Signature-based Witness Encryption with Compact Ciphertext

Gennaro Avitabile[1] Nico Döttling[2] Bernardo Magri[3,4] Christos Sakkas[3] Stella Wohnig[2,5]

[1]IMDEA Software Institute, Spain

[2]CISPA Helmholtz Center for Information Security, Germany

[3]The University of Manchester, United Kingdom

[4]Primev

[5]Saarland University, Germany

gennaro.avitabile@imdea.org,
{doettling,stella.wohnig}@cispa.de,
{bernardo.magri,christos.sakkas}@manchester.ac.uk

September 20, 2024

Abstract. Signature-based witness encryption (SWE) is a recently proposed notion that allows to encrypt a message with respect to a tag T and a set of signature verification keys. The resulting ciphertext can only be decrypted by a party who holds at least k different valid signatures w.r.t. T and k different verification keys out of the n keys specified at encryption time. Natural applications of this primitive involve distributed settings (e.g., blockchains), where multiple parties sign predictable messages, such as polling or randomness beacons. However, known SWE schemes without trusted setup have ciphertexts that scale linearly in the number of verification keys. This quickly becomes a major bottleneck as the system gets more distributed and the number of parties increases.

Towards showing the feasibility of SWE with ciphertext size sub-linear in the number of keys, we give a construction based on indistinguishability obfuscation (iO) for Turing machines and strongly puncturable signatures (SPS).

1 Introduction

Threshold cryptography focuses on distributing the security of a cryptosystem among multiple parties, ensuring that a minimum number of these parties, known as the threshold, is required to operate the system. In recent years, threshold schemes have garnered significant attention from the community [9,29,7,16,26,5], primarily due to their applications in decentralized systems like blockchains. A recent example is the notion of threshold signature-based witness encryption (SWE) that was first introduced by Döttling et al. [9]. It allows to encrypt a plaintext with respect to a tag and a set of verification keys for a signature scheme. Once a sufficient number of signatures of this tag under these verification keys are provided, these signatures can be used to efficiently decrypt the SWE ciphertext. The main application of SWE described in [9]

is to build a time-release encryption scheme by combining SWE with a proof-of-stake (PoS) blockchain: The idea is to use the verification keys of the blockchain committee members that sign every new block created in the chain as the SWE verification keys, and the block number of a block in the chain as the tag; then, once a block with that number is built the signatures of the committee members on that number can be used to decrypt the SWE ciphertext. Madathil *et al.* [29] construct an almost equivalent primitive called *verifiable witness encryption based on threshold signatures* (VweTS), and provide another compelling use-case for such a scheme: They facilitate blockchain transactions conditioned on "real life" events that happen outside of the blockchain system, by requiring a threshold number of oracles' signatures (third-party services that certify real-world events like weather data, outcomes of bets etc.) to unlock data to complete the transaction.

Conceptually, the notion of SWE can be seen as an interesting special case of witness encryption [15] which allows one to encrypt a message under any NP statement such that a witness for that statement allows to decrypt the ciphertext.

Another seemingly related notion is the one of threshold encryption schemes [8,12]. They are encryption schemes where decryption takes place with the help of a threshold of decryption servers. A significant difference w.r.t. SWE is that in threshold encryption a setup phase is assumed. During this setup, a public key for the scheme and correlated secret keys for the decryption servers are computed via a protocol. Furthermore, it is necessary to communicate with these servers in order to decrypt the message. A simple approach to threshold encryption [8] is to create the correlated secret keys as t-of-n linear secret shares a_i of an ElGamal secret key a, the corresponding public key is g^a. When a client gets an encryption (g^r, mg^{ar}) of m under g^a , they send g^r to each of the decryption servers and receive back decryption shares $g^{r(a_i^{-1})}$. The client then recombines the shares to get $g^{r(a^{-1})}$ given that at least t servers participate.

A big advantage of SWE over the above approach is that SWE does not require to setup correlated keys or communication between decryptors and signers. In fact, the servers may even be unaware that encryptions are being made w.r.t. signatures they may release in the future. This requirement is in line with the original application of SWE in the blockchain setting, as no additional load should be put on the blockchain committee other than simply producing a signature.

A common feature of the applications of SWE is the potentially large set of signers, as it is desirable that the members of blockchain committees or the number of trusted oracles can increase as much as possible to tackle centralization. Therefore, a critical drawback of the SWE/VweTS constructions in [9,29] is that the size of their ciphertexts grows linearly with the number of verification keys used in the signing procedure. A natural question to ask is whether it is possible to construct an SWE scheme that does not suffer from this limitation, and neither relies on a (long) trusted setup nor on strong ideal models (such as the programmable generic group model [35]). In this work, we answer this question in the affirmative by introducing the notion of *compact signature-based witness encryption* (cSWE). In a cSWE scheme, the size of the ciphertext only grows poly-logarithmically in the number of verification keys. We provide a formal definition of cSWE and present a construction based on Turing-machine indistinguishability obfuscation and strongly puncturable signatures. In the next section, we describe our contributions in more detail.

1.1 Our contributions

Compact SWE. In this work, we construct the first cSWE scheme that allows one to encrypt a message m with respect to a reference tag T and a set of verification keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ such that the ciphertext size only scales poly-logarithmically in n. Importantly, our construction

does not rely on a trusted setup. The construction is based on indistinguishability obfuscation for Turing Machines [27,17] and it achieves ciphertext size in the order of $\mathcal{O}(\mathsf{poly}(\lambda \cdot \log n))$. We prove security of our construction w.r.t. non-adaptive adversaries where the reference tag T and the indices of the corrupt verification keys are known ahead of time. Our result establishes principal feasibility of this notion in the non-adaptive setting and may pave the way towards a more efficient implementation of cSWE in the future.

1.2 Technical Overview

We will now provide an outline of our constructions and techniques.

Compact SWE. We start by reviewing the high-level idea of the SWE construction given in [9]. To encrypt a message m w.r.t. a reference tag T and a set of verification keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, the message m is first encrypted under a symmetric encryption scheme using a freshly sampled key K. The key K is then secret-shared using a t-of-n linear secret sharing scheme, in the case of [9] this is the Shamir scheme [34][1] where the shares $s_1, \ldots, s_n$ of K satisfy $\sum_{j=1}^{t} s_{i_j} \cdot L_{i_j} = K$ for Lagrange coefficients L_{i_j}. The ciphertext is structured in a way such that for each individual share, two elements are added to the ciphertext to ensure that the share s_i can be retrieved given a valid signature of T under the verification key vk_i.

In a bit more detail, the SWE construction of [9] uses BLS [2] as the underlying signature scheme. In BLS, a valid signature σ on T under vk must satisfy the equation $e(\sigma, g_2) = e(H(T), \mathsf{vk})$, where e is a bilinear map from groups G_1, G_2 into G_T, and g_2 is a generator of G_2. Then, for each share s_i, they create an encryption of that share by choosing some randomness $r_i \in \mathbb{Z}_p$, and outputting $(g_2^{r_i}, e(H(T), \mathsf{vk})^{r_i} \cdot s_i)$. Note that, having a signature σ_i of message T under key vk_i, a decryptor can compute $e(H(T), \mathsf{vk})^{r_i} = e(\sigma_i, g_2^{r_i})$, and thus easily get s_i. However, without such a signature the value $e(H(T), \mathsf{vk})^{r_i}$ is indistinguishable from random under the bilinear Diffie-Hellman assumption.

We will base our techniques on the above idea of encrypting each share s_i, such that it can be individually retrieved if a signature of T under vk_i is given. However, we will need to depart from the BLS-based approach [9] to achieve a ciphertext size independent of n.

The (non-compact) SWE construction of [9] achieves *adaptive fully malicious security*; the adversary can corrupt any unqualified set of signers and even choose their verification keys in a fully malicious manner.

Compact SWE and Adaptive Security. There seems, however, to be a substantial barrier for achieving adaptive security for *compact* SWE, and in fact it seems that a heuristic such as e.g. programmable random oracle model or the programmable generic group model may be necessary in this setting (see the discussion on [16] in the related works section below). From an information theoretic perspective, e.g. for a threshold of $k = n/2$, a ciphertext of size $o(n)$ is too small to even encode the set of corrupted parties, which requires $\Omega(n)$ bits. Hence, it seems hard to make a ciphertext behave differently for honest and corrupted keys as the ciphertext cannot "know" which keys are corrupted and which are not.

Hence, somewhat expectedly guessing-based transformations from non-adaptive to adaptive security fail in this setting. If we guess the set of corrupted parties ahead of time, we need to compensate for a security loss of order $2^{\Omega(n)}$. But this means that the underlying primitives need to at least provide $\Omega(n)$ bits of security, which in turn results in a ciphertext of size $\Omega(n)$, which is not compact by our definition.

[1] This means $s_i = K + \sum_{j=1}^{t} r_j (\xi_i)^j \mod p$ for a prime p, random r_j and distinct evaluation points $\xi_i \in \mathbb{Z}_p$.

A similar situation occurs in the setting of *adaptively secure succinct non-interactive arguments of knowledge (SNARGs)*: The object in question, in this case a certificate π, is so small that it cannot encode a sufficient amount of information about an adaptively chosen false statement x, hence the guessing-based non-adaptive to adaptive security transformation incurs a security loss incompatible with the succinctness requirement.

Gentry and Wichs [18] provided a formal barrier result which shows that there is no construction of adaptive SNARGs with a black-box security reduction from any falsifiable assumption [30]. While we do not provide a formal argument as this is beyond the scope of our work, the basic idea of this argument (likely) carries over to the setting of compact SWE. In fact, the Gentry-Wichs result [18] holds even for designated verifier SNARGs, and any succinct witness encryption scheme immediately gives rise to a designated verifier SNARG for the same language.

Somewhat surprisingly, a recent work by Wu and Waters [36] showed that this security loss can be "pushed into a CRS". That is by making the CRS scale with the logarithm of the security loss, the size of the certificate π can be kept small. Since we are interested in compact SWE constructions without setup, this is not an option in our setting.

Non-adaptive Security. Hence, we will focus on a notion of non-adaptive security in this work. Specifically, in this notion we require that the adversary chooses the reference message T and the indices of the corrupted keys ahead of time. Only after that does the adversary get $t-1$ honestly generated pairs of verification and signing keys (at the corrupted positions) and $n-t-1$ honestly chosen verification keys for the honest parties. We then give the adversary access to a signing oracle which will sign for all honest keys and all messages except the challenge tag T. Under these constraints we ask for indistinguishability security for the signature witness encryption, i.e. the adversary should not be able to distinguish SWE encryptions with respect to $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and T of two distinct messages $m_0 \neq m_1$.

Constructions with Structured Common Reference String. Before we discuss our construction in the plain model, we will briefly discuss how we can construct compact SWE if we are allowed to shift the burden into a *long and structured CRS*. The basic idea is similar to the construction of zero-knowledge SNARGs of Sahai and Waters [33]: We can *delegate* the task of generating and decrypting compact SWE ciphertexts to a pair of (large) obfuscated circuits given in the CRS. The first circuit takes verification keys $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$, the tag T, the message m as well as additional random coins and produces a ciphertext c and a succinct commitment h binding to the vk_i and c. The second obfuscated circuit takes as input $\mathsf{vk}_1, \ldots, \mathsf{vk}_n$ and c, an opening of h to these values, as well as signatures σ_i for some $I \subset [n]$ of size at least k. The circuit checks if each σ_i is a valid signature of T under vk_i and if so returns the message m. We can establish security of this construction using a standard puncturing argument and relying on standard tools such as puncturable PRFs and SSB hashing. This construction uses a long and structured CRS (consisting of obfuscated programs) in a critical way.

A First Attempt. In order to achieve a compact SWE construction without trusted setup, we have to depart from the above blueprint. Our basic idea is to adapt the construction of witness encryption from iO [14] to the setting of compact SWE, but this raises several challenges. Hence, consider the following attempt to construct a cSWE. The ciphertext consists of a symmetric key encryption of the message m under key K and an obfuscated circuit C. The circuit C pseudorandomly generates the secret shares under say the Shamir scheme of key K on demand; on input an index i and a valid signature σ for T under vk_i, the circuit produces the i-th share of the key K. The randomness for generating shares is taken from a puncturable PRF.

There are two evident issues with this approach. (1) A minor issue is that we cannot hardwire all the verification keys vk_i into the circuit C, as this would require the circuit size growing

linearly with n, contradicting our compactness requirement. (2) Computing individual Shamir shares inside the circuit C implies evaluating a degree $t-1$ polynomial (that takes time $\Theta(n)$), requiring a circuit of size $\Omega(n)$, again contradicting our compactness requirement.

There will be, however a more subtle third issue which surfaces when trying to prove security of this construction: If we were to rely on standard puncturing techniques to *erase* the shares of honest parties in the challenge ciphertext, we would need to puncture the above circuit $n-k$ times. But this would again require an obfuscated circuit of size $\Omega(n)$!

Our Basic Approach. Starting from the above sketch, we will address issues (1) and (2) above as follows. To address the issue (1), we will rely on somewhere statistically binding (SSB) hashing [24]. An SSB hash function is a keyed commitment scheme, which allows to succinctly produce a committing hash h of a database of size n. The key generation algorithm takes an index $i \in [n]$, and outputs a hashing key guaranteeing that the hash is computationally binding on all indices and statistically binding on i. The binding index is hidden by the commitment scheme. We use the SSB scheme to commit to all verification keys $(i, \mathsf{vk}_i)_{i\in[n]}$ and force the decryptor to input $(i, \mathsf{vk}_i, \sigma)$ and a valid opening τ for the SSB hash. This ensures, that except with negligible probability, on input $(i, \mathsf{vk}, \sigma, \tau)$, vk actually is the i-th key and σ is a signature for vk_i, so outputting s_i is justified. To address issue (2), we will rely on iO for Turing machines [27,17] (TM). The size of an obfuscated Turing machine depends polynomially on its description size, but only poly-logarithmically on its runtime.

Towards Proving Security. The intuition for the security proof is that we want to replace the obfuscated TM M with an equivalent TM M' which has no information about the shares or the shared key. The basic idea of the hybrid argument is as follows.

Since we are in a setting of non-adaptive security, the security reduction knows which signers will be corrupted even before their corresponding verification keys are generated. We will use a standard SSB argument to go through a sequence of hybrids, where in hybrid i we make the SSB hash statistically binding to the i-th *uncorrupted* verification key, call this key vk_i. We would like to argue that since the adversary never obtains a signature σ_i of T under vk_i, he cannot make the obfuscated TM output a share s_i of the message m_i. However, standard EUF-CMA security seems far too weak to guarantee this: In order to use indistinguishability security of iO, we need to move into a situation where it is *impossible* (rather than computationally hard) to make the TM accept some signature σ_i for T under vk_i and output the corresponding share s_i. To address this issue, we use strongly puncturable signatures (Section 3). These allow a special key generation $\mathsf{PKeyGen}(T)$, which outputs a punctured key pair $(\mathsf{vk}, \mathsf{sk})$ at message T. With overwhelming probability, no valid signature exists for message T, while the verification key remains indistinguishable from a standard non-punctured key. This type of signatures was introduced under the name of all-but-one signatures in [21] where various instantiations based on different number-theoretic assumptions (e.g., RSA, pairing-based assumptions, LWE) were proposed. Equipped with this tool, the reduction can now replace all honest keys with punctured keys, punctured at T. Hence, by additionally relying on SSB hashing as explained above we can rely on iO security to gradually replace the TM M by a TM M' which never outputs shares s_i for indices i of honest parties.

The Final Scheme. However, this transformation does not quite suffice to establish security. Recall that the shares s_i of the message m are generated *pseudorandomly*, that is an adversary with access to a plain description of the TM M' described above could still easily recover the PRF key $\tilde{K}$ used to generate the shares and just generate them by himself. Note also that iO security by itself does not guarantee that $\tilde{K}$ is hidden. The standard tool commonly used to argue security in this setting is puncturing: If we were to puncture the key $\tilde{K}$ in a suitable

manner, then we might be able to ensure that this punctured key does not leak information about the shares of honest parties. While there are some minor issues with this idea, such as the fact that the shares s_i are correlated and a contrived puncturing argument would be necessary, the big issue here is we would need to puncture at $n - k = \Omega(n)$ points, which once again contradicts our compactness requirement! The reason why we need to remove all honest shares simultaneously is that otherwise Shamir secret sharing offers no security! Removing one share at a time is useless, as this share could easily be recomputed from the other shares. Hence, puncturing will not help us, and we need to depart from the puncturing approach. But where could we store shares of the message m instead? Observe that our ciphertext already includes an SSB commitment to all the verification keys vk_i, and we will precisely leverage this feature in our solution: We will augment the verification keys vk_i by some additional auxiliary information which contains a sufficiently large slot to encode a Shamir share. In the real world this auxiliary information is just a random string. But in the security reduction we can embed an encryption of the corresponding share s_i into vk_i. We then use iO security to switch to a machine M' that reads its shares from the input, limiting its internal computation to checking the validity of the input SSB opening and the signature. Upon success, it decrypts and outputs the share in vk_i; otherwise, it aborts. This switch between computational models is feasible because we bind the SSB hash to each index $i^* \in [n]$. This ensures that on input $(i^*, \mathsf{vk}, \sigma, \tau)$, if the obfuscated machine produces an output, then $\mathsf{vk} = \mathsf{vk}_{i^*}$ must hold. Consequently, the correct share is obtained by decrypting vk, and the output remains consistent whether the shares are computed internally or decrypted from the public keys.

While this outlines our main technical ideas, there are some additional subtle technical challenges which our construction in Section 4 needs to address.

On the Use of iO. Our construction uses iO for Turing Machines in a crucial way. Specifically, we use iO to delegate the generation of shares to the decrypter, the ciphertexts in our scheme constitute compressed versions of a pseudorandom share vector. This type of compression is currently beyond the scope of other compact delegation techniques (e.g. Laconic Function Evaluation [32]), and in fact general purpose delegation schemes with even a weak amount of output compression imply iO [28].

1.3 Related Work

Signature-based Witness Encryption. The only known construction of signature-based witness encryption [9] grows linearly in the number of verification keys input to the encryption procedure. The same is true for verifiable witness encryption based on threshold signatures [29]. While the ciphertext's size of these constructions is asymptotically worse than ours, their main focus is on concrete efficiency. However, both [9,29] rely on specific assumptions in idealized models, i.e. the bilinear Diffie-Hellman assumption and the random oracle model, whereas we focus on a compact-ciphertext construction in the plain model. Furthermore, we point out that these previous works achieve fully adaptive security notions, as opposed to our non-adaptive security as discussed in Section 1.2. There exist also concretely efficient schemes (in the ROM) with constant-size ciphertexts [13], but they can handle only the special case of $t = n$.

Threshold Encryption. Threshold encryption schemes [8,12] are encryption schemes where decryption can take part only with the cooperation of a threshold number of decryption servers. These constructions achieve constant ciphertext size. However, they require a correlated set up of decryptors' secret keys. Furthermore, decryption requires communicating with t servers to get partial decryptions that are then aggregated to get the plaintext. Recently, [7] a scheme with efficient batch-decryption for threshold encryption was proposed. This solution still requires a

correlated set up of secret keys, but a server can output partial decryptions for many ciphertexts at once, reducing the communication overhead. In their scheme, the keys are generated by a trusted dealer.

In [16] the notion of silent setup for threshold encryption is introduced. This means, that the involved decryption servers are now able to choose their key pairs independently and encryptors can deterministically aggregate the public keys of their chosen decryption servers into a single succinct encryption key, which can be used to encrypt a message. Dropping correlated key creation is achieved by pushing the complexity into a highly structured CRS and "hints", which can be seen as extensions of the public keys of each party. Both the CRS and the hints are linear in the size of maximum decryption servers. For each decryption procedure direct communication with the decryption servers is still required. Their scheme achieves adaptive security in the *programmable* generic group model.

Witness Encryption and Committe-based Witness Encryption. Witness Encryption [15] is defined for an NP language $\mathcal{L}$ with poly-time relation $\mathcal{R}$. Encryption of a message m is w.r.t a statement x and decryption is possible with a witness w s.t. $(x, w) \in \mathcal{R}$. Security intuitively says that a ciphertext does not reveal any information about m if $x \notin \mathcal{L}$. Security is extended in [20] in the form of *extractable* witness encryption, which also offers security guarantees even when $x \in \mathcal{L}$. In [22] extractable witness encryption on the blockchain (eWEB) is introduced. In this scheme, a message encrypted with respect to a statement x is secret-shared among several committee members and labeled with x. The decryptor must interact with a threshold number of committee members, proving they have a valid witness to collect the shares and decrypt the message. However, the storage complexity for each committee member increases with the number of shares they hold. This limitation is addressed by [10], which proposes creating a joint public key for encryption while giving each committee member a share of a correlated secret key. Another work [11] further improves this by hiding the statement w.r.t. encryption is done. Their ciphertext size is independent of the number of keys. However, they require a correlated key setup where the committee has a single public key and the secret key is shared among the members. In another work, [4], blockchain witness encryption is independently introduced, serving a similar function to [22], but it requires deploying a smart contract on the blockchain for each encryption. The critical difference between SWE and these works is that in SWE committee keys are independently sampled, and decryption relies solely on the availability of signatures from the committee.

2 Preliminaries

Notation. We denote by $\lambda \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ the output of the algorithm $\mathcal{A}$ on input in where $\mathcal{A}$ is randomized with $r \leftarrow \{0,1\}^*$ as its randomness. We omit this randomness when it is obvious or not explicitly required. By $\mathcal{A}^O$ we denote, that we run $\mathcal{A}$ with oracle access to O, that is it may query the oracle on inputs of its choice and only receives the corresponding outputs. We denote by $x \leftarrow\!\!\$\, S$ an output x being chosen uniformly at random from a set S. We denote the set $\{1, \ldots, n\}$ by $[n]$ and $x[i]$ denotes the i-th bit of x. PPT denotes probabilistic polynomial time. Also, $\mathsf{poly}(x), \mathsf{negl}(x)$ respectively denote any polynomial or negligible function in parameter x.

In the following, we define the cryptographic building blocks necessary for our protocol.

2.1 Symmetric Encryption Scheme.

A symmetric encryption scheme SKE is a tuple of three efficient algorithms $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ such that

- $K \leftarrow\!\!\$\ \mathsf{SKE.KeyGen}(1^\lambda)$: The probabilistic key generation algorithm on input the security parameter 1^λ outputs a key $K \in \{0,1\}^*$.
- $\mathsf{ct} \leftarrow\!\!\$\ \mathsf{SKE.Enc}(K, m)$: The probabilistic encryption algorithm takes as input the key K and a message $m \in \{0,1\}^*$ and outputs a ciphertext $\mathsf{ct} \in \{0,1\}^*$.
- $m \leftarrow \mathsf{SKE.Dec}(K, \mathsf{ct})$: The deterministic decryption algorithm takes as input the key K and a ciphertext ct and outputs a message m.

We require a symmetric encryption scheme to fulfill correctness, IND-CPA security and pseudorandom ciphertexts as defined below:

Definition 1 (Correctness). *We say that a symmetric encryption scheme is correct, if for all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^*$, we have*

$$\Pr\left[\mathsf{SKE.Dec}(K, \mathsf{SKE.Enc}(K, m)) = m \mid K \leftarrow\!\!\$\ \mathsf{SKE.KeyGen}(1^\lambda)\right] = 1.$$

Definition 2 (Security). *We say that a symmetric encryption scheme is IND-CPA secure, if no adversary $\mathcal{A}$ has more than negligible advantage in the following experiment $\mathsf{Exp}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}, 1^\lambda)$:*

- *The experiment samples $b \leftarrow\!\!\$\ \{0,1\}$ and $K \leftarrow\!\!\$\ \mathsf{SKE.KeyGen}(1^\lambda)$.*
- *$\mathcal{A}$ gets access to an encryption oracle $O(K, \cdot)$, which on input m outputs $\mathsf{Enc}(K, m)$, which it may use during the whole experiment.*
- *$\mathcal{A}$ chooses two challenge messages m_0, m_1 with $|m_0| = |m_1|$.*
- *The challenger sends $\mathsf{ct} = \mathsf{SKE.Enc}(K, m_b)$ to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a bit b'.*

The advantage of $\mathcal{A}$ is defined as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}CPA}} := \left|\Pr\left[\mathsf{Exp}_{\mathsf{IND\text{-}CPA}}(\mathcal{A}, 1^\lambda)\right] - \frac{1}{2}\right|.$$

2.2 Pseudo-Random Functions.

A keyed family of functions $\mathsf{PRF}_K : \{0,1\}^\mu \to \{0,1\}^\nu$ for keys $K \in \{0,1\}^*$ and some $\mu, \nu = \mathsf{poly}(|K|)$ is a pseudo-random function (PRF) family, if

- given K, m the function $\mathsf{PRF}_K(m)$ is efficiently computable and
- for every PPT distinguisher $\mathcal{D}$, it holds $\mathcal{D}^{\mathsf{PRF}_K(.)} \approx_c \mathcal{D}^{F(.)}$, where $K \leftarrow\!\!\$\ \{0,1\}^\lambda$ and F is chosen randomly from all functions from $\{0,1\}^{\mu(\lambda)}$ to $\{0,1\}^{\nu(\lambda)}$.

We also write $\mathsf{PRF}(K, m)$ for $\mathsf{PRF}_K(m)$.

2.3 Puncturable Pseudo-Random Functions.

A puncturable family of PRFs PPRF mapping strings of length $\mu(\cdot)$ to $\nu(\cdot)$ is given by a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Punc}, \mathsf{Eval})$, satisfying the following conditions:

Definition 3 (Pseudorandomness). *For every PPT distinguisher $\mathcal{D}$, it holds $\mathcal{D}^{\mathsf{Eval}(K,.)} \approx_c \mathcal{D}^{F(.)}$, where $K \leftarrow\!\!\$\ \mathsf{KeyGen}$ and F is chosen randomly from all functions from $\{0,1\}^{\mu(\lambda)}$ to $\{0,1\}^{\nu(\lambda)}$.*

Definition 4 (Functionality preserved under puncturing). *For every PPT adversary A such that $A\left(1^\lambda\right)$ outputs a set $S \subseteq \{0,1\}^{\mu(\lambda)}$, then for all $x \in \{0,1\}^{\mu(\lambda)}$ where $x \notin S$, we have that:*

$$\Pr\left[\mathsf{Eval}(K, x) = \mathsf{Eval}(K_S, x) : K \leftarrow \mathsf{KeyGen}(1^\lambda), K_S \leftarrow \mathsf{Punc}(K, S)\right] = 1$$

Definition 5 (Pseudorandom at punctured points). *For every PPT adversary* (A_1, A_2) *such that* $A_1(1^\lambda)$ *outputs a set* $S \subseteq \{0,1\}^{\mu(\lambda)}$ *and state* σ*, consider an experiment where* $K \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $K_S \leftarrow \mathsf{Punc}(K, S)$*. Then, we have*

$$\left|\Pr\left[A_2\left(\sigma, K_S, S, \mathsf{Eval}(K, S)\right) = 1\right] - \Pr\left[A_2\left(\sigma, K_S, S, U_{\nu(\lambda)\cdot|S|}\right) = 1\right]\right| = \mathsf{negl}(\lambda)$$

where $\mathrm{Eval}(K, S)$ *denotes the concatenation of* $(\mathsf{Eval}(K, x_1), \ldots, \mathsf{Eval}(K, x_k))$ *where* $S = \{x_1, \ldots, x_k\}$ *is the enumeration of the elements of* S *in lexicographic order, and* U_ℓ *denotes the uniform distribution over* ℓ *bits.*

In abuse of notation, we write $\mathsf{PPRF}(K, m)$ to denote $\mathsf{PPRF.Eval}(K, m)$. The construction given in [3] fulfills the following efficiency guarantees:

- A key punctured at a single point is of size $\mu \cdot \mathsf{poly}(\lambda)$.
- Evaluation of PPRF at a key punctured at most a single point runs in $\mathcal{O}(\mu \cdot \mathsf{poly}(\lambda))$.

2.4 Somewhere Statistically Binding Hashing.

Somewhere statistically binding (SSB) hashing was initially introduced by [24] and several constructions of SSB hashing followed. [31]

Definition 6. *A somewhere statistically binding (SSB) scheme* SSB *is composed of the following algorithms:*

- $\mathsf{hk} \leftarrow \mathsf{KeyGen}(1^\lambda, n, i)$ *takes as input the security parameter* λ*,* $n \in \mathbb{N}$ *and an index* $i \in [n]$*. It outputs a hashing key* hk*.*
- $h \leftarrow \mathsf{Hash}(\mathsf{hk}, D)$ *takes as input a hashing key* hk *and a database* $D = (x_i)_{i\in[n]}$*. It outputs a digest* h*.*
- $\tau \leftarrow \mathsf{Open}(\mathsf{hk}, D, i)$ *takes as input a hashing key* hk*, a database* $D = (x_i)_{i\in[n]}$ *and an index* i*. It outputs a proof* τ*.*
- $b \leftarrow \mathsf{Vrfy}(\mathsf{hk}, h, i, x, \tau)$ *takes as input a hashing key* hk*, a digest* h*, an index* $i \in [n]$*, a value* x *and a proof* τ*. It outputs a bit* b*.*

We require that an SSB hashing scheme fulfills the following efficiency guarantees:

1. The length ℓ_{hk} of the hashing key hk and the length ℓ_τ of proof τ are both of size $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$;
2. The Vrfy algorithm can be represented by a Turing machine of description size and runtime $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$.
3. The hash h is of size $\ell_h(\lambda) = \mathcal{O}(\mathsf{poly}(\lambda))$.

Additionally, an SSB hashing scheme fulfills the following properties.

Definition 7 (Correctness). *We say that an SSB hashing scheme is correct if for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* D *of size* n*, all indices* $j, i \in [n]$ *we have that*

$$\Pr\left[1 \leftarrow \mathsf{Vrfy}(\mathsf{hk}, h, i, x, \tau) : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{KeyGen}(1^\lambda, n, j) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, D) \\ \tau \leftarrow \mathsf{Open}(\mathsf{hk}, D, i) \end{array}\right] = 1.$$

Definition 8 (Somewhere statistically binding). *We say that an SSB hashing scheme is somewhere statistically binding if for all* $\lambda \in \mathbb{N}$*, all* $n = \mathsf{poly}(\lambda)$*, all databases* D *of size* n*, all indices* $i \in [n]$*, all database values* x *and all proofs* τ *we have that*

$$\Pr\left[D_i = x : \begin{array}{c} \mathsf{hk} \leftarrow \mathsf{KeyGen}(1^\lambda, n, i) \\ h \leftarrow \mathsf{Hash}(\mathsf{hk}, D) \\ 1 \leftarrow \mathsf{Vrfy}(\mathsf{hk}, h, i, x, \tau) \end{array}\right] = 1.$$

Definition 9 (Index Hiding). *We say that an SSB hashing scheme is index hiding if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have that*

$$\left| \Pr \left[b \leftarrow \mathcal{A}_2(\mathsf{hk}, aux) : \begin{array}{c} (n, i_0, i_1, aux) \leftarrow \mathcal{A}_1(1^\lambda) \\ b \leftarrow\!\!\$ \{0,1\} \\ \mathsf{hk} \leftarrow \mathsf{KeyGen}(1^\lambda, n, i_b) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

2.5 Secret Sharing.

We will introduce linear secret sharings and the well-known Shamir secret sharing [34].

Definition 10 (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 $(s_1, \ldots, s_n)$.*
- $m \leftarrow \mathsf{Reconstruct}(s_{i_1}, \ldots, s_{i_t})$ *takes as input t shares $(s_{i_1}, \ldots, s_{i_t})$. It outputs a message m.*

We expect an (t, n)-LSS to be correct and t-private.

Definition 11 (Correctness). *An LSS scheme* LSS *is said to be correct if for all messages m and all subsets $\{i_1, \ldots, i_t\} \subseteq [n]$*

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

Definition 12 (Privacy). *We say that a (t, n)-LSS scheme* LSS *is t-private if for all subsets $\{i_1, \ldots, i_z\} \subset [n]$ where $z < t$, all pairs of messages (m_0, m_1) and all PPT adversaries $\mathcal{A}$ we have that*

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

One such LSS scheme is the popular Shamir secret sharing [34]. Let $\mathbb{Z}_p$ be the finite field of prime order p and fix distinct elements $\xi = \xi_1, \ldots, \xi_n \in \mathbb{Z}_p$.

$\mathsf{Shamir.Share}(m)$ picks a random degree $t-1$ polynomial f with $f(0) = m$ and sets $s_i = f(\xi_i)$ for $i \in [n]$ as its shares.

To reconstruct, we use **Lagrange Interpolation**: For a set of supporting points $\chi_1, \ldots, \chi_k$ from a finite field $\mathbb{Z}_p$, where $p \in \mathbb{N}$ is prime, the Lagrange basis polynomials are given by $L_1, \ldots, L_k$, where

$$L_i(x) = \prod_{j \in [k]; j \neq i} \frac{x - \chi_j}{\chi_i - \chi_j}.$$

These are chosen such that $L_i(\chi_j) = 1$ iff $i = j$ and 0 otherwise. Consequently, given a set of k data points (ξ_i, y_i), we can output a polynomial $f_L(x) = \Sigma_{i \in [k]} L_i(x) \cdot y_i$ that will run through these points and which has degree at most k-1. This process is called Lagrange Interpolation. Hence, if $k > t - 1$, we will get back the original polynomial used in sharing and can evaluate $f(0) = m$.

2.6 Indistinguishability Obfuscation for Turing Machines.

Indistinguishability Obfuscation ($i\mathcal{O}$) [14] is a primitive to encode functionalities (usually represented as circuits) in a way such that the encodings of functionally equivalent circuits are indistinguishable.

In this work we need indistinguishability obfuscation for Turing Machines as constructed in [27,17]. This is defined in [17] as follows:

Definition 13 (Succinct Indistinguishability Obfuscator). *A succinct indistinguishability obfuscator for a machine class $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of a uniform PPT machine $i\mathcal{O}\mathsf{M}$ as follows:*

- $\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M\right)$*: $i\mathcal{O}\mathsf{M}$ takes as input the security parameter 1^λ, a description M of the Turing machine to obfuscate, and an input length n and time bound t for M.*
- *$i\mathcal{O}\mathsf{M}$ outputs a machine $\mathsf{ob}M$, which is an obfuscation of M corresponding to input length n and time bound t. $\mathsf{ob}M$ takes as input $x \in \{0,1\}^n$ and $t' \leq t$.*

The scheme should satisfy the following three requirements:

- ***Correctness:*** *For all security parameters $\lambda \in \mathbb{N}$, for all $M \in \mathcal{M}_\lambda$, for all inputs $x \in \{0,1\}^n$, time bounds t and $t' \leq t$, let y be the output of M on t' steps, then we have that:*

$$\Pr\left[\mathsf{ob}M\left(x, t'\right) = y : \mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M\right)\right] = 1$$

- ***Security:*** *For any (not necessarily uniform) PPT distinguisher D, there exists a negligible function α such that the following holds: For all security parameters $\lambda \in \mathbb{N}$, time bounds t, and pairs of machines $M_0, M_1 \in \mathcal{M}_\lambda$ of the same size such that for all running times $t' \leq t$ and for all inputs x, $M_0(x) = M_1(x)$ when M_0 and M_1 are executed for time t', we have that:*

$$\left|\Pr\left[D\left(i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M_0\right)\right) = 1\right] - \Pr\left[D\left(i\mathcal{O}\mathsf{M}\left(1^\lambda, 1^n, t, M_1\right)\right) = 1\right]\right| \leq \alpha(\lambda)$$

- ***Efficiency and Succinctness:*** *We require that the running time of $i\mathcal{O}\mathsf{M}$ and the length of its output, namely the obfuscated machine $\mathsf{ob}M$, is in $\mathcal{O}(\mathsf{poly}(|M|, \log t, n, \lambda))$. We also require that the obfuscated machine on input x and t' runs in time $\mathcal{O}(\mathsf{poly}\left(|M|, t', n, \log t, \lambda\right))$.*

3 Strongly Puncturable Signatures

Strongly puncturable signature (SPS) schemes are signature schemes with additional features and were introduced in [21] under the name of all-but-one signatures. In particular, such a scheme comes with a punctured key generation algorithm that, on input a message m^*, generates a pair of punctured keys $(\mathsf{vk}^*, \mathsf{sk}^*)$ at message m^*. A SPS scheme has to satisfy: (1) *puncturability*, meaning that given a punctured key $(\mathsf{vk}^*, \mathsf{sk}^*)$ w.r.t. a message m^* there *does not exist*, except with negligible probability, a valid signature for m^* w.r.t. the key vk^*; (2) *punctured-key indistinguishability* ensuring that punctured verification keys are indistinguishable from regular verification keys, as long as no signature on m^* is requested. Additionally, an SPS has to satisfy the usual correctness and EUF-CMA unforgeability properties of digital signatures.

We call this type of signature *strongly* puncturable to remark the difference with other notions of puncturable signatures [6,1,25] where a verifying signature for the punctured message m^* might exist, but it is infeasible to compute (given e.g. a punctured signing key).

In [21], various instantiations that can be based on different number-theoretic assumptions (e.g., RSA, pairing-based assumptions, LWE) were proposed. In Appendix A, we propose an alternative instantiation based on simulation-sound non-interactive zero-knowledge (NIZK) proofs and pseudorandom generators (PRGs).

Definition 14 (Strongly Puncturable Signature Scheme).
A strongly puncturable signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of the following algorithms:*

– $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$*: On input the security parameter* 1^λ*, the key generation algorithm* KeyGen *returns a verification key* vk *and a signing key* sk*.*

– $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKeyGen}(1^\lambda, m^*)$*: On input the security parameter* 1^λ*, a message* m^* *to puncture at, the punctured key generation algorithm* PKeyGen *returns a verification key* vk *and a signing key* sk*, that allows to sign any message except* m^**.*

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

– $d \leftarrow \mathsf{Vrfy}(\mathsf{vk}, \sigma, m)$*,* $d \in \{0,1\}$*: On input a verification key* vk*, a signature* σ *and a message* m*, the verification algorithm* Vrfy *returns a bit* $d \in \{0,1\}$*.*

A strongly puncturable signature scheme should have the following properties:

Definition 15 (Correctness). *For all* λ *and all messages* m *in the underlying message space*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1 \;\middle|\; (\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)\right] = 1.$$

Definition 16 (Puncturability). *For all* λ *and for all messages* m^**, we require*

$$\Pr\left[\exists \sigma \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}^*, \sigma, m^*) = 1 \;\middle|\; (\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow\!\!\$\ \mathsf{PKeyGen}(1^\lambda, m^*)\right] = \mathsf{negl}(\lambda).$$

Definition 17 (Punctured-Key Indistinguishability). *We say that a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *has* key indistinguishability*, if no PPT adversary* $\mathcal{A}$ *has more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$*. We define* $\mathcal{A}$*'s advantage by*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{IND\text{-}SPS}} := \left|\Pr[\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda) = 1] - \frac{1}{2}\right|.$$

Experiment $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$

1. *Adversary* $\mathcal{A}$ *provides a message* m^* *to puncture at.*
2. *The challenger picks a challenge bit* $b \leftarrow\!\!\$\ \{0,1\}$*. If* $b = 0$*, then* $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$*. If* $b = 1$*, then* $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKeyGen}(1^\lambda, m^*)$*. The verification key* vk *is returned to* $\mathcal{A}$*.*
3. $\mathcal{A}$ *gets oracle-access to* $\mathsf{Sign}_{m^*}(\mathsf{sk}, \cdot)$*. The oracle signs any message under the secret key* sk*, except* m^**.*
4. $\mathcal{A}$ *returns a guess bit* b' *and wins iff* $b' = b$*.*

Note that we can leverage punctured-key indistinguishability combined with a guessing argument to achieve standard EUF-CMA security. However, this incurs a security loss in the order of the message space. We will therefore independently require and prove EUF-CMA security of our alternative construction in Appendix A.

Definition 18. *We say that* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* unforgeable*, if for all PPT adversaries* $\mathcal{A}$ *we have* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{EUF\text{-}CMA}} = \mathsf{negl}(\lambda)$ *in the following experiment.*

Experiment $\mathsf{Exp}_{\mathsf{EUF\text{-}CMA}}(\mathcal{A}, 1^\lambda)$

1. *The challenger runs* KeyGen *to get* $(\mathsf{vk}, \mathsf{sk}) = \mathsf{KeyGen}(1^\lambda)$*. Set* $S \leftarrow \emptyset$*. The verification key* vk *is returned to* $\mathcal{A}$*.*

2. *$\mathcal{A}$ has oracle-access to* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*, to sign any message m of its choice. Upon each query for m, we set $S \leftarrow S \cup \{m\}$. Finally, $\mathcal{A}$ returns a message-signature pair (m', σ').*
3. *The experiment outputs* 1*, if* $\mathsf{Vrfy}(\mathsf{vk}, m', \sigma') = 1$ *and* $m' \notin S$*. Otherwise, it outputs* 0*.*

We define $\mathcal{A}$'s EUF-CMA advantage by

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

4 Compact Threshold SWE

In this section we introduce a compact t-of-n SWE scheme. This is a special purpose (threshold) witness encryption where we encrypt with regard to a set of signature verification keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ and a reference message T, such that decryption becomes available upon receiving t signatures $(\sigma_{i_j})_{j\in[t]}$ which verify for the reference message under t of the verification keys i.e. $\mathsf{Sig.Vrfy}(\mathsf{vk}_{i_j}, \sigma_{i_j}, T) = 1$. We say that such a scheme is compact if its ciphertext size grows polylogarithmically in the number of verification keys n, and we say it is secure if IND-CPA security holds in the absence of at least t such signatures.

4.1 Definition

Compared to the original proposal of SWE in [9], we have relaxed the definitions here in the following ways:

- The security is non-adaptive as discussed in Section 1.2.
- The underlying signature in this work needs to be puncturable and we take multiple signatures instead of one aggregated signature[2] as arguments.

Definition 19 (Compact Signature-Based Witness Encryption). *A t-out-of-n cSWE for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is a tuple of two algorithms* $(\mathsf{Enc}, \mathsf{Dec})$ *where:*

- $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n), T, m)$*: Encryption takes as input a security parameter λ, a set V of n verification keys of the underlying scheme* Sig*, a reference signing message T and a message m of arbitrary length $\ell \in \mathsf{poly}(\lambda)$. It outputs a ciphertext* ct*.*
- $m \leftarrow \mathsf{Dec}(\mathsf{ct}, (\sigma_i)_{i\in I}, I, V)$*: Decryption takes as input a ciphertext* ct*, a list of signatures $(\sigma_i)_{i\in I}$, an index set $I \subseteq [n]$ and a set V of verification keys of the underlying scheme* Sig*. It outputs a message m.*

We require such a scheme to fulfill three properties: correctness, compactness and security.

Definition 20 (Correctness). *A t-out-of-n cSWE scheme* $\mathsf{cSWE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* correct *if $\forall\lambda \in \mathbb{N}$, sets of keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, index sets $I \subseteq [n]$ with $|I| \geq t$, messages m, T and signatures $(\sigma_i)_{i\in I}$ with $\mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma_i, T) = 1$ for all $i \in I$, it holds $\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, T, m), (\sigma_i)_{i\in I}, I, V) = m$.*

Definition 21 (Compactness). *Given $\mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, V, T, m)$, its size $|\mathsf{ct}|$ is $\mathcal{O}(\mathsf{poly}(\lambda, \log n))$, where $n = |V|$.*

[2] The original paper uses BLS signatures, which allow to compress multiple signatures on different messages and from different signers into a single aggregated signature, that can be efficiently checked against all messages/signers in one step.

Definition 22 (Security). *A t-out-of-n cSWE scheme* $\mathsf{cSWE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a strongly puncturable signature scheme* $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* secure *if for all $\lambda \in \mathbb{N}$, all $t, n = \mathsf{poly}(\lambda), t < n$, there is no PPT adversary $\mathcal{A}$ that has more than negligible advantage in the experiment* $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$.

Experiment $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$

1. *The adversary $\mathcal{A}$ specifies signing reference message T^* and indices $J \subset [n]$ with $|J| \leq t-1$.*
2. *The experiment generates n key pairs for $i \in [n]$ as $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and provides $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ to $\mathcal{A}$, as well as all sk_i for $i \in J$.*
3. *$\mathcal{A}$ gets to make signing queries for pairs (i, T). If $i \in J$ or $T = T^*$, the experiment aborts, else it returns $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$.*
4. *$\mathcal{A}$ announces challenge messages m_0, m_1 with $|m_0| = |m_1|$.*
5. *The experiment flips a bit $b \leftarrow_\$ \{0,1\}$, and sends $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$ to $\mathcal{A}$.*
6. *$\mathcal{A}$ gets to make further signing queries for pairs (i, T). If $i \in J$ or $T = T^*$, the experiment aborts, else it returns $\mathsf{Sig.Sign}(\mathsf{sk}_i, T)$.*
7. *Finally, $\mathcal{A}$ outputs a guess b'.*
8. *If $b = b'$, the experiment outputs 1, else 0.*

We define $\mathcal{A}$'s advantage by $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{Sec}} := \left|\Pr\left[\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda) = 1\right] - \frac{1}{2}\right|$.

4.2 Construction

Our construction relies on indistinguishability obfuscation for Turing Machines and a Strongly Puncturable Signature scheme. The following scheme works for $n = \mathsf{poly}(\lambda)$ potential signers and requires t-of-n signatures to decrypt.

Given a strongly puncturable signature scheme Sig, our protocol will work for a slightly altered signature scheme Sig'. We define Sig' to behave exactly as Sig, but additionally the public keys vk output by $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Sig}'.\mathsf{KeyGen}(1^\lambda)$ have a random part $R_i \leftarrow_\$ \mathbb{Z}_p$ appended to vk_i, hence its keys are of the form (vk_i, R_i). Let $\ell_{vk} = |(\mathsf{vk}_i, R_i)|$ be the size of its keys, $\mathcal{M}$ its message space, and ℓ_σ be the length of its signatures.

Let further

- $p > 2^\lambda$ be a prime number.
- $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^\mu \to \mathbb{Z}_p$ be a PRF with $\log n \leq \mu \leq \log p$.
- $\mathsf{PPRF} = (\mathsf{KeyGen}, \mathsf{Punc}, \mathsf{Eval})$ with $\mathsf{Eval} : \{0,1\}^\lambda \times \{0,1\}^\mu \to \mathbb{Z}_p$ be a PPRF and ℓ_{pkey} be the size of any key punctured at most one point.
- SKE be a symmetric key encryption scheme
- SSB be an SSB-hashing scheme and $\ell_{\mathsf{hk}}, \ell_\tau, \ell_h$ be the length of the hashing keys hk, proofs τ and hashes h of SSB on input a database D with length n and with a maximum size of its entries of ℓ_{vk}.
- $\mathsf{ob}M$ be an indistinguishability obfuscator for the class of machines $\mathcal{TM}_\lambda = \{M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')\}$ where each such machine takes: indices $i^*, \mathsf{ind}_2 \in \{0, \ldots, n+1\}$, values $K, K_1 \in \{0,1\}^\lambda$, $h \in \{0,1\}^{\ell_h}, T \in \mathcal{M}, \mathsf{hk} \in \{0,1\}^{\ell_{hk}}$, a PPRF key $K_2 \in \{0,1\}^\ell_{pkey}$ and a value $U \in \mathbb{Z}_p$ as hardwired inputs and an index $\mathsf{ind} \in [n]$, values vk, R of combined length ℓ_{vk}, as well as $\tau \in \{0,1\}^{\ell_\tau}$ and $\sigma \in \{0,1\}^{\ell_\sigma}$ as run-time inputs.
  $M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$ is defined as:

- If $\mathsf{ind} < 1$ or $\mathsf{ind} > n$, abort and output $\perp$.
- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig.Vrfy}(\mathsf{vk}, \sigma', T)$, abort and output $\perp$.
- If $\mathsf{ind} = \mathsf{ind}_2$: Output $R + U \mod p$ and halt.
- If $\mathsf{ind} \leq i^*$: Output $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ and halt.
- Else if $\mathsf{ind} > i^*$:
  - Write on the tape variables $i = 1, R = 0, X = 1, S = K$.
  - While $i < t$: Set $R = \mathsf{PRF}(K_1, i)$, $X = X \cdot \mathsf{ind} \mod p$, $S = S + X \cdot R \mod p$, $i = i + 1$.
  - Output S.

Note that all these machines have the same description size. Let ℓ_{inp} be shorthand for its input size and $T_{\mathcal{TM}}$ denote the maximum runtime within this machine class. It is syntactically clear that these machines always halt, so $T_{\mathcal{TM}}$ is well-defined.

We will now present the construction of our t-of-n compact SWE scheme.

Protocol Compact cSWE for Sig' - from iO

$\mathsf{cSWE.Enc}(1^\lambda, V = (\mathsf{vk}_i, R_i)_{i\in[n]}, T, m)$:
- Generate PRF key $K_1 \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
- Generate symmetric key $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$
- Generate PPRF key $K_2 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$
- Generate a hashing key $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, 1)$
- Set $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i\in[n]})$
- Choose $U \leftarrow_\$ \mathbb{Z}_p$.
- Let $M' = M_0[K, K_1, h, T, \mathsf{hk}, K_2, U, 0](\cdot, (\cdot, \cdot), \cdot, \cdot)$.
- Compute $\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}(1^\lambda, M', \ell_{inp}, T_{\mathcal{TM}})$.
- Compute $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$
- Output $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$

$\mathsf{cSWE.Dec}(\mathsf{ct}, (\sigma), I, V = (\mathsf{vk}_i, R_i)_{i\in[n]})$:
- Parse $\mathsf{ct} = (\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$
- If $|I| < t$, abort.
- Parse $\sigma = (\sigma_i)_{i\in I}$.
- For $i \in I$:
  - Compute $\tau_i = \mathsf{Open}(\mathsf{hk}, V, i)$
  - Run $c_i = \mathsf{ob}M(i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$
  - Notice $c_i = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot i^j$ i.e. evaluations $S_i = f(i)$ on polynomial $f = K + \Sigma_{j=1}^{t-1}\mathsf{PRF}(K_1, j) \cdot x^j$.
- Compute $K' = \sum_{i\in I} c_i \cdot L_i$ where $L_i = \prod_{j\in I; j\neq i} \frac{-j}{i-j}$.
- Output $m = \mathsf{SKE.Dec}(K', \mathsf{ct}')$.

We point out that by encrypting a symmetric encryption key, we can get an encryption scheme that allows messages m of arbitrary length without impacting the size of $\mathsf{ob}M$.

4.3 Proofs

Theorem 1. *Our construction* cSWE *is correct, given that the underlying primitives* Sig', SKE, SSB *and* $\mathsf{ob}M$ *are correct.*

Proof. Let $\lambda \in \mathbb{N}$, a set of keys $V = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$, an index set $I \subseteq [n]$ with $|I| \geq t$, messages m, T and signatures $(\sigma_i)_{i \in I}$ be given such that for all $i \in I$, we have $\mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma_i, T) = 1$. Let us show $\mathsf{Dec}(\mathsf{Enc}(1^\lambda, V, T, m), (\sigma_i)_{i \in I}, I, V) = m$. $\mathsf{Enc}(1^\lambda, V, T, m)$ yields an output $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$. In $\mathsf{Dec}((\mathsf{ob}M, \mathsf{ct}', \mathsf{hk}), (\sigma_i)_{i \in I}, I, V)$, we do not abort before calling $\mathsf{ob}M$, as $|I| \geq t$ by requirement. We compute for $i \in I$: $\tau_i = \mathsf{Open}(\mathsf{hk}, V, i)$, $c_i = \mathsf{ob}M(i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$. By correctness of $\mathsf{ob}M$, this outputs the value generated by $M_0[K, K_1, h, T, \mathsf{hk}, K_2, U, 0](i, (\mathsf{vk}_i, R_i), \tau, \sigma_i)$, which runs the following code:

- If $i < 1$ or $i > n$, abort and output $\perp$.
- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{vk}_i, R_i), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma_i, T)$, abort and output $\perp$.
- If $i = 0$: Output $R + U \mod p$ and halt.
- If $i \leq 0$: Output $R + \mathsf{PPRF}(K_2, i) \mod p$ and halt.
- Else if $i > 0$:
  - Write on the tape variables $j = 1, R = 0, X = 1, S = K$
  - While $j < t$:
    - Set $R = \mathsf{PRF}(K_1, j)$, $X = X \cdot i \mod p$, $S = S + X \cdot R \mod p$, $j = j + 1$.
  - Output S

And has hardcoded values $K_1 \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$, $K \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$, $K_2 \leftarrow \mathsf{PPRF.KeyGen}(1^\lambda)$, $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, 1)$, $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i \in [n]})$.

Clearly $1 \leq i \leq n$. $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, i, (\mathsf{vk}_i, R_i), \tau) = 1$ holds by correctness of SSB since hk, V in Dec are the same as in the call to Enc. $\mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma_i, T)$ holds by definition.

This means we get for $i \in I$ $c_i = K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot i^j$ i.e. evaluations $c_i = f(i)$ of a polynomial $f = K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot x^j$ with $f(0) = K$.

By computing $K' = \sum_{i \in I} c_i \cdot L_i$ with $L_i = \prod_{j \in I; j \neq i} \frac{-j}{i-j}$ being the Langrange polynomials for supporting points $i \in I$ evaluated at 0, we are guaranteed $K' = f(0) = K$.

So Dec finally outputs $\mathsf{SKE.Dec}(K, \mathsf{ct}')$, but since $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$, by the correctness of SKE, we output the original message m. □

Theorem 2. *Our construction* cSWE *is compact.*

Proof. The bottleneck of our ciphertext size is the size of the obfuscated Turing machine, which depends polynomially on its description size and polylogarithmically on its runtime. We notice, that we can describe all inputs (runtime as well as hardwired ones) in $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$. All operations (comparisons, modular arithmetic in $\mathbb{Z}_p$, SSB verification, signature verification and (P)PRF evaluations) can be described and evaluated in size/time $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ and we can describe the whole code including the while-loop in $\mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$, while the maximum runtime is in $\mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda))$, leading to the desired size.

In more detail, the output of $\mathsf{cSWE}'.\mathsf{Enc}(1^\lambda, V = (\mathsf{vk}_i, R_i)_{i \in [n]}, T, m)$ is $(\mathsf{ob}M, \mathsf{ct}', \mathsf{hk})$, where $\mathsf{hk} = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i \in [n]})$, $|\mathsf{hk}| = \ell_{hk} = \mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by the efficiency guarantees on SSB.

And $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m)$, so $|\mathsf{ct}'| = \mathcal{O}(\mathsf{poly}(\lambda) \cdot |m|) = \mathcal{O}(\mathsf{poly}(\lambda))$. It remains to show that $\mathsf{ob}M$ is in $\mathcal{O}(\mathsf{poly}(\lambda, \log n))$.

$\mathsf{ob}M \leftarrow i\mathcal{O}\mathsf{M}(1^\lambda, M', \ell_{inp}, T_{\mathcal{TM}})$ with $M' = M_{i^*=0}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2 = 0]$ By the efficiency requirements on $i\mathcal{O}\mathsf{M}$, $|\mathsf{ob}M|$ is in $\mathcal{O}(\mathsf{poly}(|M'|, \log T_{\mathcal{TM}}, \ell_{inp}, \lambda))$, where $|M'|$ is the description size, $T_{\mathcal{TM}}$ the maximum runtime and ℓ_{inp} the maximum input size of machines in $\mathcal{TM}_\lambda$. The machines in this class are defined as
$M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$:

- If $\mathsf{ind} < 1$ or $\mathsf{ind} > n$, abort and output $\perp$.

– If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$, abort and output $\perp$.
– If $0 = \mathsf{Sig.Vrfy}(\mathsf{vk}, \sigma', T)$, abort and output $\perp$.
– If $\mathsf{ind} = \mathsf{ind}_2$: Output $R + U \mod p$ and halt.
– If $\mathsf{ind} \leq i^*$: Output $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ and halt.
– Else if $\mathsf{ind} > i^*$:
  • Write on the tape variables $i = 1, R = 0, X = 1, S = K$
  • While $i < t$: Set $R = \mathsf{PRF}(K_1, i)$, $X = X \cdot \mathsf{ind} \mod p$, $S = S + X \cdot R \mod p$, $i = i + 1$.
  • Output S

By construction, the sizes of hardwired inputs are as follows: $|i^*|, |\mathsf{ind}_2| = \log n$, $|K|, |K_1| = \lambda$, $|h| = \ell_h = \mathcal{O}(\mathsf{poly}(\lambda))$ by efficiency of SSB, $|T| = \mathcal{O}(\mathsf{poly}(\lambda))$ as $T \in \mathcal{M}$ is from the message space of Sig with security parameter λ, $|\mathsf{hk}| = \ell_{hk} = \mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by the efficiency guarantees on SSB, $|K_2| = \ell_{pkey} = \mu \cdot \mathsf{poly}(\lambda)$ by the efficiency guarantees of PPRF and $|U| = \log p$.

The **runtime inputs** are bounded in size as follows: $\mathsf{ind} = \log n$, $|(\mathsf{vk}, R)| = \ell_{\mathsf{vk}} = \lambda + \log p$, $|\tau| = \ell_\tau = \mathcal{O}(\mathsf{poly}(\lambda)) \cdot \log n$ by efficiency of SSB and $|\sigma| = \ell_\sigma = \mathsf{poly}(\lambda)$.

We can bound $\mu, \log p = \mathcal{O}(\mathsf{poly}(\log n, \lambda))$, as we only required $\log p > \lambda, \log p \geq \mu \geq \log n$. So, we see directly that the input size is bounded as

$$\ell_{inp} = \mathcal{O}(\mathsf{poly}(\lambda \cdot \log n)).$$

Towards analysing the **maximum runtime** $T_{\mathcal{TM}}$: Comparisons with ind, i^* can be executed in time $\mathcal{O}(\mathsf{poly}(\log n))$. The Turing machine code for $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$ has runtime $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by efficiency of SSB. The code for $\mathsf{Sig.Vrfy}(\mathsf{vk}, \sigma', T)$ has runtime $\mathcal{O}(\mathsf{poly}(\lambda))$. Modular arithmetic in $\mathbb{Z}_p$ can be executed in time $\mathsf{poly}(\log p) = \mathcal{O}(\mathsf{poly}(\lambda, \log n))$. $\mathsf{PPRF}(K_2, \mathsf{ind})$ can be executed in $\mathcal{O}(\mu \cdot \mathsf{poly}(\lambda)) = \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$ by our efficiency requirements on PPRF and so can $\mathsf{PRF}(K_1, i)$. Since $t \leq n$, the while-loop has runtime in $\mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda))$. So the maximum runtime is bounded by

$$T_{\mathcal{TM}} = \mathcal{O}(n \cdot \mathsf{poly}(\log n \cdot \lambda)).$$

Towards analysing the **description size** $|M'|$: Comparisons with ind, i^* can be described in size $\mathcal{O}(\mathsf{poly}(\log n))$. The Turing machine code for $\mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$ has size $\mathcal{O}(\mathsf{poly}(\lambda) \cdot \log n)$ by efficiency of SSB. The code for $\mathsf{Sig.Vrfy}(\mathsf{vk}, \sigma', T)$ has size $\mathcal{O}(\mathsf{poly}(\lambda))$. Modular arithmetic in $\mathbb{Z}_p$ can be described in size $\mathsf{poly}(\log p) = \mathcal{O}(\mathsf{poly}(\lambda, \log n))$. $\mathsf{PPRF}(K_2, \mathsf{ind})$ can be described in $\mathcal{O}(\mu \cdot \mathsf{poly}(\lambda)) = \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$ by our efficiency requirements on PPRF and so can $\mathsf{PRF}(K_1, i)$. Since $\log t \leq \log n$, the while-loop can be described in size $\log n + \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda))$. We conclude that the description size is bounded by

$$|M'| = \mathcal{O}(\mathsf{poly}(\log n \cdot \lambda)).$$

Thus, $|\mathsf{ob}M| = \mathcal{O}(\mathsf{poly}(|M'|, \log T_{\mathcal{TM}}, \ell_{inp}, \lambda)) = \mathcal{O}(\mathsf{poly}(\mathsf{poly}(\log n \cdot \lambda), \log(n \cdot \mathsf{poly}(\log n \cdot \lambda)), \mathsf{poly}(\lambda \cdot \log n), \lambda)) = \mathcal{O}(\mathsf{poly}(\log n, \lambda))$.

So the whole ciphertext is in size $\mathcal{O}(\mathsf{poly}(\log n, \lambda))$. □

Theorem 3. *Our construction of* cSWE *is secure, given that* Sig *is punctured key indistinguishable and puncturable,* SSB *is index hiding and somewhere perfectly binding,* $\mathsf{ob}M$ *is secure,* SKE *is correct, IND-CPA secure and has pseudorandom ciphertexts,* PRF *is a pseudorandom function and* PPRF *is a puncturable pseudorandom function.*

Proof. Let us give a proof sketch first:

We define a series of indistinguishable hybrids and show, that an adversary $\mathcal{A}$ with non-negligible advantage in the last hybrid could break IND-CPA security of SKE.

Our strategy is to puncture all honest keys at message T^*, then gradually move each party's shares from being computed inside the Turing machine into being embedded in the key part R_i in encrypted form. The Turing machine should in the end only have to decrypt the shares from its input and can forget the key K that it was supposed to share. Then, we can observe, that it never decrypts the honest parties' shares, as there is no accepting signature due to puncturability of our signing scheme. We bind the SSB hash to each honest position i^*, puncture the PPRF at i^* to forget the decryption key, hardwire the output share s_{i^*} into the machine instead and replace the share s_{i^*} with an encryption of garbage inside R_{i^*}. Then, we observe, that by SSB binding and puncturability, no input exists anymore for which s_{i^*} is output and we can forget the hardwired value again. Lastly, we notice that we now need less than t shares of K to run this experiment and can therefore replace them with shares of an unrelated key K' by t-privacy. What remains is a single encryption of m under K, where K is not known or used in any other part of the output.

Full proof: Let $\lambda \in \mathbb{N}$, such that $t = \mathsf{poly}(\lambda)$. Let $\mathcal{A}$ be an adversary for $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$ with more than negligible advantage.

$\mathcal{H}_0 = \mathcal{H}_1^0$: This game is identical to $\mathsf{Exp}_{\mathsf{Sec}}(\mathcal{A}, 1^\lambda)$.

To recap, we get indices $J \subset [n]$, $|J| \leq t-1$ and a reference message T^* from $\mathcal{A}$, then the experiment generates key pairs for $i \in [n]$ as $(\mathsf{vk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $R_i \leftarrow \mathbb{Z}_p$ and provides $V = ((\mathsf{vk}_1, R_1), \ldots, (\mathsf{vk}_n, R_n))$ to $\mathcal{A}$, as well as all sk_i for $i \in J$. We allow any signing queries for honest keys $\mathsf{vk}_i, i \in [n] \setminus J$ on any message except T^*, before and after the adversary announces challenge messages m_0, m_1 and the experiment responds to this challenge by choosing $b \leftarrow\!\!\$\ \{0,1\}$, and sending $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$ to $\mathcal{A}$.

In the end, $\mathcal{A}$ outputs a guess b' to our choice bit b.

$\mathcal{H}_1^{i^*}$ for $i^* \in \{1, \ldots, n\}$:
- If $i^* \in J$, this is identical to $\mathcal{H}_1^{i^*-1}$.
- Else, it is identical to $\mathcal{H}_1^{i^*-1}$, except that the key vk_{i^*} is created as a punctured key as $(\mathsf{vk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow\!\!\$\ \mathsf{Sig.PKeyGen}(1^\lambda, T^*)$ for the signing reference messsage T^* specified by $\mathcal{A}$ instead of $(\mathsf{vk}_{i^*}, \mathsf{sk}_{i^*}) \leftarrow\!\!\$\ \mathsf{Sig.KeyGen}(1^\lambda)$.

We observe that our experiment never needs to compute signatures of T^* for honest keys vk_i with $i \notin J$. So clearly, $\mathcal{H}_1^{i^*}$ and $\mathcal{H}_1^{i^*-1}$ are indistinguishable by the punctured-key indistinguishability of Sig'. In the last hybrid $\mathcal{H}_1^n$, all honest keys will be chosen punctured at message T^*.

We define an event

$$\mathbf{bad} = \{\exists \sigma, i \in [n] \setminus J \text{ such that } \mathsf{Sig.Vrfy}(\mathsf{vk}_i, \sigma, T^*) = 1\}.$$

Note that by puncturability of Sig, it holds

$$\Pr[\mathbf{bad}] \leq \Sigma_{i \in [n] \setminus J} \Pr\left[\begin{array}{c} \exists \sigma \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}^*, \sigma, T^*) = 1 : \\ (\mathsf{vk}^*, \mathsf{sk}^*) \leftarrow\!\!\$\ \mathsf{PKeyGen}(1^\lambda, T^*) \end{array}\right] \leq n \cdot \mathsf{negl}(\lambda),$$

which is negligble. In the following we condition on **bad** not happening.

$\mathcal{H}_2$: This is identical to $\mathcal{H}_1^n$ except for conceptual changes:
- The reduction computes already at the start of the experiment the keys K, K_1, K_2 it will use in the call to $\mathsf{Enc}(1^\lambda, V, T^*, m_b)$.
- We define the polynomial $f' = K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot x^j$ and $s_i = f'(i)$ for $i \in [n]$. Note that s_{ind} corresponds to the output of M' on an accepting input $(\mathsf{ind}, (\mathsf{vk}_{\mathsf{ind}}, R_{\mathsf{ind}}), \tau, \sigma')$, where $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}_{\mathsf{ind}}, R_{\mathsf{ind}}), \tau)$ and $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}_{\mathsf{ind}}, \sigma', T)$ for $\mathsf{ind} \in [n]$. The shares s_i for $i \in J$ correspond to the outputs the adversary is guaranteed to get by just signing T^* himself and following the decryption procedure.

We will now loop through $\mathcal{H}_3^i, \mathcal{H}_4^i, \ldots, \mathcal{H}_7^i$ to gradually replace all honest verification keys by punctured ones and switch to a setting where the obfuscated machine pulls all of its outputs out of the R_i key parts instead of computing them itself. We start with $\mathcal{H}_7^0 = \mathcal{H}_2$ for notational convenience.

$\mathcal{H}_3^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_7^{i^*-1}$ (with $\mathcal{H}_7^0 = \mathcal{H}_2$) except that we generate $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ binding to i^*.

This is clearly indistinguishable from $\mathcal{H}_7^{i^*-1}$ by SSB being index hiding.

$\mathcal{H}_4^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_3^{i*}$ except in the way we choose R_{i^*} in vk_{i^*}. We sample $u_{i^*} \leftarrow_\$ \mathbb{Z}_p$ and set $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ with s_{i^*} defined as above instead of $R_{i^*} \leftarrow \mathbb{Z}_p$.

This is indistinguishable from $\mathcal{H}_3^{i*}$ as $s_{i^*} - u_{i^*}$ is still uniform in $\mathbb{Z}_p$.

$\mathcal{H}_5^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_4^{i*}$ except we generate $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$ the machine M' we use is replaced by $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ instead of $M' = M_{i^*-1}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$.

This is indistinguishable from $\mathcal{H}_4^{i*}$ by security of $\mathsf{ob}M$. To show this, let us argue, that $M_0 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ and $M_1 \coloneqq M_{i^*-1}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ are functionally equivalent.
We recall that $M_{i^*}[K, K_1, h, T, \mathsf{hk}, K_2, U, \mathsf{ind}_2](\mathsf{ind}, (\mathsf{vk}, R), \tau, \sigma')$ is defined as follows:

- If $0 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}, (\mathsf{vk}, R), \tau)$, abort and output $\perp$.
- If $0 = \mathsf{Sig.Vrfy}(\mathsf{vk}, \sigma', T)$, abort and output $\perp$.
- If $\mathsf{ind} = \mathsf{ind}_2$: Output $R + U \mod p$ and halt.
- If $\mathsf{ind} \leq i^*$: Output $R + \mathsf{PPRF}(K_2, \mathsf{ind}) \mod p$ and halt.
- Else if $\mathsf{ind} > i^*$:
  - Write on the tape variables $i = 1, R = 0, X = 1, S = K$
  - While $i < t$:
    - Set $R = \mathsf{PRF}(K_1, i)$, $X = X \cdot \mathsf{ind} \mod p$, $S = S + X \cdot R \mod p$, $i = i + 1$.
  - Output S (Note that $S = f'(\mathsf{ind}) = s_{\mathsf{ind}}$.)

Assuming that there is an input $(\mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*)$, $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$, $0 < \mathsf{ind} \leq n$ must hold. Otherwise they both output $\perp$.

We distinguish 3 cases:

$\mathsf{ind}^* > i^*$ In this case both machines output s^*_{ind}.

$\mathsf{ind}^* = i^*$ In this case M_1 outputs s^*_{ind}, but M_0 outputs $R^* + u_{i^*} \mod p$.
We note that $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ and $h = \mathsf{SSB.Hash}(\mathsf{hk}, (\mathsf{vk}_i, R_i)_{i \in [n]})$ are honestly created by the reduction and so since SSB is somewhere statistically binding, we know that $\mathsf{vk}^* = \mathsf{vk}_{i^*}, R^* = R_{i^*}$.
That means that M_0 outputs $R_{i^*} + u_{i^*} = s_{i^*}$, which is the same output as in M_1.

$\mathsf{ind}^* < i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}}, \mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_2, \mathsf{ind}^*) \mod p$.
As $\mathsf{ind} \neq i^*$ and $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$, we can conclude that these outputs are identical by the PPRF preserving functionality under puncturing.

It follows that there does not exist an input that makes M_0 and M_1 have a different output, thus $\mathcal{H}_5^{i^*}$ and $\mathcal{H}_4^{i^*}$ are indistinguishable.

$\mathcal{H}_6^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_5^{i*}$ except that we now choose $R_{i^*} = s_{i^*} - \mathsf{PPRF}(K_2, i^*) \mod p$ instead of $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ and $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$ instead of $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$

$\mathcal{H}_6^{i^*}$ and $\mathcal{H}_5^{i^*}$ are indistinguishable as PPRF is pseudorandom at the punctured point i^*. Given $K_{\{i^*\}}, i^*$ and either a uniform value $\bar{u} \leftarrow \mathbb{Z}_p$ or $\mathsf{PPRF}(K_2, i^*)$, we can obliviously emulate either $\mathcal{H}_5^{i*}$ or $\mathcal{H}_6^{i*}$ to build a distinguisher for pseudorandomness at punctured points of PPRF.

$\mathcal{H}_7^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_6^{i*}$ except that we now choose $M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ instead of
$M' = M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$

This is indistinguishable from $\mathcal{H}_6^{i*}$ by security of $\mathsf{ob}M$.

To show this, let us argue, that $M_0 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_2, U, 0]$ and $M_1 \coloneqq M_{i^*}[K, K_1, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$ are functionally equivalent.

Assuming that there is an input $(\mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*)$, $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$, $0 < \mathsf{ind} \leq n$ must hold.

We distinguish 3 cases:

$\mathsf{ind}^* > i^*$ In this case both machines output s^*_{ind}.

$\mathsf{ind}^* = i^*$ In this case both machines output $R^* + \mathsf{PPRF}(K_2, i^*) \mod p$.

$\mathsf{ind}^* < i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_2, \mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}}, \mathsf{ind}^*) \mod p$.
As $\mathsf{ind} \neq i^*$ and $K_{\{i^*\}} \leftarrow \mathsf{PPRF.Punc}(K_2, \{i^*\})$, we can conclude that these outputs are identical by the PPRF preserving functionality under puncturing.

This means such a resulting in different outputs can not exist - $\mathcal{H}_6^{i^*}$ and $\mathcal{H}_7^{i^*}$ are indeed indistinguishable.

$\mathcal{H}_8$: This is identical to $\mathcal{H}_7^n$, except that we use $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_2, U, 0]$ which is the same as the previous machine, except of it having dummy inputs instead of K, K_1.

As the output of M' does no longer depend on K, K_1 in $\mathcal{H}_7^n$, this is indistinguishable from the previous hybrid by security of $\mathsf{ob}M$.

$\mathcal{H}_9$ =: This is identical to $\mathcal{H}_8$ except that we set the $s_i^* = (K + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j))$ in $R_i = s_i^* - \mathsf{PPRF}(K_2, i) \mod p$ for $i \in [n]$, where we pick $r_j \leftarrow \{0,1\}^\mu$ randomly instead of $s_i^* = (K + \Sigma_{j=1}^{t-1} \mathsf{PRF}(K_1, j) \cdot (i)^j))$.

This is indistinguishable from $\mathcal{H}_8$ by pseudorandomness of PRF.

Now we will almost "revert" the moves in $\mathcal{H}_3$ to $\mathcal{H}_7$ again to put random values into R_i for $i \notin J$, but this time we do not let M' keep the information to decrypt them - we rely on the puncturability of Sig to make sure that the case of decryptions would never have been reached anyways and invoke security of $\mathsf{ob}M$. This deletes all traces of honest shares in the R_i.

$\mathcal{H}_{10}^{i^*}$ for $i^* \in \{1, \ldots, n\}$: This is identical to $\mathcal{H}_{13}^{i^*-1}$ (with $\mathcal{H}_{13}^0 = \mathcal{H}_9$) except that we generate $\mathsf{hk} \leftarrow \mathsf{SSB.KeyGen}(1^\lambda, n, i^*)$ binding to i^*.

This is clearly indistinguishable from $\mathcal{H}_{13}^{i^*-1}$ by SSB being index hiding.

$\mathcal{H}_{11}^{i^*}$ for $i^* \in \{1, \ldots, n\}$: If $i^* \in J$, this is identical to $\mathcal{H}_{10}^{i*}$.
Else, this is identical to $\mathcal{H}_{10}^{i*}$ except that the machine M' we use is replaced by $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$ instead of $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_2, U, 0]$.

This is indistinguishable from $\mathcal{H}_{10}^{i^*}$ by security of $\mathsf{ob}M$ and the argument is analogous to $\mathcal{H}_7^{i^*}$ and $\mathcal{H}_6^{i^*}$ being indistinguishable.

$\mathcal{H}_{12}^{i^*}$ for $i^* \in \{1, \ldots, n\}$: If $i^* \in J$, this is identical to $\mathcal{H}_{11}^{i*}$.
Else this is identical to $\mathcal{H}_{11}^{i*}$ except that we now choose $R_{i^*} = s_{i^*} - u_{i^*} \mod p$ instead of $R_{i^*} = s_{i^*} - \mathsf{PPRF}(K_2, i^*) \mod p$ and $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ instead of $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_{\{i^*\}}, \mathsf{PPRF}(K_2, i^*), i^*]$.

$\mathcal{H}_{12}^{i^*}$ and $\mathcal{H}_{11}^{i^*}$ are indistinguishable by PPRF being pseudorandom at punctured points and the argument is analogous to $\mathcal{H}_6^{i^*}$ and $\mathcal{H}_5^{i^*}$ being indistinguishable.

$\mathcal{H}_{13}^{i^*}$ for $i^* \in \{1, \ldots, n\}$: If $i^* \in J$, this is identical to $\mathcal{H}_{12}^{i*}$.
This is identical to $\mathcal{H}_{12}^{i*}$ except that we now choose $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_2, U, 0]$ instead of $M' = M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$.

This is indistinguishable from $\mathcal{H}_{12}^{i*}$ by security of $\mathsf{ob}M$.

To show this, let us argue, that $M_0 \coloneqq M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_2, U, 0]$ and $M_1 \coloneqq M_n[\emptyset, \emptyset, h, T^*, \mathsf{hk}, K_{\{i^*\}}, u_{i^*}, i^*]$ are functionally equivalent.

Assuming that there is an input $(\mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ for which M_0, M_1 produce different outputs, then $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, \mathsf{ind}^*, (\mathsf{vk}^*, R^*), \tau^*)$, $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$, $0 < \mathsf{ind} \leq n$ must hold.

We distinguish 2 cases:

$\mathsf{ind}^* \neq i^*$ In this case M_0 outputs $R^* + \mathsf{PPRF}(K_2, \mathsf{ind}^*) \mod p$, while M_1 outputs $R^* + \mathsf{PPRF}(K_{\{i^*\}}, \mathsf{ind}^*) \mod p$. We can conclude that these outputs are identical by the PPRF preserving functionality under puncturing.

$\mathsf{ind}^* = i^*$ We claim, that no input $(i^*, (\mathsf{vk}^*, R^*), \tau^*, \sigma^*)$ exists where $1 = \mathsf{SSB.Vrfy}(\mathsf{hk}, h, i^*, (\mathsf{vk}^*, R^*), \tau^*)$ and $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}^*, \sigma^*, T^*)$ hold, hence the output is always $\perp$ in both machines.
Let us assume towards contradiction, that such an input exists. By SSB being somewhere statistically binding, we know $\mathsf{vk}^* = \mathsf{vk}_{i^*}$, $R^* = R_{i^*}$. So, it must hold $1 = \mathsf{Sig.Vrfy}(\mathsf{vk}_{i^*}, \sigma^*, T^*)$ for $i^* \in J$ - As the event **bad** has not happened, no such σ^* can exist.

This means such an input with differing output can not exist - $\mathcal{H}_{12}^{i^*}$ and $\mathcal{H}_{13}^{i^*}$ are indeed indistinguishable.

$\mathcal{H}_{14}$ This is identical to $\mathcal{H}_{13}^n$ except that we now compute $R_i \leftarrow\!\!\$\ \mathbb{Z}_p$ for all $i \in [n] \setminus J$ instead of $R_i = s_i - u_i \mod p$ for $u_i \leftarrow\!\!\$\ \mathbb{Z}_p$ being a fresh uniform value for each $i \in [n] \setminus J$.

As the u_{i^*} are not re-used anywhere in the experiment, this is indistinguishable from $\mathcal{H}_{13}^n$ as both distributions lead to uniform R_i for all honest indices $i \in [n] \setminus J$.

It is clear now, that we only ever compute $|J| \leq t - 1$ shares of the key K, which are of the from $R_i = K + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j + \mathsf{PPRF}(K_2, i) \mod p$. This clearly corresponds to $|J|$ Shamir shares corresponding to interpolation points $i \in J$ out of a t-of-n sharing for points $i \in [n]$. So in the next hybrid, we can make the key K disappear.

$\mathcal{H}_{15}$: This is identical to $\mathcal{H}_{14}$, except that we choose a random $K' \leftarrow \{0,1\}^\lambda$ and make $R_i = K' + \Sigma_{j=1}^{t-1} r_j \cdot (i)^j + \mathsf{PPRF}(K_2, i) \mod p$ for $i \in J$. Where $r_j \leftarrow \mathbb{Z}_p$ randomly.

This is indistinguishable from $\mathcal{H}_{14}$ by the t-privacy of the Shamir secret sharing scheme.

Let us assume now that there is an adversary $\mathcal{A}$ which breaks security of cSWE with probability ε. Conditioned on **bad** not happening, $\mathcal{A}$ has negligible advantage $n'(\lambda)$ of distinguishing the real experiment from $\mathcal{H}_{15}$.

In hybrid $\mathcal{H}_{15}$, there is no information about K contained anywhere except in the ciphertext $\mathsf{ct}' = \mathsf{SKE.Enc}(K, m_b)$. So we can make an IND-CPA distinguisher $\mathcal{D}$, that simulates $\mathcal{H}_{15}$ by asking the IND-CPA security game of SKE for an encryption with challenge messages m_0, m_1, receiving a ciphertext ct^* and then making all the outputs in $\mathcal{H}_{15}$ honestly, except setting ciphertext part $\mathsf{ct}' = \mathsf{ct}^*$ and outputting whatever $\mathcal{A}$ does.

The advantage we get from this distinguisher in the IND-CPA game is guaranteed to be $\mathsf{Adv}(\mathcal{D}) \geq \varepsilon - n'(\lambda) - \Pr[\mathbf{bad}] = \varepsilon - \mathsf{negl}(\lambda)$. This means that ε must be negligible due to SKE being IND-CPA secure. □

Acknowledgements

We thank the reviewer B of Asiacrypt 2024 for pointing out the existence of the notion and constructions of all-but-one signatures. Gennaro Avitabile received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), and from the Spanish Government under projects PRODIGY (TED2021-132464B-I00) and ESPADA (PID2022-142290OB-I00). The last two projects are co-funded by European Union EIE, and NextGenerationEU/PRTR funds. Nico Döttling: Funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666. p. 792–821. Springer-Verlag, Berlin, Heidelberg (2016)
2. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
3. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_29
4. Campanelli, M., David, B., Khoshakhlagh, H., Konring, A., Nielsen, J.B.: Encryption to the future - A paradigm for sending secret messages to future (anonymous) committees. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 151–180. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_6
5. Cerulli, A., Connolly, A., Neven, G., Preiss, F.S., Shoup, V.: vetkeys: How a blockchain can keep many secrets. Cryptology ePrint Archive, Paper 2023/616 (2023), https://eprint.iacr.org/2023/616, https://eprint.iacr.org/2023/616
6. Chakraborty, S., Prabhakaran, M., Wichs, D.: Witness maps and applications. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 220–246. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45374-9_8
7. Choudhuri, A.R., Garg, S., Piet, J., Policharla, G.V.: Mempool privacy via batched threshold encryption: Attacks and defenses. In: 33rd USENIX Security Symposium (USENIX Security 24). pp. 3513–3529. USENIX Association, Philadelphia, PA (Aug 2024), https://www.usenix.org/conference/usenixsecurity24/presentation/choudhuri
8. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28
9. Döttling, N., Hanzlik, L., Magri, B., Wohnig, S.: Mcfly: Verifiable encryption to the future made practical. In: Baldimtsi, F., Cachin, C. (eds.) Financial Cryptography and Data Security. pp. 252–269. Springer Nature Switzerland, Cham (2024)
10. Erwig, A., Faust, S., Riahi, S.: Large-scale non-interactive threshold cryptosystems through anonymity. Cryptology ePrint Archive, Report 2021/1290 (2021), https://eprint.iacr.org/2021/1290

11. Faust, S., Hazay, C., Kretzler, D., Schlosser, B.: Statement-oblivious threshold witness encryption. In: CSF 2023 Computer Security Foundations Symposium. pp. 17–32. IEEE Computer Society Press (Jul 2023). https://doi.org/10.1109/CSF57540.2023.00026
12. Frankel, Y.: A practical protocol for large group oriented networks. In: Quisquater, J.J., Vandewalle, J. (eds.) EUROCRYPT'89. LNCS, vol. 434, pp. 56–61. Springer, Heidelberg (Apr 1990). https://doi.org/10.1007/3-540-46885-4_8
13. Gailly, N., Melissaris, K., Romailler, Y.: tlock: Practical timelock encryption from threshold bls. Cryptology ePrint Archive, Paper 2023/189 (2023), https://eprint.iacr.org/2023/189, https://eprint.iacr.org/2023/189
14. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.13
15. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488667
16. Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. Cryptology ePrint Archive, Paper 2024/263 (2024), https://eprint.iacr.org/2024/263, https://eprint.iacr.org/2024/263
17. Garg, S., Srinivasan, A.: A simple construction of iO for Turing machines. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 425–454. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_16
18. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011). https://doi.org/10.1145/1993636.1993651
19. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)
20. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run Turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 536–553. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_30
21. Goyal, R., Vusirikala, S., Waters, B.: Collusion resistant broadcast and trace from positional witness encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 3–33. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_1
22. Goyal, V., Kothapalli, A., Masserova, E., Parno, B., Song, Y.: Storing and retrieving secrets on a blockchain. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part I. LNCS, vol. 13177, pp. 252–282. Springer, Heidelberg (Mar 2022). https://doi.org/10.1007/978-3-030-97121-2_10
23. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_29
24. Hubacek, P., Wichs, D.: On the communication complexity of secure function evaluation with long output. In: Roughgarden, T. (ed.) ITCS 2015. pp. 163–172. ACM (Jan 2015). https://doi.org/10.1145/2688073.2688105
25. Jiang, M., Duong, D.H., Susilo, W.: Puncturable signature: A generic construction and instantiations. In: Computer Security – ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part II. p. 507–527. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-17146-8_25, https://doi.org/10.1007/978-3-031-17146-8_25
26. Kondi, Y., Magri, B., Orlandi, C., Shlomovits, O.: Refresh when you wake up: Proactive threshold wallets with offline devices. In: 2021 IEEE Symposium on Security and Privacy. pp. 608–625. IEEE Computer Society Press (May 2021). https://doi.org/10.1109/SP40001.2021.00067
27. Koppula, V., Lewko, A.B., Waters, B.: Indistinguishability obfuscation for Turing machines with unbounded memory. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 419–428. ACM Press (Jun 2015). https://doi.org/10.1145/2746539.2746614
28. Lin, H., Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation with non-trivial efficiency. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part II. LNCS, vol. 9615, pp. 447–462. Springer, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49387-8_17
29. Madathil, V., Thyagarajan, S.A.K., Vasilopoulos, D., Fournier, L., Malavolta, G., Moreno-Sanchez, P.: Cryptographic oracle-based conditional payments. In: 30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023. The Internet Society (2023), https://www.ndss-symposium.org/ndss-paper/cryptographic-oracle-based-conditional-payments/
30. Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_6

31. 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.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 121–145. Springer, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_6
32. Quach, W., Wee, H., Wichs, D.: Laconic function evaluation and applications. In: Thorup, M. (ed.) 59th FOCS. pp. 859–870. IEEE Computer Society Press (Oct 2018). https://doi.org/10.1109/FOCS.2018.00086
33. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014). https://doi.org/10.1145/2591796.2591825
34. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
35. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0_18
36. Waters, B., Wu, D.J.: Adaptively-sound succinct arguments for np from indistinguishability obfuscation. Cryptology ePrint Archive, Paper 2024/165 (2024), https://eprint.iacr.org/2024/165, https://eprint.iacr.org/2024/165

A An Alternative SPS Construction

As our SPS is based on PRGs and simulation-sound NIZK, we define them below before giving our construction.

A.1 Pseudorandom Generators

Let $\mu = \mu(\lambda)$ and $\nu = \nu(\lambda)$ be two polynomials. A pseudorandom generator $\mathsf{PRG} : \{0,1\}^\mu \to \{0,1\}^\nu$ is a function such that for all PPT adversaries $\mathcal{A}$ we have that

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\nu) : \begin{array}{c} s \leftarrow\$ \{0,1\}^\mu \\ \nu \leftarrow \mathsf{PRG}(s) \end{array}\right] - \Pr\left[1 \leftarrow \mathcal{A}(\nu) : \nu \leftarrow\$ \{0,1\}^\nu\right]\right| = \mathsf{negl}(\lambda).$$

A.2 Non-Interactive Zero-Knowledge Proofs

Let $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be an *NP* relation on pairs (x, y), with corresponding language $\mathcal{L} := \{x : \exists y \text{ s.t. } \mathcal{R}(x,y) = 1\}$. A non-interactive proof system for $\mathcal{R}$ allows a prover P to convince a verifier V that a common element x belongs to the language $\mathcal{L}$ (where both P and V are modeled as PPT algorithms); the prover P is facilitated by knowing a witness y for $x \in \mathcal{L}$.

Definition 23 (Non-Interactive Proof System). *A non-interactive proof system for an NP relation $\mathcal{R}$ is a tuple of efficient algorithms $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ specified as follows.*

- $\mathsf{crs} \leftarrow\$ \mathsf{Setup}(1^\lambda)$*: Setup takes as input the security parameter $\lambda \in \mathbb{N}$, and outputs the public common reference string (CRS)* crs*.*
- $\pi \leftarrow\$ \mathsf{P}(\mathsf{crs}, x, y)$*: The Prover takes as input the CRS* crs *and a pair x, y s.t. $\mathcal{R}(x,y) = 1$, and returns a proof π for membership of $x \in \mathcal{L}$.*
- $d = \mathsf{V}(\mathsf{crs}, x, \pi)$*: Verification takes as input the CRS* crs *and a pair (x, π), and returns a decision bit $d \in \{0,1\}$.*

We require the non-interactive proof system to satisfy completeness, perfect soundness, zero-knowledge, and simulation soundness defined below. We remark that a CRS is necessary to achieve non-interactive zero-knowledge (see, e.g., [19]).

Definition 24 (Completeness). *Let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a non-interactive proof system for an NP relation $\mathcal{R}$. We say that Π satisfies completeness if for all pairs (x, y) such that $\mathcal{R}(x,y) = 1$, it holds*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 : \ \pi \leftarrow\$ \mathsf{P}(\mathsf{crs}, x, y); \mathsf{crs} \leftarrow\$ \mathsf{Setup}(1^\lambda)\right] = 1.$$

Definition 25 (Perfect Soundness). *Let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a non-interactive proof system for an NP relation $\mathcal{R}$. We say that Π is perfectly sound if for all (unbounded) $\mathcal{A}$ we have that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin \mathcal{L} : (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs}); \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)\right] = 0.$$

Definition 26 (Zero-Knowledge). *Let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be a non-interactive proof system for an NP relation $\mathcal{R}$. We say that Π is zero-knowledge if there exists a PPT simulator $\mathsf{Sim} := (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for all non-uniform PPT adversaries $\mathcal{A}$ we have*

$$\left| \begin{matrix} \Pr[1 \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{P}(\mathsf{crs},\cdot,\cdot)}(\mathsf{crs}) : \mathsf{crs} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)] - \\ \Pr[1 \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{Sim}_2'(t,\cdot,\cdot)}(\mathsf{crs}) : (\mathsf{crs}, t) \leftarrow\!\!\$\ \mathsf{Sim}_1(1^\lambda)] \end{matrix} \right| = \mathsf{negl}(\lambda)$$

where $\mathsf{Sim}_2'(t, x, y) = \mathsf{Sim}_2(t, x)$ for $(x, y) \in \mathcal{R}$ and both oracles output $\perp$ if $(x, y) \notin \mathcal{R}$.

Simulation Soundness. The simulation soundness property says that a malicious prover cannot convince a verifier of the validity of a false statement, even after seeing simulated proofs of arbitrary statements [23].

Definition 27 (Simulation Soundness). *Let $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ be an efficient NIZK proof system for an NP relation $\mathcal{R}$ with corresponding language $\mathcal{L}$, with zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$. We say that Π is simulation-sound, if for all (unbounded) adversaries $\mathcal{A}$, the following probability is negligible:*

$$\Pr\left[((x, \pi) \notin Q) \wedge (x \notin \mathcal{L}) \wedge \mathsf{V}(\mathsf{crs}, x, y) = 1 \;\middle|\; \begin{matrix} (\mathsf{crs}, t) \leftarrow \mathsf{Sim}_1(1^\lambda); \\ (x, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}_2(\mathsf{crs}, t, \cdot)}(\mathsf{crs}) \end{matrix}\right]$$

where Q is the list of simulation queries and responses (x_i, π_i).

A.3 Construction

Let $\mathcal{M} = \{0,1\}^k$ be the message space, $m \in \mathcal{M}$ be a message, and $\mathsf{PRG} : \{0,1\}^\lambda \longrightarrow \{0,1\}^{2\lambda}$ be a pseudo-random generator. In addition, let $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vrfy})$ be a simulation sound NIZK proof system for the following relation

$$\mathcal{R} := \{((\mathsf{vk}, m), (i, s)) : b = m[i] \wedge \mathsf{vk}_{b,i} = \mathsf{PRG}(s)\},$$

where vk is of the form $\mathsf{vk} = \left(\begin{pmatrix} \mathsf{vk}_{0,1} & \cdots & \mathsf{vk}_{0,k} \\ \mathsf{vk}_{1,1} & \cdots & \mathsf{vk}_{1,k} \end{pmatrix}, \mathsf{CRS}\right)$. Next, we describe the algorithms of our SPS scheme $\Sigma = (\mathsf{KeyGen}, \mathsf{PKeyGen}, \mathsf{Sign}, \mathsf{Vrfy})$.

$\mathsf{KeyGen}(1^\lambda)$:
- Sample $\mathsf{CRS}_p \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$.
- Pick a random seed $s_{i,j} \leftarrow\!\!\$\ \{0,1\}^\lambda$ for all $(i,j) \in \{0,1\} \times [k]$.
- $\mathsf{sk} := \begin{pmatrix} s_{0,1} & \cdots & s_{0,k} \\ s_{1,1} & \cdots & s_{1,k} \end{pmatrix}$, $\mathsf{vk} := \left(\begin{pmatrix} \mathsf{PRG}(s_{0,1}) & \cdots & \mathsf{PRG}(s_{0,k}) \\ \mathsf{PRG}(s_{1,1}) & \cdots & \mathsf{PRG}(s_{1,k}) \end{pmatrix}, \mathsf{CRS}_p\right)$.
- Return $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{PKeyGen}(1^\lambda, m^*)$:
- Sample $\mathsf{CRS}_p \leftarrow\!\!\$\ \Pi.\mathsf{Setup}(1^\lambda)$.
- For all $(i,j) \in \{0,1\} \times [k]$, sample $\mathsf{sk} := \begin{cases} s_{i,j} \leftarrow \perp & \text{if } i = m^*[j] \\ s_{i,j} \leftarrow\!\!\$\ \{0,1\}^\lambda & \text{otherwise} \end{cases}$

– For all $(i,j) \in \{0,1\} \times [k]$, sample $\mathsf{vk}' := \begin{cases} p_{i,j} \leftarrow\!\!\$\ \{0,1\}^{2\lambda} & \text{if } i = m^*[j] \\ p_{i,j} = \mathsf{PRG}(s_{i,j}) & \text{otherwise} \end{cases}$
– Set $\mathsf{vk} := (\mathsf{vk}', \mathsf{CRS}_p)$.
– Return $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{Sign}(\mathsf{sk}, m)$:
– Parse $\mathsf{sk} := \begin{pmatrix} s_{0,1} & \cdots & s_{0,k} \\ s_{1,1} & \cdots & s_{1,k} \end{pmatrix}$
– Let $\ell \in [k]$ be the first index such that $s_{m[\ell],\ell} \neq \bot$. If such an index does not exist, return $\bot$.
– Set $b = m[\ell]$.
– Compute[3] $\pi = \Pi.\mathsf{Prove}(\mathsf{CRS}_p, (\mathsf{vk}, m), (\ell, s_{b,\ell}))$.
– Return $\sigma = \pi$.

$\mathsf{Vrfy}(\mathsf{vk}, \sigma, m)$:
– Return $\Pi.\mathsf{Vrfy}(\mathsf{CRS}_p, (\mathsf{vk}, m), \sigma)$.

A.4 Correctness, Puncturability and Security of the Construction

Theorem 4. *The SPS scheme Σ from Section A.3 is correct (Def. 15).*

Proof. Let $m \in \mathcal{M}$, and let $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$, for

$$(\mathsf{vk}, \mathsf{sk}) = \left(\left(\begin{pmatrix} \mathsf{PRG}(s_{0,1}) & \cdots & \mathsf{PRG}(s_{0,k}) \\ \mathsf{PRG}(s_{1,1}) & \cdots & \mathsf{PRG}(s_{1,k}) \end{pmatrix}, \mathsf{CRS}_p \right), \begin{pmatrix} s_{0,1} & \cdots & s_{0,k} \\ s_{1,1} & \cdots & s_{1,k} \end{pmatrix} \right).$$

We set σ as $\pi = \Pi.\mathsf{Prove}(\mathsf{CRS}_p, (\mathsf{vk}, m), (\ell, s_{m[\ell],\ell}))$ where $\ell := \min\{i \in [k] : s_{m[i],i} \neq \bot\} = 1$ by construction of sk. Since $\mathsf{vk}_{m[\ell],\ell} = \mathsf{PRG}(s_{m[\ell],\ell})$, it is $(\mathsf{vk}, m) \in \mathcal{L}_\mathcal{R}$ and $(\ell, s_{m[\ell],\ell}) \in \mathcal{R}(\mathsf{vk}, m)$. The completeness requirement of the underlying NIZK proof system for $\mathcal{L}_\mathcal{R}$ gives $\Pi.\mathsf{Vrfy}(\mathsf{CRS}_p, (\mathsf{vk}, m), \pi) = 1$ and hence $\mathsf{Vrfy}(vk, \sigma, m) = 1$. □

Theorem 5. *The SPS scheme Σ of Section A.3 is puncturable (Def 16), given that Π is a NIZK and* PRG *is a pseudorandom generator.*

Proof. Notice that all the k indices in the punctured verification key vk^* that correspond to the bits of the message m^* are sampled as random elements instead of PRG values. Thus any potential signature for vk^* and message m^* would have to contain a NIZK proof for a statement that is false with overwhelming probability. This is the case because the probability of even one PRG pre-image existing for the punctured vk^* is less than $k \cdot 2^{-\lambda}$. To conclude the proof it suffices to observe that since Π is perfectly sound, there exists no valid proofs for false statements. Thus, there does not exist a valid signature for the key vk^* and the message m^*, except with negligible probability. □

Theorem 6. *The SPS scheme Σ of Section A.3 is punctured-key indistinguishable (Def. 17), if Π is NIZK and* PRG *is a pseudorandom generator.*

Proof. To show this, we define a series of indistinguishable hybrids starting from $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$ with $b = 0$ and ending up with $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$ with $b = 1$.

$\mathcal{H}_0$: $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$ with $b = 0$, that is $(\mathsf{vk}, \mathsf{sk})$ are generated running $\mathsf{KeyGen}(1^\lambda)$.

[3] Notice that CRS_p can be part of sk as well, and thus the algorithm Sign can be seen as having CRS_p as an implicit input.

$\mathcal{H}_1$: This hybrid is identical to the previous one, except that CRS_p is generated running $\Pi.\mathsf{Sim}_1$ instead of $\Pi.\mathsf{Setup}$ and, when replying to signature generation queries, the proof π is computed running $\Pi.\mathsf{Sim}_2$ instead of $\Pi.\mathsf{Prove}$. Computational indistinguishability from the previous hybrid follows from the zero-knowledge of Π.

$\mathcal{H}_{2,1} \to \mathcal{H}_{2,k}$: $\mathcal{H}_{2,j}$ is identical to $\mathcal{H}_{2,j-1}$ (where $\mathcal{H}_{2,0} := \mathcal{H}_1$), except that $(\mathsf{vk}, \mathsf{sk})$ are generated as follows. Let m^* be the message given in output by $\mathcal{A}$, then for $i \in \{0,1\}$ the element $\mathsf{vk}_{i,j}$ is generated as $\mathsf{vk}_{i,j} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ if $i = m^*[j]$. Observe, that we compute all our proofs from the simulator, so swapping the PRF values in vk does not impact our ability to generate signatures. Computational indistinguishability from the previous hybrid follows from the fact that PRG is a pseudorandom generator. Note that in $\mathcal{H}_{2,k}$ the verification key generation is identical to $\mathsf{PKeyGen}(1^\lambda, m^*)$ and the signing key is not used.

$\mathcal{H}_3$ This hybrid is identical to $\mathcal{H}_{2,k}$, except that $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{PKeyGen}(1^\lambda, m^*)$, CRS_p is generated running $\Pi.\mathsf{Setup}$ instead of $\Pi.\mathsf{Sim}_1$ and, when replying to signature generation queries, the proof π is computed running $\Pi.\mathsf{Prove}$ instead of $\Pi.\mathsf{Sim}_2$. Indistinguishability from the previous hybrid follows from the zero-knowledge of Π and the fact, that the public verification key distribution is identical to the one in $\mathcal{H}_{2,k}$. Notice that $\mathcal{H}_3$ is identical to $\mathsf{Exp}_{\mathsf{IND\text{-}SPS}}(\mathcal{A}, 1^\lambda)$ with $b = 1$.

□

Theorem 7. *The SPS scheme Σ of Section A.3 is unforgeable (Def. 18), if Π is a simulation sound NIZK and* PRG *is a pseudorandom generator.*

Proof. To show this, we define a series of indistinguishable hybrids and we show that if $\mathcal{A}$ has a non-negligible advantage in the last hybrid, we can use $\mathcal{A}$ to break the simulation soundness of the NIZK Π.

$\mathcal{H}_0$: This game is identical to $\mathsf{Exp}_{\mathsf{EUF\text{-}CMA}}(\mathcal{A}, 1^\lambda)$.

$\mathcal{H}_1$: This hybrid is identical to the previous one, except that CRS_p is generated running $\Pi.\mathsf{Sim}_1$ instead of $\Pi.\mathsf{Setup}$ and, when replying to signature generation queries, the proof π is computed running $\Pi.\mathsf{Sim}_2$ instead of $\Pi.\mathsf{Prove}$. Computational indistinguishability from the previous hybrid follows from the zero-knowledge of Π.

$\mathcal{H}_2$: This hybrid is identical to the previous one, except that the vk is generated as a uniformly random matrix, instead of being derived using the PRG. Computational indistinguishability from the previous hybrid follows from the pseudorandomness property of PRG. This can be easily argued via another series of indistinguishable hybrids where one element at a time of vk is replaced with a uniformly random one.

Given an $\mathcal{A}$ that wins in $\mathcal{H}_2$ with non-negligible probability, we can construct an adversary $\mathcal{B}$ winning the simulation soundness game w.r.t. Π with the same probability. $\mathcal{B}$ plays the simulation soundness game with oracle access to $\Pi.\mathsf{Sim}_2$, while running $\mathcal{A}$ internally. $\mathcal{B}$ replies to all the queries of $\mathcal{A}$ by perfectly simulating $\mathcal{H}_2$ by querying its oracle to get the simulated proof π to complete the signature. However, since all the $2k$ elements in vk are random, the probability of even one pre-image existing for vk is less than $2k \cdot 2^{-\lambda}$. Thus, any valid signature forgery output by $\mathcal{A}$ will contain, with overwhelming probability, a proof w.r.t. a false statement. Since the message is part of our NIZK proof statement and a valid forge must be on a message that was not previously queried, this is guaranteed to be a fresh proof. Then, $\mathcal{B}$ simply takes this statement and proof pair and outputs it in the simulation soundness game. □