Technical report 15-02, Nov. 30, 2015
Bar Ilan University, Dept. of Computer Science

Topology-based Plug-and-Play Key-Setup for Industrial Control Systems

Amir Herzberg

School of Computer Science
Bar-Ilan University,
Israel
Email: herzbea@cs.biu.ac.il

Yehonatan Kfir

School of Computer Science
Bar-Ilan University,
Israel
Email: yehonatank@gmail.com

***Abstract*—**

The deployment of cryptography in Industrial Control Systems (ICS) is critical, yet very limited. One reason is the difficulty associated with key management and upgrading devices. This problem is further exacerbated by the fact that remote, unmanned devices are often vulnerable to key exposure attacks.

We present the *topology-based key setup protocol ()* to facilitate the *plug and play* deployment of cryptography. eases the adoption of security by eliminating the need to manually initialize every device with its own key, as is currently done. Furthermore, limits the impact of key exposures by ensuring both *perfect forward secrecy* and *proactive key refresh*, re-establishing security after exposure.

We analyze the properties of the protocol and show sufficient topology conditions for its applicability. Then, we evaluate its applicability in typical ICS networks. For example, for the IEEE 118-bus benchmark for power systems, about 70% of the devices can be securely initialized using , in the presence of a compromised device or a compromised communication link.

1. Introduction

Critical infrastructures such as power grids, water networks, and oil/gas distribution systems, all depend on wide-area Industrial Control Systems (ICS) networks. These networks are used to communicate with both human operators, via a Human Machine Interface (HMI) [1], and devices such as Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs). The availability of the communication to remote devices is critical. Consequently, the ICS networks are built with redundancy to contain the failure of even a single link or site.

Ensuring redundancy of the communication is not enough; we must also protect the integrity and confidentiality of the communication, including commands sent to the devices and values sent back. Manipulating these values, or even sending arbitrary commands, could cause significant damage. The exposure of values sent may also be abused in different ways. Thus, cryptography is essential for security against attackers with powerful capabilities, in particular, *Man-in-the-Middle (MitM)* attackers.

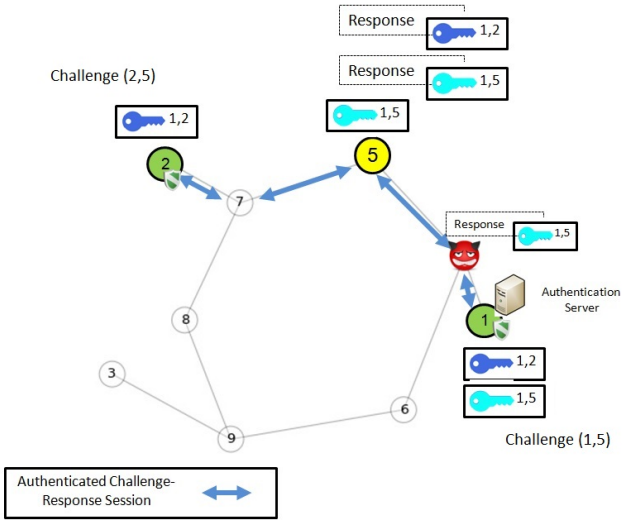

Figure 1. The IEEE 9-bus model [1] of a small ICS network. The topology-based key setup protocol provides plug-and-play initial key setup and proactive key refresh. The protocol uses multiple authenticated challenge-response sessions, between a client (device 5) and several trusted devices (devices 1 and 2). In this way, the network provides security from an attacker that controls part of the routes between the client and the trusted devices

That said, cryptography is rarely deployed in ICS networks. One reason is that most ICS components do not yet support cryptographic functions. Another barrier to cryptographic deployment, and the focus of this manuscript, is *key management*. ICS systems are constructed from thousands of components, often spread over a wide geographic area. Each component that supports cryptography needs to be se-

curely initialized with keys. Current systems require manual initialization of the keys in each device, a process that many operators find risky, inconvenient, and expensive.

Once deployed, ICS devices are often located at remote unmanned sites, and prone to private-key exposure due to weak cryptography, side-channels, and software vulnerabilities. If key exposure is suspected, the keys need to be manually re-initialized. This is a significant problem, and likely one reason for the low motivation to deploy cryptography in ICS devices.

We address these challenges by presenting a *topology-based key setup* scheme to facilitate the *plug and play* deployment of cryptography, with *proactive key refresh*. Our topology-based scheme leverages the fact that the topology of ICS networks is usually known and quite stable, with considerable redundancy (for resilience). This facilitates *plug and play* deployment, in which upgrading to secure devices requires only new software/hardware, but does not require manual setup of each device. Our scheme supports *proactive key refresh*, allowing a completely automated recovery of remote, unmanned devices from private key exposures.

We take advantage of the known topology and redundancy to *authenticate* messages, and in particular to set up keys, rather than for confidentiality, as in previous works, e.g., [2], based on secret-sharing a message and sending each share over a disjoint path. We authenticate the sender by measuring the *distance* from the sender to multiple trusted nodes, based on standard assumptions about internet routing. This also allows us to perform proactive key refresh, similar to Canetti et al. [3], but without depending on attack-detection at the remote (and usually unmanned) node.

For example, consider the network in Figure 1, which is based on the IEEE 9-bus model [1]. This system is built from 9 devices. For this example, devices 1 and 2 are assumed secure (and already upgraded). The server knows the topology, as shown in the figure. As device 5 is upgraded, the server will ask it for evidence that it is connected with path of length 2, to devices 1 and 2, through edges 1a and 2a, respectively. Only after receiving this evidence, will the server authenticate device 5 and its cryptographic keys.

To facilitate our design and analysis, we present the *ICS model*, formalizing properties of ICS networks:
(1) Known topology: As availability is the main concern, ICS network topology is usually known, mostly stable [4]–[7], and must be monitored [8].
(2) Trusted nodes: ICS networks usually contain several highly-secure and trusted nodes, such as control centers.
(3) Known public key: One or more of the trusted nodes has a private key, with the corresponding public key known to all (upgraded) nodes.
(4) Safe recovery: An adversary may corrupt some nodes, exposing all of their secrets; however, upon recovery from corruption, nodes return to run the protocol as designed, with the correct public key for trusted node(s).
(5) Disjoint paths: Most nodes in ICS networks have (at least) two disjoint shortest paths to (different) trusted nodes.

CONTRIBUTIONS. We make the following contributions:

(1) A topology-based plug-and-play initial key setup scheme for large-scale ICS networks.

(2) A topology-based proactive key refresh scheme, that does not require (manual) attack-detection capabilities in remote devices.

(3) The *ICS model*, including modeling observations about ICS networks (see above and in Section 2): known topology, trusted nodes, known public key, safe recovery, and disjoint paths. This model may have additional applications in the design of security mechanisms for ICS networks.

(4) Evaluation of the security and applicability of the scheme to typical, realistic ICS networks, using the IEEE ICS models of [1].

1.1. Related Work

Several works [9]–[12] discuss the significant risks to infrastructure, due to the fact that most of the devices in ICS networks are insecure legacy devices. These works argue that it is critical to deploy security in ICS networks and devices.

Recently, several works suggested mitigation of these vulnerabilities, mainly by requiring the upgrade to devices with cryptographic modules [13] [14]. Newer protocols, such as DNP3sec, suggest the cryptographic authentication of sensitive commands [15]. Other works suggest using trusted-computing to verify that the firmware did not change [16].

However, even after upgrading to devices with cryptographic modules, a major barrier still lies in the key management system. The key management system must provide a unique key for each device and handle key compromises. Choi et al. [17] suggest an architecture for efficient key management in ICS systems, based on the Iolus [18] framework. However, they did not provide mechanisms for efficient key setup, as presented in our approach. Many works ignore the key setup phase or assume the device keys are initialized in a secure environment,(e.g., pre-installed manually on each device, via secure channel [19] or in a "secure zone" [20]). In this case, each device must be manually configured before deployment.

Several works investigate cryptographic schemes that are appropriate for ICS systems. The main concern is that ICS systems are time-critical, and latency is often unacceptable. In addition, a large number of devices use serial communication and may not work well with IP-based cryptographic protocols. Several works deal with these issues, suggesting authentication with low-latency for both IP and serial communication [21]–[24]. These aspects are complementary to our work; we focus on key setup and refresh, and do not discuss specific cryptosystems.

2. Model

2.1. Network Model

We model the ICS network as an undirected hypergraph, $G = (V, E)$, where $N = |V|$ denotes the numbers of

devices (nodes), and E is a set of hyper edges representing connections between devices. Some edges are simple edges representing point-to-point communication, and some are hyper-edges representing a connection to multiple devices on the same interface.

A device can send messages to other devices. The message may pass through several intermediary devices that act as routers, before reaching its destination. Every device can block, pass, or change messages that pass through it.

Every device in the network has an identifier that uniquely represents it in the network. An example for such an identifier can be a combination of the device IP and MAC address. For simplicity, we denote the identifier of device $v \in V$, by v.

2.1.1. Devices and Adversary. Since many ICS networks were created without a focus on security, most of their devices do not support cryptography. We call this group of devices, *legacy devices*. However, in recent years, the heightened awareness regarding vulnerabilities due to lack of security mechanisms, has created an increased demand to upgrade legacy devices. The *upgraded devices* support cryptographic modules and need to be initialized with keys. These devices also need to have a recovery plan to receive a new key in case of (suspected) key leakage. Let $L \subset V$ denote the legacy devices.

Because availability is a primary concern, at least some devices in the ICS network must be well monitored to ensure (with high probability) that they will not be compromised. We call these devices *trusted devices*, $T \subset V - L$. Since this high level of security requires large operational efforts, often only a small portion of the upgraded devices are also trusted. Some of the non-trusted devices may be *compromised*; both legacy and upgraded (but not trusted) devices may be compromised.

Examples of trusted devices are the control center servers. These servers are responsible for monitoring physical measurements from field devices and sending them commands. Since these servers are critical for the availability of the ICS, they are highly-secure.

Another one of the *trusted devices* is the authentication server s, which has a known public key, $s.pu$. It also has a private key that is shared with all of the trusted devices and with part of the upgraded devices. Using these keys, the server can send and receive encrypted and authenticated messages.

On each graph $G = (V, E)$, we define a coloring function $\phi : V \times V \rightarrow \{Legacy, Upgraded, Trusted, Compromised\}$, which defines the type for each device in the network. Using this definition, a *Legacy* or *Upgraded* device that was compromised will change its color to *Compromised*.

We consider an attacker $\mathcal{A}$, who controls all *Compromised devices*[1]. The attacker tries to disrupt key setup by an upgraded device, to register its own key for some device, or to learn the key setup in the server for some (upgraded) device.

We define n_A*-nodes attacker* as an attacker that controls n_A devices. From the coloring function definition it is clear that $n_A = |A| = |\{v \in V \ s.t. \ \phi(v) = Compromised\}|$ devices.

The attacker is able to initiate, delay, block, or manipulate messages that pass through its devices.

The attacker is able to eavesdrop on messages in the entire network, even messages that do not pass through its devices.

2.1.2. Routing Model. The *routing method* defines the way each device forwards incoming messages. We model the following routing methods:

Source-routing: each device can set the route in the network, for messages that it initiates, by setting a *route* in the message. The route contains the sequence of devices that relay the message until it reaches its destination. The only exception is that compromised devices are not obliged to forward the message as per the route carried in the message.

Shortest-path routing: messages are sent on the shortest-path between the source and destination device. Routing in a shortest-path network is formulated as a function $\Re : V \times V \rightarrow V$, which receives the current device and the destination, and returns the neighbor of the current device, to which the message is forwarded $\Re(current, destination)$, such that the sequence of forwarding from source to destination is always the shortest path. If there is more than one shortest-path route between the source and destination, the shortest-path routing function consistently chooses the same route.

Legacy and *Upgraded* devices always send messages according to the routing method: the routing list in source-routing networks; the $\Re$ function in shortest-path network; and $\Re_{\mathcal{A}}$ in adversarial routing network. In contrast, *Trusted* and *Compromised* devices are not bound by the routing method and can freely select the edge from which to forward each message.

2.2. Protocol Model

A *topology-based key setup* protocol π is a message-driven-protocol [25] that has three types of participants in its execution:

Server - A *Trusted* device that is initialized with a public key $s.pu$ and correlated private key $s.pr$. In addition, the server is initialized with the network topology $G = (V, E)$, the coloring function ϕ, the routing method $\rho \in \{$*source*,-*shortest-path*$\}$, the security parameter 1^l, and $\Re$ for non source-routing networks. At the end of the protocol execution, the server has three possible outputs: *Alert*; *Success* with a pair (k_c,c) of key k_c of device c; and *Timeout*, when it waits for messages longer than a predefined time threshold.

Client - An *Upgraded* device that is initialized with the server's public key $s.pu$, the routing method $\rho \in \{$*source*,-*shortest-path*$\}$, and the security parameter 1^l. At the end of a successful execution, this device will register its key k_c.

1. Controlling a device effectively controls all of its links; for simplicity, we do not discuss an attacker that is able to control only specific links.

Collaborator - A *Trusted* device that has a shared secret key with the server $k_{s,i}$; this key is different for each collaborator.

The goal of the protocol is to set the same secret key at the server and client, $k_{s,c}$. The ability to securely set such a shared key with a device $v \in V$, depends on the topology of the network $G = (V, E)$, the type (color) of each device ϕ, the routing method ρ, and the routing function $\Re$. Let $P(v, G, \phi, \rho, \Re)$ be a *topology availability predicate* that returns 1 if several topology conditions are met for device $v \in V$.

We define the *availability* of protocol π with respect to predicate P, as the fraction of devices that have $P(v, G, \phi, \rho, \Re) = 1$ from all the devices in the network.

In Section 4 we present the implementation details of the topology-based key setup protocol.

3. Problem Formulation

Our problem formulation is based on the execution model by Bellare et al. [25], and extended to support known topology and routing models. Execution of a topology-based key setup protocol depends on the network properties and on the attacker capabilities. As input, the execution receives the attacker algorithm $\mathcal{A}$, a topology-based key setup protocol π, a topology predicate P, and a security parameter 1^l. We denote this execution by **EXEC**$(\mathcal{A}, \pi, P, 1^l)$. Due to length restrictions, we present the extended execution model in [26].

The execution process is *adversarial* in the sense that the attacker $\mathcal{A}$ chooses all the network parameters: ρ, ϕ, and the topology $G = (V, E)$. In addition, the attacker chooses one *Upgraded* device as the client and one *Trusted* device as the server.

The output of the execution is the attacker state $\sigma_{\mathcal{A}}$, and one of the following *results*: (1) *"Failure"* - if the server registers a key that is not the same as the client key (probably because of an attacker); (2) *"Alert"* - if the server detects an attacker that prevented the key setup; (3) *("Success", k_c^{OUT})* - if the server registers the same key as the client, k_c^{OUT}; and (4) *"Timeout"* - if the server output is "Timeout".

We define the following properties of topology-based key setup protocols.

Secrecy. A key-setup protocol ensures *Secrecy* if no PPT attacker can retrieve any information about the key from the protocol messages. In other words, there is no probabilistic polynomial-time attacker that can distinguish the key from a randomly-generated string of the same length. Formally:

Definition 1. **[Secrecy]**

Protocol π ensures *Secrecy with respect to predicate* P, if $\left|Pr\left[IND_{\mathcal{A},\mathcal{A}_1,\pi,P}(l) = 1\right] - \frac{1}{2}\right|$ is a negligible function (in security parameter l), for all PPT attackers $\mathcal{A}$ and $\mathcal{A}_1$, and where $IND_{\mathcal{A},\mathcal{A}_1,\pi,P}(l)$ is defined in Algorithm 1.

Correctness. A key-setup protocol ensures *Correctness*, if whenever the server outputs a key k_c^{OUT} for specific client c, then, with overwhelming probability, c outputs the same key k_c^{OUT}. In addition, if the server outputs Alert, then, with overwhelming probability, there is an attacker in the network (i.e., no false alerts). The formal definition is simple and, due to length restrictions, is presented in [26].

$IND_{\mathcal{A},\mathcal{A}_1,\pi,P}(l)$

1. $\mathcal{A}_1$ is choosing $k_0, k_1, S_{ID}^0, S_{ID}^1 \leftarrow \{0,1\}^l$. With them, it creates two activation messages m_0, m_1 *s.t.* $m_i = \{k_i, S_{ID}^i\}$.
2. A random bit is chosen, $b \xleftarrow{\$} \{0,1\}$
3. A successful execution, with activation message equal to m_b, is chosen randomly,
$(\text{"}Success\text{"}, k_c, \sigma_{\mathcal{A}}) \xleftarrow{\$}$
EXEC$(\mathcal{A}, \pi, P, 1^l)$ *where* $m_{init} = m_b$
4. Return 1 if $\mathcal{A}_1(k_c^{OUT}, \sigma_{\mathcal{A}}) = b$, and 0 otherwise.

Algorithm 1: Indistinguishability experiment

Guaranteed Key-Setup. A key-setup protocol ensures *Guaranteed Key-Setup* with respect to predicate P, if, with overwhelming probability, executions terminate successfully (and correctly) - even in the presence of an attacker. Trivially, Guaranteed Key-Setup cannot be ensured if the attacker can permanently block messages of the protocol and thereby prevent termination. Hence, we expect protocols to ensure Guaranteed Key-Setup only for *eventually delivering attackers*, i.e., attackers that do not permanently block messages between non-compromised devices.

Bounded Termination. - A key-setup protocol ensures Bounded Termination if the protocol's execution time is bounded, possibly as a function of the network topology $G = (V, E)$.

Definition 2. **[Bounded Termination]** Let t_0 be the time the execution game started and $G = (V, E)$ the network topology.
Protocol π ensures Bounded Termination, if there exists $T_{EXEC}(G)$ s.t. for every attacker $\mathcal{A}$, and the execution is finished after the time $t_0 + T_{EXEC}(G)$:
EXEC$^{\textbf{SYN}}(\mathcal{A},\pi,P,1^l,t_0)) \in$
$\{\text{"}Alert\text{"}, \text{"}Success\text{"}, \text{"}Failure\text{"}, \text{"}Timeout\text{"}\}$

Forward Secrecy. - A key-setup protocol ensures Forward Secrecy if compromising a single key will not allow the compromise of other keys that were set before the first one was compromised.

In order to achieve this, the protocol must replace the keys that it uses for encrypting and authenticating the data. In addition, these data keys must be created randomly and change their values in a non-deterministic way.

Proactive Security. - A key-setup protocol ensures Proactive Security if the key is able to recover from being compromised, even without the need to be detected.

Instead of searching for compromised devices, this property requires that the protocol initiate mechanisms to deal with compromised devices. For example, this might be

done by changing the device's keys frequently and re-authenticating them.

In addition, this property divides the network coloring function ϕ into time slots. Each time slot is T_P long. We denote the coloring function at time t as $\phi(t)$. In proactive security, the protocol will use the non-compromised devices at time $\phi(t)$ to decrease the number of compromised devices at $\phi(t+1)$. This helps in resisting an attacker that tries to increase the number of compromised devices.

4. Topology-based Key-Setup Protocol (TobKES)

The goal of the protocol is to provide cryptographic keys to upgraded devices, without requiring manual installation. The protocol is executed whenever new keys are needed, and specifically upon upgrading a device or to recover from (possible) compromise of the secret keys of the device.

The protocol uses a public key encryption scheme ξ and a MAC scheme $\mathcal{M}$, as defined at [27] . We denote a that uses these schemes as $^{\xi,\mathcal{M}}$. In cases we would like to emphasize that $\mathcal{M}$ is not relevant for the discussion, we will omit it.

After creating the symmetric key k, the server authenticates the key holder's identity. Using challenge-response sessions, the server validates that the key holder is the client that it claims to be. The challenge-response sessions validate the topological location of the key holder. By that, under several conditions that will be discussed, the key holder is authenticated.

4.1. Protocol Design

Before initiating the protocol, the client $c \in V$ is loaded with the server's public key $s.pu$, and the security parameter 1^l. In order to initiate the protocol, the client generates a random key $k \xleftarrow{\$} \{0,1\}^l$.

In addition, the server is loaded with the network graph $G = (V, E)$, the device-type (coloring) function ϕ , the routing model $\Re$, and the number of compromised devices that it should handle, n_A.

The client activates the protocol session by sending an *activation message*, m_{Init}, to the server. The activation message contains a randomly chosen session ID S_{ID} and the device random key k. In addition, the client sends its previous key shared with the server k_c^{OUT}. If it is the first time the client is requesting a key, k_c^{OUT} will be set to $Null$. These values are sent encrypted using ξ with the server public key $s.pu$. In addition, the client sends its identification (e.g., its IP address, as described in Section 2.1), unencrypted.

Using the device identifications, the server finds the location of the device in the topology. These identifications are not considered trusted and the server will have to validate the claimed device location.

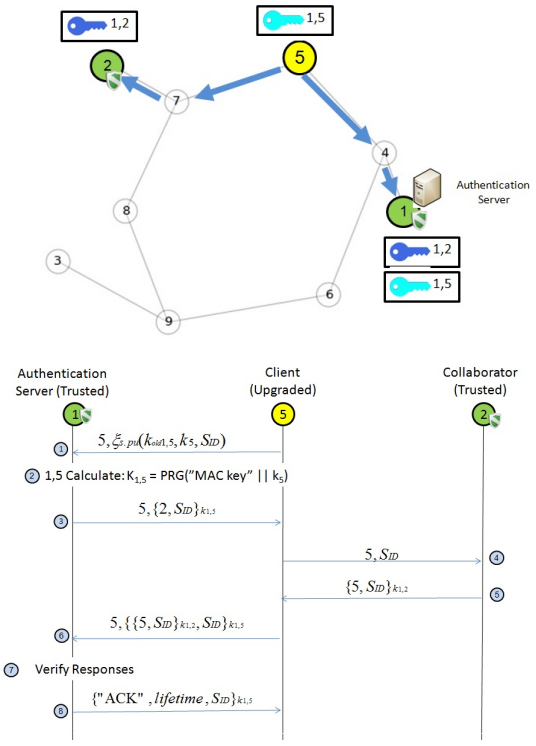

Figure 2. Example for a successful key-setup session. Device 5 receives keys, after two challenge-response sessions, with device 1 and device 2. Device 1 is the authentication server.

Using the generated key k, and a pseudo-random-function (PRF), the server and client derive a new symmetric authentication key $k_c^{AUTH} = PRF_k(\text{"}Authentication\text{"})$, as well as a shared secret key $k_c^{OUT} = PRF_k(\text{"}Client\_Key\text{"})$. k_c^{AUTH} will be used with the MAC scheme $\mathcal{M}$ to authenticate all the messages between the server and the client during the key setup protocol. k_c^{OUT} will be used as the registered of the client.

After generating k_c^{AUTH}, the server selects a group of $2n_A+1$ collaborators from the trusted devices $T_c \subset T$. One of the collaborators can be the authentication server itself.

If $2n_A + 1$ collaborators are not available, the server will use the amount that it can use, as long as there are at least n_A+1 collaborators. If the number of collaborators is below n_A+1, the server will not continue with the protocol execution and will neglect the client request for key setup.

In order to validate the client location, the server sends the client the identification of the chosen collaborators T_c. For each collaborator, the client sends the session ID as a challenge. In response, each collaborator sends back the session ID and the client identification, authenticated with the key the collaborator has with the server.

After the client receives all the responses, it sends them to the server.

A challenge-response session is defined *successful*, if (1) the session ID and the client ID in the responses are as expected; (2) all the messages between the server and

the client are authenticated with the key k_c^{AUTH}; (3) each response is authenticated with the appropriate collaborator key k_i^{AUTH}.

The protocol behavior is defined relative to the number of devices that can be under control of the attacker n_A. Using this parameter, the protocol defines three behaviors:

Key Registry at the Server: If $n_A + 1$ sessions succeeded, the server will register the key k_c^{OUT} for device c, and will send an *ACK message* to the client, through all of the collaborators. The ACK message contains the string "ACK", the session ID S_{ID}, and the key lifetime. After the lifetime period, the key will no longer be used to authenticate messages between the server and the client. Henceforth the client will have to initiate new key setup sessions with the server.

Key Registry at the Client: Upon receiving one ACK message from the server and that ACK message is authenticated with the key k_c^{AUTH}, the client will register its long time key k_c^{OUT}.

We denote the maximal network delay between two devices as T_{delay}. For simplicity, we assume that the processing time of messages in the devices is zero.

Alerting: The server will wait up to $4T_{max}$ time after sending the challenges to the client. The server output will be Alert if there are no n_A+1 successful challenge-response sessions.

An example for a successful key setup session for $n_A = 1$ can be seen in Figure 2.

4.2. Protocol Analysis

In this section we will prove that fulfill the requirement from section 2.

Theorem 1. **[Secrecy]**:
For every CPA-secure public-key scheme ξ protocol $^{\xi}$ ensures Secrecy, as defined at Definition 1.

Assume to the contrary that exists $\mathcal{A}_1, \mathcal{A}$, s.t. for all negligible function $negl(l)$:

$|Pr\left[IND_{\mathcal{A},\mathcal{A}_1,\pi,P}(l) = 1\right] - \frac{1}{2}| \ > \ negl(l)$

Using those attackers, we define an attacker on the encryption scheme ξ, $\mathcal{A}_\xi$. Using this attacker, we will prove a contradiction to the CPA-secure property of ξ.

We denote the PRF that is being used by π as PRF : $\{0,1\}^l \rightarrow \{0,1\}^{l'}$, $l < l'$.

Assume that PRF is an ideal random function U, U : $\{0,1\}^l \rightarrow \{0,1\}^{l'}$, $l < l'$. If the property holds for U and not for PRF, then it is easy to build a distinguisher between PRF and U - which is a contradiction to the assumption that PRF is a PRF. Thus, it is sufficient to assume that PRF is ideal.

Following the CPA experiment $PubK^{cpa}_{\mathcal{A}_\xi,\xi}(l)$ [27] :

1. Keys $s.pu, s.pr$ are chosen randomly.

2. The adversary $\mathcal{A}_\xi$ is given as input 1^l, the public key $s.pu$ and oracle access to the encryption scheme. $\mathcal{A}_\xi$ will choose two activation messages m_0, m_1.

3. A random bit $b \xleftarrow{\$} \{0,1\}^l$ is chosen, and then a ciphertext $c = \xi_{s.pu}(m_b)$ is computed and given to the adversary $\mathcal{A}_\xi$.

4. The adversary $\mathcal{A}_\xi$ executes the process $EXEC^{\mathcal{A}_\xi}(\mathcal{A}, \pi, P, 1^l, m_b)$ until the output is "Success". $\mathcal{A}_\xi$ returns $b' = \mathcal{A}_1(k, \sigma_{\mathcal{A}})$.

5. The output of the experiment is 1 if b=b', and 0 otherwise.

According to the assumption:

$| \ Pr\ [\ \mathcal{A}_1(k_c^{OUT}, \sigma_{\mathcal{A}}) = b\ ] \ > \ negl(l) \Rightarrow$

$| \ Pr\ [\ PubK^{cpa}_{k_c^{OUT},\mathcal{A}_\xi,\xi}(l) = 1\ ] \ > \ negl(l)$

and this is contradiction to the assumption that ξ is CCA secure. Thus, if ξ is CPA-secure then $|Pr\left[IND_{\mathcal{A},\mathcal{A}_1,\pi,P}(l) = 1\right] - \frac{1}{2}| \ < \ negl(l)$.

Thus, for every CPA-secure ξ, protocol ensures secrecy.

Theorem 2. **[Correctness]**: For every CPA-secure public-key scheme ξ , and MAC-secure scheme $\mathcal{M}$, and with predicate P^{DET}, protocol $^{\xi,\mathcal{M}}$ ensures Correctness as defined by Definition **??**.

According to the predicate P^{DET}, there are at least n_A+1 disjoint routes between the client client c and the server.

According to protocol design at **Key Registry at the Server**, if the server registers a key, it sends the authenticated ACK message, through all of the collaborators, through at least $n_A + 1$ disjoint routes.

Thus, if the server registers a key k_s^{OUT}, at least one ACK message had reached to the client, and the client registers a key k_c^{OUT}. If not, than the attacker was able to block all the ACK messages from $n_A + 1$ disjoint routes, with its n_A devices. Hence, at least one attacker device was at more than one route. This is contradiction to the disjoint routes assumption.

According to the protocol design at **Key Registry at the Client**, the client registers a key if it receives even one authenticated ACK message from the server.

Assume in negative that exist a PPT attacker $\mathcal{A}$ s.t. $Pr(\textbf{EXEC}(\mathcal{A}, \pi, P, 1^l) = (\text{"}Failure\text{"}, \sigma_{\mathcal{A}})) > negl(l)$.

If **EXEC**$(\mathcal{A}, \pi, P, 1^l)$=Failure, then $k_c^{OUT} \neq k_s^{OUT}$. In that case, the server did not register the same key as the client. Thus, the attacker $\mathcal{A}$ was able (1) to create and authenticate the ACK message without knowing the key k_c^{OUT} or (2) to change and authenticate the key agreement message, or (3) to retrieve the key k_c^{OUT} . Each operation done with non-negligible probability greater than $negl(x)$.

Assume (1) or (2): Than, the attacker $\mathcal{A}$ was able to create message m' and tag tag' for message he did not seen before. This is in contradiction to the assumption that $\mathcal{M}$ is MAC secure.

Assume (3): Then, the attacker was able to retrieve the key k_c^{OUT} from the activation message, and this is contradiction to the Secrecy property.

Thus, for all PPT attacker $\mathcal{A}$ there exists negligible function $negl(l)$ s.t. $Pr(\textbf{EXEC}(\mathcal{A}, \pi, P, 1^l) = (\text{"}Failure\text{"}, \sigma_{\mathcal{A}})) < negl(l)$.

Theorem 3. **[Guaranteed Key-Setup]**: Protocol , with predicates P^{FUL}, ensures *Guaranteed Key-Setup* property, as defined at Definition **??**.

First, we will prove that is bounded protocol by the number: 2 + 4 $|V|$.

Let $G = (V, E)$ be the network graph. According to the protocol details, the client sends a key request message. Than, the server sends the client list of collaborators. The number of collaborator is limited by the number of devices in the network, $|V|$. The client initiate challenge-response session with each collaborator. and the number of messages is twice (for challenge and response messages with each device).

The client send all the responses through all its neighbours, and hence, this limit the number of messages to $|V|$. Upon successful authentication at the server, the server initiate ACK message through all the routes to the device. Those messages number is also bounded by the number of devices in the network.

Summarizing the upper limit for messages: 1+1+2$|V|$+ $|V|$+$|V|$ = 2 + 4$|V|$.

Thus, is a Bounded Protocol.

We will now prove that with predicate P^{FUL} is always finished with key setup.

If $P^{FUL}(c, G, \Phi, \rho, \Re) = 1$, then there are at least $2n_A+1$ disjoint routes between the client c and the server s.

Because of the disjoint routes, attacker that control n_A devices will be able to manipulate or complete maximum n_A challenge-response sessions. There will still be $n_A + 1$ routes between the server and client, without any attacker node. Using the n_A+1 responses, the server will be able to distinguish the client from the attacker, and to authenticate the client.

Now, it is left to prove that the server will receive the n_A+1 responses. Since the attacker is eventually delivering, all the messages on the routes that it does not control, will eventually reach their destination. This includes the challenge-responses sessions, and the ACK from the server to the client.

This complete the proof.

In the synchronous model we will prove also the Bounded Termination property.

Theorem 4. **[Bounded Termination]**: Protocol ensures *Bounded Termination* property, as defined at Definition 2.

Let t_0 be the time the execution process started, and let T_{delay} be the maximal delay of message between neighbor devices. We will prove that the execution time is bounded by $T_bounded = 6\,|V|\,T_{delay}$.

The longest route a message can pass is a route that includes all the devices in the network. Thus, the maximal time delay of a message from a sender to a receiver device is $|V|\,T_{delay}$. We assume that the processing time of a message, at each device, is zero. For simplicity, we will assume that every device in the network waits that period for receiving response, even if the routes for its message's destination is shorter that the maximal route.

According to the design there are 6 synchronous transactions (can be seen in Fig. 2). Each transaction, as explained, is bounded by $|V|\,T_{delay}$.

Thus, the maximal time that it takes to the protocol to execute is: $6\,|V|\,T_{delay}$

5. Experimental Evaluation

In this section we evaluate several practical aspects related to . Since there is no public information on ICS control networks, we assumed that the topology of the control network is similar to the power system topology. This is a reasonable assumption, since the communication lines often pass through the same infrastructure as the power lines. For the power system, we used several IEEE benchmark topologies: IEEE 300, IEEE 118, and IEEE 57 bus systems.

5.1. Availability

We defined the properties with respect to topology predicates P^{DET} and P^{FULL}. These predicates require that the number of disjoint routes be greater than the number of compromised devices. In this section we evaluate the number of devices with $P^{DET}(v, G, \phi, \rho, \Re) = 1$. In other words, these are devices with $n_A + 1$ disjoint routes, where n_A is the number of compromised devices in the network.

The routing method ρ of the networks was chosen to be shortest-path or source-route. In the shortest-path routing, each edge is considered to have a weight of 1.

The coloring function ϕ required that we choose a group of trusted devices and n_A compromised devices. The group of trusted devices we chose were the devices with the higher degree of edges. The size of the trusted devices group was from 1 to 5.

For each device v, we assumed that the n_A compromised devices are located in the worst case, with respect to v's location. In other words, the compromised devices are located on n_A disjoint routes between v and the group of trusted devices (if v has at least n_A such disjoint routes). The number of compromised devices was chosen to be from 1 to 4, $n_A \in \{1, 2, 3, 4\}$

The results are shown in Figures 3 and 4.

From the results, it can be seen that even with a small number of trusted devices (about 5 devices), most of the devices in the network can use in the presence of one compromised device or one compromised link. In IEEE 118 with source routing, more than 80% of the devices can use , while in shortest-path routing, more than 65% of the network can use .

Another conclusion from the results is that the networks present significantly more percentage of devices that available for in the presence of one compromised device, $n_A = 1$, than for a higher number of compromised devices. Motivated by this conclusion, we compared the availability for between different networks. The results can be seen in Figure 5. These results indicate that for most of the evaluated network topologies, even 5 trusted devices are sufficient to have an availability of more than half of the network, in the presence of a single compromised device.

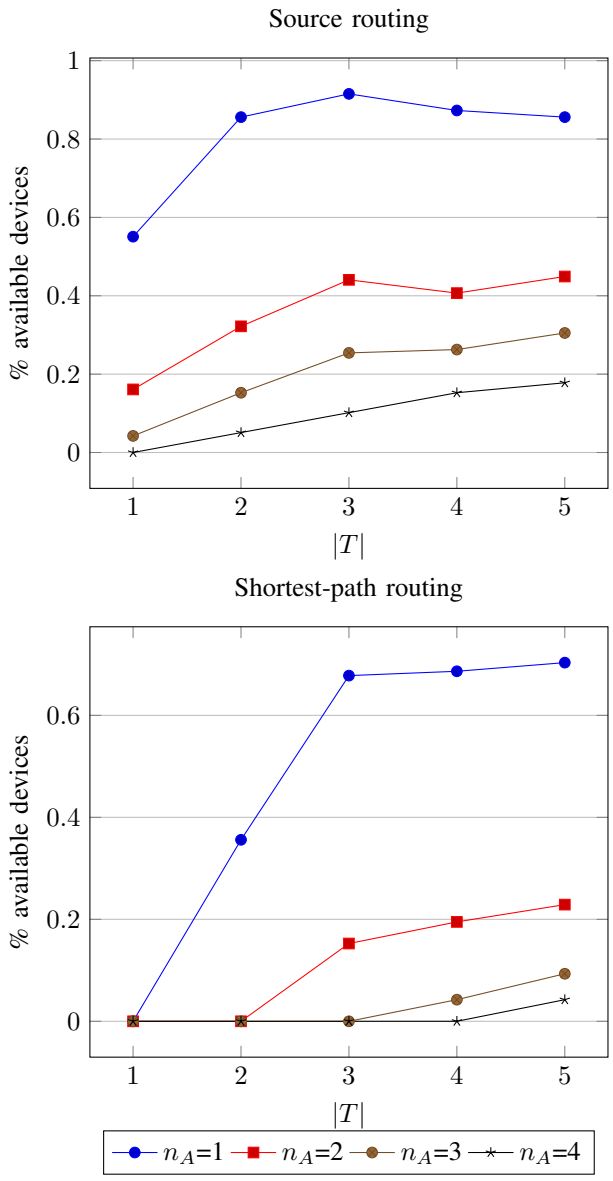

Figure 3. Percentage of devices available for secure ToBKES in IEEE 118. Compromised devices were chosen to be at the worst-case position.

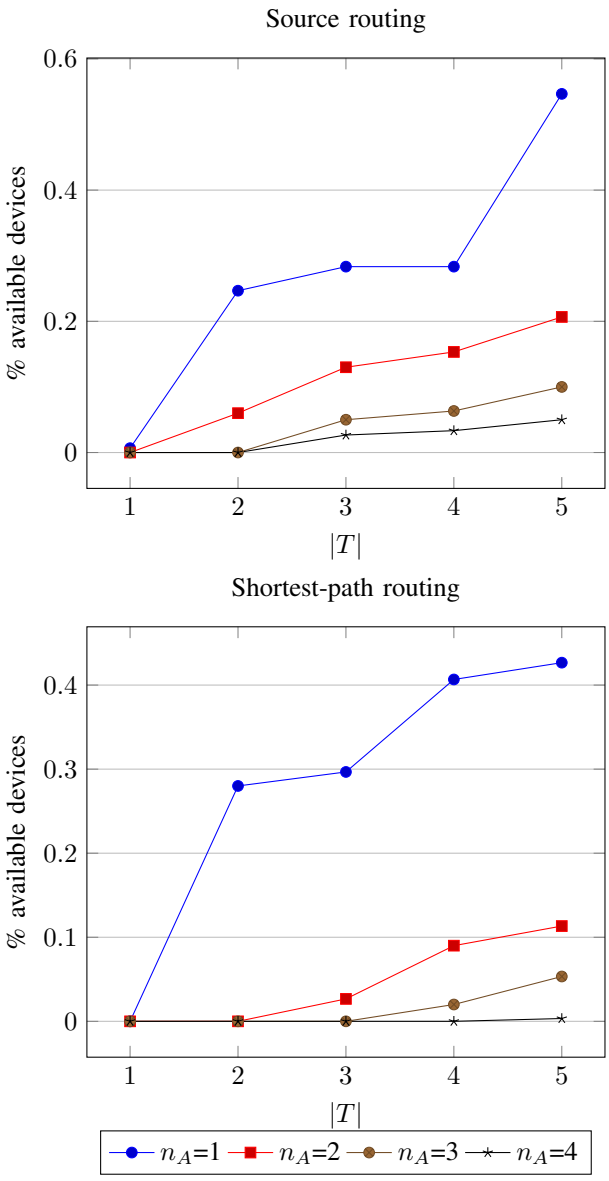

Figure 4. Percentage of devices available for secure ToBKES in IEEE 300. Compromised devices were chosen to be at the worst-case position.

5.2. Proactive Security

In this section we evaluate the applicability of the proactive mechanism. For the networks IEEE 300 and IEEE 118, we assumed a large number of upgraded devices that support : 30%,40% and 50% of the devices in the network. We also assumed that one of those devices is trusted and considered to be the authentication server.

For each device in the network, we evaluated the number of compromised devices that are needed in order to prevent it from receiving a refresh key. Similar to the previous section, we assume that the compromised devices are located at the worst place, with respect to the device location.

We evaluated the number of devices in the network that will be able to receive a refreshed key for a number of compromised devices from 1 to 5, $n_A \in \{1, 2, 3, 4, 5\}$.

The results are shown in Figure 6. From the results, it can be seen that even with partial deployment of for only 30% of the network, more than 85% of the devices will be able to receive a refresh key.

Moreover, the results show that at 50% deployment of , the proactive key refresh is available for more than 80% of the devices, even in the case where there are 5 compromised devices (and keys). This result strengthens as a proactive method for plug-and-play key setup in ICS networks.

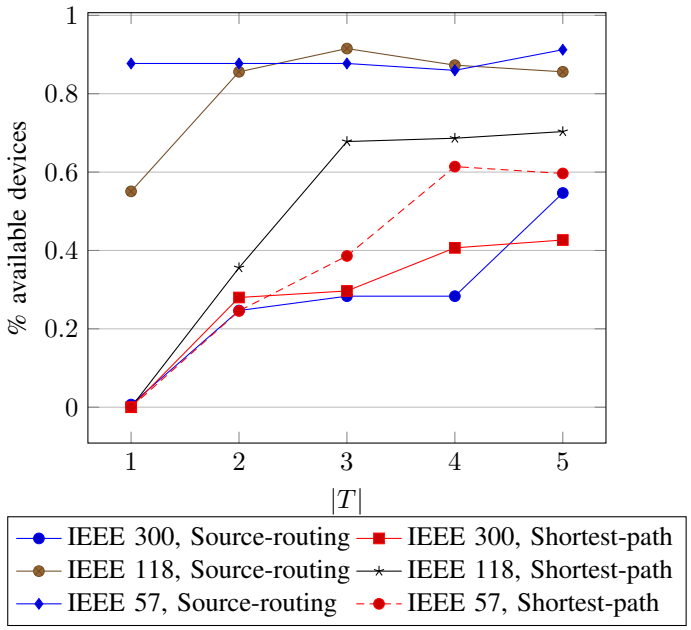

Figure 5. Comparison between different network topologies showing devices available for secure ToBKES in the presence of a compromised device, $n_A = 1$.

6. Conclusions

We present TBKS, a novel method for plug-and-play key setup in ICS networks. The method is based on authenticating the sender location, using authenticated challenge-response sessions. We demonstrate that TBKS has the following properties: Secrecy, Correctness, Guaranteed Key-Setup, Bounded Termination, Forward Secrecy, and Proactive Security.

We evaluated the applicability of TBKS on several IEEE power network benchmarks: IEEE 300, IEEE 118, and IEEE 57. Our results were good in the presence of a single compromised device (or link): even a small number of (deployed) trusted devices will allow a large percentage of the network devices to use TBKS.

In addition, TBKS shows good availability for proactive key refresh. This refresh mode does not require any manual handling, and allows the setup of keys for a large percentage of the network. TBKS allowed the key refresh mechanism to recover from compromised devices, without the need to detect their exact location.

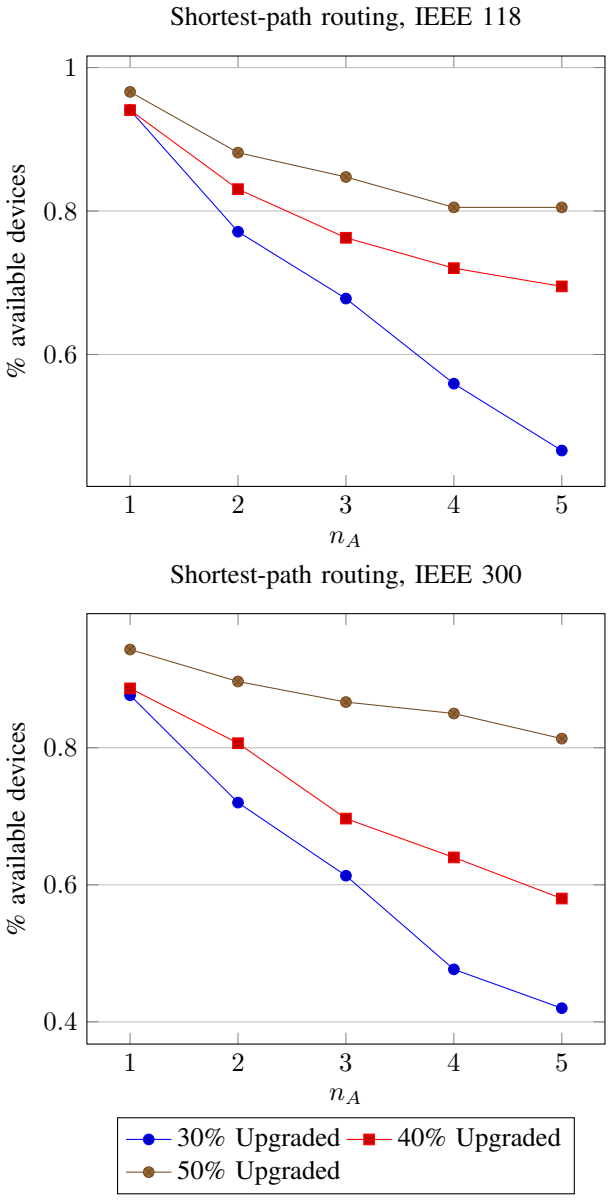

Figure 6. Percentage of devices that can securely receive a refreshed key in the presence of n_A compromised devices. For large scale deployment of ToBKES, most of the network is secured against a small number of compromised devices.

References

[1] A. B. Smith, "IEEE std c37. 1-1994, IEEE standard definition, specification, and analysis of systems used for supervisory control, data acquisition, and automatic control," *IEEE Power Engineering Society*, 1994.

[2] D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly secure message transmission," *J. ACM*, vol. 40, no. 1, pp. 17–47, 1993.

[3] R. Canetti, S. Halevi, and A. Herzberg, "Maintaining authenticated communication in the presence of break-ins," *J. Cryptology*, vol. 13, no. 1, pp. 61–105, 2000.

[4] R. R. R. Barbosa, R. Sadre, and A. Pras, "A first look into scada network traffic," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 518–521.

[5] R. R. R. Barbosa and A. Pras, "Intrusion detection in scada networks," in *Mechanisms for Autonomous Management of Networks and Services*. Springer, 2010, pp. 163–166.

[6] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," in *3rd USENIX Workshop on Hot Topics in Security, HotSec'08*, 2008.

[7] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46, 2007, pp. 1–12.

[8] G. A. Weaver, C. Cheh, E. Rogers, W. H. Sanders, and D. Gammel, "Toward a cyber-physical topology language: applications to NERC CIP audit," in *SEGS'13, Proceedings of the 2013 ACM Workshop on Smart Energy Grid Security, Co-located with CCS 2013*.

[9] M. Franz, "Vulnerability testing of industrial network devices," in *Cisco Critical Infrastructure Assurance Group (Ciag), ISA Industrial Network Security Conference*, 2003.

[10] S. Sridhar and G. Manimaran, "Data integrity attacks and their impacts on scada control system," in *Power and Energy Society General Meeting, 2010 IEEE*. IEEE, 2010, pp. 1–6.

[11] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[12] S. E. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers." in *HotSec*, 2011.

[13] H. Okhravi and D. M. Nicol, "Application of trusted network technology to industrial control networks," *IJCIP*, vol. 2, no. 3, pp. 84–94, 2009.

[14] A. Shahzad and S. Musa, "Cryptography and authentication placement to provide secure channel for scada communication," *Int. J. Secur*, vol. 6, pp. 28–44, 2012.

[15] M. Majdalawieh, F. Parisi-Presicce, and D. Wijesekera, "Dnpsec: Distributed network protocol version 3 (dnp3) security framework," in *Advances in Computer, Information, and Systems Sciences, and Engineering*. Springer, 2006, pp. 227–234.

[16] N. Gottert, N. Kuntze, C. Rudolph, and K. F. Wahid, "Trusted neighborhood discovery in critical infrastructures," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*. IEEE, 2014, pp. 976–981.

[17] D. Choi, H. Kim, D. Won, and S. Kim, "Advanced key-management architecture for secure scada communications," *Power Delivery, IEEE Transactions on*, vol. 24, no. 3, pp. 1154–1163, 2009.

[18] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proceedings of the ACM SIGCOMM 1997 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, 1997, pp. 277–288.

[19] C. Beaver, D. Gallup, W. Neumann, and M. Torgerson, "Key management for scada," *Cryptog. Information Sys. Security Dept., Sandia Nat. Labs, Tech. Rep. SAND2001-3252*, 2002.

[20] "AGA report no. 12. cryptographic protection of SCADA communications: General recommendations."

[21] Y. Wang, "sscada: Securing SCADA infrastructure communications," *CoRR*, vol. abs/1207.5434, 2012.

[22] A. K. Wright, J. A. Kinast, and J. McCarty, "Low-latency cryptographic protection for SCADA communications," in *ACNS 2004*, 2004, pp. 263–277.

[23] P. P. Tsang and S. W. Smith, "YASIR: A low-latency, high-integrity security retrofit for legacy SCADA systems," in *Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008*, pp. 445–459.

[24] Z. Liu, R. Zhang, X. Zhan, D. Li, and R. C. Cheung, "Design and implementation of cryptographic protection for scada communication," *Journal of Convergence Information Technology*, vol. 8, no. 4, 2013.

[25] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 419–428.

[26] A. Herzberg and Y. Kfir, "Technical report 15-02, bar ilan university - department of computer science,http://u.cs.biu.ac.il/ herzbea/security/15-02-tbks."

[27] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC Press, 2014.