A New Approach to Constructing Digital Signature Schemes (Extended Paper)

Ahto Buldas[1], Denis Firsov[1,2], Risto Laanoja[1,2],
Henri Lakk[2], and Ahto Truu[1,2,✉] *

[1] Tallinn University of Technology, Akadeemia tee 15a, 12618 Tallinn, Estonia
[2] Guardtime AS, A. H. Tammsaare tee 60, 11316 Tallinn, Estonia
ahto.truu@guardtime.com

Abstract. A new hash-based, server-supported digital signature scheme was proposed recently in [13]. We decompose the concept into *forward-resistant tags* and a generic *cryptographic time-stamping* service. Based on the decomposition, we propose more tag constructions which allow efficient digital signature schemes with interesting properties to be built. In particular, the new schemes are more suitable for use in personal signing devices, such as smart cards, which are used infrequently. We define the forward-resistant tags formally and prove that (1) the discussed constructs are indeed tags and (2) combining such tags with time-stamping services gives us signature schemes.

1 Introduction

Recently, Buldas, Laanoja, and Truu [13] proposed a new type of digital signature scheme (which we will refer to as the *BLT scheme* in the following) based on the idea of combining one-time time-bound keys with a time-stamping service. The scheme is post-quantum secure against known attacks and the integrity of the signed messages does not depend on the long-term secrecy of any keys. Existential unforgeability against adaptive chosen-message attack (EUF-ACM) was proven in the random oracle (RO) model.

A limitation of the BLT scheme is the fact that keys are pre-generated and have to be used at their designated time-slots only. On practical parameters the number of keys is rather large, which would make key generation on resource-constrained platforms prohibitively slow.

BLT scheme prevents other parties from misusing keys by making each key expire immediately after a legitimate use. First, each key is explicitly bound to a time slot at the key-generation time, and keys would automatically expire when their designated time-slots passed. Second, the legitimate use of a key is proven by time-stamping the message-key pair. Back-dating a new pair (a new message

* This research was supported by the European Regional Development Fund through the Estonian smart specialization program NUTIKAS and by the research measure of the Estonian IT Academy program. This is an extended version of the short paper available at Springer via https://doi.org/10.1007/978-3-030-26834-3_21

with an already used key) would allow a signature to be forged. Therefore, the hash-then-publish time-stamping [28, 18] and the so-called keyless signatures [11] are particularly suitable for the scheme. The aim of keyless time-stamping is to avoid key-based cryptography and trusted third parties so that time-stamps become irrefutable proofs of time.

Based on this observation, we generalize and decompose the scheme into two functional components: *forward-resistant tags* and a *cryptographic time-stamping* service. As the forward-resistant tag is a novel construct, we define it formally (Sec. 3.1) and prove that the BLT scheme is indeed an instance of forward-resistant tag-based schemes (Sec. 3.4).

Thanks to the formal definition, it is possible to propose other forward-resistant tag systems and prove their security. Getting rid of pre-assigned time-slot allows to achieve greater efficiency: keys could be spent in sequence. A tag system inspired by the Lamport hash-based signature scheme [22] is presented in Sec. 3.5 and another one inspired by the Winternitz optimization [34] in Sec. 3.6. As discussed in Sec. 5, the resulting new signature schemes are efficient and have some interesting properties.

Because the server-assisted signature schemes have not enjoyed their deserved interest, we will start with an overview of related work in Sec. 2 and give some justification why we believe in their usefulness.

2 Related Work and Background

2.1 Hash-Based Signatures

The earliest signature scheme constructed from hash functions is due to Lamport [22]. His scheme, as well as the refinements proposed in [33, 25, 23, 7, 29], are *one-time*: they require generation of a new key pair and distribution of a new public key for each message to be signed.

Merkle [33] introduced the concept of *hash tree* which aggregates a large number of inputs into a single root hash value so that any of the N inputs can be linked to it with a proof consisting of $\log_2 N$ hash values. This allowed combining N instances of a one-time signature scheme into an *N-time* scheme. This approach has been further studied in [18, 20, 39, 21]. A common drawback of these constructs is that the whole tree has to be built at once.

Merkle [34] proposed a method to grow the tree gradually as needed. However, to authenticate the lower nodes of the tree, a chain of one-time signatures (rather than a sequence of sibling hash values) is needed, unless the scheme is used in an interactive environment and the recipient keeps the public keys already delivered as part of earlier signatures. This multi-level approach has subsequently been refined in [32, 6, 9, 8, 30, 31].

A complication with the N-time schemes is that they are *stateful*: as each of the one-time keys may be used only once, the signer has to keep track of them. If this information is lost (for example, when a previous state is restored from a backup or when multiple concurrent processes use the same key), key re-use

may result in a catastrophic security loss. Perrig [35] proposed a *few-time* scheme where a private key can be used to sign several messages, and the security level decreases gradually with each additional use.

Bernstein *et al.* [4] combined the optimized few-time scheme of [38] with the multi-level tree of [8] to create SPHINCS, a *stateless* scheme that uses keys based on a pseudo-random schedule, making the risk of re-use negligible even without tracking the state.

2.2 Server-Assisted Signatures

In server-assisted schemes the signer has to co-operate with a server to produce a signature. The two main motivations for such schemes are: (a) performance: costly computations can be offloaded from an underpowered signing device (such as a smart card) to a more capable server; and (b) security: risks of key misuse can be reduced by either keeping the keys in a server environment (which can presumably be managed better than an end-user's personal computer) or by having the server perform additional checks as part of the signature generation protocol.

An obvious solution is to let the server handle all asymmetric-key operations based on requests from the signers [37]. In this case the server has to be completely trusted, but it's not clear whether that is in fact less secure than letting end-users manage their own keys [17].

To reduce the need to trust the server, Asokan *et al.* [2] proposed and others in [5, 27] improved methods where asymmetric-key operations are performed by a server, but a user can prove the server's misbehavior when presented with a signature that the server created without the user's request. However, such signatures appear to be valid to a verifier until challenged by the user. Thus, these protocols are usable in contexts where a dispute resolution process exists, but unsuitable for applications with immediate and irrevocable effects, such as authentication for access control purposes or committing transactions to append-only ledgers.

Several methods have been proposed for outsourcing the more expensive computation steps of specific signature algorithms, notably RSA, but most early schemes have subsequently been shown to be insecure. In recent years, probably due to increasing computational power of handheld devices and wider availability of hardware-accelerated implementations, attention has shifted to splitting keys between end-user devices and back-end servers to improve the security of the private keys [19, 10].

2.3 Interactive Signature Protocols

Interactive signature protocols, either by interaction between parties or with an external time-stamping service, were considered by Anderson *et al.* [1]. They proposed the "Guy Fawkes Protocol", where, once bootstrapped, a message is preceded by publishing the hash of the message and each message is authenticated by accompanying it with a secret whose hash was published together with

an earlier message. Although the verification is limited to a single party, the protocol is shown to be a signature scheme according to several definitions. The broadcast commitment step is critical for providing non-repudiation of origin. Similar concept was used in the TESLA protocol [36], designed to authenticate parties who are constantly communicating with each other. Due to this, it has the same inflexibility of not supporting multiple independent verifiers.

Buldas *et al.* [13] presented a generic hash-based signature scheme which depends on interaction with a time-stamping service. The principal idea of the scheme is to have the signer commit to a sequence of secret keys so that each key is assigned a time slot when it can be used to sign messages and will transition from signing key to verification key at the end of the time slot. In order to prove timely usage of the keys, a digital time-stamping service is used. Signing then comprises of time-stamping the message-key commitment in order to prove that the signing operation was performed at the correct time.

2.4 Non-Repudiation

An important property of digital signatures (as an alternative to hand-written ones) [24] is non-repudiation, i.e. the possibility to use the signature as evidence against the signer. Solutions where trusted third parties are (technically) able to sign on behalf of their clients are not desirable for non-repudiation, because clients may use that argument to fraudulently call their signatures into question. Therefore, solutions where clients have personal signature devices are preferable to those relying entirely on trusted parties.

Another real-world complexity is key revocation. Without such capability clients may (fraudulently) claim that their private keys were stolen and someone else may have created signatures in their name. With revocation tracking, signatures created before a key revocation event can be treated as valid, whereas signatures created afterwards can be considered invalid. Usually this is implemented using cryptographic time-stamping and certificate status distribution services. No matter the implementation details, this can not be done without online services, which means that most practical deployments of digital signatures are actually server-supported.

2.5 Cryptographic Time-Stamping

Cryptographic time-stamps prove that data existed before a particular time. The proof can be a statement that the data hash existed at a given time, cryptographically signed by a trusted third party. Such statements are useful for data archiving, supporting non-repudiable digital signatures, etc.

Haber and Stornetta [28] made the first steps towards trustless time-stamping by proposing a scheme where each time-stamp would include some information from the immediately preceding one and a reference to the immediately succeeding one. Benaloh and de Mare [3] proposed to increase the efficiency of hash-linked time-stamping by operating in rounds, where messages to be time-stamped within one round would be combined into a hierarchical structure from

which a compact proof of participation could be extracted for each message. The aggregation structures would then be linked into a linear chain. Buldas *et al.* [15, 14, 16] proposed a series of time-stamping schemes based on binary linking that allowed any two tokens to be ordered in time, even if they were issued within the same aggregation round.

The security of linking-based hash-then-publish schemes has been proven in a very strong model where even the time-stamping service provider does not have to be trusted [18, 12], making them particularly suitable for our use-case. It is possible to provide such service efficiently and in global scale [11].

3 Forward-Resistant Tags

3.1 Definitions

Definition 1 (Tag system). *By a tag system we mean a triple* $(\mathrm{Gen}, \mathrm{Tag}, \mathrm{Ver})$ *of algorithms, where:*

- Gen *is a probabilistic key-generation algorithm that, given as input the tag range* T*, produces a secret key* sk *and a public key* pk*.*
- Tag *is a tag-generation algorithm that, given as input the secret key* sk *and an integer* $t \in \{1, \ldots, T\}$*, produces a tag* $\tau \leftarrow \mathrm{Tag}(\mathsf{sk}, t)$*.*
- Ver *is a verification algorithm that, given as input a tag* τ*, an integer* t*, and the public key* pk*, returns either* 0 *or* 1*, such that*

$$\mathrm{Ver}(\mathrm{Tag}(\mathsf{sk}, t), t, \mathsf{pk}) = 1 \ ,$$

whenever $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathrm{Gen}(T)$ *and* $1 \leq t \leq T$*.*

The above definition of a tag system is somewhat similar to that of a signature scheme consisting of procedures for key generation, signature generation, and signature verification [26]. The fundamental difference is that a signature binds the use of the secret key to a message, while a tag binds the use of the secret key to a time.

Definition 2 (Forward-resistant tag system). *A tag system* $(\mathrm{Gen}, \mathrm{Tag}, \mathrm{Ver})$ *is* S*-forward-resistant if every tag-forging adversary* A *using computational resources* ρ *has success probability*

$$\mathsf{Pr}\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(T), (\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk}, \cdot)}(\mathsf{pk}) : \mathrm{Ver}(\tau, t, \mathsf{pk}) = 1\Big] < \frac{\rho}{S} \ ,$$

where A *makes one oracle call* $\mathrm{Tag}(\mathsf{sk}, t')$ *with* $1 \leq t' < t$*.*

The restriction for A to make just one oracle call stems from the fact that the very purpose of a tag system is to bind the use of the secret key to a specific time.

Informally, in order to implement a forward resistant tag system, we have to bind each tag to a time t so that the tag can't be re-bound to a later time. This notion could be seen as dual to time-stamping that prevents back-dating.

In the BLT scheme, the binding is very simple: the time is the sequence number of the key, given as the value of the sibling leaf in the hash tree. Alternatively, the time could be encoded as the shape of the authentication hash chain.

The resources represented by ρ are computation time and memory. The total resource budget of the adversary is $\rho = \alpha \cdot \text{time} + \beta \cdot \text{memory}$, where α and β are the costs of a unit of computation time and a unit of memory, respectively.

Security proofs of the proposed tag systems will be based on the following definitions of basic cryptographic properties of functions:

Definition 3 (One-way function). *A function $f : D \to R$ is S-secure one-way (S-OW in short) if every f-inverting adversary A using computational resources ρ has success probability*

$$\mathsf{Pr}\Big[x \leftarrow D, x' \leftarrow A^{f(\cdot)}(f(x)) : f(x') = f(x)\Big] < \frac{\rho}{S} .$$

Definition 4 (Collision resistant function). *A function $f : D \to R$ is S-secure collision resistant (S-CR) if for every collision-finding adversary A using computational resources ρ:*

$$\mathsf{Pr}\Big[x_1, x_2 \leftarrow A^{f(\cdot)} : x_1 \neq x_2,\ f(x_1) = f(x_2)\Big] < \frac{\rho}{S} .$$

Definition 5 (Undetectable function). *A function $f : D \to D$ is S-secure undetectable (S-UD) if for every detecting adversary A using computational resources ρ:*

$$\Big|\mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(x) = 1\Big] - \mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(f(x)) = 1\Big]\Big| < \frac{\rho}{S} ,$$

where $\mathcal{U}$ generates random values uniformly from D.

Lemma 1. *If $f : D \to D$ is S-UD, then f^n is $\frac{S}{n}$-UD.*

Proof. We prove the claim by reduction. We assume there is an f^n-detecting adversary A and construct an f-detecting adversary B based on oracle access to A. We construct B to process the input y by uniformly randomly picking $i \leftarrow \{0, \ldots, n-1\}$ and returning $A(f^i(y))$. Then B's success probability is

$$\begin{aligned}
\delta' &= \Big|\mathsf{Pr}\Big[x \leftarrow \mathcal{U} : B(x) = 1\Big] - \mathsf{Pr}\Big[x \leftarrow \mathcal{U} : B(f(x)) = 1\Big]\Big| \\
&= \left|\frac{1}{n}\sum_{0 \le i < n} \mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(f^i(x)) = 1\Big] - \frac{1}{n}\sum_{0 \le i < n} \mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(f^{i+1}(x)) = 1\Big]\right| \\
&= \frac{1}{n}\Big|\mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(x) = 1\Big] - \mathsf{Pr}\Big[x \leftarrow \mathcal{U} : A(f^n(x)) = 1\Big]\Big| = \frac{1}{n} \cdot \delta ,
\end{aligned}$$

where δ is A's success probability.

If f is S-UD, then the success probability of any f-detecting ρ-adversary must be less than $\frac{\rho}{S}$. Therefore, we have $\frac{1}{n} \cdot \delta = \delta' < \frac{\rho}{S}$, which gives $\delta < \frac{\rho}{S/n}$, and thus f^n is indeed $\frac{S}{n}$-UD. □

In the following, we will consider general hash functions $f : \{0,1\}^\star \to \{0,1\}^k$ mapping arbitrary-length inputs to fixed-length outputs. We will write $f(x_1, x_2)$ or $f(x_1, x_2, \ldots, x_n)$ to mean the result of applying f to a bit-string encoding the pair (x_1, x_2) or the tuple $(x_1, x_2, \ldots, x_n)$, respectively.

3.2 Cryptographic Time-Stamping

We model the ideal time-stamping service as a trusted repository R that works as follows:

- The time t is initialized to 1, and all the cells R_i to $\perp$.
- The query $R.\mathtt{time}$ is answered with the current value of t.
- The query $R.\mathtt{get}(t)$ is answered with R_t.
- On the request $R.\mathtt{put}(x)$, first $R_t \leftarrow x$ is assigned and then the value of t is incremented by 1.

This is done for the sake of simplicity. It turns out that refining the model of the time-stamping service would make the proofs really complex. For example, even for a seemingly trivial change, where R publishes a hash $h(m, \tau)$ instead of just (m, τ), one needs non-standard security assumptions about h such as non-malleability. In this paper, we try to avoid these technical difficulties and focus on the basic logic of the security argument of the tag-based signature scheme.

3.3 One-Time Signature Scheme

Definition 6 (Induced signature scheme). *A tag system* $(\mathrm{Gen}, \mathrm{Tag}, \mathrm{Ver})$ *and a time-stamping repository* R *induce a one-time signature scheme as follows:*

The signer $S^R(m)$ *queries* $t \leftarrow R.\mathtt{time}$*, then creates* $\tau \leftarrow \mathrm{Tag}(\mathsf{sk}, t)$*, stores* $R.\mathtt{put}((m, \tau))$*, and returns* $\sigma = (\tau, t)$*.*

The verifier $V^R(m, (\tau, t), \mathsf{pk})$ *queries* $x \leftarrow R.\mathtt{get}(t)$*, and checks that* $x = (m, \tau)$ *and* $\mathrm{Ver}(\mathsf{pk}, t, \tau) = 1$*.*

Definition 7 (Existential unforgeability). *A one-time signature scheme is* S*-secure existentially unforgeable (*S*-EUF), if every forging adversary* A *using computational resources* ρ *has success probability*

$$\mathsf{Pr}\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(T), (m, \sigma) \leftarrow A^{S^R, R}(\mathsf{pk}) : V^R(m, \sigma, \mathsf{pk}) = 1\Big] < \frac{\rho}{S} \ ,$$

where A *makes only one* S*-query and not with* m*.*

Theorem 1. *If the tag system is* S*-secure forward-resistant then the induced one-time signature scheme is (almost)* S*-secure existentially unforgeable.*

Proof. Having a ρ-adversary $A^{S^R, R}$ for the signature scheme, we construct an adversary $B^{\mathrm{Tag}(\mathsf{sk}, \cdot)}$ for the tag scheme as follows.

The adversary B simulates the adversary A by creating a simulated R of its own. A signing query $S(m)$ is simulated by making an oracle query $\tau \leftarrow$

$\mathrm{Tag}(\mathsf{sk}, t)$, where t is the time value in the simulated R, and then assigning $R_t \leftarrow (m, \tau)$.

Every time the simulated A makes (a direct) query $R.\mathtt{put}(x)$, B checks whether x is in the form (m, τ) and $\mathrm{Ver}(\mathsf{pk}, \tau, t) = 1$, where t is the current time in the simulated R, and then returns (τ, t) if either:

- A has never made any S-calls, or
- A has made an S-call with $m' \neq m$.

It is easy to see that one of these events must occur whenever A is successful. In the first case, B is also successful, because it outputs a correct tag without making any $\mathrm{Tag}(\mathsf{sk}, \cdot)$-calls. In the second case, the $S(m')$-query was made at $t' < t$ (as every S-query makes one $R.\mathtt{put}(\cdot)$-query which advances t) and then also the $\mathrm{Tag}(\mathsf{sk}, t')$-query was made at $t' < t$ and hence B is successful again.

If the overhead of B in simulating the environment for A is small, the reduction is tight and thus the signature scheme must indeed be almost as secure as the underlying tag scheme. □

3.4 The BLT Scheme as a Tag System (BLT-TB)

Ignoring the aggregation of individual time-bound keys into a hash tree, the essence of the BLT signature scheme proposed in [13] can be modeled as a tag system as follows:

- The secret key sk is a list $(z_1, z_2, \ldots, z_T)$ of T unpredictable values.
- The public key pk is the list $(f(z_1), f(z_2), \ldots, f(z_T))$, where f is a one-way function.
- The tagging algorithm $\mathrm{Tag}(z_1, z_2, \ldots, z_T; t)$ outputs z_t.
- The verification algorithm Ver, given as input a tag τ, an integer t, and the public key $(x_1, x_2, \ldots, x_T)$, checks that $1 \leq t \leq T$ and $f(\tau) = x_t$.

We will refer to this model as the BLT-TB tag system.

Theorem 2. *If f is S-OW, then BLT-TB is an $\frac{S}{T}$-forward-resistant tag system.*

Proof. We assume there's a tag-forging adversary A and construct an f-inverting adversary B based on oracle access to A. Since f is S-OW, irrespective of B's construction, its success probability

$$\delta' = \mathsf{Pr}\Big[z \leftarrow \{0,1\}^k, z' \leftarrow B(f(z)) : f(z') = f(z)\Big] < \frac{\rho}{S} \,.$$

We construct B to process the input $x = f(z)$ as follows:

- generate the secret key components $z_i \leftarrow \{0,1\}^k$ and compute the corresponding public key components $x_i = f(z_i)$ for $1 \leq i \leq T$;
- uniformly randomly pick an index $j \leftarrow \{1, \ldots, T\}$;
- call A on a modified public key to produce a forged tag and its index

$$(\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk}, \cdot)}(x_1, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_T) \,;$$

– if A succeeded and $t = j$ then return τ, else return $\perp$.

By construction, B's success probability $\delta_j = \mathsf{Pr}\big[A \text{ succeeded} \wedge t = j\big]$. Since the distribution of x is identical to the distribution of x_i, the events "A succeeded" and "$t = j$" are independent and thus we have $\delta_j = \mathsf{Pr}\big[A \text{ succeeded}\big] \cdot \mathsf{Pr}\big[t = j\big]$. Since j was drawn uniformly from $\{1, \ldots, T\}$, we further have $\delta_j = \delta \cdot \frac{1}{T}$, where

$$\delta = \mathsf{Pr}\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(T), (\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk}, t')}(\mathsf{pk}) : t' < t, \mathrm{Ver}(\tau, t, \mathsf{pk}) = 1\Big]$$

is A's success probability.

From f being S-OW, we have $\frac{\delta}{T} = \delta_j \leq \delta' < \frac{\rho}{S}$. Thus, $\delta < \frac{\rho}{S/T}$, and BLT-TB is indeed an $\frac{S}{T}$-forward-resistant tag system. □

3.5 The BLT-OT Tag System

We now define the BLT-OT tag system (inspired by Lamport's one-time signatures [22]) as follows:

- The secret key sk is a list $(z_0, z_1, \ldots, z_{\ell-1})$ of $\ell = \lceil \log_2(T+1) \rceil$ unpredictable values.
- The public key pk is the list $(f(z_0), f(z_1), \ldots, f(z_{\ell-1}))$, where f is a one-way function.
- The tagging algorithm $\mathrm{Tag}(z_0, \ldots, z_{\ell-1}; t)$ outputs an ordered subset $(z_{j_1}, z_{j_2}, \ldots, z_{j_m})$ of components of the secret key such that $0 \leq j_1 < j_2 < \ldots < j_m \leq \ell - 1$ and $2^{j_1} + 2^{j_2} + \ldots + 2^{j_m} = t$.
- The verification algorithm Ver, given as input a sequence $(z_{j_1}, z_{j_2}, \ldots, z_{j_m})$, an integer t, and the public key $(x_0, x_1, \ldots, x_{\ell-1})$, checks that:
  1. $f(z_{j_1}) = x_{j_1}, \ldots, f(z_{j_m}) = x_{j_m}$; and
  2. $0 \leq j_1 < j_2 < \ldots < j_m \leq \ell - 1$; and
  3. $2^{j_1} + 2^{j_2} + \ldots + 2^{j_m} = t$; and
  4. $1 \leq t \leq T$.

Theorem 3. *If f is S-OW, then BLT-OT is an $\frac{S}{\ell}$-forward-resistant tag system.*

Proof. The proof is again by assuming there's a tag-forging adversary A and constructing an f-inverting adversary B based on oracle access to A. As before, f being S-OW implies B's success probability

$$\delta' = \mathsf{Pr}\Big[z \leftarrow \{0,1\}^k, z' \leftarrow B(f(z)) : f(z') = f(z)\Big] < \frac{\rho}{S}\,.$$

Again, let δ be A's success probability

$$\delta = \mathsf{Pr}\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(T), (\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk}, t')}(\mathsf{pk}) : t' < t, \mathrm{Ver}(\tau, t, \mathsf{pk}) = 1\Big]\,.$$

We construct B to process the input $x = f(z)$ as follows:

- generate the secret key components $z_i \leftarrow \{0,1\}^k$ and compute the corresponding public key components $x_i = f(z_i)$ for $0 \leq i \leq \ell - 1$;

– uniformly randomly pick an index $j \leftarrow \{0, \ldots, \ell - 1\}$;
– call A on a modified public key to produce a forged tag and its index

$$(\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk},\cdot)}(x_0, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_{\ell-1}) \; ;$$

– in case of A's success let j' be the leftmost bit position that is 0 in t', but 1 in t (with $t' < t$, at least one such bit position must exist);
– if A succeeded and $j' = j$ then return the component of τ corresponding to bit position j in t, else return $\perp$.

By construction, B's success probability $\delta_j = \mathsf{Pr}\big[A \text{ succeeded } \wedge \; j' = j\big]$. Since the distribution of x is identical to the distribution of x_i, the events "A succeeded" and "$j' = j$" are independent and thus we have $\delta_j = \mathsf{Pr}\big[A \text{ succeeded}\big] \cdot \mathsf{Pr}\big[j' = j\big]$. Since j was drawn uniformly from $\{0, \ldots, \ell - 1\}$, we further have $\delta_j = \delta \cdot \frac{1}{\ell}$.

From f being S-OW, we have $\frac{\delta}{\ell} = \delta_j \leq \delta' < \frac{\rho}{S}$. Thus, $\delta < \frac{\rho}{S/\ell}$, and BLT-OT is indeed an $\frac{S}{\ell}$-forward-resistant tag system. □

3.6 The BLT-W Tag System

We now define the BLT-W tag system (inspired by Winternitz's idea [34] for optimizing the size of Lamport's one-time signatures) as follows:

– The secret key sk is an unpredictable value z.
– The public key pk is $f^T(z)$, where f is a one-way function.
– The tagging algorithm $\mathrm{Tag}(z; t)$ outputs the value $f^{T-t}(z)$.
– The verification algorithm Ver, given as input a tag τ, an integer t, and the public key x, checks that $1 \leq t \leq T$ and $f^t(\tau) = x$.

Theorem 4. *If f is S_1-OW and S_2-CR and S_3-UD function, then BLT-W is a $\frac{\min(S_1,S_2,S_3)}{2 \cdot T}$-forward-resistant tag system.*

Proof. The proof is again by assuming there's a tag-forging adversary A and constructing an f-breaking adversary B based on oracle access to A. As before, f being S_1-OW implies B's success probability at finding a preimage of f

$$\delta_1 = \mathsf{Pr}\Big[z \leftarrow \{0,1\}^k, z' \leftarrow B(f(z)) : f(z') = f(z)\Big] < \frac{\rho}{S_1}$$

and f being S_2-CR implies B's success probability at finding a collision of f

$$\delta_2 = \mathsf{Pr}\Big[z_1, z_2 \leftarrow B(f(\cdot)) : f(z_1) = f(z_2)\Big] < \frac{\rho}{S_2} \; .$$

Once again, let δ be A's success probability

$$\delta = \mathsf{Pr}\Big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathrm{Gen}(T), (\tau, t) \leftarrow A^{\mathrm{Tag}(\mathsf{sk},t')}(\mathsf{pk}) : t' < t, \mathrm{Ver}(\tau, t, \mathsf{pk}) = 1\Big] \; .$$

We construct B to process the input $x = f(z)$ as follows:

– uniformly randomly pick an index $j \leftarrow \{1, \ldots, T\}$;
– compute $y = f^j(x)$ and construct a modified tagging oracle $\mathrm{Tag}'(x;t)$ that returns $f^{j-t}(x)$ for $t \leq j$ and $\perp$ for $t > j$;
– call A on y with the modified oracle Tag' to produce a forged tag and its index

$$(\tau, t) \leftarrow A^{\mathrm{Tag}'(x,\cdot)}(y) \ ;$$

– if A succeeded with $t' = j$ then
  • if $f^{t-t'}(\tau) = x$ then return $f^{t-t'-1}(\tau)$ as the preimage of x under f;
  • otherwise find the smallest i such that $f^{t-t'+i}(\tau) = f^i(x)$ and return $(f^{t-t'+i-1}(\tau), f^{i-1}(x))$ as a collision of f.

By construction, B's success probability $\delta_j = \mathsf{Pr}\big[A \text{ succeeded } \wedge\ t' = j\big]$. With f being S_3-UD, the probability of A detecting the difference between $f^i(x)$ for $i = 0$ and $i > 0$ is at most $\frac{\rho}{S_3/T} = \frac{\rho \cdot T}{S_3}$. Then, with probability at least $1 - \frac{\rho \cdot T}{S_3}$, we have $\delta_j = \mathsf{Pr}\big[A \text{ succeeded}\big] \cdot \mathsf{Pr}\big[t' = j\big] = \delta \cdot \frac{1}{T}$.

On the other hand, from f being S_1-OW and S_2-CR, we have

$$\delta_j \leq \max(\delta_1, \delta_2) < \max(\frac{\rho}{S_1}, \frac{\rho}{S_2}) = \frac{\rho}{\min(S_1, S_2)} \ .$$

Thus, with probability at least $1 - \frac{\rho \cdot T}{S_3}$, we have $\frac{\delta}{T} < \frac{\rho}{\min(S_1, S_2)}$ and with probability at most $\frac{\rho \cdot T}{S_3}$ we don't have a non-trivial upper bound on δ. Therefore,

$$\begin{aligned}\delta &< \left(1 - \frac{\rho \cdot T}{S_3}\right) \cdot \frac{\rho \cdot T}{\min(S_1, S_2)} + \frac{\rho \cdot T}{S_3} = \frac{\rho \cdot T}{\min(S_1, S_2)} + \frac{\rho \cdot T}{S_3} \cdot \left(1 - \frac{\rho \cdot T}{\min(S_1, S_2)}\right) \\ &< \frac{\rho \cdot T}{\min(S_1, S_2)} + \frac{\rho \cdot T}{S_3} = \rho \cdot T \cdot \frac{\min(S_1, S_2) + S_3}{\min(S_1, S_2) \cdot S_3} \leq \rho \cdot T \cdot \frac{2}{\min(S_1, S_2, S_3)}\end{aligned}$$

and BLT-W is indeed at least $\frac{\min(S_1, S_2, S_3)}{2 \cdot T}$-forward-resistant tag system. □

4 New Practical Signature Schemes

4.1 BLT-OT One-Time Signature Scheme

The signature scheme induced by the BLT-OT tag system according to Definition 6 would come with the requirement that the signer must know in advance the time at which its request reaches the time-stamping service. This is hard to achieve in practice, in particular for personal signing devices such as smart cards that lack built-in clocks. To overcome this limitation, we construct the BLT-OT one-time signature scheme as follows.

Key Generation. Let ℓ be the number of bits that can represent any time value t when the signature may be created (e.g. $\ell = 32$ for POSIX time up to year 2106). The private key is generated as $\mathsf{sk} = (z_0, z_1, \ldots, z_{\ell-1})$, where z_i are unpredictable values, and the public key as $\mathsf{pk} = f(X)$, where $X = (x_0, x_1, \ldots, x_{\ell-1})$, $x_i = f(z_i)$, and f is a one-way function.

Public Key Certificate. The certificate should contain the following data:

- the public key pk;
- the identity ID_c of the client;
- the identity ID_s of the designated time-stamping service.

Recording the identity of the designated time-stamping service in certificate enables instant key revocation. Upon receiving a revocation notice, the designated service stops serving the affected client, and thus it is not possible to generate signatures using revoked keys.

Signing. To sign a message m, the client:

- gets a time-stamp S_t on the record (m, X, ID_c) from the time-stamping service designated by ID_s;
- extracts the ℓ-bit time value t from S_t and creates the list $W = (w_0, w_1, \ldots, w_{\ell-1})$, where
  - $w_i = z_i$ if the i-th bit of t is 1, or
  - $w_i = x_i = f(z_i)$ otherwise;
- disposes of the private key $(z_0, z_1, \ldots, z_{\ell-1})$ to prevent its re-use;
- emits (W, S_t) as the signature.

Verification. To verify the signature (W, S_t) on the message m against the certificate $(\mathsf{pk}, ID_c, ID_s)$, the verifier:

- extracts time t from the time-stamp S_t;
- recovers the list $X = (x_0, x_1, \ldots, x_{\ell-1})$ by computing
  - $x_i = f(w_i)$ if the i-th bit of t is 1, or
  - $x_i = w_i$ otherwise;
- checks that the computed X matches the public key: $f(X) = \mathsf{pk}$;
- checks that S_t is a valid time-stamp issued at time t by service ID_s on the record (m, X, ID_c).

Using the reduction techniques from previous sections to formally prove the security of this optimized signature scheme is complicated by both the iterated use of f and the more abstract view of the time-stamping service.

4.2 Other Schemes

Details of other optimized signature schemes are skipped for brevity. Practical properties are discussed in the following sections.

When we combine linking-based time-stamping—as abstracted by Buldas *et al.* [18]—with the BLT-TB tag scheme, we get a signature scheme equivalent to one presented in [13] and proven to be existentially unforgeable against adaptive chosen-message attack (EUF-ACM) in the random oracle (RO) model.

In this paper we modeled time-stamping in a simplistic way, but nevertheless, Theorem 1 shows that a forward-resistant tag system induces a secure (one-time) signature scheme; and we have provided a forward-resistance proof for each proposed tag system.

We do note that security guarantees offered by digital signature-based time-stamping are not adequate and linking-based time-stamping services, e.g. [11] must be used. Publishing in blockchains can be considered as an alternative.

5 Discussion

The BLT-TB scheme proposed in [13] works well for powerful devices that are constantly running and have reliable clocks. These are not reasonable assumptions for personal signing devices such as smart cards, which have very limited capabilities and are not used very often. Generating keys could take hours or even days of non-stop computing on such devices. This is clearly impractical, and also wasteful as most of the keys would go unused.

The BLT-OT scheme proposed in Sec. 4 solves the problems described above at the cost of introducing state on the client side. As the scheme is targeted towards personal signing devices, the statefulness is not a big risk, because these devices are not backed up and also do not support parallel processing. The benefit in addition to improved efficiency is that the device no longer needs to know the current time while preparing a signing request. Instead, it can just use the time from the time-stamp when composing the signature.

5.1 Efficiency as One-Time Scheme

When implemented as described in Sec. 4, the cost of generating a BLT-OT key pair is ℓ random key generations and $\ell + 1$ hashing operations, the cost of signing $\ell + 1$ hashing operations and one time-stamping service call, and the cost of signature verification at most $\ell + 1$ hashing operations and one time-stamp verification. In this case the private key would consist of ℓ one-time keys and the public key of one hash value, and the signature would contain ℓ hash values and one time-stamp token. The private storage size can be optimized by generating

Table 1. Efficiency of hash-based one-time signature schemes. We assume 256-bit hash functions, 32-bit time values, and time-stamping hash-tree with 33 levels. Times are in hashing operations and signature sizes in hash values. TS in BLT schemes stands for the time-stamping service call.

Scheme	Key generation	Signing time	Verification time	Signature size
Lamport	1 025	1 024	513	256
Winternitz ($w = 4$)	1 089	1 088	1 021	68
BLT-OT	65	64 + TS	33 + 33	32 + 33
BLT-W ($w = 2$)	65	64 + TS	49 + 33	16 + 33

the one-time keys from one true random seed using a pseudo-random generator. Then the cost of signing increases by ℓ operations, as the one-time keys would have to be re-generated from the seed before signing. This version is listed as BLT-OT in Table 1.

Winternitz's idea [34] for optimizing the size of Lamport's one-time signatures [22] can also be applied to BLT-OT. Instead of using one-step hash chains $z_i \to h(z_i) = x_i$ to encode single bits of t, we can use longer chains $z_i \to h(z_i) \to \ldots \to h^n(z_i) = x_i$ and publish the value $h^{n-j}(z_i)$ in the signature to encode the value j of a group of bits of t. When encoding groups of w bits of t in this manner, the chains have to be $n = 2^w$ steps long. This reduces the size of the signature by w times, but increases the costs of key generation and signing by a bit less than $\frac{2^{w-1}}{w}$ times and the cost of verification by a bit less than $\frac{2^w-1}{w}$ times. Note that for $w = 2$, only the verification cost increases by about 50%! Also note that in contrast to applying this idea to Lamport's signatures, in BLT-W no additional countermeasures are needed to prevent an adversary from stepping the hash chains forward: the time in the time-stamp takes that role. This version is listed as BLT-W in Table 1.

To compare BLT-OT signature sizes and verification times to other schemes, we also need to estimate the size of hash-trees built by the time-stamping service. Even assuming the whole world (8 billion people) will use the time-stamping service in every aggregation round, an aggregation tree of 33 layers will suffice. We also assume that in all schemes one-time private keys will be generated on-demand from a single random seed and public keys will be aggregated into a single hash value. Therefore, the key sizes will be the same for all schemes and are not listed in Table 1.

5.2 Efficiency as Many-Time Scheme

A one-time signature scheme is not practical by itself. Merkle [33] proposed aggregating multiple public keys of a one-time scheme using a hash tree to produce so-called N-time schemes. Assuming 10 signing operations per day, a set of 3 650 BLT-OT keys would be sufficient for a year. The key generation costs would obviously grow correspondingly. The change in signing time would depend on how

Table 2. Efficiency of hash-based many-time signature schemes. We assume key supply for at least 3 650 signatures, 256-bit hash functions, 32-bit time values, and time-stamping hash-tree with 33 levels. Times are in hashing operations and signature sizes in hash values. TS in BLT schemes stands for the time-stamping service call.

Scheme	Key generation	Signing time	Verification time	Signature size
XMSS	897 024	8 574	1 151	79
SPHINCS	ca 16 000	ca 250 000	ca 7 000	ca 1 200
BLT-TB	ca 96 000 000	50 + TS	25 + 33	25 + 33
BLT-OT-N	240 900	64 + TS	45 + 33	44 + 33
BLT-W-N ($w = 2$)	240 900	64 + TS	61 + 33	28 + 33

the hash tree would be handled. If sufficient memory is available to keep the tree (which does not contain private key material and thus may be stored in regular memory), the authenticating hash chains for individual one-time public keys could be extracted with no extra hash computations. Signature size and verification time would increase by the 12 additional hashing steps linking the one-time public keys to the root of the aggregation tree. This scheme is listed as BLT-OT-N in Table 2, where we compare it with the following schemes:

- XMSS is a stateful scheme, like the N-time scheme built from BLT-OT; the values in Table 2 are computed by taking $N = 2^{12} = 4\,096$ and leaving other parameters as in [8];
- SPHINCS is a stateless scheme and can produce an indefinite number of signatures; the values in Table 2 are inferred from [4] counting invocations of the ChaCha12 cipher on 64-byte inputs as hash function evaluations;
- the values for BLT-TB in Table 2 are from [13].

As can be seen from the table, the performance of BLT-OT as a component in N-time scheme is very competitive when signing and verification time and signature size are concerned. Only SPHINCS has significantly faster key generation, but much slower signing and verification and much larger signatures.

6 Conclusions and Outlook

We have presented a new approach to constructing digital signature schemes from forward-resistant tags and time-stamping services. We observe that this new framework can be used to model an existing signature scheme, and also to construct new ones.

The newly derived signature schemes are practical and it would be interesting to further study their security properties, e.g. present security proofs in the standard model.

The novel concept of forward-resistant tags has already proven useful, and thus certainly merits further research.

References

1. R. J. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. M. Needham. A new family of authentication protocols. *Operating Systems Review*, 32(4):9–20, 1998.
2. Asokan, G. Tsudik, and M. Waidner. Server-supported signatures. *Journal of Computer Security*, 5(1):91–108, 1997.
3. J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical report, Clarkson University, 1991.
4. D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In *EUROCRYPT 2015, Proceedings, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, 2015.

5. K. Bicakci and N. Baykal. Server assisted signatures revisited. In *CT-RSA 2004, Proceedings*, volume 2964 of *LNCS*, pages 143–156. Springer, 2004.
6. J. A. Buchmann, L. C. Coronado García, E. Dahmen, M. Döring, and E. Klintsevich. CMSS—An improved Merkle signature scheme. In *INDOCRYPT 2006, Proceedings*, volume 4329 of *LNCS*, pages 349–363. Springer, 2006.
7. J. A. Buchmann, E. Dahmen, S. Ereth, A. Hülsing, and M. Rückert. On the security of the Winternitz one-time signature scheme. *IJACT*, 3(1):84–96, 2013.
8. J. A. Buchmann, E. Dahmen, and A. Hülsing. XMSS—A practical forward secure signature scheme based on minimal security assumptions. In *PQCrypto 2011, Proceedings*, volume 7071 of *LNCS*, pages 117–129. Springer, 2011.
9. J. A. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *ACNS 2007, Proceedings*, volume 4521 of *LNCS*, pages 31–45. Springer, 2007.
10. A. Buldas, A. Kalu, P. Laud, and M. Oruaas. Server-supported RSA signatures for mobile devices. In *ESORICS 2017, Proceedings, Part I*, volume 10492 of *LNCS*, pages 315–333. Springer, 2017.
11. A. Buldas, A. Kroonmaa, and R. Laanoja. Keyless signatures' infrastructure: How to build global distributed hash-trees. In *NordSec 2013, Proceedings*, volume 8208 of *LNCS*, pages 313–320. Springer, 2013.
12. A. Buldas, R. Laanoja, P. Laud, and A. Truu. Bounded pre-image awareness and the security of hash-tree keyless signatures. In *ProvSec 2014, Proceedings*, volume 8782 of *LNCS*, pages 130–145. Springer, 2014.
13. A. Buldas, R. Laanoja, and A. Truu. A server-assisted hash-based signature scheme. In *NordSec 2017, Proceedings*, volume 10674 of *LNCS*, pages 3–17. Springer, 2017.
14. A. Buldas and P. Laud. New linking schemes for digital time-stamping. In *ICISC'98, Proceedings*, pages 3–14. KIISC, 1998.
15. A. Buldas, P. Laud, H. Lipmaa, and J. Villemson. Time-stamping with binary linking schemes. In *CRYPTO'98, Proceedings*, volume 1462 of *LNCS*, pages 486–501. Springer, 1998.
16. A. Buldas, H. Lipmaa, and B. Schoenmakers. Optimally efficient accountable time-stamping. In *PKC 2000, Proceedings*, volume 1751 of *LNCS*, pages 293–305. Springer, 2000.
17. A. Buldas and M. Saarepera. Electronic signature system with small number of private keys. In *2nd Annual PKI Research Workshop, Proceedings*, pages 96–108. NIST, 2003.
18. A. Buldas and M. Saarepera. On provably secure time-stamping schemes. In *ASIACRYPT 2004, Proceedings*, volume 3329 of *LNCS*, pages 500–514. Springer, 2004.
19. J. Camenisch, A. Lehmann, G. Neven, and K. Samelin. Virtual smart cards: How to sign with a password and a server. In *SCN 2016, Proceedings*, volume 9841 of *LNCS*, pages 353–371. Springer, 2016.
20. L. C. Coronado García. *Provably Secure and Practical Signature Schemes*. PhD thesis, Darmstadt University of Technology, Germany, 2005.
21. E. Dahmen, K. Okeya, T. Takagi, and C. Vuillaume. Digital signatures out of second-preimage resistant hash functions. In *PQCrypto 2008, Proceedings*, volume 5299 of *LNCS*, pages 109–123. Springer, 2008.
22. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

23. C. Dods, N. P. Smart, and M. Stam. Hash based digital signature schemes. In *Cryptography and Coding, Proceedings*, volume 3796 of *LNCS*, pages 96–115. Springer, 2005.
24. European Commission. Regulation no 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/EC (eIDAS regulation). *Official Journal of the European Union*, L 257:73–114, 2014.
25. S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.
26. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
27. V. Goyal. More efficient server assisted one time signatures. Cryptology ePrint Archive, Report 2004/135, 2004. https://eprint.iacr.org/2004/135.
28. S. Haber and W. S. Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.
29. A. Hülsing. W-OTS+—Shorter signatures for hash-based signature schemes. In *AFRICACRYPT 2013, Proceedings*, volume 7918 of *LNCS*, pages 173–188. Springer, 2013.
30. A. Hülsing, L. Rausch, and J. A. Buchmann. Optimal parameters for XMSS MT. In *CD-ARES 2013, Proceedings*, volume 8128 of *LNCS*, pages 194–208. Springer, 2013.
31. A. Hülsing, J. Rijneveld, and F. Song. Mitigating multi-target attacks in hash-based signatures. In *PKC 2016, Proceedings, Part I*, volume 9614 of *LNCS*, pages 387–416. Springer, 2016.
32. T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *EUROCRYPT 2002, Proceedings*, volume 2332 of *LNCS*, pages 400–417. Springer, 2002.
33. R. C. Merkle. *Secrecy, Authentication and Public Key Systems*. PhD thesis, Stanford University, 1979.
34. R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO'87, Proceedings*, volume 293 of *LNCS*, pages 369–378. Springer, 1987.
35. A. Perrig. The BiBa one-time signature and broadcast authentication protocol. In *ACM CCS 2001, Proceedings*, pages 28–37. ACM, 2001.
36. A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *CryptoBytes*, 5(2):2–13, 2002.
37. T. Perrin, L. Bruns, J. Moreh, and T. Olkin. Delegated cryptography, online trusted third parties, and PKI. In *1st Annual PKI Research Workshop, Proceedings*, pages 97–116. NIST, 2002.
38. L. Reyzin and N. Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *ACISP 2002, Proceedings*, volume 2384 of *LNCS*, pages 144–153. Springer, 2002.
39. P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *ACM CCS'99, Proceedings*, pages 93–100. ACM, 1999.