

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 [15] and the Ukraine power grid attacks [1], [10], [25], exploited the deficiency of checking for monitoring and control in the cyber infrastructure of the smart grid. For example, the success of CrashOverride malware [1] used in the Ukraine incident in 2016 resulted from the fact that there were ICS 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 [20]. 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 ICS can significantly raise 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 [9] using public-key cryptography (see, e.g., [12]).

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 ICS that addresses both requirements. Our solution meets the stringent latency requirement in ICS 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 ICS devices as well as to allow flexible deployments. Our measurements on a low-cost BeagleBoard-X15 platform [2] 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 [14].

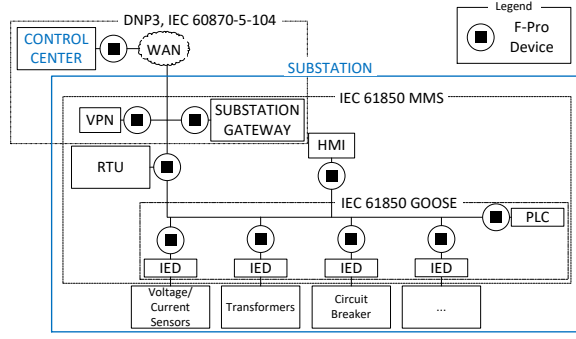


Fig. 1: Substation Automation and Remote Control: A Conceptual Architecture, Key Components, and Protocols and Protection with F-Pro Devices

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 [13].

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 [13]. 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 performance requirements derived from the public guideline [11], [14]. 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 [14]).

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 to decide if the corresponding message is legitimate. For instance, by enforcing policy in terms of the message source and its delivery path, we can block malicious commands injected by CrashOverride [1]. Policy checking is orthogonal to F-Pro and thus left outside of the scope this paper.

In summary, our design goals are: (1) providing ICS devices with verifiable provenance information for message authentication; (2) developing a low-latency, cryptographic mechanism for generating and verifying provenance information; (3) developing flexible solution for supporting various smart grid setting and 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 [8], [22], [24], 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. Also, it is possible to implement F-Pro functionality as a new software module on existing devices to help reduce the deployment overhead.

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 ICS 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 (e.g., firewall/gateway) or combining extend and intermediate verify (e.g., message aggregation by a data concentrator when reporting measurements from IEDs in the substation). Due to page restrictions, we defer the said sections to the technical report [?].

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 through 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. We also defer the detailed definitions of algorithms, the correctness definition and the security proof, all of which can be found in the technical report [?].

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 ($AT_{SD} = \text{hash}(sk_S, \text{salt}_{SD})$) that has been generated by the source node. The authentication request, which proves identity to the destination node is a cryptographic evidence ($h_{SD} = \text{hash}(\text{hash}(sk_S, \text{salt}_{SD}), ts)$) 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

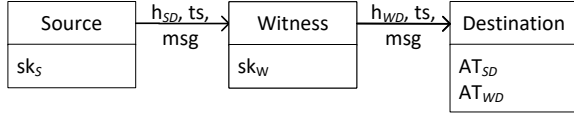


Fig. 2: A Single-Witness Example

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 with:

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

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 ($h' = \text{hash}(AT_{SD}, ts, msg)$) 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 authentication tokens (ATs). One for the source node and one for the witness node ($AT_{WD} = \text{hash}(sk_W, salt_{WD})$). 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.

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

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  $h'_0 = \text{hash}(AT_{source}, ts, messages[0])$ 
2  $c = 1$ 
3 sourceCounter = 0
4 while  $c++ \leq \text{numberOfNodesOnThePath}$  do
5   if  $Witness_c$  is another source then
6     | sourceCounter++
7     |  $h'_c = \text{hash}(h'_{c-1}, AT_c, ts, messages[0 : \text{sourceCounter}])$ 
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 [13], [18]. 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.

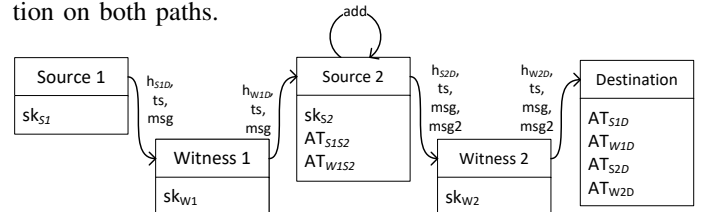


Fig. 3: An Example Where the Second Source Modifies the Message.

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: $\text{hash}(h_{S1D}, \text{hash}(\text{sk}_W, \text{salt}_{WD}), \text{ts}, \text{msg})$. Instead, the source two renders it as follows: $h_{S2D} = \text{hash}(h_{S1D}, \text{hash}(\text{sk}_{S2}, \text{salt}_{S2D}), \text{ts}, \text{msg}, \text{msg2})$, *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 counted into account at Line 7.

V. 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 hereafter) because it implements two onboard Ethernet interfaces and therefore suitable for use as a BITW device. Detailed hardware specification can be found in [2].

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 *eatables* (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 ICS-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 4a and Figure 4b 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 [2]. 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 ICS device, and again for the processed packet leaving the associated device. In comparison, Figure 4c shows the overheads for signing a packet using RSA signatures. We have employed openssl library [23] 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 4d shows the time spent for all the four core operations by F-Pro on BeagleBoard [2]. As found in the figure, the time overhead of F-Pro operations remains low regardless of the hash function employed. Figure 4e 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 4f 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 4g 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 4a. Therefore, as seen in Figure 4g, 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 4b, 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.

End-to-end latency overhead: To measure end-to-end latency

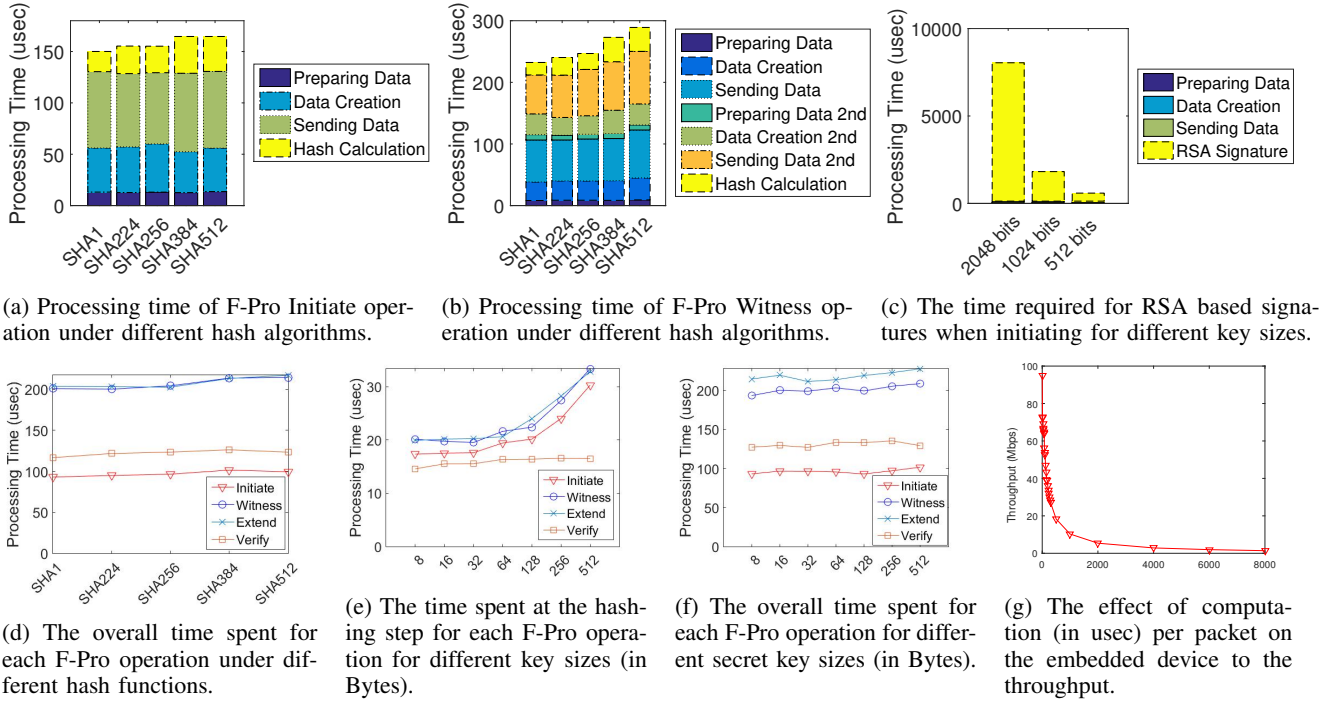


Fig. 4: Time Overhead Measurements

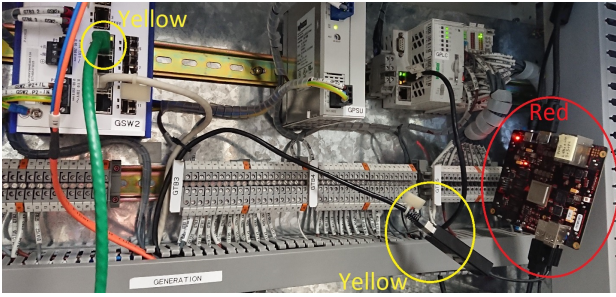


Fig. 5: F-Pro integration into EPIC Testbed. The red circle indicates one of F-Pro devices connected to a switch in the generation segment (GSW2) with the two cables highlighted with yellow in bump-in-the-wire manner.

TABLE II: 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

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 II. 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 [3]. 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 [21].

We connected our BeagleBoard F-Pro implementations into the generation segment (as seen in Figure 5) of the testbed for adding and verifying the cryptographic evidence as defined in Section IV. We demonstrated compatibility with the ICS devices and network in the testbed for both IEC 61850 MMS and GOOSE communication.

VI. RELATED WORK

Several schemes employ public key infrastructure and ameliorate it for the path authentication with aggregate signatures [4], [7], [16], [17]. 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 [26] and the space-efficient techniques of aggregate signatures [6] have been proposed. However, under stringent latency constraints in ICS, they are no longer options. Path verification using a public-key scheme in smart grid context was explored in [19], but it focused on demand response services, which are less latency stringent.

Hash chaining technique is used by Google's Macaroons scheme [5] 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 [22]. Closely related to our work are BITW solutions for implementing confidentiality and integrity protection, such as [8], [22], [24]. [22], [24] 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 aim at providing provenance checking including path verification.

VII. CONCLUSIONS

In this work, we designed and developed F-Pro, a fast and flexible provenance-aware message authentication scheme for ICS. Using smart grid 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 in an ICS (e.g., below 2ms in the most time-critical smart grid operations), and, as a BITW solution, F-Pro supports a variety of different ICS deployment settings. As future work, we will develop enhanced policy checking mechanisms, including systematic and efficient consistency verification on message transformation.

REFERENCES

- [1] Crashoverride malware. [Online]. Available: <https://www.us-cert.gov/ncas/alerts/TA17-163A>, 2017. (Date last accessed on Aug. 18, 2017).
- [2] BeagleBoard-X15. [Online]. Available: <https://beagleboard.org/x15>, 2018. (Date last accessed on May 17, 2018).
- [3] Electric power and intelligent control (epic) testbed. [Online]. Available: https://itrust.sutd.edu.sg/wp-content/uploads/sites/3/2019/02/EPIC_technical_details_v1.2.pdf, 2018. (Date last accessed on Feb. 12, 2019).
- [4] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Unrestricted aggregate signatures. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming*, pages 411–422, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- [5] Arnar Birgisson, Joe Gibbs Politz, Ulfar Erlingsson, Ankur Taly, Michael Vrabie, and Mark Lentzner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *NDSS*, 2014.
- [6] Dan Boneh, Craig Gentry, Ben Lynn, et al. A survey of two signature aggregation techniques. *RSA cryptobytes*, 6(2):1–10, 2003.
- [7] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [8] John Henry Castellanos, Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa. Legacy-compliant data authentication for industrial control system traffic. In *Proceedings of the Conference on Applied Cryptography and Network Security (ACNS)*, July 2017.
- [9] 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*, pages 1079–1087, May 2006.
- [10] Defence Use Case. Analysis of the cyber attack on the ukrainian power grid. 2016.
- [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] Frank Hohlbaum, Markus Braendle, and Fernando Alvarez. Cyber security practical considerations for implementing iec 62351. In *PAC World Conference*, 2010.
- [13] 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.
- [14] IEEE Power and Energy Society. IEEE Standard Communication Delivery Time Performance Requirements for Electric Power Substation Automation. 2004.
- [15] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.
- [16] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 465–485, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [17] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 74–90, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [18] Daisuke Mashima, Prageeth Gunathilaka, and Binbin Chen. Artificial command delaying for secure substation remote control: Design and implementation, 2018. To appear in *IEEE Transactions on Smart Grid*.
- [19] Daisuke Mashima, Ulrich Herberg, and Wei-Peng Chen. Enhancing demand response signal verification in automated demand response systems. In *Innovative Smart Grid Technologies Conference (ISGT), 2014 IEEE PES*, pages 1–5. IEEE, 2014.
- [20] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, et al. The provenance of electronic data. *Communications of the ACM*, 51(4):52–58, 2008.
- [21] Ahnaf Siddiqi, Nils Ole Tippenhauer, Daisuke Mashima, and Binbin 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.
- [22] Patrick P Tsang and Sean W Smith. Yasir: A low-latency, high-integrity security retrofit for legacy scada systems. In *IFIP International Information Security Conference*, pages 445–459. Springer, 2008.
- [23] John Viega, Matt Messier, and Pravir Chandra. *Network Security with OpenSSL: Cryptography for Secure Communications*. ” O’Reilly Media, Inc.”, 2002.
- [24] Andrew K Wright, John A Kinast, and Joe McCarty. Low-latency cryptographic protection for scada communications. In *International Conference on Applied Cryptography and Network Security*, pages 263–277. Springer, 2004.
- [25] Kim Zetter. Inside the cunning, unprecedented hack of ukraine’s power grid. <http://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/>, 2016.
- [26] Meiyuan Zhao, Sean W Smith, and David M Nicol. Aggregated path authentication for efficient bgp security. In *Proceedings of the 12th ACM conference on Computer and communications security*, pages 128–138. ACM, 2005.