Ultra Lightweight Multiple-time Digital Signature for the Internet of Things Devices

Attila A. Yavuz, *Member, IEEE*, Muslum Ozgur Ozmen

***Abstract*—Digital signatures are basic cryptographic tools to provide authentication and integrity in the emerging ubiquitous systems in which resource-constrained devices are expected to operate securely and efficiently. However, existing digital signatures might not be fully practical for such resource-constrained devices (e.g., medical implants) that have energy limitations. Some other computationally efficient alternatives (e.g., one-time/multiple-time signatures) may introduce high memory and/or communication overhead due to large private key and signature sizes.**

In this paper, our contributions are two-fold: First, we develop a new lightweight multiple-time digital signature scheme called Signer Efficient Multiple-time Elliptic Curve Signature (`SEMECS`), which is suitable for resource-constrained embedded devices. `SEMECS` achieves optimal signature and private key sizes for an EC-based signature *without* requiring any EC operation (e.g., EC scalar multiplication or addition) at the signer. We prove `SEMECS` is secure (in random oracle model) with a tight security reduction. Second, we fully implemented `SEMECS` on 8-bit AVR microprocessor with a comprehensive energy consumption analysis and comparison. Our experiments confirm up to 19× less battery-consumption for `SEMECS` as compared to its fastest (full-time) counterpart, SchnorrQ, while offering significant performance advantages over its multiple-time counterparts in various fronts. We open-source our implementation for public testing and adoption.

***Index Terms*—Applied cryptography; Digital signatures; Lightweight cryptography; Internet of Things Security; Embedded device security.**

I. Introduction

Resource-constrained devices (e.g., low-end sensors, smart-cards, RFID-tags) play an important role in emerging ubiquitous systems like Internet of Things and Systems (IoTS) and Wireless Sensor Networks (WSNs). Using service oriented architecture (SOA) further broadens the horizons of IoTS, opening up many applications where resource-constrained devices can participate as service consumers and/or providers. [1]–[3] It is vital for such systems to operate securely and efficiently. Hence, it is necessary to provide authentication and integrity for resource-constrained devices. For instance, guaranteeing the integrity and authentication of financial transactions in a smart-card or RFID-tag is critical for any commercial application. However, this is a challenging task due to the memory, processor, bandwidth and battery limitations of these devices.

Attila A. Yavuz and Muslum Ozgur Ozmen are with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA.

E-mail: attilaayavuz@usf.edu, ozmen@mail.usf.edu

Part of this work is completed when Attila A. Yavuz and Muslum Ozgur Ozmen were with the Department of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA.

It is also important to be able to publicly verify the authentication tags produced by resource-constrained devices. This enables any resourceful device (e.g., a laptop or a base station) to publicly audit transactions and system status.

Symmetric cryptography primitives such as Message Authentication Codes (MACs) are computationally efficient and therefore are preferred for resource-constrained devices in small-scale systems. However, such primitives are not scalable for large-distributed systems and are not publicly verifiable [4], [5]. They also cannot achieve the non-repudiation property, which is necessary for various applications (e.g., transportation payment systems, medical implants, logical/physical access with tiny devices) that may need public auditing.

Digital signatures rely on public key infrastructures for the signature verification [6], [7]. They are publicly verifiable, scalable for large systems and achieve non-repudiation. Therefore, they are highly useful authentication tools for security-critical applications such as medical devices, payment systems, secure auditing in embedded devices and security systems (e.g., building access control). However, existing digital signatures also have some limitations that might prevent them to be fully practical for highly resource-constrained devices.

In the following, we first briefly discuss some prominent digital signature alternatives and their limitations when employed on resource-limited devices. We then present our contributions by summarizing the desirable properties of our scheme, followed by its limitations.

A. Limitations of Signer Efficient Signatures

The existing digital signature alternatives do not offer small private key size, small signature size, and high efficiency at the signer, at the same time. We elaborate on some of these alternatives below.

Traditional Signatures: RSA [8], one of the most well-known signature schemes, is computationally efficient at the verifier's side. However, it requires an *expensive operation*[1] at the signer's side and have large key/signature sizes. Hence, it may not be suitable for resource-constrained embedded devices (e.g., medical implants).

Elliptic Curve (EC) based schemes are highly popular on such devices due to their small key and signature sizes, along with better efficiency compared to RSA [11]–[13]. Various different curves, and signature schemes on these curves have been proposed, which offer improved computational efficiency and security [14]–[18]. Some of these curves are

[1]We refer to operations such as modular exponentiation [6], elliptic curve scalar multiplication [9] or pairing [10] as expensive operations.

also implemented in embedded devices [19]–[21] such as 8-bit AVR microprocessors. However, these signature schemes still require an expensive operation (i.e., EC scalar multiplication) at the signer's side. This requirement may hinder an efficient adoption of these schemes to low-end microprocessors with critical battery limitations (e.g., medical implants).

Online/Offline Signatures: Many techniques have been proposed to improve the efficiency of traditional signatures. These include online/offline signatures that eliminate expensive operations in signature generation via pre-computed tokens generated offline [22], [23]. Although these schemes are computationally efficient, they incur large storage overhead to the signer. Later, Shamir et al. in [24] proposed an improved online/offline signature that is more space efficient. However, by nature, these schemes require linear storage with respect to the number of signatures that can be generated, which is impractical for storage-limited signers.

One-time Signatures (OTSs): These schemes rely on one-way functions without trapdoors and offer very efficient computations [25]–[27]. Specifically, Lamport [25] proposed the first one-time signature scheme, where for each bit of the hash of the message, two hash outputs were stored as the public key that resulted in a very large size. Then, in HORS signature scheme [26], special message encoding techniques have been considered to significantly reduce the public while preserving the computational efficiency. Hash-based schemes also offer post-quantum security that is lacked in most of the traditional signatures. On the other hand, they have a large signature and very large public key sizes. Some EC-based OTSs also exist [28] that offer small signature size, but with a trade-off between private key size and signature generation efficiency. Moreover, in OTSs, a private/public key pair can be used only once. This may require costly private key generations at the resource-limited device for each message to be signed.

Multiple-time Signatures: For these schemes, after K signatures, the key pair must be re-generated. Therefore, *we refer to these schemes as K-time signatures*. Inherently, OTSs (e.g., Lamport [25], HORS [26]) can be used as K-time signatures if K key pairs are generated at the key generation phase. Some hash-based multiple-time signatures were proposed [29]–[32] based on HORS signature [26]. However, these schemes either suffer from large key/signature sizes [29]–[31], or provide security only for a short-limited amount of time [32]. Some stateless hash-based schemes were also proposed [33], extending these multiple-time signatures to full-time (traditional) signature schemes. Although it is shown that they can be implemented on resource-constrained devices [34], it is highly computationally costly at the signer's side, and it also requires the transmission of large signatures (e.g., up to 41 KB).

Our proposed scheme also falls into this category and inherits some of the limitations of K-time signatures (e.g., after K signatures, the key must be re-generated). However, due to the unique construction of our scheme that leverages Fiat-Shamir transform with compact and efficient elliptic curves, it offers the highest signer efficiency among all the aforementioned K-time signatures (including the use of OTSs as K-time signatures). For instance, as shown in Table I, our scheme outperforms HORS [26] (most efficient counterpart) $6\times$ at signature generation and $12\times$ at signature size on an 8-bit microprocessor.

Lattice-based and Code-based Signatures: The main advantage of these schemes is *post-quantum security*. Although some of these schemes offer computational efficiency, they are still relatively computationally expensive (e.g., require heavy operations such as Gaussian Sampling [35]) and key/signature sizes might be prohibitive for resource-constrained devices [35]–[39]. Since these schemes provide long-term security (security against quantum attackers), they might be ideal for resource-constrained devices in the future, if the sizes are reduced.

There is a need for a signer efficient digital signature that prioritizes the signer efficiency by achieving optimal private key and signature sizes without requiring any expensive operations for the signature generation.

B. Our Contribution

In this paper, we create a new multiple-time signature scheme, which we refer to as *Signer Efficient Multiple-time Elliptic Curve Signature (SEMECS)*. We summarize some desirable properties of SEMECS below. A detailed performance analysis is given in Section V.

- *High Computation & Energy Efficiency at the Signer*: SEMECS only requires *two hash function calls, a single modular multiplication, and a modular subtraction* to generate a signature. Therefore, its cost is even comparable to symmetric hash-based MACs that are not suitable for large and distributed systems. SEMECS offers very fast signature generation, (1.23 microseconds on an i7 Skylake processor). On a resource-constrained processor, this translates into high energy efficiency. Our experiments confirmed that SEMECS signature generation has $6\times$ lower energy consumption compared to its closest counterpart and $118\times$ lower than Ed25519 [17], [19] on 8-bit AVR microprocessor (see Table I).
- *Compact Private Key & Signature*: SEMECS only requires storing a 32-byte private key (that can be derived from a 16-byte seed with a PRF) and incurs an additional 32 Bytes to the message as the signature, for $\kappa = 128$-bit security level. SEMECS has two signature components of 32 Bytes, where one of them is used to recover the first 32 Bytes of the message. Therefore, the transmission overhead is just 32 Bytes, that is optimal for EC-based digital signatures (as in BLS [12]). Thus, SEMECS is also lightweight in terms of signer storage and transmission. Due to its small signature size, SEMECS is also very energy efficient in terms of communication, in addition to its high energy efficiency in signer computation. Moreover, since SEMECS does not require any EC operation at the signer's side, the signer does not need to store any curve parameters and codes. This is specifically important for resource-constrained devices that have limited space for the code (e.g., AVR ATmega 2560 has 256 KB).
- *Open-source Implementation and Comprehensive Analysis*: We fully implemented SEMECS on a laptop and the signature generation of SEMECS on an 8-bit AVR microprocessor. We open-sourced all of our implementations for broad testing,

TABLE I: Signer-side performance of SEMECS and its counterparts on 8-bit AVR microprocessor

Scheme	K	Signature Generation Time (CPU cycle)	Private Key (Byte)	Signature Size (Byte)	Computation energy (mJ)	Communication energy (μJ)
Full-time signatures						
SPHINCS [33]	2^{κ}	2 681 600 389	1088	41000	16760.00	6115.56
ECDSA [11]	2^{κ}	48 188 992	32	64	301.18	9.55
Ed25519 [17], [19]	2^{κ}	23 211 611	32	64	145.07	9.55
μKummer [16], [20]	2^{κ}	10 404 033	32	64	65.03	9.55
SchnorrQ [18], [21]	2^{κ}	3 740 000	32	64	23.38	9.55
K-time signatures						
HORS [26]	$\frac{1}{2^{17}}$	1 180 618	16	384	7.38	57.28
HORSE [30]	$\frac{1}{2^{17}}$	1 180 618 19 644 106	16384 278528	384	7.38 122.78	57.28
XMSS [31]	$\frac{1}{2^{17}}$	10 233 600 101 509 850	16	2080 2592	63.96 634.44	310.25 386.62
Zaverucha et al. [28]	$\frac{1}{2^{17}}$	6 250 660	16	48	39.07	7.16
SEMECS	$\frac{1}{2^{17}}$	**195 776**	**32**	**32**	**1.22**	**4.77**

The cost of hash-based schemes are estimated based on the cost of a single hash operation.

benchmarking and adoption purposes. We also analyzed and compared the efficiency of SEMECS with a wide variety of efficient signature schemes (see Section V) on both platforms.

• *Provable Security with a Tight Reduction*: We prove that SEMECS is existentially unforgeable against chosen-message attacks in Random Oracle Model (ROM) [40]. In Section IV, we show that SEMECS has a tight reduction to the Discrete Logarithm Problem (DLP), without the need for the forking lemma [41], as Fiat-Shamir type signatures do. In our security analysis, we exploit the fact that SEMECS is a multiple-time signature, and therefore it has higher security for a limited number of queries (as the nature of multiple-time signatures).

All the above properties show that SEMECS is potentially an ideal alternative to provide authentication and integrity for resource-constrained devices.

Differences of this work with its preliminary version appeared in WiSec 13' [42]: (i) In this work, we developed a new signature scheme that we refer to as SEMECS that reduces the signature/private key size of our preliminary scheme ETA [42], and thereby achieves optimal signature and key sizes for an EC-based signature scheme. Moreover, SEMECS generates private key components deterministically, and therefore offers improved security against weak pseudo-random number generators. (ii) In this work, we provided a full-fledged open-source implementation of SEMECS on 8-bit AVR ATmega 2560 microprocessor with a comprehensive energy consumption analysis and comparison. We also gave comprehensive performance comparison of SEMECS with some of the most recent and efficient digital signatures (including but not limited to Ed25519 [17], SchnorrQ [15], [18], SPHINCS [33], XMSS [31], HORS [26]). (iii) In this work, we provided an improved security proof that achieves a significantly tighter security reduction compared to that of ETA.

Limitations: Despite all its merits, SEMECS also has its limitations that are inherent to multiple-time signatures: (i) It can sign up to a pre-determined K messages, but then needs to be bootstrapped. (ii) It is a stateful signature scheme. (iii) In SEMECS, the public key size is linear with respect to K. This requires verifiers to be storage resourceful.

Potential Use-cases: Remark that for our envisioned applications, the signer computational/storage/communication efficiency is much more important than the verifier storage efficiency alone. Furthermore, these applications permit verifiers to be storage resourceful (e.g., a cloud server for medical systems, base stations in WSNs, control centers in cyber-physical systems). Similarly, it is feasible to perform the key generation phase offline in these applications. In the following, we discuss some of the potential applications for SEMECS.

Medical implants are equipped with resource-constrained microprocessors (e.g., 8-bit AVR [43], as we used in our experiments) that need to report sensitive data to doctors, hospital servers, etc. Symmetric key authentication (MACs) is usually preferred for these systems. However, these mechanisms lack non-repudiation, and public verification that are highly desirable for some medical systems, because of digital forensics and legal issues [44], [45]. Thus, there is a need for low-cost public key primitives (e.g., authentication) for these systems [46]. SEMECS can be considered as an ideal alternative for medical implants due to its signer efficiency. Our experiments on 8-bit AVR showed that SEMECS consumes less energy compared to its counterparts. In practice, this translates into a longer battery life that is critical for medical implants.

Additionally, SEMECS is highly suitable to provide authentication for SOA based IoT systems. SOA based IoT infrastructures are comprised of networked, resource-constrained devices [1], [2] that require efficient authentication mechanisms. Some essential applications of SOA based IoT includes but not limited to e-health, smart product management and smart events for emergency management [2], [47], [48]. Similarly with medical implants, non-repudiation and public verification are critical for these applications. Moreover, a server (or a broker - i.e., coordinators that operate between the server and the devices) is usually utilized the connected resource-constrained devices to provide these services [3]. Servers and brokers are usually equipped with higher-end processors, compared to the IoT devices, that has expandable memories. Therefore, in such SOA based IoT applications, we believe that the server or broker can tolerate the storage of a larger public key in exchange of a significantly higher signer efficiency

that translates into longer battery life for resource-constrained devices.

Many secure WSN protocols such as clone detection [49], secure code dissemination [50] and secure logging [51] include a low-end signer that reports to resourceful servers, and base stations. SEMECS can substantially increase the lifespan of WSNs by serving as the authentication mechanism for such protocols. Moreover, SEMECS can be deployed in some token-based logical/physical access control systems.

II. Preliminaries

In this section, we first give our notation and definitions. We then describe our system and security models.

A. Definitions and Algorithms

Notation. $||$ denotes the concatenation operation. $|r|$ denotes the bit length of variable r. $r \xleftarrow{\$} \mathcal{S}$ denotes that variable r is randomly and uniformly selected from set $\mathcal{S}$. We denote by $\{0,1\}^*$ the set of binary strings of any finite length. $\mathcal{A}^{\mathcal{O}_0,\ldots,\mathcal{O}_i}(\cdot)$ denotes algorithm $\mathcal{A}$ is provided with oracles $\mathcal{O}_0,\ldots,\mathcal{O}_i$. For example, $\mathcal{A}^{SGN.Sig_{sk}}(\cdot)$ denotes that algorithm $\mathcal{A}$ is provided with a *signing oracle* of signature scheme SGN under a private key sk. $H_i : \{0,1\}^* \rightarrow \mathbb{Z}_q^*, i \in \{0,1\}$ are distinct Full Domain Hash Functions [52], where q is a large prime.

Definition 1 A K-time signature scheme SGN is comprised of a tuple of three algorithms (Kg, Sig, Ver) defined as follows:

- $(sk, PK) \leftarrow$ SGN.Kg$(1^\kappa, K)$,: The key generation algorithm takes the security parameter 1^κ and the maximum number of messages to be signed K as the input. It returns a private/public key pair (sk_0, PK) as the output.
- $\sigma_j \leftarrow$ SGN.Sig(sk_j, M_j): The signature generation algorithm takes the private key sk_j, $0 \leq j \leq K-1$ and a message M_j to be signed as the input. It returns a signature σ_j on M_j as the output, and then updates sk_j to sk_{j+1}.
- $b \leftarrow$ SGN.Ver(PK, M_j, σ_j): The signature verification algorithm takes PK, message M_j and its corresponding signature σ_j, $0 \leq j \leq K-1$ as the input. It returns a bit b, with $b = 1$ meaning *valid*, and $b = 0$ otherwise.

SEMECS, and its preliminary version ETA in [42], are inspired from the Schnorr signature scheme [13], which is described in the algorithm below.

B. Models

We give our system and security models as below.

System Model: There are two types of entities in the system.

1) *Resource-constrained Signers*: Signers are storage, computational, bandwidth and power limited devices (e.g., medical implants, wireless sensors, RFID-tags). The objective of SEMECS is to minimize the cryptographic overhead of signers.
2) *Resourceful Verifiers*: Storage resourceful verifiers (e.g., a laptop, base station) that can be any (untrusted) entity.

We assume that the key generation/distribution is performed *offline* before deployment. For instance, a key generation center can generate private/public keys and distribute them to each entity in the system. Otherwise, the signer can also perform the key generation, before deployment, when it does not have any battery limitations.

Algorithm 1 Schnorr Signature Scheme

$(y, Y) \leftarrow$ Schnorr.Kg(1^κ):

1: Generate large primes q and $p > q$ such that $q|(p-1)$.
2: Select a generator α of the subgroup G of order q in $\mathbb{Z}_p^*$.
3: **return** a private/public key pair $(y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow \alpha^y \bmod p)$ and system-wide parameter $I \leftarrow (q, p, \alpha)$ as the output.

$(s, e) \leftarrow$ Schnorr.Sig(y, M): Given y, compute signature σ on a message M as follows:

1: $R \leftarrow \alpha^r \bmod p$.
2: $e \leftarrow H_0(M||R)$.
3: $s \leftarrow (r - e \cdot y) \bmod q$, where $r \xleftarrow{\$} \mathbb{Z}_q^*$.
4: **return** $\sigma = (s, e)$.

$b \leftarrow$ Schnorr.Ver$(Y, M, \langle s, e \rangle)$:

1: $R' \leftarrow Y^e \alpha^s \bmod p$.
2: **if** $e = H_0(M||R')$ **then return** $b = 1$
3: **else return** $b = 0$

Security Model: A standard security notion for a signature scheme is Existential Unforgeability under Chosen Message Attack ($EU\text{-}CMA$) [53]. We define K-time *EU-CMA* experiment (in random oracle model [40]) for SGN as below. In this experiment, Adversary $\mathcal{A}$ is provided with two oracles: (i) A *random oracle* $RO(.)$ from which $\mathcal{A}$ can request the hash of any message M of their choice up to (polynomially unbounded) K' messages. (ii) A signing oracle SGN.Sig$_{sk}(.)$ from which $\mathcal{A}$ can request a SGN signature on any message M of their choice up to (pre-determined constant) K messages.

Definition 2 *EU-CMA experiment* for SGN is as follows:
Experiment $Expt_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A})$
$(sk_0, PK) \leftarrow$ SGN.Kg$(1^\kappa, K)$,
$(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.),\text{SGN.Sig}_{sk}(.)}(PK)$,
If SGN.Ver$(PK, M^*, \sigma^*) = 1$ and M^* was not queried to SGN.Sig, return 1, else, return 0.

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

$$Adv_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A}) = Pr[Expt_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A}) = 1].$$

The EU-CMA-advantage of SGN is defined as

$$Adv_{\text{SGN}}^{EU\text{-}CMA}(t, K', K) = \max_{\mathcal{A}}\{Adv_{\text{SGN}}^{EU\text{-}CMA}(\mathcal{A})\},$$

where the maximum is over all $\mathcal{A}$ having time complexity t, making at most K' queries to $RO(.)$ and at most K queries to SGN.

The security of SEMECS relies on the intractability of *Discrete Logarithm Problem (DLP)* [53], which is defined below.

Definition 3 Given a cyclic group G of order prime q and a generator α of G, let $\mathcal{A}$ be an algorithm that returns an element of $\mathbb{Z}_q^*$. Consider the following experiment:

Experiment $Expt_G^{DL}(\mathcal{A})$

$y \xleftarrow{\$} \mathbb{Z}_q^*$,
$Y \leftarrow \alpha^y \bmod p$,
$y' \leftarrow \mathcal{A}(Y)$,
If $\alpha^{y'} \bmod p = Y$, return 1, else, return 0.

The *DL-advantage of* $\mathcal{A}$ in this experiment is defined as

$$Adv_G^{DL}(\mathcal{A}) = Pr[Expt_G^{DL}(\mathcal{A}) = 1].$$

The *DL-advantage of* (G, α) in this experiment is defined as

$$Adv_G^{DL}(t) = \max_{\mathcal{A}}\{Adv_G^{DL}(\mathcal{A})\},$$

where the maximum is over all $\mathcal{A}$ having time complexity t.

III. The Proposed Scheme

Some DLP-based signatures (e.g., ECDSA [22], Schnorr [13]) can eliminate expensive operations from the signature generation by pre-computing the component $R = \alpha^r \bmod p$ for a random $r \xleftarrow{\$} \mathbb{Z}_q^*$ during the key generation. The signer stores (r, R) and then use them to compute signatures during the online phase, without any expensive operation. However, this approach incurs linear storage to the signer's side (i.e., one token per message).

It is highly desirable to construct a multiple-time signature scheme, which has constant signer storage and yet avoids expensive operations. However, this is a challenging task due to the nature of the aforementioned schemes. That is, in these schemes, the token R is directly used during the signature computation and therefore its storage cannot be trivially off-loaded to the verifier's side. This forces the signer either to store or to compute a token for each message.

A. Preliminary Scheme: Efficient and Tiny Authentication

In our preliminary work *Efficient and Tiny Authentication (ETA)* [42], we designed a signature scheme that can shift the storage of ephemeral public keys to the verifier's side without disrupting the security and verifiability of signatures. We outline our preliminary scheme `ETA` in Algorithm 2 for the sake of completeness.

In the following, we focus on our newly proposed `SEMECS` digital signature scheme and also highlight the differences between `ETA` and our improved scheme `SEMECS`.

B. Signer Efficient Multiple-time Elliptic Curve Signature (SEMECS)

We first discuss the challenges of eliminating ephemeral keys from the signature generation in Schnorr-like signatures, which is an important step to achieve signer optimal elliptic curve signatures. We then explain our strategies in `SEMECS` towards addressing these challenges.

Algorithm 2 Efficient and Tiny Authentication (`ETA`) Scheme

$(sk_0, PK, I) \leftarrow$ `ETA.Kg`$(1^\kappa, K)$:
1: $(y, Y, \langle q, p, \alpha\rangle) \leftarrow$ `Schnorr.Kg`(1^κ).
2: $r_0 \xleftarrow{\$} \mathbb{Z}_q^*$
3: **for** $j = 0, \ldots, K-1$ **do**
4: $\quad R_j \leftarrow \alpha^{r_j} \bmod p$.
5: $\quad r_{j+1} \leftarrow H(r_j)$.
6: $\quad$ Generate verification tokens as $v_j \leftarrow H(R_j)$.
7: **return** The private and public key, as $sk_0 \leftarrow (y, r_0)$ and $PK \leftarrow (Y, \overrightarrow{v} = v_0, \ldots, v_{K-1})$, respectively.

$\sigma_j \leftarrow$ `ETA.Sig`(sk_j, M_j): Given $sk_j = (y, r_j)$, compute signature σ_j on a message M_j as follows:
1: $x_j \xleftarrow{\$} \{0,1\}^\kappa$.
2: $e_j \leftarrow H(M_j||j||x_j)$.
3: $s_j \leftarrow r_j - e_j \cdot y \bmod q$.
4: The signature σ_j on M_j is $\sigma_j \leftarrow (s_j, x_j, j)$.
5: Update r_j as $r_{j+1} \leftarrow H(r_j)$, erase r_j (to save memory).
6: **if** $j > K-1$ **then return** $\perp$ (i.e., the limit on the number of signatures is exceed).
7: **else return** σ_j

$b \leftarrow$ `ETA.Ver`(PK, M_j, σ_j): If $j \geq K$ then return $b = 0$ and *abort*. Otherwise, continue as following:
1: $R'_j \leftarrow Y^{H(M_j||j||x_j)} \cdot \alpha^{s_j}$.
2: **if** $v_j = H(R'_j)$ **then return** $b = 1$
3: **else return** $b = 0$

1) Challenges of Removing Ephemeral Key from Signature Generation: In Schnorr-like signatures [11], [17], [18], an expensive operation is required to compute the ephemeral key $(R = \alpha^r \bmod p, r \xleftarrow{\$} \mathbb{Z}_q^*)$. This ephemeral key is an essential part of the signature generation and proof of security, and therefore it is a challenging task to remove it from signing without disrupting the security. For example, R is committed to the signature as $s \leftarrow r - H(M||R) \cdot y \bmod q$ in Schnorr signatures [13]. The ephemeral key enables programming of random oracle and also used in Forking Lemma [41] in the security proof of Schnorr-like signatures [17], [18].

2) Eliminating Expensive Operations from Signature Generation: We first pre-compute K ephemeral keys as $r_j \leftarrow H_0(y||j)$, $R_j \leftarrow \alpha^{r_j} \bmod p$ and store their hash commitments at the verifier as $\beta_j \leftarrow H_1(R_j)$ for $j = 0, \ldots, K-1$ (Steps 3-7 in Algorithm 3 `SEMECS.Kg`). This permits the derivation of r_i to be used in signature s_j deterministically without requiring any expensive operation, which will later to be verified by its corresponding β_j. Since R_j is not required in the signature generation, we avoid expensive operations, but only rely on a few hash calls and a single modular addition/multiplication.

Our next step is to ensure that the correctness and provable security are still achieved in the absence of the ephemeral key in the signature generation. In `ETA` [42], we mimicked the role of R_j in e_j by replacing it with a random number $x_j \leftarrow \{0,1\}^\kappa$ as $e_j \leftarrow H(M_j||j||x_j)$ (Steps 1-2 in Algorithm 2 `ETA.Sig`). However, this requires the explicit

Algorithm 3 Signer Efficient Multiple-time Elliptic Curve Signature (SEMECS) Scheme

$(sk_0, PK) \leftarrow$ SEMECS.Kg$(1^\kappa, K)$:

1: $(y, Y, \langle q, p, \alpha\rangle) \leftarrow$ Schnorr.Kg(1^κ).
2: **for** $j = 0, \ldots, K-1$ **do**
3: $\quad r_j \leftarrow H_0(y||j)$.
4: $\quad R_j \leftarrow \alpha^{r_j} \bmod p$.
5: $\quad z_j \leftarrow H_1(y||j)$.
6: $\quad \gamma_j \leftarrow z_j \oplus H_0(R_j)$.
7: $\quad \beta_j \leftarrow H_1(R_j)$.
8: **return** $sk_0 \leftarrow y$ and $PK \leftarrow (Y, \alpha, \overrightarrow{v} = (\langle\gamma_0, \beta_0\rangle, \ldots, \langle\gamma_{K-1}, \beta_{K-1}\rangle, K)$.

$\sigma \leftarrow$ SEMECS.Sig(sk_j, M_j): Given $sk_j = (y, j)$ compute the signature as follows:

1: **if** $|M_j| < |q|$ **then** set $(\overline{M}_j = M_j, \widetilde{M}_j = 0)$,
2: **else** split M_j into two as $(\overline{M}_j||\widetilde{M}_j)$ such that $|\overline{M}_j| = |q|$.
3: $r_j \leftarrow H_0(y||j)$.
4: $z_j \leftarrow H_1(y||j)$.
5: $c_j \leftarrow \overline{M}_j \oplus z_j$.
6: $e_j \leftarrow H_0(c_j||\widetilde{M}_j)$.
7: $s_j \leftarrow r_j - e_j \cdot y \bmod q$.
8: **if** $j > K-1$ **then return** $\perp$ (i.e., the limit on the number of signatures is exceeded).
9: **else return** The signature σ_j on M_j is $\sigma_j \leftarrow (s_j, c_j)$, where the sender transmits $(\sigma_j, \widetilde{M}_j)$ to the receivers.

$b \leftarrow$ SEMECS.Ver$(PK, \widetilde{M}_j, \sigma_j)$: If $|c_j| > |q|$ or $j \geq K$ then SEMECS.Ver return 0 and *aborts*. Otherwise, continue as following:

1: $R'_j \leftarrow Y^{H_0(c_j||\widetilde{M}_j)} \cdot \alpha^{s_j} \bmod p$.
2: **if** $\beta_j \neq H_1(R'_j)$ **then return** $b = 0$.
3: **else return** $b = 1$ and recover the message M_j as follows:
4: $\quad \overline{M}_j \leftarrow \gamma_j \oplus H_0(R'_j) \oplus c_j$.
5: $\quad$ **if** $\widetilde{M}_j = 0$ **then** set $M_j = \overline{M}_j$.
6: $\quad$ **else** set $M_j = (\overline{M}_j||\widetilde{M}_j)$.

transmission of an extra κ-bit randomness and therefore is not optimal in terms of signature size. Moreover, this random number must be generated online, so requires a strong random number generator to be present in a low-end device.

In the following (Section III-B3), we first outline how SEMECS improves the signature generation of ETA by reducing the private key and signature sizes. We then elaborate on how SEMECS achieves the correctness and a tight security reduction in Section III-B4.

3) Achieving Compact Key and Signature Sizes: Our idea is to embed randomness into the message itself by creating a "randomized message recovery" strategy, thereby avoiding an explicit transmission of randomness.

We first split message M_j into two pieces as $(\overline{M}_j||\widetilde{M}_j)$ such that $|\overline{M}_j| = |q|$ and $\widetilde{M}_j$ is the rest of message. If $|M_j| < |q|$ then we simply set $\overline{M}_j = M_j$ and $\widetilde{M}_j = 0$ (Steps 1-2 in Algorithm 3 SEMECS.Sig). We then deterministically derive $z_j \leftarrow H_1(y||j)$, generate a randomness as $c_j \leftarrow \overline{M}_j \oplus z_j$ and compute the hash of the message as $e_j \leftarrow H_0(c_j||\widetilde{M}_j)$. Finally, we compute $s_j \leftarrow r_j - e_j \cdot y \bmod q$ (Step 7 in Algorithm 3 SEMECS.Sig).

Our signature σ_j on M_j is $\sigma_j = (s_j, c_j)$, where the sender transmits $(\sigma_j, \widetilde{M}_j)$ to the receivers. Remark that, we only transmit c_j that carries $|q|$-bit part of the message since $c_j \leftarrow \overline{M}_j \oplus z_j$. Therefore, the only component of the signature that introduces cryptographic transmission overhead is $s_j \in \mathbb{Z}_q^*$, which is optimal for an elliptic curve based signature scheme[2]. This is as small as some of the most compact signatures (e.g., BLS [12]) but without requiring expensive operations at the signer's side. Morever, it is also smaller than SchnorrQ [15], [18] and ETA [42] that transmit e_j and x_j, respectively, as an extra information on top of s_j.

SEMECS achieves a small private key $y \in \mathbb{Z}_q^*$, which is identical to that of traditional Schnorr-like signatures [11], [17], [18] and only a half of the size that of ETA's private key. The small and constant private key size is achieved by generating the random values with a deterministic function (e.g., a hash function) just using a seed value (y). Therefore, the signer doesn't need to store all the random values generated at key generation, but only stores the seed and deterministically derives all random values from it (SEMECS.Sig Step 3-4). Moreover, unlike ETA, SEMECS signature generation does not require any fresh randomness and therefore avoids potential hurdles of weak pseudo-random number generators on the signer device [54], [55].

4) Signature Verification and Tight Security Reduction: The verifier first checks the range of randomness $c_j \in \mathbb{Z}_q^*$ and the limit on number of permitted signatures. The verifier then computes $R'_j \leftarrow Y^{H(c_j||\widetilde{M}_j)} \cdot \alpha^{s_j}$ and checks whether it matches with $\beta_j = H_1(R'_j) \in PK$. If it does not hold, the verifier returns $b = 0$. Otherwise, the verifier returns $b = 1$ and uses auxiliary value γ_j to recover the q-bit piece of message $\overline{M}_j$ from c_j as $\overline{M}_j \leftarrow \gamma_j \oplus H_0(R'_j) \oplus c_j$ forming the original message as $M_j = (\overline{M}_j||\widetilde{M}_j)$.

Note that the verifier should either know which public key component (β_j) it should use at SEMECS.Ver Step 2 or have a simple search operation among all βs to see if there is one that matches the calculated $H_1(R'_j)$. Therefore, there is a trade-off between a verifier computation and transmission overhead. However, both of these costs are almost negligible. Since j is a value up to K ($K = 2^{17}$ in our experiments), the transmission of it only incurs 2-3 Bytes of extra overhead. If the verifier computation is preferred, this only adds an overhead of a binary search operation, that has a complexity of $log_2(K)$. In the binary search option, we basically assume that the verifier stores the public key sorted, and after the value $H_1(R'_j)$ is calculated, binary search is made on sorted βs.

We now elaborate the design rationale behind the use of two separate verification tokens (β_j, γ_j) in SEMECS, as opposed to only one token v_j in ETA, for $j = 0, \ldots, K-1$.

(i) In Schnorr-like schemes, the randomness incorporated into message hashing is released with s_j but *not before*. This

[2] c_j does not offer confidentiality. After $\sigma_j = (s_j, c_j)$ is released, the message and z_j are publicly recovered to permit signature verification.

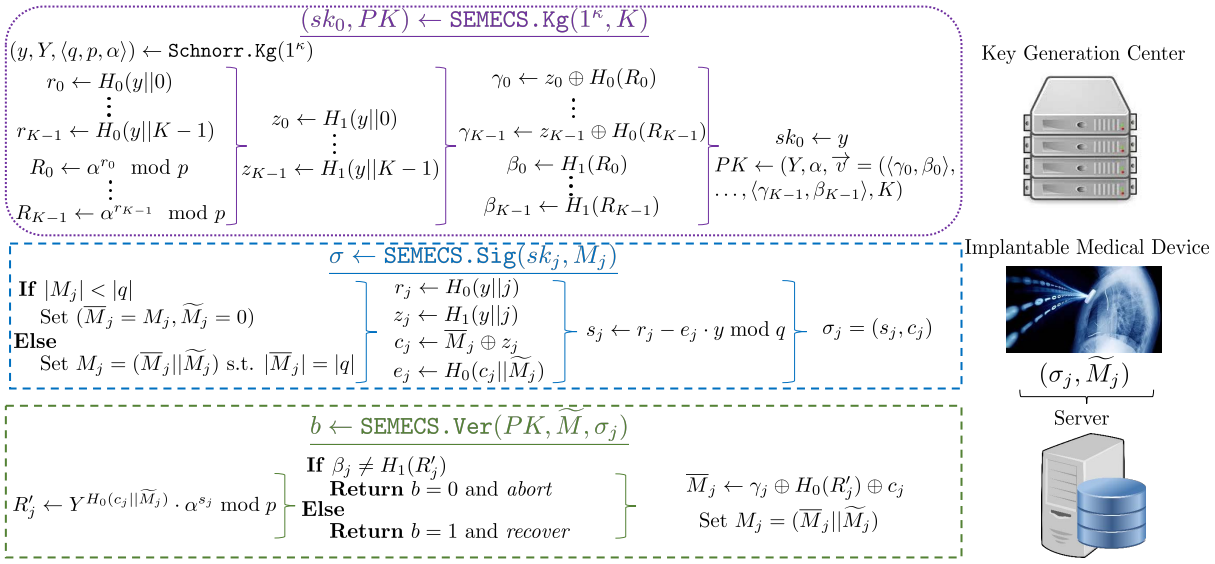


Fig. 1: High-level description of SEMECS algorithms.

is useful to construct an indistinguishable simulation in the security proof of Schnorr-like signatures[3]. In SEMECS, c_j that randomizes the message hash as $H_0(c_j||\widetilde{M}_j)$, is computed from z_j as $c_j \leftarrow \overline{M}_j \oplus z_j$. Our idea is to store z_j at the verifier's side as $\gamma_j \leftarrow z_j \oplus H_0(R_j)$ so that it can be recovered *only after* s_j is released. We avoid an online transmission of z_j but yet randomize the message hash via c_j including q-bit part of the message $\overline{M}_j$ (with no extra transmission overhead). After $\sigma_j = (c_j, s_j)$ is released, the verifier computes z_j from γ_j via $H_0(R_j)$. Note that $\beta_j = H_1(R_j)$ does not reveal z_j but yet permits the verification of $R'_j \leftarrow Y^{H_0(c_j||\widetilde{M}_j)} \cdot \alpha^{s_j} \mod p$.

(ii) In SEMECS, we present an improved security proof with a reduction to DLP with a much tighter bound compared to that of ETA. The security of ETA is reduced to Schnorr signatures, whose security proof relies on Forking Lemma [41]. Intuitively, if there is an adversary $\mathcal{A}$ making at most K' $RO(.)$ queries, and forging signatures with probability ϵ, then the Forking Lemma states that one can compute discrete logarithms with constant probability by rewinding the forger $O(K'/\epsilon)$ times. Therefore, the security reduction loses a factor $O(K')$ that can be very large [56].

Our key observation is that, since SEMECS is a K-time signature with pre-determined ephemeral public keys, we can avoid using Forking Lemma and obtain a reduction to DLP. That is, the hash of ephemeral keys are committed at the key generation phase as $\{\beta_j\}_{j=0}^{K-1} \in PK$. At the forgery phase, if $\mathcal{A}$ outputs a forgery on PK as (M^*, σ^*), where $\sigma^* = (s_j^*, c_j^*)$, $0 \leq j \leq K-1$, by validity condition, this forgery has to be on a $\beta_j \in PK$. Therefore, $\mathcal{F}$ can extract private key y *without* a need of rewinding $\mathcal{A}$. This permits us to avoid a large factor of $O(K')$ but only need a small constant factor $O(K)$ in our security reduction. We stress that this is possible due to special K-time nature of SEMECS, but it does *not* apply to polynomially unbounded Schnorr signature variants as proven in [56].

The detailed description of SEMECS is given at Algorithm 3 and further outlined in Figure 1.

IV. Security Analysis

We prove that SEMECS is a K-time *EU-CMA* signature scheme in Theorem 1 (in the random oracle model [40]). We ignore terms that are negligible in terms of κ.

Theorem 1 *$Adv^{EU\text{-}CMA}_{SEMECS}(t, K', K) \leq Adv^{DL}_{G}(t')$, where $t' = O(t) + (2K) \cdot O(\kappa^3) + (6K + K') \cdot RNG$.*

Proof: Let $\mathcal{A}$ be a SEMECS attacker. We construct a *DL-attacker* $\mathcal{F}$ that uses $\mathcal{A}$ as a sub-routine. That is, we set $(y' \xleftarrow{\$} \mathbb{Z}_q^*, Y' \leftarrow \alpha^{y'} \mod p)$ as defined in *DL-experiment* (i.e., Definition 3) and then run the simulator $\mathcal{F}$ by Definition 2 (i.e., *EU-CMA* experiment) as follows:

Algorithm $\mathcal{F}(Y')$

- *Setup:* $\mathcal{F}$ keeps three lists $\overrightarrow{\mathcal{M}}$, $\overrightarrow{\mathcal{L}}$, and $\overrightarrow{\mathcal{L}'}$, all initially empty. $\overrightarrow{\mathcal{M}}$ is a message list that records each M_j queried to ETA.Sig oracle. $\overrightarrow{\mathcal{L}}[j]$ and $\overrightarrow{\mathcal{L}'}[j]$ record (M_j, i) queried to $RO(.)$ oracle and its corresponding $RO(.)$ answer (h_j, i), respectively, for cryptographic hash functions $H_i, i \in \{0, 1\}$. $(h_j, i) \leftarrow \overrightarrow{\mathcal{L}'}[M_j, i]$ denotes the retrieval of $RO(.)$ oracle answer of (M_j, i) that has been queried before. If (M_j, i) has not been queried before then $\perp \leftarrow \overrightarrow{\mathcal{L}'}[M_j, i]$. $\mathcal{F}$ sets counters $(l \leftarrow 0, n \leftarrow 0)$ and continues as follows:
  - $h \leftarrow H\text{-}Sim(M, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'}, i)$: $\mathcal{F}$ implements a function $H\text{-}Sim$ to handle $RO(.)$ queries. That is, cryptographic functions $H_i, i \in \{0, 1\}$ are modeled as random oracles via $H\text{-}Sim$. If $\exists j : (M, i) = \overrightarrow{\mathcal{L}}[j]$ then $H\text{-}Sim$ returns $\overrightarrow{\mathcal{L}'}[j]$. Otherwise, it returns $h \xleftarrow{\$} \mathbb{Z}_q^*$ as the answer for given H_i, assigns $(\overrightarrow{\mathcal{L}}[l] \leftarrow (M, i),\ \overrightarrow{\mathcal{L}'}[l] \leftarrow (h, i))$ and $l \leftarrow l + 1$.

[3] In our security proof for SEMECS in Theorem 1, the simulator $\mathcal{F}$ programs random oracle $RO(.)$ such that the probability that adversary $\mathcal{A}$ querying $RO(.)$ on $c_j||M_j$ *before* querying it to the signature oracle $\texttt{SEMECS.Sig}_{sk}$ is as difficult as random guessing $c_j \in \mathbb{Z}_q^*$. Hence, the probability that simulator $\mathcal{F}$ aborts during the query phase is negligible in terms of κ (see success probability analysis in Theorem 1).

- $\mathcal{F}$ creates a simulated SEMECS public key PK as follows:
  - $Y \leftarrow Y'$,
  - For $j = 0, \ldots, K-1$,
    a) $(s_j, e_j, z_j) \xleftarrow{\$} \mathbb{Z}_q^*$
    b) $R_j \leftarrow Y^{e_j} \cdot \alpha^{s_j} \bmod p$
    c) $\gamma_j \leftarrow z_j \oplus H\text{-}Sim(R_j, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'}, 0)$
    d) $\beta_j \leftarrow H\text{-}Sim(R_j, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'}, 1)$
  - $PK \leftarrow (Y, \alpha, \overrightarrow{v} = (\langle\gamma_0, \beta_0\rangle, \ldots, \langle\gamma_{K-1}, \beta_{K-1}\rangle, K)$.
- *Execute* $(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.), \texttt{SEMECS.Sig}_{sk}(.)}(PK)$:
  - Queries: $\mathcal{F}$ handles $\mathcal{A}$'s queries as follows:
    (i) $\mathcal{A}$ queries $RO(.)$ on a message M for $H_i, i \in \{0, 1\}$. $\mathcal{F}$ returns $h \leftarrow H\text{-}Sim(M, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'}, i)$.
    (ii) $\mathcal{A}$ queries SEMECS.Sig oracle on a message M_n. If $n > K-1$ then $\mathcal{F}$ rejects the query (i.e., the query limit is exceeded). Otherwise, $\mathcal{F}$ continues as follows:
    a) If $|M_n| < |q|$ set $(\overline{M}_n = M_n, \widetilde{M}_n = 0)$, else split M_n into two as $(\overline{M}_n || \widetilde{M}_n)$ such that $|\overline{M}_n| = |q|$.
    b) $\mathcal{F}$ generates $c_n \leftarrow z_n \oplus \overline{M}_n$ and checks if $(c_n || M_n, 0) \in \overrightarrow{\mathcal{L}}$. If it holds then $\mathcal{F}$ *aborts* (i.e., the simulation fails). Otherwise, $\mathcal{F}$ continues as follows.
    c) $\mathcal{F}$ inserts $(\overrightarrow{\mathcal{L}}[l] \leftarrow (c_n || M_n, 0), \overrightarrow{\mathcal{L}'}[l] \leftarrow (e_n, 0))$.
    d) $\mathcal{F}$ returns $\sigma_n \leftarrow (s_n, c_n)$ to $\mathcal{A}$, sets $\overrightarrow{\mathcal{M}}[n] \leftarrow M_n$ and then increments $(n \leftarrow n+1, l \leftarrow l+1)$.
  - Forgery of $\mathcal{A}$: Eventually, $\mathcal{A}$ outputs a forgery on PK as (M^*, σ^*), where $\sigma^* = (s_j^*, c_j^*)$, $0 \le j \le K-1$. By definition 2, $\mathcal{A}$ wins the K-time $EU\text{-}CMA$ experiment for SEMECS if $\texttt{SEMECS.Ver}(PK, M^*, \sigma^*) = 1$ and $M^* \notin \overrightarrow{\mathcal{M}}$ hold. If these conditions hold, $\mathcal{A}$ returns 1, else, returns 0.
- Forgery of $\mathcal{F}$: If $\mathcal{A}$ loses in the K-time $EU\text{-}CMA$ experiment for SEMECS, $\mathcal{F}$ also loses in the DL experiment, and therefore $\mathcal{F}$ *aborts* and returns 0. Otherwise, if $(c_j^* || M^*) \in \overrightarrow{\mathcal{L}}$ then $\mathcal{F}$ *aborts* and returns 0 (i.e., $\mathcal{A}$ wins the experiment without querying $RO(.)$ oracle). Otherwise, $\mathcal{F}$ sets $s_j^* \leftarrow \overrightarrow{\mathcal{L}'}[c_j^* || M^*]$ and continues as follows:

Recall that $R_j \equiv Y^{e_j} \cdot \alpha^{s_j} \bmod p$ holds, where $0 \le j \le K-1$. Moreover, since $\texttt{SEMECS.Ver}(PK, M^*, \sigma^*) = 1$ holds, $R_j \equiv Y^{e_j{}^*} \cdot \alpha^{s_j^*} \bmod p$ also holds. Therefore, we write the following equations:

$$R_j \equiv Y^{e_j} \cdot \alpha^{s_j} \bmod p,$$
$$R_j \equiv Y^{e_j{}^*} \cdot \alpha^{s_j^*} \bmod p,$$

$\mathcal{F}$ then extracts $y' = y$ by solving the below modular linear equations (note that only unknowns are y and r_j), where $Y = Y'$ as defined in simulation:

$$r_j \equiv y' \cdot e_j + s_j \bmod q,$$
$$r_j \equiv y' \cdot e_j^* + s_j^* \bmod q,$$

Note that $Y' \equiv \alpha^{y'} \bmod p$ holds, since $\mathcal{A}$'s forgery is valid and non-trivial on $Y' = Y$. Therefore, by Definition 3, $\mathcal{F}$ wins the *DL-experiment*.

The execution time and probability analysis of the above experiment are as follows:

Execution Time Analysis: In this experiment, the running time of $\mathcal{F}$ is that of $\mathcal{A}$ plus the time it takes to respond q_H $RO(.)$ queries and K ETA.Sig.

- *Setup phase:* $\mathcal{F}$ draws $3K$ random numbers, performs $2K$ modular exponentiations, K XOR operations, and then invokes $RO(.)$ $2K$ times by drawing additional $2K$ random numbers. Hence, the total cost of this phase is $(2K) \cdot O(\kappa^3) + (5K) \cdot \texttt{RNG}$, where $O(\kappa^3)$ denotes the cost of modular exponentiation and RNG denotes the cost of drawing a random number. We omit the costs of XOR operations.
- *Query phase:* $\mathcal{F}$ draws K random numbers to handle $\mathcal{A}$'s ETA.Sig queries, whose cost is $K \cdot \texttt{RNG}$. $\mathcal{F}$ also draws K' random numbers to handle $\mathcal{A}$'s $RO(.)$ queries, whose cost is at most $K' \cdot \texttt{RNG}$.

Therefore, the approximate total running time of $\mathcal{F}$ is $t' = O(t) + (2K) \cdot O(\kappa^3) + (6K + K') \cdot \texttt{RNG}$.

Success Probability Analysis: $\mathcal{F}$ succeeds if all below events occur.

- $\overline{E1}$: $\mathcal{F}$ does not abort during the query phase.
- $E2$: $\mathcal{A}$ wins the K-time $EU\text{-}CMA$ experiment for SEMECS.
- $\overline{E3}$: $\mathcal{F}$ does not abort after $\mathcal{A}$'s forgery.
- Win: $\mathcal{F}$ wins the K-time $EU\text{-}CMA$ experiment for DL-*experiment*.
- $Pr[Win] = Pr[\overline{E1}] \cdot Pr[E2|\overline{E1}] \cdot Pr[\overline{E3}|\overline{E1} \wedge E2]$

• *The probability that event* $\overline{E1}$ *occurs*: During the query phase, $\mathcal{F}$ aborts if $(M_j || x_j) \in \overrightarrow{\mathcal{L}}$, $0 \le j \le K-1$ holds, *before* $\mathcal{F}$ inserts $(c_j || M_j)$ into $\overrightarrow{\mathcal{L}}$ (i.e., the simulation fails). This occurs if $\mathcal{A}$ guesses the randomized output c_j and then queries $(c_j || M_j)$ to $RO(.)$ *before* querying it to SEMECS.Sig. The probability that this occurs is $\frac{1}{2^{|q|}}$, which is negligible in terms of κ. Hence, $Pr[\overline{E1}] = (1 - \frac{1}{2^{|q|}}) \approx 1$.

• *The probability that event* $E2$ *occurs*: If $\mathcal{F}$ does not abort, $\mathcal{A}$ also does not abort since the simulated view of $\mathcal{A}$ is *indistinguishable* from the real view of $\mathcal{A}$ (see the indistinguishability analysis). Therefore, $Pr[E2|\overline{E1}] = Adv_{\texttt{SEMECS}}^{EU\text{-}CMA}(t, K', K)$.

• *The probability that event* $\overline{E3}$ *occurs*: $\mathcal{F}$ does not abort if the following conditions are satisfied:

i $\mathcal{A}$ wins the $EU\text{-}CMA$ experiment for SEMECS on a message M^* by querying it to $RO(.)$. The probability that $\mathcal{A}$ wins without querying M^* to $RO(.)$ is as difficult as a random guess.

ii After $\mathcal{F}$ extracts y' by solving modular linear equations, the probability that $Y' \not\equiv \alpha^{y'} \bmod p$ is negligible in terms κ, since $(Y = Y') \in PK$ and $\texttt{SEMECS.Ver}(PK, M^*, \sigma^*) = 1$. Hence, $Pr[\overline{E3}|\overline{E1} \wedge E2] = Adv_{\texttt{SEMECS}}^{EU\text{-}CMA}(t, K', K)$.

Omitting the terms that are negligible in terms of κ, the upper bound on $EU\text{-}CMA$-*advantage of* SEMECS is as follows:

$$Adv_{\texttt{ETA}}^{EU\text{-}CMA}(t, K', K) \le Adv_G^{DL}(t),$$

Indistinguishability Argument: The real-view of $\mathcal{A}$ is comprised of the public key $PK = (Y, \alpha, p, q, \overrightarrow{v} = (\langle\gamma_0, \beta_0\rangle, \ldots, \langle\gamma_{K-1}, \beta_{K-1}\rangle, K)$, the answers of $\texttt{SEMECS.Sig}_{sk}(.)$ as

TABLE II: Private/public key sizes, signature size and signature generation/verification costs of SEMECS and its counterparts

Scheme	Signer			Verifier	
	Private Key Size	Signature Size	Signature Generation	Public Key Size	Signature Verification
Full-time signatures					
SPHINCS [33]	n_S	$n_S(k_S(\lvert t_S \rvert - x_S + 1) + 2^{x_S})$	$(2t_S - 1) \cdot H$	n_S	$(k_S((\log t_S) - x_S + 1) + 2^{x_S} - 1) \cdot H$
ECDSA [11]	$\lvert q \rvert$	$2\lvert q \rvert$	$EMul$	$\lvert q \rvert$	$1.3 \cdot EMul$
Ed25519 [17]	$\lvert q \rvert$	$2\lvert q \rvert$	$EMul$	$\lvert q \rvert$	$1.3 \cdot EMul$
Kummer [16]	$\lvert q \rvert$	$2\lvert q \rvert$	$EMul$	$\lvert q \rvert$	$1.3 \cdot EMul$
SchnorrQ [18]	$\lvert q \rvert$	$2\lvert q \rvert$	$EMul$	$\lvert q \rvert$	$1.3 \cdot EMul$
K-time signatures					
HORS [26]	κ	$\kappa \cdot u$	$(u + 1) \cdot H$	$t \cdot \lvert H \rvert \cdot K$	$(u + 1) \cdot H$
HORSE [30]	$(\kappa \cdot t \cdot log_2(K))$	$\kappa \cdot u$	$(u \cdot log_2(K) + 1) \cdot H$	$t \cdot \lvert H \rvert$	$(u + 1) \cdot H$
XMSS [31]	κ	$(l + log_2(K)) \cdot \lvert H \rvert$	$(((log_2(K) + 2) \cdot (log_2(K) + l \cdot (w + 2)))/2 + 4 \cdot log_2(K)) \cdot H$	$(2(log_2(K) + log_2(l)) + 1) \cdot \lvert H \rvert$	$(log_2(K) + l \cdot (w + 1)) \cdot H$
Zaverucha et al. [28]	κ	$\kappa + \lvert q \rvert$	$(m/2) \cdot (Add_q + H)$	$m \cdot \lvert q \rvert \cdot K$	$1.3 \cdot EMul$
SEMECS	$\lvert q \rvert$	$\lvert q \rvert$	$2 \cdot H + Mul_q + Sub_q$	$(2K + 1) \cdot \lvert q \rvert$	$1.3 \cdot EMul$

K denotes the number of signatures that can be generated using a single key pair for K-time signature schemes.
$Emul$ and $Eadd$ denote the costs of EC scalar multiplication over modulus p', and EC addition over modulus p', respectively. ECDSA [11], Ed25519 [17], Kummer [16], and SchnorrQ [18] only differ from each other in terms of the underlying curve. The operations that are required are the same for these schemes. H and Mul_q denote a cryptographic hash and a modular multiplication over modulus q, respectively. We omit the constant number of negligible operations if there is an expensive operation (e.g., integer additions are omitted if there is an $Emul$). We use double-point scalar multiplication for verifications of ECC based schemes ($1.3 \cdot Emul$ instead of $2 \cdot Emul$ [9]). t_S, k_S and x_S are SPHINCS [33] parameters where t_S is the number of secret key elements, k_S is the number of revealed secret key elements and x_S is a small integer. SPHINCS [33] parameter n_S denotes the bit length of hashes. Zaverucha et al. [28] parameter m should be selected such that $\binom{m}{m/2} \geq 2^{2\kappa}$. Integers t and u denote the parameters used in HORS [26] and HORSE [30]. w is the Winternitz parameter and l is the tree parameter in XMSS [31].

Remark: For HORS [26] and Zaverucha et al. [28], similarly to SEMECS, we deterministically generate the necessary private key components from a seed (i.e., using a keyed hash) to have a small constant private key that can be deployed to low-end devices.

$\overrightarrow{\sigma} = (s_j, c_j)$ for $j = 0, \ldots, K-1$, and the answer of $RO(.)$ as $\overrightarrow{\mathcal{L}} = (h_0, \ldots, h_{K'-1})$ on corresponding $H_i, i \in \{0, 1\}$, respectively. That is, $\overrightarrow{\mathcal{A}}_{real} = \langle PK, \overrightarrow{\sigma}, \overrightarrow{\mathcal{L}} \rangle$, where all values are generated/computed by SEMECS algorithms as in the real system. All variables in $\overrightarrow{\mathcal{A}}$ are computed from the values $\{y, r_n, z_n, h_j, \alpha, p, q\}_{n=0,j=0}^{K-1,K'-1}$. Hence, the joint probability distribution of all other variables in $\overrightarrow{\mathcal{A}}$ are determined by the joint probability of these values. All these are random in $\mathbb{Z}_q^*$. Therefore, the joint probability distribution of $\overrightarrow{\mathcal{A}}$ is,

$$\begin{aligned} Pr[\overrightarrow{A}_{real} = \overrightarrow{a}] &= Pr[\overline{y} = y | \overline{r}_0 = r_0 \wedge, \ldots, \overline{h}_{K'-1} = h_{K'-1}] \\ &= \frac{1}{(q-1)^{1+2K+K} \cdot (p-1)^2} \end{aligned}$$

We denote the simulated view of $\mathcal{A}$ is as $\overrightarrow{A}_{sim}$, and it is equivalent to $\overrightarrow{A}_{real}$ except that in the simulation, values (s_j, e_j, z_j, c_j) for $j = 0, \ldots, K-1$ are randomly selected from $\mathbb{Z}_q^*$. Note that the joint probability distribution of these variables are identical to original signature and hash outputs (since hash function is modeled as RO). Hence, we write $Pr[\overrightarrow{A}_{real} = \overrightarrow{a}] = Pr[\overrightarrow{A}_{sim} = \overrightarrow{a}]$. □

V. Performance Analysis and Comparison

In this section, we first present the analytical analysis of SEMECS and its counterparts. Then, we present the results of our experiments on a commodity laptop and an 8-bit AVR embedded processor. Our evaluation metrics include key sizes, signature size, and computation costs. On the 8-bit microprocessor, we focus on the signer cost (i.e., private key, signature size, signature generation) since our system model includes resource-constrained devices as signers. For our counterparts, we consider state-of-the-art K-time signature schemes as well as some traditional (full-time) signatures.

Remark: Our envisioned applications require high signer efficiency to be practical on resource-constrained devices. Hence, *optimizing the online signer efficiency is the essential performance objective for SEMECS*. Recall that we assume verifiers are resourceful entities, which is a reasonable assumption for our envisioned applications (see Section I). Also note that in SEMECS system model, private/public keys are generated before the system deployment (see Section II-B). Hence, the key generation cost (i.e., *offline cost*) is not a critical performance metric for SEMECS.

A. Analytical Performance Analysis

Here, we describe the analytical costs of our scheme, where the online costs are summarized in Table II.

Key Generation: Key generation of SEMECS requires K EC scalar multiplications that is higher than its full-time counterparts. For instance, for EC-based signature schemes (e.g., ECDSA [11], Ed25519 [17], Kummer [16], and SchnorrQ [18]) keys are generated with only one EC scalar multiplication. However, it is comparable to its K-time counterparts as their key generation also depends on K. Note that in our system model, key distribution is performed before the deployment. Thus, we believe that this does not pose a limitation for our considered use-cases.

Signer Overhead: In SEMECS, signer stores a small private key that is the same size as its full-time elliptic curve counterparts. The private key of some K-time signatures can be deterministically derived from a κ-bit seed, which is $2\times$ smaller than that of SEMECS. However, this makes a small difference in practice (i.e., 16 Bytes vs 32 Bytes). Signature generation of SEMECS only requires 2 hash function calls, a single multiplication, and subtraction under $\bmod q$. This introduces a significantly smaller overhead compared to its alternatives. The counterparts of SEMECS either require expensive operations (i.e., EC scalar multiplication) or a very large number of hash function calls for signature generation. Only HORS [26] and Zaverucha et al. [28] have comparable signature generation speed. However, when we generate the

TABLE III: Experimental performance comparison of SEMECS and its counterparts on a commodity hardware

Scheme	K	Signature Generation Time (CPU cycle)	Private Key¶ (Byte)	Signature Size (Byte)	Signature Verification Time (CPU cycle)	Public Key‡	End-to-End Delay (CPU cycle)
Full-time signatures							
SPHINCS [33]	2^κ	37 466 005	1088	41000	1 051 562	1056	38 517 567
ECDSA [11]	2^κ	1 510 320	32	64	1 932 650	32	3 442 970
Ed25519 [17]	2^κ	146 620	32	64	286 750	32	433 370
Kummer [16]	2^κ	58 450	32	64	98 560	32	157 010
SchnorrQ [18]	2^κ	30 481	32	64	54 241	32	84 722
K-time signatures							
HORS [26]	1 2^{17}	16 823	16	384	8 975	32 KB 4 GB	25 798
HORSE [30]	1 2^{17}	16 823 280 247	16384 278 528	384	8 975	32 KB	25 798 287 503
XMSS [31]	1 2^{17}	137 856 1 367 431	16	2080 2592	115 239 120 983	416 1504	253 095 1 488 414
Zaverucha et al. [28]	1 2^{17}	89 180	16	48	52 872	4160 520 MB	142 052
SEMECS	1 2^{17}	**2 425**	**32**	**32**	52 872	96 8 MB	**55 297**

‡ The sizes are in terms of **Bytes**, if otherwise not specified.
¶ System wide parameters I (e.g., p,q,α) for each scheme are included in their corresponding codes, and private key size denote to specific private key size.
The cost of hash-based schemes are estimated based on the cost of a single hash operation.

private key components from a seed, these hash function calls dominate the signature generation cost for these schemes due to their large private key size.

Signature Transmission: Signature size of SEMECS is the smallest compared to its counterparts. Note that since the signature component c_j contains the information to recover the first $|q|$ Bytes of the message, we do not consider its transmission as an overhead. SEMECS only requires additional $|q|$ Bytes to be transmitted. Since the transmission of signatures introduces an overhead to the energy consumption of signer (and verifier), we believe it is essential to minimize its size.

Verifier Overhead: In SEMECS the public key is linear with the messages to be signed with a single key pair. Therefore, it increases as K increases (similar to its K-time counterparts except HORSE [30]). Considering that the verifier device is a resourceful device (e.g., server, command center) in SEMECS applications, we believe this is tolerable. The signature verification of SEMECS requires an EC double scalar multiplication (can be accelerated with Shamir's trick [9]).

Parameters: *We selected parameters to provide $\kappa = 128$-bit security for both SEMECS and its counterparts.* For elliptic-curve based schemes (including SEMECS), we selected $|q| =$ 256-bit. For Zaverucha et al., we selected $m = 260$, for HORS and HORSE, we selected $t = 1024$ and $u = 24$ to provide the desired security level. For XMSS and SPHINCS, we used the parameters suggested in the base papers. We refer the interested readers to the base papers of these schemes for the detailed explanation of their parameter choice.

B. Performance Evaluation on Commodity Laptop

We implemented SEMECS on a laptop and compared its cost to its state-of-the-art counterparts.

Hardware Configurations and Software Libraries: As our commodity hardware, we used a laptop equipped with an Intel i7 Skylake 2.6 GHz CPU with 12 GB RAM.

We implemented SEMECS on FourQ curve [15] to offer fast verification. We used the open-source implementation of this curve which can be found at[4]. We used our hash function as blake due to its high efficiency and high security [57]. Specifically, we used blake2s due to its optimization on low-end devices. We open-source our implementations at

www.github.com/ozgurozmen/SEMECS

We ran the open-sourced implementations of our counterparts on our hardware setting, if possible. For the hash-based constructions, we conservatively simulated their costs with blake2s hash function, to be fair with them.

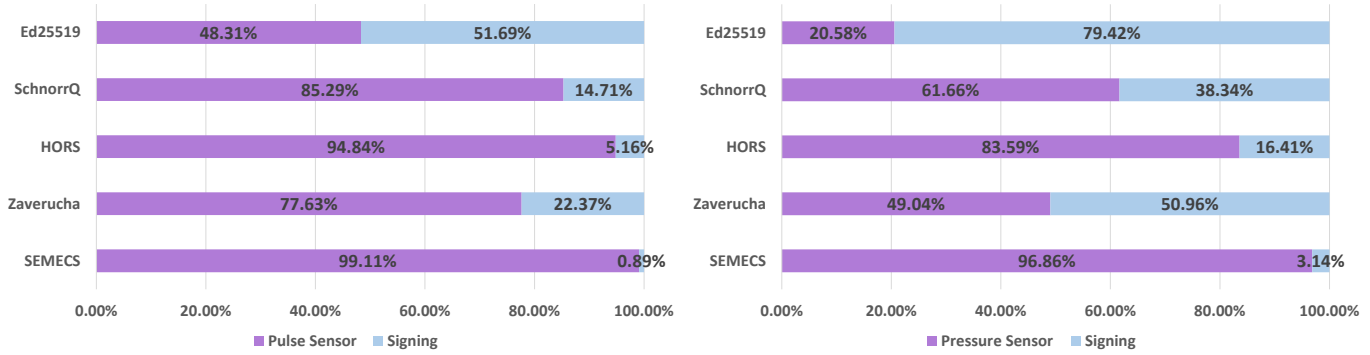
Experimental Results: Table III shows the benchmarks and specific key/signature sizes for SEMECS and its counterparts. We observe that SEMECS is 7× faster than its closest counterpart (HORS [26]) in terms of signature generation. Specifically, *it takes only* 1.23 *microseconds to generate a signature with SEMECS*. Moreover, it has a compact private key of 32 Bytes and the smallest signature (32 Bytes) among its counterparts. Signature verification of SEMECS is also fast since we used the optimized FourQ [15] curve to implement our scheme. Therefore, only HORS [26] and HORSE [30] offer faster verification. The main limitation of SEMECS is its public key size. Specifically, when $K = 2^{17}$, which *allows signing a message in every 20 minutes for 5 years without a key replacement*, the public key size is 8 MB. However, this is much smaller than some of the most efficient K-time counterparts such as HORS, and Zaverucha et al., that have 4 GB and 520 MB public key, respectively. We also implemented the key generation of SEMECS on this experimental setting and observed that generating the key for $K = 2^{17}$ takes 1.75 seconds.

C. Performance/Energy Evaluation on 8-bit Microprocessor

We fully implemented the signature generation of SEMECS on an 8-bit microprocessor to assess its energy and time efficiency on low-end embedded devices.

Hardware Configurations and Software Libraries: We used 8-bit AVR ATmega 2560 microprocessor to measure

[4]https://github.com/Microsoft/FourQlib



(a) Energy of Signature Generation vs Pulse Sensor

(b) Energy of Signature Generation vs Pressure Sensor

Fig. 2: Energy consumption of signature generation vs IoT sensors

the signer efficiency of our scheme compared to its counterparts. We selected this low-end device due to its low energy consumption and extensive use in practice, especially in IoT applications and medical devices [43], [58], [59]. It is equipped with 256 KB flash memory, 8 KB SRAM, 4 KB EEPROM, and its maximum clock frequency is 16 MHz.

We implemented SEMECS using Rhys Weatherley's crypto library [60] that offers high-speed operations for low-end devices. Specifically, we used its blake2s implementation and modified its reduction algorithm to compute $\bmod q$ (where q is FourQ parameter) using Barrett reduction. We also open-source our 8-bit implementations at the link given above to facilitate the test and broad adoption of SEMECS.

We used the results of our counterparts that were given in 8-bit AVR microprocessors, if possible. For instance, we used the results provided in [19] for Ed25519, [20] for μKummer and [21] for SchnorrQ. We ran the ECDSA implementation of microECC [61] on our hardware. Similar to the laptop implementation, we measured the cost of a single hash (blake2s) call on our microprocessor and conservatively estimated the hash-based schemes' cost.

Experimental Results: As summarized in Table I, our analysis confirmed that SEMECS is highly efficient at the signer's side. Signature generation of SEMECS is performed with less than 200 thousand cycles, which is 6$\times$ faster than HORS [26] and 19$\times$ faster than SchnorrQ (its fastest counterparts). In addition to this, SEMECS requires a small private key and a signature size that is the smallest among its counterparts. This makes SEMECS very desirable for applications that include resource-limited signers.

Energy Consumption Analysis: We analyzed the energy consumption of SEMECS and its counterparts on our experimental setting and compared with the energy consumption of two common IoT sensors (a pulse and a pressure sensor). We first derived a generic energy consumption estimation (as in [62] that offers an estimation for MICAz) for 8-bit AVR ATmega 2560 based on our SEMECS implementation and used it to estimate the energy consumption of our counterparts (similarly with [21] that uses [62]). We also calculated how much energy is required to operate IoT sensors. We took into consideration (i) energy drawn by the sensor (ii) energy drawn by the microprocessor to read data from the sensor and (iii) energy drawn by the microprocessor during the idle time.

We powered our microprocessor with a 2200 mAh power pack. This allowed us to use an ammeter/power meter connected between the battery and the microprocessor. We measured 5V of voltage and 20mA of current on load, which is verified by the datasheet of the processor[5]. Then, we used the formula $E = V \cdot I \cdot t$ to calculate the energy consumption in Joules (as in [63]). We also considered the potential deployment of nRF24L01 Single Chip 2.4 GHz Transceiver to 8-bit ATmega for signature transmission. Based on its datasheet, we also estimated the energy consumption of signature transmission with this low-power transceiver. Specifically, nRF24L01 operates at 3.3 V, 11.3 mA and support a transmission rate of 2Mbps. *Our results showed that 8-bit AVR ATmega 2560 consumes roughly* $6.25nJ$ *per cycle of computation and* $18.65nJ$ *per bit of transmission.*

We also calculated how much energy is necessary to operate IoT sensors. Specifically, we used a pulse sensor[6] (that could serve as an example of a medical sensor) and a BMP183 pressure sensor[7] (that could be an example of a daily IoT application). In our energy calculations, we considered a sampling frequency of 10 seconds for the pulse sensor and 10 minutes for the pressure sensor, due to the difference/urgency in their usage. Figure 2 shows how many percentage of the battery is spent on the IoT sensor, compared with that of the cryptographic operations (i.e., signing) of different schemes. One can observe that for pulse sensor (see Figure 2a), HORS and SchnorrQ require the 5.16% and 14.71%, whereas, with SEMECS, this is decreased to a negligible level (0.89%). For the pressure sensor, the energy consumption of SEMECS is only 3.14% where the closest counterpart is 16.41%. This shows that preferring SEMECS as the authentication mechanism in 8-bit AVR microprocessors significantly reduces the impact of cryptographic operations on battery life.

Based on this analysis, we noticed that SEMECS outperforms its counterparts for both computation energy and communication energy at the signer's side. We believe that this is essential in practice to extend the battery lives of critical embedded devices such as implantable medical devices.

[5]http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[6]https://pulsesensor.com/

[7]https://cdn-shop.adafruit.com/datasheets/1900_BMP183.pdf

VI. Conclusion

In this paper, we proposed a new signature scheme called SEMECS, which achieves several desirable properties that are critical for resource-constrained devices. Specifically, SEMECS only requires two hash, a modular multiplication, and a modular subtraction to compute a signature. Moreover, it has a constant-small private key and signature, that is optimal for an EC-based signature scheme. Our experiments on both laptop and 8-bit AVR confirmed the energy and computational efficiency of SEMECS. Therefore, we believe SEMECS is an ideal alternative for providing authentication and integrity services for resource-constrained devices.

Acknowledgments

This work is supported by the NSF CAREER Award CNS-1652389. We would like to thank Rouzbeh Behnia for his valuable comments.

References

[1] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Transactions on Services Computing*, vol. 3, no. 3, pp. 223–235, July 2010.

[2] I. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, May 2016.

[3] J. Leu, C. Chen, and K. Hsu, "Improving heterogeneous soa-based iot message stability by shortest processing time scheduling," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 575–585, Oct 2014.

[4] J. Lopez, "Unleashing public-key cryptography in wireless sensor networks," *Journal of Computer Security*, pp. 469–482, Sep. 2006.

[5] A. A. Yavuz, P. Ning, and M. K. Reiter, "Efficient, compromise resilient and append-only cryptographic schemes for secure audit logging," in *Proceedings of 2012 Financial Cryptography and Data Security (FC 2012)*, March 2012.

[6] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[7] A. Perrig and J. Tygar, *Secure broadcast communication in wired and wireless networks*. Kluwer Academic Publishers, 2003. [Online]. Available: http://books.google.com/books?id=h5qXzbliKNIC

[8] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[9] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[10] M. Mass, "Pairing-based cryptography," Master's thesis, Technische Universiteit Eindhoven, 2004.

[11] *ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American Bankers Association, 1999.

[12] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Springer Berlin Heidelberg, 2001, pp. 514–532.

[13] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[14] D. J. Bernstein, *Curve25519: New Diffie-Hellman Speed Records*. Springer Berlin Heidelberg, 2006, pp. 207–228. [Online]. Available: http://dx.doi.org/10.1007/11745853_14

[15] C. Costello and P. Longa, "Four $\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$ -curve over the mersenne prime," in *Advances in Cryptology – ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds. Springer Berlin Heidelberg, 2015, pp. 214–235.

[16] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe, "Kummer strikes back: New dh speed records," in *Advances in Cryptology – ASIACRYPT 2014*, P. Sarkar and T. Iwata, Eds. Springer Berlin Heidelberg, 2014, pp. 317–337.

[17] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Sep 2012. [Online]. Available: https://doi.org/10.1007/s13389-012-0027-1

[18] C. Costello and P. Longa, "Schnorrq: Schnorr signatures on fourq," MSR Tech Report, 2016. Available at: https://www. microsoft. com/en-us/research/wp-content/uploads/2016/07/SchnorrQ. pdf, Tech. Rep., 2016.

[19] M. Hutter and P. Schwabe, "Nacl on 8-bit avr microcontrollers," in *Progress in Cryptology – AFRICACRYPT 2013*, A. Youssef, A. Nitaj, and A. E. Hassanien, Eds. Springer Berlin Heidelberg, 2013, pp. 156–172.

[20] J. Renes, P. Schwabe, B. Smith, and L. Batina, "μkummer: Efficient hyperelliptic signatures and key exchange on microcontrollers," in *Cryptographic Hardware and Embedded Systems – CHES 2016*, B. Gierlichs and A. Y. Poschmann, Eds. Springer Berlin Heidelberg, 2016, pp. 301–320.

[21] Z. Liu, P. Longa, G. C. C. F. Pereira, O. Reparaz, and H. Seo, "Four$\mathbb{Q}$ on embedded devices with strong countermeasures against side-channel attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 665–686.

[22] D. Naccache, D. M'Raïhi, S. Vaudenay, and D. Raphaeli, "Can D.S.A. be improved? Complexity trade-offs with the digital signature standard," in *Proceedings of the 13th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '94)*, 1994, pp. 77–85.

[23] S. Even, O. Goldreich, and S. Micali, "Online/offline digital signatures," in *Proceedings on Advances in Cryptology (CRYPTO '89)*. Springer-Verlag, 1989, pp. 263–275.

[24] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '01. London, UK: Springer-Verlag, 2001, pp. 355–367.

[25] L. Lamport, "Constructing digital signatures from a one-way function," Tech. Rep. CSL-98, October 1979.

[26] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proceedings of the 7th Australian Conference on Information Security and Privacy (ACIPS '02)*. Springer-Verlag, 2002, pp. 144–153.

[27] Y. W. Law, Z. Gong, T. Luo, S. Marusic, and M. Palaniswami, "Comparative study of multicast authentication schemes with application to wide-area measurement system," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, ser. ASIA CCS '13. New York, NY, USA: ACM, 2013, pp. 287–298.

[28] G. Zaverucha and D. Stinson, "Short one-time signatures," Cryptology ePrint Archive, Report 2010/446, 2010, https://eprint.iacr.org/2010/446.

[29] J. Pieprzyk, H. Wang, and C. Xing, "Multiple-time signature schemes against adaptive chosen message attacks," in *Selected Areas in Cryptography (SAC)*, 2003, pp. 88–100.

[30] W. Neumann, "HORSE: An extension of an r-time signature scheme with fast signing and verification," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, vol. 1, april 2004, pp. 129 – 134 Vol.1.

[31] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme," RFC 8391, May 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8391.txt

[32] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt, "Time valid one-time signature for time-critical multicast data authentication," in *IEEE INFOCOM 2009*, April 2009, pp. 1233–1241.

[33] 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 *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Springer Berlin Heidelberg, 2015, pp. 368–397.

[34] A. Hülsing, J. Rijneveld, and P. Schwabe, "Armed SPHINCS - computing a 41 KB signature in 16 KB of RAM," in *Public-Key Cryptography - PKC 2016 - 19th IACR International Conference on Practice and Theory in Public-Key Cryptography*, March 2016, pp. 446–470.

[35] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal gaussians," in *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference. Proceedings, Part I*, R. Canetti and J. A. Garay, Eds. Springer Berlin Heidelberg, 2013, pp. 40–56.

[36] L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "Crystals – dilithium: Digital signatures from module lattices,"

Cryptology ePrint Archive, Report 2017/633, 2017, http://eprint.iacr.org/2017/633.
[37] W. Lee, Y.-S. Kim, Y.-W. Lee, and J.-S. No, "pqsigrm," Submission to the NIST's post-quantum cryptography standardization process, 2018, https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/pqsigRM.zip.
[38] N. T. Courtois, M. Finiasz, and N. Sendrier, "How to achieve a mceliece-based digital signature scheme," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Springer Berlin Heidelberg, 2001, pp. 157–174.
[39] R. Behnia, M. O. Ozmen, A. A. Yavuz, and M. Rosulek, "Tachyon: Fast signatures from compact knapsack," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 1855–1867.
[40] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and Communications Security (CCS '93)*. NY, USA: ACM, 1993, pp. 62–73.
[41] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 390–399. [Online]. Available: http://doi.acm.org/10.1145/1180405.1180453
[42] A. A. Yavuz, "Eta: Efficient and tiny and authentication for heterogeneous wireless systems," in *Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '13. New York, NY, USA: ACM, 2013, pp. 67–72. [Online]. Available: http://doi.acm.org/10.1145/2462096.2462108
[43] M. Rushanan, A. D. Rubin, D. F. Kune, and C. M. Swanson, "Sok: Security and privacy in implantable medical devices and body area networks," in *2014 IEEE Symposium on Security and Privacy*, May 2014, pp. 524–539.
[44] O. J. Rubio, J. D. Trigo, A. Alesanco, L. Serrano, and J. Garcia, "Analysis of iso/ieee 11073 built-in security and its potential ihe-based extensibility," *Journal of Biomedical Informatics*, vol. 60, pp. 270 – 285, 2016.
[45] C. Camara, P. Peris-Lopez, and J. E. Tapiador, "Security and privacy issues in implantable medical devices: A comprehensive survey," *Journal of Biomedical Informatics*, vol. 55, pp. 272 – 289, 2015.
[46] M. O. Ozmen and A. A. Yavuz, "Low-cost standard public key cryptography services for wireless iot systems," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, ser. IoTS&P '17. New York, NY, USA: ACM, 2017, pp. 65–70. [Online]. Available: http://doi.acm.org/10.1145/3139937.3139940
[47] N. Bui and M. Zorzi, "Health care applications: A solution based on the internet of things," in *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, ser. ISABEL '11. New York, NY, USA: ACM, 2011, pp. 131:1–131:5. [Online]. Available: http://doi.acm.org/10.1145/2093698.2093829
[48] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An internet of things–based personal device for diabetes therapy management in ambient assisted living (aal)," *Personal and Ubiquitous Computing*, vol. 15, no. 4, pp. 431–440, Apr 2011. [Online]. Available: https://doi.org/10.1007/s00779-010-0353-1
[49] M. Conti, R. D. Pietro, L. V. Mancini, and A. Mei, "Distributed detection of clone attacks in wireless sensor networks," *IEEE Trans. on Dependable Secure Compuation*, pp. 685–698, 2011.
[50] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: Secure and DoS-resistant code dissemination in wireless sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks*, ser. IPSN '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 445–456.
[51] A. A. Yavuz and P. Ning, "Self-sustaining, efficient and forward-secure cryptographic constructions for unattended wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1204–1220, 2012.
[52] C. Jean-Sébastien, "On the exact security of full domain hash," in *Advances in Crpytology (CRYPTO '00)*. Springer-Verlag, 2000, pp. 229–235.
[53] M. Bellare and P. Rogaway, "Introduction to modern cryptography," in *UCSD CSE Course*, 1st ed., 2005, p. 207, http://www.cs.ucsd.edu/~mihir/cse207/classnotes.html.
[54] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic attacks on pseudorandom number generators," in *Proceedings of the 5th International Workshop on Fast Software Encryption*, ser. FSE '98. London, UK, UK: Springer-Verlag, 1998, pp. 168–188. [Online]. Available: http://dl.acm.org/citation.cfm?id=647933.740748
[55] P. Q. Nguyen and I. E. Shparlinski, "The insecurity of the elliptic curve digital signature algorithm with partially known nonces," *Designs, Codes and Cryptography*, vol. 30, no. 2, pp. 201–217, Sep 2003.
[56] Y. Seurin, "On the exact security of schnorr-type signatures in the random oracle model," in *Advances in Cryptology – EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds. Springer Berlin Heidelberg, 2012, pp. 554–571.
[57] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal blake," Submission to NIST (Round 3), 2010. [Online]. Available: http://131002.net/blake/blake.pdf
[58] P. Szakacs-Simon, S. A. Moraru, and F. Neukart, "Signal conditioning techniques for health monitoring devices," in *2012 35th International Conference on Telecommunications and Signal Processing (TSP)*, July 2012, pp. 610–614.
[59] P. Szakacs-Simon, S. A. Moraru, and L. Perniu, "Pulse oximeter based monitoring system for people at risk," in *2012 IEEE 13th International Symposium on Computational Intelligence and Informatics (CINTI)*, Nov 2012, pp. 415–419.
[60] R. Weatherley, "Arduino cryptolibs," Github Repository, 2016. [Online]. Available: https://github.com/rweather/arduinolibs/tree/master/libraries/Crypto
[61] K. MacKay, "micro-ecc: Ecdh and ecdsa for 8-bit, 32-bit, and 64-bit processors," Github Repository. [Online]. Available: https://github.com/kmackay/micro-ecc
[62] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks*, ser. SASN '06. New York, NY, USA: ACM, 2006, pp. 169–176. [Online]. Available: http://doi.acm.org/10.1145/1180345.1180366
[63] G. Ateniese, G. Bianchi, A. Capossele, and C. Petrioli, "Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Precomputation Techniques?" in *Proceedings of the 20th Annual Network & Distributed System Security Symposium, NDSS 2013*, ser. NDSS2013, San Diego, CA, February 24-27 2013.

Attila Altay Yavuz (M '11) is an Assistant Professor in the Department of Computer Science and Engineering, University of South Florida (August 2018). He was an Assistant Professor in the School of Electrical Engineering and Computer Science, Oregon State University (2014-2018). He was a member of the security and privacy research group at the Robert Bosch Research and Technology Center North America (2011-2014). He received his PhD degree in Computer Science from North Carolina State University in August 2011. He received his MS degree in Computer Science from Bogazici University (2006) in Istanbul, Turkey. He is broadly interested in design, analysis and application of cryptographic tools and protocols to enhance the security of computer networks and systems. Attila Altay Yavuz is a recipient of NSF CAREER Award (2017). His research on privacy enhancing technologies (searchable encryption) and intra-vehicular network security are in the process of technology transfer with potential world-wide deployments. He has authored more than 40 research articles in top conferences and journals along with several patents. He is a member of IEEE and ACM.

Muslum Ozgur Ozmen received the bachelor's degree in electrical and electronics engineering from the Bilkent University, Turkey and the M.S. degree in computer science from Oregon State University. He is currently pursuing a PhD degree in computer science with the Department of Computer Science and Engineering, University of South Florida. His research interests include lightweight cryptography for IoT systems (drones and medical devices), digital signatures, privacy enhancing technologies (dynamic symmetric and public key based searchable encryption) and post-quantum cryptography.