F-Pro: a Fast and Flexible Provenance-Aware Message Authentication Scheme for Smart Grid

Ertem Esiner, Daisuke Mashima, Binbin Chen

Advanced Digital Sciences Center
{e.esiner, daisuke.m, binbin.chen}@adsc-create.edu.sg

Zbigniew Kalbarczyk, David Nicol

University of Illinois at Urbana Champaign
{kalbarcz, dmnicol}@illinois.edu

***Abstract*—Successful attacks against smart grid systems often exploited the insufficiency of checking mechanisms — e.g., commands are largely executed without checking whether they are issued by the legitimate source and whether they are transmitted through the right network path and hence undergone all necessary mediations and scrutinizes. While adding such enhanced security checking into smart grid systems will significantly raise the bar for attackers, there are two key challenges: 1) the need for real-time, and 2) the need for flexibility — i.e., the scheme needs to be applicable to different deployment settings/communication models and counter various types of attacks. In this work, we design and implement F-Pro, a transparent, bump-in-the-wire solution for *fast* and *flexible* message authentication scheme that addresses both challenges. Specifically, by using a lightweight *hash-chaining-based* scheme that supports provenance verification, F-Pro achieves less than 2 milliseconds end-to-end proving and verifying delay for a single or 2-hop communication in a variety of smart grid communication models, when implemented on a low-cost BeagleBoard-X15 platform.**

I. Introduction

Industrial control systems (ICS), including smart power grid, are critical for the daily operation of our modern society. Over the past decade, cybersecurity attacks have become a major risk factor faced by smart grid operators. Many successful attacks, such as Stuxnet [1] and the Ukraine power grid attacks [2]–[4], exploited the deficiency of checking for monitoring and control in the cyber infrastructure of the smart grid. For example, the success of CrashOverride malware [2] used in the Ukraine incident in 2016 resulted from the fact that there were field devices that executed malicious commands without checking whether they were issued by the right user from the right source under the right context. Very often, only basic encryption is used and only a single signature is checked. This allows an attacker to steal a single key to bypass the checking. State-of-the-art ICS protection mechanisms introduce security appliances, such as application-layer firewalls, intrusion detection systems, etc., along a message's expected transmission path. These middle boxes add more security checking points. However, there is no checking at a destination device to ensure that the messages have indeed gone through the right network paths (and hence, have undergone all necessary mediations and scrutinies) before reaching the destination. As a result, an attacker can launch an attack from another source point (e.g., a node impersonating a SCADA master system) that bypasses all these en-route defense mechanisms.

In this work, we propose to introduce provenance checking into the smart grids for flexible, extended message authentication to counter the aforementioned threats. Provenance of electronic data is, in general, defined as "the derivation from a particular source to a specific state of an item" in [5]. In our particular context, we focus on authenticating the message source and verifying the message delivery path and the transformation of the message en route. Our scheme allows a message to securely gather and carry cryptographically-verifiable evidence about its source and the path it travels, which can then be checked at the destination before the message is processed by the destination device. Adding such checking into smart grid effectively raises the bar for attackers.

While desirable, designing such a scheme for a latency-stringent ICS like a power grid presents two key challenges:
The need for *real-time* processing: Some time-critical packets in smart power grids need to be delivered within 2 milliseconds. In fact, real-time need in power grids has caused difficulty for the adoption of IEC 62351 [6] using public-key cryptography (see, e.g., [7]).
The need for *flexibility*: Even within standard-compliant smart grid systems, a number of different communication models need to be supported, as will be elaborated in Section II. Specifically, some of the communication involves multiple hops (e.g., firewall, substation gateway, etc.) between the control center and intelligent electronic devices (IEDs), while the automated control within a substation may be a single-hop communication. Thus, a flexible message authentication scheme is demanded.

We have designed and implemented F-Pro, a *fast* and *flexible* message authentication scheme for smart grid that addresses both requirements. Our solution meets the stringent latency requirement in smart grid by using a lightweight, *hash-chaining-based* cryptographic scheme. F-Pro is implemented as a bump-in-the-wire (BITW) solution to ensure compatibility with legacy devices as well as to allow flexible deployments. Our measurements on a low-cost BeagleBoard-X15 platform [8] shows that F-Pro can achieve an end-to-end communication latency below 2 milliseconds in an intra-substation-alike setting, which meets the criteria defined in IEEE's guideline [9].

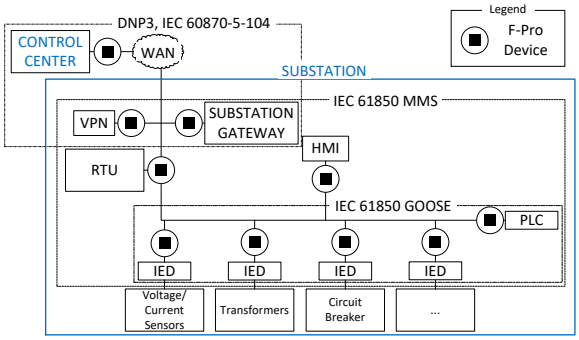


Fig. 1: Substation Automation and Remote Control: A Conceptual Architecture, Key Components, and Protocols and Protection with F-Pro Devices

II. Substation and Communication Models

A typical smart grid system includes a control center and multiple (possibly thousands of) substations in the field. Figure 1 shows one substation, connected to a control center via wide-area network (WAN). Modern substations use standardized technologies like IEC 60870-5-104 or DNP3.0 for telecontrol and IEC 61850 for substation automation [10].

Within the substation, intelligent electronic devices (IEDs) serve as the communication end points in the cyber side. They are responsible for operating on physical power system devices, e.g., circuit breakers and transformers. Real-time communication among IEDs is crucial for automated protection. Programmable logic controllers (PLCs) are also common devices and are in charge of automated control based on various power grid measurements. The substation gateway often performs protocol translation, e.g., between IEC 60870-5-104 and IEC 61850 [10]. Lastly, in order to enable remote maintenance, by grid operators or device vendors, virtual private network (VPN) devices connected to the public network are increasingly deployed.

Table I summarizes the typical communication patterns observed in the smart grid system. As shown in the table, some communication models involve only a single hop between a source and a destination. In IEC 61850-compliant substations, *Status Update in Substation*, *Automated Control in Substation*, and *Protection to Switchgear* are done using GOOSE (Generic Object Oriented Substation Events) protocol, which are publisher-subscriber-type communication using multicast.

In other cases, multiple entities (hops) are involved. For instance, in *SCADA Control/Monitoring*, commands from a control center to a field device in a low-voltage (or distribution level) substation may be mediated by devices in a high-voltage (or transmission level) substation. Another complicated communication model is the *Reporting by Field Devices*, where measurements from PLCs or IEDs are first sent to the substation gateway or RTU, which may perform protocol translation and/or message aggregation before forwarding them.

The last column of Table I shows the message delivery latency requirements found in the public guidelines [9], [11]. Two use cases corresponding to maintenance are not sensitive to latency, and therefore are left blank. As seen in the table, the latency requirements vary depending on use cases, and communication within substations has very stringent delivery latency requirements. In particular, *Protection to Switchgear* requires very short latency (below 2ms [9]).

III. Design Goals

To counter attacks mounted at various places, it is crucial to authenticate the source and integrity of messages. Protection of this type can be typically realized by making a sender *sign* the message to ensure integrity and authenticity. Checking of message delivery path is as important. When an attacker steals some authorized source's credential and tries to insert a message via an alternative path (e.g., from the malware on a field device), the verification of message delivery path allows the destination to block such attacks. In particular, the path verification checks whether a message has gone through the expected set of nodes (i.e., who has *witnessed* the message). For example, when a message has gone through a well-protected high-voltage substation system, downstream nodes can regard the messages trusted.

In addition, as discussed in Section II, it is often the case where conversion of messages, such as protocol translation, is involved. In such a case, to detect misbehaving nodes, knowing who performed the conversion and also how the message is modified helps decision-making. Such information allows the destination to check if, for example, the conversion is performed by a legitimate protocol translator. While the proposed F-Pro solution can provide verifiable information for such a consistency checking, the design of efficient checking algorithm is left for our future work.

Based on these observations, we introduce the concept of *provenance* for addressing security concerns. Specifically, we focus on checking the authenticity of the message source and verifying the end-to-end message delivery path (i.e., which nodes are involved and in what order) as well as message transformation en route. The crucial thing is systematic verifiability of the provenance information. In the machine-to-machine communication, such a verifiability relies on cryptographic schemes. The verified provenance can be then utilized for a variety of policy checking at the destination. Some of the nodes on the path (e.g., the gateway) could be less vulnerable to attacks, for instance, due to better physical protection. Hence, enforcing the packets to pass through these nodes intrinsically raises the bar for the attackers. Enforcement of such policies can, for example, block malicious commands injected by CrashOverride [2]. Policy checking is orthogonal to F-Pro and thus left outside of the scope this paper.

In summary, our design goals are: (1) providing smart grid with verifiable provenance information for message authentication; (2) developing a low-latency, cryptographic mechanism for provenance verification; (3) developing flexible solution for supporting various communication models.

IV. F-Pro: Fast and Flexible Provenance

This section provides the detailed description of our proposed solution, named F-Pro, to meet the design goals.

TABLE I: Key Communication Models in Substation Automation & Remote Control

Use Case Description	SCADA Master	Field Service Device	High-volt. Substation	Substation VPN	Substation Gateway	Substation HMI	RTU	PLC	IED	Maximum Delivery Time Allowed
SCADA Control/Monitoring	S		M	M	M, S		M, S	D	D	$<$ 100ms
Operation on Substation HMI						S	M	D	D	$>$100ms
Reporting by Field Device	D				M, S	D	M, S	S	S	$>$100ms
Automated Control in Substation								S	D	10-100ms
Status Update in Substation								D	S, D	2-10ms
Protection to Switchgear								D	S, D	$<$ 2ms
Remote Maintenance		S		M	D, M			D	D	-
Local Maintenance		S				D		D	D	-

S: Source, D: Destination, M: Intermediary

Overview: As attempted in [12]–[14], a practical way to introduce additional security into existing ICS is to deploy transparent, bump-in-the-wire (BITW) devices. Namely, while legacy, existing ICS devices can send and receive messages in an as-is manner, added BITW devices intercept messages and provide extra protection and verification without affecting endpoints. F-Pro is designed on top of such BITW devices. As can be seen in Figure 1, to provide a comprehensive coverage, an F-Pro-enabled BITW device (or F-Pro device for short) can be introduced for each key communication node (e.g., a SCADA master, IEDs, PLCs, substation gateways, etc.). Alternatively, a smaller number of F-Pro devices could be strategically installed for critical components. One notable advantage of F-Pro is that the BITW devices can be designed with less resource constraint and use latest security technologies (e.g., Trusted Execution Environment) as compared to legacy devices. This will facilitate the secure and automated distribution of keys used in F-Pro via public key infrastructure.

At the high level, an F-Pro device at the sender side intercepts and "wraps" messages for additional security and the one at the receiver side performs verification and security policy enforcement (e.g., ones defined based on provenance information) and then "unwraps" and forwards the original message to the target device. Mediation by F-Pro devices can be done selectively based on, for example, types of messages, target devices, and so forth. This way, F-Pro has minimal impact on system throughput.

To meet the stringent latency requirements, when an F-Pro device *initiates* (i.e., signs) a message for the source node or *verifies* a message for the destination node, we have to avoid public-key based solutions. Therefore our solution is based on pre-shared symmetric keys. The pre-shared symmetric keys are derived for each pair of source and destination nodes. While every node keeps a secret key for themselves, they share a derived key (called *authentication tokens* — AT) with every destination node. For the operation not to be mistaken with public-key based signatures, hereafter we will use the term "generating cryptographic evidence" instead of "signing".

A node is a *witness* when a message passes through it, and it conducts a hash calculation to endorse that it sees the message. On the other hand, when a message is modified (e.g., because of protocol translation) in an intermediate node, the corresponding F-Pro device needs to *extend* the message. The *content* of a *message* can be a command, information, a caveat (e.g., regarding authorization information), or all of them. In case the network topology is not known to the destination node, part of the message content may be a list of nodes involved, on which all the nodes on the path should add their identifier. The *destination* node can verify the source (the sender) and all the witnesses between the source and the destination. In some cases, there can be two or more source or destination nodes on a message delivery path to support advanced smart grid communication models. In the following, we first present the basic construction of F-Pro. Concretely, the F-Pro protocol creates a chain of keyed cryptographic digests derived from the messages which we find as the go-to primitive to build a time stringent provenance scheme. F-Pro supports extended functionality that are required in an ICS such as "Verification by Intermediate Nodes" or "Combine Extend and Intermediate Verify".

F-Pro construction: F-Pro is a tuple of algorithms (Setup, Initiate, Witness, Extend, Verify). *Setup* creates the keys and authentication tokens and distributes them. *Initiate* generates a cryptographic evidence for a message that has been sent by the physical device that the F-Pro device is attached to. *Witness* alters the cryptographic evidence that is already attached to a message that is passing though the physical device that the F-Pro device is attached to. *Extend* is witnessing, where the message is also changed. The F-Pro then alters the cryptographic evidence accordingly. Finally, *Verify* checks the authenticity of the cryptographic evidence and either accepts or rejects.

Initiate-verify: Figure 2 presents the protocol with 2 hops. Removing the witness from the figure leaves the simplest use case of F-Pro with one source node and one destination node. The aim is to allow the user to prove its identity and message integrity to the destination with minimal overhead in terms of time added to the original protocol. Note that, although destination here means an immediate neighbor in network topology, in general, it refers to the eventual recipient of the message. The source possesses a secret key (sk_S) and the destination stores an authentication token (AT_{SD} = hash (sk_S, salt_{SD})) that has been generated by the source node. The authentication request, which proves identity to the destination node is a cryptographic evidence (h_{SD} = hash (hash (sk_S, salt_{SD}), ts)) of a message with no content. There is a salt value that is associated with a particular destination node since there may be more than one possible destination node. The salt value is a randomly generated value that either can be stored or recreated on the fly. To prevent the replay attacks, the evidence should be time-bound. Hence, to send the authentication request, the source adds a timestamp (ts) to the equation. To authenticate a message along with its identity, the source adds the message content (msg) to AT_{SD}, resulting in:

$$\text{h}_{SD} = \text{hash (hash (sk}_S\text{, salt}_{SD}\text{), ts, msg)}$$

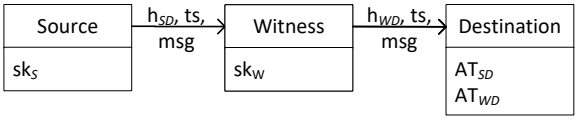


Fig. 2: A Single-Witness Example

The said message is protocol/syntax agnostic and works as long as it is deserialized the same way at both ends. For verification the destination node needs to add the timestamp and the message to its stored authentication token (h' = hash (AT_{SD}, ts, msg)) and check two things; if the timestamp is within limits and if h' equals the received h_{SD}.

This basic case covers some of the communication models discussed in Section II, namely *Automated Control in Substation*, *Status Update in Substation*, *Protection to Switchgear*, and *Local Maintenance*. In the following, we demonstrate the flexibility of this design to support various smart grid communication models before we move to the more time stringent segments.

Adding Witnesses En-route: As discussed in Section II, many smart grid communication models involve multi-hop communication. In some cases, a message is received by an intermediate node and then simply forwarded to the destination. For example, in *Remote Maintenance* scenario, a VPN interface may serve in this way. Besides, in *SCADA Control/Monitoring* case, the high-voltage substation may also be simply forwarding information.

In Figure 2, note that the destination node has two ATs. One for the source node and the other for the witness node (AT_{WD} = hash (sk_W, $salt_{WD}$)). The source node initiates the chain by sending h_{SD} to the witness node. The witness node then naively needs to add its own cryptographic evidence to the message with the same timestamp therefore when the destination node receives the message with two cryptographic evidence, it can verify that the message passed through the witness node. However, the more the witness nodes on the path, the bigger the message becomes with the addition of cryptographic evidence on the path. Instead, the witness node derives its cryptographic evidence in a nested manner using the cryptographic evidence from the source node (or from the previous witness if it is not the first witness) so that it is used as a second key for its witness cryptographic evidence.

$$h_{WD} = \text{hash } (h_{SD}, \text{hash } (sk_W, salt_{WD}), ts, msg)$$

This way, the next witness cannot generate the cryptographic evidence for the initial message from the source and cannot craft the message in a way that it never passed from a previous witness either.

Verification at the destination is done by calculating the nested hash values from the first one to the last (there are only two in this example). And it only checks if the last h' is equal to the received cryptographic evidence.

$$h'_0 = \text{hash } (AT_{SD}, ts, msg)$$
$$h'_1 = \text{hash } (h'_0, AT_{WD}, ts, msg)$$

For verification, the destination node calculates the chain to see if it can recalculate the received cryptographic evidence (h), which is the cryptographic evidence from the last intermediary (or the source if there is no intermediary) to the destination. We present verification loop in Algorithm IV.1.

Algorithm IV.1: Provenance Verification

```
Data: h, ts, messages, numberOfNodesOnThePath
Result: accept/reject
1  h'_0 = hash(AT_source, ts, messages[0])
2  c = 1
3  sourceCounter = 0
4  while c++ ≤ numberOfNodesOnThePath do
5  |  if Witness_c is another source then
6  |  |  sourceCounter++
7  |  h'_c = hash(h'_{c-1}, AT_c, ts, messages[0 : sourceCounter])
8  if h'_c = h then
9  |  accept
10 else
11 |  reject
```

While the Initiate and Witness algorithms' complexities are both constant (O(1)), the complexity of the verification algorithm is linear on the number of intermediaries (O(N), where N is the number of intermediaries). This makes the overall proof and verify complexity of one packet traverse O(N). In the naive approach where all upstream nodes prove provenance to the downstream nodes, the complexity is $O(N^2)$, which highlights the efficiency of our scheme.

Message Transformation (Extend): In *SCADA Monitoring/Control*, it is often the case that protocol translation is performed en route, for example, from IEC 60870-5-104 to IEC 61850 MMS at a substation gateway [10], [15]. In such a case, the destination may require verification of by whom the transformation is performed and consistency between the two messages (before and after translation). In this example, the SCADA master is regarded as a source for IEC 61870-5-104 message while the substation gateway works as another source for the IEC 61850 MMS message. From the receivers' perspective, the complete provenance should include verification on both paths.

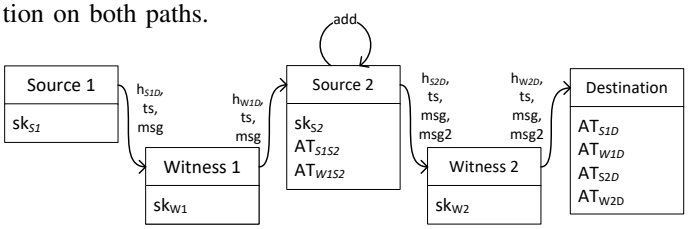


Fig. 3: An Example Where the Second Source Modifies the Message.

In the case with two sources (in Figure 3), the second source partially acts as a witness while concatenating a new (transformed) message and calculating the next ring of the chain. For instance, receiving the inputs "h_{S1D}, ts, msg" a witness would calculate its cryptographic evidence as follows: hash (h_{S1D}, hash (sk_W, $salt_{WD}$), ts, msg). Instead, the source two renders it as follows: h_{S2D} = hash (h_{S1D}, hash (sk_{S2}, $salt_{S2D}$), ts, msg, msg2), *extending* the message. Then, it sends h_{S2D}, ts, msg, msg2 to the next node. Consequently, the Lines 5-6 are executed in the Algorithm IV.1, which

increases the number of messages to be processed at Line 7.

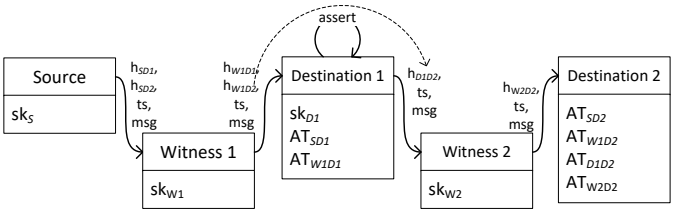


Fig. 4: An Example Where the First Destination Decides if the Message Can Pass Through

Verification by Intermediate Nodes: It is often demanded to authenticate the upstream provenance before forwarding to downstream end field devices. Such a feature may be desired at a boundary of the substation. For instance, F-Pro device associated with a substation gateway may need to perform verification before forwarding messages inward. In this case, we need to enable the intermediate nodes to verify the upstream provenance first, then witness the message, and send it to the target IED. Such a case can be supported by regarding the intermediate nodes as another destination.

Figure 4 is a generic example of two destinations. As in all cases, the source node starts the chain in the above manner. Since the message is targeted to two destinations, the source generates two proofs of provenance, one for destination node one (h_{SD1}) and one for destination node two (h_{SD2}). The witnesses in between then witness both of these cryptographic evidence. When the message arrives to destination one, it runs Algorithm IV.1 for the cryptographic evidence intended for him (h_{W1D1}). Once he verifies the authenticity of the message, it then proceeds to check its assertion. If the assertion is satisfied, then forwards the message witnessing the cryptographic evidence intended for destination 2. The destination 2 then runs the algorithm IV.1 for the cryptographic evidence it receives at the end (h_{D1D2} if there is no witness in between or the cryptographic evidence of the last witness).

Combine Extend & Intermediate Verify: As a variant of the case discussed in the previous subsection, it is also possible that an intermediate node has autonomy to control downstream nodes. For example, a data concentrator may perform message aggregation when reporting measurements from IEDs in the substation (*Reporting by Field Device*). In such cases, an intermediate node is required first to authenticate incoming message(s) and then to send new, derived message to the destination.

This case is the combination of "Adding Witnesses Enroute" in Section IV and the Section 11. First, the intermediate node acts as the first destination and verifies the part of the incoming message that is intended for it as in Section 11. Second, instead of witnessing the message that is passing through (as in Section 11), it adds its own message.

Failure of an F-Pro device itself or its network connectivity: A failure of the F-Pro device can be handled by redundancy. In existing deployment, a critical smart grid device often has a hot standby system, or/and itself may have multiple network interfaces. Redundant F-Pro devices can be connected to the hot standby devices and to each of the standby interfaces of a critical device. Hence, the failure of a single F-Pro device can be tolerated. The degree of redundancy should be decided based on the criticality of associated smart grid devices. Such an architecture would also include a watchdog mechanism that monitors the availability of the F-Pro devices, e.g., based on timeout, to enable automated fail-over.

V. Definition of F-Pro Algorithms

Below we denote the set of messages as "MSG". We subscript to denote an element in this set and we superscript to denote the state of the set. The cryptographic evidence (h) is to be sent along with the plaintext timestamp (ts) and message. See Table II in Appendix VII for the list of notations used in the rest of the paper.

- Setup: $sk_S \leftarrow$ KeyGen($1^{\mathcal{K}}$) – Given the security parameter $\mathcal{K}$, a secret key (sk) is generated by every node in the system.
  $AT_{SD} \leftarrow$ hash(sk_S, D) – Every source node (S), given the secret key and a salt value, generates an authentication token (AT) per destination (D). We assume that destinations for each source is known in advance, which practically holds given the well-defined nature of ICS, and we also assume all the ATs are distributed to the corresponding destinations in a secure manner.
- h_0, MSG $\leftarrow$ Initiate(sk_0, ts, message, D) – Given the secret key, timestamp (ts), a message and the destination, outputs the first cryptographic evidence (h_0) of the chain and the set of messages with one message in — the later nodes in the path may add messages to the set (see "Extend" below).
- $h_c \leftarrow$ Witness(h_{c-1}, D, sk_c, ts, MSG) – Given the previous node's cryptographic evidence, the destination, the secret key, timestamp and the set of messages, outputs a witness cryptographic evidence.
- h_c, MSG' $\leftarrow$ Extend(h_{c-1}, D, sk_c, ts, message, MSG) – Given the previous node's cryptographic evidence, the destination, the secret key, timestamp, a message and the set of previous messages, outputs an extension cryptographic evidence and the extended message set.
- accept/reject $\leftarrow$ Verify(h_c, ts, AT, MSG) – Given the last node's cryptographic evidence (h_c), the timestamp, the set of authentication tokens (that consists of the authentication tokens of the source node and other nodes on the path) and the set of messages, outputs a bit (1 for verify, 0 for reject).

VI. Correctness of F-Pro protocol

In its core, F-Pro uses a chain of keyed cryptographic digests. To create the chain, a function (e.g., cryptographic hash function, HMAC) is employed to map data of arbitrary size to

data of fixed size. We need a one-way function that is collision resistant (e.g., SHA256). Let hash : I_1 x $I_2 \rightarrow$ O be a family of hash functions, a function is collision resistant if $\forall$ PPT adversaries A, $\exists$ a negligible function neg s.t. $P[(I_2, I_2') \leftarrow A(I_1) : I_2 \neq I_2' \wedge (\text{hash}_{I_1}(I_2) = \text{hash}_{I_1}(I_2'))] \leq neg(n)$. In this work we use HMAC construction employing SHA256 and We denote this function as "hash" in the following. Each digest is derived from one previous digest in the chain. That is the notion of chaining. Let || denote concatenation. In this study, we use hash(i_1, i_2,..., i_m) to mean hash($i_1||i_2||...||i_m$).

We define the **correctness** of F-Pro as: $\forall$ MSG_1, ..., MSG_m $\in \{0,1\}^*$, if [$sk_S \leftarrow$ KeyGen($1^{\mathcal{K}}$) **AND** $AT_{SD} \leftarrow$ hash(sk_S, D) for $D \in$ [1, ..., # of destinations per node]] for $S \in$ [1, ..., # of nodes]; (h_0, MSG) $\leftarrow$ Initiate(sk_S, ts, MSG_1, D); [$h_x \leftarrow$ Witness(h_{x-1}, D, sk_x, ts, $MSG^{current}$) **OR** (h_x, MSG^y) $\leftarrow$ Extend(h_{x-1}, D, sk_x, ts, MSG_y, MSG^{y-1})] for x $\in$ [1, ..., # of nodes on the path] and y $\in$ [1, ...,# of sources on the path]; **then** Verify(h_x, AT, MSG^y) = 1 with probability 1. In short, as long as a secure hash function (e.g., HMAC-SHA256) is used, F-Pro does not allow any forgery and tampering of provenance chain. We have conducted formal security proof of the correctness of F-Pro. See our security proof in Appendix VII. After the successful verification, the verified provenance information is handed over to policy checking, which is outside of the core F-Pro algorithms.

VII. Security Proof

Table II lists the main notations used in F-Pro as well as security proof shown next.

The main security feature we are after from a cryptographic evidence for provenance verification is that an adversary cannot forge one at any step (hop) of the protocol. This includes three things. First, an adversary cannot *initiate* a cryptographic evidence to be verified by an authentication token, without knowing the corresponding secret key or the authentication token. Second, an adversary cannot *witness* or more importantly *extend* a hash chain, without knowing the secret key or the corresponding authentication token (from the witness or the extender to the destination) of the actual node that is to *witness/extend* the message. Third, an adversary cannot remove a *witness/extend* block from the chain.

Our scheme is secure if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the probability that $\mathcal{A}$ wins in the below game is negligible in $\mathcal{K}$ (the security parameter).

Forgery Game: If any node on the path of the message from a source to destination deviates from the honest behavior, then the destination will detect it with high probability. We define a forgery game to be played between the adversary $\mathcal{A}$ who acts as the malicious parties and the challenger $\mathcal{C}$ who plays the role of the destination and honest users.

- **Setup:** $\mathcal{C}$ defines the hash function runs the "Setup" algorithm to initialize the environment, generate the secret keys for all honest nodes $\mathcal{HN}$ and calculates the authentication tokens for each of them and stores the authentication tokens. $\mathcal{C}$ returns hash function details and the node IDs to $\mathcal{A}$. $\mathcal{A}$ generates a set of secret keys, corresponding authentication tokens and node IDs that represent the malicious nodes, and returns authentication tokens and node IDs to $\mathcal{C}$.
- **Query (Message initiation and witnessing/extending session):** $\mathcal{A}$ selects IDs $n_D, n_S \in \mathcal{HN}$ and asks $\mathcal{C}$ to either *initiate* a message for destination node n_D from source node n_S and return the cryptographic evidence to $\mathcal{A}$ or gives a message and a cryptographic evidence as well and asks $\mathcal{C}$ to *witness/extend* it on behalf of the node S to destination D. $\mathcal{A}$ repeats this interaction polynomial many times.
- **Challenge:** $\mathcal{A}$ returns the set of messages MSG and the last cryptographic evidence of the witness/extender (if any, otherwise sends the cryptographic evidence of the initiator), where MSG has a message initiated or witnessed/extended by $D \in \mathcal{HN}$. $\mathcal{C}$ outputs an accept/reject signal based on the verification result.
- **Winning Condition:** $\mathcal{A}$ wins if $\mathcal{C}$ accepts and either MSG has a message that was not initiated/witnessed/extended by an intended honest node D (represented by $\mathcal{C}$ in the game) or $\mathcal{A}$ has removed a witness/extender (without reusing a cryptographic evidence generated by $\mathcal{A}$ – since otherwise, it can ask $\mathcal{C}$ to witness to a cryptographic evidence $h_{\mathcal{A}}$ that has been generated by $\mathcal{A}$, then drop the result and reuse the cryptographic evidence $h_{\mathcal{A}}$, which is not removing a cryptographic evidence from the chain).

Our Provable Provenance scheme is secure according to Definition VII if the underlying hash function is secure.

We reduce the security of our scheme to that of the underlying hash function. If a PPT adversary $\mathcal{A}$ wins security game of our scheme with non-negligible probability, we use it to construct another PPT algorithm $\mathcal{B}$ who breaks collision resistance of the hash function with non-negligible probability. $\mathcal{B}$ 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 $\mathcal{A}$. We consider three scenarios.
First scenario: identity forgery. The Adversary $\mathcal{A}$ is trying to generate a cryptographic evidence for an honest node (*initiate*).

Setup: $\mathcal{HC}$ picks a secure hash function (*hash*) from a hash function family – we use SHA256 in our construction of the HMAC employed – and passes the parameters to $\mathcal{B}$, then $\mathcal{B}$ passes it to $\mathcal{A}$. $\mathcal{A}$ chooses two sets of node IDs, where one set consists of malicious nodes $\mathcal{MN}$, the other consists of the honest nodes $\mathcal{HN}$ and sends them to $\mathcal{B}$. For each node $n_s \in \mathcal{HN}$, $\mathcal{B}$ randomly generates a secret key sk_S of size $\mathcal{K}$ and stores them locally. $\mathcal{B}$ computes authentication tokens AT_{SD} (= *hash*(sk_S || n_D)) from any source node $n_S \in \mathcal{HN}$ to destination nodes $n_D \in \mathcal{HN} \cup \mathcal{MN}$ and stores them locally. $\mathcal{B}$ then sends the AT_{SD} to $\mathcal{A}$, for all $n_D \in \mathcal{MN}$.

Query: $\mathcal{A}$ chooses n_S, $n_D \in \mathcal{HN}$, a message msg and a timestamp ts. $\mathcal{A}$ sends them to $\mathcal{B}$ and requests a cryptographic evidence h_{SD} from node n_S to node n_D, generated by *initiate*. $\mathcal{B}$ computes cryptographic evidence h_{SD} = *hash*(*hash*(sk_S || n_D) || ts || msg) and sends it to $\mathcal{A}$. Or $\mathcal{A}$ also chooses a message msg_e and provides a previous cryptographic evidence

TABLE II: Notations

Notation	Description
sk_i	A secret key that belongs to a node i. $i \in \{0, 1, \ldots, n\}$ where $i = 0$ refers to the source and n is the destination.
$\mathcal{K}$	The security parameter
AT	The authentication token. It is subscripted to denote its "from" and "to" (s.a., AT_{SD} to denote it is the authentication token that servers between the source, S, and the destination D).
h	The Cryptographic evidence every node prepares when they initiate, witness or extend a message. h_i is the cryptographic evidence for the i^{th} node on the path of the message. To denote a cryptographic evidence's "from" and "to", we use the same notation with AT, where h_{SD} means the cryptographic evidence from S to D.
S,W,D	The source, witness and destination node IDs, respectively.
ts, msg	A timestamp, a message.
MSG	A set of messages. When a message is initiate by a source, if it is extended by any downstream node, the new message is added to this set.
$\mathcal{A}$	The blackbox PPT adversary.
$\mathcal{HC}$	The hash challenger.
$\mathcal{C}$	The challenger in the proof that plays the role of the destination node and the honest nodes. It uses the output of $\mathcal{A}$ to win against $\mathcal{HC}$, which means to break the security assumption.
$\mathcal{B}$	$\mathcal{B}$ plays the role of $\mathcal{C}$ against $\mathcal{A}$ and acts as the adversary to $\mathcal{HC}$.
$\mathcal{HN}$, $\mathcal{MN}$	The sets of honest nodes and the malicious nodes

h_p and asks $\mathcal{B}$ to *extend*. Note that if msg_e is null, then it is *witnessing*. $\mathcal{B}$ computes cryptographic evidence h_{SD} = *hash*(h_p || *hash*(sk_S || n_D) || ts || msg || msg_e) and sends it to $\mathcal{A}$. $\mathcal{A}$ can repeat this queries for polynomial many times.

Challenge: $\mathcal{A}$ prepares and sends n_S, $\text{n}_D \in \mathcal{HN}$, a message msg', a timestamp ts', sk'_S, AT'_{SD} and h'_{SD} to $\mathcal{B}$, where msg' and ts' have not already been queried together for source node n_S and destination node n_D.

If $\mathcal{B}$ verifies (if h'_{SD} is equal to *hash*(*hash*(sk_S || n_D) || ts' || msg')), then $\mathcal{A}$ wins. For this, $\mathcal{A}$ needs to either compute the inner hash value *hash*(sk_S || n_D), or needs to calculate outer hash value without knowing *hash*(sk_S || n_D).

$\mathcal{A}$ **cannot win:** If $\mathcal{A}$ wins with a non-negligible (in $\mathcal{K}$) probability p, then $\mathcal{B}$ uses it to break the security of *hash* ($\mathcal{B}$ finds a collision in *hash*). This means $\mathcal{A}$ either found sk_S, or it found $\text{sk'}_S \neq \text{sk}_S$ while the computed hash value is the same. For the first case, the probability that $\mathcal{A}$ can find sk_S is negligible (in $\mathcal{K}$) since otherwise, we can define a $\mathcal{B}$' that breaks the preimage resistance of *hash*. For the second case, if $\mathcal{A}$ has computed AT' correctly as *hash*(sk'_S || n_D), then $\mathcal{B}$ can send [sk_S || n_D] and [sk'_S || n_D] (as the collision to output AT'_{SD}) to the hash challenger $\mathcal{HC}$. Otherwise ($\text{AT'}_{SD} \neq \text{AT}_{SD}$), $\mathcal{B}$ can send [AT'_{SD} || ts' || msg'] and [AT_{SD} || ts' || msg'] (as the collision to output h'_{SD}) to the hash challenger $\mathcal{HC}$. In either case, $\mathcal{B}$ can break the collision resistance of *hash* with probability p - neg($\mathcal{K}$). In our construction, we have employed SHA256 which is known to be secure (preimage resistant and collision resistant) at the moment. Therefore p - neg($\mathcal{K}$) must be negligible, thus p is negligible, therefore $\mathcal{A}$ can win the game with negligible probability.

We provide proof for *initiate*. The proof for *witness/extend* and removing a witness from the chain are very similar. For *witness/extend*, to win the game, instead of the initiation of a message/cryptographic evidence, $\mathcal{A}$ extends a message/cryptographic evidence without knowing the corresponding secret key or authentication token. And for $\mathcal{A}$ to remove a *witness/extend* from the last cryptographic evidence, it needs to output (compute) the previous cryptographic evidence without knowing it beforehand. Both can trivially be reduced to the problem in the proof above.

Authentication Token Security: The security of the authentication tokens depends on the preimage resistance of the employed hash function family. Simply put, if a probabilistic polynomial time adversary can find the secret key sk, given an authentication token (= hash(sk, salt)), then we can break the preimage resistance of the chosen hash function. Therefore if the hash function employed is preimage resistant, then the authentication tokens are secure, in the sense that they do not reveal the secret key to their holders.

VIII. Implementation and Evaluation

We first elaborate our prototype implementation of BITW F-Pro module on an embedded platform. Specifically, we have selected BeagleBoard-X15 (called BeagleBoard for short) [8], and then discuss the performance and compatibility evaluation.

A. F-Pro Implementation

The BITW device is deployed for each smart grid device to be protected and is responsible for transparently intercepting packets that are incoming into or outgoing from the protected device. F-Pro implementation utilizes either *iptables* (in case F-Pro only handles IP-based protocols such as IEC 60870-5-104 and IEC 61850 MMS) or *ebtables* (in case of IEC 61850 GOOSE), along with *NFQUEUE* to intercept packets of interest and pass them to the F-Pro module deployed in the user space. The F-Pro module is responsible for parsing an incoming packet to extract the accompanying cryptographic evidence if any, verifying and/or generating cryptographic evidence according to the algorithm discussed in Section IV, and compiling an outgoing packet. If the cryptographic evidence is not valid or any other issue is found, the packet is dropped.

In future work, it is possible to implement F-Pro module also in kernel space for better performance.

B. Performance Evaluation

Our evaluation of F-Pro focuses on its time overhead, including the time spent on cryptographic hash calculations, other packet-processing operations, and the end-to-end delay when deploying our implementation in an smart-grid-alike network environment. We vary the hash functions and key sizes used by F-Pro, and compare it with the overhead of public-key encryption scheme of RSA. We also report the message size overhead of F-Pro and throughput of our implementation.

Time spent on cryptographic hash calculations and other packet processing operations: Figure 5a and Figure 5b show the breakdown of the time spent for each operation during Initiate and Witness respectively. The performance overhead for Extend is similar to Witness, thus we do not include the corresponding figure to save space. As can be seen, the time spent on cryptographic hash operation is very short (i.e. less than 40 microseconds even for HMAC construction with SHA512), even when implemented on a low-cost embedded device [8]. In fact, it only constitutes a small fraction of the overall packet processing latency. Other necessary operations (such as preparing, creating, and sending the data packets) take significantly longer time than the hash operations. Note that Initiate takes less time than Witness. This is because Initiate only processes a packet once, while Witness needs to receive/process/send a packet twice — first for the incoming packet to its associated smart grid device, and again for the processed packet leaving the associated device. In comparison, Figure 5c shows the overheads for signing a packet using RSA signatures. We have employed openSSL library [16] to sign the same amount of data that we required in our setting. With the key size of 2048 bits — which is the minimum of the ones considered secure — and with SHA256 as the digest function, the signing operation alone takes 8 ms and dominates the other packet processing operations. Thus, it is obvious that they are not suitable for the use in time stringent operations (e.g., those require end-to-end delay of a few milliseconds). Figure 5d shows the time spent for all the four core operations by F-Pro on BeagleBoard [8]. As found in the figure, the time overhead of F-Pro operations remains low regardless of the hash function employed. Figure 5e shows the time spent by F-Pro for the hashing operations for the cryptographic evidence or the verification using different key sizes. In our implementation, the AT is stored by the destination F-Pro device, so the time spent for verification does not change with the key size. The other three operations compute ATs on the fly using the secret keys, hence the time consumed increases with the key size. Figure 5f shows the overall time spent by F-Pro to perform each operation for different key sizes. Again, there is little increase in the overall time overhead with the increase of the secret keys utilized. Overall, F-Pro's latency is short enough when used with practically-secure key size and hash function.

Communication overhead: The communication overhead is very minimal at 320 bits (h: 256 bits and ts: 64 bits) per packet, and most importantly its size remains the same regardless of the hop count.

Throughput measurements: We have measured the throughput of F-Pro on BeagleBoard. Figure 5g shows throughput measurements using *iperf* over TCP with message size 1514 Bytes and with different computational latency added on BeagleBoard. The *initiate* operation on the BeagleBoard takes 140-160 usec as shown in Figure 5a. Therefore, as seen in Figure 5g, the F-Pro on BeagleBoard can support 39.5 MBits/s at a source F-Pro device. And, the *witness* operation takes 280-300 usec as shown in Figure 5b, therefore the BeagleBoard can support 27 MBits/s at an intermediate device. This translates to 3420 packets per second for the *initiate* operation. The computation time needed for *verify* operation and to support multicast can be longer, but F-Pro can still sustain about 866 packets per second throughput when each packet incurs an extra 1ms of computation time. In addition, our packet throughput measurement is a conservative lower bound since the packet sizes in both MMS and GOOSE are usually smaller than 1514 Bytes. Therefore, the throughput of F-Pro on BeagleBoard is sufficient for typical traffic intensity.

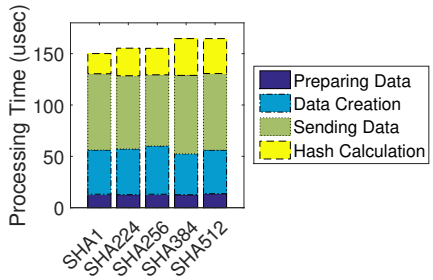
TABLE III: Extra end-to-end latency introduced by F-Pro

1 hop	2 hops	3 hops	4 hops	5 hops
<0.7ms	<1.7ms	<2.6ms	<3.5ms	<4.5ms

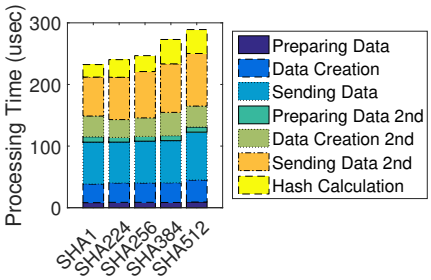
End-to-end latency overhead: To measure end-to-end latency overhead introduced by F-Pro, we set up a system with multiple hops using virtual machines. The total time overhead introduced by F-Pro is shown in Table III. The overhead is measured as the difference between the original end-to-end latency of a system without F-Pro devices and that of a system where an F-Pro device is attached to each device along the path. While 1-hop case only involves F-Pro devices initiating and verifying, the other cases intermediate nodes witnessing. The overall latency for 1-hop case, which corresponds to the most latency-stringent scenarios, the latency added by F-Pro is low enough to meet 2ms message delivery time. The cases with more than three hops are for SCADA communication and the time requirement is not as strict as those with one or two hops, and the added latency is not significant.

Compatibility testing: We tested our implementation on Electric Power and Intelligent Control (EPIC) testbed [17]. EPIC is a smart power grid testbed (open to external access), which includes generation, transmission, microgrid, and smart home components. Each of the four segments of EPIC has its PLCs, IEDs, and communication systems in a fiber optic ring network. EPIC testbed utilizes IEC 61850 MMS and GOOSE protocols as summarized in [18].

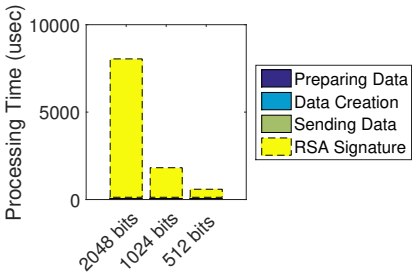
We connected our BeagleBoard F-Pro implementations into the generation segment (as seen in Figure 6) of the testbed for adding and verifying the cryptographic evidence as defined in Section IV. We demonstrated compatibility with the smart grid devices and network in the testbed for both IEC 61850 MMS and GOOSE communication.



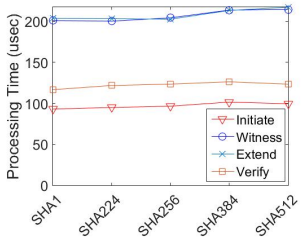
(a) Processing time of F-Pro Initiate operation under different hash algorithms.



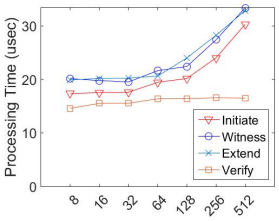
(b) Processing time of F-Pro Witness operation under different hash algorithms.



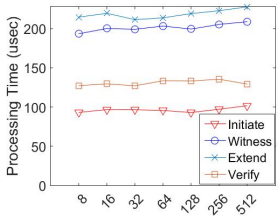
(c) The time required for RSA based signatures when initiating for different key sizes.



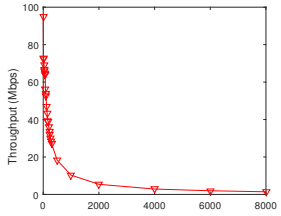
(d) The overall time spent for each F-Pro operation under different hash functions.



(e) The time spent at the hashing step for each F-Pro operation for different key sizes (in Bytes).



(f) The overall time spent for each F-Pro operation for different secret key sizes (in Bytes).



(g) The effect of computation (in usec) per packet on the embedded device to the throughput.

Fig. 5: Time Overhead Measurements

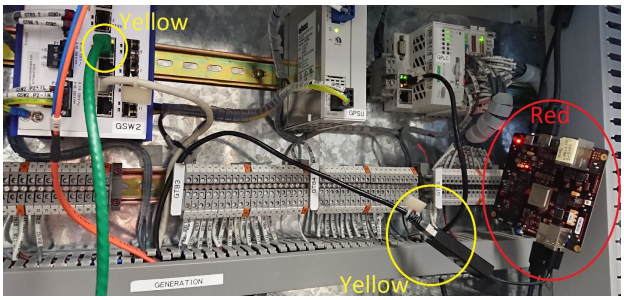


Fig. 6: F-Pro integration into EPIC Testbed [17]. The red circle indicates one of F-Pro devices connected to a switch in the generation segment (GSW2) with the two cables highlighted with yellow in bump-in-the-wire manner.

IX. Related Work

Several schemes employ public key infrastructure and ameliorate it for the path authentication with aggregate signatures [19]–[21]. Aggregate signature enables multiple senders to sign different messages without increasing the signature size. To reduce computation and communication complexity, techniques such as signature amortization [22] and the space-efficient techniques of aggregate signatures [23] have been proposed. However, under stringent latency constraints in ICS, they are no longer options. Path verification in smart grid context was explored in [24], but it focused on demand response services, which are less latency stringent.

Hash chaining technique is used by Google's Macaroons scheme [25] to provide decentralized authorization in a cloud environment. The Macaroons scheme uses nested, chained message authentication codes (MACs) to achieve efficiency and facilitate ease-of-deployment, which are the same two desirable properties our scheme achieves using hash chaining technique. Despite this similarity, the construction of our scheme is rather different from Macaroons, due to the different goals, i.e., provenance vs. authorization.

Bump-in-the-wire (BITW) security solutions have been proposed for ICS systems, as they enable easy deployment and updating of security mechanisms without requiring major upgrade or replacement on existing ICS devices [12]. Closely related to our work are BITW solutions for implementing confidentiality and integrity protection, such as [12]–[14]. [12], [13] deploys BITW devices at each end of the communication, which are responsible for adding security metadata (e.g., MAC) at the sender side and verifying and stripping it at the other end. While our focus is on the provenance of ICS messages, ours also adopts the similar deployment model to maintain compatibility with legacy ICS devices. Unfortunately, all of these do not consider the end-to-end path verification in multi-hop network. In ICS, multi-hop communication is common and often involves protocol translation or checking at middle boxes. Thus, we aims at providing provenance checking including path verification.

X. Conclusions

In this work, we designed and developed F-Pro, a fast and flexible provenance-aware message authentication scheme for smart grid systems. Using substation automation and remote control as a concrete case study, we show that F-Pro, which provides information about the source of messages, message delivery path, and message transformation en route, can counter a number of high-risk security threats. At the same time, F-Pro meets stringent latency requirements (e.g., below 2ms in the most time-critical operations), and, as a BITW solution, F-Pro supports a variety of different deployment settings.

As future work, we will develop enhanced policy checking mechanisms, including systematic and efficient consistency verification on message transformation.

REFERENCES

[1] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[2] "Crashoverride malware," [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA17-163A, 2017, (Date last accessed on Aug. 18, 2017).

[3] Defence Use Case, "Analysis of the cyber attack on the ukrainian power grid," 2016.

[4] K. Zetter, "Inside the cunning, unprecedented hack of ukraines power grid," http://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/, 2016.

[5] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan *et al.*, "The provenance of electronic data," *Communications of the ACM*, vol. 51, no. 4, pp. 52–58, 2008.

[6] Cleveland, "Iec tc57 security standards for the power system's information infrastructure - beyond simple encryption," in *2005/2006 IEEE/PES Transmission and Distribution Conference and Exhibition*, May 2006, pp. 1079–1087.

[7] F. Hohlbaum, M. Braendle, and F. Alvarez, "Cyber security practical considerations for implementing iec 62351," in *PAC World Conference*, 2010.

[8] "BeagleBoard-X15," [Online]. Available: https://beagleboard.org/x15", 2018, (Date last accessed on May 17, 2018).

[9] IEEE Power and Energy Society, "IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation," 2004.

[10] IEC TC57, "IEC 61850-90-2 TR: Communication networks and systems for power utility automation part 90-2: Using iec 61850 for the communication between substations and control centres," *International Electro technical Commission Std*, 2015.

[11] Department of Energy, "Communications requirements of smart grid technologies," [Online]. Available: https://www.energy.gov/gc/downloads/communications-requirements-smart-grid-technologies, 2010, (Date last accessed on Jun. 8, 2018).

[12] P. P. Tsang and S. W. Smith, "Yasir: A low-latency, high-integrity security retrofit for legacy scada systems," in *IFIP International Information Security Conference*. Springer, 2008, pp. 445–459.

[13] A. K. Wright, J. A. Kinast, and J. McCarty, "Low-latency cryptographic protection for scada communications," in *International Conference on Applied Cryptography and Network Security*. Springer, 2004, pp. 263–277.

[14] J. H. Castellanos, D. Antonioli, N. O. Tippenhauer, and M. Ochoa, "Legacy-compliant data authentication for industrial control system traffic," in *Proceedings of the Conference on Applied Cryptography and Network Security (ACNS)*, July 2017.

[15] D. Mashima, P. Gunathilaka, and B. Chen, "Artificial command delaying for secure substation remote control: Design and implementation," 2018, to appear in IEEE Transactions on Smart Grid.

[16] J. Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL: Cryptography for Secure Communications*. " O'Reilly Media, Inc.", 2002.

[17] "Electric power and intelligent control (epic) testbed," [Online]. Available: https://itrust.sutd.edu.sg/testbeds/electric-power-intelligent-control-epic/, 2018, (Date last accessed on Feb. 12, 2019).

[18] A. Siddiqi, N. O. Tippenhauer, D. Mashima, and B. Chen, "On practical threat scenario testing in an electric power ics testbed," in *Proceedings of the Cyber-Physical System Security Workshop (CPSS), co-located with ASIACCS*, June 2018.

[19] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology — EUROCRYPT 2003*, E. Biham, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 416–432.

[20] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 465–485.

[21] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 74–90.

[22] M. Zhao, S. W. Smith, and D. M. Nicol, "Aggregated path authentication for efficient bgp security," in *Proceedings of the 12th ACM conference on Computer and communications security*. ACM, 2005, pp. 128–138.

[23] D. Boneh, C. Gentry, B. Lynn *et al.*, "A survey of two signature aggregation techniques," *RSA cryptobytes*, vol. 6, no. 2, pp. 1–10, 2003.

[24] D. Mashima, U. Herberg, and W.-P. Chen, "Enhancing demand response signal verification in automated demand response systems," in *Innovative Smart Grid Technologies Conference (ISGT), 2014 IEEE PES*. IEEE, 2014, pp. 1–5.

[25] A. Birgisson, J. G. Politz, U. Erlingsson, A. Taly, M. Vrable, and M. Lentczner, "Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud." in *NDSS*, 2014.