned before, the size of $\bar{\mathcal{M}}$ depends only logarithmically on N and, hence, the size of the ciphertext is $\mathcal{O}(\log N \cdot \mathsf{poly}(\lambda))$.

Furthermore, since the obfuscated TM $\bar{\mathcal{M}}$ has a polynomial-size domain, its security proof incurs only a polynomial security loss compared to the underlying hardness assumption.

3 Preliminaries

Throughout this work, λ denotes the security parameter and PPT stands for "probabilistic polynomial-time". A negligible function $\mathsf{negl}(n)$ in n is a function that vanishes faster than the inverse of any polynomial in n.

For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, \ldots, n\}$. If S is a (finite) set, we denote by $x \leftarrow\!\$\, S$ an element $x \in S$ sampled according to a uniform distribution. If D is a distribution over S, $x \leftarrow\!\$\, D$ denotes an element $x \in S$ sampled according to D. If $\mathcal{A}$ is an algorithm, $y \leftarrow \mathcal{A}(x)$ denotes the output y after running $\mathcal{A}$ on input x. If $\mathcal{A}$ and $\mathcal{O}$ are algorithms, $\mathcal{A}^{\mathcal{O}}$ means that $\mathcal{A}$ has oracle access to $\mathcal{O}$.

Additionally, we assume familiarity with the following notions from standard literature: Signature Schemes, Non-Interactive Witness-Indistinguishable Proof Systems, Commitment Schemes, Somewhere Statistically Binding and Somewhere Perfectly Binding Hashing Schemes, Pseudorandom Generators, Witness Encryption Schemes, Indistinguishability Obfuscation, and Puncturable Pseudorandom Functions. For completeness, a full collection of definitions of these notions and possible instantiations can be found in the full version of the paper. We also require Linear Secret Sharing with a slightly modified definition given below.

Linear Secret Sharing Linear secret sharing (LSS) is used to divide a secret into shares such that if one is in possession of an authorized set of shares, then one can reconstruct the secret. In this work, we use threshold LSS (which, for simplicity, we simply refer to as LSS).

Definition 5 (Linear Secret Sharing) *Let $t \leq N$. A (t, N)-linear secret sharing (LSS) LSS scheme is composed of the following algorithms:*

- $(s_1, \ldots, s_N) \leftarrow \mathsf{Share}(m)$ *takes as input a message m. It outputs N shares $(s_1, \ldots, s_N)$.*

[12] We remark that the underlying WE also has a domain of polynomial size hence it only looses a polynomial factor in security if it is based on $i\mathcal{O}$ [17,21].

– $m \leftarrow \mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t})$ *takes as input* t *shares* $(s_{i_1}, \ldots, s_{i_t})$*. It outputs a message* m*.*

A (t, N)*-LSS scheme, which is generated by a generating matrix in the systematic form, has the following additional algorithm:*

– $(s_{i_{z+1}}, \ldots, s_{i_N}) \leftarrow \mathsf{RemainShare}(m, s_{i_1}, \ldots, s_{i_z})$ *that takes as input a message* m *and uniformly chosen shares* $s_{i_j} \leftarrow\!\!\$\, \{0,1\}^\lambda$ *for* $j \in [z]$ *with* $z < t$*, and outputs* $N - z$ *remaining shares* $(s_{i_{z+1}}, \ldots, s_{i_N})$*.*

Definition 6 (Correctness) *A LSS scheme* LSS *is said to be correct if for all messages* m*, all subsets* $\{i_1, \ldots, i_t\} \subseteq [N]$ *and any* $z < t$ *it holds that:*

$$\Pr\left[m = \mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t}) : (s_1, \ldots, s_N) \leftarrow \mathsf{Share}(m)\right] = 1.$$

$$\text{and } \Pr\left[\begin{array}{c} m = \mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t}) : \\ s_{i_j} \leftarrow\!\!\$\, \{0,1\}^\lambda \text{ for } j \in [z] \\ (s_{i_{z+1}}, \ldots, s_{i_N}) \leftarrow \mathsf{RemainShare}(m, s_{i_1}, \ldots, s_{i_z}) \end{array}\right] = 1.$$

Definition 7 (Privacy) *We say that a* (t, N)*-LSS scheme* LSS *is private if for all subsets* $\{i_{i_1}, \ldots, i_{i_z}\} \subset [N]$ *where* $z < t$*, all pairs of messages* (m_0, m_1) *and all PPT adversaries* $\mathcal{A}$ *we have that*

$$\left|\begin{array}{l} \Pr\left[1 \leftarrow \mathcal{A}(s_{0,i_1}, \ldots, s_{0,i_z}) : (s_{0,1}, \ldots, s_{0,N}) \leftarrow \mathsf{Share}(m_0)\right] - \\ \Pr\left[1 \leftarrow \mathcal{A}(s_{1,i_1}, \ldots, s_{1,i_z}) : (s_{1,1}, \ldots, s_{1,N}) \leftarrow \mathsf{Share}(m_1)\right] \end{array}\right| \leq \mathsf{negl}(\lambda).$$

4 Universal Ring Signatures

In this section we present the definition of URS. A URS is composed of a signing and a verification algorithm.

Definition 8 (Universal Ring Signature) *A universal ring signature (URS) scheme* URS *is composed of the following algorithms:*

– $\Sigma \leftarrow \mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R, i, S)$ *takes as input a security parameter* 1^λ*, a signing key* sk_i*, a message* m*, a ring of keys* $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *an index* $i \in [\ell]$ *and a list of signature schemes* $S = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i \in [M]}$*, where each* vk_j *is a public verification key under exactly one*[13] *of the schemes* Sig_i*. It outputs a signature* Σ*.*
– $b \leftarrow \mathsf{Verify}(\Sigma, m, R, S)$ *takes as input a signature* σ*, a message* m*, a ring of keys* R *and a list of signature schemes* S*. It outputs a bit* $b \in \{0,1\}$*.*

We want a URS to fulfill correctness, unforgeability and anonymity.

[13] In practice, keys/certificates are usually annoted with their respective schemes and we assume such a labelling here.

Definition 9 (Correctness) *We say that a URS* $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ *is* correct *if for all* $\lambda \in \mathbb{N}$*, all* $\ell, M = \mathsf{poly}(\lambda)$*, all correct signature schemes* Sig'*, all* $j \in [\ell]$*, all messages* m *and all* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'.\mathsf{KeyGen}(1^\lambda)$*, we have that*

$$\Pr\left[1 \leftarrow \mathsf{Verify}\left(\mathsf{Sign}(1^\lambda, \mathsf{sk}, m, R, j, S), m, R, S\right)\right] = 1$$

for any $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *such that* $\mathsf{vk}_j = \mathsf{vk}$ *and any* $S = \{\mathsf{Sig}_i\}_{i\in[M]}$ *such that* $\mathsf{Sig}' \in S$*. That is, the remaining elements in* R, S *may be arbitrarily chosen.*

We now define the unforgeability of a URS. A URS scheme should be compatible with any signature scheme. Hence, we would like to let the adversary choose signature schemes for the URS scheme. However, the adversary could choose an insecure signature scheme and, in this case, we cannot guarantee unforgeability. Hence, the experiment should provide a list of secure signature schemes and verification keys at the beginning of the experiment. The forge given by the adversary must be with respect to these verification keys.[14] Our definition is similar to the one of *unforgeability with respect to insider corruption* for standard ring signatures [5], which is the strongest unforgeability definition.

Definition 10 (Unforgeability) *Let* $\mathcal{A}$ *be an adversary. We denote by* Ls *a list of challenge signature schemes*

$$\mathsf{Ls} = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i\in[M]}.$$

Consider the following experiment, denoted by $\mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)$*:*

1. *The experiment provides* Ls *to* $\mathcal{A}$*.*
2. *The adversary outputs a list of indices* $\{\mathsf{ind}_i\}_{i\in[\ell]}$*.*
3. *For all* $i \in [\ell]$*, the experiment computes* $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda)$ *and outputs* $R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$ *to the adversary. Also it initialises a set* $\mathcal{K} = \emptyset$ *and remembers the indices* ind_i*.*
4. *The adversary may now make three types of requests*[15]*:*
   - $\mathtt{Corrupt}(i)$*, which the experiment answers with the secret key* sk_i*. Also it adds* vk_i *to* $\mathcal{K}$*.*
   - $\mathtt{URSSign}(m, \bar{R}, i, \bar{S})$ *takes as input an index* $i \in [\ell]$*, a message* m*, a ring of keys* $\bar{R}$ *(not necessarily contained in* R*) and a list of signature schemes* $\bar{S}$*. If* $\mathsf{vk}_i \in \bar{R}$*, we denote its position as* i^**. If additionally* $\mathsf{Sig}_{\mathsf{ind}_i} \in \bar{S}$*, the experiment answers with* $\Sigma \leftarrow \mathsf{URS.Sign}(1^\lambda, \mathsf{sk}_i, m, \bar{R}, i^*, \bar{S})$*.*
   - $\mathtt{Sign}(m, i)$ *takes as input an index* $i \in [\ell]$ *and a message* m*. The experiment answers with* $\Sigma \leftarrow \mathsf{Sig.Sign}_{\mathsf{ind}_i}(1^\lambda, \mathsf{sk}_i, m)$*.*

[14] Note that, in the unforgeability definition for standard ring signatures in [5] a similar situation happens: The forge of the adversary must be with respect to verification keys created honestly and not with respect to maliciously chosen verification keys.

[15] Note that as the key generation algorithms are publicly available, the adversary may honestly generate key pairs itself. The corruption oracle simply serves to corrupt the initial honest keys. Arbitrary additional adversarially chosen keys can be included in ring signature queries, as we do not require $\bar{R} \subseteq R$.

5. $\mathcal{A}$ *outputs* $(\Sigma^*, m^*, R^*, S^*)$.
6. *If* $1 \leftarrow \mathsf{Verify}(\Sigma^*, m^*, R^*, S^*)$, $R^* \subseteq R \setminus \mathcal{K}$, $S^* \subseteq \mathsf{Ls}$ *and the message* m^* *was never queried in a* `URSSign` *or* `Sign` *request, the experiment outputs* 1.[16] *Else, it outputs* 0.

We say that a URS $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ *is* unforgeable, *if for all* $\lambda \in \mathbb{N}$, $M = \mathsf{poly}(\lambda)$, *all lists of EUF-CMA secure signature schemes* $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i \in [M]}$ *and all PPT adversaries* $\mathcal{A}$ *we have that*

$$\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Unf}}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)\right] = \mathsf{negl}(\lambda).$$

In the anonymity experiment, the goal of the adversary is to guess which user created a given signature. We give a general definition called t-anonymity, which mandates that at least t honest keys in the anonymity set must be honestly chosen for anonymity to hold. The adversary may include at least t honest and additional maliciously chosen verification keys (potentially from insecure signature schemes) in a challenge ring. It should still be unable to determine which of the honest parties signed a given URS under that ring.

The case of 2-anonymity coincides with the definition of *anonymity against full key exposure* of [5]. This is the strongest anonymity definition for ring signatures and is even known to imply unrepudiability, meaning that a member in the ring cannot prove that they did not sign the message [31]. As it is the standard case, we will refer to 2-anonymity as *anonymity* throughout this work.

Definition 11 (t-Anonymity) *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *be an adversary. We denote a list of challenge signature schemes by* $\mathsf{Ls} = \{\mathsf{Sig}_i = (\mathsf{Sig.KeyGen}_i, \mathsf{Sig.Sign}_i, \mathsf{Sig.Verify}_i)\}_{i \in [M]}$. *We define the* t*-anonymity experiment* $\mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Anon}_t}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)$ *as follows:*

1. $(\{\mathsf{ind}_i\}_{i \in [\ell]}, \mathsf{aux}_1) \leftarrow \mathcal{A}_1(1^\lambda, \mathsf{Ls})$.
2. *For all* $i \in [\ell]$, *the experiment computes* $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}_{\mathsf{ind}_i}(1^\lambda; \mathsf{r}_i)$ *with random coins* r_i *and sets* $K = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell)$.
3. $(m^*, R^* = (\mathsf{vk}'_1, \ldots, \mathsf{vk}'_p), S^* = (\mathsf{Sig}'_1, \ldots, \mathsf{Sig}'_q), (j_k)_{k \in [t]}, \mathsf{aux}_2) \leftarrow \mathcal{A}_2(K, (\mathsf{r}_1, \ldots, \mathsf{r}_\ell), \mathsf{aux}_1)$ *where* $\mathsf{vk}'_{j_k} \in K$ *for* $k \in [t]$ *with indices* l_k *in* K *(i.e.* $\mathsf{vk}'_{j_k} = \mathsf{vk}_{l_k}$*). Additionally, the signature schemes corresponding to these public keys,* $\mathsf{Sig}_{\mathsf{ind}_{l_k}}$, *must be in the set* S^*. *If these conditions are violated, the experiment aborts.*
4. $\Sigma^* \leftarrow \mathsf{URS.Sign}(1^\lambda, \mathsf{sk}_{l_k}, m^*, R^*, j_k, S^*)$ *where* $k \leftarrow\!\!\$\, [t]$.
5. $k' \leftarrow \mathcal{A}_3(\Sigma^*, \mathsf{aux}_2)$.
6. *If* $k = k'$, *then output* 1. *Else, output* 0.

[16] We can consider the stronger notion, where a forge is valid, if no query of the form $\mathtt{URSSign}(m^*, R^*, \cdot, \cdot)$ or $\mathtt{Sign}(m^* || R^*, i)$ for $\mathsf{vk}_i \in R^*$ was made. This can be achieved by the standard trick of signing the message $(m^* || R^*)$ instead of m^* or a hash $H(m^* || R^*)$ thereof for compactness.

We say that a URS $\mathsf{URS} = (\mathsf{Sign}, \mathsf{Verify})$ *is* t-anonymous*, if for all* $\lambda \in \mathbb{N}$ *, all sizes* $M = \mathsf{poly}(\lambda)$*, all lists of signature schemes* $\mathsf{Ls} = \{\mathsf{Sig}_i\}_{i\in[M]}$ *and all PPT adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ *we have that*

$$\left| \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{URS}}_{\mathsf{Anon}_t}(\mathsf{Ls}, \mathcal{A}, 1^\lambda)\right] - \frac{1}{t} \right| = \mathsf{negl}(\lambda).$$

Efficiency of URS. We remark that URS inherits the efficiency of the most inefficient signature scheme in the ring. For this reason, it is unlikely that we can construct a URS with good and practical parameters and efficiency.

5 Universal Ring Signature from Signature Schemes with Superpolynomial Security

In this section, we present a construction of URS that is based on signature schemes that are superpolynomially hard to forge. From this hardness, we can prove security of the URS scheme using complexity leveraging.

5.1 Construction

We start by presenting the construction of this URS scheme.

For simplicity, we assume, that there is an upper bound on the size of all descriptions of signature verification circuits. Also, for public keys $\mathsf{vk} \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, we assume that they are labeled with their respective schemes. That is, there is a function $\mathsf{tag}(.,.)$ which takes vk and a signature scheme Sig and outputs 1, iff the key vk was made under Sig, but 0 for any other signature verification scheme as input. $\mathsf{Sig.Verify}$ should only accept keys vk with the corresponding tag to Sig, that is $\mathsf{tag}(\mathsf{vk}, \mathsf{Sig}) = 1$.

In the scheme below, we assume that all used signature schemes are unforgeable against superpolynomial adversaries running in $\mathcal{O}(T'(\lambda) \cdot \mathsf{poly}(\lambda))$. We then use a commitment scheme whose hiding property holds against PPT adversaries but can be broken in time $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$. A signature of our URS for a message m includes a commitment to a signature of m in one of the underlying signature schemes. This will give our reduction, which runs in superpolynomial time, an advantage in the unforgeability experiment, where it may extract the commitments and provide a forge against the underlying signature scheme. However, this opening strategy cannot be used by an adversary against anonymity, as they are running in polynomial time.

Construction 1 *Let:*

- CS *be a commitment scheme such that the hiding property holds against polynomial-time adversaries but can be broken in time* $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$*, which is super-polynomial.*
- SPB *be a SPB hashing scheme;*

– $\mathcal{L}$ *be a language such that*

$$\mathcal{L} = \left\{ \begin{array}{c} (m, \mathsf{com}, \mathsf{hk}, h, \mathsf{rhk}, rh) : \exists(\mathsf{vk}, i, \mathsf{Sig.Verify}, \mathsf{ind}, \tau, \rho, \sigma, \gamma) \ s.t. \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau) \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho) \\ 1 \leftarrow \mathsf{CS.Verify}(\mathsf{com}, \sigma, \gamma) \\ 1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}, m, \sigma) \end{array} \right\};$$

where Sig.Verify *is a description of the verification algorithm of a signature scheme* Sig.[17]

– NIWI *be a NIWI scheme for the language*

$$\mathcal{L}_{\mathsf{OR}} = \left\{ \begin{array}{c} (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1) : \\ \exists b \in \{0,1\} \ s.t. \ (m, \mathsf{com}_b, \mathsf{hk}_b, h_b, \mathsf{rhk}_b, rh_b) \in \mathcal{L} \end{array} \right\}.$$

We now describe our scheme in full detail.

$\mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell), i, S = \{\mathsf{Sig}_i\}_{i\in[M]})$

– *Determine an index* ind *such that* $\mathsf{tag}(\mathsf{vk_i}, \mathsf{Sig_{ind}})$. *Parse* $\mathsf{Sig_{ind}} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$. *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i\in[M]}$ *to be the list of verification algorithms in* S.
– *Compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$.
– *Compute* $(\mathsf{hk}_j, \mathsf{shk}_j) \leftarrow \mathsf{SPB.Gen}(1^\lambda, \ell, i)$ *and* $h_j \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_j, R)$ *for* $j \in \{0,1\}$. *Also, compute the proof* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R, i)$.
– *Compute* $(\mathsf{rhk}_j, \mathsf{rshk}_j) \leftarrow \mathsf{SPB.Gen}(1^\lambda, M, \mathsf{ind})$ *and* $rh_j \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_j, S')$ *for* $j \in \{0,1\}$. *Also, compute the proof* $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind})$.
– *Compute* $(\mathsf{com}_0, \gamma_0) \leftarrow \mathsf{CS.Commit}(1^\lambda, \sigma)$ *and* $(\mathsf{com}_1, \gamma_1) \leftarrow \mathsf{CS.Commit}(1^\lambda, 0)$.
– *Set* $x = (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$.
– *Set* $w = (\mathsf{vk}, i, \mathsf{Sig.Verify_{ind}}, \mathsf{ind}, \tau, \rho, \sigma, \gamma_0)$.
– *Compute the proof* $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$.
– *Output* $\Sigma = (\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$.

$\mathsf{Verify}(\Sigma, m, R, S = \{\mathsf{Sig}_i\}_{i\in[M]})$:

– *Parse* Σ *as* $(\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. *Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i\in[M]}$ *to be the list of verification algorithms in* S.
– *Compute* $h_j \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_j, R)$ *for* $j \in \{0,1\}$.
– *Compute* $rh_j \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_j, S')$ *for* $j \in \{0,1\}$.
– *Set* $x = (m, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1)$.
– *If* $1 \leftarrow \mathsf{NIWI.Verify}(x, \pi)$, *output* 1. *Else, output* 0.

We remark, that we only require the verification algorithms of the underlying signature schemes to verify a URS signature. Therefore, we only include these algorithms in S', which is hashed down by SPB and provided to NIWI. This is to reduce size. Essentially, our verification algorithm URS.Verify could only take the list of signature verification algorithms S' as an input, but we state the full list of signature schemes to fit our more general definition.

[17] We assume that for all schemes, |Sig.Verify| is bounded by a polynomial $\beta(\lambda)$.

Signature size. A signature for a message m with respect to a ring R (of size ℓ) and a list of schemes S (of size M) is composed of $\Sigma = (\mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. Both $\mathsf{com}_0, \mathsf{com}_1$ are of size $\mathcal{O}(\mathsf{poly}(\lambda))$ and independent of ℓ and M. The size of the hashing keys $\mathsf{hk}_0, \mathsf{hk}_1$, the proof τ and the circuit $\mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau)$ can be bounded by $\mathcal{O}(\log(\ell) \cdot \mathsf{poly}(\lambda))$. Analogously, $\mathsf{rhk}_0, \mathsf{rhk}_1$, ρ and the runtime of $\mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho)$ are bounded by $\mathcal{O}(\log(M) \cdot \mathsf{poly}(\lambda))$.[18]

Given that, we conclude that the circuit that verifies the relation of language $\mathcal{L}$ has size at most $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. Hence, the proof π has size $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. We conclude that the total size of the signature is $\mathcal{O}((\log(M) + \log(\ell)) \cdot \mathsf{poly}(\lambda))$. Thus, it grows only logarithmic in the number of users in the ring and logarithmic in the number of signature schemes.

5.2 Proofs

We now show that the construction presented above fulfills the required properties for a URS. We start by showing correctness. Then we proceed to prove unforgeability and anonymity. Our proof of unforgeability uses a superpolynomial-time reduction.

Theorem 12 (Correctness). *The scheme presented in Construction 1 is correct, given that* NIWI *is perfectly complete and* SPB *and* CS *are correct.*

Theorem 13 (Unforgeability). *We assume the challenge signature schemes* $\mathsf{LS} = \{\mathsf{Sig}_i\}_{i \in [M]}$ *to be unforgeable against adversaries running in superpolynomial time* $T'(\lambda) \cdot \mathsf{poly}(\lambda)$ *for* $T'(\lambda) \in \omega(\mathsf{poly}(\lambda))$*. We assume, that our commitment scheme allows extraction in time* $T'(\lambda)$*, but is secure against PPT adversaries. Then the scheme presented in Construction 1 is unforgeable against PPT adversaries, given that* NIWI *is perfectly sound and* SPB *is somewhere perfectly binding.*

At a high level, we will build a superpolynomial-time reduction that breaks unforgeability for the underlying signature scheme. The reduction, upon receiving the challenge URS signature $\Sigma^* = (\mathsf{com}_0^*, \mathsf{com}_1^*, \mathsf{hk}_0^*, \mathsf{hk}_1^*, \mathsf{rhk}_0^*, \mathsf{rhk}_1^*, \pi^*)$ from the adversary, opens the commitments com_0^* and com_1^* using brute force. Note that, since we allow the reduction to run in superpolynomial time, it will succeed in breaking the hiding property of the commitment scheme. Then, by the perfect soundness of the NIWI scheme, the reduction can extract a valid signature from either com_0 or com_1 with non-negligible probability and, thus, break the unforgeability of the signature scheme.

Theorem 14 (Anonymity). *Assume that* SPB *is index hiding,* NIWI *is witness-indistinguishable and* CS *is hiding. Then the scheme presented in Construction 1 is anonymous.*

[18] This holds, as we assumed, that we can bound $|\mathsf{Sig.Verify}|$ by a polynomial $\beta(\lambda)$ for all signature schemes Sig.

To prove the theorem above, we build a sequence of hybrids starting from the 2-anonymity game where $k = 1$ and ending at a hybrid describing the game for $k = 2$. Let vk'_{j_1} and vk'_{j_2} be the challenge verification keys in the anonymity game and let $\Sigma = (m^*, \mathsf{com}_0, \mathsf{com}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ be the challenge signature build using vk'_{j_1}. First note that the length of π does not reveal information even if the keys $\mathsf{vk}'_{j_1}, \mathsf{vk}'_{j_2}$ or signature verification circuits are of different length. This is due to the Or-statement in $\mathcal{L}_{\mathsf{OR}}$. In the first hybrid, we change hk_1 and rhk_1 to be SPB hashing keys binding to index j_1. Next, we replace com_1 by a commitment of a valid signature under vk'_{j_2}. In the next hybrid, we can replace the proof π by a new one computed using the new signature under vk'_{j_2} (this change goes unnoticed by the witness indistinguishability of the NIWI). We can now replace com_0 by a commitment of a valid signature under vk'_{j_2}. In the next step, we replace hk_0 and rhk_0 to be SPB hashing keys binding to index j_2 and, finally, compute π as the proof that com_0 is a commitment to a valid signature under vk'_{j_2} for which hk_0 and rhk_0 bind to.

6 Non-compact Universal Ring Signature from Witness Encryption

In this section we present a URS scheme from falsifiable assumptions. The resulting URS has a signature size that scales with the size of the ring. We first present the construction. Then, we proceed to the analysis of the scheme.

6.1 Construction

We now present our construction for URS from WE.

Construction 2 *Let*

- $\mathsf{PRF} : \mathcal{K} \times [\ell] \to \{0,1\}^\lambda$ *be a PRF.*
- $\mathcal{L}'$ *be a language such that*

$$\mathcal{L}' = \left\{ \begin{array}{c} (\{\mathsf{vk}_i\}_{i\in[\ell]} : \exists \left(\{\mathsf{Sig}_{i_j}\}_{j\in[\ell-1]}, K\right) \ \ s.t. \\ r_{i_j} \leftarrow \mathsf{PRF}(K, i_j) \\ (\mathsf{vk}_{i_j}, \mathsf{sk}_{i_j}) \leftarrow \mathsf{Sig.KeyGen}_{i_j}(1^\lambda; r_{i_j}) \end{array} \right\}.$$

- WE *be a witness encryption scheme for language* $\mathcal{L}'$*.*
- SPB *be a SPB hashing scheme;*
- $\mathcal{L}$ *be a language such that*

$$\mathcal{L} = \left\{ \begin{array}{c} (m, \mathsf{ct}, \mathsf{hk}, h, \mathsf{rhk}, rh, x) : \exists(\mathsf{vk}, i, \mathsf{Sig.Verify}, \mathsf{ind}, \tau, \rho, \sigma, r_{\mathsf{ct}}) \ \ s.t. \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{hk}, h, i, \mathsf{vk}, \tau) \\ 1 \leftarrow \mathsf{SPB.Verify}(\mathsf{rhk}, rh, \mathsf{ind}, \mathsf{Sig.Verify}, \rho) \\ \mathsf{ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, x, \sigma; r_{\mathsf{ct}}) \\ 1 \leftarrow \mathsf{Sig.Verify}(\mathsf{vk}, m, \sigma) \end{array} \right\};$$

  where $\mathsf{Sig.Verify}$ *is a description of the verification algorithm of a signature scheme* Sig*.*[19]

[19] We assume again, that for all schemes, $|\mathsf{Sig.Verify}|$ is bounded by a polynomial $b(\lambda)$.

– NIWI *be a NIWI scheme for the language*

$$\mathcal{L}_{\mathsf{OR}} = \left\{ \begin{array}{c} (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, x) : \\ \exists b \in \{0,1\} \ s.t. \ (m, \mathsf{ct}_b, \mathsf{hk}_b, h_b, \mathsf{rhk}_b, rh_b, x) \in \mathcal{L} \end{array} \right\}.$$

We now describe the scheme in full detail.

$\mathsf{Sign}(1^\lambda, \mathsf{sk}_i, m, R = (\mathsf{vk}_1, \ldots, \mathsf{vk}_\ell), i, S = \{\mathsf{Sig}_i\}_{i\in[M]})$*:*

– *Determine an index* ind *with* $\mathsf{tag}(\mathsf{vk_i}, \mathsf{Sig}_{\mathsf{ind}})$*. Parse* $\mathsf{Sig}_{\mathsf{ind}} = (\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$*. Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i\in[M]}$ *to be the list of verification algorithms in* S*.*
– *Compute* $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}_i, m)$*.*
– *Compute* $(\mathsf{hk}_j, \mathsf{shk}_j) \leftarrow \mathsf{SPB.Gen}(1^\lambda, \ell, i)$ *and* $h_j \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_j, R)$ *for* $j \in \{0,1\}$*. Also, compute the proof* $\tau \leftarrow \mathsf{SPB.Open}(\mathsf{hk}_0, \mathsf{shk}_0, R, i)$*.*
– *Compute* $(\mathsf{rhk}_j, \mathsf{rshk}_j) \leftarrow \mathsf{SPB.Gen}(1^\lambda, M, \mathsf{ind})$ *and* $rh_j \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_j, S')$ *for* $j \in \{0,1\}$*. Also, compute the proof* $\rho \leftarrow \mathsf{SPB.Open}(\mathsf{rhk}_0, \mathsf{rshk}_0, S', \mathsf{ind})$*.*
– *Encrypt* $\mathsf{ct}_0 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', \sigma; r_{\mathsf{ct}})$ *and* $\mathsf{ct}_1 \leftarrow \mathsf{WE.Enc}(1^\lambda, x', 0)$*, where* $x' = R$*.*
– *Set* $x = (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, x')$*.*
– *Set* $w = (\mathsf{vk}, i, \mathsf{Sig.Verify}_{\mathsf{ind}}, \mathsf{ind}, \tau, \rho, \sigma, r_{\mathsf{ct}})$*.*
– *Compute the proof* $\pi \leftarrow \mathsf{NIWI.Prove}(x, w)$*.*
– *Output* $\Sigma \leftarrow (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$*.*

$\mathsf{Verify}(\Sigma, m, R, S)$*:*

– *Parse* $\Sigma = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$*. Set* $S' = \{\mathsf{Sig.Verify}_i\}_{i\in[M]}$ *to be the list of verification algorithms in* S*.*
– *Compute* $h_j \leftarrow \mathsf{SPB.Hash}(\mathsf{hk}_j, R)$ *for* $j \in \{0,1\}$*.*
– *Compute* $rh_j \leftarrow \mathsf{SPB.Hash}(\mathsf{rhk}_j, S')$ *for* $j \in \{0,1\}$*.*
– *Set* $x = (m, \mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, h_0, h_1, \mathsf{rhk}_0, \mathsf{rhk}_1, rh_0, rh_1, R)$*.*
– *If* $1 \leftarrow \mathsf{NIWI.Verify}(x, \pi)$*, output* 1*. Else, output* 0*.*

Signature size. A signature for a message m under a ring R (of size ℓ) and a list of schemes S (of size M) is of the form $\Sigma = (\mathsf{ct}_0, \mathsf{ct}_1, \mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$. We first analyze the size of the ciphertexts $\mathsf{ct}_0, \mathsf{ct}_1$. The circuit that verifies the relation $\mathcal{R}'$ of language $\mathcal{L}'$ needs to have size at least $\mathcal{O}(\ell \cdot \mathsf{poly}(\lambda))$ since witnesses for this language are of that size. It is clear that the conditions can be checked in a circuit of this size.

Moreover, a similar analysis as the one made for Construction 1 shows that the total size of $(\mathsf{hk}_0, \mathsf{hk}_1, \mathsf{rhk}_0, \mathsf{rhk}_1, \pi)$ is $\mathcal{O}((\log \ell + \log M) \cdot \mathsf{poly}(\lambda))$. We may assume, that $M \leq \ell$ because signature schemes that no corresponding key exists for in R may be omitted without altering functionality.

We conclude that the signatures in this scheme have size $\mathcal{O}(\ell \cdot \mathsf{poly}(\lambda))$.

6.2 Proofs

We now give the proofs of the security of the proposed scheme.

Theorem 15 (Correctness). *The scheme presented in Construction 2 is correct, given that* NIWI *is perfectly complete.*

Theorem 16 (Unforgeability).
Assume that Sig_i *is EUF-CMA,* PRF *is a pseudorandom function,* NIWI *is perfectly sound and* WE *is correct. Then, the scheme presented in Construction 2 is unforgeable.*

To prove unforgeability, we first build a hybrid where the experiment computes all verification keys, except for vk_{i^*}, using randomness from a PRF (instead of using truly random coins). Note that this change goes unnoticed given that PRF is a PRF. Next, we build a reduction to the unforgeability of the underlying signature scheme. The idea is similar to the proof of Theorem 13. Namely, the goal of the reduction is to extract a valid signature from either ct_0 or ct_1. To do this, note that the reduction is in possession of the key K such that vk_i is created using random coins $\mathsf{PRF}(K, i)$, for all $i \neq i^*$ where vk_{i^*} is the challenge verification key. Then, by the correctness of the WE and the perfect soundness of the NIWI, the reduction can use K to decrypt both ct_0 and ct_1. In the end, there is a non-negligible probability that the reduction can extract a valid signature under vk_{i^*}, thus breaking the unforgeability of the signature scheme.

Theorem 17 (t-Anonymity). *Assume that* NIWI *is witness-indistinguishable,* SPB *is index hiding and* WE *is soundness secure. Then the scheme presented in Construction 2 is t-anonymous where $t = (\lambda - \omega(\log \lambda))/q$ and q is a lower bound of the min-entropy of verification keys in the ring.*

The proof of the theorem is similar to the proof of Theorem 14. However, now we would like to use the security of the WE to replace ct_1 by an encryption of a valid signature under one key (and then replace back by an encryption of 0). To do this, we note that (unlike the unforgeability security proof described above) all verification keys in K are computed using truly random coins. The challenge ring given by the adversary must include at least t of these keys. A simple information-theoretical argument states that there is only a negligible probability that there is a PRF key K such that $t-1$ of these honestly generated verification keys are malformed. This is because they are sampled independently and thus it is unlikely that they are correlated via a PRF key. Hence, we can conclude that $\ell - 1$ verification keys in the adversary's ring are not created using random coins $\mathsf{PRF}(K, i)$, except with negligible probability. In other words, there is a negligible probability that $x' \in \mathcal{L}'$. We can thus use the security of the WE to safely replace encryptions of signatures and encryptions of 0. That is, we switch out the encrypted signature in ct_0 from one under one challenge key to a signature under another one.

7 Compact Witness Encryption for Threshold Conjunction Languages

In this section we present a WE scheme that is compact for threshold conjunction languages. We first define the notion of threshold conjunction languages.

Definition 18 (Threshold Conjunction Languages) *Let $\mathcal{L}$ be an* NP *language with relation $\mathcal{R}$. We define a (t, N)-threshold conjunction language $\mathcal{L}'$ as:*

$$\mathcal{L}' = \left\{(x_1, \ldots, x_N) : \exists \{i_j\}_{j\in[t]} \in [N] \text{ s.t. } x_{i_j} \in \mathcal{L}\right\}.$$

In other words, an accepting instance $(x_1, \ldots, x_N)$ of $\mathcal{L}'$ is one such that there are at least t accepting instances x_{i_j}.

7.1 Construction from Indistinguishability Obfuscation

We now describe our WE scheme for any (t, N)-threshold conjunction language $\mathcal{L}'$. The protocol achieves compact ciphertexts, i.e., of size $\mathcal{O}(\log N)$, when $N - t \in \mathcal{O}(\log N)$.

Construction 3 *Let $N \in \mathsf{poly}(\lambda)$ and t be such that $N - t \in \mathcal{O}(\log N)$ and $\mathcal{L}$ be an* NP *language. Let*

- LSS *be a (t, N)-LSS scheme. In the following, we assume that shares can be written as strings in $\{0,1\}^\lambda$.*
- WE *be a (non-compact) WE scheme for language $\mathcal{L}$.*
- iO *be an obfuscator for all circuits.*
- PPRF *be a puncturable PRF.*
- SSB *be an SSB hashing scheme.*

Additionally, consider the following circuit $\mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N]$ which has the values λ, hk, h, k_0, k_1, t and N hardwired.

$\mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N](i, \tau_i, x_i)$:

- *If* $0 \leftarrow \mathsf{SSB.Verify}(\mathsf{hk}, h, i, x_i, \tau_i)$ *or* $i \geq t$, *return* $\perp$.
- *Compute* $s_i \leftarrow \mathsf{PPRF.Eval}(k_0, i)$ *and random coins* $r_i \leftarrow \mathsf{PPRF.Eval}(k_1, i)$.
- *Compute* $\mathsf{ct}_i \leftarrow \mathsf{WE.Enc}(1^\lambda, x_i, s_i; r_i)$. *Output* ct_i.

We now define the WE scheme for the (t, N)-conjunction language $\mathcal{L}'$.

$\mathsf{Enc}(1^\lambda, x, m)$:

- *Parse* $x = (x_1, \ldots, x_N)$.
- *Create PPRF keys* $k_0 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$ *and* $k_1 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$.
- *For* $i \in [t-1]$, *compute pseudorandom shares* $s_i \leftarrow \mathsf{PPRF}(k_0, i)$. *Compute the remaining shares* $(s_t, \ldots, s_N) \leftarrow \mathsf{LSS.RemainShare}(m, s_1, \ldots, s_{t-1})$.
- *Compute* $\mathsf{hk} \leftarrow \mathsf{SSB.Gen}(1^\lambda, t-1, j)$ *for* $j \leftarrow\!\!\$\, [t-1]$. *Moreover, compute* $h \leftarrow \mathsf{SSB.Hash}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\})$.
- *Consider the circuit* $\mathcal{C} = \mathcal{C}[\lambda, \mathsf{hk}, h, k_0, k_1, t, N]$. *Compute* $\bar{\mathcal{C}} \leftarrow \mathsf{iO}(1^\lambda, \mathcal{C})$.
- *For* $i \in \{t, \ldots, N\}$, *compute encryptions* $\mathsf{ct}_i \leftarrow \mathsf{WE.Enc}(1^\lambda, x_i, s_i)$.
- *Output* $\mathsf{ct} = (\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$.

$\mathsf{Dec}(w, \mathsf{ct})$:

- *Parse* $w = (w_{i_1}, \ldots, w_{i_t})$ *and* ct *as* $(\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$
- *For* $i \in [t-1]$, *compute* $\tau_i \leftarrow \mathsf{SSB.Open}(\mathsf{hk}, \{x_1, \ldots, x_{t-1}\}, i)$ *and run* $\mathsf{ct}_i \leftarrow \bar{\mathcal{C}}(i, \tau_i, x_i)$.
- *For* $j \in [t]$, *decrypt* $s_{i_j} \leftarrow \mathsf{WE.Dec}(w_{i_j}, \mathsf{ct}_{i_j})$.
- *Reconstruct* $m \leftarrow \mathsf{LSS.Reconstruct}(s_{i_1}, \ldots, s_{i_t})$. *Output* m.

Ciphertext size. The ciphertext is of the form $(\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}, \bar{\mathcal{C}}, \mathsf{hk})$. Assume that the language $\mathcal{L}$ has a verification circuit $\mathcal{C}_{\mathcal{L}}$. The ciphertexts ct_i for $i \in \{t, \ldots, N\}$ have size $\mathcal{O}(|\mathcal{C}_{\mathcal{L}}| \cdot \mathsf{poly}(\lambda))$. Since $N - t \in \mathcal{O}(\log(N))$, the size of $\{\mathsf{ct}_i\}_{i\in\{t,\ldots,N\}}$ is $\mathcal{O}(\log(N) \cdot |\mathcal{C}_{\mathcal{L}}| \cdot \mathsf{poly}(\lambda))$. The obfuscated circuit $\mathcal{C}$ implements the $\mathsf{SSB.Verify}$ algorithm which is of size $\mathcal{O}(\log(N))$. Moreover, all other operations in $\mathcal{C}$ are independent of N and depend only on $|\mathcal{C}_{\mathcal{L}}|$. Hence, $|\mathcal{C}| \in \mathcal{O}(\log(N) \cdot |\mathcal{C}_{\mathcal{L}}| \cdot \mathsf{poly}(\lambda))$. Finally, the hashing key hk is of size $\mathcal{O}(\log(N))$ by the efficiency requirements of SSB.

We conclude that the scheme presented above outputs ciphertexts of size $\mathcal{O}(\log(N) \cdot |\mathcal{C}_{\mathcal{L}}| \cdot \mathsf{poly}(\lambda))$.

7.2 Proofs

We now prove that the scheme is correct and soundness secure.

Theorem 19 (Correctness). *The scheme presented in Construction 3 is correct, given that* LSS, SSB *and* WE *are correct.*

Theorem 20 (Soundness security). *The scheme presented in Construction 3 is soundness secure given that* SSB *is index hiding and somewhere statistically binding,* iO *is a secure* $i\mathcal{O}$ *obfuscator,* PPRF *is pseudorandom at punctured points,* WE *is soundness secure and* LSS *is private.*

Before presenting the formal proof, we give a brief outline. The proof follows a sequence of hybrids, where the last one can be reduced to the privacy of the LSS. First, note that if $x \notin \mathcal{L}'$, then there do not exist t instances $x_i \in \mathcal{L}$. Assume, for simplicity that $t = N$, then there exists an index i^* such that $x_{i^*} \notin \mathcal{L}$. We start with a hybrid that is identical to the real soundness security game.

Then, we use the index hiding of the SSB hashing scheme to replace hk by a hashing key that is binding to index i^*. We then use the *puncturing* technique of [34]. That is, we create punctured PRF keys k_0' and k_1' (by puncturing the PPRF keys k_0 and k_1 respectively) at the point i^*. At the same time, we embed into the obfuscated circuit the ciphertext $\mathsf{ct}_{i^*} \leftarrow \mathsf{WE.Enc}(1^\lambda, x_{i^*}, s_{i^*}; r_{i^*})$ where $s_{i^*} \leftarrow \mathsf{PPRF.Eval}(k_0, i^*)$ and $r_{i^*} \leftarrow \mathsf{PPRF.Eval}(k_1, i^*)$. Given that the SSB is somewhere statistically binding at the point i^*, the circuits are functionally equivalent and we can use the security of the $i\mathcal{O}$ obfuscator to argue indistinguishability. We can now replace the values s_{i^*}, r_{i^*} by uniform ones since the PPRF is pseudorandom at punctured points. Finally, we replace ct_{i^*} by an encryption of 0. To conclude the proof, we can easily build a reduction to the security of the LSS.

In the more general case, some WE encryptions with respect to false statements are computed using the obfuscated program and some are given in the plain. For the former ones, we simply repeat the process above. For the latter ones, we use security of WE to replace these encryptions by encryptions of 0.

In the full version of the paper we present a variant of this protocol that does not incur in exponential loss of security in the reduction.

7.3 Compact Universal Ring Signature from Compact WE for Threshold Conjunction Languages

Consider again the URS construction of Section 6. One of the requirements of this URS scheme is a (non-compact) WE for a language $\mathcal{L}'$ which is itself a $(N-1, N)$ -threshold conjunction language. When we plug the WE scheme for (t, N)-threshold conjunction languages as a drop-in replacement for non-compact WE, we obtain a compact URS scheme.

Specifically, the following theorem is a direct consequence of plugging the compact WE scheme for (t, N)-threshold conjunction languages described above with the URS signature from Section 6.

Theorem 21. *Let*

- $\mathsf{PRG} : \{0,1\}^{\lambda/2} \to \{0,1\}^{\lambda}$ *be a PRG.*
- $\mathcal{L}'$ *be the* $(\ell-1, \ell)$ *threshold conjunction language defined in Construction 2.*
- WE *be a compact witness encryption scheme for the* $(\ell-1, \ell)$ *threshold conjunction language* $\mathcal{L}'$*. As we have just established, this primitive can be built from secure* $i\mathcal{O}$*,* $(\ell-1, \ell)$*-LSS, (non-compact) WE for* NP*, PPRF and SSB.*
- SPB *be a SPB hashing scheme;*
- $\mathcal{L}$ *and* $\mathcal{L}_{\mathsf{OR}}$ *be the languages defined in Construction 2.*
- NIWI *be a NIWI scheme for* $\mathcal{L}_{\mathsf{OR}}$*.*

Then there exists a URS scheme that satisfies correctness, anonymity and unforgeability. Moreover, a signature Σ *with respect to a ring of users* R *and a ring of signature schemes* S *has size* $|\Sigma| \in \mathcal{O}((\log \ell + \log M)\mathsf{poly}(\lambda))$ *where* $\ell = |R|$ *and* $M = |S|$*.*

Acknowledgments

Nico Döttling: Funded by the European Union. 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. (ERC-2021-STG 101041207 LACONIC)

References

1. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of-n signatures from a variety of keys. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 415–432. Springer, Heidelberg, Germany, Queenstown, New Zealand (Dec 1–5, 2002)
2. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 308–326. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)
3. Backes, M., Döttling, N., Hanzlik, L., Kluczniak, K., Schneider, J.: Ring signatures: Logarithmic-size, no setup - from standard assumptions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 281–311. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93: 1st Conference on Computer and Communications Security. pp. 62–73. ACM Press, Fairfax, Virginia, USA (Nov 3–5, 1993)
5. Bender, A., Katz, J., Morselli, R.: Ring signatures: Stronger definitions, and constructions without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006: 3rd Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 3876, pp. 60–79. Springer, Heidelberg, Germany, New York, NY, USA (Mar 4–7, 2006)
6. Bitansky, N., Garg, S., Lin, H., Pass, R., Telang, S.: Succinct randomized encodings and their applications. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing. pp. 439–448. ACM Press, Portland, OR, USA (Jun 14–17, 2015)
7. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th Annual Symposium on Foundations of Computer Science. pp. 171–190. IEEE Computer Society Press, Berkeley, CA, USA (Oct 17–20, 2015)
8. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 103–112. ACM Press, Chicago, IL, USA (May 2–4, 1988)
9. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 416–432. Springer, Heidelberg, Germany, Warsaw, Poland (May 4–8, 2003)
10. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part II. Lecture Notes in Computer Science, vol. 8270, pp. 280–300. Springer, Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013)
11. Boyen, X.: Mesh signatures. In: Naor, M. (ed.) Advances in Cryptology – EUROCRYPT 2007. Lecture Notes in Computer Science, vol. 4515, pp. 210–227. Springer, Heidelberg, Germany, Barcelona, Spain (May 20–24, 2007)
12. Brakerski, Z., Koppula, V., Mour, T.: NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 12172, pp. 738–767. Springer (2020)

13. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 33–65. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)
14. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 609–626. Springer, Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004)
15. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st Annual Symposium on Foundations of Computer Science. pp. 308–317. IEEE Computer Society Press, St. Louis, MO, USA (Oct 22–24, 1990)
16. Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. Lecture Notes in Computer Science, vol. 6110, pp. 197–215. Springer, Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010)
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th Annual Symposium on Foundations of Computer Science. pp. 40–49. IEEE Computer Society Press, Berkeley, CA, USA (Oct 26–29, 2013)
18. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th Annual ACM Symposium on Theory of Computing. pp. 467–476. ACM Press, Palo Alto, CA, USA (Jun 1–4, 2013)
19. Garg, S., Gupta, D.: Efficient round optimal blind signatures. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 477–495. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014)
20. Garg, S., Rao, V., Sahai, A., Schröder, D., Unruh, D.: Round optimal blind signatures. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 630–648. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)
21. Garg, S., Srinivasan, A.: A simple construction of iO for turing machines. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018: 16th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 11240, pp. 425–454. Springer, Heidelberg, Germany, Panaji, India (Nov 11–14, 2018)
22. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: A framework to compose σ-protocols for disjunctions. Cryptology ePrint Archive, Report 2021/422 (2021), `https://ia.cr/2021/422`
23. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (Dec 1994)
24. Goldwasser, S., Kalai, Y.T.: On the (in)security of the Fiat-Shamir paradigm. In: 44th Annual Symposium on Foundations of Computer Science. pp. 102–115. IEEE Computer Society Press, Cambridge, MA, USA (Oct 11–14, 2003)
25. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 339–358. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006)

26. Hohenberger, S., Koppula, V., Waters, B.: Universal signature aggregators. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 3–34. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)
27. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) ITCS 2015: 6th Conference on Innovations in Theoretical Computer Science. pp. 163–172. Association for Computing Machinery, Rehovot, Israel (Jan 11–13, 2015)
28. Jain, A., Jin, Z.: Non-interactive zero knowledge from sub-exponential DDH. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 3–32. Springer (2021)
29. Libert, B., Peters, T., Qian, C.: Logarithmic-size ring signatures with tight security from the DDH assumption. In: López, J., Zhou, J., Soriano, M. (eds.) ESORICS 2018: 23rd European Symposium on Research in Computer Security, Part II. Lecture Notes in Computer Science, vol. 11099, pp. 288–308. Springer, Heidelberg, Germany, Barcelona, Spain (Sep 3–7, 2018)
30. Okamoto, T., Pietrzak, K., Waters, B., Wichs, D.: New realizations of somewhere statistically binding hashing and positional accumulators. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9452, pp. 121–145. Springer, Heidelberg, Germany, Auckland, New Zealand (Nov 30 – Dec 3, 2015)
31. Park, S., Sealfon, A.: It wasn't me! - Repudiability and claimability of ring signatures. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part III. Lecture Notes in Computer Science, vol. 11694, pp. 159–190. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)
32. Peikert, C., Shiehian, S.: Noninteractive zero knowledge for NP from (plain) learning with errors. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology – CRYPTO 2019, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 89–114. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)
33. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) Advances in Cryptology – ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 552–565. Springer, Heidelberg, Germany, Gold Coast, Australia (Dec 9–13, 2001)
34. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing. pp. 475–484. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)
35. Shacham, H., Waters, B.: Efficient ring signatures without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 4450, pp. 166–180. Springer, Heidelberg, Germany, Beijing, China (Apr 16–20, 2007)
36. Tso, R.: A new way to generate a ring: Universal ring signature. Computers & Mathematics with Applications 65(9), 1350–1359 (2013), `https://www.sciencedirect.com/science/article/pii/S0898122112000491`, advanced Information Security