ETA: E̲fficient and T̲iny and A̲uthentication for Heterogeneous Wireless Systems

Attila Altay Yavuz
University of Pittsburgh
135 N. Bellefield Avenue, Pittsburgh, PA 15260
attila.yavuz@gmail.com

ABSTRACT

Authentication and integrity are vital security services for wireless ubiquitous systems, which require various resource-constrained devices to operate securely and efficiently. Digital signatures are basic cryptographic tools to provide these security services. However, existing digital signatures are not practical for resource-constrained systems (e.g., wireless sensors, RFID-tags). That is, traditional signatures (e.g., RSA, DSA) require expensive operations (e.g., modular exponentiation) that bring high computational cost and power-consumption. Some alternative schemes (e.g., multiple-time signatures, online/offline signatures, pre-computed tokens) are computationally efficient. However, they have large key and signature sizes and therefore are impractical for resource-constrained systems.

In this paper, we develop a new cryptographic scheme called *E̲fficient and T̲iny A̲uthentication (ETA)*, which is especially suitable for resource-constrained devices. That is, *ETA* does not require any expensive operation at the signer side and therefore is more computationally efficient than traditional signatures. Moreover, *ETA* has much smaller private key, signature and public key sizes than that of its counterparts (e.g., multiple-time and online/offline signatures, pre-computed tokens). *ETA* is also fully tolerant to packet loss and does not require time synchronization. All these properties make *ETA* an ideal choice to provide authentication and integrity for heterogeneous systems, in which resource-constrained devices produce publicly verifiable signatures that are verified by resourceful devices (e.g., gateways, laptops, high-end sensors).

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Security

Keywords

Lightweight cryptography; applied cryptography; broadcast authentication; wireless network security

1. INTRODUCTION

Nowadays ubiquitous systems like Internet of Things and Systems (IoTS) and Wireless Sensor Networks (WSNs) are composed of resource-constrained devices (e.g., low-end sensors, smart-cards, RFID-tags). It is vital for such systems to operate securely and efficiently. Hence, it is necessary to provide authentication and integrity services 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 power limitations of these devices.

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 [13, 27]. They also cannot achieve the non-repudiation property, which is necessary for various applications (e.g., transportation payment systems, logical/pyhsical access with tiny devices).

Digital signatures rely on public key infrastructures for the signature verification [9]. Hence, they are publicly verifiable, scalable for large systems and achieve the non-repudiation. However, digital signatures also have limitations (discussed in Section 1.1) that prevent them to be practical for resource-constrained devices.

We first briefly discuss some existing alternatives and their limitations. We then present our contribution by summarizing the desirable properties of our scheme. Last, we provide a brief discussion on some prospective application areas of *ETA* and its limitations.

1.1 Related Work

Well-known modular-root based signatures (e.g., RSA [21]) are computationally efficient at the verifier side. However, these schemes require expensive operations[1] at the signer side and have large key/ signature sizes. Hence, they are not suitable for computing and transmitting signatures for resource-constrained devices.

Some DLP-based signature schemes (e.g., Schnorr [22], DSA [1]) offer small key and signature sizes with an implementation on Elliptic Curves (ECs) [7]. However, they still require an expensive operation at the signer side. It is possible to eliminate the expensive operation during signature generation via pre-computed cryptographic tokens, which can be generated offline (e.g., DSA with tokens [16]). Despite being computationally efficient, this ap-

[1]We refer operations such as modular exponentiation [9], elliptic curve scalar multiplication [7] or pairing [15] as an expensive operation.

Table 1: Private/public key sizes, signature size and signature generation/verification costs of ETA and previous schemes

K-time overhead	Signer			Verifier	
	Private Key Size	***Signature Size***	***Signature Generation Cost***	***Public Key Size***	***Signature Verification Cost***
ETA	$2\|q\|$	$\|q\| + \kappa$	$2H + Mulq$	$\|q\| + \|H\| \cdot O(K)$	$1.3 \cdot EMul$
HORS	$(t \cdot \kappa)O(K)$	$u \cdot \kappa$	H	$(t \cdot \|H\|)O(K)$	$u \cdot H$
HORSE	$(t \cdot \kappa)O(log_2 K)$	$u \cdot \kappa$	$log_2(K) \cdot H$	$t \cdot \|H\|$	$u \cdot H$
HORS++	$(\kappa \cdot \|H\|) \cdot O(K^2)$	$(\kappa \cdot \|H\|) \cdot O(K)$	H	$\|H\|^2 \cdot O(K^2)$	$u \cdot H$
Online/offline	$2\|n\|$	$2\|n\|$	$H + 0.1 \cdot Muln$	$2\|n\|$	$Expn$
ECDSA	$\|q\|$	$2\|q\|$	$EMul$	$\|q\|$	$1.3 \cdot EMul$
Token-ECDSA	$(\|q\| + \kappa) \cdot O(K)$	$2\|q\|$	$H + 2Mulq$	$\|q\|$	$1.3 \cdot EMul$

(i) K denotes the number of messages to be signed. H and $|H|$ denote a cryptographic hash operation and output bit length of H, respectively. $Expn$ and $Emul$ denote a modular exponentiation over modulus n and an ECC scalar multiplication over modulus q, respectively. $Mulq$ and $Muln$ denote a modular multiplication over modulus q and modulus n, respectively. We omit constant number of low-cost operations if there is an expensive operation (e.g., a single H is omitted if there is an $EMul$). We use double-point scalar multiplication for ETA and ECDSA verifications ($1.3 \cdot Emul$ instead of $2 \cdot EMul$). Integers t and u denote the parameters used in HORS and HORSE.
(ii) Given $\kappa = 80$, suggested parameter sizes are as follows: $|H| = 160,\ |q| = 160,\ |n| = 1024,\ t = 256, u = 20$.
(iii) ETA only needs a few low-cost operations for the signature generation, which makes it much more computationally efficient than ECDSA and more efficient than token-ECDSA and HORSE. Note that in our system model (see Section 2), verifiers are resourceful and therefore they can perform $Emul$s. Despite being more computationally efficient than ETA, HORS and HORS++ are impractical for resource-constrained devices due to their extremely high storage and communication overheads (see Table 2).

Table 2: Private/public key and signature sizes of ETA and previous schemes (numerical)

K-time overhead	Signer				Verifier		
	Private Key Size			***Signature Size***	***Public Key Size***		
	K=1	$K = 10^2$	$K = 10^4$	$K = 10^4$	K=1	$K = 10^2$	$K = 10^4$
ETA	40 byte	40 byte	40 byte	30 byte	40 byte	1.9 KB	195 KB
HORS	2.5 KB	12 MB	24 MB	200 byte	5 KB	24 MB	48 MB
HORSE	2.5 KB	16 KB	32 KB	200 byte	5 KB	24 MB	48 MB
HORS++	1.5 KB	15 MB	> 1 GB	15 MB	3 KB	30 MB	> 1 GB
Online/Offline	256 byte	256 byte	256 byte	256 byte	256 byte	256 bye	256 byte
ECDSA	20 byte	20 byte	20 byte	40 byte	20 byte	20 byte	20 byte
Token-ECDSA	30 byte	2.9 KB	29 KB	40 byte	20 byte	20 byte	20 byte

ETA has *the smallest signature size* among its counterparts. It also has a small-constant private key, which is significantly smaller than that of its counterparts (see for $K = 10^4$). Notice that, despite being compact, ECDSA requires an expensive operation (i.e., $Emul$) for each data item to be signed and therefore it is much more computationally costly than ETA at the signer side (e.g., the estimated execution time of ECDSA is 1330 μsec, while it is only 4 μsec for ETA, see Section 5 for implementation details).

proach requires storing a pre-computed one-time token for each message to be signed. This incurs a linear storage overhead to the signer side, which is impractical for resource-constrained devices.

Online/offline signatures are generic transformations that can shift expensive signature computations to the offline phase for *any* signature scheme. Previous online/offline signatures (e.g., [6]) have very large key and signature sizes. Recent constructions (e.g., [4,23]) are more space efficient. However, similar to the pre-computed tokens, online/offline signatures also require storing a pre-computed value for each message to be signed. This introduces a linear storage overhead to the signer side.

One-time signatures (OTSs) [10, 20] rely on one-way functions without trapdoors. Hence, they offer a computationally efficient alternative to the traditional signature schemes. One of the most efficient OTSs is Hash-to-Obtain Random Subset (HORS) [20], whose signature generation is as efficient as a single hash computation. However, all these OTSs have extremely large private/public key (e.g., 2.5KB/5KB in HORS) and large signature sizes. Moreover, a private/public key pair can be used only once. This requires distribution of new public keys, which causes further overhead. *Multiple-time signatures* (e.g., HORS++ [19], HORSE [17]), also called K-time signatures, can compute a constant number of signatures under the same private/public key. However, the key and signature sizes of these schemes are larger than that of HORS, which make them prohibitive for resource-constrained devices.

1.2 Our Contribution

In this paper, we develop a new signature scheme, that we call *Efficient and Tiny Authentication (ETA)*, which is ideal for resource-constrained devices. We summarize the desirable properties of ETA below (the representative key sizes are given for the security parameter $\kappa = 80$). Table 1 and Table 2 further compare ETA and its counterparts with respect to various performance metrics.

1) Small Key Sizes: In ETA, regardless of the value of K (i.e., the number of messages to be signed), the size of private key is small-constant (i.e., 320 bytes). This is significantly smaller than that of token-based techniques (e.g., tokens with ECDSA [16]) and online/offline signatures (e.g., [6, 23]) that require $O(K)$ at the signer side. This is more efficient than that of multiple-time signatures (e.g., [17,19,20]) with large key sizes (e.g., 2.5 KB in HORS).

In ETA, the size of public key is also much smaller than that of HORS and its extensions. The size of public key component for each message is $|H| = 160$ bits in ETA, whereas it is 5KB in HORS and HORSE ($|H|$ denotes the bit length of hash output). The size of K-time public key is $|H| \cdot O(K)$ in ETA, whereas it is $|H|^2 O(K^2)$ in adaptive-secure HORS++.

2) Highly Compact Signature: In ETA, the size of signature can be as small as 240 bits. This is much more efficient than that of one-time/multiple-time (e.g., 1600-3200 bits in HORS) and online/offline signatures (e.g., 2048 bits [4, 23]). It is also more efficient than that of some traditional schemes (e.g., ECDSA, RSA) and token-based alternatives (e.g., 320 bits for ECDSA tokens [16]).

3) Expensive Operation-free Signature Computation: ETA does not require any expensive operation to compute signatures (i.e., one modular addition/multiplication and two hash operations for each message). Hence, ETA is several orders of magnitude more computationally efficient than traditional signature schemes (e.g., ECDSA, RSA) and as efficient as token-based alternatives. It is also comparable to HORS and its multiple-time extensions, while being much more compact than those alternatives.

4) Practical Signing Capability: Existing multiple-time signatures are not practical even for a small K. In ETA, all the signer performance metrics are independent from K, which permits a resource-constrained device to sign very large number of messages without having any performance issue.

5) Immediate Verification and No Synchronization Requirement: Unlike some existing alternatives (e.g., [18,25]), ETA does not require any time synchronization between the signer and verifier. ETA also achieves immediate verification, which is important for real-time applications (e.g., vehicular networks).

6) Individual Message Verification: ETA allows receivers to verify each individual message independently (unlike some HORS variants such as [17,25]). This eliminates packet loss, time synchronization and security problems that stem from the interdependency between messages (or their corresponding signatures).

1.3 Applications

ETA is suitable for heterogeneous systems and applications, in which signers are computational, storage and bandwidth limited but verifiers are resourceful. We discuss some illustrative applications, in which ETA can be useful.

• *Increasing the Life Span of WSNs*: Many secure WSN protocols such as clone detection (e.g., [5]), secure code dissemination [8] and secure logging [26] need a signer optimal signature scheme. ETA can substantially increase the life span of WSNs by serving as a building block for such protocols, since it is compact and does not require any expensive operation at the signer side.

Token-based Payment and Access Systems: Nowadays intelligent transportation and mobile systems need mass producible low-cost payment devices [2]. Some examples include toll systems (e.g., E-ZPass), pre-paid systems (e.g., MetroCards in NYC, Pay-as-You-Drive Carsharing), NFC-based mobile payment systems and token-based logical/pyhsical access (e.g., with USBs).

ETA is an ideal primitive for such applications, since it can sign very large number of transactions by incurring near-zero computation, storage and communication overhead to the resource-constrained device. Recall that its counterparts either require an expensive operation for per transaction or incur a linear token storage overhead (which is impractical for such devices).

• *Real-time Authentication in Cyber Physical Systems*: Secure and rapid dissemination of system measurements (e.g., voltage, frequency) is essential to prevent the cascade failures [12] in cyber physical systems like smart-grids. Hence, measurement devices (e.g., phasor measurement units) must sign system readings before multicasting them to the the control centers [14]. Time and security critical nature of these applications require real-time authentication of system readings. ETA is much more practical than its counterparts (e.g., one-time/multiple time schemes [11, 25]) for such applications as shown in Section 5.

• *Limitations*: In ETA, the public key size is linear with respect to K. This requires verifiers to be storage resourceful. Moreover, despite being very efficient during signature generation (online phase), ETA requires expensive operations at its key generation (offline phase). Therefore, the initial key generation must be performed offline before system deployment (similar to online/offline signatures [4] and cryptographic tokens [16]).

Observe 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., base stations in WSNs, control centers in cyber physical systems). Similarly, it is feasible to perform key generation phase offline in these applications. For instance, a key generation center can generate and distribute keys to the devices in payment systems and WSNs. Moreover, ETA's ability to select a very large K without creating a burden at the signer side further remedies these limitations.

2. SYNTAX AND MODELS

Notation. $||$ denotes the concatenation operation. $|r|$ denotes the bit length of variable r. $r \xleftarrow{\$} \mathcal{M}$ denotes that variable r is randomly and uniformly selected from set $\mathcal{M}$. We denote by $\{0,1\}^*$ the set of binary strings of any finite length. H is an ideal cryptographic hash function, which is defined as $H : \{0,1\}^* \rightarrow \{0,1\}^{|H|}$, where $|H|$ denotes the output bit length of H. $\mathcal{A}^{\mathcal{O}_0,\ldots,\mathcal{O}_i}(\cdot)$ 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.

ETA relies on the Schnorr signature scheme [22].

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

- $(y, Y, I) \leftarrow Schnorr.Kg(1^\kappa)$*: The key generation algorithm takes 1^κ as the input. It generates large primes q and $p > q$ such that $q|(p-1)$, and generates a generator α of the subgroup G of order q in $\mathbb{Z}_p^*$. It returns a private/public key pair ($y \xleftarrow{\$} \mathbb{Z}_q^*, Y \leftarrow \alpha^y \bmod p$) and a system parameter $I \leftarrow (q, p, \alpha)$ as the output.*

- $(s, e) \leftarrow Schnorr.Sig(y, M)$*: The signature generation algorithm takes private key y and a message M as the input. It returns a signature (s, e) as follows:*

  $R{\leftarrow}\alpha^r \bmod p$, $e{\leftarrow}H(M||R)$, $s{\leftarrow}(r{-}e{\cdot}y) \bmod q$, *where* $r \xleftarrow{\$} \mathbb{Z}_q^*$ *and H is defined as* $H : \{0,1\}^* \rightarrow Z_q^*$.

- $b \leftarrow Schnorr.Ver(Y, M, \langle s, e\rangle)$*: The signature verification algorithm takes Y, M and $\langle s, e\rangle$ as the input. It computes $R' \leftarrow Y^e\alpha^s \bmod p$ and returns a bit b, with $b = 1$ meaning* valid*, if $e = H(M||R')$ and $b = 0$ otherwise.*

Definition 2 *Existential Unforgeability under Chosen Message Attack (EU-CMA) [9] experiment is as follows:*
Experiment $Expt^{EU\text{-}CMA}_{SGN}(\mathcal{A})$

$(sk, PK, I) \leftarrow SGN.Kg(1^\kappa)$, $(M^*, \sigma^*) \leftarrow \mathcal{A}^{SGN.Sig_{sk}(\cdot)}(PK)$,

If $SGN.Ver(PK, M^*, \sigma^*) = 1$ *and* M^* *was not queried, return* 1*, else, return* 0.

The EU-CMA advantage of $\mathcal{A}$ is defined as $Adv^{EU\text{-}CMA}_{SGN}(\mathcal{A}) = Pr[Expt^{EU\text{-}CMA}_{SGN}(\mathcal{A}) = 1]$. *The EU-CMA advantage of SGN is defined as* $Adv^{EU\text{-}CMA}_{SGN}(t, K) = \max_{\mathcal{A}}\{Adv^{EU\text{-}CMA}_{SGN}(\mathcal{A})\}$*, where the maximum is over all $\mathcal{A}$ having time complexity t and making at most K oracle queries.*

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

(i) *Resource-constrained Signers*: Signers are storage, computational, bandwidth and power limited devices (e.g., wireless sensor, RFID-tag). The objective of ETA is to minimize the cryptographic overhead of signers. (ii) *Resourceful Verifiers:* Storage and computational resourceful verifiers (e.g., a laptop, base station) who can be any (untrusted) entity.

We assume that the key generation/distribution is performed *offline* before deployment. A key generation center generates private/public keys offline and distributes them to the system entities.

Syntax and Security Model: ETA is a multiple-time (K-time) signature scheme, which can sign a pre-determined number of messages under the same public key.

Definition 3 *ETA is comprised of a tuple of three algorithms (Kg, Sig, Ver) defined as follows:*

- $(sk_0, PK, I) \leftarrow ETA.Kg(1^\kappa, K)$: *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) and a system parameter I as the output, where PK is a vector with $K+1$ elements.*
- $\sigma_j \leftarrow ETA.Sig(sk_j, M_j)$: *The signature generation algorithm takes the current private key sk_j, $0 \le j \le K-1$ 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}.*
- $b \leftarrow ETA.Ver(PK, M_j, \sigma_j)$: *The signature verification algorithm takes PK, message M_j and its corresponding signature σ_j, $0 \le j \le K-1$ as the input. It returns a bit b, with $b = 1$ meaning* valid, *and $b = 0$ otherwise.*

The details of ETA algorithms are given in Section 3.

ETA is proven to be (K-time) *Existentially Unforgeable against Chosen Message Attack (EU-CMA)* based on the experiment defined in Definition 4. 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 her choice up to K' messages. (ii) A signing oracle $ETA.Sig_{sk}(.)$ from which $\mathcal{A}$ can request a ETA signature on any message M of her choice up to K messages.

Definition 4 EU-CMA experiment *for ETA is as follows:*
Experiment $Expt_{ETA}^{EU\text{-}CMA}(\mathcal{A})$
$(sk, PK, I) \leftarrow ETA.Kg(1^\kappa, K)$,
$(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.), ETA.Sig_{sk}(.)}(PK)$,
If $ETA.Ver(PK, M^*, \sigma^*) = 1$ and M^* was not queried to $ETA.Sig_{sk}(.)$, return 1, else, return 0.

The EU-CMA-advantage of $\mathcal{A}$ is defined as $Adv_{ETA}^{EU\text{-}CMA}(\mathcal{A}) = Pr[Expt_{ETA}^{EU\text{-}CMA}(\mathcal{A}) = 1]$. The EU-CMA-advantage of ETA is defined as $Adv_{ETA}^{EU\text{-}CMA}(t, K', K) = \max_{\mathcal{A}}\{Adv_{ETA}^{EU\text{-}CMA}(\mathcal{A})\}$, 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 $ETA.Sig_{sk}(.)$.

3. THE PROPOSED SCHEME

Some DLP-based signatures (e.g., ECDSA [16], Schnorr [22]) can eliminate expensive operations from the signature generation by pre-computing and storing the components (R, r), where $r \xleftarrow{\$} \mathbb{Z}_q^*$ and $R = \alpha^r \bmod p$. However, this approach incurs linear storage to the signer side (i.e., one token for each message).

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

We outline our strategies that overcome the above limitation.

• *Eliminate R from Signature Generation and Transmission*: We enable signer to compute signatures by relying on random r instead of token R. In contrast to R, random r can be evolved without requiring any expensive operation (e.g., with a hash operation). This offers a small-constant storage at the signer side, since the current r_j can be derived from the previous r_{j-1} efficiently.

In Schnorr, different from ElGamal and ECDSA, not R but the hash of it is used in signature generation as $s \leftarrow r - e \cdot y \bmod q$, where $e \leftarrow H(M||R)$. We mimic R in e by replacing it with a random number $x_j \leftarrow \{0,1\}^\kappa$ as $e_j \leftarrow H(M_j||j||x_j)$. Our modification preserves the provable security assuming that H is a random oracle (see Theorem 1 in Section 4). This strategy offers small private key (i.e., 320 bits) and signature sizes (i.e., 240 bits).

• *Offload Token Storage to the Verifier Side*: Since we eliminate R from the signature generation, it is possible to store it at the verifier side. Note that R does not disclose r and verifiers are storage resourceful in our system model.

We pre-compute the corresponding R_j of each r_j and give the hash of each R_j to the verifier as $v_j = H(R_j)$ for $j = 0, \ldots, K-1$ before the deployment. This allows a much smaller public key size than that of existing multiple-time signatures. Note that since $\{v_j\}_{j=0}^{K-1}$ are a part of public key, corresponding $\{R_j\}_{j=0}^{K-1}$ are *authenticated*, despite they are not signed during the signature generation (in contrast to Schnorr).

To verify signatures, we rely on the fact that Schnorr verification algorithm recovers component R_j as $R_j \leftarrow Y^{H(.)} \cdot \alpha^{s_j}$. Given the signature (s_j, x_j, j), the verifier recovers R'_j and checks whether it is the same with the original $R_j \in PK$ as $v_j = H(R'_j)$.

The detailed description of ETA algorithms is given below.

<u>1) $(sk_0, PK, I) \leftarrow ETA.Kg(1^\kappa, K)$</u>:

a) Invoke $(y, Y, \langle q, p, \alpha\rangle) \leftarrow Schnorr.Kg(1^\kappa)$. Set $I \leftarrow (K, q, p, \alpha)$ for ETA.
b) Generate $R_j \leftarrow \alpha^{r_j} \bmod p$, where $r_0 \xleftarrow{\$} \mathbb{Z}_q^*$ and $r_j \leftarrow H(r_{j-1})$ for $j = 1, \ldots, K-1$. Generate verification tokens as $v_j \leftarrow H(R_j)$ for $j = 0, \ldots, K-1$.
c) The private and public key are $sk_0 \leftarrow (y, r_0)$ and $PK \leftarrow (Y, \overrightarrow{v} = v_0, \ldots, v_{K-1})$, respectively.

<u>2) $\sigma_j \leftarrow ETA.Sig(sk_j, M_j)$</u>: Given $sk_j = (y, r_j)$, compute signature σ_j on a message M_j as follows,

a) $e_j \leftarrow H(M_j||j||x_j)$ and $s_j \leftarrow r_j - e_j \cdot y \bmod q$, where $x_j \xleftarrow{\$} \{0,1\}^\kappa$. The signature σ_j on M_j is $\sigma_j \leftarrow (s_j, x_j, j)$.
b) Update r_j as $r_{j+1} \leftarrow H(r_j)$ and erase r_j (to save memory).
c) If $j > K-1$ then return $\perp$ (i.e., the limit on the number of signatures is exceed).

<u>3) $b \leftarrow ETA.Ver(PK, M_j, \sigma_j)$</u>: Recall that $\sigma_j = (s_j, x_j, j)$ and $PK = (Y, \overrightarrow{v})$, where $\overrightarrow{v} = v_0, \ldots, v_{K-1}$. If $j > K-1$ then return $\perp$. Otherwise, if $v_j = H(R'_j)$ $ETA.Ver$ returns 1, else, it returns 0, where $R'_j \leftarrow Y^{H(M_j||j||x_j)} \cdot \alpha^{s_j}$.

4. SECURITY ANALYSIS

We prove that ETA is a (K-time) *EU-CMA* signature scheme in Theorem 1 (in the random oracle model [3]). We ignore terms that are negligible in terms of κ.

Theorem 1 $Adv_{ETA}^{EU\text{-}CMA}(t, K', K) \le Adv_{Schnorr}^{EU\text{-}CMA}(t', K)$, *where $t' = O(t) + 3K \cdot (O(\kappa^3) + RNG) + K' \cdot RNG$.*

Proof: Let $\mathcal{A}$ be a *ETA attacker*. We construct a *Schnorr attacker* $\mathcal{F}$ that uses $\mathcal{A}$ as a sub-routine. That is, we set $(y, Y, \langle q, p, \alpha\rangle) \leftarrow Schnorr.Kg(1^\kappa)$ by Definition 1 and then run the simulator $\mathcal{F}$ by Definition 2 (i.e., $EU\text{-}CMA$ experiment) as follows:

Algorithm $\mathcal{F}^{Schnorr.Sig_y(.)}(Y)$

- 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 queried to $RO(.)$ oracle and its corresponding $RO(.)$ answer h_j, respectively. $\mathcal{F}$ sets counters ($l' \leftarrow 0, l \leftarrow 0, n \leftarrow 0$) and continues as follows:
  - $h \leftarrow H\text{-}Sim(M, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'})$: $\mathcal{F}$ implements a function $H\text{-}Sim$ to handle $RO(.)$ queries. That is, cryptographic function H is modeled as a random oracle via $H\text{-}Sim$. If $\exists j : M = \overrightarrow{\mathcal{L}}[j]$ then $H\text{-}Sim$ returns $\overrightarrow{\mathcal{L}'}[j]$. Otherwise, it returns $h \xleftarrow{\$} \mathbb{Z}_q^*$ as the answer, assigns ($\overrightarrow{\mathcal{L}}[l] \leftarrow M$, $\overrightarrow{\mathcal{L}'}[l] \leftarrow h$) and $l \leftarrow l + 1$.
  - $\mathcal{F}$ creates a simulated ETA public key PK as follows:

    (i) Query $Schnorr.Sig_y(.)$ on $d_j \xleftarrow{\$} \mathbb{Z}_q^*$ for $j = 0, \ldots, K-1$. $Schnorr.Sig_y(.)$ returns a signature (s_j, e_j), where $R_j \leftarrow Y^{e_j} \cdot \alpha^{s_j} \bmod p$ for $j = 0, \ldots, K-1$.

    (ii) Set $PK \leftarrow (Y, \overrightarrow{v})$, where $\overrightarrow{v} = \{H\text{-}Sim(R_j, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'})\}_{j=0}^{K-1}$.
- Execute $(M^*, \sigma^*) \leftarrow \mathcal{A}^{RO(.), ETA.Sig_{sk}(.)}(PK)$:
  - Queries: $\mathcal{A}$ queries $RO(.)$ and $ETA.Sig$ oracles on up to K and K' messages of her choice, respectively. $\mathcal{F}$ handles these queries as follows:

    (i) $\mathcal{A}$ queries $RO(.)$ on a message M. If $l' > K' - 1$ then $\mathcal{F}$ rejects the query (i.e., the query limit is exceeded). Otherwise, $\mathcal{F}$ invokes $h \leftarrow H\text{-}Sim(M, l, \overrightarrow{\mathcal{L}}, \overrightarrow{\mathcal{L}'})$, returns h as the answer and increments $l' \leftarrow l' + 1$.

    (ii) $\mathcal{A}$ queries $ETA.Sig$ oracle on a message M_n. If $n > K - 1$ then $\mathcal{F}$ rejects the query (i.e., the query limit is exceeded). Otherwise, $\mathcal{F}$ generates $x_n \xleftarrow{\$} \{0,1\}^\kappa$ and checks if $(M_n||n||x_n) \in \overrightarrow{\mathcal{L}}$. If it holds then $\mathcal{F}$ *aborts* (i.e., the simulation fails). Otherwise, $\mathcal{F}$ simulates the hash output of $M_n||n||x_n$ with e_n by inserting the tuple $(M_n||n||x_n, e_n)$ to $(\overrightarrow{\mathcal{L}}[l] \leftarrow M_n||n||x_n, \overrightarrow{\mathcal{L}'}[l] \leftarrow e_n)$. $\mathcal{F}$ then simulates the ETA signature by setting $\sigma_n \leftarrow (s_n, x_n, n)$. $\mathcal{F}$ returns σ_n to $\mathcal{A}$, assigns $\overrightarrow{\mathcal{M}}[n] \leftarrow M_n$ 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 $\sigma^* = (s_j^*, x_j^*, j)$, $0 \le j \le K-1$. By definition 4, $\mathcal{A}$ wins the $EU\text{-}CMA$ experiment for ETA if $ETA.Ver(PK, M^*, \sigma^*) = 1$ and $M^* \notin \overrightarrow{\mathcal{M}}$ hold. If these conditions hold, $\mathcal{A}$ returns 1, else she returns 0.
- Forgery of $\mathcal{F}$: If $\mathcal{A}$ loses in the $EU\text{-}CMA$ experiment for ETA, $\mathcal{F}$ also loses in the $EU\text{-}CMA$ experiment for *Schnorr*, and therefore $\mathcal{F}$ *aborts* and return 0. Otherwise, if $(M^*||j||x_j^*) \notin \overrightarrow{\mathcal{L}}$ then $\mathcal{F}$ *aborts* and returns 0 (i.e., $\mathcal{A}$ wins the experiment without querying $RO(.)$ oracle).
  Otherwise, given ETA forgery $(M^*, \sigma^* = \langle s_j^*, x_j^*, j\rangle)$ on PK, $\mathcal{F}$ continues as follows: $\mathcal{F}$ sets the *Schnorr forgery* on public key Y as $(M^*||j||x_j^*, \gamma^* = \langle s^*, e^*\rangle)$, where $s^* = s_j^*$ and $e^* = \overrightarrow{\mathcal{L}'}[i]$ such that $\exists i, 0 \le i \le K' - 1 : (M^*||j||x_j^*) = \overrightarrow{\mathcal{L}}[i]$ and $0 \le j \le K - 1$. By Definition 2, $\mathcal{F}$ checks if γ^* is a valid Schnorr signature and it has not been queried to $Schnorr.Sig_y(.)$. That is, $R_j = Y^{e^*} \cdot \alpha^{s^*}$ and $(M^*||j||x_j^*) \notin \{d_0, \ldots, d_{K-1}\}$ hold (*Setup phase step-i*). If these conditions hold then $\mathcal{F}$ wins the $EU\text{-}CMA$ experiment for *Schnorr* and returns 1. Otherwise, $\mathcal{F}$ loses and returns 0.

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

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

• *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, $Pr[E2|\overline{E1}] = Adv_{ETA}^{EU\text{-}CMA}(t, K', K)$.

• *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 ETA 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) $\mathcal{F}$'s forgery is valid and non-trivial. The probability that $(M^*||j||x_j^*) \in (d_0, \ldots, d_{K-1})$ (i.e., $\mathcal{F}$'s forgery is trivial) is $\frac{K}{2^{|q|}}$, which is negligible in terms of κ, where $|q| = 2\kappa$. Since (s^*, e^*) is valid on $(Y, R_j) \in PK$, it is also a valid signature on Schnorr public key Y. Hence, $Pr[\overline{E3}|\overline{E1} \wedge E2] = Adv_{ETA}^{EU\text{-}CMA}(t, K', K)$.

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

$$Adv_{ETA}^{EU\text{-}CMA}(t, K', K) \quad \le \quad Adv_{Schnorr}^{EU\text{-}CMA}(t', K),$$

In this experiment, the running time of $\mathcal{F}$ is that of $\mathcal{A}$ plus the time it takes to respond K' $RO(.)$ queries and K $ETA.Sig$ queries (RNG denotes the cost of drawing a random number). □

5. PERFORMANCE ANALYSIS AND COMPARISON

ETA is implemented on an Elliptic Curve (EC) [7], which offers small key and signature sizes.

The private key and signature sizes of ETA are constant as $2|q|$ and $|q| + \kappa$, respectively. The public key size is $|q| + |H| \cdot O(K)$.

In ETA, the key generation is performed *offline* (before the deployment) by a key generation center, whose cost is $(H + EMul)$ $O(K)$. The signature generation is performed *online* by a resource-constrained device, whose cost is $2H + Mulq + Addq$. The signature verification is performed by a resourceful device, whose cost is $H + 1.3 \cdot EMul$.

Table 1 (see Section 1) compares ETA and its counterparts with respect to their storage, communication and computational costs. Table 2 numerically compares the storage/communication overhead of ETA with that of its counterparts for the growing K values.

ETA has *the smallest signature size* (i.e., 240 bits) among all of its counterparts. That is, the signature size of ETA is 6, 8, 1.3 and orders of magnitudes times smaller than that of HORS/HORSE, online/offline signatures, ECDSA/token-ECDSA and HORS++, respectively. ETA also has a small-constant private key size (i.e., 320 bits), which is one of the smallest among all of its counterparts. For instance, the private key size of ETA is *several orders of magnitude smaller* than that of token-ECDSA, HORSE, HORS, HORS++ and online/offline signatures ($K = 10^4$). Similarly, the size of K-time public key is also much smaller than that of HORS, HORSE and HORS++. However, it is larger than that of ECDSA, token-ECDSA and online/offline signatures.

We prototyped ETA on a computer with an Intel(R) Core(TM) i7 Q720 at 1.60GHz CPU and 2GB RAM running Ubuntu 10.10 using MIRACL library [24] (also see item (ii) under Table 1 for the recommended key/parameter sizes used in this comparison). ETA does not require any expensive operation at the signer side and therefore is much more efficient than traditional signature schemes. For instance, the estimated execution time of ETA is 4 μsec, while it is 1330 μsec for ECDSA. ETA is also faster than HORSE (15 μsec) and token-ECDSA (6 μsec), but it is only 3μsec slower than HORS and HORS++.

The signature verification of ETA is more efficient than that of online/offline signature [23] (1774 μsec) and is equally efficient to that of ECDSA and token-ECDSA (1554 μsec). Note that in ETA system model, *verifiers are resourceful* and therefore they can easily perform a double-point scalar multiplication. ETA is less efficient than HORS, HORSE and HORS++, since these schemes only rely on hash functions for the signature verification.

The computational cost of *offline phase* (i.e. the key generation phase) is $O(K)$ expensive operations for ETA, online/offline signatures and token-ECDSA. ECDSA requires a single expensive operation, while HORS, HORSE and HORS++ require $H \cdot O(K)$ operations. We focus on the *online computational efficiency* as it is the most important metric for our envisioned applications.

In summary, ETA *is the only scheme among its counterparts that achieves the signer computational, storage and communication efficiency at the same time*.

6. CONCLUSION

In this paper, we proposed a new signature scheme called ETA, which achieves several desirable properties needed for resource-constrained devices. That is, ETA does not require any expensive operation at the signer side and therefore is much more computationally efficient than traditional PKC-based schemes. ETA has a small-constant private key and signature sizes, which are much more efficient than that of token-based signatures, online/offline signatures and multiple-time signatures. In ETA, the size of K-time public key is much smaller than that of previous multiple-time signatures. Moreover, ETA is packet loss tolerant and it achieves immediate verification property. ETA is also proven to be secure against the adaptive chosen message attacks (in random oracle model). Therefore, ETA is an ideal choice for providing authentication and integrity services for resource-constrained devices.

7. REFERENCES

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

[2] F. Baldimtsi, G. Hinterwalder, A. Rupp, A. Lysyanskaya, C. Paar, and W. P. Burleson. Pay as you go. In *Proc. of HotPETs*, July 2012.

[3] 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)*, pages 62–73, NY, USA, 1993. ACM.

[4] D. Catalano, M. D. Raimondo, D. Fiore, and R. Gennaro. Off-line/on-line signatures: Theoretical aspects and experimental results. Public Key Cryptography (PKC), pages 101–120. Springer-Verlag, 2008.

[5] 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*, pages 685–698, 2011.

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

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

[8] 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*, IPSN '08, pages 445–456, Washington, DC, USA, 2008. IEEE Computer Society.

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

[10] L. Lamport. Constructing digital signatures from a one-way function. Technical report, October 1979.

[11] Q. Li and G. Cao. Multicast authentication in the smart grid with one-time signature. *IEEE Transactions on Smart Grid*, 2(4):686 –696, December 2011.

[12] Y. Liu, M. K. Reiter, and P. Ning. False data injection attacks against state estimation in electric power grids. In *ACM Conference on Computer and Communications Security*, pages 21–32, 2009.

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

[14] Z. Lu, X. Lu, W. Wang, and C. Wang. Review and evaluation of security threats on the communication networks in the smart grid. In *Military Communication Conference (MILCOM)*, November 2010.

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

[16] 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)*, pages 77–85, 1994.

[17] W.D. 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*, volume 1, pages 129 – 134 Vol.1, april 2004.

[18] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2000.

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

[20] 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)*, pages 144–153. Springer-Verlag, 2002.

[21] R.L. Rivest, A. Shamir, and L.A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[22] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

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

[24] Shamus. Multiprecision integer and rational arithmetic c/c++ library (MIRACL). `http://www.shamus.ie/`.

[25] Q. Wang, H. Khurana, Y. Huang, and K. Nahrstedt. Time valid one-time signature for time-critical multicast data authentication. In *INFOCOM 2009, IEEE*, April 2009.

[26] A. A. Yavuz and P. Ning. Self-sustaining, efficient and forward-secure cryptographic constructions for unattended wireless sensor networks. *Ad Hoc Networks*, 10(7):1204–1220, 2012.

[27] 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.