Flexible Signatures: Making Authentication Suitable for Real-Time Environments

Duc Le, Mahimna Kelkar, and Aniket Kate

Purdue University
{le52,mkelkar,aniket}@purdue.edu

Abstract. This work introduces the concept of flexible signatures. In a flexible signature scheme, the verification algorithm quantifies the validity of a signature based on the number of computations performed, such that the signature's validation (or confidence) level in $[0, 1]$ improves as the algorithm performs more computations. Importantly, the definition of flexible signatures does *not* assume the resource restriction to be known in advance, a significant advantage when the verification process is hard stopped by a system interrupt. Although prominent traditional signature schemes such as RSA, (EC)DSA, EdDSA seem unsuitable towards building flexible signatures, we find the use of the Lamport-Diffie one-time signature and Merkle authentication tree to be suitable for building flexible signatures. We present a flexible signature construction based on these hash-based primitives and prove its security with a concrete security analysis. We also perform a thorough validity-level analysis demonstrating an attractive computation-vs-validity trade-off offered by our construction: a security level of 80 bits can be ensured by performing only around $\frac{2}{3}$rd of the total hash computations for our flexible signature construction with a Merkle tree of height 20. Finally, we have implemented our constructions in a resource-constrained environment on a Raspberry Pi. Our analysis demonstrates that the proposed flexible signature design is comparable to other standard signature schemes in term of running time while offering a quantified level of security at each step of the verification algorithm.
We see this work as the first step towards realizing the flexible-security cryptographic primitives. Beyond flexible signatures, our flexible-security conceptualization offers an interesting opportunity to build similar primitives in the asymmetric as well as symmetric cryptographic domains. Apart from being of independent theoretical interest, these flexible security primitives can be of practical value to real-time systems as well as the Internet of things: rigid all-or-nothing guarantees offered by the traditional cryptographic primitives have been particularly unattractive in these unpredictably resource-constrained environments.

1 Introduction

Security for embedded and real-time systems has become a greater concern with manufacturers increasing connectivity of these traditionally isolated control networks to the outside world. The computerization of hitherto purely mechanical elements in vehicular networks, such as connections to the brakes, throttle, and steering wheel, has led to a life-threatening increase of exploitation power. In the event that an attacker gains access to an embedded control network, safety-critical message traffic can be manipulated inducing catastrophic system failures. In recent years, numerous attacks have impressively demonstrated that the software running on embedded controllers could be successfully exploited, often even remotely [17, 23, 26]. With the rise of the Internet of Things (IoT), more non-traditional embedded devices have started to get integrated into personal and commercial computing infrastructures, and security will soon become a paramount issue for the new-age embedded systems [10, 28].

Well-established authentication and integrity protection mechanisms such as digital signatures or MACs can effectively solve many of the security issues with embedded systems. However, the industry is hesitant to adopt those as most embedded devices pose severe resource constraints on the security architecture regarding memory, computational capacity, energy and time. Given the real-time deadlines, the embedded devices might not be able complete verifications by the deadline rendering *all* verification efforts useless.

Indeed, traditional cryptographic primitives are not designed for such uncertain settings with unpredictable resource constraints. Consider prominent digital signature schemes (such as RSA, EC-DSA, and EdDSA) that allow a signer who has created a pair of private and public keys to sign messages so that any verifier can later verify the signature with the signer's public key. The verification algorithms of those signature schemes are deterministic and only return a binary answer for the validity of the signature (i.e., 0 or 1). Such verification mechanisms may be unsatisfactory for an embedded module with unpredictable computing resources or time to perform the verification: if the module can only partially complete the verification process due to resource constraints or some *unplanned* real-time system interrupt, there are no partial validity guarantees available.

This calls for a signature scheme that can quantify the validity of the signature based on the number of computations performed during the verification. In particular, for a signature scheme instantiation with 128-bit security, we expect the verification process to be flexible enough to offer a validity (or confidence) level in $[0, 1]$ based on the resources available during the verification process. We observe that none of the existing signature schemes offer such a trade-off between the computation time/resource and the security level in a flexible manner.

Our Contributions. This paper initiates the study of cryptographic primitives with flexible security guarantees that can be of tremendous interest to real-time systems and the emerging IoT. In particular, we investigate the notion of a flexible signature scheme that offers partial security for an unpredictably partial verification.

As the first step, based on the standard definition of digital signatures, we propose a new definition of a signature scheme with a flexible verification algorithm. Here, instead of returning a binary answer, the verification algorithm returns a value, $\alpha \in [0, 1] \cup \perp$ that quantifies the validity of the signature based on number of computations performed.

Next, we provide a provably secure construction of the flexible signature scheme based on the Lamport-Diffie one-time signature construction [19] and the Merkle authentication tree [21]. The security of our signature relies on the difficulty of finding a ℓ-near-collision pair for a collision-resistant hash function. Through our analysis, we demonstrate that our construction still offers a high security level against adaptive chosen message attacks despite performing fewer computations during verification. For example, a security level of 80-bit security can be secured by performing only around $\frac{2}{3}$rd of the total required hash computations for a Merkle tree of height 20.

Finally, we prototype our solution in a resource-constrained environment by implemented our construction on a Raspberry Pi. We find that performance of the proposed constructions are comparable to other prominent signature schemes in term of running time, while offering flexible trade-off between the security level and the number of computations. Importantly, neither the security level nor the number of computations have to be pre-determined during verification.

Related Work. Fischlin [13] proposed a similar framework for progressively verifiable message authentication codes (MACs). In particular, he presented two concrete constructions for progressively verifiable MACs that allow the verifier to spot errors or invalid tags after a reasonable number of computations. Also, the paper introduced the concept of detection probability to denote the probability that the verifier detects errors after verifying a certain number of blocks. In this work, we address the open problem of a progressively verifiable digital signature scheme, and we incorporate the detection probability concept into the security analysis of our schemes.

Bellare, Goldreich, and Goldwasser [3] introduced incremental signatures. Here, given a signature on a document, a signer can obtain a (new) signature on a similar document by partially updating the available signature. The incremental signature computation is more efficient than computing a signature from scratch and thus can offer some advantage to a resource-constrained signer. However, it provides no benefit for a resource-constrained verifier; the verifier still needs to perform a complete verification of the signature.

Signature scheme with batch verification [2, 8] is a cryptographic primitive that offers an efficient verifying property. Namely, after receiving multiple signatures from different sources, a verifier can efficiently verify the entire set of signatures at once. Batch verification signature scheme and flexible signature scheme are similar in that they offer an efficient and flexible verification mechanism. However, while the batch verification signature merely seeks to reduce the load on a busy server, the flexible signature focuses on a resource-constrained verifier who can tolerate a partial security guarantee from a signature.

Freitag et. al. [14] proposed the concept of signature scheme with randomized verification. Here, the verifying algorithm takes as input the public key along with some random coin to determine the validity of the signature. In those schemes, the attacker's advantage of forging a valid message-signature pair, (m^*, σ^*), is determined by the fraction of coins that accept (m^*, σ^*). Freitag et. al. constructed a signature scheme with randomized identity-based encryption (IBE) schemes using Naor's transformation and show that the security level of their signature scheme is fixed to size of the underlying IBE scheme's identity space. While our work can be formally defined as a signature scheme with randomized verification, our scheme offers a more flexible verification in which the security level of the scheme can be efficiently computed based on the output of the verifying algorithm.

Finally, Fan, Garay, and Mohassel [11] proposed the concept of short and adjustable signature scheme. They offered three variants, namely setup adjustable, signing adjustable, and verification adjustable signatures offering different trade-offs between the length and the security of the signature. The first two variants allow the signer to adjust the length of the signature, while the last variant allows the verifier to shorten the signature during the verification phase. They presented three constructions for each variant based on indistinguishably obfuscation ($i\mathcal{O}$), and one concrete construction *only* for the setup-adjustable variant based on the BLS Signature Scheme [5]. Unfortunately, none of those constructions is suitable for constructing flexible signatures tolerating unpredictable interrupts.

2 Preliminaries

Fig. 1 presents prominent notational conventions that we use throughout this work. Our constructions employ the following standard properties of cryptographic hash functions.
We use $H : \mathcal{K}\times\mathcal{M} \to \{0,1\}^n$ to denote a family of hash functions that is parameterized by a key $k \in \mathcal{K}$ and message $m \in \mathcal{M}$ and outputs a binary string of length n. For this work, we consider two security properties for hash functions from [25], preimage resistance, collision resistance, and one weaker security notion from [18, 20], ℓ-near collision resistance.

Preimage Resistance: We call a family H of hash functions (t_{ow}, ϵ_{ow})-preimage resistant, if for any $\mathcal{A}$ that runs for at most t_{ow}, the adversary's advantage is:

$$\mathsf{Adv}^{ow}_{H}(\mathcal{A}) = \Pr\left[\begin{matrix} k \xleftarrow{\$} \mathcal{K}, x \xleftarrow{\$} \mathcal{M} \\ y \leftarrow H(k,x), x' \leftarrow \mathcal{A}(k,y) \end{matrix} : H(k,x') = y\right] \leq \epsilon_{ow}$$

Collision Resistance: We call a family H of hash functions (t_{cr}, ϵ_{cr})-collision resistant, if for any $\mathcal{A}$ that runs for at most t_{cr}, the adversary's advantage is:

$$\mathsf{Adv}^{cr}_{H}(\mathcal{A}) = \Pr\left[\begin{matrix} k \xleftarrow{\$} \mathcal{K} \\ (x,x') \leftarrow \mathcal{A}(k) \end{matrix} : (x \neq x') \wedge (H(k,x) = H(k,x'))\right] \leq \epsilon_{cr}$$

ℓ-near-collision Resistance: We call a family H of hash functions $(t_{\ell\text{-}ncr}, \epsilon_{\ell\text{-}ncr})$-$\ell$-near-collision resistant, if for any $\mathcal{A}$ that runs for at most $t_{\ell\text{-}ncr}$ and $0 \leq \ell \leq n$, the adversary's advantage is:

$$\mathsf{Adv}^{ncr}_{H,\ell}(\mathcal{A}) = \Pr\left[\begin{matrix} k \xleftarrow{\$} \mathcal{K}; \\ (x,x') \leftarrow \mathcal{A}(k,\ell) \end{matrix} : (x \neq x') \wedge (\mathit{\Delta}(H(k,x), H(k,x')) \leq \ell)\right] \leq \epsilon_{\ell\text{-}ncr}$$

Generic Attacks. To find the preimage $t_{ow} = 2^q$ is required to achieve $\epsilon_{ow} = 1/2^{n-q}$ using exhaustive search. Due to the birthday paradox, however, only $t_{cr} = 2^{n/2}$ is required to find a collision with a success probability of $\epsilon_{cr} \approx 1/2$. Finally, Lamberger et. al. showed in [18] that at least $t_{\ell\text{-}ncr} = 2^{n/2}/\sqrt{\sum_{i=0}^{\ell}\binom{n}{i}}$ is required to find a ℓ-near-collision with a success probability of $\epsilon_{\ell\text{-}ncr} \approx 1/2$.

n	Security parameter
$[m]$	$\{1, \ldots, m\}$
$m_1 \| m_2$	Concatenation of strings m_1 and m_2
$(d_i)_{i\in[m]}$	Concatenation of m elements, $d_1 \| d_2 \| ... \| d_m$
$x \xleftarrow{\$} \mathcal{X}$	x is chosen uniformly at random from some set $\mathcal{X}$
$\Delta(x, y)$	Hamming distance between two binary strings x and y
$f(m) = \mathsf{poly}(m)$	$f(m)$ is a polynomial function in m
$f(m) = \mathsf{negl}(m)$	$f(m)$ is a negligible function in m, if $f(m) = o(1/m^c)\ \forall c \in \mathbb{N}$
$[\![r]\!]$	Optional parameter r in an algorithm definition

Fig. 1: Notations

Unkeyed Hash Functions. In practice, the key for standard hash functions is public; therefore, from this point, we refer to the cryptographic hash function H as a fixed function $H : \mathcal{M} \to \{0, 1\}^n$.

3 Security Definition

In this section, we define a flexible signature scheme. We adapt the standard definition of the signature scheme [16] to the flexible security setting. An instance of an interrupted flexible signature verification is expected to return a validity value, α, in the range $[0, 1]$.

To model the notion of runtime interruptions in the signature definition, we introduce the concept of an interruption oracle $\mathsf{iOracle}_\Sigma(1^n)$ for signature scheme Σ and give the verification algorithm access to it. The interruption oracle outputs an interruption position r in the sequence of computation steps involved the verification algorithm. For simplicity, if we denote max to be the maximum number of computations needed (e.g. clock cycles, number of hash computations, or modular exponentiations) for a signature verification, then $\mathsf{iOracle}_\Sigma(1^n)$ outputs a value $r \in \{0, \ldots, \mathsf{max}\}$. The specification of the interruption position may vary depending on the choice of signature scheme; e.g., in this work, we define the interruption position as the number of hash computations performed in the verification algorithm.

Definition 1. *A flexible signature scheme, Σ, contains three algorithms (*$\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}$*):*

- $\mathsf{Gen}(1^n)$ *is a probabilistic algorithm that takes a security parameter 1^n as input and outputs a pair (pk, sk) of public key and secret key.*
- $\mathsf{Sign}(sk, m)$ *is a probabilistic algorithm that takes a private key sk and a message m from a message space $\mathcal{M}$ as inputs and outputs a signature σ from signature space $\mathcal{S}$.*
- $\mathsf{Ver}(pk, m, \sigma, [\![r]\!])$ *is a probabilistic algorithm that takes a public key pk, a message m, a signature σ, and an optional interruption position $r \in \{0, \ldots, \mathsf{max}\}$ as inputs. If r is not provided, then the algorithm will query an interruption oracle,* $\mathsf{iOracle}_\Sigma(1^n)$ *to determine $r \in \{0, \ldots, \mathsf{max}\}$. The algorithm outputs a real value $\alpha \in [0, 1] \cup \{\perp\}$. The signature is invalid if $\alpha = \perp$.*

The following correctness condition must hold:

$$\begin{rcases} \forall (pk, sk) \leftarrow \mathsf{Gen}(1^n), \\ \forall m \in \mathcal{M}, \forall r \in \{0, ..., \mathsf{max}\} \end{rcases} : \Pr[\mathsf{Ver}(pk, m, \mathsf{Sign}(sk, m), r) = \perp] = 0$$

Remark 1. The interruption oracle only serves as a virtual party for definitional reasons. In practice, the verification algorithm does not receive the interruption position r as an input, and the algorithm continues to perform computations until it receives an interruption. To model runtime interruptions using the interruption oracle $\mathsf{iOracle}_\Sigma(1^n)$, in this work, we expect the flow of the verification algorithm to *not* be affected/biased by the r value offered by $\mathsf{iOracle}_\Sigma(1^n)$ at the beginning of the verification. Also, we note that depending on signature schemes, there can be more than one way to define the interruption position, r (e.g. clock cycles, number of hash computations, or modular exponentiations).

Extracting function. We assume that for a flexible signature scheme, there exists an efficient function, $\mathsf{iExtract}_\Sigma(\cdot)$, that takes as input the validity of the signature α and outputs the interruption position r. Intuitively, for the case of an unexpected interruption, the verifier needs not not know when the verification algorithm is interrupted. However, based on the validity output α, the verifier should be able to use $\mathsf{iExtract}_\Sigma(\cdot)$ to learn the interruption position, r. The definition of extracting function depends on the specification of the interruption position and signature scheme. We will define our $\mathsf{iExtract}_\Sigma(\cdot)$ for each of our proposed constructions in Section 4 and Section 5.

Security of flexible signature scheme. We present a corresponding definition to the existential unforgeability under adaptive chosen message attack (EUF-CMA) experiment in order to prove the security of our scheme. For a given flexible signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ and $\alpha \in [0, 1]$, the attack experiment is defined as follows:

Experiment $\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n, \alpha)$:

1. The challenger $\mathcal{C}$ runs $\mathsf{Gen}(1^n)$ to obtain (pk, sk) and $\mathsf{iExtract}_\Sigma(\alpha)$ to obtain position r. $\mathcal{C}$ sends (pk, r) to $\mathcal{A}$.
2. Attacker $\mathcal{A}$ queries $\mathcal{C}$ for signatures of its adaptively chosen messages. Let $Q_{\mathcal{A}}^{\mathsf{Sign}(sk,\cdot)}$ $= \{m_i\}_{i\in[q]}$ be the set of all messages that $\mathcal{A}$ queries $\mathcal{C}$ where the i^{th} query is a message $m_i \in \mathcal{M}$. After receiving m_i, $\mathcal{C}$ computes $\sigma_i \leftarrow \mathsf{Sign}(sk, m_i)$, and sends σ_i to $\mathcal{A}$.
3. Eventually, $\mathcal{A}$ outputs a pair $(m^*, \sigma^*) \in \mathcal{M} \times \mathcal{S}$, where message $m^* \notin Q_{\mathcal{A}}^{\mathsf{Sign}(sk,\cdot)}$ and sends the pair to $\mathcal{C}$.
4. $\mathcal{C}$ computes $\alpha^* \leftarrow \mathsf{Ver}(pk, m^*, \sigma^*, r)$. If $(\alpha^* \neq \perp)$ and $(\alpha^* \geq \alpha)$, the experiment returns 1; else, it returns 0.

Definition 2. *For the security parameter n and $\alpha \in [0, 1]$, a flexible signature scheme Σ is (t, ϵ, q) existential unforgeable under adaptive chosen-message attack if for all efficient adversaries $\mathcal{A}$ that run for at most time t and query $\mathsf{Sign}(sk, \cdot)$ at most q times, the success probability is:*

$$\mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{flex}}(n) = \Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n, \alpha) = 1] \leq \epsilon$$

Here, t and ϵ are functions of α and n, and $q = \mathsf{poly}(n)$.

4 Flexible Lamport-Diffie One-time Signature

In this section, we present our concrete construction of the flexible one-time signature scheme. This construction is based on the Lamport-Diffie one time signature construction introduced in [19]. The idea of the Lamport Diffie one-time signature scheme is to sign/verify the message/signature bit-by-bit using a preimage-resistant hash function.

4.1 Construction

We show the concrete construction of the flexible Lamport-Diffie one-time signature in Fig. 2. In this construction, we use the same key generation and signing algorithms from the Lamport-Diffie signature and modify the verification algorithm.

Key Generation Algorithm. The key generation algorithm takes a parameter 1^n as input, and generates a private key by choosing $2n$ bit strings each of length n uniformly at random from $\{0,1\}^n$, namely, $\mathsf{SK} = (sk_i[b])_{i\in[n],b\in\{0,1\}} \in \{0,1\}^{2n^2}$. The public key is obtained by evaluating the preimage-resistant hash function on each of the private key's n bit string, such that $\mathsf{PK} = (pk_i[b])_{i\in[n],b\in\{0,1\}}$ where $pk_i[b] = F(sk_i[b])$ and $F(\cdot)$ is the preimage-resistant hash function.

Signing Algorithm. The signing algorithm takes as input the message m and the private key SK. First, it computes the digest of the message $d = G(m) = (d_i)_{i\in[n]}$ where $d_i \in \{0,1\}$ and $G(\cdot)$ is a collision-resistant hash function that outputs digests of length n. The signature is generated based on the digest d as $\sigma = (sk_i[d_i])_{i\in[n]}$.

Flexible Verification Algorithm. This algorithm takes as input a message m, a public key PK, a signature σ, and an optional interruption position $[\![r]\!]$ and outputs the validity of the signature α. In this construction, we model the interruption condition $r \in \{0,1,\ldots,n\}$, as the number of hash $F(\cdot)$ computations performed during verification. As mentioned earlier in Section 3, to faithfully model the interruption process, the flow of the verification algorithm should not be biased by the r value in any intelligent manner. First, the verification algorithm will query the interruption oracle to determine the interruption position r. The algorithm then computes the digest of the message, $d = G(m) = (d_i)_{i\in[n]}$. Now, instead of sequentially verifying the signature bits like the verification in the standard scheme, the flexible verification algorithm randomly selects a position i of the signature and checks whether $F(\sigma_i[d_i]) = pk_i[d_i]$. If there is one invalid preimage, the verification aborts and returns $\alpha = \perp$. Otherwise, once the interruption condition is met or all positions are verified, the algorithm returns the validity as the fraction of the number of bits that passed the verification check over the length of the signature. In this Lamport-Diffie construction, given the validity α value output by the verification algorithm, the verifier simply computes the interruption position as follows: $\mathsf{iExtract}_{\Sigma_{fots}}(\alpha) = \lfloor \alpha \cdot n \rfloor$

4.2 Security Analysis

In the flexible Lamport-Diffie one-time signature setting, as the verification algorithm does not perform verification at every position of the signature, the adversary can increase the probability of winning by outputting two messages whose hash digests are

Flexible Lamport-Diffie One-time Signature

Given the security parameter n, a preimage resistant hash function $F : \{0,1\}^n \to \{0,1\}^n$, a collision resistant hash function $G : \{0,1\}^* \to \{0,1\}^n$, the flexible Lamport-Diffie one-time signature scheme Σ_{fots} works as follows:

$$
\begin{aligned}
\mathsf{Gen}(1^n) :& \textit{ for each } i \in [n], b \in \{0,1\} : \\
& \quad choose\ sk_i[b] \xleftarrow{\$} \{0,1\}^n,\ set\ pk_j[b] = F(sk_i[b]) \\
& output:\ \mathsf{SK} = (sk_i[b])_{i\in[n],b\in\{0,1\}},\ \mathsf{PK} = (pk_i[b]))_{i\in[n],b\in\{0,1\}} \\
\mathsf{Sign}(\mathsf{SK}, m) :& \textit{compute } d = G(m) = (d_i)_{i\in[n]},\ parse\ \mathsf{SK} = (sk_i[b])_{i\in[n],b\in\{0,1\}}. \\
& output:\ \sigma = (sk_i[d_i])_{i\in[n]} \\
\mathsf{Ver}(\mathsf{PK}, m, \sigma, [\![r]\!]) :& \ if\ r \textit{ is not provided: set } r \leftarrow \mathsf{iOracle}(1^n), \\
& k_F = 0,\ N = [n] \\
& \textit{compute } d = G(m) = (d_i)_{i\in[n]} \\
& \textit{write } \mathsf{PK} = (pk_i[b])_{i\in[n],b\in\{0,1\}},\ \sigma = (\sigma_i)_{i\in[n]} \\
& while\ (r > 0)\ and\ (N \neq \emptyset) : \\
& \quad choose\ i \xleftarrow{\$} N \\
& \quad if\ F(\sigma_i) \neq pk_i(d_i),\ return\ \alpha = \perp \\
& \quad N = N - \{i\}, k_F = k_F + 1, r = r - 1 \\
& output:\ \alpha = k_F/n
\end{aligned}
$$

Fig. 2: Construction of the Flexible Lamport-Diffie Signature

close. This is equivalent to finding an ℓ-near-collision pair where ℓ is determined by the adversary. Theorem 1 offers the trade-off between computation time and success probability for the adversary.

Theorem 1. *Let F be (t_{ow}, ϵ_{ow}) preimage-resistant hash function, G be $(t_{\ell\text{-}ncr}, \epsilon_{\ell\text{-}ncr})$ ℓ-near-collision-resistant hash function, k_F, k_G be the number of times $F(\cdot), G(\cdot)$ evaluated in the verification respectively, d be the Hamming distance between two message digests output by $\mathcal{A}$, and $t_{gen}, t_{sign}, t_{ver}$ be the time it takes to generate keys, sign the message, and verify the signature respectively. With $1 \leq k_F \leq n$, $k_G = 1$, the flexible Lamport-Diffie one-time signature Σ_{fots} is $(t_{fots}, \epsilon_{fots}, 1)$ EUF-CMA where:*

$$
\begin{aligned}
&\alpha = k_F/n \\
&t_{fots} = \min\{t_{ow}, t_{\ell\text{-}ncr}\} - t_{sign} - t_{ver} - t_{gen} \textit{ where } 0 \leq \ell \leq n - k_F \\
&\epsilon_{fots} \leq \min\left\{1, 2\cdot\max\left\{\prod_{i=0}^{k_F-1}(1 - \frac{d}{n-i}), 4n\cdot\epsilon_{ow}\right\}\right\} \textit{ where } 0 \leq d \leq \ell
\end{aligned}
$$

Due to space constraints, the proof of Theorem 1 is shifted to Appendix A.

Security Level. Towards making the security of flexible Lamport-Diffie one-time signatures more comprehensible, we adapt the security level computation from [7]. For any (t, ϵ) signature scheme, we define the security of the scheme to be $\log_2(t/\epsilon)$. As, in the flexible setting, the value of the pair (t, ϵ) may vary as the adversary decides the Hamming distance ℓ, for each value of $k_F \in \{0, \ldots, n\}$, we compute the adversarial advantage for all values $0 \leq \ell \leq n - k_F$ and output the minimum value of $\log_2\left(\frac{t_{fots}}{\epsilon_{fots}}\right)$ as the security level of our scheme. A detailed security level analysis for the Lamport-Diffie one-time signature is available in Section 6.1.

5 Flexible Merkle Tree Signature

We use the Merkle authentication tree [21] to convert the flexible Lamport-Diffie one-time signature scheme into a flexible many-time signature scheme.

5.1 Construction

In the Merkle tree signature scheme, in addition to verifying the validity of the signature, the verifier uses the authentication nodes provided by the signer to check the authenticity of the one-time public key. We are interested to quantify such values under an interruption. To achieve such a requirement, we require the signer to provide additional nodes in the authentication path.

Key Generation Algorithm. Our key generation remains the same as the one proposed in the original Merkle tree signature scheme [21]. For a tree of height h, the generation algorithm generates 2^h Lamport-Diffie one-time key pairs, $(\mathsf{PK_i}, \mathsf{SK_i})_{i\in[2^h]}$. The leaves of the tree are digests of one-time public keys, $H(\mathsf{PK}_i)$, where $H(\cdot)$ is a collision-resistant hash function. An inner node of the Merkle tree is the hash digest of the concatenation of its left and right children. Finally, the public key of the scheme is the root of the tree, and the secret key is the set of 2^h one-time secret keys.

Modified Signing Algorithm. In the original Merkle signature scheme, a signature consists of four parts: the signature state s, a one-time signature σ_s, a one-time public key PK_s and a set of authentication nodes $\mathsf{Auth_s} = (a_i)_{i\in[h]}$. The verifier can use PK_s to verify the validity of the σ_s and use nodes in $\mathsf{Auth_s}$ and state s to efficiently verify the authenticity of $\mathsf{PK_s}$.

For our signing algorithm, along with authentication nodes in the old construction, we require the signer to send the nodes that complete the direct authentication path from the one-time public key to the root. We call this set of nodes complement authentication nodes, $\mathsf{Auth}_s^c = (a'_i)_{i\in[h]}$. The reason for including additional authentication nodes is to allow the verifier to randomly verify any level of the tree. Moreover, with additional authentication nodes, verifier can verify different levels of the tree in parallel. Fig. 3 describes an example of the new requirement for a tree of height three.

The modified signature now consists of five parts: a state s, a Lamport-Diffie one-time signature σ_s, a one-time public key PK_s, a set of authentication nodes Auth_s, and a set of complement authentication nodes Auth_s^c.

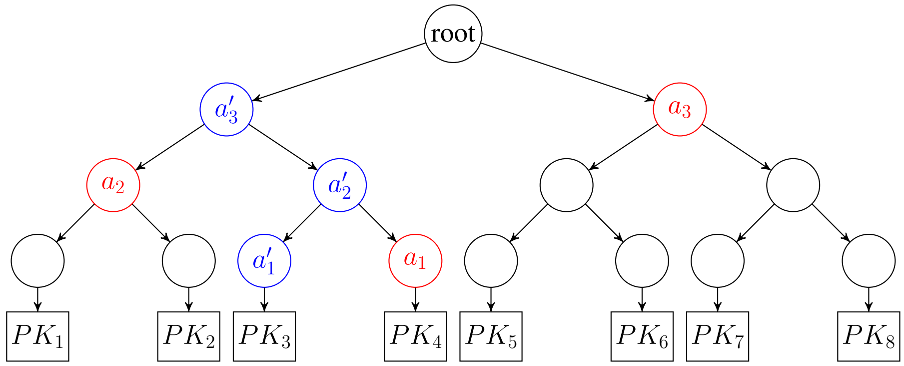

Fig. 3: An example of new authentication nodes for PK_3 where $\mathsf{Auth}_3 = (a_1, a_2, a_3)$ is the set of authentication nodes in the original scheme and $\mathsf{Auth}^c_3 = (a'_1, a'_2, a'_3)$ is the set of additional authentication nodes

Flexible Verification Algorithm. With additional authentication nodes, the verification algorithm can verify the authenticity of the public key at arbitrary levels of the authentication tree as well as use the flexible verification described in Section 4 to partially verify the validity of the one-time signature. In the end, the verification returns $\alpha = (\alpha_v, \alpha_a)$ that contains both the validity of the signature and the authenticity of the public key. In this construction, we define the interruption $r \in \{0, 1, \ldots, n + h + 1\}$, as the number of computations performed during the verification step.

In contrast to the verification performed in the one-time signature scheme, the security guarantee the verifier gains from the authenticity verification of the one-time public key only increases linearly as the number of computations performed on the authentication path increase: The adversary can always generate a new one-time key pair to sign the message that is not a part of one-time key pairs created by the generation algorithm. In the original Merkle scheme, such key pair will fail the authenticity check with overwhelming probability because the verifier can use the authentication nodes to compute and verify the root. However, in the flexible setting, the verifier may not be able to complete the authenticity verification, and there is a non-negligible probability that an invalid one-time public key will be used to verify the validity of the signature. Therefore, the verifier gains exponential security guarantee about the validity of the one-time signature but only a linear guarantee about the authenticity of the public key as the number of computations increases.

To address this issue, the verification algorithm needs to balance the computations performed on the authentication path and the computations performed on the one-time signature. We define the confidence for the validity of the one-time signature as $1 - 1/2^{k_F/2}$ and the confidence for authenticity of the one-time public key as $k_H/(h + 1)$, where k_F is the number of computations performed on the one-time signature, k_H is the number of computations performed on the one-time public key, and h is the height of the Merkle tree. To balance the number of computations, the verifier needs to maintain $1 - 1/2^{k_F/2} \approx k_H/(h + 1)$. With the new signing and verifying algorithms described above, we present a detailed construction of the flexible Merkle signature scheme in

Flexible Merkle Tree Signature Scheme

Given the security parameter n, the height of the tree h, a preimage resistant hash function $F : \{0,1\}^n \rightarrow \{0,1\}^n$, a collision resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$, $G : \{0,1\}^* \rightarrow \{0,1\}^n$, and a flexible Lamport-Diffie one-time signature scheme $\Sigma_{fots} = (\mathsf{Gen}_{fots}, \mathsf{Sign}_{fots}, \mathsf{Ver}_{fots})$. The stateful flexible Merkle scheme Σ_{fms} works as follows:

$\mathsf{Gen}(1^n)$: *generate* 2^h *ots pairs* $\{(\mathsf{PK_i}, \mathsf{SK_i})\}_{i\in[2^h]}$ *using* $\mathsf{Gen}_{fots}(1^n)$

compute the inner nodes of the Merkle tree as follows:

$$node_i[j] = H(node_{i-1}[2j-1]||node_{i-1}[2j])$$
$$2 \leq i \leq h+1, 1 \leq j \leq 2^{h+1-i}$$
$$node_1[i] = H(PK_i), 1 \leq i \leq 2^h$$

output : $\mathsf{SK} = \{\mathsf{SK}_i\}_{i\in[2^h]}, \mathsf{PK} = root\ (i.e.\ node_{h+1}[1]),\ s = 1$

$\mathsf{Sign}(\mathsf{SK}, m, s)$: *compute* $\sigma_s = \mathsf{Sign}_{fots}(\mathsf{SK}_s, m)$

compute $\mathsf{Auth_s} = (a_i)_{i\in[h]},\ where$

$$a_i = \begin{cases} node_i[\lceil s/2^{i-1}\rceil + 1] & if\ \lceil s/2^{i-1}\rceil \equiv 1\ mod\ 2 \\ node_i[\lceil s/2^{i-1}\rceil - 1] & if\ \lceil s/2^{i-1}\rceil \equiv 0\ mod\ 2 \end{cases}$$

compute $\mathsf{Auth_s^c} = (a'_i)_{i\in[h]},\ where\ a'_i = node_i[\lceil s/2^{i-1}\rceil]$

output : $\sigma = (s, \sigma_s, \mathsf{PK_s}, \mathsf{Auth_s}, \mathsf{Auth_s^c}), s = s+1$

$\mathsf{Ver}(\mathsf{PK}, m, \sigma, [\![r]\!])$: if *r is not provided: set* $r \leftarrow \mathsf{iOracle}(1^n)$,

$set\ N = [n], T = [h+1], k_F = 0, k_H = 0$

compute $G(m) = d = (d_i)_{i\in[n]}$

extract $(s, \sigma_{fots}, \mathsf{PK_{fots}}, \mathsf{Auth}, \mathsf{Auth^c}) \leftarrow \sigma$

write $\sigma_{ots} = (\sigma_i)_{i\in[n]},\ \mathsf{PK_{fots}} = (pk_i[b])_{i\in[n], b\in\{0,1\}}$,

$\mathsf{Auth_s} = (a_i)_{i\in[h]},\ \mathsf{Auth_s^c} = (a'_i)_{i\in[h]}$

$while\ r > 0\ and\ H \neq \emptyset\ and\ N \neq \emptyset\ do:$

$if\ 1 - 1/2^{k_F/2} \leq k_H/(h+1):$

$choose\ i \xleftarrow{\$} N,\ if\ F(\sigma_i) \neq pk_i(d_i),\ output:\ \alpha = \perp$

$N = N - \{i\},\ k_F = k_F + 1$

$else:\ choose\ j \xleftarrow{\$} T,\ set\ a'_{h+1} = \mathsf{PK}$

$if\ j = 1: if\ a'_1 \neq H(\mathsf{PK_s}), output:\ \alpha = \perp$

$if\ j > 1: if\ a'_j$ *is not a parent of* a_{j-1} *and* a'_{j-1} :

$output\ \alpha = \perp.$

$T = T - \{j\}, k_H = k_H + 1$

$r = r - 1$

$output:\ \alpha = (k_F/n, k_H/(h+1))$

Fig. 4: The Flexible Merkle Signature Construction

Fig. 4. In this Merkle signature construction, given the validity $\alpha = (\alpha_v, \alpha_a)$ value output by the verification algorithm, the verifier can compute the interruption position as follow: $\mathsf{iExtract}_{\Sigma_{fms}}(\alpha) = \lfloor \alpha_v n \rfloor + \lfloor \alpha_a (h+1) \rfloor$.

5.2 Security Analysis

Theorem 2 presents the trade-off between computation time and success probability for the adversary $\mathcal{A}$.

Theorem 2. *Let F be (t_{ow}, ϵ_{ow}) preimage-resistant hash function, G be $(t_{\ell\text{-}ncr}, \epsilon_{\ell\text{-}ncr})$ ℓ-near-collision-resistant hash function, H be (t_{cr}, ϵ_{cr}) collision-resistant hash function, k_F, k_G, k_H be the number of times $F(\cdot), G(\cdot), H(\cdot)$ performed respectively, d be the smallest Hamming distance between the forged message digest and other queried message digests, and $t_{gen}, t_{sign}, t_{ver}$ be the time it takes to generate keys, sign the message, and verify the signature respectively. With $1 \leq k_F \leq n$, $0 \leq k_H \leq h+1$, and $k_G = 1$, the flexible Merkle signature construction (Σ_{fms}) from flexible Lamport-Diffie one-time signature scheme is $(t_{fms}, \epsilon_{fms}, 2^h)$ EU-CMA, where*

$$\alpha = (k_F/n, k_H/(h+1))$$

$$t_{fms} = \begin{cases} \mathcal{O}(1) & \textit{when } k_H < h+1, \\ \min\left\{t_{ow}, t_{\ell\text{-}ncr}, t_{cr}\right\} - 2^h \cdot t_{sign} - t_{ver} - t_{gen} & \textit{where } 0 \leq \ell \leq n - k_F \end{cases}$$

$$\epsilon_{fms} \leq \min\left\{1, 4 \cdot \max\left\{1 - \frac{k_H}{(h+1)}, 2^h \prod_{i=0}^{k_F-1} (1 - \frac{d}{n-i}), 2^{h+\log_2 4n} \cdot \epsilon_{ow}, \epsilon_{cr}\right\}\right\}$$

$$\textit{where } 0 \leq d \leq \ell$$

Due to space constraints, the proof of Theorem 2 is shifted to Appendix A.

5.3 Other Signature Schemes

Over the last few years, several optimized versions of Merkle tree signature and one-time signature schemes have been proposed. This includes XMSS [6] and SPHINCS [4] for the tree signatures, and HORS [22], BIBA [24], HORST [4] and Winternitz [21] for one-time signatures. While the security analysis for each scheme may vary a little, we can use the same technique described above to transform those schemes into signature schemes with a flexible verification. In this work, we choose to use Lamport-Diffie One-time signature for our construction because of two reasons. First, the number of hash evaluations in Lamport-Diffie Signature verification is fixed for constant size messages, and this gives better and more precise security proofs. Second, Lamport-Diffie one-time signature has better performance in term of the running time. Also, other optimized versions of hash-based signature schemes like Winternitz reduce the size of the public key and the signature with the expense of verification time and generation time. Thus, according to our experiment and analysis, Lamport-Diffie One-time signature combined with Merkle Tree provides a better speed performance and more concrete security proofs.

We also investigate number-theoretic signature schemes and observe that the similar verification technique can be applied to the Fiat-Shamir Signature Scheme [12] as its signature is partitioned into different verifiable sets. However, compared to hash function evaluations, the computation of modular exponentiation is significantly more expensive and thus may not be suitable for flexible security application environments. Moreover, lattice-based signature schemes such as GPV signatures [15] can be an interesting candidate for a flexible signature construction. For GPV signatures, a public key is a matrix output by a trapdoor sampling algorithm, and a signature is output by a pre-image sampling algorithm. The signature verification is performed using matrix and vector multiplication. The same randomized verification technique seems to be applicable here on different rows of the matrix. In the future, we plan to explore a flexible version of GPV signatures.

6 Evaluation, Performance Analysis and Discussion

In this section, we evaluate the performance and the security level of the flexible Lamport-Diffie one-time signature and flexible Merkle signature schemes. For both schemes, the validity value α suggests the number of computations performed (i.e., k_H, k_F) during verification. Based on the value α, the verifier determines the security level achieved by the (interrupted) verification instance.

6.1 Security Level of Flexible Lamport-Diffie One-time Signature

The security level of a flexible Lamport-Diffie signature depends on the actual Hamming distance between two message digests output by the adversary, and it can increase its advantage by putting more time to find a near-collision pair. However, it is unclear how to precisely measure an exact Hamming distance between those two digests. Therefore, we outline some possible assumptions in order to estimate precisely the value of $\Delta(G(m), G(m^*))$. Using the generic attack on finding near collision pair [18], we can assume that an adversary $\mathcal{A}$ who uses a generic birthday attack can always output a pair (m, m^*) such that $\Delta(G(m), G(m^*)) \leq \ell$ after spending $t_{\ell\text{-}ncr} = 2^{n/2}/\sqrt{\sum_{i=0}^{\ell}\binom{n}{i}}$. Second, for a fixed value ℓ, if the adversary finds a pair (m, m^*) such that $\Delta(G(m), G(m^*)) \leq \ell$, we let $d = \Delta(G(m), G(m^*))$ is equal to the expected value of $\Delta(G(m), G(m^*))$. The intuition behind the second assumption is that as we let the Hamming distance d decrease by 1, the probability that $\Delta(G(m), G(m^*)) = d$ decreases by factor of n; therefore, the actual value of d should be closer to ℓ than to 0.

We define the set $B_\ell(G(m)) = \{x \mid x \in \{0,1\}^n \wedge \Delta(x, G(m)) \leq \ell\}$. If $G(m)$ and $G(m^*)$ is a ℓ-near-collision pair, then $G(m^*) \in B_\ell(G(m))$. If $G(\cdot)$ behaves as an uniformly random function, then given ℓ, the expected value of $\Delta(G(m), G(m^*))$ is:

$$\mathbb{E}(\Delta(G(m), G(m^*))) = \sum_{j=0}^{\ell} j \cdot \frac{\binom{n}{j}}{|B_\ell(G(m))|} = \sum_{j=0}^{\ell} j \cdot \frac{\binom{n}{j}}{\sum_{i=0}^{\ell}\binom{n}{i}} \qquad (1)$$

For the case of Lamport-Diffie one-time signature, we have $t_{gen} = 2n, t_{sign} = t_{ver} = n$. Combining Theorem 1 and equation 1, we have:

$$t_{fots} = \max\left\{1, \frac{2^{n/2}}{\sqrt{\sum_{i=0}^{\ell}\binom{n}{i}}} - 4 \cdot n\right\} \text{ for } \ell \leq n - k_F$$

$$\epsilon_{fots} \leq \min\left\{1, 2 \cdot \prod_{i=0}^{k_F-1}(1 - \frac{d}{n-i})\right\} \text{ where } d = \mathbb{E}(\Delta(G(m), G(m^*)),$$
$$\text{given } \Delta(G(m), G(m^*)) \leq \ell$$

Finally, the adversary's advantage varies depending on the value of ℓ. Therefore, for a fixed value k_F, we compute the adversarial advantage all values $\ell \leq n - k_F$ and output the minimum value of $\log_2\left(\frac{t_{fots}}{\epsilon_{fots}}\right)$ as the security level of the scheme.

Fig. 5 gives the trade-off between the number of computations and the security level of the flexible Lamport-Diffie scheme. Compared to the original Lamport-Diffie scheme, our construction offers a reasonable security level despite a smaller number of computations. For example, while a complete verification requires 256 evaluations of $F(\cdot)$ to achieve the 128-bit security level, with only 128 evaluations of $F(\cdot)$, the scheme still offers around the 92-bit security level.

6.2 Security Level of Flexible Merkle Tree Signature

For the Merkle tree signature scheme, using the results from [9], [27], we have $t_{gen} = 2^h \cdot 2n + 2^{h+1} - 1, t_{ver} = n + h + 1, t_{sign} = (h+1) \cdot n$. There are two cases for the Merkle tree signature: the authenticity check is complete, $k_H = h+1$ or the authenticity check is not complete, $k_H < h + 1$.

When $k_H < h+1$, the adversary's probability of winning is non-negligible, and the time it needs to spend on the attack is constant; therefore, when the authenticity check is not complete, we simply let: $t_{fms} = 1$, $\epsilon_{fms} = 1 - k_H/(h+1)$. When the authenticity verification is complete, $k_H = h + 1$, using the equation described in Theorem 2, we obtain the following parameters for the flexible Merkle tree scheme:

$$t_{fms} = \max\left\{1, t_{\ell\text{-}ncr} - 2^{h+\log_2(h+1)n} - 2^{h\cdot\log_2 2n} - 2^{\log_2(n-h-1)}\right\} \text{ for } \ell \leq n - k_F$$

$$\epsilon_{fms} \leq \min\left\{1, 2^h \cdot \prod_{i=0}^{k_F-1}(1 - \frac{d}{n-i})\right\} \text{ where } d = \mathbb{E}(\Delta(G(m), G(m^*)))$$

Using those formulas, we compute the security level of the flexible Merkle signature as $\log_2(\frac{t_{fms}}{\epsilon_{fms}})$. Fig. 6 shows the trade-off between the security level of the scheme and the number of computations of the flexible Merkle tree signature with $h = 20$. Notice that, for small number of computations, the security level of Merkle tree construction does not increase. The reason is that if the authenticity of the public key is not completely checked, the probability that the adversary wins the forgery experiment is always the fraction of the number of computations on the authentication path over the

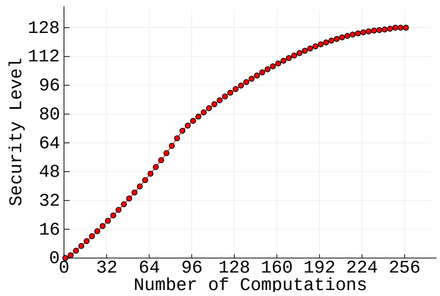

Fig. 5: Security Level of Flexible Lamport-Diffie One-time Signature

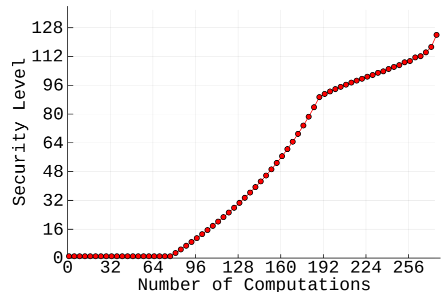

Fig. 6: Security Level of Flexible Merkle Tree Signature

height of the tree, and the forging time remains constant. Moreover, for a tree of height h, there are 2^h instances of flexible Lamport-Diffie one-time signature. Therefore, if $F(\cdot)$ evaluated only for a small number of times, the cost of finding an ℓ-near-collision pair (for $\ell \leq n - k_F$) is cheap. The probability that such a pair passes the one-time verification step in one instance of 2^h instances of flexible Lamport-Diffie one-time signature is high. This leads to an undesirable security level during the first few computations.

6.3 Implementation and Performance

We have implemented prototypes of our proposed constructions in C, using the SHA-256 implementation of OpenSSL[1]. We evaluated the performance of our proposed constructions on a Raspberry Pi 3, Model B equipped with 1GB RAM.

Table 1 gives the performance and security levels of the flexible verification algorithm of both schemes compared to other standard signature schemes (i.e., RSA, DSA, ECDSA, and EdDSA) based on the percentage of computations $p = 20\%, 40\%, 60\%,$ 80%, and 100% for messages of size 256.[2] For other signature schemes, we obtain the performance of those schemes using the OpenSSL library. More specifically, for ECDSA, we used two standard curves: Ed25519 and nistp256. For the RSA signature scheme, we used the smallest recommended public key $2^{16} + 1$ for the verification algorithm. For the security levels of other signature schemes, we use the information from [1, 6]. As shown in Table 1, the performance of both flexible signature schemes is comparable to other standard schemes in term of the verification running time. More importantly, both constructions offer an increasing security level at each step of the algorithm while other signature schemes can only provide such information at the end of the verification algorithm, and Table 1 demonstrates that in the form of (Timings, Security Level) pairs. Also, notice that as the number of verification computations increases, the Lamport-Diffie OTS gives higher security level than the signing shorter hash digest

[1] https://www.openssl.org/

[2] We focus on the verification algorithm in this work, and for the performance of signing and generation algorithms we refer readers to [6, 7].

	Signature Verification; Output Format: (**Timings, Security Level**)				
Percentage of Computations	20%	40%	60%	80%	100%
RSA 3072, $pk = 2^{16} + 1$	-	-	-	-	(1.43ms, 128)
DSA 2048	-	-	-	-	(4.93ms, 87)
EdDSA (Ed25519 curve)	-	-	-	-	(3.21ms, 128)
ECDSA (nistp256 curve)	-	-	-	-	(3.39ms, 128)
Lamport-Diffie OTS verification, $n = 256$	(0.16ms, 35)	(0.31ms, 79)	(0.43ms, 105)	(0.47ms, 121)	(0.54ms, 127)
Merkle signature verification, $n = 256, h = 20$	(0.85ms, 1)	(0.93ms, 19)	(1.00ms, 61)	(1.06ms, 99)	(1.23ms, 127)

Table 1: Comparing flexible signature schemes performance for different levels of signature verification with other signature schemes

approach which offers the security level that is equal to half of the length of the hash digest. The main reason is that the verification algorithm verifies the signature at random locations, and while the adversary may learn about the number of computations performed, the adversary does not know which indices of the signature get verified. Thus, the adversary has to decide how close the two digests should be to maximize his adversarial advantage. For the case of Merkle tree signature scheme, we do not see a huge improvement in the performance of the verification despite smaller number of computations. This is because the computation of $H(\mathsf{PK}_{\mathsf{fots}})$ is costly, bebause the use the Merkle-Damgård transformation in SHA2 hash family, as it requires n calls to the compression function for an one time public key of size $2n^2$.

7 Conclusion

In this paper, we defined the concept of a signature scheme with a flexible verification algorithm. We presented two concrete constructions based on the Lamport-Diffie one-time signature scheme and the Merkle signature scheme and formally proved their security. We also implemented prototypes of our proposed construction and showed that the running time performance of our proposed designs is comparable to other signature schemes in a resource-constrained environment. More importantly, compared to standard signature schemes with deterministic verification, our schemes allow the verifier to put different constraints on the verification algorithm in a spontaneous manner and still guarantee a reasonable security level. Our proposed signature scheme is one of the few cryptographic primitives that offers a trade-off between security and resources. It can be highly useful for cryptographic mechanisms in unpredictably resource constrained environments such as real-time systems and the IoT.

In the long run, significant research will be required in this challenging flexible security area. We plan to explore similar ideas for confidentiality in (symmetric or asymmetric) encryptions, integrity with MACs, and possibly beyond. We believe these cryptographic protocols will make the security mechanisms more prevalent in the real-time systems and IoT in general.

Acknowledgment. We thank Dominique Schröder and Mikhail Atallah and the anonymous reviewers for encouraging discussions and suggestions.

References

1. Barker, E.: Recommended for key management-part 1: General, `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf`
2. Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In: Advances in Cryptology — EUROCRYPT'98. pp. 236–250. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
3. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing. In: Advances in Cryptology — CRYPTO '94. pp. 216–233 (1994)
4. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: Sphincs: Practical stateless hash-based signatures. In: Advances in Cryptology – EUROCRYPT 2015. pp. 368–397 (2015)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptology **17**(4), 297–319 (Sep 2004)
6. Buchmann, J., Dahmen, E., Hülsing, A.: Xmss - a practical forward secure signature scheme based on minimal security assumptions. In: Post-Quantum Cryptography. pp. 117–129 (2011)
7. Buchmann, J., Dahmen, E., Szydlo, M.: Hash-based digital signature schemes. In: Post-Quantum Cryptography. pp. 35–93 (2009)
8. Camenisch, J., Hohenberger, S., Pedersen, M.Ø.: Batch verification of short signatures. In: Naor, M. (ed.) Advances in Cryptology - EUROCRYPT 2007. pp. 246–263. Berlin, Heidelberg (2007)
9. Dahmen, E., Okeya, K., Takagi, T., Vuillaume, C.: Digital signatures out of second-preimage resistant hash functions. In: Post-Quantum Cryptography: Second International Workshop, PQCrypto 2008. pp. 109–123 (2008)
10. Denning, T., Kohno, T., Levy, H.M.: Computer security and the modern home. Commun. ACM (1), 94–103 (Jan 2013)
11. Fan, X., Garay, J.A., Mohassel, P.: Short and adjustable signatures. IACR Cryptology ePrint Archive **2016**, 549 (2016)
12. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: CRYPTO' 86. pp. 186–194 (1987)
13. Fischlin, M.: Progressive verification: The case of message authentication. In: Progress in Cryptology - INDOCRYPT 2003. pp. 416–429. Springer Berlin Heidelberg, Berlin, Heidelberg (2003)
14. Freitag, C., Goyal, R., Hohenberger, S., Koppula, V., Lee, E., Okamoto, T., Tran, J., Waters, B.: Signature schemes with randomized verification. In: Applied Cryptography and Network Security. pp. 373–389 (2017)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1374376.1374407, `http://doi.acm.org/10.1145/1374376.1374407`
16. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, chap. 12, pp. 442–443 (2007)
17. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: 31st IEEE Symposium on Security and Privacy, S&P 2010. pp. 447–462 (2010)

18. Lamberger, M., Teufl, E.: Memoryless near-collisions, revisited. CoRR (2012)
19. Lamport, L.: Constructing digital signatures from a one way function. SRI intl. CSL-98 (1979)
20. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. 1st edn. (1996)
21. Merkle, R.C.: A certified digital signature. In: CRYPTO' 89 Proceedings (1990)
22. Perrig, A.: The biba one-time signature and broadcast authentication protocol. In: Proceedings of the 8th ACM Conference on Computer and Communications Security. pp. 28–37. CCS '01 (2001)
23. Petit, J., Stottelaar, B., Feiri, M., Kargl, F.: Remote attacks on automated vehicles sensors: Experiments on camera and liDAR. Black Hat Europe **11** (2015)
24. Reyzin, L., Reyzin, N.: Better than biba: Short one-time signatures with fast signing and verifying. In: Proceedings of the 7th Australian Conference on Information Security and Privacy. pp. 144–153. ACISP '02 (2002)
25. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Fast Software Encryption. pp. 371–388 (2004)
26. Sadeghi, A.R., Wachsmann, C., Waidner, M.: Security and privacy challenges in industrial internet of things. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). pp. 1–6 (June 2015)
27. Szydlo, M.: Merkle tree traversal in log space and time. In: Cachin, C., Camenisch, J.L. (eds.) Advances in Cryptology - EUROCRYPT 2004. pp. 541–554. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
28. Yu, T., Sekar, V., Seshan, S., Agarwal, Y., Xu, C.: Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In: Proceedings of the 14th ACM Workshop on Hot Topics in Networks. pp. 5:1–5:7 (2015)

A Proofs

In this section, we provided the formal proofs of two stated theorems.

A.1 Proof of theorem 1

Proof. Let m be the message asked by $\mathcal{A}$ during the experiment $\mathsf{FlexExp}_{\Sigma,\mathcal{A}}(1^n, \alpha)$, and (m^*, σ^*) be the forgery pair. We define the distance, $d = \Delta(G(m), G(m^*))$.

We notice that for a pair (m, m^*) output by the adversary during the forgery experiment, if $\Delta(G(m), G(m^*)) > n - k_F$, then by pigeonhole principle, at least one of different positions will be checked. Therefore, in order to maximize the success probability, the adversary has to choose ℓ and find a ℓ-near-collision pair where the Hamming distance of $G(m)$ and $G(m^*)$ is less than ℓ where $\ell \leq (n - k_F)$. In order to output such near-collision pair, $\mathcal{A}$ requires at least $t = t_{\ell\text{-}ncr} = 2^{n/2} / \sqrt{\sum_{i=0}^{\ell} \binom{n}{i}}$. Also, on the other hand, $\mathcal{A}$ may win the forgery experiment by spending t_{ow} to break the underlying preimage resistant hash function. Thus, subtracting the running time of generating, signing, and verifying algorithms, we have:

$$t_{fots} = \min\{t_{ow}, t_{\ell\text{-}ncr}\} - t_{sign} - t_{gen} - t_{ver} \text{ where } 0 \leq \ell \leq n - k_F \qquad (2)$$

For the success probability, we let Miss be the event that no different bit gets verified. Since d is the Hamming distance between 2 message digests, either none of those different positions were checked, or some of those positions passed the check (i.e. the preimage was found). Thus, we rewrite $\mathcal{A}$'s advantage for the forging experiment as follows: $\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1] \leq \Pr[\mathsf{Miss}] + \Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \overline{\mathsf{Miss}}]$

The event $(\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \overline{\mathsf{Miss}})$ implies that $\mathcal{A}$ wins the forgery experiment by providing a preimage of $F(\cdot)$. Therefore, we can use $\mathcal{A}$ to construct a preimage finder $\mathcal{B}$. The reduction is presented in [7]. One can show:

$$\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \overline{\mathsf{Miss}}] \leq 4n \cdot \mathsf{Adv}^{\mathsf{pre}}_{\mathcal{B},\mathsf{F}}(n) \tag{3}$$

Finally, $\Pr[\mathsf{Miss}]$ implies the adversary can win the forging experiment if the challenger does not perform verification on the different bits. Since d is the number of different bits between two digests, the probability that the challenger does not perform verification on those positions is:

$$\Pr[\mathsf{Miss}] = \prod_{i=0}^{k_F-1} \frac{n-d-i}{n-i} = \prod_{i=0}^{k_F-1} \left(1 - \frac{d}{n-i}\right) \tag{4}$$

From (3) and (4), we have:

$$\begin{aligned}\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1] &\leq \prod_{i=0}^{k_F-1} \frac{n-d-i}{n-i} + 4n \cdot \mathsf{Adv}^{\mathsf{pre}}_{\mathcal{B},\mathsf{F}}(n) \\ &\leq \min\left\{1, 2\cdot \max\left\{\prod_{i=0}^{k_F-1}\left(1-\frac{d}{n-i}\right), 4n\cdot\epsilon_{ow}\right\}\right\}\end{aligned}$$

which completes the proof. ■

A.2 Proof of theorem 2

Proof. Intuitively, if adversary $\mathcal{A}$ provides an invalid one-time public key, the verification must fail for at least one level of tree. Otherwise, $\mathcal{A}$ successfully finds a collision of H. However, in our scheme, since every level of the tree may not be verified, there is a possibility that the forged level is not checked. We formalize the intuition as following; we let $\mathsf{InvalidOPK}$ be the event that $\mathcal{A}$ provides an invalid one-time public key. Consider the Merkle tree construction based on the one-time signature construction.

$$\begin{aligned}\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1] = &\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \mathsf{InvalidPK}] \\ &+ \Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \overline{\mathsf{InvalidPK}}]\end{aligned} \tag{5}$$

The $\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \mathsf{InvalidPK}$ implies that $\mathcal{A}$ provided an invalid one-time public key but won the forgery experiment. Thus, either the verifier failed to check a "bad" level of the tree or $\mathcal{A}$ found a collision of $H(\cdot)$. For a tree of height h, there are

$h+1$ levels that one needs to verify for the complete authentication. Since k_H is the number of times $H(\cdot)$ is evaluated, using a union bound, we have:

$$\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \wedge \mathsf{InvalidPK}] \leq 2\cdot\max\left\{1-\frac{k_H}{h+1},\epsilon_{cr}\right\} \tag{6}$$

If $\mathcal{A}$ found a collision of $H(\cdot)$, then we can construct a collision finder [7].

The event $\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha) = 1 \wedge \overline{\mathsf{InvalidPK}}$ implies that $\mathcal{A}$ won the flexible forgery experiment for one-time signature scheme. Since we defined k_F to be the number of $F(\cdot)$ evaluated, the underlying flexible one-time signature is $(t_{fots},\epsilon_{fots},1)$. Therefore, using Theorem 1, we get:

$$\epsilon_{fots} \leq 2\cdot\max\left\{\prod_{i=0}^{k_F-1}(1-\frac{d}{n-i}),4n\cdot\epsilon_{ow}\right\} \text{ where } 0\leq d\leq \ell\leq n-k_F \tag{7}$$

Since there are 2^h instances of the flexible Lamport-Diffie one-time signature, it means that $\mathcal{A}$ wins the forgery game with probability:

$$\begin{aligned}\Pr[\mathsf{FlexExp}_{\mathcal{A},\Sigma}(1^n,\alpha)=1 \ \wedge \ \overline{\mathsf{InvalidPK}}]\\ &\leq 2\cdot\max\left\{2^h\cdot\prod_{i=0}^{k_F-1}(1-\frac{d}{n-i}),2^{h+\log_2 4n}\cdot\epsilon_{ow}\right\}\\ &\text{where } 0\leq d\leq\ell\leq n-k_F\end{aligned} \tag{8}$$

From 5 and 7, we have:

$$\epsilon_{fms} \leq 4\cdot\max\left\{1-k_H/(h+1),2^h\cdot\prod_{i=0}^{k_F-1}(1-\frac{d}{n-i}),2^{h+\log_2 4n}\cdot\epsilon_{ow},\epsilon_{cr}\right\} \\ \text{where } 0\leq d\leq\ell\leq n-k_F \tag{9}$$

When $k_H < h+1$ we simply let $t_{fms}=\mathcal{O}(1)$ because $\mathcal{A}$ will win the forgery experiment with probability $1-k_H/(h+1)$. Consider the case when $k_H = h+1$, we have:

$$\epsilon_{fms} \leq 4\cdot\max\left\{2^h\cdot\prod_{i=0}^{k_F-1}(1-\frac{d}{n-i}),2^{h+\log_2 4n}\cdot\epsilon_{ow},\epsilon_{cr}\right\} \text{ where } 0\leq d\leq\ell\leq n-k_F$$

and using [7, Theorem 5], we have $t_{fms}=\min\{t_{cr},t_{fots}\}-2^h\cdot t_{sign}-t_{ver}-t_{gen}$. Now, using Theorem 1, we get:

$$t_{fms}=\min\{t_{ow},t_{\ell\text{-}ncr},t_{cr}\}-2^h\cdot t_{sign}-t_{ver}-t_{gen} \text{ where } 0\leq\ell\leq n-k \tag{10}$$

This completes the proof. ■