ARIS: Authentication for Real-Time IoT Systems

Rouzbeh Behnia
Oregon State University
Corvallis, Oregon
behniar@oregonstate.edu

Muslum Ozgur Ozmen
Oregon State University
Corvallis, Oregon
ozmenmu@oregonstate.edu

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

***Abstract*—Efficient authentication is vital for IoT applications with stringent minimum-delay requirements (e.g., energy delivery systems). This requirement becomes even more crucial when the IoT devices are battery-powered, like small aerial drones, and the efficiency of authentication directly translates to more operation time. Although some fast authentication techniques have been proposed, some of them might not fully meet the needs of the emerging delay-aware IoT.**

In this paper, we propose a new signature scheme called ARIS that pushes the limits of the existing digital signatures, wherein a commodity hardware can verify 83,333 signatures per second. ARIS also enables the fastest signature generation along with the lowest energy consumption and end-to-end delay among its counterparts. These significant computational advantages come with a larger storage requirement, which is a highly favorable trade-off for some critical delay-aware applications. These desirable features are achieved by harnessing message encoding with cover-free families and special elliptic curve based one-way function. We prove the security of ARIS under the hardness of the elliptic curve discrete logarithm problem in the random oracle model. We provide an open-sourced implementation of ARIS on commodity hardware and 8-bit AVR microcontroller for public testing and verification.

***Keywords*—Authentication; Internet of Things; digital signatures; delay-aware systems; applied cryptography.**

I. Introduction

IoT systems often need authentication for applications that need to verify a large volume of incoming transactions or commands. While symmetric key primitives (e.g., HMAC) can provide very fast authentication, they fail to offer non-repudiation which is often vital for these applications. For instance, Visa handles millions of transactions every day [1]. Each transaction corresponds to multiple authentications of the user's request and card information being done on merchant's side, payment gateway and credit card issuer [2]. Therefore, creating more efficient solutions can significantly reduce the overall authentication overhead of such systems that results in substantial financial gains.

The need for efficient authentication becomes even more imperative for applications in which IoT devices must operate in safety-critical settings and/or with battery limitations. For instance, battery-powered aerial drones [3] might communicate and authenticate streams of commands and measurements with an operation center in a short period of time. A fast and energy-efficient authentication can improve flight and response time of such aerial drones [4]. Other IoT applications such as smart grid systems, which involve battery-powered sensors, will also benefit from fast and energy-efficient digital signatures which minimize the authentication delay/overhead and improve the operation time of the sensors [5]. Additionally, in vehicular networks, safety significantly hinges on the end-to-end delay [6], and therefore attaining a signature scheme with the lowest end-to-end delay is always desired.

A. Our Contributions

In this paper, we propose a new efficient signature scheme called **ARIS**. **ARIS** makes use of an Elliptic Curve Discrete Logarithm Problem (ECDLP) based one-way function and exploits the homomorphic properties of such functions to (i) linearly add the private key elements to attain a shorter signature and (ii) mask this addition with a one-time randomness r to achieve a (polynomially-bounded) multiple-time signature scheme. We outline the main properties of **ARIS** as below.

- *Fast Verification:* **ARIS** provides the fastest signature verification among its counterparts. More specifically, **ARIS** pushes the limits of elliptic curve (EC) based signature schemes by providing nearly $2\times$ faster verification as compared to its fastest counterpart [7].
- *Fast Signing:* The signature generation of **ARIS** avoids expensive computations such as fixed-based scalar multiplication. Therefore, **ARIS** achieves 33% faster signing as compared to its fastest counterpart [7].
- *Low End-to-End Delay:* Due to having the fastest signature generation and verification algorithms, **ARIS** achieves nearly 40% lower end-to-end delay, as compared to its fastest counterpart [7]. This might encourage the potential adoption of **ARIS** for applications that require delay-aware authentication.
- *Energy Efficiency:* By avoiding any computationally expensive operation in the signing and verification algorithms, **ARIS** achieves the lowest energy consumption as compared to its state-of-the-art efficient counterparts. Specifically, as shown in Figure 1, the verification algorithm in **ARIS** attains 40% lower energy consumption as compared to its most energy efficient counterpart. This makes **ARIS** potentially suitable for IoT applications wherein the battery-powered devices authenticate telemetry and commands (e.g., aerial drones).
- *Tunable Parameters:* **ARIS** enjoys from a highly tunable parameters. This allows **ARIS** to be instantiated with different properties for different applications. For instance, the parameters set that we considered for our

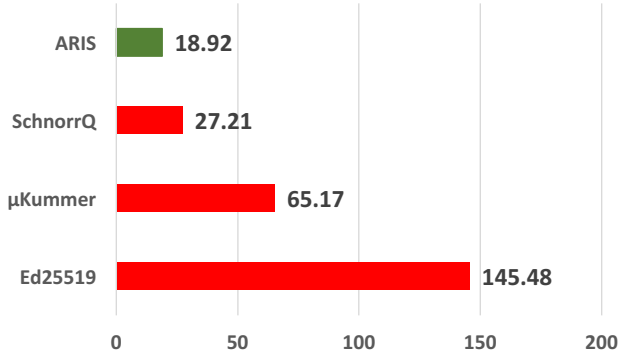

Fig. 1: Energy consumption (mJ) for signature generation of **ARIS** and its counterparts on AVR microcontroller

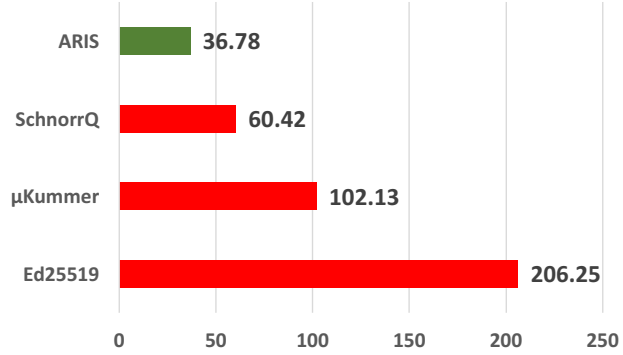

Fig. 2: Energy consumption (mJ) for signature verification of **ARIS** and its counterparts on AVR microcontroller

implementation on AVR microcontroller enjoys from a smaller public key and private key pair, and if the same scheme is implemented on commodity hardware, it can enjoy from a faster signature generation ($2\times$ faster than the scheme in [7]) by incurring a few microseconds on the verification algorithm.

Limitations: All of the desired properties and efficiency gains in **ARIS** come with the cost of larger key sizes. For instance, in the verification efficient instantiation of **ARIS** (as in Table I), which has the largest key sizes, the size of the public key and private key could be as large as 32KB. However, this can be decreased to 16KB and 8KB for the private key and public key sizes (respectively) while still maintaining the fastest signature generation and verification algorithms among its counterparts. We have shown that even with these parameters sizes, **ARIS** can be implemented on 8-bit AVR while enjoying from the most computation and energy efficient algorithms as shown in Figure 1, Figure 2 and Table II.

II. Related Work

One-time signatures (e.g. HORS [8]) have been proposed to offer fast signing and verification. Following HORS, many schemes with different performance and security trade-offs such as time valid one-time signatures (i.e., TV-HORS [9]) have been proposed. However, these schemes suffer from security and performance penalties incurred due to the need for time-synchronization and their low tolerance for packet loss. Multiple-time hash-based signatures (e.g., XMSS [10]) utilize Merkle-Tree and can sign multiple messages by keeping the signer's state. Recently, stateless variations (e.g., SPHINCS [11]) have been proposed, however such schemes suffer from large signatures ($\approx$ 41 KB) and slow signing algorithms.

Some methods rely on shifting the computation extensive operations to the key generation algorithms [12]. For instance, the Rapid Authentication scheme proposed in [13] exploits the aggregatable property of the underlying signature schemes and obtains fast signature generation where the signer only aggregates tokens (i.e., signatures of the underlying scheme) during the online phase that are precomputed in the key generation (offline). However, RA requires messages to be in a predefined and fixed-length format. A recently proposed scheme called CEDA [14] exploits the aggregatable property of RSA-based one-way permutation functions and message encoding (as proposed in [8]) to attain efficient signing. However, the large parameter sizes not only incur very large public keys but also make the exponentiations that takes place during signature generation and verification quite costly. Therefore CEDA, while being among the most efficient schemes, does not surpass the latest implementations of signatures on fast elliptic curves.

In the line of proposing fast elliptic curves, Renes et al. [15] presented an efficient instantiations of the scheme in [16] based on Kummer software that shows significant performance gains as compared to its base scheme [16]. In 2016, Costello et al. [7] proposed a new implementation of [16] based on another elliptic curve called FourQ which shows to even outperform the implementation in [15].

III. Preliminaries

Notation. Given two primes p and q we define a finite field $\mathbb{F}_q$ and a group $\mathbb{Z}_p$. We also work on $E(\mathbb{F}_q)$ as an elliptic curve over $\mathbb{F}_q$. We commonly denote $P \in E(\mathbb{F}_q)$ as a generator of the points on the curve. $x \xleftarrow{\$} S$ denotes randomly selecting x from a set S. We denote scalars as small letters (e.g., x) and points on curve as capital letters (e.g., P). We denote tables/matrices as bold capital letters (e.g., $\mathbf{P}$). We define the bit-length of a variable as $|x|$, i.e., $|x| = \log_2 x$. Scalar and point multiplication is denoted as xP. We define two Pseudo Random Functions $\mathtt{PRF}_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $\mathtt{PRF}_2 : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ and three hash function $\mathtt{H}_1 : \{0,1\}^* \times \mathbb{Z}_p \rightarrow \{0,1\}^{l_1}$, $\mathtt{H}_2 : E(\mathbb{F}_q) \rightarrow \{0,1\}^{l_2}$, and $\mathtt{H}_3 : \{0,1\}^* \times \{0,1\}^{l_2} \rightarrow \{0,1\}^{l_1}$ for some integers l_1 and l_2, to be defined in Section VI.

Definition 1. (*Elliptic Curve Discrete Logarithm Problem*) For $E(\mathbb{F}_q)$ as an elliptic curve over a finite field $\mathbb{F}_q$, given $P, Q \in E(\mathbb{F}_q)$, the Elliptic Curve Discrete Log Problem (ECDLP) asks to find $k \in \mathbb{Z}_p$, if it exists, such that $Q = kP$.

Definition 2. A signature scheme consists of three algorithms $\mathtt{SGN} = (\mathtt{Kg}, \mathtt{Sig}, \mathtt{Ver})$ defined as follows.

– $(sk, pk) \leftarrow \texttt{SGN.Kg}(1^\kappa)$: Given the security parameter κ, it outputs the private and public key pair (sk, pk).
– $\sigma \leftarrow \texttt{SGN.Sig}(m, sk)$: Given the message m and the signer's private key sk, it outputs the signature σ.
– $\{0,1\} \leftarrow \texttt{SGN.Ver}(m, \sigma, pk)$: Given a message-signature pair (m, σ), and the claimed signer's public key pk, it outputs a decision bit $d \leftarrow \{0,1\}$.

In the following definition, we define the security of signature schemes based on the methodology proposed in [17]. After the initialization phase i.e., $\texttt{SGN.Kg}(\cdot)$, The adversary $\mathcal{A}$ is given access to the signature generation oracle. $\mathcal{A}$ wins, if it outputs a *valid* message-signature pair (that was not previously outputted from the sign oracle) after making polynomially-bounded number of queries.

Definition 3. Existential Unforgeability under Chosen Message Attack (EU-CMA) experiment $Expt^{EU-CMA}_{\texttt{SGN}}$ is defined as follows.

– $(sk, pk) \leftarrow \texttt{SGN.Kg}(1^\kappa)$
– $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\texttt{SGN.Sig}(\cdot)}(pk)$
– If $1 \leftarrow \texttt{SGN.Ver}(m^*, \sigma^*, pk)$ and m^* was not queried to $\texttt{SGN.Sig}(\cdot)$, return 1, else, return 0.

The EMU-CMA advantage of $\mathcal{A}$ is defined as $Adv^{\textit{EU-CMA}}_{\texttt{SGN}} = \Pr[Expt^{EU-CMA}_{\texttt{SGN}} = 1]$.

IV. Proposed Scheme

ARIS leverages the homomorphic property of its underlying ECDLP-based one-way function, which is due to the exponent product of powers property, to achieve (polynomially-bounded) multiple-time signatures from the one-time signature scheme proposed in [18], with more compact signatures. More specifically, in **ARIS**, the private key consists of t randomly generated values x_i (generated using a κ bit seed z) and the corresponding public key consists of all $Y_i \leftarrow x_i P$ for $i \in \{1, \ldots, t\}$.

To sign a message, the signer obtains k indexes $(i_1, \ldots, i_k)$ by hashing the message (and a random input), uses the indexes $(i_1, \ldots, i_k)$ to retrieve the corresponding private key elements (i.e., x_{i_j} where $j \in \{1, \ldots, k\}$) and sums them along with a one-time randomness r. The signature consists of s and h, which is obtained by applying the hash function $\texttt{H}_2(\cdot)$ on R, that is computed as the output of applying the one-way function on the one-time randomness r.

Verification takes place by computing the summation of the corresponding public key elements (i.e., Y_{i_j}) and their subtraction from the output of the ECDLP-based one-way function applied on s. The verifier outputs *valid* if the subtraction yields the same value of R as computed in the signature generation. Additionally, **ARIS** uses the BPV method in [19] to convert an EC scalar multiplication to only k (where $k = 18$ or $k = 28$ for our proposed parameter sets) EC point additions with the cost of storing a small, constant-size table.

Our scheme consists of the following algorithms. $(sk, pk) \leftarrow \textbf{ARIS}.\texttt{Kg}(1^\kappa)$: Given the security parameter κ, this algorithm selects parameters (t, k) such that $\binom{t}{k} \geq 2^\kappa$ and $z \xleftarrow{\$} \mathbb{Z}_p$ and works as follows.

1) Compute $x_i \leftarrow \texttt{PRF}_1(z, i)$ and $Y_i \leftarrow x_i P$ for $i \in \{1, \ldots, t\}$ and set $\mathbf{Y} \leftarrow \{Y_i\}_{i=1}^t$.
2) Compute $r_i \leftarrow \texttt{PRF}_2(z, i)$ and $R_i \leftarrow r_i P$ for $i \in \{1, \ldots, t\}$ and set $\mathbf{R} \leftarrow \{R_i\}_{i=1}^t$.
3) Output $pk \leftarrow \mathbf{Y}$ and $sk \leftarrow (z, \mathbf{R})$ as the public key and private key, respectively.

$\sigma \leftarrow \textbf{ARIS}.\texttt{Sig}(m, sk)$: Given a message $m \in \{0,1\}^*$ to be signed, this algorithm works as follows.

1) Compute $(i'_1, \ldots, i'_k) \leftarrow \texttt{H}_1(m, z)$ where $|i'_j| \leq |t|$ for $j \in \{1, \ldots, k\}$.
2) Compute $r_{i'} \leftarrow \texttt{PRF}_2(z, i'_j)$ for $j \in \{1, \ldots, k\}$, set $r \leftarrow \sum_{i'=1}^k r_{i'}$.
3) Retrieve $R_{i'} \leftarrow \mathbf{R}[i'_j]$ for $j \in \{1, \ldots, k\}$, compute $R \leftarrow \sum_{i'=1}^k R_{i'}$ and $h \leftarrow \texttt{H}_2(R)$.
4) Compute $(i_1, \ldots, i_k) \leftarrow \texttt{H}_3(m, h)$ (where $|i_j| \leq |t|$) and $x_i \leftarrow \texttt{PRF}_1(z, i_j)$ for $j \in \{1, \ldots, k\}$.
5) Compute $s \leftarrow r - \sum_{i=1}^k x_i$ and output $\sigma \leftarrow (s, h)$.

$\{0,1\} \leftarrow \textbf{ARIS}.\texttt{Ver}(m, \sigma, pk)$: Given a message-signature pair (m, σ) and pk, this algorithm works as follows.

1) Parse $(s, h) \leftarrow \sigma$ and compute $(i_1, \ldots, i_k) \leftarrow \texttt{H}_3(m, h)$, where $|i_j| \leq |t|$ for $j \in \{1, \ldots, k\}$.
2) Retrieve $Y_i \leftarrow \mathbf{Y}[i_j]$ for $j \in \{1, \ldots, k\}$) and set $Y \leftarrow \sum_{i=1}^k Y_i$.
3) Compute $R' \leftarrow sP + Y$ and check if $\texttt{H}_2(R') = h$ holds output *valid*, and *invalid* otherwise.

V. Security Analysis

We prove that **ARIS** is EU-CMA secure, as defined in Definition 3, in the Random Oracle Model (ROM) [20]. The proof uses the Forking Lemma [21].

Theorem 1. *In the ROM, if an adversary $\mathcal{A}$ can (q_S, q_H)-break the EU-CMA security of **ARIS** after making q_H and q_S random oracles and signature queries, respectively; then we can build another algorithm $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine and can solve an instance of the ECDLP (as defined in Definition 1).*

Proof. We let $Y^* \xleftarrow{\$} E(\mathbb{F}_q)$ be an instance the ECDLP for algorithm $\mathcal{B}$ to solve. On the input of Y^* and $z \xleftarrow{\$} \mathbb{Z}_p$, $\mathcal{B}$ works as follows.

Setup: $\mathcal{B}$ keeps three lists $\mathcal{L}_i$ for $i \in \{1, 2, 3\}$ to keep track of the outputs of the random oracles and a list $\mathcal{L}_m$ to store the messages submitted to the sign oracle. $\mathcal{B}$ sets up the random oracle $RO\text{-}Sim(\cdot)$ to handle the hash functions and generates the users' public keys as follows.

- *Setup* $RO\text{-}Sim(\cdot)$: $\mathcal{B}$ implements $RO\text{-}Sim(\cdot)$ to handle queries to hash functions $\texttt{H}_1, \texttt{H}_2$ and $\texttt{H}_3$, which are modeled as random oracles, as follows.
  1) $\alpha_1 \leftarrow RO\text{-}Sim(m, z, \mathcal{L}_1)$: If $(m, z) \in \mathcal{L}_1$, it returns the corresponding value α_1. Else, it returns $\alpha_1 \xleftarrow{\$} \{0,1\}^{l_1}$ as the answer and adds (m, z, α_1) to $\mathcal{L}_1$.
  2) $\alpha_2 \leftarrow RO\text{-}Sim(R, \mathcal{L}_2)$: If $R \in \mathcal{L}_2$, it returns the corresponding value α_2. Else, it returns $\alpha_2 \xleftarrow{\$} \{0,1\}^{l_2}$ as the answer and adds (R, α_2) to $\mathcal{L}_2$.

3) $\alpha_3 \leftarrow RO\text{-}Sim(m, h, \mathcal{L}_3)$: If $(m, h) \in \mathcal{L}_3$, it returns the corresponding value α_3. Else, it returns $\alpha_3 \xleftarrow{\$} \{0,1\}^{l_1}$ as the answer and adds (m, h, α_3) to $\mathcal{L}_3$.

- *Setup Public Key*: Given the parameters (p, q, P, t, k), $\mathcal{B}$ works as follows to generate the user public key.
  1) Select $j \xleftarrow{\$} [1, t]$ and sets the challenge public key element $Y_j \leftarrow Y^*$.
  2) Generate $x_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in \{1, \ldots, t\}$ and $i \neq j$.
  3) Compute $Y_i \leftarrow x_i P$ for $i \in \{1, \ldots, t\}$ and $i \neq j$.
  4) Set $sk \leftarrow \{x_i\}_{i=1, i\neq j}^{t}$ and $pk \leftarrow \{Y_1, \ldots, Y_t\}$.

$\mathcal{A}$'s Queries: $\mathcal{A}$ queries the hash functions $\mathtt{H}_i$ for $i \in \{1, 2, 3\}$ and the sign oracle for up to q_H and q_S times, respectively. $\mathcal{B}$ works as follows to handle these queries.

- *Hash Queries:* $\mathcal{A}$'s queries to hash functions $\mathtt{H}_1, \mathtt{H}_2$ and $\mathtt{H}_3$ are handled by the $RO\text{-}Sim(\cdot)$ function described above.
- *Signature Queries:* $\mathcal{B}$ works as follows to answer $\mathcal{A}$'s signature query on message m. If $m \in \mathcal{L}_m$, $\mathcal{B}$ retrieves the corresponding signature from $\mathcal{L}_m$ and returns to $\mathcal{A}$. Else, if $m \notin \mathcal{L}_m$, it works as follows.
  1) Select $s \xleftarrow{\$} \mathbb{Z}_p$ and compute $S \leftarrow sP$.
  2) Select k indexes $(i_1, \ldots i_k) \xleftarrow{\$} [1, \ldots, t]$.
  3) Set $R \leftarrow S - \sum_{i=1}^{k} Y_i$ and $\alpha_2 \leftarrow \{0,1\}^{l_2}$ and add (R, α_2) to $\mathcal{L}_2$.
  4) If $(\langle i_1, \ldots i_k \rangle, h) \in \mathcal{L}_3$ abort. Else, add $(m, h, \langle i_1, \ldots i_k \rangle)$ to $\mathcal{L}_3$.
  5) Output $\sigma = (s, h)$ to $\mathcal{A}$ and add $(m, \sigma) \in \mathcal{L}_m$.

$\mathcal{A}$'s Forgery: Eventually, $\mathcal{A}$ outputs a forgery $\sigma^* = (s^*, h^*)$ on message m^* and public key pk. Following the EU-CMA definition (as in Definition 3), $\mathcal{A}$ only wins the game if $\mathbf{ARIS}.\mathtt{Ver}(m^*, \sigma^*, pk)$ returns $valid$ and m^* was never submitted to signature queries in the previous stage (i.e., $m^* \notin \mathcal{L}_m$).

Solving the ECDLP: If $\mathcal{A}$ does not output a valid forgery before making q_H hash queries and q_S signature queries, $\mathcal{B}$ also fails to solve the instance of ECDLP. Otherwise, if $\mathcal{A}$ outputs a valid forgery $(m^*, \sigma^* = \langle s^*, h^* \rangle)$, using the forking lemma, $\mathcal{B}$ rewinds $\mathcal{A}$ with the same random tape as in [21], to get a second forgery $(m', \sigma' = \langle s', h' \rangle)$ where, with an overwhelming probability $s^* \neq s'$ and $h^* = h'$. Based on [21, Lemma 1], $\mathtt{H}_3(m^*, h^*) \neq \mathtt{H}_3(m^*, h')$, therefore, given $(m^*, h^*) \in \mathcal{L}_3$ and $(m^*, h') \in \mathcal{L}_3$, $\mathcal{B}$ can solve a random instance of the ECDLP problem (i.e., Y^*) if one of the following conditions hold.

- **Case 1:** For $(i_1^*, \ldots, i_k^*) \leftarrow \mathtt{H}_3(m^*, h^*)$ and $(i_1', \ldots, i_k') \leftarrow \mathtt{H}_3(m^*, h')$ we have $j \in (i_1^*, \ldots, i_k^*)$ and $j \notin (i_1', \ldots, i_k')$.
- **Case 2:** For $(i_1^*, \ldots, i_k^*) \leftarrow \mathtt{H}_3(m^*, h^*)$ and $(i_1', \ldots, i_k') \leftarrow \mathtt{H}_3(m^*, h')$ we have $j \notin (i_1^*, \ldots, i_k^*)$ and $j \in (i_1', \ldots, i_k')$.

If any of the above cases holds, $\mathcal{B}$ works as follows. If Case 1 holds, $x_j \leftarrow s^* - \sum_{\eta=1, \eta\neq j}^{k} x_{i_\eta^*} - s' - \sum_{\eta=1}^{k} x_{i_\eta'} \mod p$. Else, if Case 2 hold, $x_j \leftarrow s' - \sum_{\eta=1, \eta\neq j}^{k} x_{i_\eta'} - s^* - \sum_{\eta=1}^{k} x_{i_\eta^*} \mod p$.

□

VI. Performance Evaluation

We have fully implemented **ARIS** on Four$Q$ curve [22] which is known to be the fastest EC that provides 128-bits of security. We provide implementations of **ARIS** on both commodity hardware and 8-bit microcontroller to evaluate its performance since most IoT applications are comprised of them both (e.g., commodity hardware as servers or control centers and microcontrollers as IoT devices connected to sensors). We compare the performance of **ARIS** with state-of-the-art digital signature schemes for both of these platforms, in terms of computation, storage and communication. Our implementation is open-sourced in the following link.

https://github.com/rbehnia/ARIS

A. Performance on Commodity Hardware

1) Hardware Configurations: We used a laptop equipped with Intel i7 Skylake processor @ 2.60 GHz and 12 GB RAM.

2) Software Libraries: We implemented **ARIS** using the open-sourced Four$Q$ implementation [22], that offers the fastest EC operations, specifically EC additions that is critical for the performance of **ARIS**. We used an Intel processor for our commodity hardware and leveraged Intel intrinsics to optimize our implementation. Specifically, we implemented our $\mathtt{PRF}$ functions with Intel intrinsics (AES in counter mode). We used blake2 as our hash function [23] due to its efficiency.

We ran the open-source implementations of our counterparts on our hardware to compare their performance with **ARIS**.

3) Parameter Choice: Since we implement **ARIS** on Four$Q$ curve, we use its parameters given in [22], which provide 128-bit security. Other than the curve parameters, the choice of t, k also plays a crucial role for the security of **ARIS**. Specifically, k-out-of-t combinations should also provide 128-bit security to offer this level of security overall. On the other hand, we can tune these parameters to achieve our desired security level with different performance trade-offs. If we increase t and decrease k, this results in a larger storage with faster computations, and vice versa. For our commodity hardware implementation, we choose $t = 1024$ and $k = 18$, that we believe offers a reasonable trade-off between storage and computation as well as offering the desired 128-bit security level. We set $l_1 = 180$ and $l_2 = 256$.

4) Experimental Results: We present the results of our experiments in Table I. We observe that **ARIS** offers very fast signature generation and verification. It only takes 9 microseconds to generate a signature and 12 microseconds to verify it. This is the fastest among our counterparts, where the closest is the SchnorrQ. Furthermore, if we use the same parameters set as for the AVR microcontroller, we can further speed up the signature generation to 6.5 microseconds, with the cost of a few microseconds on the verification speed. In SchnorrQ, a scalar multiplication is required in signature generation and a double scalar multiplication in verification. In **ARIS**, EC additions are required for signature generation and verification is done with a scalar multiplication and

TABLE I: Experimental performance comparison of **ARIS** and its counterparts on a commodity hardware

Scheme	Signature Generation Time (μs)	Private Key† (KB)	Signature Size (KB)	Signature Verification Time (μs)	Public Key (KB)	End-to-End Delay (μs)
SPHINCS [11]	13458	1.06	41000	370	1.03	13828
RSA [24]	8083	0.75	0.41	48	0.38	8131
CEDA [14]	55	0.41	0.41	115	384.38	170
ECDSA [25]	725	0.03	0.06	927	0.03	1652
Ed25519 [16]	132	0.03	0.06	335	0.03	467
Kummer [15]	23	0.03	0.06	38	0.03	61
SchnorrQ [7]	12	0.03	0.06	22	0.03	34
ARIS	**9**	32.03	0.06	**12**	32	**21**

† System wide parameters (e.g., p,q,α) for each scheme are included in their corresponding codes, and private key size denote to specific private key size.

TABLE II: Experimental performance comparison of **ARIS** and its counterparts on 8-bit AVR microcontroller

Scheme	Signature Generation Time (s)	Private Key (KB)	Signature Size (KB)	Signature Verification Time (s)	Public Key (KB)	End-to-End Delay (s)
ECDSA [25]	1.77	0.03	0.06	1.80	0.03	3.57
Ed25519 [16], [26]	1.45	0.03	0.06	2.06	0.03	3.51
μKummer [15], [27]	0.65	0.03	0.06	1.02	0.03	1.67
SchnorrQ [7], [28]	0.27	0.03	0.06	0.60	0.03	0.87
ARIS	**0.19**	16	0.06	**0.37**	8	**0.56**

EC additions. This corresponds to a 33% faster signature generation and 83% faster verification for **ARIS**, compared to SchnorrQ. Therefore, we believe **ARIS** can be an ideal alternative for real-time applications.

ARIS signature size is the same with its EC-based counterparts [25], [16], [15], [7] , that is significantly lower than its RSA-based and hash-based counterparts [24], [14], [11]. On the other hand, **ARIS** comes with a larger private and public key, that is 32 KB.

B. Performance on 8-bit AVR

1) Hardware Configurations: We used an 8-bit AVR ATmega 2560 microcontroller as our IoT device to implement **ARIS**. ATmega 2560 is equipped with 256 KB flash memory, 8 KB SRAM and 4 KB EEPROM, with a maximum clock frequency of 16 MHz. ATmega 2560 is extensively used in practice for IoT applications (especially in medical implantables) due to its energy efficiency [29].

2) Software Libraries: We implemented **ARIS** on ATmega 2560 using the 8-bit AVR implementation of Four$Q$ curve [28], that provides the basic EC operations and a blake2 hash function. We implemented our scheme with IAR embedded workbench and used its cycle-accurate simulator for our benchmarks.

As for our counterparts, we used their open-sourced implementations [28], [26], [27], [30]. Note that we only compare **ARIS** with its EC-based counterparts, due to their communication and storage efficiency. Moreover, resource-contrained processors such as ATmega 2560 may not be suitable for heavy computations (e.g., exponentiation with 3072-bit numbers in RSA [24] and CEDA [14]).

3) Parameter Choice: As mentioned, **ARIS** can be instantiated with different t, k values that offers a trade-off between storage and computation. Since ATmega 2560 is a storage-limited device, we select our parameters as $t = 256$ and $k = 28$ to offer storage efficiency. Moreover, this allows us to store the private components (x_i and r_i), instead of deterministically generating them at signature generation, and still have a tolerable storage even for an 8-bit microcontroller. We also set $l_1 \leftarrow 224$ and $l_2 \leftarrow 256$.

4) Experimental Results: Table II shows the performance of **ARIS** compared with its counterparts. The speed improvements of **ARIS** can also be observed for ATmega 2560. **ARIS** is 42% faster in signature generation and 76% faster in signature verification compared to its closest counterpart [7]. This can translate into a significant practical difference when considered real-time applications that require fast authentication. Note that these benchmarks are obtained with a more "storage friendly" parameter choice, and can be further accelerated with different parameter choices where the microcontroller is not memory-constraint.

One may notice that due to our parameter choice, the key sizes in our 8-bit microcontroller implementation are smaller. As aforementioned, this is because we select a different parameter set for t, k. Moreover, we store the private components as well, that correspond the 8 KB of the signer storage. Since we store these keys on the flash memory of ATmega 2560, they only correspond to 6% and 3% of the total memory, for private key and public key, respectively. Therefore, although we have significantly larger keys than our EC-based counterparts, it is still feasible to store them even on highly resource-constrained 8-bit microcontrollers.

5) Energy Efficiency: It is highly desirable to minimize the energy consumption of cryptographic primitives in IoT applications to offer a longer battery life. For microcontrollers, energy consumption of the device can be measured with the formula $E = V * I * t$, where V is voltage, I is current and t is the computation time [31]. Considering that the voltage and the current of a microcontroller are constant when the device is active, the energy consumption linearly increases with the

computation time. Since **ARIS** offers the fastest signature generation and verification, energy consumption of **ARIS** is the lowest among its counterparts, and therefore would be preferred in applications that require longer battery life.

VII. Conclusion

In this paper, we presented a new efficient signature scheme to meet the strict minimum delay requirements of some real-time IoT systems. This is achieved by harnessing the homomorphic property of the underlying ECDLP-based one-way function and the precomputation technique proposed in [19]. Our experimental results showed that the proposed scheme outperforms its state-of-the-art counterparts in signing and verification speed as well as in energy efficiency. The proposed scheme is shown to be secure, in the Random Oracle Model, under the harnesses of the ECDLP. We open-sourced our implementation to enable public testing and verification.

Acknowledgment. This work is supported by NSF award #1652389.

References

[1] J. Steele. (2018) Debit card statistics. [Online]. Available: https://www.creditcards.com/credit-card-news/debit-card-statistics-1276.php

[2] O. Papadimitriou. (2009) How credit card transaction processing works: Steps, fees & participants. [Online]. Available: https://wallethub.com/edu/credit-card-transaction/25511/

[3] J. Won, S.-H. Seo, and E. Bertino, "A secure communication protocol for drones and smart objects," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. ACM, 2015, pp. 249–260.

[4] M. O. Ozmen and A. A. Yavuz, "Dronecrypt - an efficient cryptographic framework for small aerial drones," in *Milcom 2018 Track 3 - Cyber Security and Trusted Computing (Milcom 2018 Track 3)*, Los Angeles, USA, 2018.

[5] T. Tesfay and J. Y. L. Boudec, "Experimental comparison of multicast authentication for wide area monitoring systems," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, 2017.

[6] "IEEE standard for wireless access in vehicular environments security services for applications and management messages," *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, pp. 1–289, April 2013.

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

[8] 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.

[9] 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.

[10] J. Buchmann, E. Dahmen, and A. Hülsing, "Xmss - a practical forward secure signature scheme based on minimal security assumptions," in *Proceedings of the 4th International Conference on Post-Quantum Cryptography*, ser. PQCrypto'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 117–129.

[11] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "Sphincs: Practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT 2015*, E. Oswald and M. Fischlin, Eds. Springer Berlin Heidelberg, 2015, pp. 368–397.

[12] 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.

[13] A. A. Yavuz, "An efficient real-time broadcast authentication scheme for command and control messages," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1733–1742, Oct 2014.

[14] 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.

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

[16] 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

[17] M. Bellare and P. Rogaway, "The security of triple encryption and a framework for code-based game-playing proofs," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Springer Berlin Heidelberg, 2006, pp. 409–426.

[18] L. Reyzin and N. Reyzin, "Better than biba: Short one-time signatures with fast signing and verifying," in *Information Security and Privacy: 7th Australasian Conference, ACISP 2002 Melbourne, Australia, July 3–5, 2002 Proceedings*, L. Batten and J. Seberry, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 144–153.

[19] 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*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 221–235.

[20] 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*, ser. CCS '93. New York, NY, USA: ACM, 1993, pp. 62–73.

[21] 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.

[22] C. Costello and P. Longa, "Four$Q$: Four-dimensional decompositions on a 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.

[23] 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

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

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

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

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

[28] 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.

[29] 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.

[30] K. MacKay, "micro-ecc: Ecdh and ecdsa for 8-bit, 32-bit, and 64-bit processors," Github Repository. [Online]. Available: https://github.com/kmackay/micro-ecc

[31] G. Ateniese, G. Bianchi, A. Capossele, and C. Petrioli, "Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Precomputation Techniques?" in *Proceedings of the 20th Annual Network & Distributed System Security Symposium, NDSS 2013*, ser. NDSS2013, San Diego, CA, February 24-27 2013.