Bilinear Map Based One-Time Signature Scheme with Secret Key Exposure

Marten van Dijk[1,2], Deniz Gurevin[2], Chenglu Jin[1], Omer Khan[2], and Phuong Ha Nguyen[3]

[1] CWI Amsterdam
{marten.van.dijk,chenglu.jin}@cwi.nl
[2] University of Connecticut
{deniz.gurevin,omer.khan}@uconn.edu
[3] eBay
phuongha.ntu@gmail.com

Abstract. Dijk et al. [6] presents Remote Attestation (RA) for secure processor technology which is secure in the presence of an All Digital State Observing (ADSO) adversary. The scheme uses a combination of hardware security primitives and design principles together with a new cryptographic primitive called a Public Key Session based One-Time Signature Scheme with Secret Key Exposure (OTS-SKE). [6] shows a hash based realization of OTS-SKE which is post quantum secure but suffers long 8.704 KB signatures for 128-bit quantum security or 256-bit classical security. From a classical cryptographic perspective we complete the picture by introducing a bilinear map based OTS-SKE with short 0.125 KB signatures, 65 times shorter, and for which the security reduces to the Computational Diffie-Hellman Problem (CDHP) – at the cost of a 9× longer initialization phase in the RA scheme if implemented in software (this can be improved with appropriate elliptic curve hardware acceleration). Signing takes 560 ms at most 60% of the > 936 ms needed for the hash based scheme.

Keywords: Remote Attestation · One Time Signatures · Secret Key Exposure

1 Introduction

An All Digital State Observing (ADSO) adversary observes all digital state including all intermediate computed digital values as well as all digital storage such as register values, permanent storage, and fused keys. From the digital computing perspective the ADSO adversary is a white-box adversary who can see the internal working of any implemented crypto primitive including its manipulation of secret keys.

Yet, as demonstrated for the first time in [6], the ADSO adversary allows secure Remote Attestation (RA) by realizing that key material can be hidden in non-digital form using Physical Unclonable Function (PUF) technology. The main idea is to only extract a message dependent subset of the PUF's key material for RA and by implementing proper access control to the underlying PUF so that no other relevant key material can leak (here we notice that processor hardware implements 'secure enclaves', each enclave only able to access its own 'partition' of the PUF's challenge response space; details are in [6]). This means that signature generation can simply extract and use this message dependent key material as a one-time signature.

This is exactly the working of the hash based realization in [6]. In essence the PUF is used to de-obfuscate a message dependent subset of 'leaf keys' in an authentication tree. The subset of leaf keys together with additional node values for authenticating the path from the subset of leafs to the root of the authentication tree forms a signature. The signature can be verified by the public key which includes the root. This hash based scheme is post quantum secure but suffers from long signatures because an $O(\lambda)$ number of leaves, where λ is the security parameter, needs to be transmitted as part of the final signature.

The hash based scheme realizes a Public Key Session based One-Time Signature Scheme with Secret Key Exposure (OTS-SKE) as introduced and defined in [6]. Informally, an OTS-SKE scheme (1) has one (universal) public key that can be used to verify all session signatures, with the property that (2) each logical session with its own secret session key generates at most one signature for which it uses a "subset" of the secret session key, and (3) this "subset" is exposed to the (ADSO) adversary "for free." Despite leaking these subsets of secret session keys (for each of the sessions), the adversary cannot impersonate a signature for a new message for any session. Our main contribution is a new bilinear map based OTS-SKE scheme which realizes compact signatures (the algebraic structure allows one to compress the subset of session key material into a single value in the signature).

Table 1. Comparison bilinear map based OTS-SKE (this paper) and hash based OTS-SKE [6] for 256-bit classical security. N indicates the number of sessions in the overarching RA scheme.

	Billinear Map based OTS-SKE	Hash based OTS-SKE
Key Generation	7587.3 ms	819 ms
Sign	560 ms	$935 + 0.6 \log_2 N$ ms
Verify	100 ms	$118 + 0.4 \log_2 N$ ms
Storage	116 MB	$0.12N$ MB
Communication	0.125 KB	$8.2 + 0.03 \log_2 N$ KB
Lines of code	7k+	~ 500

Table 1 compares both OTS-SKE schemes. Clearly, for large $N \geq 1024$ the sign, verify, storage, and communication costs of the hash based OTS-SKE scheme scale to large values beyond those required for the new bilinear map based OTS-SKE scheme introduced in this paper. In particular, the bilinear map based OTS-SKE scheme reduces the communication cost by a factor > 65

and its sign costs is at most 60% of the sign costs of the hash based OTS-SKE scheme. The bilinear map based scheme has a factor 9 longer key generation (which happens during an initialization/set-up phase in the RA scheme) and a factor 14 times more lines of code (in the secure enclave that implements the RA scheme) giving a larger Trusted Compute Base (TCB). In the implementation section we discuss how key generation can be made much faster by means of elliptic curve hardware acceleration (our numbers are for the much slower software implementation). We also notice that key generation happens off-line while sign and communication happens on-line with throughput, latency, and bandwidth constraints in practice. This means that the factor > 65 shorter signatures and 60% of the sign cost show that the proposed bilinear map based ORS-SKE scheme outperforms the existing hash based OTS-SKE scheme.

Our bilinear map based OTS-SKE is not post quantum secure as its security reduces to the Computational Diffie-Hellman Problem (CDHP). In fact, by solving CDHP for the public key itself using a quantum computer, "master key" can be extracted which can be used to impersonate any signature from past or future using a classical computation. Thus, as soon as quantum computing resources are available on-line and cost-effective with respect to the gain from impersonating signatures for an instantiation of the RA scheme, then the bilinear map based OTS-SKE should not be used. However, without sufficiently available quantum resources, Table 1 shows that the bilinear map based OTS-SKE scheme is the best choice with significant improvement over the hash based OTS-SKE scheme. The current study shows into what extent classical crypto can improve effciency and this completes the picture around OTS-SKE constructions.

Outline. Section 2 discusses related work including more detail on the hash based OTS-SKE scheme with Figure 3. We formally define OTS-SKE schemes in Section 3. We introduce the new bilinear map based OTS-SKE scheme in Section 4 and show implementation results (leading to Table 1) in Section 5.

2 Related Work

Forward-secure signature schemes aim to protect the privacy of past keys even when the current secret key has been leaked. However, these schemes have a major shortcoming when it comes to preventing the exposure of next keys as all the next keys $sk_{t+1}, \ldots, sk_{N-1}$ can be computed if sk_t is the first compromised key. In such case, the public key of the scheme has to be revoked and renewed. In order to minimize the effects of key exposure, intrusion detection and prompt key revocation mechanisms have been proposed, which can be inefficient in practice [16].

In order to prevent the leakage of next as well as past secret keys, key-insulated schemes (KIS) [9,10,12,14,26] and intrusion-resilient schemes (IRS) [7, 8] have been introduced. These schemes typically have an architectural design with a *user* (Bob) and a *base* (Bob's base) as depicted in Figure 1. Alice uses Bob's public key, which does not change over time, to encrypt messages sent

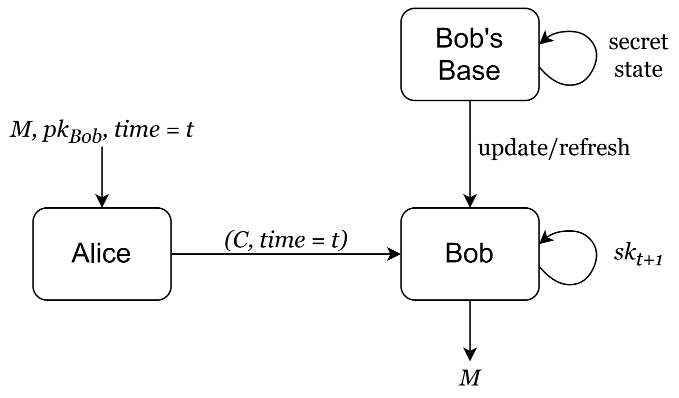

Fig. 1. The structure of an Intrusion Resilient Scheme. M is the plaintext and C is the ciphertext.

to Bob. Even though the public key does not change over time, Bob and Bob's base have secret information which does change over time. Bob needs his current secret and the secret from the Bob's base to update towards a new decryption key for the next period (sk_{t+1}). Bob's base on the other hand can update its secret information by itself based on its current secret state. User Bob and his base are two separate components. It is assumed that the base is secure against any key attacks for KIS but this requirement is relaxed in IRS. IRS will be vulnerable if and only if the secret information at the user and the base is compromised at the same time, see Figure 2. We notice that we require a secure communication channel between the user and the base for exchanging secret information and this can also be a point of attack in practice.

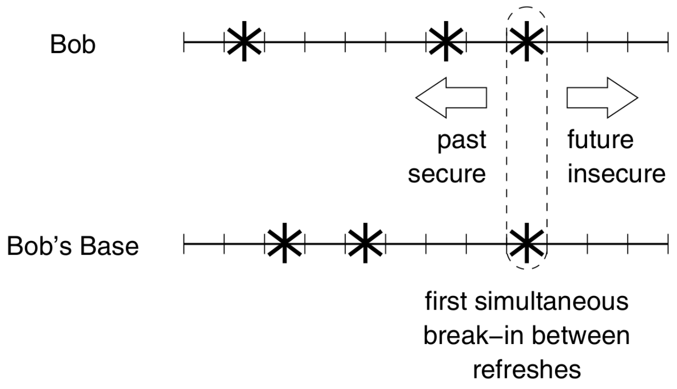

Fig. 2. The security of Intrusion Resilient Scheme.

Dijk *et al.* [6] proposed a Remote Attestation protocol which can remain secure in the presence of an All Digital State Observing (ADSO) adversary (i.e. both the 'user' and 'base' can leak at the same time) and solves the shortcomings of IRS and KIS by removing the dependence on a trusted third party. The scheme uses a Public Key Session based One-Time Signature Scheme with Secret Key Exposure (OTS-SKE) which is post quantum secure. However, one of the shortcomings of [6] is the usage of large sized signatures.

2.1 Hash-based OTS-SKE Scheme

Dijk *et al.* [6] proposed a Remote Attestation protocol which can remain secure in the presence of an All Digital State Observing (ADSO) adversary (i.e. both the 'user' and 'base' can leak at the same time) and solves the shortcomings of IRS and KIS by removing the dependence on a trusted third party. The scheme uses a Public Key Session based One-Time Signature Scheme with Secret Key Exposure (OTS-SKE) which is post quantum secure. However, one of the shortcomings of [6] is the usage of large sized signatures.

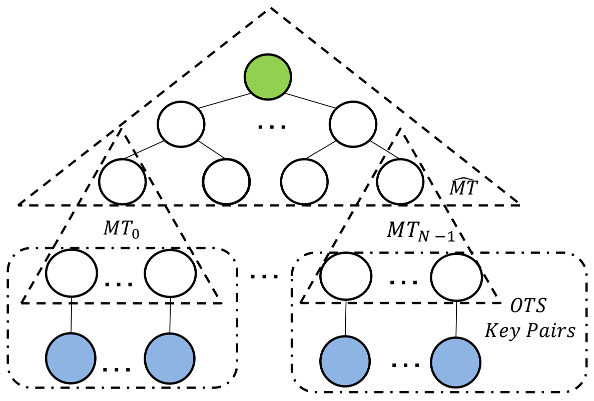

Fig. 3. Construction of Hash-based OTS-SKE in [6]. The bottom blue nodes are secret keys in Lamport One Time Signature (OTS) scheme, and the signer needs to keep these secret keys private. The top green node represents the root hash of the top Merkle tree $\hat{M}T$, which will be treated as a univeral public key to be shared with verifiers.

In [6], Dijk *et al.* introduced a hash-based scheme of one-time signature with secret key exposure (OTS-SKE). The key idea is based on an optimized Lamport one-time signature scheme [2,17] and an optimized Merkle tree construction [5, 19].

Fig. 3 presents the construction of the hash-based OTS-SKE scheme in [6]. At the bottom, there are N groups of secret-verification key pairs of Lamport one-time signature scheme [2,17]. Each group will be used in a unique session, so the structure can support N sessions in total. Each secret key is randomly generated, and its corresponding verification key is the hash of the secret key. Lamport one-time signature signs a message by sending a message-dependent subset of all the secret keys in one session to the verifier. By checking the indices of the received secret keys and their hash values against the verification keys, the verifier can verify whether the signature is valid. To avoid sending N groups of verification keys (vk) to the verifier for initialization, the scheme in [6] uses two layers of Merkle tree structure. Each of the lower layer Merkle trees ($MT_0, \ldots, MT_{N-1}$ in Fig. 3) manages all the verification keys for one session i. The higher layer Merkle tree ($\hat{MT}$ in Fig. 3) compresses all lower layer Merkle trees to one root hash, and the root hash (green node in Fig. 3) is sent over to the verifier as a universal public key for verifying the signatures generated in every session. In order to reduce the security of the Merkle tree to the second pre-image resistance of the underlying hash functions, every hash evaluation in the Merkle trees needs a random bit-mask [5]. To mitigate multi-target attacks, the authors of [6] also suggest incorporating a random number as a hash key in every hash calls in the OTS computation and the Merkle tree construction. In their implementation, a pseudorandom number generator is used to generate all needed random numbers from a seed. The seed will be shared with the verifier for verification.

3 Definition of an OTS-SKE Scheme

The following definition of an OTS-SKE scheme is a verbatim citation from [6]. An OTS-SKE scheme $\mathcal{S}$ consists of three procedures

$$\mathcal{S} = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify}) :$$

Key generation. Based on a security parameters λ, KeyGen generates a public key pk together with session secret keys

$$sk_i = \{sk_{i,j}\}_{j=0}^{q-1}$$

and auxilairy variables aux_i for each session $i \in \{0, \ldots, N-1\}$ and a-priori fixed parameter q. We have

$$(pk, \{sk_i, aux_i\}_{i=0}^{N-1}) \leftarrow \textsc{KeyGen}(\lambda).$$

Sign. Sign takes as input the session id i with session secret key sk_i and auxiliary variable aux_i. Together with a message $M \in \{0,1\}^n$ as input Sign produces a signature σ,

$$\sigma \leftarrow \textsc{Sign}(sk_i, aux_i; M).$$

The computation of Sign is split in three steps:

1. We have a keyed pseudo random permutation $\mathrm{PRP}(key; x)$ which, for each key, is a bijective mapping from strings $x \in \{0,1\}^n$ to $\{0,1\}^n$. We also have an injective mapping ϕ from $\{0,1\}^n$ to subsets of $\{0, \ldots, q-1\}$ (here, $q \geq n$). Sign first selects a random key and computes the subset

$$I = \phi(\mathrm{PRP}(key; M)) \subseteq \{0, \ldots, q-1\}.$$

2. Sign extracts a corresponding subset of the i-th session secret key:
$$sk_{i,I} = \{sk_{i,j}\}_{j \in I}.$$
3. Sign uses $sk_{i,I}$ together with aux_i and input message M to produce a signature σ'. In order to make the dependence on the subset of the session key explicit, we write
$$\sigma' \leftarrow \text{Sign'}(sk_{i,I}, aux_i; M).$$
Sign returns $\sigma = (\sigma', key)$.

Verify. Verify outputs
$$\{\mathbf{true}, \mathbf{false}\} \leftarrow \text{Verify}(pk, i; \sigma, M)$$
for a signed message (σ, M) for session id i. Notice that the same public key pk is used for all sessions.

Correctness. OTS-SKE scheme $\mathcal{S}$ is correct if for all $\sigma \leftarrow \text{Sign}(sk_i, aux_i; M)$ we have $\mathbf{true} \leftarrow \text{Verify}(pk, i; \sigma, M)$.

Security. Even if an adversary has knowledge of subsets of session keys
$$\{sk_{i,I_i}\}_{i=0}^{N-1}$$
together with auxiliary information $\{aux_i\}_{i=0}^{N-1}$, the adversary cannot impersonate a signature for some session with id i^* for a new message that has not yet been signed in session i^*. This security notion is formalized by GameOTS-SKE for $\mathcal{S}$ as the following security game:

– **Setup**: The challenger runs KeyGen which returns
$$(pk, \{\{sk_{i,j}\}_{j=0}^{q-1}, aux_i\}_{i=0}^{N-1}).$$
The challenger gives pk as well as $\{aux_i\}_{i=0}^{N-1}$ to the adversary.
– **Query**: The adversary adaptively issues a sequence of messages M_i at most one message for each session id i. The challenger computes
$$I_i = \phi(\text{PRP}(key_i; M_i)) \text{ and } sk_{i,I_i} = \{sk_{i,j}\}_{j \in I_i}$$
for random key_i. The challenger gives the extracted information sk_{i,I_i} with key_i to the adversary (as soon as M_i is received).
Notice that the adversary can use this information to sign message M_i for session i by applying Sign'. This may lead to multiple signatures for M_i (since fresh randomness can be used for each signature generation). However, no signatures for other messages $\neq M_i$ for session id i can be forged if the following Guess does not succeed.
– **Guess**: The adversary selects a session number $i^* \in \{0, \ldots, N-1\}$ which refers to the session for which the adversary will want to forge a signature: The adversary outputs a signed message (σ, M^*) for session i^* such that $M^* \neq M_{i^*}$ The adversary wins the game if the signature verifies, that is,
$$\mathbf{true} \leftarrow \text{Verify}(pk, i^*; \sigma, M^*).$$

In this game, $\mathcal{A}$ is called an OTS-SKE-EUF-CMA (OTS-SKE Existential UnForgeability under Chosen Message Attack) adversary.

If $\mathcal{A}$ wins GameOTS-SKE with probability $\geq \epsilon$ in time $\leq T$, then we call $\mathcal{A}$ a (T, Q_H, Q_P, ϵ)-OTS-SKE adversary for $\mathcal{S}$, where Q_H and Q_P are the maximum number of queries allowed to be made by $\mathcal{A}$ to a hash function oracle and PRP oracle in GameOTS-SKE. We say scheme $\mathcal{S}$ is (T, Q_H, Q_P, ϵ)-secure against OTS-SKE-EUF-CMA attacks if no (T, Q_H, Q_P, ϵ)-OTS-SKE adversary exists.

4 Bilinear Map based OTS-SKE

Bilinear map. Let $\mathbb{G}$ be a bilinear group of prime order p and g be a generator of $\mathbb{G}$. Here, size p of $\mathbb{G}$ is determined (by some functional relation) by the security parameter λ of the to-be-explained constructions. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map, i.e., we have the following properties

- *Bilinear* [4]: For all $x, y \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$,
$$e(x^a, y^b) = e(x, y)^{ab}.$$
- *Non-degenerate* [5]: $e(g, g) \neq 1$.

For practical usage the bilinear map should be efficiently computable. The above properties can be realized by the modified Weil pairing based on supersingular curves.

Hash function. In addition to the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ we use a cryptographic hash function $H : \{0,1\}^{\lceil \log_2 N \rceil + n} \to \mathbb{Z}_p$. $H(\cdot)$ is used in the signing algorithm to hash an index message pair into an element in $\mathbb{Z}_p$.

OTS-SKE scheme. Below we describe our OTS-SKE scheme

$$\mathcal{S} = (\textsc{KeyGen}, \textsc{Sign}, \textsc{Verify}) :$$

Key generation. We use parameters $q = tn$ and represent index $tj + b \in \{0, \ldots, q-1\}$ as the pair (j, b). KeyGen sets parameters, computes the public key, and all secret keys

$$(pk, \{\{sk_{i,j,b}\}_{j=0,b=0}^{n-1,t-1}, aux_i\}_{i=0}^{N-1}) \leftarrow \textsc{KeyGen}(\lambda)$$

as follows:

[4] The definition implies $e(x, yz) = e(x, y)e(x, z)$ and $e(xz, y) = e(x, y)e(z, y)$ for all $x, y, z \in \mathbb{G}$. Also, $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$, hence, $e(x, y) = e(y, x)$.

[5] The definition implies for $x = g^a \in \mathbb{G}$, if $e(g, x) = 1$, then $1 = e(g, x) = e(g, g^a) = e(g, g)^a$, hence, $a = 0$ (since $e(g, g) \neq 1$), i.e., $x = 1$. Therefore, if $e(g, x) = e(g, y)$, then $e(g, x/y) = 1$ which implies $x/y = 1$ or equivalently $x = y$. The property "$e(g, x) = e(g, y)$ implies $x = y$" will be used in the security proof.

- $(p, \mathbb{G}, \mathbb{G}_1, e, g, g_2) \leftarrow IG(1^\lambda)$ where λ is the security parameter and algorithm IG generates a suitable mathematical structure for our signature scheme. g and g_2 are generators of $\mathbb{G}$ and $\mathbb{G}_1$, respectively.
- Randomly generate $\alpha \in \mathbb{Z}_p^*$ and set $g_1 = g^\alpha$. Define $F(i) = g_1^i h$ where h is a random number chosen from $\mathbb{G}$. Note that $F : \mathbb{Z}_p \rightarrow \mathbb{G}$.
- Generate N secret keys $\{sk_i\}_{i=0}^{N-1}$ with auxiliary information $\{aux_i\}_{i=0}^{N-1}$ as follows:
$$sk_i = \{sk_{i,j,b}\}_{j=0,b=0}^{n-1,t-1}$$
with
$$sk_{i,j,b} = g_2^\alpha F(it^n + bnt^j)^{r_i} v_{i,j} \text{ and } aux_i = g^{r_i}, \tag{1}$$
where r_i is a random number chosen from $\mathbb{Z}_p$ and $v_{i,j} = g^{\beta_{i,j}}$, where $\beta_{i,j}$ are random numbers from $\mathbb{Z}_p$ such that $\sum_{j=0}^{n-1} \beta_{i,j} = 0$, or equivalently, $\prod_{j=0}^{n-1} v_{i,j} = 1$.
- Parameters $pk = (p, \mathbb{G}, \mathbb{G}_1, e, g, g_1, g_2, h)$ are made public and secret keys $\{sk_i\}_{i=0}^{N-1}$ are kept private. The auxiliary information $\{aux_i\}_{i=0}^{N-1}$ is kept at the signer but is not kept secret (it can be accessed by anyone who wants to). All other values such as α (or g_2^α) and random numbers $\{r_i\}$ and $\{v_{i,j}\}$ are deleted.

Our key generation procedure deviates from [3] in that we have 'flatten' their 'tree structure' and do not have a KeyGen algorithm which keeps on generating new secret keys by maintaining a secret state. We do not want any digital secret state as this leaks to the ADSO-adversary in the remote attestation protocol of [6] for which our signature scheme is used.

A second deviation is the secret sharing mechanism based on the $\{v_{i,j}\}$, which role will become clear in the security analysis and allows us to achieve resistance against OTS-SKE-EUF-CMA attacks.

Signing. We compute $B = \sum_{j=0}^{n-1} b_j t^j$ with $0 \leq b_j < t$ as the pseudo random permutation output $B = \text{PRP}(key; M)$ for a random key. We define subset $\phi(B) = \{tj + b_j\}_{j=0}^{n-1}$ of $\{0, \ldots, q-1\}$ for $q = tn$. We represent its elements by the pairs (j, b_j). We produce
$$\sigma' \leftarrow \text{SIGN'}(\{sk_{i,j,b_j}\}_{j=0}^{n-1}, sk1_i; M),$$
which signs a plaintext $M \in \mathbb{G}_1$ as follows:

- Select a random $s \in \mathbb{Z}_p$.
- Compute $x = g_2^{ns}$.
- Compute hash value $u = H(M, x)$.
- Compute $y = aux_i^{n(s+u)} = (g^{r_i})^{n(s+u)}$.

– Compute

$$\begin{aligned} sk_i = \prod_{j=0}^{n-1} sk_{i,j,b_j} &= \prod_{j=0}^{n-1} g_2^{\alpha} F(it^n + b_j nt^j)^{r_i} v_{i,j} \\ &= g_2^{n\alpha} \left(\prod_{j=0}^{n-1} F(it^n + b_j nt^j) \right)^{r_i} \\ &= g_2^{n\alpha} (g_1^{it^n+B} h)^{nr_i} = (g_2^{\alpha} F(it^n + B)^{r_i})^n \end{aligned}$$

and $z = sk_i^{s+u} = (g_2^{\alpha} F(it^n + B)^{r_i})^{n(s+u)}$.

– Return signature $\sigma = (\sigma', key)$ for

$$\begin{aligned} \sigma' &= (x, y, z) \\ &= (g_2^{ns}, (g^{r_i})^{n(s+u)}, (g_2^{\alpha} F(it^n + B)^{r_i})^{n(s+u)}). \end{aligned}$$

Verification. $\textsc{Verify}(pk, i; \sigma, M)$ with $\sigma = (\sigma', key)$ verifies signature $\sigma' = (x, y, z)$ for message M, where σ' is generated during the i-th session:

– Compute $u = H(M, x)$ and $B = \text{PRP}(key; M)$.
– The signature verifies if and only if

$$e(g, z) = e(g_1, g_2^{nu} x y^k) \times e(y, h) \text{ with } k = it^n + B.$$

4.1 Correctness

The correctness of this check follows from $g_1 = g^{\alpha}$, $x = g_2^{ns}$, $y = g^{r_i n(s+u)}$, $z = (g_2^{\alpha} F(k)^{r_i})^{n(s+u)}$, and

$$\begin{aligned} & e(g^{\alpha}, g_2^{nu} x y^k) \times e(y, h) \\ &= e(g^{\alpha}, g_2^{nu} g_2^{ns} (g^{r_i n(s+u)})^k) \times e(g^{r_i n(s+u)}, h) \\ &= e(g^{\alpha}, g_2^{n(u+s)} (g^{r_i n(s+u)})^k) \times e(g^{r_i n(s+u)}, h) \\ &= (e(g^{\alpha}, g_2 g^{kr_i}) \times e(g^{r_i}, h))^{n(s+u)}. \end{aligned}$$

Since $e(a^c, b) = e(a, b^c)$, the right hand side is equal to

$$\begin{aligned} & (e(g, g_2^{\alpha} g^{\alpha k r_i}) \times e(g, h^{r_i}))^{n(s+u)} \\ &= (e(g, g_2^{\alpha} (h g^{\alpha k})^{r_i}))^{n(s+u)} = e(g, (g_2^{\alpha} (h g_1^k)^{r_i})^{n(s+u)}) \\ &= e(g, (g_2^{\alpha} F(k)^{r_i})^{n(s+u)}) = e(g, z). \end{aligned}$$

4.2 Security

Our security analysis will apply the forking lemma [21] in order to reduce an OTS-SKE adversary $\mathcal{A}$ to being able to solve the Computational Diffie-Hellman Problem (CDHP) in $\mathbb{G}$. We first give some background:

Forking lemma [21]. Our scheme corresponds to a generic digital signature scheme [21], i.e., given an input message M, the scheme produces a triple (σ_1, u, σ_2) where σ_1 $(= x$ for our scheme) randomly takes its values in a large set (in our case $x \in \mathbb{G}$), u is the hash value $H(M, \sigma_1)$ and σ_2 $(= (y, z)$ for our scheme) only depends on σ_1, message M, u, and secret keys (but does not depend on other randomly selected values). We notice that u does not need to be transmitted as part of the signature since it can be computed by the verifier by applying the hash function.

We denote by Q_H the number of queries that $\mathcal{A}$ can ask a random oracle which models the hash functionality. Suppose that within a time bound T and with probability $\epsilon \geq 7Q_H/2^\lambda$, where λ is the security parameter of the scheme, $\mathcal{A}$ can produce a valid signature $(M, \sigma_1, u, \sigma_2)$. Then there exists another algorithm which has control over $\mathcal{A}$ and produces two valid signatures $(M, \sigma_1, u, \sigma_2)$ and $(M, \sigma_1, u', \sigma'_2)$ such that $u \neq u'$ in expected time $T' \leq 84480 \cdot TQ_H/\epsilon$. The new algorithm is in control of the random oracle and simulates $\mathcal{A}$ for a first hash function $H(.)$ with $u = H(M, \sigma_1)$ and for a second hash function $H'(.)$ with $u' = H'(M, \sigma_1)$.

Computational Diffie-Hellman problem. Our security analysis shows that the two valid signatures can be algebraically combined allowing us to solve the Computational Diffie-Hellman Problem (CDHP) in $\mathbb{G}$: Given a triple $(g, g^a, g^b) \in \mathbb{G}^3$, compute $g^{ab} \in \mathbb{G}$.[6]

The following theorem states the reduction of OTS-SKE-EUF-CMA security to CDHP.

Theorem 1. *Let $\mathcal{S}$ be the bilinear map based OTS-SKE scheme with N sessions. Let $\mathcal{A}$ be a (T, Q_H, Q_P, ϵ)-OTS-SKE adversary for $\mathcal{S}$ with $\epsilon \geq 7Q_H/2^\lambda$, where λ is the security parameter of $\mathcal{S}$. Then, there exists an algorithm that solves CDHP in $\mathbb{G}$ in expected time $\leq 84480 \cdot 2TQ_H(Q_P + 1)N/\epsilon$.*

The proof of our main result is as follows: Let $\mathcal{A}$ be a (T, Q_H, Q_P, ϵ)-OTS-SKE adversary with $\epsilon \geq 7Q_H/2^\lambda$. We will show how to build an algorithm $\mathcal{B}$ which can solve CDHP in $\mathbb{G}$ based on $\mathcal{A}$.

$\mathcal{B}$ takes on the role of challenger in GameOTS-SKE and plays against $\mathcal{A}$, and whenever $\mathcal{A}$ needs to compute a hash value then $\mathcal{B}$ plays the role of the hash oracle, that is, the random oracle which simulates hash function $H(.)$; $\mathcal{B}$ keeps a list L_H to store the answers of the random oracle, that is, if $\mathcal{A}$ queries a hash value of some input w, then $\mathcal{B}$ checks list L_H and if an entry (w, u) for the query is found, the same answer u will be returned to $\mathcal{A}$, otherwise, $\mathcal{B}$ randomly generates a value u from $\mathbb{Z}_p$ and outputs u and appends (w, u) to L_H. We only talk about algorithm $\mathcal{A}$ querying the hash oracle. As it turns out $\mathcal{B}$ does not query its own hash oracle at all.

$\mathcal{B}$ is also in charge of the random tape used by $\mathcal{A}$, i.e., $\mathcal{B}$ is able to replay $\mathcal{A}$ for the same random tape and with a random oracle that simulates a different

[6] We say that the (t', ϵ)-CDH assumption holds in $\mathbb{G}$ if no t'-time algorithm has advantage at least ϵ in solving the CDHP in $\mathbb{G}$.

hash function $H'(.)$. As we will see, by exploiting this, $\mathcal{A}$ can be used by $\mathcal{B}$ to solve CDHP.

Finally, $\mathcal{B}$ also plays the role of the PRP oracle, that is, the random oracle which simulates the pseudo random permutation PRP. Now $\mathcal{B}$ selects at the start of its execution, for each $i \in \{0, \ldots, N-1\}$, a set of distinct random values

$$Set_{PRP}(i) = \{B_0^*, \ldots, B_{Q_P+1}^*\} \subseteq \{0,1\}^n$$

together with a random key_i. Notice that the size of each set is $Q_P + 2$. The set is used to answer PRP oracle queries from both $\mathcal{A}$ as well as $\mathcal{B}$ (it queries itself). $\mathcal{B}$ maintains a list L_P of triples (i, M, B): If $\mathcal{B}$ or $\mathcal{A}$ queries a PRP value for some input $(key_i; M)$, then $\mathcal{B}$ checks list L_P and if an entry (i, M, B) for the query is found, the same answer B will be returned to itself, otherwise, $\mathcal{B}$ randomly generates a value B from

$$Set_{PRP}(i) \setminus \{B' : (i, M', B') \in L_P\}$$

and outputs B and appends (i, M, B) to L_P. Notice that the PRP oracle only outputs values from $Set_{PRP}(i)$ for each i. This is possible because $\mathcal{A}$ makes at most a total of Q_P PRP queries and, as we will see below, $\mathcal{B}$ makes at most 2 PRP queries per session (one during Setup and one during Guess).

Let

$$g_1 = g^a \text{ and } g_2 = g^b.$$

We define algorithm $\mathcal{B}$ with its interactions with $\mathcal{A}$ as follows:

– **Setup**: $\mathcal{B}$ selects a random session id $i^* \in \{0, \ldots, N-1\}$ together with one random element B^* from $Set_{PRP}(i^*)$. $\mathcal{B}$ selects a value $\alpha' \in \mathbb{Z}_q$ at random and defines

$$h = g_1^{-k^*} g^{\alpha'} \text{ with } k^* = i^* t^n + B^*.$$

$\mathcal{B}$ prepares $pk = (p, \mathbb{G}, \mathbb{G}_1, e, g, g_1, g_2, h)$ and transmits pk to $\mathcal{A}$.
For each key_i, $\mathcal{B}$ queries the PRP oracle for a random dummy message D_i with the slight adaptation that the PRP oracle does not output B^* for session id i^* (this corresponds to the attacker choosing a new message for which a signature is constructed/guessed as we will see later). This results in numbers B_i randomly selected from $Set_{PRP}(i)$ with $B_{i^*} \neq B^*$. $\mathcal{B}$ selects random numbers r_i in $\mathbb{Z}_p$. Next $\mathcal{B}$ computes for

$$k_i = it^n + B_i$$

the auxiliary information

$$aux_i = g_2^{\frac{-1}{k_i - k^*}} g^{r'_i},$$

which is given to $\mathcal{A}$. Notice that $k_{i^*} \neq k^*$ because $B_{i^*} \neq B^*$. Also $k_i = it^n + B_i \neq i^* t^n + B^* = k^*$ for $i \neq i^*$ because $0 \leq B^* < t^n$ and $0 \leq B_i < t^n$.

– **Query**: When $\mathcal{A}$ issues a message M_i for session id i, then $\mathcal{B}$ replaces the triple (i, D_i, B_i) with (i, M_i, B_i) in the list L_P of the PRP oracle. This is allowed since $\mathcal{B}$ has used the PRP oracle only a single time for key_i and, since it is chosen at random and therefore unknown to $\mathcal{A}$, no extra queries have been issued to the PRP oracle for key_i. From the perspective of $\mathcal{A}$ it is as if the PRP oracle produced B_i for $(key_i; M_i)$ (without loss of generality the dummy message D_i took the shape of M_i).

  Value $B_i = \sum_{j=0}^{n-1} b_{i,j} t^j$ is used to compute, as before, $k_i = it^n + B_i$. Together with the random r'_i previously chosen in Setup, $\mathcal{B}$ constructs

$$sk_{i,j,b_{i,j}} = g_2^{\frac{-\alpha'}{k_i - k^*}} F(it^n + b_{i,j}nt^j)^{r'_i} v'_{i,j},$$

  for $j \in \{0, \ldots, n-1\}$, with

$$v'_{i,j} = g^{\beta'_{i,j}} \text{ such that } \sum_{j=0}^{n-1} \beta'_{i,j} = 0.$$

  Values $\beta'_{i,j}$ are random in $\mathbb{Z}_p$ conditioned on their sum being equal to 0. $\mathcal{B}$ gives

$$sk_{i,B_i} = \{sk_{i,j,b_{i,j}}\}_{j=0}^{n-1} \text{ and } key_i$$

  to $\mathcal{A}$. Notice that from this moment onward $\mathcal{A}$ can query the PRP oracle for key_i (by assumption, at most Q_P times; notice that $\mathcal{B}$ queried the PRP oracle for key_i at most two times – if $i = i^*$ – and the size of $Set_{PRP}(i)$ is $Q_P + 2$).

We need to verify that the sk_{i,B_i} and aux_i have the correct form. That is, see (1),

$$aux_i = g^{r_i} \text{ and } sk_{i,j,b_{i,j}} = g_2^{\alpha} F(it^n + b_{i,j}nt^j)^{r_i} v_{i,j}$$

for some α, r_i, and $v_{i,j}$: We will use

$$\begin{aligned}\alpha &= a,\\ r_i &= r'_i - \frac{b}{k_i - k*},\\ v_{i,j} &= g_2^{\frac{-\alpha'}{k_i - k^*} - \alpha} F(it^n + b_{i,j}nt^j)^{r'_i - r_i} v'_{i,j}.\end{aligned}$$

Since r'_i is random, also r_i is random. Since all $v'_{i,j}$ are random conditioned on $\prod_{j=0}^{n-1} v'_{i,j} = 1$, the definition of $v_{i,j}$ transforms $sk_{i,j,b_{i,j}}$ into the required form if the product of all $v_{i,j}$ equals 1. We derive

$$\begin{aligned}
\prod_{j=0}^{n-1} v_{i,j} &= \prod_{j=0}^{n-1} g_2^{\frac{-\alpha'}{k_i-k^*}-\alpha} F(it^n + b_{i,j}nt^j)^{r'_i - r_i} \\
&= g_2^{\frac{-\alpha' n}{k_i-k^*}-n\alpha} \left(\prod_{j=0}^{n-1} g_1^{it^n+b_{i,j}nt^j} h \right)^{r'_i - r_i} \\
&= g_2^{\frac{-\alpha' n}{k_i-k^*}-n\alpha} (g_1^{(it^n+B_i)n} h^n)^{r'_i - r_i} \\
&= g_2^{\frac{-\alpha' n}{k_i-k^*}-na} (g_1^{k_i n} (g_1^{-k^*} g^{\alpha'})^n)^{\frac{b}{k_i-k^*}} \\
&= g^{\frac{-\alpha' nb}{k_i-k^*}-nab} g^{k_i na \cdot \frac{b}{k_i-k^*}} (g^{-k_i^* na} g^{\alpha' n})^{\frac{b}{k_i-k^*}} = 1.
\end{aligned}$$

Also

$$aux_i = g_2^{\frac{-1}{k_i-k^*}} g^{r'_i} = g^{\frac{-b}{k_i-k^*}} g^{r'_i} = g^{r_i}$$

as required.

– **Guess**: Adversary $\mathcal{A}$ produces a signature (σ, M^*) for session $\hat{i}$ such that $M^* \neq M_{\hat{i}}$. $\mathcal{B}$ calls the PRP oracle for input $(\hat{i}, M^*)$ which returns a value $\hat{B}$ from $Set_{PRP}(\hat{i})$. If $(\hat{i}, \hat{B}) = (i^*, B^*)$, then $\mathcal{B}$ will use signature σ in its attempt to solve CDHP.

If $(\hat{i}, \hat{B}) \neq (i^*, B^*)$, then $\mathcal{B}$ repeats the same computation with the same random tape for $\mathcal{A}$, the same hash and PRP oracles, but with a fresh random chosen (i^*, B^*) in Setup and fresh random choices for $v'_{i,j}$ and r'_i by $\mathcal{B}$ in Query and Setup. Notice that even though this changes k^* and as a consequence also the values for h and all the key material that will be revealed to $\mathcal{A}$, to $\mathcal{A}$ it seems that all this key material is independent of k^* as it matches their generation with the random looking values $v_{i,j}$ and r_i. Therefore, $\mathcal{A}$'s choice of $(\hat{i}, M^*)$ and the resulting $\hat{B}$ are independent of k^*. Since there are N choices for i^* and $Q_P + 1$ choices for B^* (notice that $B^* \in Set_{PRP}(i^*) \setminus \{B_{i^*}\}$ which has $Q_P + 1$ elements) it will take $T(Q_P + 1)N$ expected time until a usable signature σ is computed (for which $(\hat{i}, \hat{B}) = (i^*, B^*)$ is a lucky guess during Setup).

$\mathcal{B}$ simulates $\mathcal{A}$ a second time for (1) the same random tape (this implies that both the first and second simulation select the same value s when producing the final signature, hence, the same value x is used), (2) the same PRP oracle with the same sequence $\{key_i\}$, but with (3) a random oracle which simulates a different hash function $H'(\cdot)$. $\mathcal{B}$ does its simulation for the same (i^*, B^*) pair and we call a signature valid if it verifies properly using Verify and is usable in that $(\hat{i}, \hat{B}) = (i^*, B^*)$ as described above (this takes another $T(Q_P + 1)N$ expected time). Since the PRP oracle defines permutations, this is equivalent to generating a signature for (i^*, M^*) that verifies properly and where M^* is output by the first simulation. Since the PRP oracle with $\{key_i\}$ is fixed, the signature scheme becomes generic in that, for $\sigma' = (x, y, z)$, values y and z only depend on x, $u = H(M^*, x)$, and secret keys, but no other randomly selected values. This allows us to apply the forking lemma.

We apply the forking lemma which states that $\mathcal{B}$ is able to use $\mathcal{A}$ to produce two valid signatures for (i^*, M^*):

$$\sigma' = (x, y, z) \text{ with hash value } u = H(M, x)$$

and

$$\sigma" = (x, y', z') \text{ with hash value } u' = H'(M, x)$$

such that $u \neq u'$ in expected time $T' \leq 84480 \cdot 2T(Q_P + 1)NQ_H/\epsilon$.

Since both signatures are valid we know

$$\begin{aligned} e(g, z) &= e(g_1, g_2^{nu} x y^{k^*}) \times e(y, h), \\ e(g, z') &= e(g_1, g_2^{nu'} x y'^{k^*}) \times e(y', h). \end{aligned}$$

This implies that

$$(e(g, z) \times e(g, z')^{-1})^{\frac{1}{n(u-u')}} = e(g, (z/z')^{\frac{1}{n(u-u')}})$$

is equal to the product of

$$\begin{aligned} &(e(g_1, g_2^{nu} x y^{k^*}) \times e(g_1, g_2^{nu'} x y'^{k^*})^{-1})^{\frac{1}{n(u-u')}} \\ &= e(g_1, g_2^{n(u-u')} (y/y')^{k^*})^{\frac{1}{n(u-u')}} \\ &= e(g_1, g_2 (y/y')^{\frac{k^*}{n(u-u')}}) \\ &= e(g_1, g_2) \times e(g_1, (y/y')^{\frac{k^*}{n(u-u')}}) \end{aligned}$$

and

$$\begin{aligned} &(e(y, h) \times e(y', h)^{-1})^{\frac{1}{n(u-u')}} \\ &= e(y/y', h)^{\frac{1}{n(u-u')}} \\ &= e(y/y', g_1^{-k^*} g^{\alpha'})^{\frac{1}{n(u-u')}} \\ &= (e(y/y', g_1^{-k^*}) \times e(y/y', g^{\alpha'}))^{\frac{1}{n(u-u')}} \\ &= e((y/y')^{\frac{-k^*}{n(u-u')}}, g_1) \times e((y/y')^{\frac{\alpha'}{n(u-u')}}, g). \end{aligned}$$

This proves

$$\begin{aligned} &e(g, (z/z')^{\frac{1}{n(u-u')}}) \\ &= e(g_1, g_2) \times e(g_1, (y/y')^{\frac{k^*}{n(u-u')}}) \\ &\quad \times e((y/y')^{\frac{-k^*}{n(u-u')}}, g_1) \times e((y/y')^{\frac{\alpha'}{n(u-u')}}, g) \\ &= e(g_1, g_2) \times e(g, (y/y')^{\frac{\alpha'}{n(u-u')}}). \end{aligned}$$

By using $g_1 = g^a$ and $g_2 = g^b$ we obtain

$$e(g, ((z/z')/(y/y')^{\alpha'})^{\frac{1}{n(u-u')}}) = e(g, g^{ab})$$

from which we conclude

$$((z/z')/(y/y')^{\alpha'})^{\frac{1}{n(u-u')}} = g^{ab}.$$

This means that the two valid signatures allow $\mathcal{B}$ to compute g^{ab} and this solves CDHP. We notice that the above reduction, which only uses the validity checks of the two signatures, is not present in [3]; the above analysis can be adapted to [3] in order to properly complete their security analysis.

Checking $s + u \neq 0$. The signing procedure of [3] repeats selecting a random s and computing $u = H(M, x)$ until $s + u \neq 0$. When verifying we may check $g_2^{nu}x \neq 1$ as this corresponds to $s + u \neq 0$. We notice that in our security analysis we do not need this additional property $s + u \neq 0$. But in practice we may exclude $s + u = 0$ since this is easy to do and if not excluded, then any one who happens to solve $s + H(M, g_2^{ns}) = 0$ (by using some table enumeration method) will immediately be able to use this to impersonate a corresponding signature (for $s + u = 0$, $y = z = 1$).

5 Implementation and Performance Evaluation

In this section, we give the details of our implementation and performance analysis for our bilinear map based OTS-SKE scheme. We evaluate the performance of our scheme and compare our results to the hash-based OTS-SKE in [6]. This is done by implementing the Remote Attestation protocol of [6] and simulate its performance with our bilinear map based OTS-SKE scheme and with the hash-based scheme.

5.1 Experimental Setup

All experiments are conducted on a PC with Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz and 32GB memory and on Windows 10 based operating system. Intel Software Guard Extensions (SGX) is used to provide secure enclaves for the RA scheme. We have written our own C++ implementation along with an EPUF-extended PUF interface simulation (see [6] for details and explanation of the EPUF functionality).

We constructed our bilinear map based OTS-SKE scheme using the open-source MIRACL Multiprecision Integer Cryptographic Library[7] that includes elliptic cryptography arithmetic. We used C++ language for our implementation, along with fast in-line assembly language alternatives for most critical parts of our code to speed up performance, such as modular multiplication and exponentiation. For the random number generation (RNG) in our implementation,

[7] https://github.com/miracl/MIRACL

Table 2. Amortized Cost Analysis per Remote Attestation (RA) session of the bilinear map based OTS-SKE scheme in Comparison with the hash based OTS-SKE scheme in a 1024-session setup. In the bilinear map based scheme we use $t = 4$ (such that $(t/\log_2 t) \cdot 128 = 256$ and $128/\log_2 t = 64$). In the hash based OTS-SKE scheme q' is the number of hash calls needed for constructing a Merkle tree with q leaves, s is chosen such that $\binom{q}{s} > 2^{256}$, and parameter $l = \log_2 N$. The classical security of both schemes is 256 bits and a message signed during a RA session has 128 bits.

	Bilinear Map based OTS-SKE				**Hash based OTS-SKE**			
	Operation	**Time(ms)**	**Repeats**	**Total(ms)**	**Operation**	**Time(ms)**	**Repeats**	**Total(ms)**
Key Generation	sk and aux Gen.	25	257	6425	sk Gen.	0.1	261 (q)	26.1
	EPUF Interface	2.24	256	573.4	vk ($= aux$) Gen.	0.1	261 (q)	26.1
	Encryption	0.3	256	76.8	EPUF Interface	2.24	261 (q)	584.6
	Masking Encryption Key	0.0003	256	0.077	Encryption	0.3	261 (q)	78.3
	$v_{i,j}$ Generation	4	128	512	Masking Encryption Key	0.0003	261 (q)	0.078
					Bit-Mask and Hash-Key Gen.	0.3	260 ($q' + \frac{N-1}{N}$)	80.7
					Merkle Tree Construction	0.1006	260 ($q' + \frac{N-1}{N}$)	26.2
	Total			***7587.3***	Total			***819.3***
	Total without PUF			***7013.9***	Total without PUF			***234.7***
Sign	EPUF Interface	6.88	64	440.3	EPUF Interface	6.88	130 (s)	894.4
	Unmasking Encryption Key	0.0003	64	0.019	Unmasking Encryption Key	0.0003	130 (s)	0.039
	Decryption	0.3	64	19.2	Decryption	0.3	130 (s)	39
	Signing	100	1	100	Mapping (ϕ)	1.5	1	1.5
					$\widehat{MT}$ Authentication Path Gen.	0.4006	15 ($\frac{3(l-1)}{2} + 1$)	6.01
	Total			***559.5***	Total ($934.9 + 0.6 \cdot l$)			***940.9***
	Total without PUF			***119.2***	Total without PUF ($40.55 + 0.6 \cdot l$)			***46.55***
Verify	Pairing for verification	33.3	3	100	Mapping (ϕ)	1.5	1	1.5
	Precomputation for pairing	0.01	1	0.01	vk Gen.	0.1	130 (s)	13
					MT_i Root Verification	0.1006	259 (q')	26
					Bit-Mask and Hash-Key Gen.	0.3	269 ($q' + l$)	80.7
					$\widehat{MT}$ Authentication Path Ver.	0.1006	10 (l)	1.006
	Total			***100***	Total ($118.2 + 0.4 \cdot l$)			***122.2***

we used a PRNG provided by MIRACL library, whose seed is then replaced by a random number generated from the Intel SGX's RNG.

In order to fairly compare our results against the hash-based OTS-SKE [6], we conduct our experiments on the same processor using the same experimental settings. We use the same RNG and TCB for both schemes. For the Extended PUF and One Time Pad (OTP) interfaces in the RA scheme, we use the same simulation and parameter settings as in [6]. Therefore, for the experimental details and results of EPUF and OTP interfaces, we refer readers to Sections 5.2 and 5.3 with Tables 4, 5 and 6 of [6] for timing results. The runtime, communication and storage cost analysis of hash based OTS-SKE, which is presented here for comparison in this section, are directly retrieved from Section 5.4 with Tables 7, 8 and 9 of [6].

5.2 Experimental Results of Bilinear Map based OTS-SKE and Comparison with Hash based OTS-SKE

Runtime Cost Analysis. Today, the wide employment of elliptic curve cryptography (ECC) in various applications relies on a variety of implementation types from pure software or hardware implementations to hardware and software co-design. However, pure software implementations of ECC, despite offering best flexibility at lowest cost, cannot cope with the speed demands of many application areas as general purpose processors are not designed for efficient handling of

ECC's underlying finite field arithmetic. This makes software-based implementations impractical for applications in time-constrained environments that require high throughput and processing. Considering these limitations, hardware-based implementations turn out to be the more suitable alternatives. Despite this, in this paper, we evaluate a software-based implementation of the ECC-based signature scheme that has its own computational disadvantages, but can also be potentially accelerated using HW techniques which will be discussed later.

– **Key Generation.** During key generation the public key $pk = (p, \mathbb{G}, \mathbb{G}_1, e, g, g_1, g_2, h)$ together with $tn \times N$ secret keys are generated. For our evaluation, we experiment with 1 session and choose $N = 1$ with $n \log_2 t = 128$ (we will extrapolate the experimental results to a larger $N = 1024$). This means that a message of n bits can be mapped by means of a PRP to a string of size $128/\log_2 t$ with t-ary symbols that are each represented by $\log_2 t$ bits. We notice that such a string can indeed again be characterized by $(128/\log_2 t) \cdot \log_2 t = 128$ bits. This gives $(t/\log_2 t) \cdot 128$ secret keys in total. We will use $t = 4$ as this minimizes key generation run time and as secondary objective minimizes the signing run time.
For pk, the points g, g_1, g_2 are randomly generated from $\mathbb{G}$ (on the elliptic curve), which takes about 100 ms for each. In addition the points are preprocessed after they are generated; this prepares them for faster computation for pairing (as well as elliptic curve arithmetic operations) for the future, which takes about 10^{-2} ms for each. These points are generated and preprocessed only once at the beginning of key generation. Thus, despite being very costly, the overhead of the pk generation can be considered as negligible while working in a sufficiently large timing window (for example, if we have $N > 100$ sessions). Therefore, in a 1024-session setup, we ignore this pk generation cost. On the other hand, in the initialization phase of the hash based OTS, along with the generation of sk, we also include the vk generation in the timing table, since a different set of vk needs to be generated in each section (unlike the pk in bilinear map based OTS).
After generating pk, secret keys are generated. In order to generate secret keys, random elements $\beta_{i,j} \in \mathbb{Z}_p$, $i \in \{0, \ldots, N-1\}$, $j \in \{0, \ldots, n-1\}$, are first generated. As previously stated, since random point generation on the elliptic curve is costly, we reduce the cost of generating random points $v_{i,j} \in \mathbb{G}$ by computing $v_{i,j} = g^{\beta_{i,j}}$ which takes approximately 4 ms. Compared to the method of selecting $v_{i,j}$ from $\mathbb{G}$ randomly for each i and j (100 ms), this is a significant speedup of almost 25 times.
Generating 1 secret key $sk_{i,j,b}$ takes approximately 25 ms. Since we have $(t/\log_2 t) \cdot 128 = 256$ keys for $t = 4$, we repeat this $(t/\log_2 t) \cdot 128 = 256$ times. The elliptic curve computations during key generation take 6.4 sec for 1 session, see Table 2. After these computations, EPUF calls are made to obtain responses for 256 symmetric encryption keys which takes about 573 ms. This is the second biggest overhead after the costly ECC key generation. Secret keys $sk_{i,j,b}$ are then encrypted with the symmetric encryption keys and EPUF responses are then used to mask the symmetric encryption keys.

Table 3. Storage Cost of Bilinear Map based and Hash based OTS-SKE Schemes during Initialization for $N = 1024$ sessions.

Bilinear Map based OTS-SKE				Hash based OTS-SKE			
	Size (bits)	**Quantity**	**Total (MB)**		**Size (bits)**	**Quantity**	**Total (MB)**
pk	2048	1	2.5×10^{-4}	$vk\ (= aux)$	256	261×1024	8.2
sk	256	256×1024	8	sk	256	261×1024	8.2
aux	256	1024	3×10^{-2}	CRP	3312	261×1024	105.5
CRP	3312	256×1024	103.5	Masked AES Key	128	261×1024	4.1
Masked AES Key	128	256	4	RNG Seed	256	1	3.2×10^{-5}
				MT_i Roots	256	1024 (N)	3.3×10^{-2}
Total			115.5	Total ($3 \cdot 10^{-5} + 0.12 \cdot N$)			125.9

As it can be seen from Table 2, EPUF interface is an overhead for both schemes during key generation. However, for bilinear map based OTS-SKE, key generation excluding PUF is still costly due to the costly arithmetic of ECC during key generation.

– **Signing.** During this phase, based on the received nonce, $128/\log_2 t = 64$ encrypted secret keys along with their corresponding masked encryption keys are fetched from the memory. In order to unmask the encryption keys, 64 calls are made to the EPUF interface to obtain the responses which takes about 440 ms. The encryption keys are then unmasked, which is a much faster operation, and the secret keys are decrypted which takes 19.2 ms. Then the signature is generated which takes approximately 100 ms. As Table 2 shows, all components taken together costs 560 ms, about half a second. If we exclude calls to the extended PUF interface, then it takes only 119 ms in which the majority of the overhead comes from ECC related operations.

– **Verification.** Verification contains less components compared to the signing phase and is performed outside the enclave by the remote user. Despite containing less operations, verification is still costly (100 ms) since pairing on the elliptic curve is performed 3 times in order to verify the signature. On the other hand, hash based OTS-SKE verification phase is more complex and costlier than bilinear based solution due to the additional merkle tree construction and authentication computations.

Storage Cost. In addition to runtime cost of both methods presented previously, we also compare the storage cost in Table 3. Here, we assume that key generation is for 1024 sessions and analyze the cost based on this assumption. During initialization, we use 256 keys for bilinear map based OTS and 261 keys for hash based OTS, which both use 256-bit keys. Therefore, the memory cost for storing the sk, EPUF challenges and 128-bit masked AES keys are slightly more for the hash based scheme. However, the vk that constitute the auxiliary information aux in the hash based OTS is much larger compared to the aux

Table 4. Communication Cost between the RA Enclave and Remote User during Signing (N=1024 sessions) of the Bilinear based and Hash based OTS-SKE. (E=Sent by Enclave, U=Sent by Remote User)

Bilinear Map based OTS-SKE				**Hash based OTS-SKE**			
Component	**Size (bits)**	**Quantity**	**Total Size (KB)**	**Component**	**Size (bits)**	**Quantity**	**Total Size (KB)**
Nonce (U)	256	1	0.031	Nonce (U)	256	1	0.031
Signature (E)	768	1	0.093	sk in Signature (E)	256	130 (s)	4.06
				vk in Signature (E)	256	131 ($q - s$)	4.09
				Authentication Path (E)	256	10 (l)	0.31
Total			0.125	Total ($8.2 + 0.03 \cdot l$)			8.5

in the bilinear map based scheme. This is because each $sk_{i,j}$ in sk_i in the hash based scheme has a corresponding $vk_{i,j}$, while the auxiliary information in the bilinear map based scheme is a single element for the whole session key sk_i. In this case, the hash based scheme has to store 8.2 MB of vk for each session it sets up while the bilinear map based scheme only needs to store 1 KB per session.

For the hash based OTS-SKE, the majority of the storage cost for Merkle tree is eliminated, since the remote user generates upper-level Merkle tree authentication nodes on the fly during verification (as reflected in Table 2). However, for this, the 256-bit RNG seed needs to be stored, as it is used to regenerate the bit-mask and hash-keys. Also, since constructing the lower-level Merkle tree was costly during signing, only the root hash of lower-level Merkle trees in each session to avoid this computation. Overall, the hash based signature scheme has slightly more storage cost (approx. 10 MB) than bilinear map based scheme for 1024 sessions. However, this cost grows linearly in the number of sessions N.

Communication. Table 4 compares the size of data transmitted between the remote user and the remote attestation enclave during the signing phase. The bilinear map based solution is quite straightforward: After receiving the 256-bit nonce (which is merged with the to-be signed message in order to prove freshness), a 768-bit signature σ is sent to the remote user. No other communication needs to be performed between the remote user and the enclave. Although the procedure is the same for hash based OTS-SKE, the signature is 43× longer. Additionally, the rest of the $q - s = 131$ $vk_{i,j}$ keys from aux for the current session need to be sent to the user along with the $\log_2 1024 = 10$ merkle tree nodes required in the authentication path for the user to calculate the root hash for verification. This shows that the bilinear map based OTS-SKE significantly reduces communication cost.

Lines of Code (LoC) within the Enclave. The code that remains inside the enclave has to be trusted to not have a vulnerability, although this can be exploited during interactions with the outside world. Therefore, the LoC is an important indicator with respect to trust. The hash based OTS-SKE scheme is a much more straightforward and compact solution in terms of implementation compared to bilinear map based OTS-SKE. We have used the MIRACL library for the bilinear map based solution and have only included the components that are needed to perform necessary computations for the signature scheme. 7k+ lines of code have to reside within the enclave for the bilinear map based solution whereas the hash based solution includes less than 500 lines of code.

Potential for Acceleration. For both solutions, the extended EPUF interface leads to a significant overhead during key generation and signing which is due to the excessive amount of PUF calls performed. Potentially, in a practical application, the EPUF interface can entirely be pushed into hardware with more circuitry and can become a component of the TCB. Depending on the availability of resources, the number of PUFs can be increased in the TCB, which may allow for more parallelism and acceleration. Overall, this is a micro architectural problem and can be considered in the larger processor architecture context. Indeed, this type of PUF acceleration method can be applied in both hash based and bilinear map based OTS-SKE schemes. Excluding the PUF calls, we have shown that the bilinear map based solution remains expensive and the elliptic curve related computations are the main overhead. HW-based ECC acceleration methods have been frequently explored and many hardware architectures have been proposed for faster and more compact solutions [1, 11, 13, 15, 18, 23, 24], including several methods that exploit the computing power of graphics processing units (GPU) for ECC computations in the last decade [4, 20, 22, 25, 27]. For example, Pan *et al.* [20] has shown that, with GPU utilization, the elliptic curve digital signature algorithm (ECDSA) can achieve 8.71×10^6 Operations Per Second (OPS) for signature generation and 9.29×10^5 OPS for verification. Zhang *et al.* [27] has proposed an ECC GPU-based library for bilinear pairing called "EAGL" which can achieve 3350.9 pairings/sec on GPU at the 128-bit security level. Zhang *et al.* [27] achieved 8.7 ms per pairing on a 1024-bit security level with GPU utilization, which is a $20\times$ speedup compared to CPU implementations. This makes our bilinear map based OTS-SKE scheme more suitable for further acceleration and improvements (e.g., with the use of a cryptographic processor) in addition to improvements over the EPUF interface.

6 Conclusion

We introduced a bilinear map based OTS-SKE scheme with an orders of magnitude smaller signature size compared to the existing hash based OTS-SKE scheme. Signing takes only 60% of the time required by the hash based OTS-SKE scheme. The main disadvantages of the bilinear map based scheme are that it is not post quantum secure and key generation takes about $9\times$ longer (although the latter can be improved using ECC hardware acceleration). Without sufficient quantum computing resources easily available, the bilinear map based OTS-SKE is the best choice.

Acknowledgements

This research was supported by the National Science Foundation under Grants No. 1617774 and 1929261.

References

1. Agnew, G., Mullin, R., Vanstone, S.: An implementation of elliptic curve cryptosystems over f2155. IEEE J. Sel. Areas Commun. **11**, 804–813 (1993)
2. Bos, J.N., Chaum, D.: Provably unforgeable signatures. In: Annual International Cryptology Conference. pp. 1–14. Springer (1992)
3. Chow, S.S., Hui, L.C., Yiu, S.M., Chow, K.: Secure hierarchical identity based signature and its application. In: International Conference on Information and Communications Security. pp. 480–494. Springer (2004)
4. Cui, S., Großschädl, J., Liu, Z., Xu, Q.: High-speed elliptic curve cryptography on the nvidia gt200 graphics processing unit. In: ISPEC (2014)
5. Dahmen, E., Okeya, K., Takagi, T., Vuillaume, C.: Digital Signatures Out of Second-Preimage Resistant Hash Functions. In: Buchmann, J., Ding, J. (eds.) Post-Quantum Cryptography, vol. 5299, pp. 109–123. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
6. van Dijk, M., Gurevin, D., Jin, C., Khan, O., Nguyen, P.H.: Autonomous secure remote attestation even when all used and to be used digital keys leak. IACR Cryptol. ePrint Arch. **2021** (2021)
7. Dodis, Y., Franklin, M.K., Katz, J., Miyaji, A., Yung, M.: Intrusion-Resilient Public-Key Encryption. In: Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings. pp. 19–32 (2003)
8. Dodis, Y., Franklin, M.K., Katz, J., Miyaji, A., Yung, M.: A Generic Construction for Intrusion-Resilient Public-Key Encryption. In: Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. pp. 81–98 (2004)
9. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-Insulated Public Key Cryptosystems. In: Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings. pp. 65–82 (2002)
10. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. In: Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings. pp. 130–144 (2003)
11. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L.: A survey of lightweight-cryptography implementations. IEEE Design Test of Computers **24**(6), 522–533 (2007)
12. Franklin, M.K.: A survey of key evolving cryptosystems. IJSN **1**(1/2), 46–53 (2006)
13. Gao, L., Shrivastava, S., Sobelman, G.: Elliptic curve scalar multiplier design using fpgas. In: CHES (1999)
14. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-Based Hierarchical Strongly Key-Insulated Encryption and Its Application. In: Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings. pp. 495–514 (2005)

15. Huang, M., Gaj, K., El-Ghazawi, T.: New hardware architectures for montgomery modular multiplication algorithm. IEEE Transactions on Computers **60**(7), 923–936 (2011)
16. Itkis, G.: Forward security, adaptive cryptography: Time evolution (2004)
17. Lamport, L.: Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International (1979)
18. Leung, K.H., Ma, K.W., Wong, W., Leong, P.: Fpga implementation of a microcoded elliptic curve cryptographic processor. Proceedings 2000 IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No.PR00871) pp. 68–76 (2000)
19. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Conference on the theory and application of cryptographic techniques. pp. 369–378. Springer (1987)
20. Pan, W., Zheng, F., Zhao, Y., Zhu, W., Jing, J.: An efficient elliptic curve cryptography signature server with gpu acceleration. IEEE Transactions on Information Forensics and Security **12**(1), 111–122 (2017)
21. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**(3), 361–396 (2000)
22. Pu, S., Liu, J.: Eagl: An elliptic curve arithmetic gpu-based library for bilinear pairing. In: Pairing (2013)
23. Satoh, A., Takano, K.: A scalable dual-field elliptic curve cryptographic processor. IEEE Transactions on Computers **52**(4), 449–460 (2003)
24. Sutikno, S., Effendi, R., Surya, A.: Design and implementation of arithmetic processor f/sub 2//sup 155/ for elliptic curve cryptosystems. IEEE. APCCAS 1998. 1998 IEEE Asia-Pacific Conference on Circuits and Systems. Microelectronics and Integrating Systems. Proceedings (Cat. No.98EX242) pp. 647–650 (1998)
25. Szerwinski, R., Güneysu, T.: Exploiting the power of gpus for asymmetric cryptography. In: CHES (2008)
26. Watanabe, Y., Shikata, J.: Identity-Based Hierarchical Key-Insulated Encryption Without Random Oracles. In: Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I. pp. 255–279 (2016)
27. Zhang, Y., Xue, C., Wong, D., Mamoulis, N., Yiu, S.: Acceleration of composite order bilinear pairing on graphics hardware. IACR Cryptol. ePrint Arch. **2011**, 196 (2012)