LoMoS: Less-Online/More-Offline Signatures for Extremely Time-Critical Systems

Ertem Esiner, Utku Tefek, Hasan S. M. Erol, Daisuke Mashima, Binbin Chen, *Member, IEEE*, Yih-Chun Hu, Zbigniew Kalbarczyk, *Member, IEEE*, and David M. Nicol, *Fellow, IEEE*

***Abstract*—The state-of-the-art digital signatures incur undesirable delays, hence are impractical for time-stringent Industrial Control Systems (ICSs). The recent revision to IEC 62351-6 standard stepped back from digital signatures in favor of symmetric key based solutions, thereby sacrificing key properties, e.g., scaling well for multiple destinations, easy key distribution and management, public verifiability, and non-repudiation. Inspired by the Online/Offline signatures, this paper presents a new digital signature model to provide the key properties of digital signatures within the delay requirements, hinting that this step back can be avoided. The Online/Offline signatures concept divides the signature generation into two phases; offline (before the message is given), online (using the outputs of the former for faster signing after the message is given). The conventional solutions following this concept potentially reduce the delay, yet do not meet IEC 61850 delay requirements as they still involve expensive operations in the online phase, and their offline phase hinders throughput. This paper introduces *Less-online/More-offline Signatures* (LoMoS) to enable minimal end-to-end delay and high message throughput. LoMoS entails avoiding expensive operations entirely during the online phase. We present a construction that converts any digital signature scheme into LoMoS, retains its properties, and unlike existing solutions, benefits from shorter messages.**

***Index Terms*—Digital signatures, IEC 61850, IEC 62351, cybersecurity, message authentication, multicast, non-repudiation, real-time communication, smart grid.**

I. Introduction

IN INDUSTRIAL Control Systems (ICSs), ensuring message integrity and authenticity is crucial to ensure reliable and trustworthy operations. Symmetric key based solutions are computationally efficient, therefore, IEC 62351-6 has suggested their use in IEC 61850 in lieu of digital signatures [1]. However, these solutions lack essential features, such as scaling well for multiple, possibly large number of, destinations, the feasibility of key distribution and management, public verifiability, and non-repudiation. This is often overlooked in the literature on secure IEC 61850 communication. Digital signatures are established, widely used technologies that enable verification of source identity and message integrity providing these features. Although digital signatures are suitable for multicast/broadcast communication, their drawback, inherited from underlying asymmetric cryptography, is expensive operations, such as modular exponentiation and elliptic curve scalar multiplication. These computations introduce high latency for signing and verification, especially on resource-constrained devices.

To reduce the computational burden and latency of signature generation after a message is given, the concept of performing prior computations has been studied. This concept divides the signature generation into two phases: an offline phase before the message to be signed is given, and an online phase which utilizes the computations performed in the offline phase for faster signing after the message is given. For example, the first step in ECDSA (Elliptic Curve Digital Signature Algorithm) [2], Fiat-Shamir [3], Schnorr [4], El-Gamal [5], and DSS (Digital Signature Standard) [6] can be computed without knowing the message to be signed. Similarly, online/offline signatures [7], [8] convert any ordinary signature scheme into one with such precomputation, though it dramatically increases the verification overhead of the converted scheme. In all these schemes, the throughput still suffers from high offline computation overhead. Furthermore, their end-to-end delay is affected by the expensive operations during online verification.

In recent years, the demand for low-latency, high-throughput messaging mechanisms has been increasing in many application domains, where even offloading a part of the computational burden to the precomputation phase with the above-mentioned techniques is not enough. Smart grid is one of the most latency-stringent ICSs. IEC 61850 is the increasingly adopted standard for substation automation [9]. According to the IEEE Power and Energy Society guidelines, automated control for protection in a field substation, e.g., control on circuit breakers when overcurrent occurs on a transmission line, requires message delivery time, which consists of network transmission time and all processing time en route,

Manuscript received August 4, 2021; revised November 26, 2021 and February 3, 2022; accepted February 19, 2022. Date of publication March 7, 2022; date of current version June 21, 2022. This work was supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) Programme. Paper no. TSG-01238-2021. *(Corresponding author: Ertem Esiner.)*

Ertem Esiner, Utku Tefek, and Daisuke Mashima are with Advanced Digital Sciences Center, Singapore (e-mail: e.esiner@adsc-create.edu.sg).

Hasan S. M. Erol is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Binbin Chen is with the Singapore University of Technology and Design, Singapore.

Yih-Chun Hu, Zbigniew Kalbarczyk, and David M. Nicol are with the Electrical and Computer Engineering Department, University of Illinois Urbana–Champaign, Champaign, IL 61820 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2022.3156897.

Digital Object Identifier 10.1109/TSG.2022.3156897

including security checking, as low as 2 ms [10], [11]. Because of the tight requirement, IEC 61850 protocols utilize link-layer multicast for sharing such status/event information, exchanged among hundreds of devices [12]. Another challenge is that the measurements of power grid status are sent at a very high rate (i.e., 4,000 messages per second in 50 Hz systems according to the IEC 61850 standard). Forgery or manipulation of these messages would cause undesirable outcomes such as financial loss and massive outages.

The variable and high messaging rate and extremely low delay requirements of multicast communication in smart grid systems are essential but could not be achieved along with the guarantees of public key infrastructure (PKI). This issue has long been discussed by IEC 62351 bodies [1], [13]–[16], and our work addresses this key gap. In order to meet the stringent latency and messaging throughput requirements while retaining the benefits of public key cryptography, we introduce *Less-online/More-offline Signatures* (LoMoS) model, present a concrete construction and its implementation on an embedded system. LoMoS entails simple operations consuming negligible time in the online signing phase, such as memory reads and packet assembly. LoMoS allows the verification to be divided into online/offline phases such that online verification does not perform any expensive operations.

II. Related Work

Message authentication for time stringent systems is a challenging problem [17], [18], especially when multiple destinations are considered. There are extensive studies [19] and comparative analysis of existing solutions [20]. In this section, we investigate solutions in the literature focusing on IEC 61850 to position this paper.

A straightforward approach to enable authentication for time stringent messages is symmetric key based solutions. These solutions rely on message authentication codes (MAC) such as hash-based message authentication code (HMAC) and Advanced Encryption Standard-Galois Message Authentication Code (AES-GMAC) [11], [21], [22]. To make room for the appended MAC, the GOOSE (Generic Object Oriented Substation Event) message structure is extended [14], as per IEC 62351-6. In particular, the MAC value is appended to the *Extension* field of the GOOSE frame, and its length is specified in the *Reserved1* field. These MAC-based solutions are computationally efficient; however, they miss essential features, such as the feasibility of key distribution and management, public verifiability, and non-repudiation. Additionally, when there are multiple destinations for the message, the MAC computation is repeated per destination, increasing the computational burden. Alternatively, a group key implementation can be considered [23], yet it is vulnerable to impersonation attacks. While this provides some baseline security (such as membership proofs), any compromised group member can pretend to be another group member because all the members share the same key. Thus, we need a source of asymmetry between the source and the destinations.

The conventional solution to multicast message authentication is to use digital signatures. If there had been no hardware constraint, a candidate from the public key cryptography domain would have been ECDSA [2], thanks to its relatively efficient signing. In IEC standards (IEC 62351-6 [24]), the use of RSA is suggested for IEC 61850 GOOSE authentication, where the maximum end-to-end delay is supposed to be under 2 ms. While these schemes based on public key cryptography are suitable for multicast scenarios and offer non-repudiation, they rely on computationally expensive operations such as elliptic curve scalar multiplication or modular exponentiation. Reference [16] shows that even an FPGA implementation of RSA signatures is not feasible within this latency constraint. Reference [25] claims to achieve lower delay, but their solution exploits a certain structure in command and control messages. Another promising solution for instant message authentication is one-time signatures (OTS), such as those proposed by Lamport [26] and Rabin [27]. Later in [28], Winternitz OTS (WOTS) Scheme reduced the signature size of OTS by trading off setup and verification time. In the same work, Merkle proposed converting WOTS into a many-time signature by constructing a Merkle Hash Tree (MHT) on multiple WOTS public keys to bind them into a single root value, which serves as the public key for multiple WOTS instances. This method allows signing many messages with multiple one-time key pairs and a single public key, keeping the public key size small compared to constructing multiple independent WOTS instances. Constructing an MHT on Lamport OTS keys, in fact, results in faster setup and verification times than the WOTS variants [29], albeit with a larger communication overhead. Using MHT in OTS/WOTS enables virtually instant signing of multiple messages with a single public key after the message is given. More efficient variants of OTS such as BiBa signature [30], HORS signature [31], [32] have also been proposed. While using MHT to bind multiple public keys together is promising, OTS-based schemes lack a straightforward, efficient extension to benefit from small messages, hence do not achieve the desirable delays and throughput achieved in our LoMoS construction.

Some signature schemes can be partitioned into phases where the first phase does not require the message to be given. For instance, the first steps of El-Gamal [5], DSS, or in particular Digital Signature Algorithm (DSA) [6], and ECDSA [2] are independent of the message to be signed. Online/offline digital signatures, either based on one-time signature schemes [7] discussed above or based on chameleon commitments [8], [33] is a technique to transform every signature scheme to arm them with this feature. In [8], the authors propose a "hash" and "sign" during the offline phase and then "switch" during the online phase. The source first hashes a random message and signs the digest. Later, when the message is given, the source finds a collision (using a trapdoor function) with the said digest, adding a new value to the message to be signed. Then, it sends the new value together with the signature. If a digest is used more than once, an adversary can figure the trapdoor function out and forge signatures. Hence, the source can sign only one message per setup. Also, as discussed in our evaluation, these schemes fall short in meeting stringent latency requirements due to the expensive operations performed in the online phases.

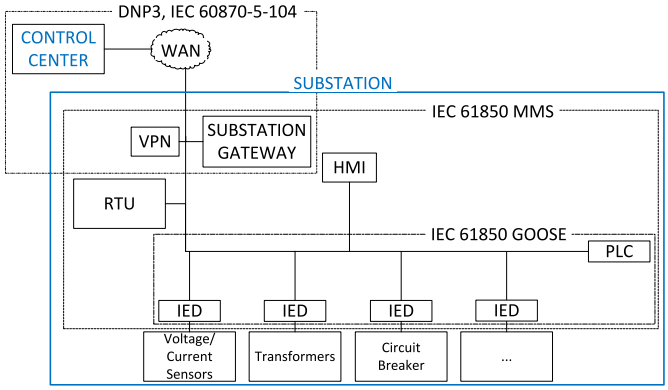


Fig. 1. Substation automation and remote control.

III. Motivating Use Case

In this section, we discuss an electric substation automation system in smart grid regarding its low latency, high message rate, and security requirements, as well as the potential to leverage small message size.

A. IEC 61850 Communication in Smart Grid

Modernized substations in smart grid utilize Ethernet as a medium for connecting intelligent devices, such as programmable logic controllers (PLCs) and intelligent electronic devices (IEDs). Nowadays, gigabit or faster networks are becoming common for connecting hundreds of intelligent devices in a large substation [12], a simplified overview of which is shown in Figure 1. Modern substations use standardized technologies like IEC 60870-5-104 or DNP 3.0 for telecontrol and IEC 61850 for substation automation [34]. For enabling timely communication among the intelligent devices over such a network medium, IEC 61850 GOOSE, and SV (Sampled Value) [9] rely on link-layer multicast.

According to the guideline provided by IEEE Power and Energy Society [10], receiving measurement data and sending protection commands to switchgears (e.g., circuit breakers) must be done within 2ms, inclusive of the transmission time and processing time. In order to ensure reliable delivery of critical information, IEDs send the same information repetitively. While in a normal, stable state, the transmission rate is once in every 1-5 seconds, when any disturbance event occurs, the rate could become as high as once in every 1-2 ms [35]. On the other hand, SV messages, which are used for sharing real-time power grid measurements, are transmitted at a constant but very high rate (e.g., 4,000 messages per second in a power system with a frequency of 50 Hz [36]).

B. Security for IEC 61850 GOOSE and SV Protocols

We consider typical remote attackers for false data/command injection and manipulation attacks, which aim at disabling or misleading automation functionalities [37]. Such remote attackers don't have physical access to substation facility but have a footprint in the system, e.g., via compromised VPN interface [38], or RTU (remote terminal unit) shown in Figure 1 and malware [39] planted on general-purpose/engineering workstations in the control system infrastructure. Such remote attackers could inject arbitrary packets and observe network traffic. For instance, as discussed in [40], an attacker may inject a GOOSE message with a maliciously crafted sequence number and/or status number so that the subsequent legitimate messages are discarded by the destination device, allowing the attacker to send further malicious messages by impersonating the legitimate sender [41], [42]. However, we assume it is not feasible for the remote attackers to manipulate the configuration of ICS devices (e.g., installation of malicious software/firmware) since the configuration of ICS devices is typically done through local, serial connections. Against such a threat model, ensuring the integrity and authenticity of messages is imperative. In addition, substation implementations lack network monitoring and log audit; therefore, the source of the malicious messages cannot be traced [22]. Hence, the non-repudiation property is crucial to prevent devices from disputing the source of their messages.

In order to provide cryptographic protection for smart grid communication, IEC 62351 standard has been proposed [1], [15]. While the use of digital signatures for IEC 61850 GOOSE and SV is specified, given the resource constraints on smart grid control devices and very stringent latency requirements, practical implementation of it remains a challenge. The recent revisions of the IEC 62351 standard stepped back from public key solutions in favor of symmetric key based ones, sacrificing some key features to achieve the required time stringency. This work leverages ICS characteristics and achieves the required time stringency without sacrificing the PKI properties.

While the typical size of GOOSE and SV messages is 140-190 Bytes, including Ethernet headers and trailers based on our observation in the smart grid testbed [43], in practice, the whole message does not have to be protected to ensure correct operation. To begin with, for a given message stream, a separate digital signature scheme can be set up, and the static data fields in the stream (e.g., the MAC/IP addresses of the source and destination) can be pre-agreed by the parties. With such a per-stream setup, we can focus on integrity protection for those non-static data fields. For instance, sequence and status numbers are represented with multiple bytes. But in practice, we only need to check the integrity of the least-significant byte (or even just a few bits), because these numbers do not show a gap more than a few under a normal situation. A manipulation beyond this can be detected by either the destination device or network-based intrusion detection systems.

The integrity of the conveyed dataset is equally important. The size of the dataset largely varies depending on the system configuration, but the dataset used for latency-stringent protection control consists of a couple of boolean flags, for instance, circuit breaker status (open or close), and relay status (normal or fault). Altogether, the data to be integrity-protected is less than 4 Bytes (32 bits) in practice.

We could additionally consider including a timestamp (8-byte long "t" field [44]). According to the protocol specification, this timestamp indicates the time of the last event

(e.g., status update), and the same value is included in the successive messages until the next event. Thus, the integrity of this timestamp is relevant only when an event happens.

Depending on the type of messages, the number of data items included may differ and may contain a "bit-string" field to indicate, for example, measurements and quality of data. While these can be represented with multiple bytes, the values in each field are similar (or identical) over time (i.e., small entropy) and fluctuate around the nominal values (e.g., 50.0 Hz for frequency measurements). Therefore, instead of the raw measurements, we can focus on integrity for the difference from the nominal values, and hence the size of data to be protected can be effectively reduced.

To sum up, based on our study on message types utilized in the smart grid testbed, the size of essential data to be protected is less than 16 Bytes (128 bits).

LoMoS is a low-latency digital signature mechanism, and thus it is most effective in time-critical, multicast machine-to-machine communication. Besides IEC 61850 GOOSE and SV, the proposed technique, for instance, is applicable and also effective to protect R-GOOSE (Routable GOOSE) and R-SV (Routable SV) defined in IEC 61850-90-5 that are used for inter-substation communication. LoMoS is theoretically applicable to unicast communication like IEC 61850 MMS (Manufacturing Message Specification), but it is less latency stringent [45], and thus protection using Transport Layer Security measures would be sufficient in such settings.

IV. LoMoS Model

Less-online/More-offline Signatures (LoMoS) offers a framework to support the stringent delay and message throughput requirements while providing security assurance equivalent to that of the converted digital signature scheme. In addition, LoMoS exhibits offline verification, whereas the conventional online/offline model does not.

LoMoS adheres to the same principle as online/offline signatures discussed earlier, but with the following additional properties, 1) the source only performs cheap operations consuming negligible time after the message is given, 2) a single setup supports the signing of multiple messages of variable size, without requiring padding to conform to a constant size, 3) online verification does not perform any expensive operation after the message is received.

Fig. 2 depicts the LoMoS model, where KeyGen, Setup and VerifySetup procedures constitute the offline phase. Prove and Verify are the signing and verification procedures, respectively. They together constitute the online phase. Hereon, we use the term "prove" instead of "sign" to reflect the fact that no signing in the conventional sense is performed in the Prove procedure.

A. Definition and Properties of LoMoS

Definition 1: In LoMoS model, there are two parties. The source (prover) wants to prove its identity and the authenticity of its messages to the destination (verifier). Neither the source nor the destination have prior information on the content of the future messages. The source does not perform any operations after the message is given, except for simple

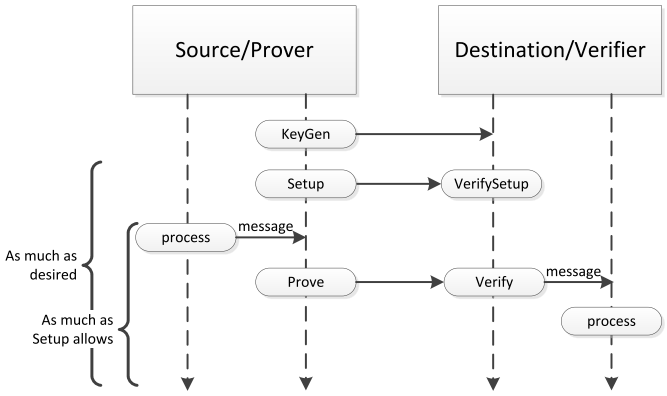


Fig. 2. LoMoS model.

operations consuming negligible time such as memory reads and packet assembly. Additionally, LoMoS allows the source to authenticate multiple messages (ideally variable size) per Setup procedure. The destination does not perform any expensive operations, such as modular exponentiation and elliptic curve scalar multiplication for online verification, keeping the end-to-end delay minimal.

KeyGen$(1^n) \longrightarrow \{\text{priv}_k, \text{pub}_k\}$ is a probabilistic procedure run by the source. It takes a security parameter as input and outputs private and public keys. The source stores the private key and sends the public key to the destination.

Setup$(\text{priv}_k, \textit{length}) \longrightarrow \{\text{md}_{priv}, \text{md}_{pub}, \text{s}\}$ is a procedure run by the source. This procedure does not require the message to be given. It generates two types of meta-data for future messages – the private meta-data md_{priv} to be kept at the source, the public meta-data md_{pub}[1] to be published to the destinations – and the signature s for the public meta-data. The *length* determines the total size of the messages that can be verified at the destination per generated meta-data.

VerifySetup$(\text{md}_{pub}, \text{s}) \longrightarrow \{\text{accept/reject}\}$ is a procedure run by the destination. The destination verifies and stores the public meta-data md_{pub}.

Prove$(\textit{message}, \text{md}_{priv}, \textit{state}) \longrightarrow \{\text{proof}\}$ is a procedure run by the source after the message is given, to generate the proof of message authenticity and source identity. As input, it takes the given *message*, the private meta-data that has been kept at the source and the *state* of the private meta-data, since the same meta-data can be used for multiple future messages. As per Definition 1 above, this procedure does not perform any computation except memory reads and packet assembly.

Verify$(\textit{message}, \text{proof}, \text{md}_{pub}) \longrightarrow \{\text{accept/reject}\}$ is the procedure run by the destination upon receipt of the *message* and its proof. It takes the public meta-data md_{pub}, proof, and *message* sent by the source. The output of accept ideally means the source is legitimate and the *message* is not tampered with. Again, according to Definition 1, this procedure does not perform any expensive operation.

[1]We reserve the term pub_k for the public key of the converted signature scheme, and md_{pub} refers to the continuously refreshed rendition of public key in the other works.

V. Using Authenticated Data Structures for LoMoS

In this section we discuss possible building blocks to instantiate the LoMoS model. We explore authenticated data structures to provide constructions for LoMoS. Authenticated data structures [46] are commonly used in a computation model (i.e., Provable Data Possession [47]–[49]) where, on behalf of a trusted source, an untrusted respondent responds to queries on a data structure and gives evidence for the validity of the response.

The main difference in LoMoS model is that the source is both the trusted source and the untrusted respondent in turns. The source uses its private key for any signature scheme during the offline phase, rendering it the trusted source. In the offline phase, the source generates an authenticated data structure, extracts the meta-data, then signs and sends the meta-data to be verified and stored. In the online phase, the source cannot use its private key (to avoid delay); hence it acts as the untrusted respondent and uses the authenticated data structure proofs to create the evidence for the integrity of the messages.

A. Straw-Man LoMoS Constructions

1) Using an Authenticated Hash List: This is a naive approach in altering Lamport OTS [26] to satisfy Definition 1. The source generates a pair of nonces (called "neighbors") per future message bit, where nonce n_1 corresponds to "1" and nonce n_0 to "0". The source then generates a list of digests, calculated using neighbors' digests, hash(hash(n_1), hash(n_0)). The source then signs the list and sends the whole list to the destination together with the signature. Later, when a message is given, the source sends the nonce and its neighbor's digest per bit of the message. The destination then performs two hash calculations to verify each bit of the message: First, it calculates the hash of the plain nonce (h_1 = hash(n_1), if the bit is set to "1") and second, a hash using the nonce hash of its neighbor (h_0) and this new digest ($h' = \text{hash}(h_1, h_0)$). Then it checks if the newly calculated digest (h') is equal to the previously signed digest in the list. This approach is useful in terms of Prove and Verify procedures' time/complexity. However, the source sends a list containing l digests in every Prove and Setup, where l is the number of bits supported per Setup procedure. This large list is sent in every Setup cycle, even when no messages are sent. In addition, this construction also has flaws (see Section V-B).

2) A Tree'd OTS for Short Messages: A major problem with Lamport OTS is, as the name suggests, the one-timeness of the key pairs. Using an MHT, an OTS (such as Lamport or Winternitz OTS) can be converted into a many-time signature to sign a number of messages with a single public key [28]. In this technique known as Merkle Signature Scheme (MSS), after generating the key pairs for an OTS, the source builds a tree on the hash of the OTS public keys and obtains a root value. This root is called the public key of MSS and is shared with the destination. MSS allows the use of this public key as many times as the number of leaves in the tree. However, in the original MSS, the OTS public keys are used to authenticate arbitrary size messages, by reducing each message to a digest (256-bit for SHA-256). In this straw-man technique, we modify tree'd OTS to benefit from messages smaller than 256 bits, by using a short message directly, without the first step of calculating its digest. This allows MSS to achieve faster setup and verification times as well as a lower communication overhead. If the message size is variable, this tweak requires the source to choose the OTS key size for the largest message to be signed. Note that the OTS key size directly affects the computations in all steps. Therefore, while MSS benefits from small message size thanks to this tweak, it still does not support variable message size as targeted in our approach.

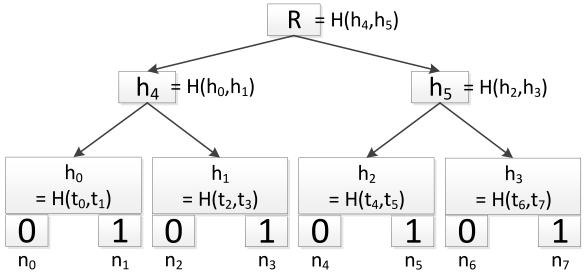


Fig. 3. MHT with two nonces per leaf. $t_i = H(n_i)$.

B. Tri-Leaf Tree for LoMoS

"A tree'd OTS for short messages" results in virtually instant online signing and fast setup/verification times. However, computation overhead for any message is as much as that of the largest possible message. In order to support variable size messages with proportionally smaller computation and communication overhead for small messages, consider the following technique. We choose Merkle Hash Tree [50] over others, such as balanced trees or skip lists, since the data structure we use is not required to be mutable.

The tree's root is again signed at the Setup procedure and shared with the destinations, where it is verified (VerifySetup). However, the difference is that, instead of generating large OTS key pairs for multiple messages and building the tree on top of them, the source directly builds the tree on pairs of nonces, each representing "0" and "1" bits of future messages. Hence, the tree keeps two nonce values per leaf node. To generate proofs for messages [1, 0] and [0, 1] consecutively, we show an example over Fig. 3. A proof generated from an authenticated data structure such as an MHT consists of the values required to calculate the root digest. For instance, the proof for [1, 0] is $\{n_1, n_2, t_0, t_3, h_5\}$ and the proof for the next message ([0, 1]) is $\{n_4, n_7, t_5, t_6, h_4\}$. A probabilistic polynomial-time (PPT) adversary cannot flip any bits of the messages unless the unused nonce values are compromised. Otherwise, we can break the collision resistance or pre-image resistance of the hash function. However, this method has fundamental flaws, i.e., vulnerability against concatenation/truncation attacks.

An adversary, having access to the proofs, can crop messages, append bits to the messages, or merge two messages, and forge proofs for them. Here is an example where the adversary appends a bit to the first message and forges the proof. It modifies the message [1, 0] to [1, 0, 0]. The proof for [1, 0, 0]

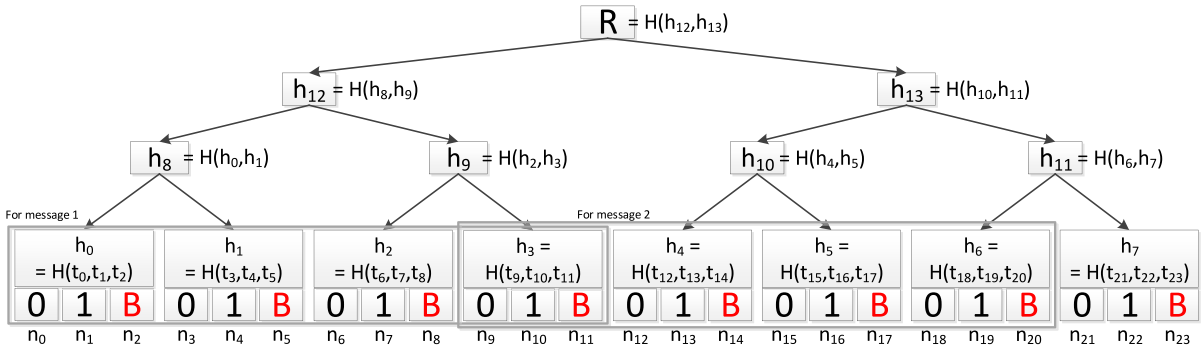


Fig. 4. A Tri-leaf Tree example. $t_i = H(n_i)$.

TABLE I
NOTATIONS

$\wedge$	Bitwise AND operator	$\oplus$	Bitwise XOR operator
$\ll$	Bitwise left shift operator	$\gg$	Bitwise right shift operator
? :	Conditional (ternary) operator	$\|\|$	Concatenation

is $\{n_1, n_2, n_4, t_0, t_3, t_5, h_3\}$, which are the values required to calculate the root digest and all are present in the genuine proofs for [1, 0] and [0, 1] above, except for h_3 which can be readily calculated as $h_3 = H(t_6, H(n_7))$.

To prevent all such attacks, we encapsulate the messages by starting and ending every message with a "BREAK" (B). As shown in Fig. 4, instead of having 2, we store 3 nonce values per leaf node. "B" is used as a separator between messages. Subsequent messages share the same intersecting leaf node "B" in their proofs as in the figure. Only one nonce per leaf shall be included in a proof; otherwise, an adversary can forge proofs.

C. Tri-Leaf Tree Definition

This modification brings us to the authenticated data structure for our construction, named Tri-leaf Tree. A Tri-leaf Tree (Fig. 4) is a static tree where the size is preset based on the expected message size and the desired message authentication rate. The tree is built and then consumed to generate proofs. After either the tree expires or its leaves are consumed, the tree is discarded. The build complexity is O(n), where n is the number of bits the data structure supports for future messages. In the following sections, we present Tri-leaf Tree algorithms adopting the notations shown in Table I. We design the algorithms with bit operators, given their performance friendliness.

Before proceeding to our construction based on Tri-leaf tree, we first present its build algorithm (Algorithm 1). Lines 2-7 generate the nonce values and calculate their digests. Line 8 calculates the collective digest for every leaf node. Lines 10-13 iteratively create the parent nodes using the digests of their children.

Algorithm 1: Tri-Leaf Tree Build Algorithm

```
   input : Tri-leaf Tree height H
   output: A Tri-leaf Tree T
1  levelSize ← 1 ≪ H
2  for i = 1 to levelSize do
3      concat ← [ ]
4      for j = 0 to 3 do
5          T[0][i].items[j] ← random()
6          T[0][i].hashedItems[j] ← hash(T[0][i].items[j])
7          concat ← concat || T[0][i].hashedItems[j]
8      T[1][i].digest ← hash(concat);
9
10 levelSize ← levelSize ≫ 1
11 for i = 1 to H − 1 do
12     for j = 1 to levelSize − 1 do
13         T[i + 1][j ≫ 1].digest ← hash(T[i][j].digest ||
             T[i][j+1].digest)
```

VI. Tri-Leaf Tree Based LoMoS Construction

In the following, we present the proof-of-concept construction of the LoMoS model and the corresponding algorithms.

KeyGen$(1^n) \longrightarrow (priv_k, pub_k)$: This procedure corresponds to key generation procedure of the employed signature scheme. We use any signature scheme only to sign the root periodically. Having the security parameter as input, this procedure returns a private key and its corresponding public key $(priv_k, pub_k)$ for the chosen signature scheme. The source stores the private key, and publishes the public key.

Setup$(priv_k, \textit{length}) \longrightarrow \{md_{priv}, md_{pub}, s\}$: Inputs are the source's private key and *length*, which determines the number of bits to be supported in future messages. This procedure calls Algorithm 1 to build the Tri-leaf Tree of size *length*. Then it signs the tree's root (and the static shared fields of the future messages it covers) with the employed signature scheme, using $priv_k$. Outputs are the Tri-leaf Tree as the private meta-data md_{priv}, its root as the public meta-data md_{pub}, and the signature s (which includes the timestamp).

VerifySetup$(pub_k, md_{pub}, s) \longrightarrow \{\text{accept/reject}\}$: The destination uses the public key to verify the public meta-data with the employed signature scheme's verification algorithm. If accepted, the public meta-data md_{pub} is stored.

Prove$(\textit{message}, md_{priv}, \textit{state}) \longrightarrow \{\text{proof}\}$: This procedure takes the tree for md_{priv}, the *state* indicating the location of the last used leaf node, and the message as inputs. It collects necessary values corresponding to *message* to allow the destination to calculate the root digest value. This corresponds to calling Algorithm 2 with the Tri-leaf Tree, its *state*, and *message*. Finally, Algorithm 2 outputs the proof vector.

Algorithm 2: Proof Generation Algorithm

```
   input : A Tri-leaf Tree T, state s, message M
   output: The proof P
1  T.offset ← s.index
2  P ← [T.offset]
3  pointerLeft ← T.offset
4  pointerRight ← T.offset + M.length - 1
5
6  for i ← 0 to M.length do
7  |  P.append(extractNonce(M[i],T[0][T.offset + i]))
8
9  for level ← 0 to T.height do
10 |  if pointerLeft ∧ 1 then
11 |  |  P.append(T[level][pointerLeft ⊕ 1])
12 |  if ¬pointerRight ∧ 1 then
13 |  |  P.append(T[level][pointerRight ⊕ 1])
14 |  pointerLeft ← pointerLeft ≫ 1
15 |  pointerRight ← pointerRight ≫ 1
16 |  level ← level + 1
17 return P
```

Algorithm 3: Verification Algorithm

```
   input : proof P, message M, root R
   output: accept, reject
1  offset ← P[0]
2  p ← 1
3  levelSize ← M.length
4
5  for i ← 0 to levelSize do
6  |  val0 ← (M[i] == '0' ) ? hash(P[p]) : P[p]
7  |  val1 ← (M[i] == '1' ) ? hash(P[p + 1]) : P[p + 1]
8  |  val2 ← (M[i] == 'B' ) ? hash(P[p + 2]) : P[p + 2]
9  |  buffer.append(val0‖val1‖val2)
10 |  p ← p + 3
11
12 while levelSize > 1 do
13 |  ind ← 0
14 |  if (ind + offset) ∧ 1 then
15 |  |  buffer[0] ← hash(P[p] ‖ buffer[0])
16 |  |  p ← p + 1
17 |  |  levelSize ← levelSize + 1
18 |  |  ind ← 1
19 |  while ind < levelSize do
20 |  |  buffer[ind ≫ 1] ← hash(buffer[ind] ‖ buffer[ind + 1])
21 |  |  ind ← ind + 2
22 |  if ind equals level then
23 |  |  buffer [ind ≫ 1 ] ← hash(buffer[ind] ‖ P[p])
24 |  |  p ← p + 1
25 |  |  levelSize ← levelSize + 1
26 |  offset ← offset ≫ 1
27 |  levelSize ← levelSize ≫ 1
28
29 (R equals buffer[0]) ? accept : reject
```

The runtime has O(m + log *length*) memory read operations, where m is the message size and *length* is the number of bits supported by the Setup.

Verify(*message*, proof, md_{pub}) $\longrightarrow$ {accept/reject}: This procedure calls Algorithm 3 with the proof, the message, and the public meta-data as inputs. It verifies the *message* if Algorithm 3 returns accept. The space complexity is O(m), and the runtime complexity is O(m + log *length*) consisting of hash calculations, where m is the *message* size and *length* is the number of bits supported by the Setup.

A. Online Proof Generation and Verification

In this section, we provide the algorithms for Prove and Verify procedures. The Setup/VerifySetup procedures discussed above run periodically and provide the authenticated root digest to the destination(s). Thus in this section, we assume the root value is known to the destination(s) and is fresh. In our context, the proofs are generated for multiple consecutive leaf level nodes.

Batch proving: Other than one nonce and two nonce digests for each leaf node, the proof contains the union of the siblings to the left of the leftmost node, and the siblings to the right of the rightmost node, at each level on the path to the root. The other nodes in between are not included because they can be readily calculated from the values in the proof. In the following, we present the proof generation for message 2 corresponding to the set $S = \{h_3, h_4, h_5, h_6\}$ in Fig. 4. The proof for the leftmost leaf node h_3 is $\{h_2, h_8, h_{13}\}$, while the proof for the rightmost leaf node h_6 is $\{h_7, h_{10}, h_{12}\}$. The left (right) siblings are the nodes that have smaller (larger) bit addresses[2] than their siblings' bit addresses. Therefore, the proof for the leftmost leaf node h_3 contains those nodes only to the left, i.e., $\{h_2, h_8\}$, while the proof for the rightmost leaf node h_6, contains those only to the right, i.e., $\{h_7\}$. Therefore, the proof for the node set S is $\{h_2, h_7, h_8\}$.

In Algorithm 2, we present the proof generation algorithm. Given a node with the bit address A, its parent and siblings have addresses $A \gg 1, A \oplus 1$, respectively. A node has a left sibling if and only if the least significant bit of its bit address is 1, otherwise it has a right sibling, with the root node being the exception. This algorithm generates the proof for the leftmost and the rightmost nodes via tracking them back to the root with pointers **pointerLeft, pointerRight**, initialized in Lines 2 and 3. In Lines 6-7, **extractNonce** routine extracts the values needed for the proof, which are then collocated for the proof. Lines 10-13 check whether **pointerLeft** and **pointerRight** have a left or right sibling and whether the sibling is to be included in the proof. Lines 14-15 update the leftmost and rightmost nodes for the processing of the next level.

Batch verification: In the verification process, the destination follows the received proof to calculate a value and compares it with the stored public meta-data. A pointer **p** points to the index to be processed starting from 1. In Algorithm 3, Lines 5-10 calculate the leaf digest using the three values (generated by the **extractNonce** routine) corresponding to each leaf in the proof. Once this process is done, the next step is to calculate the remaining leaves whose offset are also known. Taking these values as base, the subsequent levels are reconstructed. Note that we use a memory size proportional to the message length and reuse the same memory to overwrite the lower level node values, which are to be used only once. Hence, the space complexity remains O(m), where m is the message size. Lines 12-27 calculate the remaining nodes of the tree from the proof, to obtain the root value.

[2]Naturally, every node on a binary tree can be identified with a bit address. Starting from the root node, these addresses indicate which child of a node should be followed in order to reach a particular node. For instance, the bit addresses of h_9, h_3 in Fig. 4 are '01', '011', respectively.

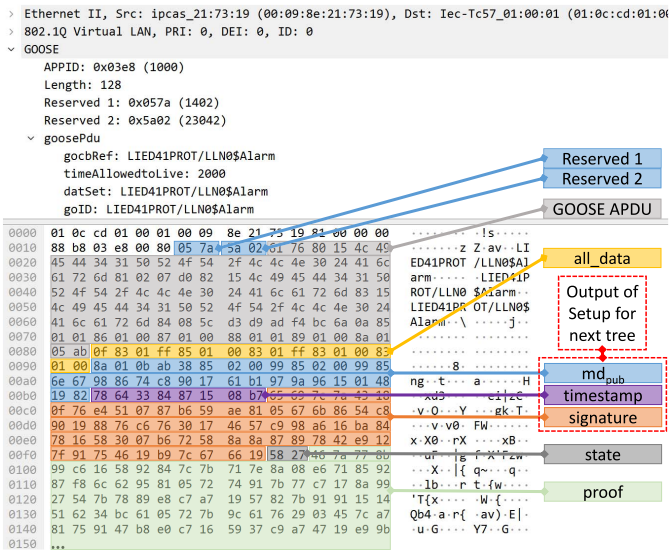


Fig. 5. GOOSE packet capture of LoMoS construction.

Line 29 compares the calculated root value with the stored, and returns accept if they are equal.

Figure 5 shows a captured GOOSE packet secured with Tri-leaf tree based LoMoS construction. The relevant fields are labeled. The overhead of LoMoS construction is added to the "Extension" field (after "all_data") of the GOOSE PDU. The size of the "Extension" field is stated in the "Reserved 1" field, and the CRC for the "Extension" field is put into the "Reserved 2" field, same as the approach done in IEC 62351. The public metadata md_{pub}, and its timestamped signature are the output of the Setup procedure above. They are sent for the destination(s) to authenticate the public metadata to be used after the current one expires. The "state" field contains a helper value for the proof (as discussed above). The "proof" field is for the message which contains the non-static values of the "all_data" field. Given the similarity in payload format between GOOSE and SV protocols, SV packets can also be structured in the same way.

B. Sketch of the Proof of Security

There are two steps to consider in the proof. First is the secure transmission of public meta-data to the destination, which is as secure as the employed signature scheme. For the second, we show that our scheme is secure if the underlying hash function is collision and pre-image resistant. If a PPT adversary $\mathcal{A}$ wins the security game of our scheme with non-negligible probability, we use it to construct other PPT algorithms $\mathcal{B}$ or $\mathcal{B}$' who break with non-negligible probability, the collision resistance or the pre-image resistance of the hash function, respectively. $\mathcal{B}$ acts as the adversary in the security game with the hash function challenger $\mathcal{HC}$. In parallel, $\mathcal{B}$ plays the role of the challenger in our game with $\mathcal{A}$. Consider the following existential forgery game under adaptive chosen-message attack:

Setup: $\mathcal{HC}$ picks a secure hash function (*hash*) from a hash function family – we use SHA-256 in our implementation – and passes the parameters to $\mathcal{B}$. $\mathcal{B}$ generates a list of nonce values – 256 bits in our implementation – and builds a large enough Tri-leaf Tree on top of them to answer all the following queries below. $\mathcal{B}$ then calculates the hash values using *hash* for every node of the authenticated data structure and passes the hash value of the root (*rh*) to $\mathcal{A}$. Since in the Query step, $\mathcal{A}$ can ask the proofs of polynomially many (p) messages of size m, $\mathcal{B}$ performs O($p\times$m) to create the tree, given that creating the tree is linear in the number of leaf nodes (see Algorithm 1). Since the message size is bounded, $\mathcal{B}$'s load is polynomial.

Query: $\mathcal{A}$ adaptively picks messages m_i, $\forall i \in \{1,\ldots,p\}$, and passes them to $\mathcal{B}$ one by one. $\mathcal{B}$ calls Algorithm 2 to collect the nonce values corresponding to each message m_i it receives, and the neighboring values required to calculate the root value, and passes them to $\mathcal{A}$. Note the following two: 1) the messages are separated with "BREAK" symbols and 2) with each proof generated, the pointer in the tree shifts forward accordingly (Algorithm 2).

Challenge: $\mathcal{A}$ prepares and sends another message m', where $m' \neq m_i$, $\forall i \in \{1,\ldots,p\}$ and a set of values (proof') that are required to calculate the hash value of the root.

If $\mathcal{B}$ verifies the proof' and m' with *rh* using Algorithm 3, then $\mathcal{A}$ wins. For this, $\mathcal{A}$ needs to find a nonce value that has not been revealed to it or needs to come up with a new set of values, which, when chained together with the given hash function *hash*, yields the hash value of the root. For the first case, the probability that $\mathcal{A}$ can find a nonce that has not been revealed to it is negligible, since otherwise, we can define a $\mathcal{B}$' that breaks the pre-image resistance of *hash*. For the second, $\mathcal{B}$ uses the new set of values to break the security of the given hash function (by finding a collision in *hash*) using the collision on the chain from a leaf node corresponding to a different bit of m' from any m_i, to the root.

VII. Evaluation

We have implemented the prototype on an embedded platform, BeagleBoard-X15 (TI AM5728 2 x 1.5-GHz ARM Cortex-A15 Processor with 2GB DDR3 RAM, running Debian 9.4) [51]. We have chosen this particular board for its interface options for ease of integration to ICSs, and its common, low-cost hardware.[3] We collected measurements from the Electric Power and Intelligent Control (EPIC) testbed [43], a comprehensive IEC 61850-compliant smart grid testbed. We connected the BeagleBoard running our solution into EPIC's generation segment in a bump-in-the-wire (BITW) manner without changing existing IEDs in the testbed. An alternative implementation would consider running our solution as a part of the firmware of ICS devices, if the firmware update is an option. We also demonstrated compatibility with the IEDs and PLCs as well as the network in this testbed.

We used C/C++ and employed the OpenSSL library to implement our LoMoS construction, RSA converted via improved online/offline signatures (IOOS-RSA), DSA, and ECDSA for comparison. We compiled the figures based on

[3]The use of generic hash functions in certain constructions (such as ours), benefits more from specialized hardware, now integrated into many processors, e.g., AES instruction set [52], Intel SHA extensions [53]. However, we evaluated our construction without relying on optimizations, instead did the comparisons on a common ground for fairness.

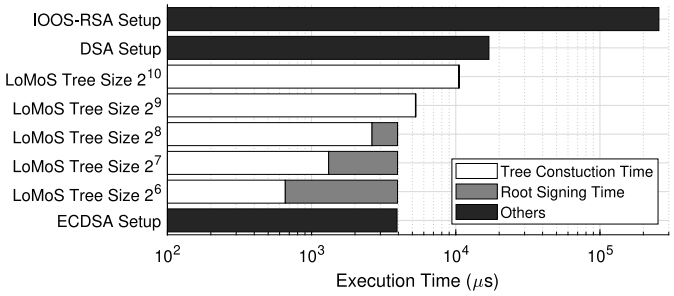


Fig. 6. Setup time comparison between our LoMoS construction, DSA, IOOS-RSA, and ECDSA.

the measurement results from the signature schemes that support offline computation, hence did not include conventional RSA. Instead, we used RSA, converted to an online/offline signature [8], hence named IOOS-RSA. Since IOOS only speeds up the online signing, RSA is selected for being verification-efficient. We implemented IOOS-RSA using the third and recommended trapdoor hash function in [8].

All measurements are taken with a single active core on the BeagleBoard, except for the Setup of LoMoS in which ECDSA's offline phase and the build algorithm of Tri-leaf tree are run in parallel. Tri-leaf tree for our LoMoS construction is kept in RAM during the operations. All results are the average of 500 runs. For all implementations, we aimed 128-bit security in choosing parameters (e.g., SHA-256 as the digest function and 256-bit nonces, 3072-bit modulus in DSA, IOOS-RSA, curve P-256 in ECDSA).

In addition to the evaluations, we have also provided analytical comparisons between our LoMoS construction and our straw-man LoMoS construction based on tree'd Lamport OTS.

Due to modular exponentiation in IOOS-RSA, the online phase is slower than DSA and ECDSA by a large margin and not drawn in the plots other than the Setup time and message authentication rate comparisons.

Message authentication rate: We call the number of messages supported by a signature scheme in unit time – be the bottleneck offline or online phase – the message authentication rate. The message authentication rate determines the throughput to be supported. Including our LoMoS construction, the message authentication rates of all the evaluated schemes are restricted by the offline phase. Each scheme's Setup time is presented in Fig. 6. There are a limited number of messages supported after each Setup. For the LoMoS construction, the tree size determines the number of messages supported per Setup. For IOOS-RSA, DSA, and ECDSA, only one message is supported per Setup. Although the Setup time of the LoMoS construction is comparable with that of ECDSA, its message authentication rate is higher because each Setup supports multiple messages. Fig. 7 shows the message authentication rates of the evaluated signature schemes and the shaded region represents the 4,000 messages/sec required by IEC 61850-SV. IOOS-RSA is not shown on this figure given its low performance. The LoMoS construction outperforms others with a larger margin as the message size gets smaller, down from 256 bits. LoMoS can support 4,000 messages/sec (in a 50 Hz system) up to around 25-bit message size on average.

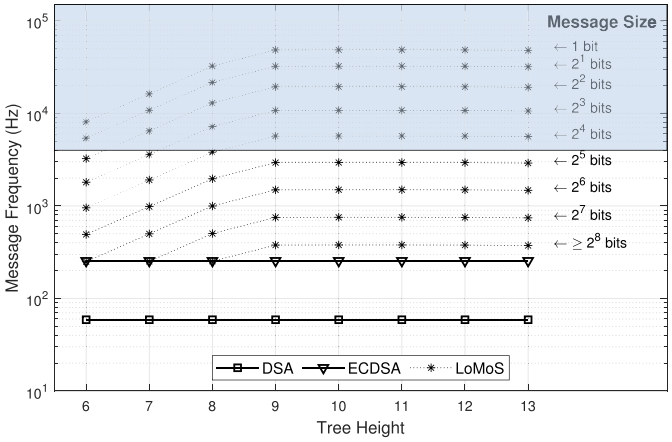


Fig. 7. Message authentication rates bound by Setup.

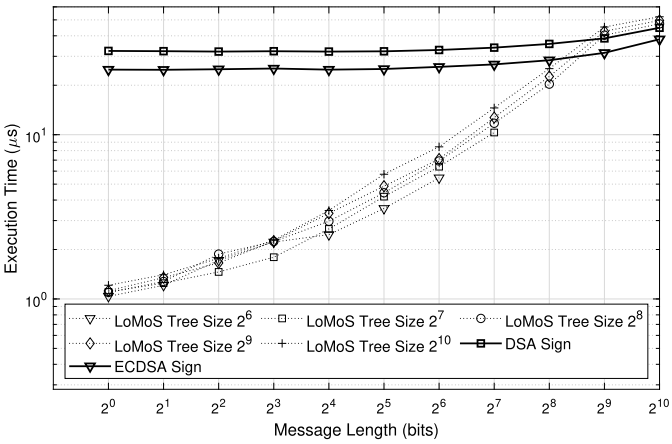


Fig. 8. Online signing time.

If we protect the least significant byte, SV messages conveying 2-3 measurements (e.g., 3-phase measurements) can be supported.

Message authentication rate is not bottlenecked by the verification time for any of the evaluated schemes, be it online or offline. For reference, the offline verification time of the LoMoS construction is 2.5 ms as we converted ECDSA.

Supporting adaptive message authentication rate: Above, we discussed the rate of message authentication to be supported by the Setup. However, in our LoMoS construction, after a Setup is performed, the source can sign messages at a rate bounded only by the online signing (Prove) rate. We refer to this feature as adaptive message authentication rate. It is effective to support, for instance, IEC 61850 GOOSE messaging discussed in Section III, whose transmission rate dramatically increases intermittently upon events.

In our LoMoS construction, online verification time limits the rate at which the received proofs are processed at the destination. The verification times given in Fig. 9 are larger than those of proving, as shown in Fig. 8. Therefore, online verification time is the limiting factor for the throughput, provided that the total size of the messages does not exceeds the number of bits supported by the Setup. When a source publishes multiple messages which the destinations selectively receive

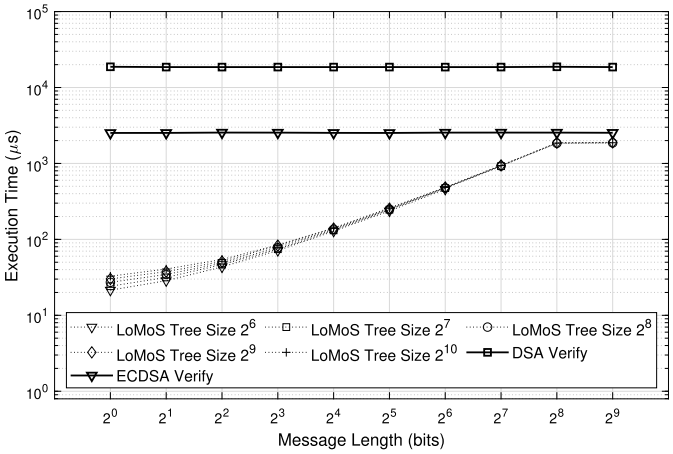


Fig. 9. Online verification time.

and authenticate, a faster proving time than the verification time is beneficial. In the LoMoS construction, unless the total size of the messages exceeds the size supported by the Setup, a source can release signed messages with 2-10 μs intervals (see Fig. 8).

End-to-end delay: The end-to-end delay constraints for many time stringent settings such as power grid control are at 1-2 ms level, of which not much is left for authentication. The times spent in online signing and verification are shown in Figures 8 and 9, respectively. The total of the values in these figures gives the end-to-end processing overhead for message authentication. As shown, the LoMoS construction widely outperforms all alternative constructions for small messages (down from 256 bits) since it only reads values from memory at the source, and computes a series of efficient hash functions at the destination.

The online verification of our LoMoS construction consists of a series of inexpensive hash calculations to verify the received message, therefore outperforms others for messages up to 256 bits. Beyond 256 bits, the LoMoS construction hashes the message to obtain a 256-bit digest to sign. Therefore, the computation time remains around the same beyond 256-bit messages, as shown in Figures 8 and 9. In both figures, from 2^8 to 2^9 bits, the execution time increases around 20 ms, which is the time to compute SHA-256.

In comparison, the symmetric key based HMAC takes 40N μs for MAC generation (online signing time as in Figure 8), where N is the number of destinations for the multicast message, and another 40 μs for MAC verification (online verification time as in Figure 9), in total, introducing $40(\mathrm{N}+1)$ μs to end-to-end delay. This latency is added as a part of the communication delay that is bounded with 2 ms.

Supporting variable message size: One distinguishing property of our LoMoS construction is its support of variable-size messages. It does so at the expense of a reasonable overhead over the straw-man "tree'd OTS for short messages" in Section V-A2. Note that the amount of computation done in SHA-256 (as a Merkle–Damgård construction) scales linearly with the input size, neglecting the time spent to initialize and finalize the hash functions. Therefore, we report the complexity of computations throughout the Setup and Verify procedures in terms of the number of hashing of 256-bit inputs. Note that nonces are of 256 bits, and the hash function (SHA-256) output is also 256 bits for both schemes.

To support x number of m-bit messages, the straw-man design performs computation amounting to the equivalent of $2mx$ for OTS public key generation, $2mx$ to compute the leaves of the MHT, and $2(x-1)$ to compute the root of MHT, hence totaling $4x(m+1/2)-2$ in the setup. To support the same number of messages of average length m, the LoMoS construction needs to construct a tree of size $l = x(m+1)$ (including "B"s). In comparison, the LoMoS construction performs $3l$ to hash the nonce triplets, $3l$ to generate the leaves, and $2(l-1)$ to obtain the root, totaling $8l-2$ or $8x(m+1)-2$.

For online verification of an m-bit message, the straw-man incurs m to calculate the OTS public key and $2m + 2log(x)$ verify the OTS public key, totaling $3m+2log(x)$. The LoMoS construction incurs $m+2$ to hash the nonces, $3(m+2)+2log(l)$ to compute the root, totaling $4m + 2log(x) + 2log(m+1) + 8$.

Communication overhead: The construction proposed in this paper achieves all said properties at the expense of communication overhead. The communication overhead from sending a proof is one nonce and two nonce digests per bit of the message, plus the digests corresponding to certain sibling nodes on the path to the root. This amounts to $3m + \log(l) - \log(m)$ and $3m + \log(l) + \log(m)$ times the digest/nonce length in the best and worst cases, respectively. Accounting for the fact that each message begins and ends with "B" leaves, the total overhead is between $3(m+2)+\log(l)-\log(m+2)$ and $3(m+2)+\log(l)+\log(m+2)$ times the digest/nonce length. On a communication link of rate R, the minimum/maximum transmission delay D_{T} incurred by the communication overhead, given 256-bit digest/nonce length would be,

$$D_{\mathrm{T}} = 256 \times \frac{3(m+2) + \log(l) \mp \log(m+2)}{R}$$

For example, for a 32-bit message, this corresponds to around 27 μs on a 1 Gbps link and is multiplied if there are multiple hops between the source and destination. In comparison, ECDSA incurs 0.5 μs transmission delay. The time spent for proving a 32-bit message in the LoMoS construction is around 4 μs, compared to ECDSA's 25 μs. The corresponding verification time is 250 μs compared to ECDSA's 2500 μs. This brings the total delay for prove-transmit-verify routine of the LoMoS construction to 281 μs against 2525 μs of ECDSA for 32-bit messages. At 256 bits, which is the digest length for messages longer than 256 bits, the prove-transmit-verify delays of the LoMoS construction are 25-200-1900 μs, against ECDSA's 28-0.5-2500 μs.

The communication overhead in the Setup procedure of the LoMoS construction is the 256-bit public meta-data and its signature, which depends on the employed signature scheme (512 bits for ECDSA). However, Setup procedure is run during the offline phase, hence it does not contribute to the transmission delay after the message is given.

Effect of tree size in LoMoS construction: The tree size affects the time and storage required at various steps. As the

tree grows, the verification gets slower, since one more hash calculation is performed at the destination per level of the tree. With the growth in tree size, the source storage requirement is also increased, since the tree is stored as the private meta-data. The timestamp is refreshed less frequently with the growth in tree size. On the other hand, with a smaller tree, the source is less flexible in benefiting from the adaptive message rate. Moreover, when the tree generation time is shorter than the offline signing time of the employed signature scheme, the Setup procedure takes about the same amount of time regardless of the tree size (see Fig. 6). That is because, the offline signing of the employed signature scheme is run in parallel with the tree's build algorithm. A rough conclusion is that the tree should be larger than the average message size multiplied by the number of messages to be supported by a single Setup, and the time spent to generate the tree should be slightly larger than the maximum of the employed signature scheme's offline signing time.

Engineering considerations: The verifiers need short-term storage for the public meta-data, loss of which may cause the later messages in the same round to be unverifiable until meta-data is refreshed. As discussed in "Effect of tree size in LoMoS construction" above, the meta-data refresh interval can be as low as the converted digital signature scheme's offline signing time (see Fig. 6). Our construction is not susceptible to packet loss for the authentication of future messages. Any verified message also proves how many bits are missing between the previous verified message and the latest. However, the loss of packets carrying the public meta-data may have a similar effect as a short-term memory loss. For tolerance against packet loss and memory loss, the source should send the public meta-data with a certain degree of redundancy. Legacy compliance is of paramount importance too. Some devices may not support public key computations or be suited for such software updates. Bump-in-the-wire deployment can be considered in such cases. Also, note that the efficiency gains from the data structure implementation are reflected in the performance results. We have presented the algorithms above with bit operators, given their performance friendliness.

If implemented in large substations or other large-scale systems, and if there is a sudden surge of volume in the number of packets, the substantial increase in the frame size might cause longer queuing delays in the network. Against this possibility, the sufficiency of network infrastructure and network equipment should be ensured.

VIII. Conclusion

Modernized power grid implementations, including the emerging ones involving diversified entities responsible for energy generation (e.g., distributed energy resources), transmission, and distribution pose security challenges including insecure communication protocols and networks, and multiple management domains leading to challenging key management issues. The public key based solutions address the security gaps in this domain. These solutions including digital signatures inherently provide the required functions (e.g., non-repudiation, public verifiability, feasibility of key management), which symmetric key based solutions struggle and end up with complicated and less secure implementations. On the other hand, even the state-of-the-art digital signatures are known to be computationally expensive, particularly for embedded industrial control system devices. To mitigate the delay for message authentication, performing prior computations has been proposed. However, even the precomputation-based schemes such as online/offline signatures do not support extremely time stringent and high throughput communication settings such as IEC 61850 GOOSE and SV communication in smart grids. Hence, we have proposed the *Less-online/More-offline Signatures* (LoMoS) model. LoMoS, in essence, entails the migration of more online computations to offline. Following the model, we have proposed a LoMoS construction, which only performs memory reads and assembly to sign after the message is given, and conventional cryptographic hash computations to verify. Our LoMoS construction achieves promising results (with proportionally better performance for smaller messages) on low-cost devices, hence supporting time stringent communication protocols such as IEC 61850 GOOSE and SV. LoMoS construction achieves all said properties at the expense of communication overhead. Today's gigabit networks can easily handle the communication overhead incurred by LoMoS and the like. We invite researchers to design alternative constructions for the LoMoS model.

References

[1] *Power Systems Management and Associated Information Exchange-Data and Communications Security—Part 6: Security for IEC 61850*, IEC Standard 62351-6:2020, 2020.

[2] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int. J. Inf. Security*, vol. 1, no. 1, pp. 36–63, 2001.

[3] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Conf. Theory Appl. Cryptogr. Techn.*, 1986, pp. 186–194.

[4] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 239–252.

[5] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[6] P. Gallagher, *Digital Signature Standard (DSS)*, FIPS Standards 186–3, 2013.

[7] S. Even, O. Goldreich, and S. Micali, "On-line/off-line digital signatures," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 263–275.

[8] A. Shamir and Y. Tauman, "Improved online/offline signature schemes," in *Proc. Annu. Int. Cryptol. Conf.*, 2001, pp. 355–367.

[9] R. E. Mackiewicz, "Overview of IEC 61850 and benefits," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, 2006, p. 8.

[10] *IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation*, IEEE Standard 1646–2004, 2004.

[11] E. Esiner, D. Mashima, B. Chen, Z. Kalbarczyk, and D. Nicol, "F-Pro: A fast and flexible provenance-aware message authentication scheme for smart grid," in *Proc. SmartGridComm*, 2019, pp. 1–7.

[12] "IEC 61850 Client Interface SCADA Configuration Guide." 2020. [Online]. Available: https://www.honeywellprocess.com/library/support/Public/Documents/IEC-61850-Client-Interface-SCADA-Configuration-Guide-EPDOC-X376-en-516A.pdf (accessed Oct 22, 2021).

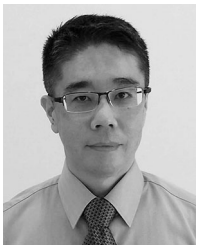
[13] S. M. Farooq, S. S. Hussain, and T. S. Ustun, "Performance evaluation and analysis of IEC 62351-6 probabilistic signature scheme for securing goose messages," *IEEE Access*, vol. 7, pp. 32343–32351, 2019.

[14] S. S. Hussain, S. M. Farooq, and T. S. Ustun, "Analysis and implementation of message authentication code (MAC) algorithms for goose message security," *IEEE Access*, vol. 7, pp. 80980–80984, 2019.
[15] F. Cleveland, "IEC 62351 security standards for the power system information infrastructure," Int. Electrotech. Comm., Geneva, Switzerland, Rep. IEC TC57 WG15, 2012.
[16] F. Hohlbaum, M. Braendle, and F. Alvarez, "Cyber security practical considerations for implementing IEC 62351," in *Proc. PAC World Conf.*, 2010, pp. 1–8.
[17] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *Proc. ACM Workshop Security Ad Hoc Sens. Netw.*, 2006, pp. 147–156.
[18] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2001, pp. 437–452.
[19] Y. Challal, H. Bettahar, and A. Bouabdallah, "A taxonomy of multicast data origin authentication: Issues and solutions," *IEEE Commun. Surveys Tuts.*, vol. 6, no. 3, pp. 34–57, 3rd Quart., 2004.
[20] T. T. Tesfay and J.-Y. Le Boudec, "Experimental comparison of multicast authentication for wide area monitoring systems," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4394–4404, Sep. 2018.
[21] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Boca Raton, FL, USA: CRC Press, 2014.
[22] J. Zhang, J. Li, X. Chen, M. Ni, T. Wang, and J. Luo, "A security scheme for intelligent substation communications considering real-time performance," *J. Mod. Power Syst. Clean Energy*, vol. 7, no. 4, pp. 948–961, 2019.
[23] J. Zhang and C. A. Gunter, "Application-aware secure multicast for power grid communications," in *Proc. 1st IEEE Int. Conf. Smart Grid Commun.*, 2010, pp. 339–344.
[24] F. Cleveland, "IEC TC57 security standards for the power system's information infrastructure—Beyond simple encryption," in *Proc. IEEE/PES Transm. Distrib. Conf. Exhibition*, May 2006, pp. 1079–1087.
[25] A. A. Yavuz, "An efficient real-time broadcast authentication scheme for command and control messages," *IEEE Trans. Inf. Forensics Security*, vol. 9, pp. 1733–1742, 2014.
[26] L. Lamport, "Constructing digital signatures from a one-way function," SRI Int., Menlo Park, CA, USA, Rep. CSL-98, 1979.
[27] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Massachusetts Inst. Technol., Cambridge Lab. Comput. Sci., Cambridge, MA, USA, Rep. MIT/LCS/TR-212, 1979.
[28] R. C. Merkle, "A certified digital signature," in *Proc. Conf. Theory Appl. Cryptol.*, 1989, pp. 218–238.
[29] A. Hülsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: Extended merkle signature scheme," Internet Res. Task Force, RFC 8391, 2018.
[30] A. Perrig, "The biba one-time signature and broadcast authentication protocol," in *Proc. 8th ACM Conf. Comput. Commun. Security*, 2001, pp. 28–37.
[31] L. Reyzin and N. Reyzin, "Better than BiBa: Short one-time signatures with fast signing and verifying," in *Proc. Aust. Conf. Inf. Security Privacy*, 2002, pp. 144–153.
[32] Q. Li and G. Cao, "Multicast authentication in the smart grid with one-time signature," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 686–696, Dec. 2011.
[33] H. M. Krawczyk and T. D. Rabin, "Chameleon hashing and signatures," U.S. Patent 6 108 783, Aug. 22, 2000.
[34] *IEC TC57: Communication Networks and Systems for Power Utility Automation—Part 90-2: Using IEC 61850 for the Communication Between Substations and Control Centres*, IETC Standard IEC 61850-90-2, 2015.
[35] J. Li, Q. Huang, F.-K. Hu, and S. Jing, "Performance testing on GOOSE and MSV transmission in one network," *Energy Procedia*, vol. 12, pp. 185–191, Sep. 2011.
[36] P. Schaub and A. Kenwrick, "An IEC 61850 process bus solution for powerlink's iPASS substation refurbishment project," *PAC World Mag.*, vol. 9, no. 2009, pp. 38–44, 2009.
[37] M. M. Roomi, P. P. Biswas, D. Mashima, Y. Fan, and E.-C. Chang, "False data injection cyber range of modernized substation system," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids (SmartGridComm)*, 2020, pp. 1–7.
[38] "Analysis of the cyber attack on the Ukrainian power grid," E-ISAC, Washington, DC, USA, Rep. 388, 2016, pp. 1–29.
[39] "Crashoverride Malware." [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA17-163A (accessed Aug. 18, 2017).
[40] J. Hoyos, M. Dehus, and T. X. Brown, "Exploiting the goose protocol: A practical attack on cyber-infrastructure," in *Proc. IEEE Globecom Workshops*, 2012, pp. 1508–1513.
[41] P. P. Biswas, H. C. Tan, Q. Zhu, Y. Li, D. Mashima, and B. Chen, "A synthesized dataset for cybersecurity study of IEC 61850 based substation," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids (SmartGridComm)*, 2019, pp. 1–7.
[42] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. S. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 675–685, Dec. 2011.
[43] "Electric Power and Intelligent Control (EPIC) Testbed." 2018. Online]. Available: https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_epic/ (accessed Jul. 12, 2021).
[44] C. Kriger, S. Behardien, and J.-C. Retonda-Modiya, "A detailed analysis of the goose message structure in an IEC 61850 standard-based substation automation system," *Int. J. Comput. Commun. Control*, vol. 8, no. 5, pp. 708–721, 2013.
[45] D. Mashima, P. Gunathilaka, and B. Chen, "Artificial command delaying for secure substation remote control: Design and implementation," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 471–482, Jan. 2019.
[46] R. Tamassia, "Authenticated data structures," in *Proc. Eur. Symp. Algorithms*, 2003, pp. 2–5.
[47] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Trans. Inf. Syst. Security*, vol. 17, no. 4, p. 15, 2015.
[48] E. Esiner, A. Kachkeev, S. Braunfeld, A. Küpçü, and Ö. Özkasap, "FlexDPDP: Flexlist-based optimized dynamic provable data possession," *ACM Trans. Storage*, vol. 12, no. 4, p. 23, 2016.
[49] E. Esiner and A. Datta, "Auditable versioned data storage outsourcing," *Future Gener. Comput. Syst.*, vol. 55, pp. 17–28, Feb. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X15002538
[50] R. C. Merkle, "Protocols for public key cryptosystems," in *Proc. IEEE Symp. Security Privacy*, 1980, p. 122.
[51] "Beagleboard-X15," Texas Instruments. 2018. [Online]. Available: https://beagleboard.org/x15 (accessed Jan. 10, 2020).
[52] J. W. Bos, O. Özen, and M. Stam, "Efficient hashing using the aes instruction set," in *Proc. Int. Workshop Cryptogr. Hardw. Embedded Syst.*, 2011, pp. 507–522.
[53] "Intel architecture instruction set extensions programming reference," Intel Corp., Mountain View, CA, USA, Rep. 319433-014, 2016.

Ertem Esiner received the French baccalaureate degree from Galatasaray Lycee in 2007, the B.Sc. and M.Sc. degrees in computer science and engineering from Koç University, Turkey, in 2011 and 2013, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2017. He currently serves as a Research Scientist and the Coordinator for Research Collaboration and Industry Relations with Advanced Digital Sciences Center, Singapore, where he is leading multiple research projects in smart grid security area. His research interests are cryptography, security, and privacy.

Utku Tefek received the B.Sc. degree in electrical and electronics engineering from Bilkent University, Turkey, in 2013, and the Ph.D. degree from the National University of Singapore in 2017. He currently serves as a Research Scientist with Advanced Digital Sciences Center, affiliate of the University of Illinois. His research interests include communication systems and networks, cyberphysical system security, and applied cryptography.

Hasan S. M. Erol received the B.Sc. degree in electrical and electronics engineering from Bilkent University, Turkey, in 2020. He is currently pursuing the Ph.D. degree in electrical engineering and computer science with the Massachusetts Institute of Technology. His research interests are information theory and machine learning.



Daisuke Mashima received the Ph.D. degree in computer science from the Georgia Institute of Technology in 2012. He is currently a Principal Research Scientist with Advanced Digital Sciences Center (ADSC), Singapore, where he is leading multiple research projects in smart grid security area. Before joining ADSC, he worked as a member of Research Staff with the Smart Energy Group, Fujitsu Laboratories of America, Inc. His research interest covers cybersecurity and privacy in cyber–physical systems in general.

Binbin Chen (Member, IEEE) received the B.Sc. degree in computer science from Peking University and the Ph.D. degree in computer science from the National University of Singapore. Since July 2019, he has been an Associate Professor with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design. He currently also holds a joint appointment as a Principal Research Scientist with Advanced Digital Sciences Center, which is a University of Illinois research center located in Singapore. His current research interests include wireless networks, cyber–physical systems, and cyber security for critical infrastructures.

Yih-Chun Hu received the Ph.D. degree from the Computer Science Department, Carnegie Mellon University in 2003. He is an Associate Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana–Champaign, affiliate faculty of Computer Science, and faculty of the Advanced Digital Sciences Center. His current research interests are in network security and wireless networks. His general research interests are in security and systems, with an emphasis on the areas of secure systems and mobile communications.

Zbigniew Kalbarczyk (Member, IEEE) is a Research Professor of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign. He currently leads research projects that are exploring and developing high-availability and secure infrastructure capable of managing redundant resources across interconnected nodes to detect errors in user applications and infrastructure components, to foil security threats, and to recover quickly from failures when they occur. His work also involves developing automated techniques and tools (including ML-based experimental methods) for validation and benchmarking of computing systems in a broad range of application domains, e.g., medical devices (such as surgical robots) and critical infrastructures (e.g., the power grid). In addition, he has led several projects sponsored by government agencies (e.g., DARPA, DOE, NSA, and DHS) to develop techniques and tools for design and validation (including the use of high-fidelity simulation and experimental methods) of highly critical systems. His research encompasses design and validation of resilient, safe, and secure computing systems and applications. He is a member of the IEEE Computer Society.

David M. Nicol (Fellow, IEEE) is the Herman M. Dieckamp Endowed Chair of Engineering with the University of Illinois Urbana Champaign, where he also serves as the Director of the Information Trust Institute. He is the Director also of the Advanced Digital Sciences Centre in Singapore. He is a Fellow of the ACM.