Energy-Aware Digital Signatures for Embedded Medical Devices

Muslum Ozgur Ozmen
University of South Florida
Tampa, Florida, USA
ozmen@mail.usf.edu

Attila A. Yavuz
University of South Florida
Tampa, Florida, USA
attilaayavuz@usf.edu

Rouzbeh Behnia
University of South Florida
Tampa, Florida, USA
behnia@mail.usf.edu

***Abstract*—Authentication is vital for the Internet of Things (IoT) applications involving sensitive data (e.g., medical and financial systems). Digital signatures offer scalable authentication with non-repudiation and public verifiability, which are necessary for auditing and dispute resolution in such IoT applications. However, digital signatures have been shown to be highly costly for low-end IoT devices, especially when embedded devices (e.g., medical implants) must operate without a battery replacement for a long time.**

We propose an Energy-aware Signature for Embedded Medical devices (`ESEM`) that achieves near-optimal signer efficiency. `ESEM` signature generation does *not* require any costly operations (e.g., elliptic curve (EC) scalar multiplication/addition), but only a small constant-number of pseudo-random function calls, additions, and a single modular multiplication. `ESEM` has the smallest signature size among its EC-based counterparts with an identical private key size. We achieve this by eliminating the use of the ephemeral public key (i.e, commitment) in Schnorr-type signatures from the signing via a distributed construction at the verifier without interaction with the signer while permitting a constant-size public key. We proved that `ESEM` is secure (in random oracle model), and fully implemented it on an 8-bit AVR microcontroller that is commonly used in medical devices. Our experiments showed that `ESEM` achieves 8.4× higher energy efficiency over its closest counterpart while offering a smaller signature and code size. Hence, `ESEM` can be suitable for deployment on resource limited embedded devices in IoT. We open-sourced our software for public testing and wide-adoption.

***Index Terms*—Lightweight Authentication, Internet of Things, Digital Signatures, Embedded Devices.**

I. Introduction

It is essential to provide authentication and integrity services for the emerging Internet of Things (IoT) systems that include resource-constrained devices. Due to their computational efficiency, symmetric key primitives (e.g., message authentication codes) are usually preferred for such systems. On the other hand, these primitives might not be scalable for large and ubiquitous systems, and they also do not offer public verifiability and non-repudiation properties, which are essential for some IoT applications [1], [2], [3]. For instance, in financial IoT applications and implantable medical devices, digital forensics (e.g., legal cases) need non-repudiation and public verifiability [2], [3], [4]. Moreover, such systems may include many devices that require scalability.

Digital signatures rely on public key infrastructures and offer scalable authentication with non-repudiation and public verifiability. Therefore, they are ideal authentication tools for the security of IoT applications. On the other hand, most of the compact digital signatures (e.g., elliptic curve (EC) based signatures) require costly operations such as EC scalar multiplication and addition during signature generation. It has been shown [5], [6], [7], and further demonstrated by our experiments that, these operations can be energy costly, and therefore, can negatively impact the battery life of highly resource-limited embedded devices. For instance, as one of the many potential applications, we can refer to a resource-limited sensor (e.g., a medical device [1]) that frequently generates and signs sensitive data (medical readings), which are verified by a resourceful cloud service provider. There is a need for lightweight signatures that can meet the computation, memory and battery limitations of these IoT applications.

The goal of this paper is to devise an energy-aware and compact digital signature scheme that can meet some of the stringent battery and memory requirements of highly resource-limited IoTs (e.g., implantable medical devices) that must operate for long periods of time with minimal intervention.

Design Objectives: *(i)* The signature generation should not require any costly operation (e.g., exponentiation, EC operations), but only symmetric cryptographic functions (e.g., pseudorandom functions) and basic arithmetics (e.g., modular addition) *(ii)* The low-end devices are generally not only computation/battery but also memory limited. Hence, the objective (i) should be achieved without consorting precomputed storage (e.g., Boyko-Peinado-Venkatesan (BPV) tables [8], or online/offline signatures [9]). *(iii)* The signing should not draw new randomness [10] to avoid potential hurdles of weak pseudo-random number generators. *(iv)* The size of the signature should be small and constant-size as in Schnorr-like signatures. *(v)* The size of the public key should be constant.

Our Contributions: (i) We create an *Energy-aware Signature for Embedded Medical devices* (`ESEM`), which is ideal for the signature generation on highly resource-limited IoT devices. We observe that the realizations of Schnorr-like signatures on efficient elliptic curves (e.g., FourQ [11]) are currently the most efficient solutions, and a generation of the commitment value via a scalar multiplication is the main performance bottleneck in these schemes. Our main idea is to completely eliminate the generation, storage, and transmission of this

TABLE I: Signature generation performance of ESEM and its counterparts on AVR ATmega 2560 microcontroller

Scheme	CPU cycles	Signing Speed (ms)	Code Size (Byte)	Signature Size (Byte)	Private Key (Byte)	CPU energy (mJ)
ECDSA	79 185 664	4949	11 990	64	32	494.91
BPV-ECDSA	23 519 232	1470	27 912	64	10 272	146.99
Ed25519	34 342 230	2146	17 373	64	32	214.64
SchnorrQ	5 174 800	323	29 894	64	32	32.34
ESEM	**616 896**	**38**	18 465	**48**	32	**3.85**

We use low-are implementations due to the memory constraints of ATmega 2560. Note that ESEM does not store any precomputed components (e.g., BPV tables)

commitment from the signing of Schnorr signature. To achieve this, we first develop a new algorithm that we call as *Signer NOn-interactive Distributed BPV* (SNOD-BPV), which permits a distributed construction of the commitment for a given signature at verifier's side, without requiring any interaction with the signer. We then transform the signature generation process such that the correctness and provable security are preserved once the commitment value is separated from message hashing and SNOD-BPV is incorporated into ESEM. We present our proposed algorithms in Section III. In Section IV, we prove that ESEM is secure in the random oracle model [12] under a semi-honest distributed setting for SNOD-BPV.

(ii) We implemented ESEM and its counterparts both on an AVR ATmega 2560 microcontroller and a commodity hardware, and provided a detailed comparison in Section V. We also conducted experiments to assess the battery consumption of ESEM and its counterparts when they are used with common IoT sensors (e.g., a pulse and pressure sensor). We make our implementation open-source for broad testing and adoption.

Desirable Properties of ESEM: We summarize the desirable properties of our scheme as follows (Table I gives a comparison of ESEM with its counterparts in terms of signing efficiency on 8-bit AVR processor):

• *Signing and Energy Efficiency*: The signature generation of ESEM does not require any EC operations (e.g., scalar multiplication, addition) or exponentiation, but only pseudo-random function (PRF) calls, modular additions and a single modular multiplication. Therefore, ESEM achieves the lowest energy consumption among their counterparts. For example, ESEM consumes $8\times$ and $55\times$ less battery than SchnorrQ [11], and Ed25519 [13], respectively. Our experiments indicate that ESEM can substantially extend the battery life of low-end devices integrated with IoT applications (see Section V). Similarly, ESEM is at least a magnitude of times faster than Ed25519 both in an 8-bit microcontroller and commodity hardware. This gap further increases when our high-speed variant ESEM_2 (introduced in Section III) is considered.

• *Small Private Key and Signature Sizes:* ESEM has the smallest signature size among its counterparts (48 Bytes for $\kappa = 128$) with an identical private key size. ESEM does not require any precomputation tables to be stored at the signer, and therefore it is significantly more storage and computation efficient than schemes relying on BPV at the signer's side. Moreover, ESEM has a small code size at the signer since it only requires symmetric primitives and basic arithmetics.

• *High Security:* (i) Side-channel attacks exploiting the EC scalar multiplication implementations in ECDSA were proposed [14]. Since ESEM does not require any EC operations at the signer, it is not vulnerable to these types of attacks. (ii) The security of Schnorr-like signatures are sensitive to weak random number generators. The signing of ESEM does not consume new randomness (as in [10]), and therefore can avoid these problems. (iii) We prove that ESEM is $EU\text{-}CMA$ secure in the random oracle model [12].

Potential Use-cases: In many IoT applications, extending the battery life of low-end processors (i.e., usually signers) is a priority, while verifiers generally use a commodity hardware (e.g., a server) with reasonable storage and communication capabilities. In particular, energy efficiency is a vital concern for embedded medical devices, as they are expected to operate reliably for long periods of time. Currently, symmetric cryptography is preferred to provide security for such devices [15]. At the same time, the ability to produce publicly verifiable authentication tags with non-repudiation is desirable for medical systems [2], [3], [4] (e.g., digital forensics and legal cases). Moreover, scalable integration of various medical apparatus to IoT realm will receive a significant benefit from the ability to deploy digital signatures on these devices [1]. ESEM takes a step towards meeting this need, as it is currently the most energy efficient alternative with small signature and private key sizes. Essentially, any IoT application involving energy/resource limited signers and more capable verifiers (e.g., wireless sensor networks and IoT sensors in smart cities) are expected to receive benefit from ESEM.

Limitations: The signature verification of ESEM is distributed, wherein a verifier reconstructs the commitment value of a signature with l parties. Therefore, verification of ESEM is not real-time, and the verifier should wait for a response from all parties. However, as confirmed with our experiments, this only results in a few milliseconds of delay. Moreover, the signer does not need interaction with any parties to compute signatures. Parties aiding the verification are assumed to be semi-honest (do not deviate from the protocol, but try to learn information) and non-colluding (as in traditional semi-honest secure multi-party computation). In our case, even $(l-1)$ parties collude, ESEM remains EU-CMA secure. Since ESEM is designed for a near-optimal signer performance, we believe that ESEM is suitable for applications as outlined above, where a small delay and interaction can be tolerated at the verifier.

II. Preliminaries and Models

We first give the notations and definitions used by our schemes, and then describe our system/security model.

A. Notation and Definitions

Notation: $||$ and $|x|$ denote concatenation and the bit length of variable x, respectively. $x \xleftarrow{\$} \mathcal{S}$ means variable x is randomly selected from set $\mathcal{S}$. $|\mathcal{S}|$ denotes the cardinality of set $\mathcal{S}$. We denote by $\{0,1\}^*$ the set of binary strings of any finite length. The set of items q_i for $i = 0, \ldots, n-1$ is denoted by $\{q_i\}_{i=0}^{n-1}$. $\log x$ denotes $\log_2 x$. $\mathcal{A}^{\mathcal{O}_0,\ldots,\mathcal{O}_i}(.)$ denotes algorithm $\mathcal{A}$ is provided with oracles $\mathcal{O}_0, \ldots, \mathcal{O}_i$. For example, $\mathcal{A}^{SGN.Sig_{sk}}(.)$ denotes algorithm $\mathcal{A}$ is provided with a *signing oracle* of algorithm Sig of signature scheme SGN under a private key sk. We define a pseudo-random function (PRF) and three hash functions to be used in our schemes as follows: $PRF_0 : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, $H_0 : \{0,1\}^* \rightarrow \{0,1\}^\kappa$, $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{v \cdot \log n}$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, where $1 < v < n$ are BPV parameters and κ is the security parameter.

Definition 1. *A signature scheme SGN is a tuple of three algorithms (Kg, Sig, Ver) defined as follows:*

- *$(sk, PK) \leftarrow SGN.Kg(1^\kappa)$: Given the security parameter 1^κ, the key generation algorithm returns a private/public key pair (sk, PK).*
- *$\sigma \leftarrow SGN.Sig(m, sk)$: The signing algorithm takes a message m and a sk, and returns a signature σ.*
- *$b \leftarrow SGN.Ver(m, \sigma, PK)$: The verification algorithm takes a message m, signature σ and the public key PK as input. It returns a bit b: 1 means* valid *and 0 means* invalid.

Our schemes are based on Schnorr signature [16].

Definition 2. *Schnorr signature scheme is a tuple of three algorithms (Kg, Sig, Ver) defined as follows:*

- *$(y, Y) \leftarrow$ Schnorr.Kg(1^κ): Given 1^κ as the input,*
  1) *The system-wide params $\leftarrow (q, p, \alpha)$, where q and p are large primes such that $p > q$ and $q|(p-1)$, and a generator α of the subgroup G of order q in $\mathbb{Z}_p^*$.*
  2) *Generate private/public key pair $(y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow \alpha^y \bmod p)$. We suppress params afterwards for the brevity.*
- *$(\sigma \leftarrow$ Schnorr.Sig(m, y): Given $m \in \{0,1\}^*$ and y as the input, it returns a signature $\sigma = (s, e)$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is a full domain hash function.*
  1) *$r \xleftarrow{\$} \mathbb{Z}_q^*, \quad R \leftarrow \alpha^r \bmod p$.*
  2) *$e \leftarrow H(m||R), \ s \leftarrow (r - e \cdot y) \bmod q$.*
- *$b \leftarrow$ Schnorr.Ver$(m, \langle s, e\rangle, Y)$: The signature verification algorithm takes m, $\langle s, e\rangle$ and Y as the input. It computes $R' \leftarrow Y^e \alpha^s \bmod p$ and returns a bit b, with $b = 1$ indicating* valid*, if $e = H(m||R')$ and $b = 0$ otherwise.*

We use Boyko-Peinado-Venkatesan (BPV) generator [8].

Definition 3. *The BPV generator is a tuple of two algorithms $(Offline, Online)$ defined as follows:*

- *$(\Gamma, v, n, q, p) \leftarrow BPV.Offline(1^\kappa)$: The offline BPV algorithm takes 1^κ as the input and generate system-wide parameters (q, p, α) as in Schnorr.Kg(1^κ).*
  1) *BPV parameters n and v are the number of pairs to be precomputed and the number of elements to be randomly selected out n pairs, respectively, for $2 < v < n$.*
  2) *$r_i \xleftarrow{\$} \mathbb{Z}_q^*, \quad R_i \leftarrow \alpha^{r_i} \bmod p, \ i = 0, \ldots, n-1$.*
  3) *Set precomputation table $\Gamma = \{r_i, R_i\}_{i=0}^{n-1}$.*
- *$(r, R) \leftarrow BPV.Online(\Gamma, v, n, q)$: The online BPV algorithm takes the table Γ and (v, n, q) as input.*
  1) *Generate a random set $S \subset [0, n-1]$, where $|S| = v$.*
  2) *$r \leftarrow \sum_{i \in S} r_i \bmod q, \ R \leftarrow \prod_{i \in S} R_i$.*

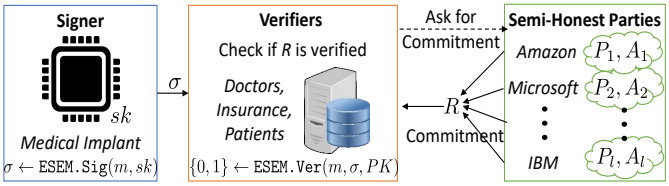


Fig. 1: System Model of ESEM

Lemma 1. *The distribution of BPV output r is* statistically close *to the uniform random distribution with an appropriate choice of parameters (v, n) [8].*

B. System and Security Model

As depicted in Figure 1, our system model includes a highly resource-limited signer that computes signatures to be verified by any receiver. Our system model also includes l distinct parties $(P_1, \ldots, P_l)$ that are involved in signature verification. In the line of [17], after the initialization phase, we consider a synchronous network which consists of a client (verifier in ESEM) and semi-honest servers $P = (P_1, \ldots, P_l)$. We assume that the communication channels are secure.

The security notion for a digital signature is Existential Unforgeability against Chosen Message Attack ($EU\text{-}CMA$) [18].

Definition 4. *EU-CMA experiment $Expt_{\text{SGN}}^{EU\text{-}CMA}$ for a signature scheme $SGN = (Kg, Sig, Ver)$ is defined as follows.*

- $(sk, pk) \leftarrow \text{SGN.Kg}(1^\kappa)$
- $(M^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SGN.Sig}(\cdot)}(pk)$

$\mathcal{A}$ wins the above experiment if $1 \leftarrow SGN.Ver(M^*, \sigma^*, pk)$ and m^* was not queried to $SGN.Sig(\cdot)$ oracle. The EMU-CMA advantage $Adv_{SGN}^{EU\text{-}CMA}$ of $\mathcal{A}$ is defined as $\Pr[Expt_{SGN}^{EU\text{-}CMA} = 1]$

Definition 5. *A protocol is t-private [17] if any set of parties $\mathcal{S}$ with $|\mathcal{S}| \leq t$ are not able to compute or achieve any output or knowledge any different than what they could have done individually from their set of private input and outputs.*

Assumption 1. *We assume that the servers are* semi-honest - *always follow the protocol, but try to learn as much as possible from the shared or observed information.*

For $t = l - 1$, where l is the total number of the servers, our proposed scheme is t-private. The signature generation in our scheme does not require the participation of the servers. In other words, the signer does not need to interact with any of the l servers during the signature generation. The participation of all l servers is however required on the verifier's side.

III. Proposed Schemes

We first discuss the design challenges to achieve our objectives outlined in Section I. We elaborate our Signer NOn-interactive Distributed BPV (SNOD-BPV) algorithm that addresses some of them. We then present ESEM that uses SNOD-BPV and other strategies to achieve our objectives.

A. High-Level Design

Schnorr-like signatures with implementations on recent ECs (e.g., FourQ [11]) are currently among the most efficient and compact digital signatures. Hence, we take them as our starting point. In these schemes, the signer generates a random value r and its commitment $R = \alpha^r \bmod p$, which is incorporated into both signing and verification (as an input to hash along with a message). This exponentiation (EC scalar multiplication) constitutes the main cost of the signature generation, and therefore we aim to *completely* eliminate it from the signing. However, this is a highly challenging task.

1) Commitment Generation without Signer Interaction: The elimination of commitment R from the signing permits removal of EC operations such as scalar multiplication/additions. It also eliminates the transmission of R and a storage of BPV table at the signer. However, the commitment is necessary for the signature verification. Hence, the verifier should obtain a correct commitment for each signature with the following requirements: (i) The verifier cannot interact with the signer to obtain the commitment (i.e., the signer does not have it). (ii) The signer non-interactive construction of the commitment should not reveal the ephemeral randomness r. (iii) Unlike some multiple-time signatures [6], the verifier should not have a fixed limit on the number of signature verifications and/or a linear-size public key.

We propose a new algorithm that we refer to as SNOD-BPV, to achieve these requirements. Our idea is to create a distributed BPV technique that permits a set of parties to construct a commitment on behalf of the signer. This distributed scheme permits the verifiers to obtain the corresponding commitment of a signature from these parties on demand without revealing r or an interaction with the signer. We elaborate on SNOD-BPV in Section III-B.

2) Separation of the Commitment from Signature Generation with SNOD-BPV: The commitment value is generally used as a part of message hashing (e.g., $H(M||R)$ in Schnorr) in Schnorr-like signatures. To eliminate R from the signature, the commitment must be separated from the message. However, the use of commitment in the message hashing plays a role in the security analysis of Schnorr-like signatures. Moreover, the removal of commitment R from the signing while using r with SNOD-BPV algorithm requires a design adjustment.

We propose our main scheme ESEM that achieves these goals. In the line of [6], we use a one-time random value x in the message hashing, but also devise an index encoding and aggregate BPV approach to integrate SNOD-BPV into signature generation. This permits a constant-size public key at the verifier without any interaction with the signer. We give the details of ESEM in Algorithm 2.

B. Signer NOn-interactive Distributed BPV (SNOD-BPV)

We conceive SNOD-BPV as a distributed realization of BPV [8] where l parties hold n public values $\{R_{i,j}\}_{i=1,j=1}^{n,l}$ of l BPV tables, and then can collaboratively derive R without learning its corresponding private key r unless all of them collude. We stress that one cannot simply shift the storage of public values in a BPV table to a single verifier. This is because the indexes needed to compute the commitment R should remain hidden in order to protect the one-time randomness r [19]. We overcome this challenge by creating a distributed BPV approach that can be integrated into a Schnorr-like signature. At SNOD-BPV.Offline, in Step 2, the secret key y is used as a seed to derive l secret values $\{z_j\}_{j=1}^{l}$. Each z_j is used to deterministically generate secret BPV values $\{r_{i,j}\}_{i=1,j=1}^{n,l}$, whose corresponding public values $\{R_{i,j}\}_{i-1,j=1}^{n,l}$ are computed in Step 4-5 and given to parties $(P_1, \ldots, P_l)$.

At the online phase, the sender (i.e., signer) generates the aggregated r on its own and the receiver (i.e., verifier) generates the aggregated R cooperatively with the parties $(P_1, \ldots, P_l)$. The sender first derives a random value x from a keyed hash function (at SNOD-BPV.Sender Step 1), and then deterministically derives z_j values (Step 3) as in SNOD-BPV.Offline. Sender uses the z_j, that is only shared with the corresponding party, and the one-time random value x to generate the set (indexes) to be used to aggregate the values. This step is of high importance since this way, the sender *commits* to the one-time random value x. Sender repeats this process for all l parties and aggregates (adds) all the corresponding $r_{i,j}$s to derive the resulting r (Step 5).

The verifier proceeds as follows to generate the corresponding $R = \alpha^r \bmod p$. At Step 1 in SNOD-BPV.Receiver, the verifier communicates with l parties to derive each R_j from them. Upon request, parties first derive the same set (indexes) as the sender (Step 1 in SNOD.P$_j$-Construct). Then, each party aggregates the corresponding $R_{i,j}$ that were assigned to them in SNOD-BPV.Offline, and returns the results to the verifier. The verifier aggregates all these values at Step 2, to derive the corresponding R. Please note that only the parties can create the set (indexes) since only they have their corresponding z_j values. Moreover, since all servers provide R_j that can be generated only by them, unless all of the servers collude, they cannot learn any information about the other indexes or the one-time randomness r. This makes our scheme t-private, as shown in Lemma 2.

C. Energy-aware Signature for Embedded Medical devices

We summarize our main scheme ESEM (see Algorithm 2), which permits a near-optimal signing by integrating SNOD-BPV into Schnorr signature with alternations.

During key generation, secret/public key pair (y, Y) and BPV parameters are generated (Step 1-2), followed by SNOD-BPV.Offline$(\cdot)$ algorithm to obtain the distributed BPV public values to be stored by parties $(P_1, \ldots, P_l)$. In ESEM.Sig$(\cdot)$, the signer generates the ephemeral random

Algorithm 1 Signer NOn-interactive Distributed BPV (SNOD-BPV)

$(A_1, \ldots, A_l) \leftarrow$ SNOD-BPV.Offline$(1^\kappa, y, v, n)$:
Given 1^κ, secret key y, and parameters (v, n) generate precomputation tables.

1: **for** $j = 1, \ldots, l$ **do**
2: $z_j \leftarrow \mathrm{PRF}_0(y||j)$
3: **for** $i = 1, \ldots, n$ **do**
4: $r_{i,j} \leftarrow \mathrm{PRF}_0(z_j||i)$
5: $R_{i,j} \leftarrow \alpha^{r_{i,j}} \bmod p$
6: Set $A_j = (z_j, v, \langle R_{1,j}, \ldots, R_{n,j}\rangle)$
7: **return** each A_j to corresponding party P_j

$(r, x) \leftarrow$ SNOD-BPV.Sender(sk):
1: $x \leftarrow H_0(sk||c)$, $c \leftarrow c + 1$
2: **for** $j = 1, \ldots, l$ **do**
3: $z_j \leftarrow \mathrm{PRF}_0(sk||j)$
4: $(i_{1,j}, \ldots, i_{v,j}) \leftarrow H_1(z_j||x)$
5: $r \leftarrow \sum_{k=1}^{v} \sum_{j=1}^{l} \mathrm{PRF}_0(z_j||i_{k,j}) \bmod q$
6: **return** r

$\bar{R}_j \leftarrow$ SNOD.P$_j$-Construct(A_j, x): Given x and A_j, each party P_j returns $\bar{R}_j$.
1: $(i_{1,j}, \ldots, i_{v,j}) \leftarrow H_1(z_j||x)$
2: $\bar{R}_j \leftarrow \prod_{k=1}^{v} R_{i_{k,j},j} \bmod p$

$R \leftarrow$ SNOD-BPV.Receiver(x) Given x, retrieve its commitment R under PK from $(P_1, \ldots, P_l)$.
1: Verifier sends x to the parties, and each party P_j returns $\bar{R}_j \leftarrow$ SNOD.P$_j$-Construct(A_j, x) for $j = 1, \ldots, l$.
2: $R \leftarrow \prod_{j=1}^{l} \bar{R}_j \bmod p$
3: **return** R

value r and one-time randomness x to be used as the commitment. Instead of the commitment in Schnorr (R), the signer uses x as the commitment in Step 2. This separation of the commitment R from the message hashing is inspired from [6]. Note that, unlike the multiple-time signature in [6] that can only compute a constant pre-determined number of signatures with a very large linear-size public key, ESEM can compute polynomially unbounded number of signatures with a constant public key size. Finally, the verifier first calls the SNOD-BPV.Receiver$(\cdot)$ algorithm to generate the public value R, by collaborating with the parties. The signature verification, which is similar to Schnorr with the exception of the commitment x, is performed at Step 2.

1) ESEM$_2$: We point out a trade-off between the private key size and signing speed, which can increase the signature generation performance with the cost of some storage. The signer can store private keys $\{r_{i,j}\}_{i=1,j=1}^{n,l}$ in the memory, and therefore avoid $v \cdot l$ PRF invocations. *We refer to this simple variant as ESEM$_2$*. As demonstrated in Section V, an extra storage of 12 KB can boost the performance of ESEM commodity hardware.

Algorithm 2 Energy-aware Signature for Embedded Medical devices (ESEM)

$(sk, PK, A_1, \ldots, A_l) \leftarrow$ ESEM.Kg(1^κ):
1: $(y, Y) \leftarrow$ Schnorr.Kg(1^κ), where (q, p, α) as in Schnorr.Kg.
2: Select (v, n) such that $\binom{n}{v} \geq 2^\kappa$
3: $(A_1, \ldots, A_l) \leftarrow$ SNOD-BPV.Offline$(1^\kappa, y, v, n)$
4: The signer stores $sk = (y)$ and $(v, n, q, c = 0)$. The verifier stores $PK = Y$ and (v, n, q, p, α).

$\sigma \leftarrow$ ESEM.Sig(m, sk):
1: $(r, x) \leftarrow$ SNOD-BPV.Sender(y)
2: $s \leftarrow r - H_2(m||x) \cdot y \bmod q$
3: **return** $\sigma = (s, x)$.

$\{0, 1\} \leftarrow$ ESEM.Ver(m, σ, PK):
1: $R \leftarrow$ SNOD-BPV.Receiver(x)
2: **if** $R = Y^{H_2(m||x)} \cdot \alpha^s \bmod p$, **return** 1, else **return** 0.

IV. Security Analysis

Lemma 2. *The scheme proposed in Algorithm 1 is t-private (in the sense of Definition 5) with regard to r and therefore, it can resist against $l - 1$ colluding servers.*

Proof. The random values $\{r_{i,j} \leftarrow \mathrm{PRF}_0(z_j||i)\}_{j=1,i=1}^{l,n}$ are generated uniformly at random in the SNOD-BPV.Offline$(\cdot)$ via private seed z_j, which is given to each server P_j. The security of SNOD-BPV (i.e., ESEM) relies on the secrecy of $r \leftarrow \sum_{i=1}^{l} \sum_{j=1}^{n} r_{i,j} \bmod q$. Given each $r_{i,j}$ is generated uniformly at random via z_j's, and due to Lemma 1, for the adversary $\mathcal{A}$to infer r, it must know all l private seeds z_j or corrupt all of the l servers. □

Theorem 1. *In the random oracle model, based on Assumption 1 and Lemma 2, if a polynomial-time adversary $\mathcal{A}$can break the EU-CMA security of ESEM in time t and after q_h hash and q_s signature queries, then one can build polynomial-time algorithm $\mathcal{F}$that breaks the EU-CMA security of Schnorr signature in time t' and q'_s signature queries.*

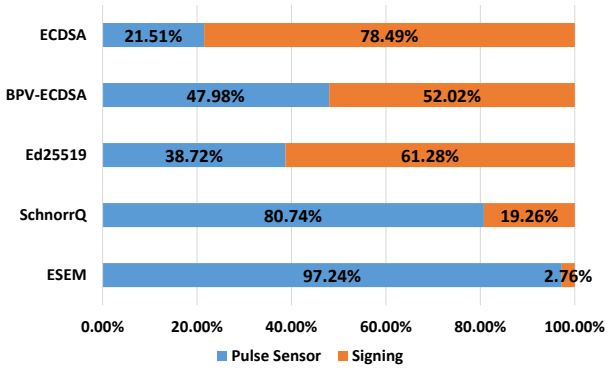
$$Adv_{ESEM}^{EU\text{-}CMA}(t, q_h, q_s) \quad \leq \quad Adv_{Schnorr}^{EU\text{-}CMA}(t', q'_s),$$

Proof: Please refer to the Appendix.

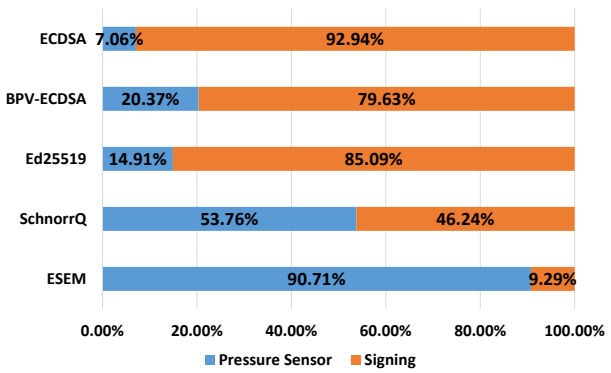
V. Performance Analysis

A. Parameter Selection

We select FourQ curve [11] that offers fast elliptic curve operations (that is desirable for our verification process, remark that signer has no EC operations) with 128-bit security level. The selection of parameters (v, n) relies on the number of v-out-of-n different combinations possible. We select $n = 1024$, $v = 18$ for ESEM and $n = 128$, and $v = 40$ for ESEM$_2$, where both offers over 2^{128} different combinations. Lastly, we select $l = 3$ (i.e., 3 parties are involved in verification).



(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

B. Evaluation Metrics and Experimental Setup

Evaluation Metrics: We implemented ESEM and its counterparts both on the low-end device (8-bit microcontroller) and a commodity hardware. (i) At the signer's side, the signature generation time and private key size were evaluated on both types of devices. The energy consumption and code size were evaluated on a low-end device. (ii) The signature size is evaluated as the communication overhead. (iii) At the verifier's side, the signature verification time and the size of public key were evaluated on the commodity hardware.

Note that the time required to transmit the ESEM signature (only 48 Bytes) is already smaller than all of its counterparts. Therefore, we do not include this in our experiments. The bandwidth overhead to construct R between the verifier and l parties is only 48 Bytes, and highly depends on the geographic location of the server (i.e., round trip time). We conservatively benchmark this network delay and include in our signature verification time, with an Amazon EC2 server in North Virginia.

Hardware Configurations and Software Libraries: We selected AVR ATmega 2560 microcontroller as our low-end device due to its low power consumption and extensive use in practice, especially for medical devices [1], [2], [20]. It is an 8-bit microcontroller with 256 KB flash memory, 8 KB SRAM, 4 KB EEPROM and maximum clock speed is 16 MHz.

We implemented our schemes using Rhys Weatherley's library[1], which enables Barrett reduction to compute modulo q. We used BLAKE2s [21] as our hash function from the same library, since it is optimized for low-end devices in terms of speed and code size. We instantiated our PRF function as CHACHA20 stream cipher [22] which offers high efficiency. To assess our counterparts, we used ECDSA implementation in microECC[2], with which we also implemented BPV-ECDSA. We used the implementations on same microcontroller to assess Ed25519 [23] and SchnorrQ [24].

We powered the microcontroller with a 2200 mAh power pack. ATmega 2560 operates at a voltage level of 5 V and takes 20 mA current[3]. We verified the current readings taken from datasheets by connecting an ammeter between the battery and ATmega 2560, and we observed an insignificant difference. Therefore, we measured the energy consumption with the formula $E = V \cdot I \cdot t$ where t is the computation time. To account the variations in time t, we run each scheme 10^4 times and took the average.

We also investigated the effect of cryptography on the battery life in some real-life IoT applications. For this purpose, we measured the energy consumption of a pulse sensor[4] and a BMP183 pressure sensor[5]. We expect that the pulse and pressure sensors provide some ideas on the use of digital signatures with sensors in medical devices and daily IoT applications, respectively.

- *Commodity Hardware*: We used an Intel i7-6700HQ 2.6 GHz processor with 12 GB of RAM as the commodity hardware in our experiments. We implemented the arithmetic and curve operations of our scheme with FourQlib[6]. We used BLAKE2b [21] as our hash function since it is optimized for commodity hardware. Lastly, we instantiated our PRF with AES in counter mode using Intel intrinsics. For our counterparts, we used their base implementations.

As the semi-honest party, we used an Amazon EC2 instance located in North Virginia. Our EC2 instance was equipped with an Intel Xeon E5 processor that operates at 2.4 GHz.

Our implementations are open-sourced at:

www.github.com/ozgurozmen/ESEM

C. Performance Evaluation and Comparisons

Low-end Device: Table I shows the results obtained from our implementations on 8-bit AVR ATmega 2560.

- *Signature Generation Speed*: ESEM has the fastest signing speed, which is 8.4× and 55× faster than that of SchnorrQ and Ed25519, respectively.
- *Energy Consumption of Signature Generation*: With a 2200 mAh battery, ESEM can generate nearly 800000 signatures, whereas SchnorrQ, Ed25519 and ECDSA can generate only 94482, 14235 and 6173 signatures, respectively. This

[1] https://github.com/rweather/arduinolibs/tree/master/libraries/Crypto
[2] https://github.com/kmackay/micro-ecc
[3] http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf
[4] https://pulsesensor.com/
[5] https://cdn-shop.adafruit.com/datasheets/1900_BMP183.pdf
[6] https://github.com/Microsoft/FourQlib

TABLE II: Experimental performance comparison of ESEM schemes and their counterparts on commodity hardware

Scheme	Signing Time (μs)	Private Key¶ (Byte)	Signature Size (Byte)	Verifier Comp. (μs)	Verifier Storage (Byte)	Server Comp. (μs)	Server Storage (KB)	End-to-End Delay† (μs)
ECDSA	725	32	64	927	32	–	–	1652
BPV-ECDSA	149	10272	64	927	32	–	–	1076
Ed25519	132	32	64	336	32	–	–	468
SchnorrQ	12	32	64	22	32	–	–	34
ESEM	**11**	32	**48**	24	32	5	32784	$11+24+5+\Delta$
ESEM_2	**4**	12416	**48**	24	32	10	4112	$4+24+10+\Delta$

¶ System wide parameters $params$ (e.g., p,q,α) for each scheme are included in their corresponding codes, and private key size denote to specific private key size.
† Δ represents the communication between the verifier and servers. Since the verifier communicates with $l = 3$ servers, the maximum communication delay is included in our end-to-end delay. This communication is measured to be 37 ms on average by our experiments, with an Amazon EC2 instance in N. Virginia.

shows that, ESEM can generate significantly higher number of signatures with the same battery.

• *Energy Consumption of Signature Generation versus IoT Sensors*: We considered a pulse and a pressure sensor to exemplify the potential medical and home automation IoT applications, respectively. We selected the sampling time (i.e., the frequency of data being read from the sensor) as every 10 seconds and every 10 minutes for the pulse and pressure sensor, respectively, to reflect their corresponding use-cases. We measured the energy consumption by considering three aspects: (i) Each sensor by default draws a certain energy as specified in its datasheet. The pulse sensor operates at 3 V and draws 4.5 mA of current, while pressure sensor operates at 2.5 V and draws 5 μA of current. These values are multiplied by their corresponding sampling rates to calculate the energy consumption of the sensor. (ii) AVR ATmega 2560 consumes energy to make readings from the sensor as well as during its waiting time. We measured the time that takes the microcontroller to have a reading from the sensor as 1 ms. Therefore, we calculated the energy consumption of the microcontroller on active time as $5V \cdot 20mA \cdot 1ms$. (iii) ATmega 2560 requires 10 μA in power-save mode, which is used to calculate the energy consumption in the idle time.

We compared the energy consumption of signature generation and IoT sensors in Figure 2. *ESEM reduces the energy consumption of signature generation to* 2.76% *and* 9.29% *compared to that of pulse and pressure sensors, respectively*. Observe that, compared with the pressure sensor, SchnorrQ as the fastest counterpart of ESEM, requires 46.24%, while Ed25519 demands 85.09% of the energy consumption. When the pulse sensor is used, while ESEM requires an almost negligible energy consumption (2.76%), its closest counterpart requires 19.29%. The energy efficiency of ESEM also translates into longer battery life in these applications. More specifically, when pressure sensor is deployed with ESEM, it takes 511 days to drain a 2200 mAh battery, while it is 303 days for our closest counterpart (SchnorrQ).

Our experiments show that the existing ECC-based digital signatures consume more energy than IoT sensors, which make them the primary source of battery consumption. On the other hand, *ESEM was able to reduce the signature generation overhead to a potentially negligible level in some cases, at minimum offering improvements over its counterparts.*

Commodity Hardware: The benchmarks of ESEM and its counterparts on commodity hardware are shown in Table II.

• *Signature Generation*: ESEM and ESEM_2 schemes offer the fastest signature generation on commodity hardware as well. Especially ESEM_2 (the high-speed variant where private key components are stored instead of generating them from a seed), is 3× faster than its closest counterpart.

• *Signature Verification*: The signature verification in ESEM includes verifier computation, server computation and communication between the verifier and servers. Due to the computational efficiency of FourQ curve, verifier and server computation of ESEM verification is highly efficient. Specifically, verifier computation takes 24 μs in ESEM and ESEM_2; and server computation takes 5 μs, and 10 μs for ESEM and ESEM_2, respectively. The communication between server and verifier is experimented with our commodity hardware and an Amazon EC2 instance at N. Virginia. This delay was measured as 37 ms on average.

The fastest verification is observed at SchnorrQ scheme, that is 22 μs. This scheme should be preferred if the verification speed is of high importance. However recall that for our envisioned applications, the signer efficiency (energy efficiency) is of top priority and a small delay at the verifier is tolerable.

VI. Related Work

There are two main lines of work to offer authentication for embedded medical devices: symmetric key primitives (e.g., MACs) and public key primitives (e.g., digital signatures). In this section, we only mention lightweight digital signature schemes that are most relevant to our work.

One-time signatures (e.g., [6], [25], [26]) offer high computational efficiency, but usually have very large key and signature sizes that hinder their adoption in implantable medical devices. Moreover, they can only sign a pre-defined number of messages with a key pair, which introduce a key renewal overhead. The extensions of hash-based one-time signatures to multiple-time signatures (e.g., SPHINCS [27]) have high signing overhead, and therefore are not suitable for medical implantables. Some MAC based alternatives (e.g., TESLA [28], [29]) use time asymmetries to offer computational efficient and compactness, they cannot offer non-repudiation and require a continuous time synchronization. EC-based digital signatures (e.g., [11], [13], [24], [30], [31], [32]) are currently the most prevalent alternatives to be used on embedded devices due to their compact size and higher signing efficiency compared to RSA-based signatures (e.g., CEDA [33]). We provided a detailed performance comparison of ESEM with its most recent EC-based alternatives in Section V.

VII. Conclusion

In this paper, we proposed ESEM, that achieves the least energy consumption, the fastest signature generation along with the smallest signature among its ECC-based counterparts. ESEM is also immune to side-channel attacks aiming EC operations/exponentiations as well as to weak pseudo random number generators at the signer's side, since ESEM does not require any of these operations in its signature generation algorithm. We believe ESEM is highly preferable for applications wherein the signer efficiency is a paramount requirement, such as implantable medical devices. We implemented ESEM and its counterparts both on a resource-contrained device commonly used in medical devices and a commodity hardware. Our experiments validate the significant energy efficiency and speed advantages of ESEM at the signer's side over its counterparts.

Acknowledgments. This work is supported by the NSF Award #1652389.

References

[1] 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 *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, ser. SP '14. IEEE Computer Society, 2014, pp. 524–539.

[2] 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

[3] 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.

[4] M. Vigil, J. Buchmann, D. Cabarcas, C. Weinert, and A. Wiesmaier, "Integrity, authenticity, non-repudiation, and proof of existence for long-term archiving: A survey," *Computers & Security*, vol. 50, pp. 16 – 32, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404814001849

[5] A. Ometov, P. Masek, L. Malina, R. Florea, J. Hosek, S. Andreev, J. Hajny, J. Niutanen, and Y. Koucheryavy, "Feasibility characterization of cryptographic primitives for constrained (wearable) iot devices," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, March 2016, pp. 1–6.

[6] 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.

[7] G. Ateniese, G. Bianchi, A. T. Capossele, C. Petrioli, and D. Spenza, "Low-cost standard signatures for energy-harvesting wireless sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 3, pp. 64:1–64:23, apr 2017.

[8] V. Boyko, M. Peinado, and R. Venkatesan, "Speeding up discrete log and factoring based schemes via precomputations," in *Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31 – June 4, 1998 Proceedings*. Springer Berlin Heidelberg, 1998, pp. 221–235.

[9] 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.

[10] D. 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, 2012.

[11] 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.

[12] 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.

[13] 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

[14] C. Pereida García, B. B. Brumley, and Y. Yarom, ""make sure dsa signing exponentiations really are constant-time"," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 1639–1650.

[15] R. R. Jueneman, "Securing wireless medicine confidentiality, integrity, nonrepudiation, malware prevention," in *Emerging Technologies for a Smarter World (CEWIT), 2011 8th International Conference Expo on*, Nov 2011, pp. 1–5.

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

[17] I. Goldberg, "Improving the robustness of private information retrieval," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, 2007, pp. 131–148.

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

[19] I. S. P. Nguyen and J. Stern, "Distribution of modular sums and the security of the server aided exponentiation," in *Proc. Workshop on Cryptography and Computational Number Theory (CCNT'99)*, vol. 20. Springer Berlin Heidelberg, pp. 257–268.

[20] 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.

[21] 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

[22] D. J. Bernstein, "New stream cipher designs," M. Robshaw and O. Billet, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, ch. The Salsa20 Family of Stream Ciphers, pp. 84–97. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68351-3_8

[23] M. Hutter and P. Schwabe, "NaCl on 8-bit AVR microcontrollers," in *Progress in Cryptology – AFRICACRYPT 2013*, ser. Lecture Notes in Computer Science, vol. 7918. Springer-Verlag Berlin Heidelberg, 2013, pp. 156–172, http://cryptojedi.org/papers/\#avrnacl.

[24] 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.

[25] 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.

[26] K. Kalach and R. Safavi-Naini, "An efficient post-quantum one-time signature scheme," in *Selected Areas in Cryptography – SAC 2015*, O. Dunkelman and L. Keliher, Eds. Cham: Springer International Publishing, 2016, pp. 331–351.

[27] 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: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer Berlin Heidelberg, April 2015, pp. 368–397.

[28] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[29] W. B. Jaballah, M. Conti, R. D. Pietro, M. Mosbah, and N. V. Verde, "Mass: An efficient and secure broadcast authentication scheme for resource constrained devices," in *2013 International Conference on Risks and Security of Internet and Systems (CRiSIS)*, Oct 2013, pp. 1–6.

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

[31] M. Wazid, A. K. Das, N. Kumar, M. Conti, and A. V. Vasilakos, "A novel authentication and key agreement scheme for implantable

medical devices deployment," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 4, pp. 1299–1309, July 2018.
[32] R. Behnia, M. O. Ozmen, and A. A. Yavuz, "ARIS: authentication for Real-Time IoT systems," in *2019 IEEE International Conference on Communications (ICC): Communication and Information Systems Security Symposium (IEEE ICC'19 - CISS Symposium)*, Shanghai, P.R. China, May 2019.
[33] M. O. Ozmen, R. Behnia, and A. A. Yavuz, "Compact energy and delay-aware authentication," in *2018 IEEE Conference on Communications and Network Security (CNS)*, May 2018, pp. 1–9.
[34] D. Pointcheval and J. Stern, "Security proofs for signature schemes," in *Proc. of the 15th International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '96)*. Springer-Verlag, 1996, pp. 387–398.
[35] 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.

APPENDIX

Proof of Theorem 1

Proof: If a polynomial-time adversary $\mathcal{A}$ breaks the EU-CMA security of ESEM, then one can build another polynomially-bounded algorithm $\mathcal{F}$ that runs $\mathcal{A}$ as a subroutine and breaks Schnorr signature. After setting $(y, Y) \leftarrow \texttt{Schnorr.Kg}(1^\kappa)$ and the corresponding parameters (q, p, α), $\mathcal{F}$ is run as in Definition 4 as follows. $\mathcal{F}$ handles $\mathcal{A}$'s and other queries in Definition 4.

Algorithm $\mathcal{F}^{\texttt{Schnorr.Sig}_y(\cdot)}(Y)$:

• *Setup:* $\mathcal{F}$ maintains five lists $\mathcal{LM}$, $\mathcal{HL}_0$, $\mathcal{HL}_1$, $\mathcal{HL}_2$ and $\mathcal{LW}$, all initially empty. $\mathcal{LM}$ is a message list that records each message M to be queried to $\texttt{ESEM.Sig}(\cdot)$ oracle by $\mathcal{A}$. $\mathcal{HL}_0$, $\mathcal{HL}_1$ and $\mathcal{HL}_2$ record the input x to be queried to $RO(\cdot)$ oracle and its corresponding $RO(\cdot)$ answer h, respectively. $h \leftarrow \mathcal{HL}_i(x)$, for $i \in \{0, 1, 2\}$, returns the corresponding $RO(\cdot)$ answer of x stored in $\mathcal{HL}_i$. $\mathcal{LW}$ keeps the record of messages that $\mathcal{F}$ queries to $\texttt{Schnorr.Sig}$ oracle. $\mathcal{F}$ sets up $RO(\cdot)$ and simulated public keys to initialize $\texttt{ESEM.Sig}$ oracle as follows:

- *Setup $RO(\cdot)$ Oracle*: $\mathcal{F}$ implements a function H-*Sim* to handle $RO(\cdot)$ queries. That is, the cryptographic hash function H is modeled as a random oracle via H-*Sim* as follows. $h \leftarrow H\text{-}Sim(x, \mathcal{HL}_i)$: If $h \in \mathcal{HL}_i$, for $i \in \{0, 1, 2\}$, then H-*Sim* returns the corresponding value $h \leftarrow \mathcal{HL}_i(M)$. Otherwise, it returns $h \xleftarrow{\$} \mathbb{Z}_q^*$ as the answer, inserts (M, h) into $\mathcal{HL}_i$, respectively.
- *Setup Simulated Keys*: $\mathcal{F}$ selects parameters (v, n) as in $\texttt{ESEM.Kg}(\cdot)$ Step 1, and works as follows:
  1) Queries $\texttt{Schnorr.Sig}_y(\cdot)$ on $w_j \xleftarrow{\$} \mathbb{Z}_q^*$, and receives signatures (s'_j, h'_j), where $\{R'_j \leftarrow Y^{h'_j} \cdot \alpha^{s'_j} \bmod p\}_{j=0}^{n-1}$.
  2) Sets $PK \leftarrow Y$ and system-wide public parameters (v, n, q, p, α) as in $\texttt{ESEM.Kg}(\cdot)$.
  3) Sets $z \leftarrow \{0,1\}^\kappa$ and $\gamma = \{s'_j, h'_j\}_{j=0}^{n-1}$ and $\beta = \{R'_j\}_{j=0}^{n-1}$ and inserts $\{w_j\}_{j=0}^{n-1}$ into $\mathcal{LW}$. $\mathcal{F}$ also inserts $\{w_j || R'_j, h'_j\}_{j=0}^{n-1}$ into $\mathcal{HL}_2$. $\mathcal{F}$ sets the counter $c \leftarrow 0$.

• *Execute* $(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(\cdot), \texttt{ESEM.Sig}_{sk}(\cdot)}(PK)$: $\mathcal{F}$ handles $\mathcal{A}$'s queries and forgery as follows:

- Queries of $\mathcal{A}$: $\mathcal{A}$ queries $RO(\cdot)$ and $\texttt{ESEM.Sig}_{sk}(.)$ on any message of its choice up to q_h and q_s times, respectively.
  1) *Handle $RO(\cdot)$ queries*: $\mathcal{A}$ queries $RO(\cdot)$ on a message M. $\mathcal{F}$ calls $h \leftarrow H\text{-}Sim(M, \mathcal{HL}_2)$ and returns h to $\mathcal{A}$.
  2) *Handle $\texttt{ESEM.Sig}(\cdot)$ queries*: $\mathcal{A}$ queries $\texttt{ESEM.Sig}_{sk}(\cdot)$ on any message of its choice M. If $M \in \mathcal{HL}_1$ then $\mathcal{F}$ *aborts*. Otherwise, $\mathcal{F}$ inserts M into $\mathcal{LM}$ and then continues as follows.
     i) $x \leftarrow H\text{-}Sim(z||c, \mathcal{HL}_0)$, $(k_0, \ldots, k_{v-1}) \leftarrow H\text{-}Sim(z||x, \mathcal{HL}_1)$, and fetch $\{s'_{k_i}, h'_{k_i}\}_{i=0}^{v-1}$ from γ.
     ii) $\overline{h} \leftarrow \sum_{i=0}^{v-1} h'_{k_i} \bmod q$, $\overline{s} \leftarrow \sum_{i=0}^{v-1} s'_{k_i} \bmod q$, put $(M||x, \overline{h})$ in $\mathcal{HL}_2$ and return $\sigma = (\overline{s}, x)$ to $\mathcal{A}$.
- Forgery of $\mathcal{A}$: Finally, $\mathcal{A}$ outputs a forgery for PK as (M^*, σ^*), where $\sigma^* = (s^*, x^*)$. By Definition 4, $\mathcal{A}$ wins the *EU-CMA* experiment for ESEM if $\texttt{ESEM.Ver}(M^*, \sigma^*, PK) = 1$ and $M^* \notin \mathcal{LM}$.

• *Forgery of $\mathcal{F}$*: If $\mathcal{A}$ fails, $\mathcal{F}$ also fails and returns 0. Otherwise, given an ESEM forgery $(M^*, \sigma^* = \langle s^*, x^* \rangle)$ on PK, $\mathcal{F}$ checks if $M^*||x^* \notin \mathcal{HL}_1$, or $x^* \notin \mathcal{HL}_0$ holds, $\mathcal{F}$ *aborts*. Otherwise, using the forking lemma [34], [35], $\mathcal{F}$ rewinds $\mathcal{A}$ with the same random tape, to get a new forgery $(M^*, \widetilde{\sigma} = \langle \widetilde{s}, \widetilde{x} \rangle)$. Given, both forgery pairs are valid, we have:

$$\begin{aligned} R^* &\equiv (\alpha^y)^{h^*} \cdot \alpha^{s^*} \bmod p \\ R^* &\equiv (\alpha^y)^{\widetilde{h}} \cdot \alpha^{\widetilde{s}} \bmod p \end{aligned}$$

These equations imply the below modular linear equations.

$$\begin{aligned} r^* &\equiv y \cdot h^* + s^* \bmod q \\ r^* &\equiv y \cdot \widetilde{h} + \widetilde{s} \bmod q \end{aligned}$$

$\mathcal{F}$ then extracts Schnorr private key y by solving the above modular linear equations (note that only unknowns are y and r^*). $\mathcal{F}$ can further verify $Y \equiv \alpha^y \bmod p$. Given that $\mathcal{F}$ extracted the Schnorr private key, it is trivial to show that $\mathcal{F}$ can produce a valid forgery on any message $M^{**} \notin \mathcal{LW}$ of its choice on Schnorr public key Y. Therefore, $\mathcal{F}$ wins *EU-CMA* experiment for Schnorr and returns 1.

Probability Analysis & Transcript Indistinguishability:
$\mathcal{F}$ may abort during simulation when $\mathcal{A}$ queries the $\texttt{ESEM.Sig}(\cdot)$ oracle, if randomly chosen indexes $(k_0, \ldots, k_{v-1})$ already exist in $\mathcal{HL}_1$. The probability that happens is $^{(q_h-1)q_s}/_{2^l}$, $l = v \cdot \log n$. Based on [35, Lemma 1], we define the success probability of $\mathcal{A}$ as ACC. This probability is defined as $\texttt{ACC} \geq \epsilon_{\mathcal{A}} - {}^{(q_h-1)q_s}/_{2^l}$. Where $\epsilon_{\mathcal{A}}$ is the winning probability of $\mathcal{A}$. The probability of $\mathcal{F}$ (i.e., $\epsilon_{\mathcal{F}}$) for breaking Schnorr is given by:

$$\begin{aligned} \epsilon_{\mathcal{F}} &\geq \frac{\texttt{ACC}^2}{q_h + q_s} - \frac{1}{2^l} \\ &\geq \frac{\epsilon_{\mathcal{A}}^2}{(q_h + q_s)} - \frac{2((q_h - 1)q_s)}{2^l(q_h + q_s)} \end{aligned}$$

$\mathcal{A}$'s view in Algorithm 2 is the public key PK, signatures $(\sigma_1, \ldots, \sigma_{q_s-1})$ and hash outputs. The public key in the simulation has the identical distribution as the one in original ESEM - they are both the output of the $\texttt{Schnorr.Kg}(\cdot)$ algorithm. As for the signatures, $\sigma = (s, x)$, both elements s and x have the same distribution as in the original scheme.