Extractable Witness Encryption for Signed Vector Digests from Pairings and Trust-Scalable One-Time Programs

Sora Suegami

suegamisora@gmail.com

May 18, 2024

Abstract

Witness encryption (WE) allows a ciphertext to be encrypted under an NP problem such that anyone holding a valid witness for that problem can decrypt it (flexible decryptors), without interaction with others (non-interaction). However, existing schemes are either impractical or achieve only a part of these WE features. We propose a novel WE scheme that 1) is based on bilinear maps such as pairings, 2) achieves the property of flexible decryptors, and 3) still requires the decryptor's communication with a trusted signer, who only performs a fixed amount of computation and communication at regular intervals, regardless of the number of ciphertexts. It provides extractable security and can be extended to a threshold multiple signers setting, avoiding reliance on a single signer. As a significant application of our WE scheme, we build a novel one-time program (OTP) scheme in which the signers' computational and communication costs remain constant, independent of the number of OTPs to be evaluated simultaneously. This feature ensures scalable OTP evaluations without risking decreased signer participation or compromised decentralization due to increased operational costs for the signers.

1 Introduction

Witness encryption (WE) [GGSW13] generalizes the concept of public key encryption (PKE). In this scheme, an encryptor encrypts a message under an instance x of an NP language $\mathcal{L}$, which serves the role typically played by a public key. A decryptor can decrypt the ciphertext if the decryptor possesses a witness ω for x, which fulfills the role usually reserved for a private key. A key feature of this scheme is that the encryptor is not required to specify the decryptor or the witness to be used; any holder of a valid witness can decrypt the ciphertext. Moreover, the decryptor does not need to interact with the

encryptor during decryption. Leveraging these two features of WE, we can build strong cryptographic primitives without the need for indistinguishability obfuscation (iO) [GVW19, FNV17, FWW23]. Besides, a combination of WE and blockchain allows novel applications that disclose secrets only when a specific future event occurs, e.g., time-lock encryption (TLE) [LJKW18] and one-time programs (OTPs) [GG17]. Especially, OTPs extend the functionality of WE from simply encrypting messages to evaluating secret functions, allowing the decryptor to evaluate a secret circuit only once with the decryptor's arbitrary input without revealing the circuit [GG17, SH23]. However, although most of these applications require WE to handle complex conditions for decryption, all of the WE schemes supporting any NP language rely on inefficient cryptographic schemes or not well-studied assumptions such as multilinear maps [GGSW13, GLW14, GKP+13, ULCG21], iO [AFP16, CJK20], affine determinant programs [BIJ+20], and not well-established variants of LWE assumption [VWW22, Tsa22].

Interestingly, some WE schemes have been proposed built from practical and well-studied cryptographic tools, specifically bilinear maps, at the cost of significantly restricting the supported NP languages. We categorize these into two categories: commitment-based WE (CWE) [DS18, BL20, CDK+22, CFK24, FHAS24] and signature-based WE (SWE) [DHMW23, MTV+22]. In the CWE scheme, the encryptor encrypts a message for the commitment of the decryptor's input, and the decryptor can decrypt the ciphertext if the input meets the specified conditions [BL20]. The decryptor does not need to interact with the encryptor at the time of decryption if they provide their commitment for the encryptor beforehand. Notably, the size of the commitment in some CWE schemes [FHAS24, CFK24] remains constant regardless of the input size. However, the decryption requires the randomness used to generate the commitment as part of the witness, which restricts decryption to one who created the commitment.

In the SWE scheme, the encryptor encrypts a secret message by specifying the verification key of a digital signature scheme and the signing target, which is the message to be signed [DHMW23, MTV+22]. The decryptor can then decrypt the ciphertext using a valid signature for the specified verification key and signing target. As a result, once the signature is available, anyone who possesses the signature can decrypt the ciphertext. Moreover, if the signer, who holds the signing key corresponding to the verification key, is different from the encryptor, only interaction with the signer is necessary at the time of decryption. This means that there is no need for the encryptor to communicate with the decryptor after the ciphertext has been released. However, the computational and communication costs for the signer are not scalable because the signer must generate a signature on each signing target for which the ciphertext is generated. In this way, no previous WE scheme built from bilinear maps requires only a sublinear amount of communication for decryption and avoids specifying the decryptor at the time of encryption.

Our question thus arises:

Can we build WE from bilinear maps that allows decryption with a signature while minimizing the signer's computational and communication costs relative to the number of signing targets?

Our Contribution: Yes, we can. We propose a WE scheme for a signature on a digest of multiple signing targets. Our scheme maintains the signer's computational and communication costs constant because the signer only needs to sign the digest that has a constant size regardless of the number of signing targets. Additionally, it provides extractable security, a stronger security definition than that of standard WE [GKP+13]. Similar to CWE and SWE, our construction relies only on bilinear maps, such as pairings, and standard cryptographic schemes, specifically symmetric-key encryption (SKE), in the random oracle model (ROM). We can extend its signature generation algorithm to a threshold multiple signers setting, where decryption requires signatures from at least a threshold number of signers.

By employing our WE scheme, we build trust-scalable OTPs (TSOTPs). Specifically, to enable the evaluation of a polynomial number of OTPs generated by independent OTP generators, the signer only needs to perform a fixed amount of computation and communication at regular intervals. Notably, this is the first OTP scheme where the computational and communication costs for trusted components prepared for OTPs, such as tamper-proof hardware [GKR08, GIS+10, EGG+22] or nodes for OTP evaluations that are assumed to be honest above a certain threshold [GKM+22, SH23], remain constant regardless of the number of independent OTPs [1]. This feature allows the signer to simply operate a machine capable of performing a fixed amount of computation and communication required for generating signatures on the digest. Consequently, when threshold multiple signers are employed, even if the number of evaluated OTPs increases, the operational costs of the signers' machines is bounded. In other words, OTP evaluation can be scaled without increasing the risk to reduce the number of signers and worsen decentralization.

2 Technical Overview

2.1 Review of Witness Encryption built from Bilinear Maps

Before explaining the construction of our scheme, we review common techniques in the CWE and SWE schemes, which are built from bilinear maps [DS18, BL20, CDK+22, CFK24, FHAS24, DHMW23, MTV+22]. Each scheme defines the NP relation $\mathcal{R}$ that should be satisfied by the instance x and the witness ω as a linear equation defined over bilinear groups $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$. Specifically, while a vector of elements in $\mathbb{G}_2$ and an element in $\mathbb{G}_T$, denoted

[1] One exception is an OTP scheme using blockchain proposed in [GG17]. However, it relies on extractable witness encryption (EWE) for arbitrary NP languages, which might be infeasible because the only known constructions of such EWE are based on non-standard assumptions [GKP+13, ABG+13, BCP14], and it has been shown that EWE for general NP languages may not be secure against adversaries with access to arbitrary auxiliary data [GGHW17].

by $[\mathbf{g}]_2$ and $[h]_T$ respectively, only depends on x, a vector of elements in $\mathbb{G}_1$, denoted by $[\mathbf{f}]_1$, is derived from ω. These satisfy the following condition:

$$(x, \omega) \in \mathcal{R} \Leftrightarrow e([\mathbf{f}]_1, [\mathbf{g}]_2) = [h]_T$$

We call $[\mathbf{f}]_1$ valid for $[\mathbf{g}]_2$ and $[h]_T$ if the above equation holds.

Using $[\mathbf{g}]_2$ and $[h]_T$, an encryptor can compute a common secret element in $\mathbb{G}_T$ such that only a decryptor who holds the $[\mathbf{f}]_1$ valid for the $[\mathbf{g}]_2$ and $[h]_T$ can derive the same element later. In more detail, the encryptor samples a random scalar $s \leftarrow\!\!\$\ \mathcal{F}_p$ and computes $s[\mathbf{g}]_2$ and $s[h]_T$. They then encrypts a message using a secret key derived from $s[h]_T$ and publishes $s[\mathbf{g}]_2$ and the ciphertext. If the decryptor knows the valid $[\mathbf{f}]_1$, they should be able to derive the same $s[h]_T$ as $e([\mathbf{f}]_1, s[\mathbf{g}]_2) = s[h]_T$ holds. Thus, the decryptor can recover the same secret key to decrypt the ciphertext. However, an adversary without the valid $[\mathbf{f}]_1$ cannot achieve this. Informally, this is because the adversary cannot predict $s[h]_T$ from $[\mathbf{g}]_2$ and $[h]_T$ due to the randomness s. In this way, the linearly verifiable equation defined over bilinear groups can be easily extended to WE.

Notably, the WE for KZG commitments [KZG10] proposed in [FHAS24] provides extractable security, which is a stronger form of the standard security definition known as semantic security. While semantic security ensures that an adversary cannot learn non-trivial information about the message from the ciphertext if the instance used for the ciphertext is not in the NP language (i.e., no witness corresponds to the instance), extractable security ensures that an adversary who can break the confidentiality of the ciphertext must be able to provide a witness for the instance [GKP+13]. This stronger security is necessary for some applications of WE, particularly OTPs, where the instance is in the NP language but the adversary cannot obtain its witness in a realistic timeframe [GG17]. The WE scheme in [FHAS24] achieves it by first constructing extractable witness key encapsulation mechanism (EKEMs) and converting it to EWE. Similarly, we apply the same techniques as those used in [FHAS24] to our linear equation for verifying a signed vector digest.

2.2 Witness Encryption for Signed Vector Digests

The review in Subsection 2.1 suggests that our task in building a new WE scheme from bilinear maps is to define a linear equation for the NP relation to verify a signature on a digest of multiple signing targets. Recall that our goal is to aggregate multiple signing targets into a constant-sized digest, requiring the signer to sign only the digest. To achieve this, we define a signed vector digest (SVD) scheme.

It is based on a vector commitment in [LY10] and its extension in [GRWZ20], with three differences outlined below.

1. Unlike vector commitments, the SVD scheme does not ensure a hiding property because anyone can test if a digest corresponds to a specific vector[2]. However, the size of the digest is constant and independent of

[2]This is why the SVD scheme is not called commitment scheme.

the vector size.

2. The SVD scheme includes a signing algorithm that uses a given signing key to output a signature on the given digest.

3. The verification algorithm of the SVD scheme requires not only a signing target in the digest and its opening but also a signature on the digest. Specifically, the verification passes if and only if 1) the opening proves that the given signing target is included in the digest at the given position and 2) the same digest is signed with a signing key corresponding to the given verifying key.

Additionally, we introduce the concept of time epochs, which is a counter incremented by the signer each time a new digest is signed. Each signature is associated with a specific time epoch t and can be verified with t. The signer is assumed to generate only one signature per time epoch[3]. Thus, the signer is stateful.

For simplicity, we first explain how the SVD scheme works in our WE scheme when a single encryptor generates ciphertexts for all indexes of a n-sized signing target vector and a signle decryptor specifies all signing targets. Let $\mathsf{sk} \in \mathbb{Z}_p$ and $\mathsf{vk} = [\mathsf{sk}]_2 \in \mathbb{G}_2$ be, respectively, the signer's signing and verifying keys, which have the same format as that of BLS signature [BLS04]. The encryptor samples a trapdoor $\alpha \leftarrow\!\!\$\ \mathbb{Z}_l$ and derives an output of a random oracle (RO) $[r_t]_1$ from vk and a time epoch t. Using them, it outputs a common reference string (CRS) as below:

$$\mathsf{crs}_1 \leftarrow (([\alpha^i]_1)_{i\in\{1,\ldots,n,n+2,\ldots,2n\}}) \in \mathbb{G}_1^{2n-1}$$
$$\mathsf{crs}_2 \leftarrow (([\alpha^i]_2)_{i\in\{1,\ldots,n\}}, \mathsf{vk}) \in \mathbb{G}_2^{n+1}$$
$$\mathsf{crs}_T \leftarrow ([v_0]_T = e([\alpha^{n+1}]_1, \mathsf{vk}), ([v_i]_T = e(\alpha^{n+1-i}[r_t]_1, \mathsf{vk}))_{i\in\{1,\ldots,n\}}) \in \mathbb{G}_T^{n+1}$$

When encrypting a message for a signing target T_i and its position index $i \in \{1, \ldots, n\}$, the encryptor prepares the following vector $[\mathbf{g}]_2$ and element $[h]_T$.

$$[\mathbf{g}]_2 \leftarrow ([\alpha^{n+1-i}]_2, -\mathsf{vk})$$
$$[h]_T \leftarrow T_i[v_0]_T + [v_i]_T$$

By applying techniques described in Subsection 2.1 to the above $[\mathbf{g}]_2$ and $[h]_T$, the encryptor can generate a ciphertext that can be decrypted with a vector $[\mathbf{f}]_1$ satisfying $e([\mathbf{f}]_1, [\mathbf{g}]_2) = [h]_T$.

The decryptor with n signing targets $\mathbf{T} = (T_1, \ldots, T_n)$ then aggregates them into a digest digest and requests the signer to generate a signature sign for digest and the time epoch t. Specifically, digest and sign are defined with the CRSs and $\mathbf{T}$ as follows:

$$\mathsf{digest} \leftarrow [d]_1 = \Sigma_{j\in[n]} T_j[\alpha^j]_1 \in \mathbb{G}_1$$
$$\mathsf{sign} \leftarrow [\sigma]_1 = \mathsf{sk}([r_t]_1 + [d]_1) \in \mathbb{G}_1$$

[3]In the threshold multiple signers setting, the same assumption is made for more than the threshold number of signers.

, where $[r_t]_1$ is the output of RO for vk and t. Notably, the signer's computational cost is constant regardless of n. To prove that T_i is included in digest at the position i, the decryptor computes an opening open_i:

$$\mathsf{open}_i \leftarrow [W_i]_1 = \Sigma_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}} T_j[\alpha^{n+1-i+j}]$$

You can see that $[\mathbf{f}]_1 = ([\sigma]_1, [W_i]_1)$ satisfies the equation $e([\mathbf{f}]_1, [\mathbf{g}]_2) = [h]_T$ as below:

$$\begin{aligned}
&e([\mathbf{f}]_1, [\mathbf{g}]_2)\\
=&e([\sigma]_1, [\alpha^{n+1-i}]_2) + e([W_i]_1, -\mathsf{vk})\\
=&[\mathsf{sk}\cdot r_t\alpha^{n+1-i}]_T + [\mathsf{sk}\cdot\Sigma_{j\in\{1,\ldots,n\}} T_j\alpha^{n+1-i+j}]_T - [\mathsf{sk}\cdot\Sigma_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}} T_j\alpha^{n+1-i+j}]_T\\
=&T_i[\mathsf{sk}\cdot\alpha^{n+1}]_T + [\mathsf{sk}\cdot r_t\alpha^{n+1-i}]_T\\
=&T_i[v_0]_T + [v_i]_T\\
=&[h]_T
\end{aligned}$$

Thus, after receiving the signature $[\sigma]_1$, the decryptor can decrypt a ciphertext for any (i, T_i) such that T_i exists in the digest $[d]_1$ at the position i.

However, an adversary cannot forge the valid $[\mathbf{f}]_1$ for $[\mathbf{g}]_2$ and $[h]_T$ corresponding to any (i, T_i) that is not included in the digest. Informally, since $[\alpha^{n+1}]_1$ is excluded from crs_1, the adversary cannot make a term of $T_i[\mathsf{sk}\cdot\alpha^{n+1}]_T$ without adding T_i to the signed digest. Besides, the adversary cannot obtain any signature for the time epoch t before the signer outputs it because the adversary cannot compute a term of $[\mathsf{sk}\cdot r_t]_T$, where r_t is unique to each t, without the signer's help. These security features are defined as the binding and the time-locking properties in Definitions 10 and 11, respectively. In this manner, we can define the secure SVD scheme of which the verification is represented by the linear equation.

2.3 Trust-Scalable One-Time Programs

Using the WE scheme defined for the SVD scheme, we construct trust-scalable OTPs (TSOTPs), where the signer's computational and communication costs remain constant for each time epoch t, independent of the number of OTP generators L and the input size n of the circuits to be evaluated. In a nutshell, our OTP construction is based on that of [SH23], which is built from a garbled circuit and SWE. However, we 1) replace SWE with our WE scheme and 2) makes the OTP secure against an adaptive adversary who chooses the input after seeing the OTP.

We first present the basic idea of our OTP construction, where the OTP is secure only against a selective adversary who chooses the input before seeing the OTP. For a circuit C with the input size n, the signer's verifying key vk, and the next time epoch t, the OTP generator generates a garbled circuit $\widetilde{C}$ and its garbled inputs $(\mathsf{lab}_{i,b})_{i\in\{1,\ldots,n\},b\in\{0,1\}}$, generates a CRS of the WE scheme for vk and t, and encrypts each $\mathsf{lab}_{i,b}$ under i and b. It implies that the $\mathsf{lab}_{i,b}$

can be recovered iff the input bit b is included in the signed digest as the i-th siging target. The garbled circuit and these ciphertexts constitute the OTP, formally referred to as the compiled circuit. Its evaluator chooses the n-sized input $\mathbf{T} \in \{0,1\}^n$, computes its digest, requests the signer to sign the digest, decrypts each encryption of lab_{i,T_i}, and evaluates the garbled circuit. This construction prevents the evaluator from evaluating C on multiple inputs as long as the signer signs a digest only once for each time epoch t.

Unfortunately, we cannot formally prove the adaptive security of the above construction because a simulator, behaving as an honest generator, cannot know the adaptive adversary's circuit input when generating the compiled circuit. In more detail, even the adaptively secure garbled circuit requires the simulator to know the circuit output, depending on the circuit input, to generate a simulated garbled input that is independent of the input [HJO$^+$15]. However, the simulator needs to complete all generation processes before the adaptive adversary chooses the input in the above construction.

To fix this issue, we employ RO to allow the simulator to simulate the garbled input after learning the adversary's input in a manner similar to [BHR12]. Specifically, the simulator first samples $2n$ random bit strings $(r_{i,b})_{i\in\{1,\ldots,n\},b\in\{0,1\}}$ and outputs each $r_{i,b}$ instead of the encryption of the garbled input $\mathsf{lab}_{i,b}$. Additionally, it generates a secret key k and the corresponding ciphertext ct for $(n+1,0)$ with the EKEM scheme. The ct is added to the compiled circuit, allowing the evaluator to recover the same k iff the evaluator holds a signature for the time epoch t on the digest in which a signing target $T_{n+1} = 0$ is included at the position $n+1$.

After the adversary chooses the input $\mathbf{T} \in \{0,1\}^n$, the simulator computes a digest aggregating the $n+1$ signing targets $(T_1,\ldots,T_n,0) \in \{0,1\}^{n+1}$, and the signer returns a signature on that digest. At the same time, the simulator derives the simulated garbled input $(\mathsf{lab}_i')_{i\in[n]}$ from the circuit output $C(\mathbf{T})$, and programs the RO for each $i \in \{1,\ldots,n\}$ as follows, where ℓ corresponds to the generator's index among multiple generators.

- Upon the input (k,ℓ,i,T_i), the RO outputs the XOR of r_{i,T_i} and the WE encryption of the simulated garbled input lab_i'.
- Upon the input $(k,\ell,i,1-T_i)$, the RO outputs the XOR of $r_{i,1-T_i}$ and the WE encryption of a zero bit string $\mathbf{0}$.

As the inputs to the RO depend on the secret key k, which requires the signature to be recovered, the adversary cannot obtain the RO outputs involving the encryptions of the garbled inputs before submitting the circuit input. Therefore, the adversary cannot distinguish this simulated case from a real case where the compiled circuit contains the XOR of $r_{i,b}$ and the encryption of the real garbled input $\mathsf{lab}_{i,b}$ for each $i \in \{1,\ldots,n\}$ and $b \in \{0,1\}$, and the RO simply outputs $r_{i,b}$ for the input (k,ℓ,i,b). Besides, in the simulated case, the simulator can output the simulated garbled circuit that is independent of C instead of the real garbled circuit because there is only the garbled input for $\mathbf{T}$ in the adversary's view. In this way, by modifying the selectively-secure OTP construction with the RO,

we can prove that even the adaptive adversary cannot distinguish the real view from the simulated view that does not reveal the non-trivial information about C.

3 Preliminaries

3.1 Notations

Let $\mathbb{N}$ and $\mathbb{R}$ the sets of natural and real numbers, respectively. For $n \in \mathbb{N}$, $[n]$ denotes a set of $\{1, \ldots, n\}$, and $[0]$ represents an empty set $\emptyset$. A vector and a matrix are denoted by bold letters, e.g., $\mathbf{a}$ and $\mathbf{A}$. For any n-length vector $\mathbf{a}$ and its index $i \in [n]$, a_i represents the i-th component of $\mathbf{a}$, and $|\mathbf{a}|$ is the length of $\mathbf{a}$. For any distribution $\mathcal{X}$, $x \leftarrow\!\!\$\, \mathcal{X}$ denotes that x is sampled from $\mathcal{X}$.

Let λ be a security parameter. A function $\mathsf{negl}(\lambda) : \mathbb{N} \to \mathbb{R}$ is called negligible, if there exists $n \in \mathbb{N}$ for all constants $c > 0$ such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ holds for all $\lambda > n$.

3.2 Bilinear Groups

For a security parameter 1^λ, bilinear groups $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ are defined by a tuple of a prime p, cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p, their generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is efficient and non-degenerat. A generator of $\mathbb{G}_T$ is defined as $g_T = e(g_1, g_2)$.

We adopt the same bracket notation for group elements as in [CFK24]. That is, for $t \in \{1, 2, T\}$, $[a]_t$ represents g_t^a, and its scalar multiplication is denoted by $x[a]_t = g_t^{xa}$. For every $[a]_1$ and $[b]_2$, the bilinear map e satisfies $e([a]_1, [b]_2) = [ab]_T$. All of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are assumed to be additive groups. For two vectors with the same length $[\mathbf{a}]_1 \in \mathbb{G}_1^n$, $[\mathbf{b}]_2 \in \mathbb{G}_2^n$, $e([\mathbf{a}]_1, [\mathbf{b}]_2)$ represents $\Sigma_{i \in [n]} e([a_i]_1, [b_i]_2)$. Similarly, for a matrix $[\mathbf{A}]_1 \in \mathbb{G}_1^{n \times m}$ and a vector $[\mathbf{b}]_2 \in \mathbb{G}_2^m$, $e([\mathbf{A}]_1, [\mathbf{b}]_2) = [\mathbf{Ab}]_T$ holds.

In the following, we show some security assumptions defined over a group or bilinear groups. They are employed to prove the security of our schemes.

Definition 1 (Hardness of Discrete Logarithm Problem (DLP))**.** For a group $\mathbb{G}$ and its generator [1], the discrete logarithm problem (DLP) is said to be hard, if for every PPT adversary $\mathcal{A}$ and $a \in \mathbb{F}_p$,

$$\Pr[a \leftarrow \mathcal{A}([1], [a])] < \mathsf{negl}(\lambda)$$

holds.

Definition 2 is taken from [BFL20].

Definition 2 (Hardness of q-Discrete Logarithm Problem (q-DLP))**.** For a group $\mathbb{G}$ and its generator [1], the q-discrete logarithm problem (q-DLP) is said to be hard, if for every PPT adversary $\mathcal{A}$ and $a \leftarrow\!\!\$\, \mathbb{F}_p$,

$$\Pr[a \leftarrow \mathcal{A}([1], [a], \ldots, [a^q])] < \mathsf{negl}(\lambda)$$

holds.

Definition 3 is based on a definition of a co-Diffie-Hellman problem with a Type 3 pairing in [CHKM10].

Definition 3 (Hardness of co-Diffie-Hellman Problem (co-DHP))**.** For bilinear groups $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, the co-Diffie-Hellman problem (co-DHP) is said to be hard, if for every PPT adversary $\mathcal{A}$, $[a]_1$, $[a]_2$, and $[b]_2$,

$$\Pr[[ab]_2 \leftarrow \mathcal{A}([1]_1, [1]_2, [a]_1, [a]_2, [b]_2)] < \mathsf{negl}(\lambda)$$

holds.

While a n-Diffie-Hellman exponent problem is initially defined with symmetric bilinear maps in [BBG05, BGW05], Definition 4 follows its extension to asymmetric bilinear maps in [LPR22].

Definition 4 (Hardness of n-Diffie-Hellman Exponent Problem (n-DHEP))**.** For bilinear groups $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, the n-Diffie-Hellman Exponent Problem (n-DHEP) is said to be hard, if for every PPT adversary $\mathcal{A}$ and $\alpha \leftarrow\!\!\$\ \mathbb{F}_p$,

$$\Pr\big[[\alpha^{n+1}]_1 \leftarrow \mathcal{A}([1]_1, [1]_2, [\alpha]_1, \ldots, [\alpha^n]_1, [\alpha^{n+2}]_1, \ldots, [\alpha^{2n}]_1, [\alpha]_2, \ldots, [\alpha^n]_2)\big] < \mathsf{negl}(\lambda)$$

holds.

3.3 Algebraic Group Model

The security of our schemes is proven in the algebraic group model (AGM), introduced in [FKL18]. This model allows an adversary to perform algebraic operations on group elements but requires it to output coefficients to represent its output group element as a linear combination of the given group elements. Our formal definition of the AGM model is based on that of [CFK24], which extends the original definition in [FKL18] to asymmetric bilinear groups.

Definition 5 (Algebraic Adversarie)**.** Let $\mathsf{bg}_\lambda = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ be bilinear groups. A PPT adversary $\mathcal{A}$ with input $[\mathbf{x}]_1 \in \mathbb{G}_1^n$, $[\mathbf{y}]_2 \in \mathbb{G}_2^m$, $[\mathbf{z}]_T \in \mathbb{G}_T^l$ is said to be algebraic, if in addition to its outputs

$$[\mathbf{p}]_1 = ([p_1]_1, \ldots, [p_{n'}]_1) \in \mathbb{G}_1^{'}$$
$$[\mathbf{q}]_2 = ([q_1]_2, \ldots, [q_{m'}]_2) \in \mathbb{G}_2^{'}$$
$$[\mathbf{r}]_T = ([r_1]_T, \ldots, [r_{l'}]_T) \in \mathbb{G}_T^{'}$$

$\mathcal{A}$ also provides coefficients $(A_{i,j})_{i\in[n'],j\in[n]}$, $(B_{i,j})_{i\in[m'],j\in[m]}$, $(C_{i,j,k})_{i\in[l'],j\in[n],k\in[m]}$, $(D_{i,j})_{i\in[l'],j\in[l]}$ such that the following equations hold:

$$\forall i \in [n^{'}],\ [p_i]_1 = \Sigma_{j\in[n]} A_{i,j}[x_j]_1$$
$$\forall i \in [m^{'}],\ [q_i]_2 = \Sigma_{j\in[m]} B_{i,j}[y_j]_2$$
$$\forall i \in [l^{'}],\ [r_i]_T = \Sigma_{j\in[n],k\in[m]} C_{i,j,k} e([x_j]_1, [y_k]_2) + \Sigma_{j\in[l]} D_{i,j}[z_j]_T$$

3.4 Symmetric-Key Encryption

We use the definition of a symmetric-key encryption scheme SKE for a key space $\mathcal{K}$ and a message space $\mathcal{M}$ from Definition 10 in [FHAS24]. Notably, it assumes that the key k is randomly sampled from $\mathcal{K}$.

The SKE scheme defined for a key space $\mathcal{K}$ and a message space $\mathcal{M}$ consists of the following algorithms:

- $\mathsf{SKE.Enc}(k, m) \to \mathsf{ct}$: it takes as input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and outputs a ciphertext ct.
- $\mathsf{SKE.Dec}(k, \mathsf{ct}) \to m$: it takes as input a key $k \in \mathcal{K}$ and a ciphertext ct, and outputs a message $m \in \mathcal{M}$.

The SKE scheme is correct if for every $\lambda \in \mathbb{N}$, $k \in \mathcal{K}$, and $m \in \mathcal{M}$, it holds that

$$\Pr[\mathsf{SKE.Dec}(k, \mathsf{SKE.Enc}(k, m)) = m] = 1$$

We require the SKE scheme to satisfy EAV-security, which is weaker than IND-CPA security because the same key is used only for the encryption of the challenge message in the EAV-security.

Definition 6 (EAV-Security of the SKE scheme)**.** Let $\mathcal{SKE} = (\mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key encryption scheme defined for a key space $\mathcal{K}$ and a message space $\mathcal{M}$. The $\mathcal{SKE}$ scheme is said to be EAV-secure, if for every security parameter $\lambda \in \mathbb{N}$ and PPT adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{EXP}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{A}}(1^\lambda) = 1\right] < \frac{1}{2} + \mathsf{negl}(\lambda)$$

holds, where the experiment $\mathsf{EXP}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{A}}(1^\lambda)$ is defined as follows:

$\mathsf{EXP}^{\mathsf{SVD}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda)$

1 : $k \leftarrow\!\!\$\ \mathcal{K}$.
2 : $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$.
3 : $b \leftarrow\!\!\$\ \{0, 1\}$.
4 : $\mathsf{ct} \leftarrow \mathsf{SKE.Enc}(k, m_b)$.
5 : $b' \leftarrow \mathcal{A}(\mathsf{ct})$.
6 : **return** $b = b'$.

3.5 Adaptively Secure Garbled Circuit

Our definition of an adaptively secure garbled circuit GC is based on Definition 1 in [HJO+15]. However, ours assumes that the garbled inputs are projective in similar to Definition 7 of [EGG+22], that is, the garbled inputs can be decomposed into parts each of which depends on only one bit of the input. This

assumption is reasonable, as there are known garbled circuit schemes that are both adaptively secure and projective [HJO+15, JO20, BHR12].

The adaptively secure and projective garbled circuit GC, defined for a circuit class $\mathcal{C}_{n,m}$ where a circuit has the input size n and the output size m, consists of the following algorithms:

- $\mathsf{GC.Circuit}(1^\lambda, C) \to (\widetilde{C}, \mathsf{state} = (\mathsf{lab}_{i,b})_{i\in[n],b\in\{0,1\}})$: it takes as a security parameter 1^λ and a circuit $C \in \mathcal{C}_{n,m}$, and outputs a garbled circuit $\widetilde{C}$ and a state state consisting of garbled wires of all input bits $(\mathsf{lab}_{i,b})_{i\in[n],b\in\{0,1\}}$.
- $\mathsf{GC.Input}(\mathsf{state}, x) \to \widetilde{x} = (\mathsf{lab}_{i,x_i})_{i\in[n]}$: it takes as input a state state and an input $x \in \{0,1\}^n$, and outputs a garbled input $\widetilde{x}$ consisting of $(\mathsf{lab}_{i,x_i})_{i\in[n]}$.
- $\mathsf{GC.Eval}(\widetilde{C}, \widetilde{x}) \to y$: it takes as input a garbled circuit $\widetilde{C}$ and a garbled input $\widetilde{x}$, and outputs the evaluation result $y \in \{0,1\}^m$.

The GC scheme is correct if for every $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_{n,m}$, and $x \in \{0,1\}^n$, it holds that

$$\mathsf{GC.Eval}(\widetilde{C}, \widetilde{x}) = C(x)$$

, where $(\widetilde{C}, \mathsf{state}) \leftarrow \mathsf{GC.Circuit}(1^\lambda, C)$ and $\widetilde{x} \leftarrow \mathsf{GC.Input}(\mathsf{state}, x)$.

Definition 7 (Adaptive Security of the GC scheme)**.** Let $\mathsf{GC} = (\mathsf{Circuit}, \mathsf{Input}, \mathsf{Eval})$ be a garbled circuit scheme defined for a circuit class $\mathcal{C}_{n,m}$. The GC scheme is said to be adaptively secure, if for every security parameter $\lambda \in \mathbb{N}$, there exist PPT simulators $\mathsf{GC.SimC}$ and $\mathsf{GC.SimIn}$ such that

$$\Pr\left[\mathsf{EXP}^{\mathsf{GC\text{-}Adap}}_{\mathcal{A}}(1^\lambda) = 1\right] < \frac{1}{2} + \mathsf{negl}(\lambda)$$

holds for every PPT adversary $\mathcal{A}$, where the experiment $\mathsf{EXP}^{\mathsf{GC\text{-}Adap}}_{\mathcal{A}}(1^\lambda)$ is defined as follows:

$\mathsf{EXP}^{\mathsf{GC\text{-}Adap}}_{\mathcal{A}}(1^\lambda)$

1 : $(C, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$, where $\mathsf{state}_{\mathcal{A}}$ is the state of $\mathcal{A}$.
2 : $b \leftarrow\!\!\$\ \{0,1\}$.
3 : If $b = 0$, let $(\widetilde{C}, \mathsf{state}_{\mathsf{GC}}) \leftarrow \mathsf{GC.Circuit}(1^\lambda, C)$.
4 : If $b = 1$, let $(\widetilde{C}, \mathsf{state}_{\mathsf{GC.Sim}}) \leftarrow \mathsf{GC.SimC}(1^\lambda, 1^{|C|})$.
5 : $x \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, \widetilde{C})$.
6 : If $b = 0$, let $\widetilde{x} \leftarrow \mathsf{GC.Input}(\mathsf{state}_{\mathsf{GC}}, x)$.
7 : If $b = 1$, let $\widetilde{x} \leftarrow \mathsf{GC.SimIn}(C(x), \mathsf{state}_{\mathsf{GC.Sim}})$.
8 : $b' \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, \widetilde{x})$.
9 : **return** $b = b'$.

4 Signed Vector Digests (SVDs)

We first construct a SVD scheme. In the SVD scheme, there are L parties, a trusted stateful signer, and a verifier. Each of the L parties specifies the signer's verifying key vk, a time epoch t, and n signing targets, and one of them computes a digest from all nL signing targets. The signer signs the data derived from the submitted digest and t using the signing key sk. However, the signer does not output the signatures for multiple digests with the same t. Once the signature is available, anyone, even a person who is not in the L parties submitting the signing targets, can show a verifier that the signing target is in the signed digest.

The significant efficiency property of the SVD scheme is that the signer's computational and communication costs are constant for each t regardless of the vector size, which is defined as trust-scalable property in Definition 9. This is because the signer only needs to sign the constant-size data for each t, and the same signature can be used for the verifications of all messages in the digest. It allows many parties to employ the same well-trusted signer for large-size vectors.

4.1 Definition

The SVD scheme defined for a signing target space $\mathcal{T}$ consists of the following algorithms:

- $\mathsf{SVD.SetupKey}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$: it takes as input a security parameter 1^λ and outputs a pair of signing and verifying keys $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{SVD.SetupCRS}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t) \to (\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$: it takes as input a public parameter pp, consisting of a security parameter 1^λ and integers $n, L \in \mathbb{N}$ bounded by a polynomial in λ, a verifying key vk, and a time epoch t. It outputs each party's common reference string crs_l and its trapdoor td_l.
- $\mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}) \to \mathsf{digest}$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, and signing targets $(\mathbf{T}_l)_{l\in[L]}$, each of which is in $\mathcal{T}^n$. It outputs a digest digest.
- $\mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t, \mathsf{digest}) \to \mathsf{sign}$: it takes as input a signing key sk, a time epoch t, and a digest digest. It outputs a signature sign.
- $\mathsf{SVD.Open}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \ell, i) \to \mathsf{open}_{\ell,i}$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, signing targets $(\mathbf{T}_l)_{l\in[L]}$, and indexes $\ell \in [L]$ and $i \in [n]$. It outputs an opening proof $\mathsf{open}_{\ell,i}$.
- $\mathsf{SVD.Verify}(\mathsf{crs}_l, \mathsf{sign}, \ell, i, T_{\ell,i}, \mathsf{open}_{\ell,i}) \to \{0, 1\}$: it takes as a input common reference string crs_ℓ, a time epoch t, a signature sign, indexes $\ell \in [L]$ and $i \in [n]$, an expected signing target $T_{\ell,i}$, and an opening proof $\mathsf{open}_{\ell,i}$. It outputs either 1 or 0.

The SVD scheme is correct if for every $\lambda \in \mathbb{N}$, $n, L, t \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $(\mathbf{T}_l)_{l\in[L]} \in \mathcal{T}^{nL}$,

$$\Pr\left[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}, \ell, i, T_{\ell,i}, \mathsf{open}_{\ell,i}) = 1\right] \geq 1 - \mathsf{negl}(\lambda)$$

holds for every $\ell \in [L]$ and $i \in [n]$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SVD.SetupKey}(1^\lambda)$, $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp}, \mathsf{vk}, t)$, $\mathsf{digest} \leftarrow \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$, $\mathsf{sign} \leftarrow \mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t, \mathsf{digest})$ and $\mathsf{open}_{\ell,i} \leftarrow \mathsf{SVD.Open}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \ell, i)$.

Before defining the security of the SVD scheme, we introduce a stateful signer oracle OSign in association with the SVD scheme.

Definition 8 (Stateful Signer Oracle)**.** Let SVD=($\mathsf{SetupKey}$, $\mathsf{SetupCRS}$, Digest, Sign, Open, Verify) be a SVD scheme defined for a signing target space $\mathcal{T}$. A stateful signer oracle OSign defined for SVD and public parameters $\mathsf{pp} = (1^\lambda, n, L)$ maintains $\mathsf{state} \leftarrow (\mathsf{sk}, t)$, where sk is the signing key of the SVD scheme and $t \in \mathbb{N}$ is the latest time epoch.

OSign is initialized as below.

1. Sample $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SVD.SetupKey}(1^\lambda)$.
2. Let state be $(\mathsf{sk}, 1)$.
3. Output vk.

Subsequently, for each query, OSign receives digest as input and performs the following procedure.

1. Parse state as (sk, t).
2. Compute $\mathsf{sign}_t \leftarrow \mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t, \mathsf{digest})$.
3. Update state to $(\mathsf{sk}, t+1)$.
4. Output sign_t.

We define the following efficiency property for cryptographic schemes that depends on OSign.

Definition 9 (Trust-Scalable Cryptographic Scheme)**.** A cryptographic scheme that employs a stateful signer oracle OSign is said to be trust-scalable, if the signer's computational and communication costs for each time epoch t is bounded by a polynomial in a security parameter $\mathsf{poly}(\lambda)$.

The secure SVD scheme should satisfy the binding and time-locking properties: the former ensures that a signer cannot open the same digest with the same position and different signing targets and the latter claims that an adversary cannot obtain any valid signature for a time epoch t^* before the trusted signer outputs the t^*-th signature. Notably, no PPT adversary should be able to break any property for ℓ as long as the ℓ-th trapdoor td_ℓ is hidden from the adversary.

Definition 10 (Binding of the SVD scheme)**.** Let SVD=($\mathsf{SetupKey}$, $\mathsf{SetupCRS}$, Digest, Sign, Open, Verify) be a SVD scheme defined for a signing target space $\mathcal{T}$. The SVD scheme is said to be binding if for every $\lambda \in \mathbb{N}$, $n, L, t^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $\ell \in [L]$, $i \in [n]$, and algebraic adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{EXP}^{\mathsf{SVD\text{-}Bind}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i}) = 1\right] < \mathsf{negl}(\lambda)$$

holds, where OSign is a stateful signer oracle for SVD and 1^λ, and $\mathsf{EXP}^{\mathsf{SVD\text{-}Bind}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$ is an experiment defined as follows:

$\mathsf{EXP}^{\mathsf{SVD\text{-}Bind}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$

1 : OSign is initialized, which outputs vk.
2 : $\mathcal{A}$ calls OSign until the time epoch in state is less than t^*. Let $\mathsf{state}_{\mathcal{A}}$ be the $\mathcal{A}$'s state.
3 : $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t^*)$
4 : $(\mathbf{T}_l)_{l\in[L]} \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$, where $T_{\ell,i} \neq T^*_{\ell,i}$
5 : $\mathsf{digest} \leftarrow \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$
6 : $\mathsf{sign}_t \leftarrow \mathsf{OSign}(\mathsf{digest})$
7 : $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i}) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, \mathsf{sign}_t)$
8 : **return** $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T^*_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1$

Definition 11 (Time-Locking of the SVD scheme)**.** Let SVD=($\mathsf{SetupKey}$, $\mathsf{SetupCRS}$, Digest, Sign, Open, Verify) be a SVD scheme defined for a signing target space $\mathcal{T}$. The SVD scheme is said to be time-locking if for every $\lambda \in \mathbb{N}$, $n, L, t^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $\ell \in [L]$, $i \in [n]$, and algebraic adversary $\mathcal{A}$,

$$\Pr\left[\mathsf{EXP}^{\mathsf{SVD\text{-}Time}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i}) = 1\right] < \mathsf{negl}(\lambda)$$

holds, where OSign is a stateful signer oracle for SVD and 1^λ, and $\mathsf{EXP}^{\mathsf{SVD\text{-}Time}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$ is an experiment defined as follows:

$\mathsf{EXP}^{\mathsf{SVD\text{-}Time}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$

1 : OSign is initialized, which outputs vk.
2 : $\mathcal{A}$ calls OSign until the time epoch in state is less than t^*. Let $\mathsf{state}_{\mathcal{A}}$ be the $\mathcal{A}$'s state.
3 : $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t^*)$
4 : $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i}) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$
5 : **return** $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T^*_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1$

Remark 1 (Security against Malicious Adversaries). The above security definitions assume semi-honest adversaries, i.e., the adversary follows the protocol but tries to learn the honest party's information as much as possible. For example, although the adversary can learn all trapdoors except for the ℓ-th one,

their corresponding CRSs are assumed to be generated honestly. We can easily make our scheme secure against malicious adversaries with zk-SNARKs [Nit20], ensuring that CRSs and digests are generated in honest manners without revealing the underlying randomnesses. However, the proof size and the verifier's computation completely of the zk-SNARKs schemes should be constant irrespective of the complexity of the NP relation to be proved, which is represented by 1^λ, n, and L in our case. Such zk-SNARKs schemes, e.g., Groth16 [Gro16] and Plonk [GWC19], maintain the signer's computational and communication costs constant for each time epoch t.

4.2 Construction

Our construction relies on a random oracle RO that maps a public parameter pp, a verifying key vk, and a time epoch t to a random element $[r_t]_1$. It is used in the $\mathsf{SVD.SetupCRS}$ and $\mathsf{SVD.Sign}$ algorithms. Additionally, the signing target space is assumed to be a subset of $\mathbb{F}_p$ or equal to $\mathbb{F}_p$, i.e., $\mathcal{T} \subseteq \mathbb{F}_p$.

The $\mathsf{SVD.SetupCRS}$ algorithm in our construction is defined as a computation among L parties, where each party performs an individual CRS generation algorithm $\mathsf{SVD.SetupCRSEach}$ as below. Specifically, for $\ell \in [L]$, the ℓ-th party generates its CRS crs_ℓ using the previous parties' CRSs $(\mathsf{crs}_l)_{l\in[\ell-1]}$. It implies that the CRSs are generated in a non-interactive but sequential manner.

- $\mathsf{SVD.SetupKey}(1^\lambda) \to (\mathsf{sk}, \mathsf{vk})$:
  1. Sample $\mathsf{sk} \leftarrow \mathbb{Z}_p$.
  2. Let $\mathsf{vk} \leftarrow [\mathsf{sk}]_2$.
  3. Output sk and vk.
- $\mathsf{SVD.SetupCRSEach}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t, \ell, (\mathsf{crs}_l)_{l\in[\ell-1]}) \to (\mathsf{crs}_\ell, \mathsf{td}_\ell)$:
  1. Sample $\alpha_\ell \in \mathbb{Z}_p$.
  2. Compute $[r_t]_1 \leftarrow \mathsf{RO}(\mathsf{pp}, \mathsf{vk}, t)$.
  3. For each $l \in [\ell-1]$, parse crs_l as $(\mathsf{pp}, \mathsf{crs}_{l,1}, \mathsf{crs}_{l,2}, \mathsf{crs}_{l,T})$.
  4. For each $l \in [\ell-1]$, extract $([\alpha_l^i]_1)_{i\in[2n]/\{n+1\}}$ from $\mathsf{crs}_{l,1}$.
  5. For each $l \in [\ell-1]$ and $i, j \in [n]$, compute $[u_{\ell,l,i,j}]_1 \leftarrow \alpha_\ell^i[\alpha_l^j]_1$.
  6. Compute $[v_{\ell,0}]_T \leftarrow e([\alpha_\ell^{n+1}]_1, \mathsf{vk})$ and $[v_{\ell,i}]_T \leftarrow e(\alpha_\ell^{n+1-i}[r_t]_1, \mathsf{vk})$ for $i \in [n]$.
  7. Let $\mathsf{crs}_{\ell,1} \leftarrow (([\alpha_\ell^i]_1)_{i\in[2n]/\{n+1\}}, ([u_{\ell,l,i,j}]_1)_{l\in[\ell-1],i,j\in[n]})$.
  8. Let $\mathsf{crs}_{\ell,2} \leftarrow (([\alpha_\ell^i]_2)_{i\in[n]}, \mathsf{vk})$.
  9. Let $\mathsf{crs}_{\ell,T} \leftarrow ([v_{\ell,0}]_T, ([v_{\ell,i}]_T)_{i\in[n]})$.
  10. Output $\mathsf{crs}_\ell \leftarrow (\mathsf{pp}, \mathsf{crs}_{\ell,1}, \mathsf{crs}_{\ell,2}, \mathsf{crs}_{\ell,T})$ and $\mathsf{td}_\ell \leftarrow (\alpha_\ell)$.
- $\mathsf{SVD.SetupCRS}(\mathsf{pp}, \mathsf{vk}, t) \to (\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$:

1. For each $\ell \in [L]$, the ℓ-th party sequentially performs $\mathsf{SVD.SetupCRSEach}(\mathsf{pp}, \mathsf{vk}, t, \ell, (\mathsf{crs}_l)_{l\in[\ell-1]})$, which returns crs_ℓ and td_ℓ.
2. Output $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$.

- $\mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}) \to \mathsf{digest}$:
  1. For each $l \in [L]$, parse crs_l as $(\mathsf{pp}, \mathsf{crs}_{l,1}, \mathsf{crs}_{l,2}, \mathsf{crs}_{l,T})$.
  2. For each $l \in [L]$, extract $([\alpha_l^i]_1)_{i\in[n]}$ from $\mathsf{crs}_{l,1}$.
  3. Compute $[d]_1 \leftarrow \Sigma_{l\in[L]}\Sigma_{j\in[n]} T_j[\alpha_l^j]_1$.
  4. Output $\mathsf{digest} \leftarrow [d]_1$.
- $\mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t, \mathsf{digest}) \to \mathsf{sign}$:
  1. Parse digest as $[d]_1$.
  2. Compute $[r_t]_1 \leftarrow \mathsf{RO}(\mathsf{pp}, \mathsf{vk}, t)$.
  3. Compute $[\sigma]_1 \leftarrow \mathsf{sk}([r_t]_1 + [d]_1)$.
  4. Output $\mathsf{sign} \leftarrow [\sigma]_1$.
- $\mathsf{SVD.Open}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \ell, i) \to \mathsf{open}_{\ell,i}$:
  1. For each $l \in [L]$, parse crs_ℓ as $(\mathsf{pp}, \mathsf{crs}_{\ell,1}, \mathsf{crs}_{\ell,2}, \mathsf{crs}_{\ell,T})$.
  2. For each $l \in [L]$, parse $\mathsf{crs}_{\ell,1}$ as $(([\alpha_\ell^i]_1)_{i\in[2n]/\{n+1\}}, ([u_{\ell,l,i,j}]_1)_{l\in[\ell-1],i,j\in[n]})$.
  3. Compute $[W^s_{\ell,i}]_1 \leftarrow \Sigma_{j\in[n]/\{i\}} T_j[\alpha_\ell^{n+1-i+j}]_1$.
  4. Compute $[W^o_{\ell,i}]_1 \leftarrow \Sigma_{j\in[n]}(\Sigma_{l\in[\ell-1]} T_{l,j}[u_{\ell,l,n+1-i,j}]_1 + \Sigma_{l\in[L]/[\ell]} T_{l,j}[u_{l,\ell,j,n+1-i}]_1)$.
  5. Compute $[W_{\ell,i}]_1 = [W^s_{\ell,i}]_1 + [W^o_{\ell,i}]_1$.
  6. Output $\mathsf{open}_{\ell,i} \leftarrow [W_{\ell,i}]_1$.
- $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}, \ell, i, T_{\ell,i}, \mathsf{open}_{\ell,i}) \to \{0, 1\}$:
  1. Parse crs_ℓ as $(\mathsf{pp}, \mathsf{crs}_{\ell,1}, \mathsf{crs}_{\ell,2}, \mathsf{crs}_{\ell,T})$.
  2. Extract $[\alpha_\ell^{n+1-i}]_2$ and vk from $\mathsf{crs}_{\ell,2}$.
  3. Extract $[v_{\ell,0}]_T$ and $[v_{\ell,i}]_T$ from $\mathsf{crs}_{\ell,T}$.
  4. Parse sign as $[\sigma]_1$.
  5. Parse $\mathsf{open}_{\ell,i}$ as $[W_{\ell,i}]_1$.
  6. Output 1 if $e([\sigma]_1, [\alpha_\ell^{n+1-i}]_2) = e([W_{\ell,i}]_1, \mathsf{vk}) + T_{\ell,i}[v_{\ell,0}]_T + [v_{\ell,i}]_T$, otherwise output 0.

The correctness proof is straightforward.

$$
\begin{aligned}
& e([\sigma]_1, [\alpha_\ell^{n+1-i}]_2) \\
=& e(\mathsf{sk} \cdot ([r_t]_1 + \{\Sigma_{l\in[L]}\Sigma_{j\in[n]} T_{l,j}[\alpha_l^j]_1\}), [\alpha_\ell^{n+1-i}]_2) \\
=& e(\Sigma_{j\in[n]/\{i\}} T_{\ell,j}[\alpha_\ell^{n+1-i+j}]_1 + \Sigma_{j\in[n]}(\Sigma_{l\in[\ell-1]} T_{l,j}[\alpha_\ell^{n+1-i}\alpha_l^j]_1 + \Sigma_{l\in[L]/[\ell]} T_{l,j}[\alpha_l^j \alpha_\ell^{n+1-i}]_1), \mathsf{vk}) \\
& + T_{\ell,i} e([\alpha_\ell^{n+1}]_1, \mathsf{vk}) + e([r_t \alpha_\ell^{n+1-i}]_1, \mathsf{vk}) \\
=& e([W_{\ell,i}^s]_1 + [W_{\ell,i}^o]_1, \mathsf{vk}) + T_{\ell,i} \cdot e([\alpha_\ell^{n+1}]_1, \mathsf{vk}) + e([\alpha_\ell^{n+1-i} r_t]_1, \mathsf{vk}) \\
=& e([W_{\ell,i}]_1, \mathsf{vk}) + T_{\ell,i}[v_{\ell,0}]_T + [v_{\ell,i}]_T
\end{aligned}
$$

The SVD scheme is trust-scalable.

Lemma 1. The construction of the SVD scheme in Subsection 4.2 is trust-scalable (Definition 9).

Proof. In that construction of the SVD scheme, the only algorithm performed by the signer is $\mathsf{SVD.Sign}$. It consists of calling RO with the fixed input size, adding $[r]_1$ to the given digest $[d]_1$, and multiplying sk by $[r]_1 + [d]_1$. It is clear that these computational and communication complexities are bounded by a polynomial in 1^λ and do not depend on the other parameters such as n and L. □

The above construction satisfies the binding and time-locking properties.

Lemma 2. The construction of the SVD scheme in Subsection 4.2 is binding (Definition 10).

Proof. A simulator of the experiment $\mathsf{EXP}^{\mathsf{SVD\text{-}Bind}}_{\mathcal{A},\mathsf{OSign}}$ programs a random oracle RO in a lazy manner. That is, for each query $(\mathsf{pp}, \mathsf{vk}, t)$, if $\mathsf{RO}[(\mathsf{pp}, \mathsf{vk}, t)] = [r_t]_1 \neq \perp$, i.e., any $[r_t]_1$ is already returned for the same query, the simulator returns the same $[r_t]_1$. Otherwise, it samples a random $[r_t]_1 \leftarrow\!\$\ \mathbb{G}_1$, sets $\mathsf{RO}[(\mathsf{pp}, \mathsf{vk}, t)] \leftarrow [r_t]_1$, and returns $[r_t]_1$.

Suppose that algebraic adversaries $\mathcal{A}$ in AGM model output $(\mathbf{T}_1, \ldots, \mathbf{T}_L, [\sigma^*]_1, [W_{i,\ell}^*]_1, T_{\ell,i}^*)$ such that the equation in the $\mathsf{SVD.Verify}$ algorithm holds. $\mathcal{A}$ computes $[\sigma^*]_1$ and $[W_{i,\ell}^*]_1$ as a linear combination of seen elements in $\mathbb{G}_1$. That is, it outputs coefficients such that the following holds:

$$
\begin{aligned}
[\sigma^*]_1 &= x_0[1]_1 + \Sigma_{j\in[2n]/\{n+1\}} x_j[\alpha_\ell^j]_1 + \Sigma_{t\in[t^*]} x_{2n+t}[r_t]_1 + \Sigma_{t\in[t^*]} x_{2n+t^*+t}[\sigma_t]_1 \\
[W_{i,\ell}^*]_1 &= y_0[1]_1 + \Sigma_{j\in[2n]/\{n+1\}} y_j[\alpha_\ell^j]_1 + \Sigma_{t\in[t^*]} y_{2n+t}[r_t]_1 + \Sigma_{t\in[t^*]} y_{2n+t^*+t}[\sigma_t]_1
\end{aligned}
$$

Notably, the elements $([u_{\ell,l,i,j}])_{l\in[L]/\{\ell\},i,j\in[n]}$ are not used as the basis because the adversary can employ $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ and $([\alpha_\ell^i])_{i\in[n]}$ to obtain $[u_{\ell,l,i,j}]_1$ by calculating $[u_{\ell,l,i,j}]_1 = \alpha_l^j[\alpha_\ell^i]$. Besides, we ignore the output of RO that is not equal to any of $([r_1]_1, \ldots, [r_t]_1)$, which is obviously independent of the other basis.

We separate the terms of $[\sigma^*]_1$ and $[W_{\ell,i}^*]_1$ into those depending on sk and those that do not. Specifically, $[\sigma^*]_1$ and $[W_{\ell,i}^*]_1$ are, respectively, represented

as $[P + Q \cdot \mathsf{sk}]_1$ and $[R + S \cdot \mathsf{sk}]_1$. Here, we show some useful claims to prove $P = 0$ and $S = 0$.

Claim 1. Assuming the hardness of the co-DHP (Definition 3), for every algebraic adversary $\mathcal{A}$ and random $x \leftarrow\!\!\$\ \mathbb{F}_p$,

$$\Pr\left[[x^2]_1 \leftarrow \mathcal{A}([1]_1, [1]_2, [x]_1, [x]_2)\right] < \mathsf{negl}(\lambda)$$

Proof. A PPT adversary $\mathcal{B}$ that breaks the hardness of the co-DHP assumption can be constructed by invoking $\mathcal{A}$. Given $([1]_1, [1]_2, [a]_1, [a]_2, [b]_1)$, $\mathcal{B}$ first randomly samples $s, t \leftarrow\!\!\$\ \mathbb{F}_p$ and provides $([1]_1, [1]_2, [x]_1 = s[a]_1 + [t]_1, [x]_2 = s[a]_2 + [t]_2)$ for $\mathcal{A}$. In the AGM model, the $\mathcal{A}$ outputs $[x^2]_1$ along with coefficients u, v such that $[x^2]_1 = u[1]_1 + v[x]_1$. Therefore, $\mathcal{B}$ can compute x as one root of the equation $X^2 - vX - u = 0$, which suggests $a = s^{-1}(x - t)$. Hence, it can output $a[b]_1 = [ab]_1$. □

Claim 2. Assuming the hardness of the co-DHP (Definition 3), for every algebraic adversary $\mathcal{A}$ and random $x \leftarrow\!\!\$\ \mathbb{F}_p$,

$$\Pr\left[[x^{-1}]_1 \leftarrow \mathcal{A}([1]_1, [1]_2, [x]_1, [x]_2)\right] < \mathsf{negl}(\lambda)$$

Proof. A PPT adversary $\mathcal{B}$ that breaks the hardness of the co-DHP assumption can be constructed by invoking $\mathcal{A}$. Given $([1]_1, [1]_2, [a]_1, [a]_2, [b]_1)$, $\mathcal{B}$ first randomly samples $s, t \leftarrow\!\!\$\ \mathbb{F}_p$ and provides $([1]_1, [1]_2, [x]_1 = s[a]_1 + [t]_1, [x]_2 = s[a]_2 + [t]_2)$ for $\mathcal{A}$. In the AGM model, the $\mathcal{A}$ outputs $[x^{-1}]_1$ along with coefficients u, v such that $[x^{-1}]_1 = u[1]_1 + v[x]_1$. Therefore, $\mathcal{B}$ can compute x as one root of the equation $X^{-1} - vX - u = 0 \Leftrightarrow vX^2 + uX - 1 = 0$, which suggests $a = s^{-1}(x - t)$. Hence, it can output $a[b]_1 = [ab]_1$. □

First, Claim 1 implies $S = 0$. Suppose $S \neq 0$, a PPT adversary $\mathcal{B}_1$ that breaks Claim 1 can be constructed by internally invoking $\mathcal{A}$. Given $([1]_1, [1]_2, [a]_1, [a]_2)$, $\mathcal{B}_1$ first randomly samples $(\alpha_l)_{l\in[L]}, (r_t)_{t\in[t^*]}$ and let $[\mathsf{sk}]_1 \leftarrow [a]_1$ and $\mathsf{vk} \leftarrow [a]_2$. It then constructs $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$ using these values, and provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns $(\mathbf{T}_l)_{l\in[L]}$. $\mathcal{B}_1$ provides $\mathcal{A}$ with $\mathsf{sign} = [\sigma]_1 = (r_t + \Sigma_{l\in[L]}\Sigma_{j\in[n]} T_{l,j}\alpha_l^j)[\mathsf{sk}]_1$. $\mathcal{A}$ finally returns $\mathsf{sign}^* = [\sigma^*]_1$, $\mathsf{open}^*_{\ell,i} = [W^*_{\ell,i}]_1$, and $T^*_{\ell,i}$. Here, $\mathcal{A}$ should also output coefficients such that the following holds:

$$\begin{aligned}
&(P + Q \cdot \mathsf{sk})\alpha_\ell^{n+1-i} = (R + S \cdot \mathsf{sk}) \cdot \mathsf{sk} + T^*_{\ell,i}\alpha_\ell^{n+1} \cdot \mathsf{sk} + r_{t^*}\alpha_\ell^{n+1-i} \cdot \mathsf{sk} \\
\Leftrightarrow &S \cdot \mathsf{sk}^2 = P\alpha_\ell^{n+1-i} + (Q\alpha_\ell^{n+1-i} - R - T^*_{\ell,i}\alpha_\ell^{n+1} - r_{t^*}\alpha_\ell^{n+1-i}) \cdot \mathsf{sk}
\end{aligned}$$

Therefore, suppose $S \neq 0$, $\mathcal{B}_1$ can compute $[a^2] = [\mathsf{sk}^2]_1 = S^{-1}\{P\alpha_\ell^{n+1-i}[1]_1 + (Q\alpha_\ell^{n+1-i} - R - T^*_{\ell,i}\alpha_\ell^{n+1} - r_{t^*}\alpha_\ell^{n+1-i})[\mathsf{sk}]_1\}$, which breaks Claim 1. This is a contradiction, hence $S = 0$ holds.

Next, we show $P = 0$ from Claim 2. Suppose $P \neq 0$, a PPT adversary $\mathcal{B}_2$ that breaks Claim 2 can be constructed by internally invoking $\mathcal{A}$. Given $([1]_1, [1]_2, [a]_1, [a]_2)$, $\mathcal{B}_2$ calls $\mathcal{A}$ in the same manner as $\mathcal{B}_1$ defined in the proof of $S \neq 0$. It then constructs $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$ using these values, and provides

$(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns $(\mathbf{T}_l)_{l\in[L]}$. $\mathcal{A}$ should output $\mathsf{sign}^* = [\sigma^*]_1$, $\mathsf{open}^*_{\ell,i} = [W^*_{\ell,i}]_1$, $T^*_{\ell,i}$, and coefficients such that the following holds:

$$\begin{aligned}&(P + Q\cdot\mathsf{sk})\alpha_\ell^{n+1-i} = R\cdot\mathsf{sk} + T^*_{\ell,i}\alpha_\ell^{n+1}\cdot\mathsf{sk} + r_{t^*}\alpha_\ell^{n+1-i}\cdot\mathsf{sk}\\ \Leftrightarrow &P\alpha_\ell^{n+1-i}\mathsf{sk}^{-1} = -Q\alpha_\ell^{n+1-i} + R + T^*_{\ell,i}\alpha_\ell^{n+1} + r_{t^*}\alpha_\ell^{n+1-i}\end{aligned}$$

Therefore, suppose $P \neq 0$, $\mathcal{B}_2$ can compute $[a^{-1}] = [\mathsf{sk}^{-1}]_1 = P^{-1}\alpha_\ell^{-n-1+i}(-Q\alpha_\ell^{n+1-i} + R + T^*_{\ell,i}\alpha_\ell^{n+1} + r_{t^*}\alpha_\ell^{n+1-i})[1]_1$, which breaks Claim 2. This is a contradiction, hence $P = 0$ holds.

The above shows the following equation.

$$\begin{aligned}&Q\alpha_\ell^{n+1-i}\cdot\mathsf{sk} = R\cdot\mathsf{sk} + T^*_{\ell,i}\alpha_\ell^{n+1}\cdot\mathsf{sk} + r_{t^*}\alpha_\ell^{n+1-i}\cdot\mathsf{sk}\\ \Leftrightarrow &Q\alpha_\ell^{n+1-i} = R + T^*_{\ell,i}\alpha_\ell^{n+1} + r_{t^*}\alpha_\ell^{n+1-i}\\ \Leftrightarrow &\Sigma_{t\in[t^*]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i} = y_0 + \Sigma_{j\in[2n]/\{n+1\}}y_j\alpha_\ell^j + \Sigma_{t\in[t^*]}y_{2n+t}r_t + T^*_{\ell,i}\alpha_\ell^{n+1} + r_{t^*}\alpha_\ell^{n+1-i}\\ \Leftrightarrow &\Sigma_{t\in[t^*-1]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i} + x_{2n+2t^*}\{r_{t^*}\alpha_\ell^{n+1-i} + \Sigma_{j\in[n]}(T_{\ell,j}\alpha_\ell^{n+1-i+j} + \Sigma_{l\in[L]/\{\ell\}}T_{l,j}\alpha_\ell^{n+1-i}\alpha_l^j)\}\\ &= y_0 + \Sigma_{j\in[2n]/\{n+1\}}y_j\alpha_\ell^j + \Sigma_{t\in[t^*]}y_{2n+t}r_t + T^*_{\ell,i}\alpha_\ell^{n+1} + r_{t^*}\alpha_\ell^{n+1-i}\\ \Leftrightarrow &(x_{2n+2t^*}\alpha_\ell^{n+1-i} - y_{2n+t^*} - \alpha^{n+1-i})r_{t^*}\\ &= -\Sigma_{t\in[t^*-1]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i} - x_{2n+2t^*}\Sigma_{j\in[n]}(T_{\ell,j}\alpha_\ell^{n+1-i+j} + \Sigma_{l\in[L]/\{\ell\}}T_{l,j}\alpha_\ell^{n+1-i}\alpha_l^j)\\ &+ y_0 + \Sigma_{j\in[2n]/\{n+1\}}y_j\alpha_\ell^j + \Sigma_{t\in[t^*-1]}y_{2n+t}r_t + T^*_{\ell,i}\alpha_\ell^{n+1}\end{aligned}$$

From the hardness of DLP (Definition 1), the coefficient of r_{t^*} should be zero, i.e., $y_{2n+t^*} = (x_{2n+2t^*}-1)\alpha_\ell^{n+1-i}$. To prove that, we construct a PPT adversary $\mathcal{B}_3$ that breaks the hardness of DLP by invoking $\mathcal{A}$. Given $([1]_1, [a]_1)$, $\mathcal{B}_3$ first randomly samples $(\alpha_l)_{l\in[L]}, \mathsf{sk}, (r_t)_{t\in[t^*-1]}\beta, \gamma \leftarrow\!\!\$\ \mathbb{F}_p$ and let $[r_{t^*}]_t \leftarrow \beta[a]_1 + [\gamma]_1$. It then constructs $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$ using these values, and provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns $(\mathbf{T}_l)_{l\in[L]}$. $\mathcal{B}_3$ provides $\mathcal{A}$ with $\mathsf{sign} = [\sigma]_1 = \mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t^*, \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}))$. $\mathcal{A}$ finally returns $\mathsf{sign}^* = [\sigma^*]_1$, $\mathsf{open}^*_{\ell,i} = [W^*_{\ell,i}]_1$, $T^*_{\ell,i}$. Suppose $y_{2n+t^*} \neq (x_{2n+2t^*} - 1)\alpha_\ell^{n+1-i}$, it holds that

$$\begin{aligned}r_{t^*} = &(x_{2n+2t^*}\alpha_\ell^{n+1-i} - y_{2n+t^*} - \alpha_\ell^{n+1-i})^{-1}\{-\Sigma_{t\in[t^*-1]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i}\\ &- x_{2n+2t^*}\Sigma_{j\in[n]}(T_{\ell,j}\alpha_\ell^{n+1-i+j} + \Sigma_{l\in[L]/\{\ell\}}T_{l,j}\alpha_\ell^{n+1-i}\alpha_l^j) + y_0 + \Sigma_{j\in[2n]/\{n+1\}}y_j\alpha_\ell^j + T^*_{\ell,i}\alpha_\ell^{n+1}\}\end{aligned}$$

Therefore, $\mathcal{B}_3$ can compute $a = \beta^{-1}(r_{t^*} - \gamma)$, which breaks the DLP. This is a contradiction, hence $y_{2n+t^*} = (x_{2n+2t^*} - 1)\alpha_\ell^{n+1-i}$ holds.

We finally show that $\mathcal{A}$ allows us to construct a PPT adversary $\mathcal{B}_4$ that breaks the hardness of the n-DHEP (Definition 4). Given $([1]_1, [\alpha_\ell]_1, \ldots, [\alpha_\ell^n]_1, [\alpha_\ell^{n+2}]_1, \ldots, [\alpha_\ell^{2n}]_1$, $[1]_2, [\alpha_\ell]_2, \ldots, [\alpha_\ell^n]_2)$, $\mathcal{B}_4$ first samples $(\alpha)_{l\in[L]/\{\ell\}}, (r_t)_{t\in[t^*]}, \mathsf{sk}$ and constructs $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$ using those values. Subsequently, it provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns $(\mathbf{T}_l)_{l\in[L]}$. $\mathcal{B}_4$ provides $\mathcal{A}$ with $\mathsf{sign} = [\sigma]_1 = \mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}))$. $\mathcal{A}$ finally returns $\mathsf{sign}^* = [\sigma^*]_1$, $\mathsf{open}^*_{\ell,i} =$

$[W^*_{\ell,i}]_1$, and $T^*_{\ell,i}$. They should satisfy the following equation:

$$(x_{2n+2t^*}T_{\ell,i} - T^*_{\ell,i})\alpha_\ell^{n+1}$$
$$= -\Sigma_{t\in[t^*-1]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i} - x_{2n+2t^*}(\Sigma_{j\in[n]/\{i\}}T_{\ell,j}\alpha_\ell^{n+1-i+j} + \Sigma_{j\in[n]}\Sigma_{l\in[L]/\{\ell\}}T_{l,j}\alpha_\ell^{n+1-i}\alpha_l^j)$$
$$+ y_0 + \Sigma_{j\in[2n]/\{n+1\}}y_j\alpha_\ell^j + \Sigma_{t\in[t^*-1]}y_{2n+t}r_t$$

When $x_{2n+2t^*}T_{\ell,i} - T^*_{\ell,i} \neq 0$, $\mathcal{B}$ can obtain $[\alpha^{n+1}]_1$ as follows:

$$\begin{aligned}[\alpha_\ell^{n+1}]_1 =&(x_{2n+2t^*}T_{\ell,i} - T^*_{\ell,i})^{-1}\{-\Sigma_{t\in[t^*-1]}x_{2n+t^*+t}\sigma_t\alpha_\ell^{n+1-i}\\ &- x_{2n+2t^*}(\Sigma_{j\in[n]/\{i\}}T_{\ell,j}[\alpha_\ell^{n+1-i+j}]_1 + \Sigma_{j\in[n]}\Sigma_{l\in[L]/\{\ell\}}T_{l,j}\alpha_l^j[\alpha_\ell^{n+1-i}]_1)\\ &+ y_0[1]_1 + \Sigma_{j\in[2n]/\{n+1\}}y_j[\alpha_\ell^j]_1 + \Sigma_{t\in[t^*-1]}y_{2n+t}r_t[1]_1\}\end{aligned}$$

When $x_{2n+2t^*}T_{\ell,i} - T^*_{\ell,i} = 0$, since $T^*_{\ell,i} \neq T_{\ell,i}$, $T^*_{\ell,i} = x_{2n+2t^*}T_{\ell,i} \neq T_{\ell,i}$, i.e., $x_{2n+2t^*} \neq 1$ and $T_{\ell,i} \neq 0$. Hence, $y_{2n+t^*} = (x_{2n+2t^*} - 1)\alpha_\ell^{n+1-i}$ follows

$$\begin{aligned}y_{2n+t^*} &= (x_{2n+2t^*} - 1)\alpha_\ell^{n+1-i}\\ \Rightarrow[\alpha^{n+1}]_1 &= (x_{2n+2t^*} - 1)^{-1}y_{2n+t^*}[\alpha^i]_1\end{aligned}$$

Therefore, $\mathcal{B}$ can obtain $[\alpha^{n+1}]_1$ with non-trivial probability. This contradicts that no PPT adversary can solve the n-DHEP, hence we can conclude that the binding property holds. □

Lemma 3. The construction of the SVD scheme in Subsection 4.2 is time-locking (Definition 11).

Proof. A simulator of the experiment $\mathsf{EXP}^{\mathsf{SVD\text{-}Time}}_{\mathcal{A},\mathsf{OSign}}$ programs a random oracle RO in a lazy manner. That is, for each query $(\mathsf{pp}, \mathsf{vk}, t)$, if $\mathsf{RO}[(\mathsf{pp}, \mathsf{vk}, t)] = [r_t]_1 \neq \perp$, i.e., any $[r_t]_1$ is already returned for the same query, the simulator returns the same $[r_t]_1$. Otherwise, it samples a random $[r_t]_1 \leftarrow\!\$\ \mathbb{G}_1$, sets $\mathsf{RO}[(\mathsf{pp}, \mathsf{vk}, t)] \leftarrow [r_t]_1$, and returns $[r_t]_1$.

Suppose that algebraic adversaries $\mathcal{A}$ in AGM model output $(\mathbf{T}_1, \ldots, \mathbf{T}_L, [\sigma^*]_1, [W^*_{i,\ell}]_1, T^*_{\ell,i})$ such that the equation in the $\mathsf{SVD.Verify}$ algorithm holds. As indicated in the proof of Lemma 2, $\mathcal{A}$ also outputs coefficients such that the following holds:

$$\begin{aligned}[\sigma^*]_1 &= \Sigma_{t\in[t^*-1]}x_{2n+t^*+t}[\sigma_t]_1\\ [W^*_{i,\ell}]_1 &= y_0[1]_1 + \Sigma_{j\in[2n]/\{n+1\}}y_j[\alpha_\ell^j]_1 + \Sigma_{t\in[t^*]}y_{2n+t}[r_t]_1\end{aligned}$$

Besides, the previous proof also shows $y_{2n+t^*} = (x_{2n+2t^*} - 1)\alpha_\ell^{n+1-i} = -\alpha_\ell^{n+1-i}$. Notably, the coefficient x_{2n+2t^*} is zero becasue $\mathcal{A}$ cannot obtain the t^*-th signature $[\sigma_{t^*}]$ in this experiment.

From the above results, we can construct a PPT adversary $\mathcal{B}$ that breaks the hardness of the n-DHEP (Definition 4). Given $([1]_1, [\alpha_\ell]_1, \ldots, [\alpha_\ell^n]_1, [\alpha_\ell^{n+2}]_1, \ldots, [\alpha_\ell^{2n}]_1, [1]_2, [\alpha_\ell]_2, \ldots, [\alpha_\ell^n]_2)$, $\mathcal{B}$ first samples $(\alpha)_{l\in[L]/\{\ell\}}, (r_t)_{t\in[t^*]}, \mathsf{sk}$ and constructs $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$ using those values. Subsequently, it provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns sign^* and $\mathsf{open}^*_{\ell,i}$ along with their corresponding coefficients. Here, since $y_{2n+t^*} = -\alpha_\ell^{n+1-i}$ holds, $\mathcal{B}$ can compute $[\alpha_\ell^{n+1}]_1 = -y_{2n+t^*}[\alpha_\ell^i]$. This contradicts that no PPT adversary can solve the n-DHEP, hence we can conclude that the time-locking property holds. □

5 Extractable Witness Key Encapsulation Mechanisms (EKEMs) for SVDs

We next show EKEMs for SVDs.

5.1 Definition

The EKEM scheme for SVDs is defined in association with a specific SVD scheme. That is, its encryption algorithm depends on the CRS of the SVD scheme.

We denote a space of the key output by the encryption and decryption algorithms by $\mathcal{K}$.

- $\mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i})) \rightarrow (\mathsf{ct}, k)$: it takes as input a common reference string crs_ℓ, and an instance x consisting of integers $\ell, i \in \mathbb{N}$ and a signing target $T_{\ell,i} \in \mathcal{T}$. It outputs a ciphertext ct and a key $k \in \mathcal{K}$.
- $\mathsf{EKEM.Dec}(\mathsf{ct}, \omega) \rightarrow k^{'}$: it takes as input a ciphertext ct and a witness ω. It outputs a key $k^{'} \in \mathcal{K}$.

The correctness of the EKEM scheme defined for a key space $\mathcal{K}$ and an SVD scheme SVD with a signing target space $\mathcal{T}$ holds if for every $\lambda \in \mathbb{N}$, $n, L, t \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $(\mathbf{T}_l)_{l\in[L]} \in \mathcal{T}^{nL}$,

$$\Pr\Big[k = k^{'}\Big] \geq 1 - \mathsf{negl}(\lambda)$$

holds for every $\ell \in [L]$ and $i \in [n]$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SVD.SetupKey}(1^\lambda)$, $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp}, \mathsf{vk}, t)$, $\mathsf{digest} \leftarrow \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$, $\mathsf{sign} \leftarrow \mathsf{SVD.Sign}(\mathsf{pp}, \mathsf{sk}, t, \mathsf{digest})$, $\mathsf{open}_{\ell,i} \leftarrow \mathsf{SVD.Open}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \ell, i)$, $(\mathsf{ct}, k) \leftarrow \mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}))$, and $k^{'} \leftarrow \mathsf{EKEM.Dec}(\mathsf{ct}, \omega = (\mathsf{sign}, \mathsf{open}_{\ell,i}))$.

Using the stateful signer oracle OSign in Definition 8, we define an extractable security of the EKEM scheme defined for the SVD scheme.

Definition 12 (Extractability of the EKEM scheme)**.** Let EKEM=(Enc, Dec) be an EKEM scheme defined for a key space $\mathcal{K}$ and a binding SVD scheme SVD with a signing target space $\mathcal{T}$. The EKEM scheme is said to be extractable, if for every $\lambda \in \mathbb{N}$, $n, L, t^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $\ell \in [L]$, $i \in [n]$, $T_{\ell,i} \in \mathcal{T}$, algebraic adversary $\mathcal{A}$, polynomial $q(\lambda)$, there exists a PPT extractor $\mathcal{E}$ and a polynomial $p(\lambda)$ such that

$$\Pr\Big[\mathsf{EXP}^{\mathsf{EKEM}}_{\mathcal{A}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i}) = 1\Big] \geq \frac{1}{2} + \frac{1}{q(\lambda)}$$
$$\Rightarrow \Pr\big[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T^*_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1\big] \geq \frac{1}{p(\lambda)}$$

holds, where $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i}) \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{pp}, t^*, x, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$, and the experiment $\mathsf{EXP}^{\mathsf{EKEM}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$ is defined associated with a stateful signer oracle OSign for $\mathsf{pp} = (1^\lambda, n, L)$ and SVD as follows:

$\mathsf{EXP}^{\mathsf{EKEM}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$

1 : OSign is initialized, which outputs vk.

2 : $\mathcal{A}$ calls OSign until the time epoch in state is less than t^*. Let $\mathsf{state}_{\mathcal{A}}$ be the $\mathcal{A}$'s state.

3 : $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t^*)$

4 : $b \in \{0, 1\}$.

5 : $(\mathsf{ct}, k_0) \leftarrow \mathsf{EKEM.Encap}(\mathsf{crs}_\ell, x = (\ell, i, T^*_{\ell,i}))$.

6 : $k_1 \leftarrow\!\!\$\ \mathcal{K}$.

7 : $b^{'} \leftarrow \mathcal{A}^{\mathsf{OSign}(\mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]},\cdot))}(\mathsf{state}_{\mathcal{A}}, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, \mathsf{ct}, k_b)$

8 : Output $b = b^{'}$.

Notably, $\mathcal{A}$ can obtain the t^*-th signature for a digest of the CRSs $(\mathsf{crs}_l)_{l\in[L]}$ and any signing targets $(\mathbf{T}_l)_{l\in[L]}$ from OSign only once, and $\mathcal{E}$ is allowed to call OSign only with $(\mathbf{T}_l)_{l\in[L]}$ submitted by $\mathcal{A}$.

5.2 Construction

We construct the EKEM scheme using the SKE scheme whose key space is $\mathcal{K}$, the SVD scheme constructed in Subsection 4.2, and a random oracle RO mapping an element in $\mathbb{G}_T$ into a key in $\mathcal{K}$.

- $\mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i})) \to (\mathsf{ct}, k)$:
  1. Parse crs_ℓ as $(\mathsf{pp}, \mathsf{crs}_{\ell,1}, \mathsf{crs}_{\ell,2}, \mathsf{crs}_{\ell,T})$.
  2. Extract $[\alpha_\ell^{n+1-i}]_2$ and vk from $\mathsf{crs}_{\ell,2}$.
  3. Extract $[v_{\ell,0}]_T$ and $[v_{\ell,i}]_T$ from $\mathsf{crs}_{\ell,T}$.
  4. Let $[\mathbf{g}]_2$ be $([\alpha_\ell^{n+1-i}]_2, -\mathsf{vk})$ and $[h]_T$ be $T_{\ell,i}[v_{\ell,0}]_T + [v_{\ell,i}]_T$.
  5. Sample $s \in \mathbb{Z}_p$.
  6. Compute $\mathsf{ct} \leftarrow s[\mathbf{g}]_2$.
  7. Compute $z \leftarrow s[h]_T$ and $k \leftarrow \mathsf{RO}(z)$.
  8. Output (ct, k).
- $\mathsf{EKEM.Dec}(\mathsf{ct}, \omega) \to k^{'}$:
  1. Parse ct as $s[\mathbf{g}]_2$.
  2. Parse ω as $([\sigma]_1, [W_{\ell,i}]_1)$.
  3. Compute $z^{'} \leftarrow e(([\sigma]_1, [W_{\ell,i}]_1), s[\mathbf{g}]_2)$.
  4. Output $k^{'} \leftarrow \mathsf{RO}(z^{'})$.

To prove the correctness of the EKEM scheme, we first show that z in the $\mathsf{EKEM.Enc}$ algorithm is equal to $z^{'}$ in the $\mathsf{EKEM.Dec}$ algorithm.

$$\begin{aligned} z &= s[h]_T \\ &= s(T_{\ell,i}[v_{\ell,0}]_T + [v_{\ell,i}]_T) \\ &= e([\sigma]_1, s[\alpha_\ell^{n+1-i}]_2) + e([W_{\ell,i}]_1, -s \cdot \mathsf{vk}) \\ &= e(([\sigma]_1, [W_{\ell,i}]_1), s[\mathbf{g}]_2) \\ &= z^{'} \end{aligned}$$

Therefore, $k = \mathsf{RO}(z) = \mathsf{RO}(z^{'}) = k^{'}$.

Next, we show the extractability of the above EKEM scheme.

Lemma 4. The construction of the EKEM scheme in Subsection 5.2 is extractable (Definition 12).

Proof. Let $[\mathbf{x}]_1$ be a vector connecting $[1]_1$, the $\mathbb{G}_1$ elements in crs_ℓ, and the outputs from the random oracle by the construction of the SVD scheme in Subsection 4.2 and the signer oracle OSign, i.e., $[\mathbf{x}]_1 \leftarrow ([1]_1, (\alpha_\ell^i)_{i\in[2n]/\{n+1\}}, ([r_t]_1)_{t\in[t^*]}, ([\sigma_t]_1)_{t\in[t^*]})$[4]. Let $[\mathbf{y}]_2$ be a vector connecting $[1]_2$ and the $\mathbb{G}_2$ elements in crs_ℓ, i.e., $[\mathbf{y}]_2 \leftarrow ([1]_2, ([\alpha_\ell^i]_2)_{i\in[N]}, \mathsf{vk})$. Let $[\mathbf{z}]_T$ be a vector connecting $[1]_T$ and the $\mathbb{G}_T$ elements in crs_ℓ, i.e., $[\mathbf{z}]_T \leftarrow ([1]_T, [v_{\ell,0}]_T, ([v_{\ell,i}]_T)_{i\in[n]})$. A vector $(\mathsf{sk}, \alpha_\ell, r_t)$ is denoted by $\mathbf{td} \in \mathbb{F}_p^3$. We define functions $\Psi(\mathbf{X})$, $\Omega(\mathbf{X})$, and $\Phi(\mathbf{X})$ such that $[\Psi(\mathbf{td})]_1 = [\mathbf{x}]_1$, $[\Omega(\mathbf{td})]_2 = [\mathbf{y}]_2$, and $[\Phi(\mathbf{td})]_T = [\mathbf{z}]_T$ holds, respectively. Similary, $\mathbf{G}(\mathbf{X})$ and $H(\mathbf{X})$ are defined as functions such that $[\mathbf{G}(\mathbf{td})]_2 = [\mathbf{g}]_2$ and $[H(\mathbf{td})]_T = [h]_T$ hold.

We construct an extractor $\mathcal{E}$ as follows. Given $(\mathsf{pp} = (1^\lambda, n, L), t, x = (\ell, i, T^*_{\ell,i}), (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$, $\mathcal{E}$ first computes (ct, k_0), samples $k_1 \leftarrow\!\!\$\ \mathcal{K}$ and $b \leftarrow\!\!\$\ \{0,1\}$. It then provides ct and $\mathbf{k}_b$ for $\mathcal{A}$, which returns $b^{'} \in \{0,1\}$. For each $\mathcal{A}$'s oracle access to OSign, $\mathcal{E}$ forwards the output of that oracle without modification. $\mathcal{E}$ also programs the random oracle RO for $\mathcal{A}$ as follows.

1. If $\mathsf{RO}[z] = k \neq \perp$, return k.
2. Else if z is equal to $s[h]_T$, which is generated in the $\mathsf{EKEM.Enc}$ algorithm, set $\mathsf{RO}[s[h]_T] = k_0$ and return k_0.
3. Else, generate a random key $k^* \in \mathcal{K}$, set $\mathsf{RO}[z] = k^*$, and return k^*.

Since $\mathcal{A}$ is an algebraic adversary, $\mathcal{A}$ should outputs z with coefficient matrixes $\mathbf{P}$, $\mathbf{Q}_0$, $\mathbf{Q}_1$, $\mathbf{R}$, and a scalar S such that $z = [\Psi(\mathsf{td}) \cdot (\Omega(\mathsf{td})\mathbf{P} + \mathbf{G}(\mathsf{td})\mathbf{Q}_0 + s\mathbf{G}(\mathsf{td})\mathbf{Q}_1) + \Phi(\mathsf{td})\mathbf{R} + SH(\mathsf{td})]_T$ holds. $\mathcal{E}$ outputs $\perp$ if $\mathcal{A}$ does not output any coefficient matrixes such that $\Psi(\mathbf{X}) \cdot Y\mathbf{G}(\mathbf{X})\mathbf{Q}_1 = YH(\mathbf{X})$ holds for any $\mathbf{X}$ and Y.

We show that the above $\mathcal{E}$ can extract the witness $\mathbf{td}$ from $\mathcal{A}$ with a non-negligible probability if the polynomial $q(\lambda)$ is non-negligible. First, if the output

[4]For the same reason described in the proof of Lemma 2, the elements generated in the $\mathsf{SVD.SetupCRS2}$ algorithm are not contained in $[\mathbf{x}]_1$.

from $\mathcal{E}$ is not $\perp$, $\mathcal{E}$ can output ω that satisfies the required condition. Since $\Psi(\mathbf{X}) \cdot Y\mathbf{G}(\mathbf{X})\mathbf{Q}_1 = YH(\mathbf{X})$ holds, $[\mathbf{f}]_1 = \Psi(\mathsf{td})\mathbf{Q}_1 = [\mathbf{x}]_1\mathbf{Q}_1$ makes the equation $e([\mathbf{f}]_1, [\mathbf{g}]_2) = [h]_T$ true. From the definitions of $[\mathbf{g}]_2$ and $[h]_T$, $[\mathbf{f}]_1$ can be parsed as $([\sigma]_1, [W_{\ell,i}]_1)$, which satisfies $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, [\sigma]_1, \ell, i, T^*_{\ell,i}, [W_{\ell,i}]_1) = 1$.

Next, the probability that the output of $\mathcal{E}$ is $\perp$ is negligible. Let Hit be an event that $\mathcal{A}$ queries RO with the input $z = s[h]_T$.

If Hit does not occur, i.e., $\mathsf{RO}[s[h]_T] = \perp$, edv cannot distinguish k_b from a random key in $\mathcal{K}$. Therefore, the following holds.

$$\begin{aligned}
\frac{1}{2} + \frac{1}{q(\lambda)} &\leq \Pr\left[\mathsf{EXP}^{\mathsf{EKEM}}_{\mathcal{A}}(1^\lambda, n, L, t, \ell, i, T_{\ell,i}) = 1\right] \\
&= \Pr\left[b = b'\right] \\
&= \Pr\left[b = b' | \mathsf{Hit}\right] \Pr[\mathsf{Hit}] + \Pr\left[b = b' | \bar{\mathsf{Hit}}\right] \Pr\left[\bar{\mathsf{Hit}}\right] \\
&= \Pr\left[b = b' | \mathsf{Hit}\right] (1 - \Pr[\mathsf{Hit}]) + \frac{1}{2} \Pr\left[\bar{\mathsf{Hit}}\right] \\
&\leq (1 - \Pr\left[\bar{\mathsf{Hit}}\right]) + \frac{1}{2} \Pr\left[\bar{\mathsf{Hit}}\right] \\
&= 1 - \frac{1}{2} \Pr\left[\bar{\mathsf{Hit}}\right]
\end{aligned}$$

Hence, $\Pr\left[\bar{\mathsf{Hit}}\right] \leq 1 - \frac{2}{q(\lambda)}$.

We next consider a case where Hit occurs but $\mathcal{A}$ outputs no coefficient matrixes such that $\Psi(\mathbf{X}) \cdot Y\mathbf{G}(\mathbf{X})\mathbf{Q}_1 = YH(\mathbf{X})$ holds. As $z = s[h]_T$, (td, s) is one root of a non-zero polynomial $\Gamma(\mathbf{X}, Y) = \Psi(\mathbf{X}) \cdot (\Omega(\mathbf{X})\mathbf{P} + \mathbf{G}(\mathbf{X})\mathbf{Q}_0 + s\mathbf{G}(\mathbf{X})\mathbf{Q}_1) + \Phi(\mathbf{X})\mathbf{R} + SH(\mathbf{X}) - YH(\mathbf{X})$. To show that the adversary cannot output the non-zero polynomial Γ, we prove the following lemma.

Claim 3. For every $n, d \in \mathbb{N}$, $s_1, \ldots, s_n \leftarrow\!\!\$\ \mathbb{F}_p$, algebraic adversary $\mathcal{A}$, it holds that

$$\Pr\left[\mathbf{F}(X_1, \ldots, X_n) \neq 0 \wedge \mathbf{F}(s_1, \ldots, s_n) = 0 : \mathbf{F} \leftarrow \mathcal{A}([1]_T, ([s_i]_T, \ldots, [s_i^d]_T)_{i\in[n]})\right] < \mathsf{negl}(\lambda)$$

, where $\mathbf{F}$ is a non-zero n-variable polynomial in which the maximum degree of X_i is less than $d+1$ for every $i \in [n]$.

Proof. Suppose an algebraic adversary $\mathcal{A}$ that outputs the non-zero polynomial $\mathbf{F}$ whose root is $(s_1, \ldots, s_n)$ with non-trivial probability, we can construct an adversary $\mathcal{B}$ that breaks the hardness of the d-DLOG (Definition 2).

If $n = 1$, given $([1]_T, [s_1]_T, \ldots, [s_1^d]_T)$, $\mathcal{B}$ provides $([1]_T, [s_1]_T, \ldots, [s_1^d]_T)$ for $\mathcal{A}$, which returns a non-zero polynomial $\mathbf{F}(X_1)$ whose one root is s_1. Therefore, $\mathcal{B}$ can find s_1 by solving the equation $\mathbf{F}(X_1) = 0$.

If $\mathcal{B}$ can find s_1 in the case of $n-1$, i.e., $\mathcal{A}$ outputs a non-zero $(n-1)$-variable polynomial whose one root is $(s_1, \ldots, s_{n-1})$, $\mathcal{B}$ can also find s_1 in the case of n as follows. Given $([1]_T, [s_1]_T, \ldots, [s_1^d]_T)$, $\mathcal{B}$ samples $s_2, \ldots, s_{n-1}, z \leftarrow\!\!\$\ \mathbb{F}_p$ and sets $[s_n^j]$ to $[s_n^j]_T = z^j[s_1^j]_T$ for every $j \in [d]$. Notably, $s_n = zs_1$ and a random s_n are statically indistinguishable. $\mathcal{B}$ provides $([1]_T, ([s_i]_T, \ldots, [s_i^d]_T)_{i\in[n]})$ for

$\mathcal{A}$, which returns a non-zero n-variable polynomial $\mathbf{F}(X_1, \ldots, X_n)$ whose one root is $(s_1, \ldots, s_n)$.

- When $\mathbf{F}(s_1, \ldots, s_{n-1}, X_n) = 0$, $\mathcal{B}$ organizes $\mathbf{F}(X_1, \ldots, X_n)$ in the variable X_n, i.e., $\mathbf{F}(X_1, \ldots, X_n) = \mathbf{C}_0(X_1, \ldots, X_{n-1}) + \mathbf{C}_1(X_1, \ldots, X_{n-1})X_n + \cdots + \mathbf{C}_d(X_1, \ldots, X_{n-1})X_n^d$. As $\mathbf{F}(X_1, \ldots, X_n) \neq 0$, there is at least one non-zero polynomial $C_j(X_1, \ldots, X_{n-1})$ in $(\mathbf{C}_i)_{i\in\{0,1,\ldots,n\}}$. At the same time, since $\mathbf{F}(s_1, \ldots, s_{n-1}, X_n) = 0$, $C_j(s_1, \ldots, s_{n-1}) = 0$ also holds. Therefore, $\mathcal{B}$ can find s_1 from $C_j(X_1, \ldots, X_{n-1})$.
- When $\mathbf{F}(s_1, \ldots, s_{n-1}, X_n) \neq 0$, $\mathcal{B}$ can find a root $s_n = zs_1$ by solving the equation $\mathbf{F}(s_1, \ldots, s_{n-1}, X_n) = 0$. Therefore, $\mathcal{B}$ can output $s_1 = \frac{s_n}{z}$.

Hence, $\mathcal{B}$ can break the hardness of d-DLOG in the case of n.

The above argument shows that if $\mathcal{A}$ outputs a non-zero polynomial $\mathbf{F}$ whose root is $(s_1, \ldots, s_n)$ with non-trivial probability, $\mathcal{B}$ can break the hardness of the d-DLOG. Since the hardness of the d-DLOG is assumed, the probability that $\mathcal{A}$ outputs a non-zero polynomial $\mathbf{F}$ whose root is $(s_1, \ldots, s_n)$ with non-trivial probability is negligible. □

By Claim 3, $\Pr[\Gamma(\mathsf{td}, s) = 0] < \mathsf{negl}(\lambda)$, i.e., $\Pr\left[\bot = \mathcal{E}^{\mathcal{A},\mathsf{OSign}}(\mathsf{pp}, t, x, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})\right] < \mathsf{negl}(\lambda)$ holds. Therefore,

$$\begin{aligned}
&\Pr\left[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1\right] \\
&= 1 - \Pr\left[\bot = \mathcal{E}^{\mathcal{A},\mathsf{OSign}}(\mathsf{pp}, t^*, x, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})\right] \\
&= 1 - \mathsf{negl}(\lambda) \\
&\geq \frac{1}{p(\lambda)}
\end{aligned}$$

holds for some polynomial $p(\lambda)$. □

6 Extractable Witness Encryption for SVDs

We are ready to present a construction of the EWE for SVDs. Similar to the EKEM scheme, it is defined for a specific SVD scheme, which employs the CRS of the SVD scheme.

6.1 Definition

- $\mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), m) \to \mathsf{ct}_x$: it takes as input a common reference string crs_ℓ, an instance x consisting of integers $\ell, i \in \mathbb{N}$ and a signing target $T_{\ell,i} \in \mathcal{T}$, and a message $m \in \mathcal{M}$. It outputs a ciphertext ct_x.
- $\mathsf{EWE.SignDigest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}) \to \mathsf{sign}_t$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, and signing targets $(\mathbf{T}_l)_{l\in[L]}$, each of which is in $\mathcal{T}^n$. It outputs a signature sign_t generated by a trusted stateful signer for the time epoch t.

- $\mathsf{EWE.Dec}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \mathsf{sign}_t, \mathsf{ct}_x) \to m$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, signing targets $(\mathbf{T}_l)_{l\in[L]}$, a signature sign_t, and a ciphertext ct_x, . It outputs a message m or $\perp$.

The EWE scheme defined for a message space $\mathcal{M}$ and an SVD scheme SVD with a signing target space $\mathcal{T}$ is correct if for every $\lambda \in \mathbb{N}$, $n, L, t \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $(\mathbf{T}_l)_{l\in[L]} \in \mathcal{T}^{nL}$, and $m_x \in \mathcal{M}$,

$$\Pr\left[\mathsf{EWE.Dec}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \mathsf{sign}_t, \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x, m_x)) = m_x\right] \geq 1-\mathsf{negl}(\lambda)$$

holds for every $x \in [L]\times[n]\times\mathcal{T}$, where $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SVD.SetupKey}(1^\lambda)$, $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp}, \mathsf{vk}, t)$, and $\mathsf{sign}_t \leftarrow \mathsf{EWE.SignDigest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$.

Its extractable security is defined with the oracle OSign.

Definition 13 (Extractability of the EWE scheme)**.** Let $\mathsf{EWE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{SignDigest}, \mathsf{Dec})$ be an EWE scheme defined for a message space $\mathcal{M}$ and a binding SVD scheme SVD with signing target space $\mathcal{T}$. The EWE scheme is said to be extractable, if for every $\lambda \in \mathbb{N}$, $n, L, t^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $\ell \in [L]$, $i \in [n]$, $T_{\ell,i} \in \mathcal{T}$, algebraic adversary $\mathcal{A}$, polynomial $q(\lambda)$, there exists a PPT extractor $\mathcal{E}$ and a polynomial $p(\lambda)$ such that

$$\Pr\left[\mathsf{EXP}^{\mathsf{EWE}}_{\mathcal{A}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i}) = 1\right] \geq \frac{1}{2} + \frac{1}{q(\lambda)}$$
$$\Rightarrow \Pr\left[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T^*_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1\right] \geq \frac{1}{p(\lambda)}$$

holds, where $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i}) \leftarrow \mathcal{E}^{\mathcal{A},\mathsf{OSign}}(\mathsf{pp}, t^*, x, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, m_0, m_1)$, and the experiment $\mathsf{EXP}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$ is defined associated with a stateful signer oracle OSign for $\mathsf{pp} = (1^\lambda, n, L)$ and SVD as follows:

$\mathsf{EXP}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T^*_{\ell,i})$

1 : OSign is initialized, which outputs vk.
2 : $\mathcal{A}$ calls OSign until the time epoch in state is less than t^*. Let $\mathsf{state}_{\mathcal{A}}$ be the $\mathcal{A}$'s state.
3 : $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{SVD.SetupCRS}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t^*)$
4 : $(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{state}_{\mathcal{A}}, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$.
5 : $b \in \{0, 1\}$.
6 : $\mathsf{ct}_x \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T^*_{\ell,i}), m_b)$.
7 : $b' \leftarrow \mathcal{A}^{\mathsf{OSign}(\mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, \cdot))}(\mathsf{state}_{\mathcal{A}}, \mathsf{ct}_x)$
8 : Output $b = b'$.

Notably, $\mathcal{A}$ can obtain the t^*-th signature for a digest of the CRSs $(\mathsf{crs}_l)_{l\in[L]}$ and any signing targets $(\mathbf{T}_l)_{l\in[L]}$ from OSign only once, and $\mathcal{E}$ is allowed to call OSign only with $(\mathbf{T}_l)_{l\in[L]}$ submitted by $\mathcal{A}$.

6.2 Construction

The following construction of the EWE scheme is based on the EKEM scheme in 5.2 and a SWE scheme whose message space is $\mathcal{M}$ and a key space $\mathcal{K}$ is the same as that of the EKEM scheme.

- $\mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), m) \rightarrow \mathsf{ct}_x$:
  1. Compute $(\mathsf{ct}_0, k) \leftarrow \mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x)$.
  2. Compute $\mathsf{ct}_1 \leftarrow \mathsf{SKE.Enc}(k, m)$.
  3. Output $\mathsf{ct}_x \leftarrow (\mathsf{ct}_0, \mathsf{ct}_1)$.
- $\mathsf{EWE.SignDigest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}) \rightarrow \mathsf{sign}_t$:
  1. Compute $\mathsf{digest} \leftarrow \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$.
  2. Output $\mathsf{sign}_t \leftarrow \mathsf{OSign}(\mathsf{digest})$.
- $\mathsf{EWE.Dec}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \mathsf{sign}_t, \mathsf{ct}_x) \rightarrow m$:
  1. Parse ct_x as $(\mathsf{ct}_0, \mathsf{ct}_1)$.
  2. Compute $\mathsf{open}_{\ell,i} \leftarrow \mathsf{SVD.Open}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \ell, i)$.
  3. Compute $k^{'} \leftarrow \mathsf{EKEM.Dec}(\mathsf{ct}_0, \omega = (\mathsf{sign}_t, \mathsf{open}_{\ell,i}))$.
  4. Compute $m^{'} \leftarrow \mathsf{SKE.Dec}(k^{'}, \mathsf{ct}_1)$.
  5. Output $m^{'}$.

The correctness of the SVD, EKEM, and SKE schemes ensures the correctness of the EWE scheme as follows:

$$\begin{aligned} m^{'} &= \mathsf{SKE.Dec}(\mathsf{EKEM.Dec}(\mathsf{ct}_0, (\mathsf{sign}_t, \mathsf{open}_{\ell,i})), \mathsf{ct}_1) \\ &= \mathsf{SKE.Dec}(k, \mathsf{SKE.Enc}(k, m)) \\ &= m \end{aligned}$$

The EWE scheme for SVDs is trust-scalable.

Theorem 1. The construction of the EWE scheme for SVDs in Subsection 6.2 is trust-scalable (Definition 9).

Proof. In that construction of the EWE scheme, the signer only needs to return the output of the signer oracle OSign. As proved in Lemma 1, it requires the signer to spend only constant computational and communication costs bounded by $\mathsf{poly}(\lambda)$. Besides, OSign can be called only once for each time epoch t. Therefore, the signer's costs in the EWE scheme are bounded by $\mathsf{poly}(\lambda)$. □

From the extractability of the EKEM scheme and the EAV security of the SKE scheme, we prove that the above construction satisfies the extractability of the EWE scheme.

Theorem 2. The construction of the EWE scheme in Subsection 6.2 is extractable (Definition 13).

Proof. Suppose a PPT adversary $\mathcal{A}$ that outputs 1 in $\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}$ with probability greater than or equal to $\frac{1}{2} + \frac{1}{q(\lambda)}$ for some polynomial $q(\lambda)$, we can construct a PPT adversary $\mathcal{B}$ with which the experiment $\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{B},\mathsf{OSign}}$ ouputs 1. $\mathcal{B}$ forwards the $\mathcal{A}$'s oracle access to OSign and returns its output without modification. When the time epoch in state is t^*, $\mathcal{B}$ receives $((\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, \mathsf{ct}_0, k)$ as defined in $\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{A},\mathsf{OSign}}$, where $b_0 \in \{0,1\}$ is randomly sampled in that experiment. It provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns m_0 and m_1. $\mathcal{B}$ then samples $b_1 \leftarrow\!\$\ \{0,1\}$ and computes $\mathsf{ct}_1 \leftarrow \mathsf{SKE.Enc}(k, m_{b_1})$ and provides $\mathsf{ct}_x = (\mathsf{ct}_0, \mathsf{ct}_1)$ for $\mathcal{A}$. $\mathcal{A}$ might submit $(\mathbf{T}_l)_{l\in[L]}$ to have $\mathcal{B}$ call the oracle $\mathsf{OSign}(\mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, \cdot)$. $\mathcal{A}$ finally returns b_2. $\mathcal{B}$ returns 0 if $b_1 = b_2$ and 1 otherwise.

Here, we define a variance of the experiment $\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}$ as $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$, where the key k is sampled randomly from $\mathcal{K}$. In other words. one of the challenge messages is encrypted under a random key $k \leftarrow\!\$\ \mathcal{K}$ in $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$. From the EAV security of the SKE scheme, we prove that $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$ outputs 1 with less than or equal to trivial probability.

Claim 4. For every $\lambda \in \mathbb{N}$, $n, L, t^* \in \mathbb{N}$ bounded by $\mathsf{poly}(\lambda)$, $\ell \in [L]$, $i \in [n]$, $T_{\ell,i} \in \mathcal{T}$, algebraic adversary $\mathcal{A}$, the following holds:

$$\Pr\left[\mathsf{EXP}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ that outputs 1 in $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$ with non-trivial probability. We can construct a PPT adversary $\mathcal{B}$ that outputs 1 in the experiment $\mathsf{Exp}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{B}}$. $\mathcal{B}$ initialize the signer oracle OSign, which generates signing and verifying keys $(\mathsf{sk}, \mathsf{vk})$ and sets state, and returns a signature sign_t for each $\mathcal{A}$'s query. Given 1^λ, $\mathcal{B}$ sets $\mathsf{pp} = (1^\lambda, n, L)$, generates $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$, and provides $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ for $\mathcal{A}$, which returns m_0 and m_1. $\mathcal{B}$ submits the same challenge messages in $\mathsf{Exp}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{B}}$. $\mathcal{B}$ then receives the SKE encryption ct_1 for m_b, where $b \leftarrow\!\$\ \{0,1\}$ in $\mathsf{Exp}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{B}}$. $\mathcal{B}$ computes $(\mathsf{ct}_0, k) \leftarrow \mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x)$ and provides $\mathsf{ct}_x = (\mathsf{ct}_0, \mathsf{ct}_1)$ for $\mathcal{A}$, which returns $(\mathbf{T}_l)_{l\in[L]}$. It then $\mathsf{digest} \leftarrow \mathsf{SVD.Digest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]})$ and $\mathsf{sign}_t \leftarrow \mathsf{OSign}(\mathsf{digest})$, which is provided for $\mathcal{A}$. $\mathcal{B}$ forwards the final $\mathcal{A}$'s output b'. Since $\mathcal{B}$ completely simulates $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$, the following holds:

$$\Pr\left[\mathsf{EXP}^{\mathsf{SKE\text{-}EAV}}_{\mathcal{B}}(1^\lambda) = 1\right] = \Pr\left[\mathsf{EXP}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1\right] > \frac{1}{2} + \mathsf{negl}(\lambda)$$

This is a contradiction to the EAV security of the SKE scheme. Therefore, the claim holds. □

The experiment $\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{B},\mathsf{OSign}}$ for $b_0 = 0$ and $b_0 = 1$, respectively, simulates $\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}$ and $\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}$. Hecen, by Claim 4, there exists a polynomial $q^{'}(\lambda)$ such that the following holds:

$$
\begin{aligned}
&\Pr\Big[\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{B},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1\Big] \\
=&\Pr[b_0 = 0]\Pr\Big[\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1 | b_0 = 0\Big] \\
+&\Pr[b_0 = 1]\Pr\Big[\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1 | b_0 = 1\Big] \\
=&\frac{1}{2}\Pr\Big[\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1 | b_0 = 0\Big] \\
+&\frac{1}{2}(1 - \Pr\Big[\mathsf{Exp}^{\widetilde{\mathsf{EWE}}}_{\mathcal{A},\mathsf{OSign}}(1^\lambda, n, L, t^*, \ell, i, T_{\ell,i}) = 1 | b_0 = 1\Big]) \\
\geq&\frac{1}{2}(\frac{1}{2} + \frac{1}{q(\lambda)} + 1 - \frac{1}{2} - \mathsf{negl}(\lambda)) \\
=&\frac{1}{2} + \frac{1}{2q(\lambda)} - \mathsf{negl}(\lambda) \\
\geq&\frac{1}{q^{'}(\lambda)}
\end{aligned}
$$

Therefore, the extractability of the EKEM scheme ensures that there exists a PPT extractor $\mathcal{E}_1$ such that

$$\Pr\big[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1\big] \geq \frac{1}{p(\lambda)}$$

holds for some polynomial $p(\lambda)$, where $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i}) \leftarrow \mathcal{E}_1^{\mathcal{B},\mathsf{OSign}}(\mathsf{pp}, t^*, x = (\ell, i, T^*_{\ell,i}), (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$. Using $\mathcal{E}_1$, we can construct a PPT extractor $\mathcal{E}_2$ that returns the same output as that of $\mathcal{E}_1$ from pp, t^*, x, $(\mathsf{crs}_l)_{l\in[L]}$, $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$, m_0, and m_1 with oracle access to $\mathcal{A}$ and OSign. $\mathcal{E}_2$ forwards the given pp, t^*, x, $(\mathsf{crs}_l)_{l\in[L]}$, and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$ to $\mathcal{A}$, which returns $\omega = (\mathsf{sign}^*, \mathsf{open}^*_{\ell,i})$. During the execution of $\mathcal{E}_1$, it should be able to call the oracle $\mathcal{B}$. Therefore, $\mathcal{E}_2$ simulates $\mathcal{B}$ by employing $\mathcal{A}$ as described above. In this way, $\mathcal{E}_2$ can extract $\omega = (\mathsf{sign}^*, \mathsf{open}^*_{\ell,i})$ such that

$$\Pr\big[\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, T^*_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1\big] \geq \frac{1}{p(\lambda)}$$

holds. Therefore, the extractability of the EWE scheme is proven. □

6.3 Threshold Multiple Signers

The above EWE scheme can be easily extended to support threshold signatures. That is, an encryptor specifies N verifying keys and an decryptor employs more than or equal to $\theta \in [N]$ signatures to decrypt each ciphertext. We propose two approaches to achieve that.

Approach 1: the first approach is that the encryptor splits a message $\mathbf{m}$ into θ-of-N shares $(\mathbf{m}_1^{'}, \ldots, \mathbf{m}_N^{'})$ using Shamir's secret sharing [Sha79], and encrypts $\mathbf{m}_i^{'}$ for the i-th verifying key vk_i. If more than or equal to θ signatures are provided, the decryptor can obtain more than θ secret shares by employing each signatre to decrypt the corresponding ciphertext, and recovers $\mathbf{m}$ from those shares.

Approach 2: in the second approach, the N signers perform distributed key generation (DKG) protocol [GJKR07] to generate linear secret shares of a main siging key such that $\mathsf{sk}_M = l_{i_1}\mathsf{sk}_{i_1} + \cdots + l_{i_\theta}\mathsf{sk}_{i_\theta}$ holds with some coefficients $l_{i_1}, \ldots, l_{i_\theta} \in \mathbb{F}_p$. Its main verifying key is $\mathsf{vk}_M = [\mathsf{sk}_M]_2$. Then, the encryptor encrypts a message $\mathbf{m}$ for vk_M, which is the same as the flow in the single-signer case. On the other hand, the decryptor with more than or equal to θ signatures, e.g., $([\sigma_{i_1}]_1, \ldots, [\sigma_{i_\theta}]_1)$, can compute a signature for vk_M as $[\sigma_M]_1 = \mathsf{sk}_M[d]_1 = (l_{i_1}\mathsf{sk}_{i_1} + \cdots + l_{i_\theta}\mathsf{sk}_{i_\theta})[d]_1 = l_{i_1}[\sigma_{i_1}]_1 + \cdots + l_{i_\theta}[\sigma_{i_\theta}]_1$. The resulting $[\sigma_M]_1$ allows the decryptor to decrypt the ciphertext.

Comparison between Approaches 1 and 2: Comparing Approaches 1 and 2, the former increases the computational and communication overhead for the encryptor and decryptor but maintains the same costs for the signer, whereas the latter does the opposite. In other words, they have opposite trade-offs. However, these approaches can be employed together. When each of M signers forms a cluster with a small communication overhead because they are geographically close to each other, i.e., there are N/M clusters, each M signers adopts Approach 2 and prepares N/M verifying keys in total. The encryptor generates N/M encryptions of each secret share in the same manner as Approach 1. We can control the above trade-off by adjusting the value of M.

7 Trust-Scalable One-Time Programs

As a significant application of our EWE scheme, we present TSOTPs with a signer oracle, where the signer's computational and communication complexities are constant irrespective of the input size n and the number of evaluated OTPs L.

7.1 Definition

We define the TSOTP scheme for a circuit class $\mathcal{C}_{n,m}$ as a tuple of the following algorithms.

- $\mathsf{TSOTP.Setup}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t) \to (\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$: it takes as input a public parameter pp, consisting of a security parameter 1^λ and integers $n, L \in \mathbb{N}$ bounded by a polynomial on λ, a verifying key vk, and a time epoch t. It outputs each party's common reference string crs_l and its trapdoor td_l.
- $\mathsf{TSOTP.Compile}(\mathsf{crs}_\ell, \ell, C_\ell) \to \mathsf{CC}_\ell$: it takes as input a common reference string crs, consisting of a security parameter 1^λ and integers $n, L \in \mathbb{N}$

bounded by a polynomial on λ, a verifying key vk, a time bound t, an integer $\ell \in [L]$, and a circuit $C_\ell \in \mathcal{C}_{n,m}$. It outputs a compiled circuit CC_ℓ.

- $\mathsf{TSOTP.SignDigest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}) \to \mathsf{sign}_t$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, and signing targets $(\mathbf{T}_l)_{l\in[L]} \in \{0,1\}^{n\times L}$. It outputs a signature sign_t.
- $\mathsf{TSOTP.Eval}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \mathsf{sign}_t, \mathsf{CC}_\ell) \to y$: it takes as input common reference strings $(\mathsf{crs}_l)_{l\in[L]}$, singing targets $(\mathbf{T}_l)_{l\in[L]} \in \{0,1\}^{n\times L}$, a signature sign_t, and a compiled circuit CC_ℓ. It outputs the evaluation result y.

The TSOTP scheme is correct if for every $\lambda \in \mathbb{N}$, $n, L, t \in \mathbb{N}$ bounded by a polynomial on λ, a verifying key vk, an integer $\ell \in [L]$, and a circuit C_ℓ,

$$\Pr\big[C_\ell(\mathbf{T}_\ell) = \mathsf{TSOTP.Eval}(\mathsf{crs}_\ell, \mathsf{CC}_\ell, \ell, \mathsf{OTP.SignDigest}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}))\big] = 1-\mathsf{negl}(\lambda)$$

, where $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{TSOTP.Setup}(\mathsf{pp} = (1^\lambda, n, L), \mathsf{vk}, t)$ and $\mathsf{CC}_\ell \leftarrow \mathsf{TSOTP.Compile}(\mathsf{crs}_\ell, \ell, C_\ell)$.

In its security definition, a view of an adversary given a compiled circuit CC_ℓ should be simulated without C and CC_ℓ by employing an one-time evaluator oracle that returns the output of C_ℓ on arbitrarily input only once. The adversary and the simulator are allowed to make queries to a stateful signer oracle OSign (Definition 8), which outputs a signature for a digest of the given signing targets.

Definition 14 (One-Time Security of the OTP scheme)**.** Let $\mathsf{TSOTP} = (\mathsf{Gen}, \mathsf{SignDigest}, \mathsf{Eval})$ be an TSOTP scheme for a circuit class $\mathcal{C}_{n,m}$. The TSOTP scheme is said to be one-time secure with a stateful signer oracle, if for every security parameter $\lambda \in \mathbb{N}$, $n, n', L, t^* \in \mathbb{N}$ bounded by a polynomial on λ, stateful signer oracle OSign defined for $\mathsf{pp}' = (1^\lambda, n', L)$ and an SVD scheme with a singing target space $\mathcal{T} = \{0,1\}$, integer $\ell \in [L]$, and circuit $C_\ell \in \mathcal{C}_{n,m}$, there exists a PPT simulator $\mathsf{TSOTP.Sim}$ such that no PPT adversary $\mathcal{A}$ can distinguish between the following distributions with a non-trivial probability:

$$\begin{aligned}&\{\mathcal{A}^{\mathsf{OSign}}(\mathsf{pp}, \ell, t^*, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, \mathsf{CC}_\ell)\}\\ \approx&\{\mathsf{TSOTP.Sim}^{\mathsf{OOTE}_{C_\ell}(\cdot),\mathsf{OSign}}(\mathsf{pp}, \ell, t^*, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]}, 1^{|C_\ell|})\}\end{aligned}$$

,where $\mathsf{pp} = (1^\lambda, n, L)$, vk is a verifying key of OSign, $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]} \leftarrow \mathsf{TSOTP.Setup}(\mathsf{pp}, \mathsf{vk}, t)$, and $\mathsf{CC}_\ell \leftarrow \mathsf{TSOTP.Compile}(\mathsf{crs}_\ell, \ell, C_\ell)$. Here, $\mathcal{A}$ can make queries to OSign until the time epoch t in its state is less than or equal to t^*, and $\mathsf{OTP.Sim}$ can call an one-time evaluator oracle that on input $\mathbf{T}_\ell \in \{0,1\}^n$ outputs the evaluation result $C_\ell(\mathbf{T}_\ell)$ only once.

7.2 Construction

We present a construction of the TSOTP scheme from a garbled circuit GC scheme, an SVD scheme constructed in Subsection 4.2, an EKEM scheme defined

for the SVD scheme with a key space $\mathcal{K}$, an EWE scheme defined for the SVD scheme with a message space $\mathcal{M}$, a stateful signer oracle OSign defined for $\mathsf{pp}^{'} = (1^{\lambda}, n+1, L)$ and the SVD scheme. Let $\mathsf{size}_{\mathsf{ct}}$ be the size of the EWE encryption of a message in $\mathcal{M}$. The construction also employs a random oracle RO that maps $\mathcal{K} \times [L] \times [n] \times \{0,1\}$ into a bit string $r \in \{0,1\}^{\mathsf{size}_{\mathsf{ct}}}$.

- $\mathsf{TSOTP.Setup}(\mathsf{pp} = (1^{\lambda}, n, L), \mathsf{vk}, t) \rightarrow (\mathsf{crs}_l, \mathsf{td}_l)_{l \in [L]}$:
  1. Let $\mathsf{pp}^{'} \leftarrow (1^{\lambda}, n+1, L)$.
  2. Output $(\mathsf{crs}_l, \mathsf{td}_l)_{l \in [L]} \leftarrow \mathsf{SVD.Setup}(\mathsf{pp}^{'}, \mathsf{vk}, t)$.
- $\mathsf{TSOTP.Compile}(\mathsf{crs}_\ell, \ell, C_\ell) \rightarrow \mathsf{CC}_\ell$:
  1. Extract $\mathsf{pp}^{'} = (1^{\lambda}, n+1, L)$ from crs_ℓ.
  2. Generate a garbled circuit and its garbled wires of all input bits $(\widetilde{C}, \mathsf{state} = (\mathsf{lab}_{i,b})_{i \in [n], b \in \{0,1\}}) \leftarrow \mathsf{GC.Circuit}(1^{\lambda}, C_\ell)$.
  3. For $i \in [n]$ and $b \in \{0,1\}$, compute $\mathsf{ct}_{\ell,i,b} \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, b), \mathsf{lab}_{i,b})$.
  4. Compute $(\mathsf{ct}_{\ell,n+1}, k) \leftarrow \mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x = (\ell, n+1, 0))$.
  5. For $i \in [n]$ and $b \in \{0,1\}$, compute $\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b} \leftarrow \mathsf{ct}_{\ell,i,b} \oplus \mathsf{RO}(k, \ell, i, b)$.
  6. Let $\mathsf{CC}_\ell \leftarrow (\ell, \widetilde{C}_\ell, (\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b})_{i \in [n], b \in \{0,1\}}, \mathsf{ct}_{\ell,n+1})$.
  7. Output CC_ℓ.
- $\mathsf{TSOTP.SignDigest}((\mathsf{crs}_l)_{l \in [L]}, (\mathbf{T}_l)_{l \in [L]}) \rightarrow \mathsf{sign}_t$:
  1. For $l \in [L]$, assert that $\mathbf{T}_l \in \{0,1\}^n$ holds.
  2. For $l \in [L]$, let $\mathbf{T}_l^{'} \leftarrow \mathbf{T}_l || 0$.
  3. Output $\mathsf{sign}_t \leftarrow \mathsf{EWE.SignDigest}((\mathsf{crs}_l)_{l \in [L]}, (\mathbf{T}_l^{'})_{l \in [L]})$.
- $\mathsf{TSOTP.Eval}((\mathsf{crs}_l)_{l \in [L]}, (\mathbf{T}_l)_{l \in [L]}, \mathsf{sign}_t, \mathsf{CC}_\ell) \rightarrow y$:
  1. Parse CC_ℓ as $(\ell, \widetilde{C}_\ell, (\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b})_{i \in [n], b \in \{0,1\}}, \mathsf{ct}_{\ell,n+1})$.
  2. Extract $\mathsf{pp}^{'} = (1^{\lambda}, n+1, L)$ from crs_ℓ.
  3. Compute $k^{'} \leftarrow \mathsf{EKEM.Dec}(\mathsf{ct}_{\ell,n+1}, (\mathsf{sign}_t, \mathsf{open}_{\ell,n+1}))$, where $\mathsf{open}_{\ell,n+1} \leftarrow \mathsf{SVD.Open}((\mathsf{crs}_l)_{l \in [L]}, (\mathbf{T}_l)_{l \in [L]}, \ell, n+1)$.
  4. For $i \in [n]$, compute $\mathsf{ct}_{\ell,i,T_{\ell,i}} \leftarrow \mathsf{ct}^{\mathsf{mask}}_{\ell,i,T_{\ell,i}} \oplus \mathsf{RO}(k^{'}, \ell, i, T_{\ell,i})$.
  5. For $i \in [n]$, compute $\mathsf{lab}_{i,T_{\ell,i}} \leftarrow \mathsf{EWE.Dec}((\mathsf{crs}_l)_{l \in [L]}, (\mathbf{T}_l)_{l \in [L]}, \mathsf{sign}_t, \mathsf{ct}_{\ell,i,T_{\ell,i}})$.
  6. Output $y \leftarrow \mathsf{GC.Eval}(\widetilde{C}, (\mathsf{lab}_{i,T_{\ell,i}})_{i \in [n]})$.

The correctness of the above construction is shown from the correctness of the EKEM, EWE, and GC schemes. First, $k' = \mathsf{EKEM.Dec}(\mathsf{ct}_{\ell,n+1}, (\mathsf{sign}_t, \mathsf{open}_{\ell,n+1})) = k$ holds. Thus, for every $i \in [n]$, it holds that,

$$\begin{aligned}
&\mathsf{ct}^{\mathsf{mask}}_{\ell,i,T_{\ell,i}} \oplus \mathsf{RO}(k', \ell, i, T_{\ell,i}) \\
=&\mathsf{ct}_{\ell,i,T_{\ell,i}} \\
=&\mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), \mathsf{lab}_{i,T_{\ell,i}})
\end{aligned}$$

Hence, $\mathsf{EWE.Dec}((\mathsf{crs}_l)_{l\in[L]}, (\mathbf{T}_l)_{l\in[L]}, \mathsf{sign}_t, \mathsf{ct}_{\ell,i,T_{\ell,i}}) = \mathsf{lab}_{i,T_{\ell,i}}$. Finally, we can show that $\mathsf{GC.Eval}(\widetilde{C}, (\mathsf{lab}_{i,T_{\ell,i}})_{i\in[n]}) = C_\ell(\mathbf{T}_\ell)$.

The TSOTP scheme is trust-scalable.

Theorem 3. The construction of the TSOTP scheme in Subsection 7.2 is trust-scalable (Definition 9).

Proof. In that construction of the TSOTP scheme, the signer only needs to return the output of the signer oracle OSign, which is executed internally in the $\mathsf{EWE.SignDigest}$ algorithm. As it is called only once for each time epoch t in the TSOTP scheme, Theorem 3 straightforwardly follows that the signer's computational and communication costs are bounded by $\mathsf{poly}(\lambda)$. □

The above construction satisfies the one-time security of the OTP scheme as follows.

Theorem 4. The construction of the TSOTP scheme in Subsection 7.2 is one-time secure (Definition 14).

Proof. We first define a simulator $\mathsf{OTP.Sim}$. It maintains a state $\mathsf{state}_{\mathcal{S}}$ consisting of bit strings $(r_{i,b})_{i\in[n],b\in\{0,1\}}$, outputs of the $\mathsf{EKEM.Enc}$ algorithm (ct_{n+1}, k), and outputs of the $\mathsf{GC.SimC}$ algorithm $(\mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell})$, i.e., $\mathsf{state}_{\mathcal{S}} = ((r_{i,b})_{i\in[n],b\in\{0,1\}}, \mathsf{ct}, k, \mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell})$. It refers to $\mathsf{state}_{\mathcal{S}}$ to generate a simulated compiled circuit and program a random oracle RO. At the beginning of the experiment, $\mathsf{OTP.Sim}$ initializes $\mathsf{state}_{\mathcal{S}}$ and OSign as follows.

1. For $i \in [n]$ and $b \in \{0,1\}$, sample a bit string $r_{i,b} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{size}_{\mathsf{ct}}}$.
2. Compute $(\mathsf{ct}_{\ell,n+1}, k) \leftarrow \mathsf{EKEM.Enc}(\mathsf{crs}_\ell, x = (\ell, n+1, 0))$.
3. Generate a simulated garbled circuit $(\mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell}') \leftarrow \mathsf{GC.SimC}(1^\lambda, 1^{|C_\ell|})$.
4. Set $\mathsf{state}_{\mathcal{S}} \leftarrow ((r_{i,b})_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{\ell,n+1}, k, \mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell}')$.
5. Initialize OSign with $\mathsf{pp}' = (1^\lambda, n+1, L)$ and SVD as defined in Definition 8, which outputs a verifying key vk.

It takes as input $\mathsf{pp}, \ell, t^*, (\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]}, 1^{|C_\ell|}$ and $\mathsf{state}_{\mathcal{S}}$, and outputs a simulated compiled circuit CC_ℓ as follows.

1. Extract $\mathsf{pp}' = (1^\lambda, n+1, L)$ from crs_ℓ.
2. Parse $\mathsf{state}_{\mathcal{S}}$ as $((r_{i,b})_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{\ell,n+1}, k, \mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell})$.
3. For $i \in [n]$ and $b \in \{0,1\}$, let $\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b} \leftarrow r_{i,b}$.
4. Let $\mathsf{CC}_\ell \leftarrow (\ell, \widetilde{C}_\ell, (\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b})_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{\ell,n+1})$.
5. Output CC_ℓ.

$\mathsf{OTP.Sim}$ programs RO in a lazy manner. That is, upon the input (k,ℓ,i,b), it outputs $r_{i,b}$ if $\mathsf{RO}[(k,\ell,i,b)] = r_{i,b}$ is already computed, and otherwise, it samples a new $r_{i,b} \in \{0,1\}^{\mathsf{size}_{\mathsf{ct}}}$, sets $\mathsf{RO}[(k,\ell,i,b,)] \leftarrow r_{i,b}$, and outputs it.

When OSign is called at the time epoch t^*, RO is updated by referring to the submitted singing targets $(\mathbf{T}_l)_{l\in[L]}$ along with calling the oracle OOTE.

1. Parse $\mathsf{state}_{\mathcal{S}}$ as $((r_{i,b})_{i\in[n],b\in\{0,1\}}, \mathsf{ct}_{\ell,n+1}, k, \mathsf{state}_{\mathsf{GC.Sim}}, \widetilde{C_\ell})$.
2. Obtain the evaluation result y by calling OOTE on input $\mathbf{T}_\ell$.
3. Simulate the garbled input $(\mathsf{lab}_i')_{i\in[n]} \leftarrow \mathsf{GC.SimIn}(y, \mathsf{state}_{\mathsf{GC.Sim}})$.
4. For $i \in [n]$, compute $\mathsf{ct}_{\ell,i,T_{\ell,i}} \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), \mathsf{lab}_i')$ and $\mathsf{ct}^0_{\ell,i,1-T_{\ell,i}} \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), \mathbf{0})$.
5. For $i \in [n]$, set $\mathsf{RO}[(k,\ell,i,T_{\ell,i})] \leftarrow \mathsf{ct}_{\ell,i,T_{\ell,i}} \oplus r_{i,T_{\ell,i}}$ and $\mathsf{RO}[(k,\ell,i,1-T_{\ell,i})] \leftarrow \mathsf{ct}^0_{\ell,i,1-T_{\ell,i}} \oplus r_{i,1-T_{\ell,i}}$.

Notably, the above process does not depend on the circuit C_ℓ.

We next define a sequence of hybrids and prove that each contiguous hybrids are indistinguishable.

Hybrid 1: it is the same as the real experiment where the adversary $\mathcal{A}$ receives a compiled circuit CC_ℓ.

For $i \in [n]$ and $b \in 0, 1$,

Hybrid $2i+b$ **for every** $i \in [n]$ **and** $b \in \{0,1\}$: it is the same as Hybrid $2i+b-1$ except that the $\mathsf{TSOTP.Compile}$ algorithm computes $\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b}$ and $\mathsf{RO}[(k,\ell,i,b)]$ in a different manner as follows:

- $r_{i,b}$ is set to $\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b}$, where $r_{i,b} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{size}_{\mathsf{ct}}}$ is randomly sampled.
- $\mathsf{ct}_{\ell,i,b} \oplus r_{i,b}$ is set to $\mathsf{RO}[(k,\ell,i,b)]$.

Hybrid $2n+2i+b$ **for every** $i \in [n]$ **and** $b \in \{0,1\}$: it is the same as Hybrid $2n+2i+b-1$ except that the value of $\mathsf{RO}[(k,\ell,i,b)]$ is set not when the $\mathsf{TSOTP.Compile}$ algorithm is executed but when the OSign is called at the time epoch t^*. Notably, if the (k,ℓ,i,b) is queried to RO before OSign with the time epoch t^* is called, a random $r \leftarrow\!\!\$\ \{0,1\}^{\mathsf{size}_{\mathsf{ct}}}$ is sampled and set to $\mathsf{RO}[(k,\ell,i,b)]$, and $\mathsf{RO}[(k,\ell,i,b)]$ is not updated anymore if it is already set.

Hybrid $4n+1+i$ **for every** $i \in [n]$: it is the same as Hybrid $4n+i$ except that the value of $\mathsf{RO}[(k,\ell,i,1-T_{\ell,i})]$ is replaced with $\mathsf{ct}^0_{\ell,i,1-T_{\ell,i}} \oplus r_{i,1-T_{\ell,i}}$, where $\mathsf{ct}^0_{\ell,i,1-T_{\ell,i}} \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell,i,1-T_{\ell,i},\mathbf{0}))$.

Hybrid $5n+2$: it is the same as Hybrid $5n+1$ except that the garbled circuit $\widetilde{C_\ell}$ and the garbled input $(\mathsf{lab}_{i,T_{\ell,i}})_{i\in[n]}$ are replaced with the simulated ones $\widetilde{C_\ell}'$ and $(\mathsf{lab}_i')_{i\in[n]}$, where $(\widetilde{C_\ell}', \mathsf{state}_{\mathsf{GC.Sim}}) \leftarrow \mathsf{GC.SimC}(1^\lambda, 1^{|C_\ell|})$, $(\mathsf{lab}_i')_{i\in[n]} \leftarrow \mathsf{GC.SimIn}(y, \mathsf{state}_{\mathsf{GC.Sim}})$, and $y \leftarrow \mathsf{OOTE}(\mathbf{T}_\ell)$.

We prove that each pair of contiguous hybrids are indistinguishable.

Indistinguishability between Hybrids $2i+b-1$ **and** $2i+b$: the only differences between these hybrids are the values of $\mathsf{ct}^{\mathsf{mask}}_{\ell,i,b}$ and $\mathsf{RO}[(k,\ell,i,b)]$. Since $\mathsf{ct}_{\ell,i,b} \leftarrow \mathsf{ct}^{\mathsf{mask}}_{\ell,i,b} \oplus \mathsf{RO}[(k,\ell,i,b)]$ is the same between two hybrids, and each randomness $r_{i,b}$ is used only once, their indistinguishability is straightforwardly proven from one-time pad security.

Indistinguishability between Hybrids $2n+2i+b-1$ **and** $2n+2i+b$: the only difference between these hybrids is when $\mathsf{ct}_{\ell,i,b} \oplus r_{i,b}$ is set to $\mathsf{RO}[(k,\ell,i,b)]$. Let bad be an event that $\mathcal{A}$ submits any query to RO that contains k before calling OSign at the time epoch t^*. If bad does not occur, these hybrids are indistinguishable because the value of $\mathsf{RO}[(k,\ell,i,b)]$ is the same between these hybrids after OSign returns the t^*-th signature. Therefore, we show that the probability that bad occurs is negligible in both hybrids as follows.

Claim 5. If no PPT adversary $\mathcal{A}$ can break the time-locking of the SVD scheme (Definition 11), $\Pr[\mathsf{bad}] < \mathsf{negl}(\lambda)$ holds both in Hybrids $2n+2i+b-1$ and $2n+2i+b$.

Proof. We show that there exists an extractor $\mathcal{E}$ that outputs a signature sign^* and an opening $\mathsf{oepn}^*_{\ell,i}$ from Definition 12, which can be used to break the time-locking of the SVD scheme. Suppose that $\Pr[\mathsf{bad}]$ is larger than or equal to $\frac{1}{q(\lambda)}$ with some polynomial $q(\lambda)$, there exists a PPT adversary $\mathcal{B}$ with which the experiment $\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{B},\mathsf{OSign}}(1^\lambda, n+1, L)$ outputs 1. $\mathcal{B}$ forwards the $\mathcal{A}$'s oracle access to OSign and returns its output without modification. At the time epoch t^*, $\mathcal{B}$ receives $((\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, \mathsf{ct}_{n+1}, k)$ as defined in $\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{A},\mathsf{OSign}}$, where $b_0 \in \{0,1\}$ is randomly sampled in that experiment. $\mathcal{B}$ generates a compiled circuit CC_ℓ using the given values and programs RO in the same manner as each Hybrid, and provides $(\mathsf{crs}_l)_{l\in[L]}$, $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$, and CC_ℓ for $\mathcal{A}$. $\mathcal{B}$ returns 0 if bad occurs, and 1 otherwise. Notably, if $\mathcal{A}$ requests $\mathcal{B}$ to call OSign at the time epoch t^*, $\mathcal{B}$ returns 1 without calling OSign. Here, the following holds:

$$
\begin{aligned}
&\Pr\Big[\mathsf{Exp}^{\mathsf{EKEM}}_{\mathcal{B},\mathsf{OSign}}(1^\lambda, n+1, L, t^*, \ell, i, T_{\ell,i}) = 1\Big] \\
=&\Pr[b_0=0]\Pr[\mathsf{bad}|b_0=0] + \Pr[b_0=1]\Pr\big[\bar{\mathsf{bad}}|b_0=1\big] \\
=&\frac{1}{2}\{\Pr[\mathsf{bad}|b_0=0] + (1-\Pr[\mathsf{bad}|b_0=1])\}
\end{aligned}
$$

When $b_0 = 0$, $\mathcal{B}$ simulates either Hybrids $2n+2i+b-1$ or $2n+2i+b$. Therefore, $\Pr[\mathsf{bad}|b_0 = 0] = \Pr[\mathsf{bad}] \geq \frac{1}{p(\lambda)}$ stands. On the other hand, $\Pr[\mathsf{bad}|b_0 = 1]$ is negligible since k is randomly sampled. Hence,

$$\begin{aligned}
&\frac{1}{2}\{\Pr[\mathsf{bad}|b_0 = 0] + (1 - \Pr[\mathsf{bad}|b_0 = 1])\} \\
>&\frac{1}{2}\{\frac{1}{q(\lambda)} + (1 - \mathsf{negl}(\lambda))\} \\
>&\frac{1}{2} + \frac{1}{q'(\lambda)}
\end{aligned}$$

holds for some polynomial $q'(\lambda)$.

Definition 12 follows that there exists an extractor $\mathcal{E}$ that outputs a signature sign^* and an opening $\mathsf{open}^*_{\ell,i}$ by employing $\mathcal{B}$. We can use $\mathcal{E}$ to construct a PPT adversary $\mathcal{C}$ with which the experiment $\mathsf{Exp}^{\mathsf{SVD\text{-}Time}}_{\mathcal{C},\mathsf{OSign}}(1^\lambda, n+1, L, t^*, \ell, i, b)$ outputs 1. At the time epoch t^*, given $(\mathsf{crs}_l)_{l\in[L]}$ and $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$, $\mathcal{C}$ passes them to $\mathcal{E}$, which returns $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i})$ such that $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, b, \mathsf{open}^*_{\ell,i}) = 1$. In this process, while $\mathcal{E}$ needs to forward oracle access from $\mathcal{B}$ to OSign, $\mathcal{B}$ described above does not call OSign at the time epoch t^*. Therefore, $\mathcal{C}$ does not conflict with the condition for the adversary in $\mathsf{Exp}^{\mathsf{SVD\text{-}Time}}_{\mathcal{C},\mathsf{OSign}}$ that the adversary cannot call OSign at the time epoch t^*. Hence, $\mathcal{C}$ breaks the time-locking of the SVD scheme. This is a contradiction, and thus $\Pr[\mathsf{bad}] < \mathsf{negl}(\lambda)$ holds. □

Claim 5 follows $\Pr[\bar{\mathsf{bad}}] = 1 - \Pr[\mathsf{bad}] > 1 - \mathsf{negl}(\lambda)$. Therefore, the indistinguishability between Hybrids $2n + 2i + b - 1$ and $2n + 2i + b$ is proven.

Indistinguishability between Hybrids $4n + i$ and $4n + 1 + i$: the only differences between these hybrids are the values of $\mathsf{RO}[(k, \ell, i, 1 - T_{\ell,i})]$. That is, while the encryption of $\mathsf{label}_{i,T_{\ell,i}}$ is used in Hybrid $4n+1+i$, the encryption of $\mathbf{0}$ is used in Hybrid $4n + i$. Suppose that the adversary $\mathcal{A}$ distinguishes these hybrids with a non-trivial probability, it should be able to distinguish between those encryptions. Therefore, Definition 13 ensures that there exists an extractor $\mathcal{E}$ that outputs a signature sign^* and an opening $\mathsf{open}^*_{\ell,i}$. We can use those outputs to break the binding property of the EWE scheme (Definition 10), which is a contradiction.

Formally, we first constructs a PPT adversary $\mathcal{B}$ with which the experiment $\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{B},\mathsf{OSign}}(1^\lambda, n+1, L, t^*, \ell, i, 1-T_{\ell,i})$ outputs 1. $\mathcal{B}$ forwards the oracle access of $\mathcal{A}$ to OSign and returns its output without modification. At the time epoch t^*, $\mathcal{B}$ receives $((\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$ and returns $m_0 = \mathsf{lab}_{i,1-T_{\ell,i}}$ and $m_1 = \mathbf{0}$. $\mathcal{B}$ then receives a ciphertext $\mathsf{ct}^* \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, 1 - T_{\ell,i}), m_{b_0})$, where $b_0 \in \{0, 1\}$ is randomly sampled in the experiment $\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{B},\mathsf{OSign}}$. $\mathcal{B}$ generates a compiled circuit CC_ℓ and programs RO using the given values and provides $((\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}}, \mathsf{CC}_\ell)$ for $\mathcal{A}$. Specifically, when the OSign is called, $\mathcal{B}$ sets $\mathsf{ct}^* \oplus r_{i,1-T_{\ell,i}}$ to $\mathsf{RO}[(k, \ell, i, 1 - T_{\ell,i})]$. After obtaining a signature sign_{t^*} for a digest of signing targets $(\mathbf{T}_l)_{l\in[L]}$, $\mathcal{A}$ returns $b_1 = 0$ if it is in Hybrid $4n + i$, and $b_1 = 1$ otherwise. $\mathcal{B}$ finally outputs b_1 as its output. As $\mathcal{B}$ simulates the $\mathcal{A}$'s view in Hybrid $4n + i + b_0$, if $\mathcal{A}$ can distinguish between these hybrids,

$\Pr\left[\mathsf{Exp}^{\mathsf{EWE}}_{\mathcal{B},\mathsf{OSign}}(1^\lambda, n+1, L, t^*, \ell, i, 1-T_{\ell,i}) = 1\right] = \Pr[b_0 = b_1] \geq \frac{1}{q(\lambda)}$ holds for some polynomial $q(\lambda)$. Therefore, from Definition 13, there exists an extractor $\mathcal{E}$ that outputs a signature sign^* and an opening $\mathsf{open}^*_{\ell,i}$ by employing $\mathcal{B}$.

We next construct a PPT adversary $\mathcal{C}$ with which the experiment $\mathsf{Exp}^{\mathsf{SVD\text{-}Binding}}_{\mathcal{C},\mathsf{OSign}}(1^\lambda, n+1, L, t^*, \ell, i, 1-T_{\ell,i})$ outputs 1. At the time epoch t^*, $\mathcal{C}$ receives $((\mathsf{crs}_l)_{l\in[L]}, (\mathsf{td}_l)_{l\in[L]/\{\ell\}})$ and passes $\mathsf{pp}' = (1^\lambda, n+1, L)$, t^*, $x = (\ell, i, 1-T_{\ell,i})$, $(\mathsf{crs}_l)_{l\in[L]}$, $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$, m_0, and m_1 to $\mathcal{E}$. When $\mathcal{E}$ requests $\mathcal{C}$ to call OSign at the time epoch t^* with signing targets $(\mathbf{T}_l)_{l\in[L]}$, which is given to $\mathcal{E}$ by $\mathcal{B}$, $\mathcal{C}$ submits them and receives sign_{t^*} in the experiment $\mathsf{Exp}^{\mathsf{SVD\text{-}Binding}}_{\mathcal{C},\mathsf{OSign}}$. $\mathcal{C}$ finally forwards the $\mathcal{E}$'s output $(\mathsf{sign}^*, \mathsf{open}^*_{\ell,i})$. Since that output satisfies $\mathsf{SVD.Verify}(\mathsf{crs}_\ell, \mathsf{sign}^*, \ell, i, 1-T_{\ell,i}, \mathsf{open}^*_{\ell,i}) = 1$, the output of the experiment $\mathsf{Exp}^{\mathsf{SVD\text{-}Binding}}_{\mathcal{C},\mathsf{OSign}}$ is 1. This is a contradiction, and thus the indistinguishability between Hybrids $4n+i$ and $4n+1+i$ is proven.

Indistinguishability between Hybrids $5n+1$ and $5n+2$: the only differences between these hybrids are the values of the garbled circuit and the garbled input. Suppose $\mathcal{A}$ distinguishes between these hybrids with a non-trivial probability, we can construct a PPT adversary $\mathcal{B}$ with which the experiment $\mathsf{Exp}^{\mathsf{GC\text{-}Adap}}_{\mathcal{B}}(1^\lambda)$ outputs 1. $\mathcal{B}$ first submits a circuit C_ℓ and receives $\widetilde{C_\ell}$, where $b \leftarrow\!\!\$\ \{0,1\}$ is sampled in the experiment $\mathsf{Exp}^{\mathsf{GC\text{-}Adap}}_{\mathcal{B}}$. $\mathcal{B}$ then forwards the oracle access of $\mathcal{A}$ to OSign and returns its output without modification until the time epoch t^*. After that, $\mathcal{B}$ generates CRSs, trapdoors $(\mathsf{crs}_l, \mathsf{td}_l)_{l\in[L]}$, and a compiled circuit CC_ℓ using the given $\widetilde{C_\ell}$, and provides $(\mathsf{crs}_l)_{l\in[L]}$, $(\mathsf{td}_l)_{l\in[L]/\{\ell\}}$, and CC_ℓ for $\mathcal{A}$. When $\mathcal{A}$ requests $\mathcal{B}$ to call OSign at the time epoch t^*, where the submitted signing targets are denoted by $(\mathbf{T}_l)_{l\in[L]}$, $\mathcal{B}$ submits $\mathbf{T}_\ell$ as the input to the garbled circuit in the experiment $\mathsf{Exp}^{\mathsf{GC\text{-}Adap}}_{\mathcal{B}}(1^\lambda)$. $\mathcal{B}$ then receives the garbled input $\widetilde{x}^* = (\mathsf{lab}^*_i)_{i\in[n]}$ and emploies them to program RO. Specifically, for every $i \in [n]$, $\mathsf{ct}_{\ell,i,T_{\ell,i}}$ is calculated as the encryption of lab^*_i, i.e., $\mathsf{ct}_{\ell,i,T_{\ell,i}} \leftarrow \mathsf{EWE.Enc}(\mathsf{crs}_\ell, x = (\ell, i, T_{\ell,i}), \mathsf{lab}^*_i)$, and $\mathsf{ct}^*_{\ell,i,T_{\ell,i}} \oplus r_{i,T_{\ell,i}}$ is set to $\mathsf{RO}[(k, \ell, i, T_{\ell,i})]$. $\mathcal{A}$ outputs $b' = 0$ if it is in Hybrid $5n+1$, and $b_1 = 1$ otherwise. $\mathcal{B}$ forwards b' as its output. As $\mathcal{B}$ simulates the $\mathcal{A}$'s view in Hybrid $5n+b$, if $\mathcal{A}$ can distinguish between these hybrids, $\Pr\left[b = b'\right]$ is non-negligible. This is a contradiction, and thus the indistinguishability between Hybrids $5n+1$ and $5n+2$ is proven.

Hybrid $5n+2$ is completely simulated by the simulator $\mathsf{OTP.Sim}$, which does not depend on the circuit C_ℓ. Therefore, the real experiment corresponding to Hybrid 1 is indistinguishable from the simulated experiment corresponding to Hybrid $5n+2$. This completes the proof. □

8 Conclusion

In this paper, we propose witness encryption (WE) for signed vector digests, a novel WE scheme built from bilinear maps. This scheme allows anyone with a valid signature to decrypt ciphertexts while maintaining constant computational and communication costs for the trusted signer at regular intervals. It provides extractable security, a stronger security definition than that of standard WE.

Our construction uses bilinear maps and symmetric key encryption in the random oracle model. Furthermore, we demonstrate that the construction can be easily extended to a threshold multiple signers setting, where decryption requires signatures from at least a threshold number of signers.

Using our WE scheme, we also construct trust-scalable one-time programs (TSOTPs). Notably, the signer only needs to perform a fixed amount of computation and communication at regular intervals to enable the evaluation of a polynomial number of OTPs generated by independent OTP generators. This feature bounds the operational costs of the signers' machines even as the number of evaluated OTPs increases. Thus, it enables scalable OTP evaluations without increasing the risk of the decrease in the number of signers and the worsening of their decentralization.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *Cryptology ePrint Archive*, 2013.

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline witness encryption. In *International Conference on Applied Cryptography and Network Security*, pages 285–303. Springer, 2016.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 440–456. Springer, 2005.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In *Theory of cryptography conference*, pages 52–73. Springer, 2014.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In *Annual International Cryptology Conference*, pages 121–151. Springer, 2020.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Annual international cryptology conference*, pages 258–275. Springer, 2005.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 134–153. Springer, 2012.

[BIJ+20] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: a framework for obfuscation and witness encryption. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.

[BL20] Fabrice Benhamouda and Huijia Lin. Multiparty reusable non-interactive secure computation. *Cryptology ePrint Archive*, 2020.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of cryptology*, 17:297–319, 2004.

[CDK+22] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. Encryption to the future: a paradigm for sending secret messages to future (anonymous) committees. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 151–180. Springer, 2022.

[CFK24] Matteo Campanelli, Dario Fiore, and Hamidreza Khoshakhlagh. Witness encryption for succinct functional commitments and applications. In *IACR International Conference on Public-Key Cryptography*, pages 132–167. Springer, 2024.

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. *Designs, Codes and Cryptography*, 55:141–167, 2010.

[CJK20] Peter Chvojka, Tibor Jager, and Saqib A Kakvi. Offline witness encryption with semi-adaptive security. In *International Conference on Applied Cryptography and Network Security*, pages 231–250. Springer, 2020.

[DHMW23] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. Mcfly: verifiable encryption to the future made practical. In *International Conference on Financial Cryptography and Data Security*, pages 252–269. Springer, 2023.

[DS18] David Derler and Daniel Slamanig. Practical witness encryption for algebraic languages or how to encrypt under groth–sahai proofs. *Designs, Codes and Cryptography*, 86(11):2525–2547, 2018.

[EGG+22] Harry Eldridge, Aarushi Goel, Matthew Green, Abhishek Jain, and Maximilian Zinkus. One-time programs from commodity hardware. In *Theory of Cryptography Conference*, pages 121–150. Springer, 2022.

[FHAS24] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for kzg commitments and efficient laconic ot. *Cryptology ePrint Archive*, 2024.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 33–62. Springer, 2018.

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *Public-Key Cryptography–PKC 2017: 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I 20*, pages 121–150. Springer, 2017.

[FWW23] Cody Freitag, Brent Waters, and David J Wu. How to use (plain) witness encryption: Registered abe, flexible broadcast, and more. In *Annual International Cryptology Conference*, pages 498–531. Springer, 2023.

[GG17] Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I 15*, pages 529–561. Springer, 2017.

[GGHW17] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. *Algorithmica*, 79:1353–1373, 2017.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings 7*, pages 308–326. Springer, 2010.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.

[GKM+22] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. In *IACR International Conference on Public-Key Cryptography*, pages 252–282. Springer, 2022.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 536–553. Springer, 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Advances in Cryptology–CRYPTO 2008: 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings 28*, pages 39–56. Springer, 2008.

[GLW14] Craig Gentry, Allison Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 426–443. Springer, 2014.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

[GRWZ20] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2007–2023, 2020.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *IACR International Workshop on Public Key Cryptography*, pages 3–33. Springer, 2019.

[GWC19] Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, 2019.

[HJO+15] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. *Cryptology ePrint Archive*, 2015.

[JO20] Zahra Jafargholi and Sabine Oechsner. Adaptive security of practical garbling schemes. In *International Conference on Cryptology in India*, pages 741–762. Springer, 2020.

[KZG10] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[LJKW18] Jia Liu, Tibor Jager, Saqib A Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Designs, Codes and Cryptography*, 86:2549–2586, 2018.

[LPR22] Benoît Libert, Alain Passelègue, and Mahshid Riahinia. Pointproofs, revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 220–246. Springer, 2022.

[LY10] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *Theory of Cryptography Conference*, pages 499–517. Springer, 2010.

[MTV+22] Varun Madathil, Sri AravindaKrishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. *Cryptology ePrint Archive*, 2022.

[Nit20] Anca Nitulescu. zk-snarks: A gentle introduction. 2020.

[SH23] Sora Suegami and Leona Hioki. Octopus contract and its applications. `https://github.com/SoraSuegami/octopus-contract-paper/blob/11ce8419a831c5db901ba9670af50335ae1e0865/octopus_contract_swe_1215.pdf`, November 2023. (Accessed on 04/19/2024).

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *Annual International Cryptology Conference*, pages 535–559. Springer, 2022.

[ULCG21] Gavin Uberti, Kevin Luo, Oliver Cheng, and Wittmann Goh. Building usable witness encryption. *arXiv preprint arXiv:2112.04581*, 2021.

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 195–221. Springer, 2022.