Securing Industrial Control System: An End-to-End Integrity Verification Approach

Sye Loong Keoh
School of Computing Science
University of Glasgow
SyeLoong.Keoh@gla.ac.uk

Ken Wai Kin Au
Singapore Institute of Technology
Ken_Au@csa.gov.sg

Zhaohui Tang
School of Infocomm
Republic Polytechnic
Linda_Tang@rp.edu.sg

ABSTRACT

The integrity and authenticity of sensor data in an Industrial Control Systems (ICS) is crucial to ensure the correctness of the processes in industrial facilities. Measurements collected from remotely connected field sensors must have their integrity and authenticity guaranteed, and any malicious tampering to the data must be detected. This paper introduces secure end-to-end data integrity verification for ICS, a security protocol that allows the field controllers to securely aggregate data collected from field devices, while enabling the central controller in the back-end to verify the integrity and data originality from its sources. Thus, compromise of field controllers can be detected swiftly. The aggregated data is protected using Chameleon Hashing and Signatures. It is then forwarded to the central controller for verification, analysis and to facilitate the control of industrial processes. Using the Trapdoor Chameleon Hash Function, the field devices can periodically send an *evidence* to the central controller, by computing an alternative message and a random value (m', r') such that m' consists of all previous sensor data measurements of the field device in a specified period of time. By verifying that the Chameleon Hash Value of (m', r') and the sensor data matches those aggregated by the field controller, the central controller can verify the integrity and authenticity of the data from the field devices. Any data anomaly between field devices and field controllers can be detected, thus indicating potential compromise of field controllers.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Integrity, Authenticity, Industrial Control Systems, SCADA security, Chameleon Hashing, End-to-end Security, Secure Data Aggregation

ICSS '15, December 8, 2015, Los Angeles, CA, USA.

1. INTRODUCTION

Industrial Control Systems (ICS) are used to monitor and control industrial facilities and processes. Many safety critical infrastructures such as power grid, chemical and nuclear plants, gas and water facilities use ICS and Supervisory Control and Data Acquisition (SCADA) to collect measurements from remotely connected field sensors and issue commands to actuators to ensure the smooth operation of the facilities. There is usually a well-defined process model in place in the management of the plant, and the model is usually deployed through ICS. With the increase connectivity in ICS, many of them are now connected to the enterprise network using Commercial-Off-the-Shell (COTS) IT products [14]. Due to the critical nature of the physical devices and services monitored by ICS, it is now a target of attacks in which attackers are incentivised to take down the whole facility and plant with the aim of crippling the national critical infrastructures.

In June 2010, a powerful worm named *Stuxnet* [20, 3] was uncovered and was known to be able to cripple numerous critical infrastructures including many industrial control systems (ICS) used by governmental organizations. *Stuxnet* exploited the vulnerability of an ICS to compromise the field controllers. After taking control of the field devices, the compromised field controller continues to fake "normal" field device readings to be sent to the central controller, thus ensuring that the device compromise remains undetected. As field devices do not protect its reading end-to-end, but rely on the field controller to aggregate the readings, tampering of data can be conducted easily. If end-to-end security were provided, the field controller would not have access to the data, and at the same time, the speed of detecting device compromise would be much faster and recovery actions could be triggered earlier.

However, providing end-to-end security leads to several inefficiency to the existing system architecture: (1) Increase in bandwidth requirement, as each field sensor reading is secured (either signed cryptographically or MAC-ed), intermediaries such as field controller, aggregators can no longer aggregate sensor readings from different sources but to forward each individual message. Thus, resulting in the increase in the number of messages that are being transmitted in the network, leading to network congestion problem. (2) Increase in computational processes, as each field sensor must protect its messages individually, it is required to sign every message it transmits, and the central server has

to verify n number of messages, where n is the number of messages. Unlike the current architecture, messages from multiple sensors are aggregated and signed in a hierarchical manner before they are transmitted to the central controller for analysis and to respond to the situation.

This paper describes the implementation of *delayed-integrity-verification* scheme [15] to provide end-to-end security in ICS, while preserving the efficiency of a distributed architecture. Devices continue to be organized in a hierarchical manner, consisting of field devices and sensors, sending data to field controllers that serve as data aggregator before the data are being forwarded to the central controller. This approach allows the data to be aggregated and then protected using an adapted version of chameleon signature scheme. The field devices periodically generate a chameleon message hash based on the sensor readings previously transmitted to the field controller, to allow for the central controller to verify the data's integrity and authenticity, and whether the field controller misbehaved by tampering with the sensor readings. Regardless of the reliability of transmission medium, end-to-end data integrity and authenticity can be verified by the central controller with minimal effort. Any tampering to the sensor data can be detected by the central controller and an alarm can be triggered to act on the misbehaving field controllers.

The paper is organized as follows: Section 2 provides some background and discusses related work. Section 3 describes ICS use case, the attacker models and security requirements. Section 4 outlines the steps in providing *delayed-integrity-verification* to achieve end-to-end integrity and authenticity verification. Section 5 describes the prototype implementation and discusses preliminary results. Section 6 presents an informal security analysis of the scheme and Section 7 concludes the paper with future work.

2. RELATED WORK

This section provides some background on Chameleon Signatures [18, 17], and discusses related work on secure data aggregation and end-to-end security.

2.1 Chameleon Hashing

Chameleon Hashing was introduced by Brassard and Chaum [9]. It has the same properties as the normal hash function except that it has a trapdoor in built for finding collisions. Without the knowledge of the trapdoor, it works as a collision resistant functions on which a regular signature function can be applied to provide authenticity and integrity to the message.

Chameleon Hashing is associated with a pair of public and private keys in which the private key serves as the trapdoor for the hash function. Similar to the public key cryptography, anyone who has knowledge of the public key can compute the associated hash function. It is collision resistant in that without the knowledge of the trapdoor, it is infeasible to find two inputs when hashed are mapped to the same hash value. Conversely, the holder of the trapdoor can easily find collisions for every given input, i.e., hash value. The distinct capability of Chameleon Hashing is that it allows for the owner of the trapdoor to change the input to the function without changing the output. Note that the efficiency of Chameleon Hashing construction is similar or better to that of regular digital signature.

2.2 Chameleon Signatures

Chameleon Signature [18, 17] was introduced as a much simpler implementation of undeniable signatures [11, 10]. This scheme has an advantage over the conventional zero-knowledge protocol as it is non-interactive. It is built similar to the traditional digital signature that is *hash-then-sign* approach. A regular digital signature scheme such as RSA, DSS or ECC is applied to a special type of hashing called *chameleon hash functions* [9] as described in Section 2.1.

The Chameleon Signature scheme allows a Signer, S to sign a message to be sent to a Recipient, R such that it gives R the ability to *forge* further signatures of S at will. This means that when R receives a signature of S on a message m, it can produce a signature of S on any other document m'. Consequently, when presenting the signature to a third party, it is not possible to prove the validity of the signature because R could have produced such a signature by himself. However, this scheme is useful in that it provides the signer S with the *exclusive* ability to prove that a forged signature is in fact a forgery. The Signer S has the ability to prove that R has forged the signature to a third party if he desires to do so. The signer only needs to provide a short piece of information as evidence for the forgery, i.e., the original message in which when hashed produces the equivalent hash value as claimed in the forged signature.

Although this scheme is conceptually simpler and more efficient in that it is non-interactive, it appears that its application is somewhat restricted. In this paper, we adapted the use of Chameleon Signatures to enable secure data aggregation in a typical wireless sensor network where data produce by multiple sources are aggregated by a third entity before they are forwarded to the final Recipient for analysis.

2.3 Sanitizable Signatures and Transitive Signatures

The notion of sanitizable signature [5] was introduced to allow a signer to partly delegate signing rights to a semi-trusted party called a sanitizer, so that it is allowed to change a pre-determined part of the signed document. This is particularly useful for applications such as authenticated multicast, and authenticated database outsourcing because multicast messages can be customized by a trusted sanitizer without compromising the integrity and authenticity of the messages from the source. However, this scheme has a strong assumption that the sanitizer is semi-trusted, and it is difficult to detect dishonest sanitization in this scheme.

Transitive signatures [22, 6] and aggregate signatures [7] both provide a mechanism to aggregate multiple signatures into one, thus allowing for the verification of the authenticity and integrity of the messages from its original sources. In this scheme, n signatures on n distinct messages from n distinct users can be aggregated into a single short signature. This single signature (and the n original messages) will convince the verifier that the n users did indeed sign the n original messages [7]. However, using such schemes would increase the communication overhead as each source must

compute a digital signature for each message it transmits, and then aggregated. Sensors and actuators are resource constrained, and adopting aggregate signature scheme would surely be too expensive.

2.4 (Datagram) Transport Layer Security

Transport Layer Security (TLS) [12] is commonly used in the Internet to provide end-to-end application security between two communicating end points. DTLS [25] is a datagram-compatible adaption of TLS that runs on top of UDP. Both protocols operate at the transport layer of TCP/IP stack and essentially based on a security handshake to establish a secure unicast link. The application data is then protected end-to-end using the (D)TLS record layer. The (D)TLS supports different types of authentication mechanism, e.g., using a pre-shared key, public-key certificates, and specifically DTLS supports raw public-keys authentication.

Although (D)TLS can be used to secure the communication between two end points, most applications such as ICS and patient monitoring systems require a gateway device to bridge an IP connection to a non-IP wireless interface such as IEEE 802.15.4 [2], Zigbee [4], Bluetooth in order to reach the sensors and actuators. Consequently, it is not feasible to establish a (D)TLS secure channel end-to-end through non-IP protocol. [8] attempted to enable E2E security between two devices located in homogeneous networks using either HTTP/TLS or CoAP/DTLS [26] by proposing a mapping between TLS and DTLS. However, this assumes that the end devices are IP-based, running 6LoWPAN [19].

2.5 ICS Communication Security

2.5.1 *Modbus*

Modbus [23] is the most widely deployed industrial control communications protocol. It is an application layer protocol based on request-reply paradigm. It is simple and efficient, making it to be easily deployable on Programmable Logic Controller (PLC) and Remote Terminal Unit (RTU) to communicate supervisory data to a SCADA system [16]. Unfortunately, Modbus lacks of authentication and encryption, hence vulnerable to various security attacks. It also facilitates the re-programming of controllers and could potentially be used to inject malicious logic into an RTU and PLC. In [21], a hash-chain based authentication scheme was proposed to authenticate the master in modbus. However, it suffers from de-synchronization attack and tampering attack [24].

2.5.2 *DNP3*

DNP3 [13] is a reliable and efficient protocol for use between a *master* device and an *outstation* in a control network. It supports bi-directional communications and exception-based reporting. Similar to other ICS protocols, DNP3 is used to send and receive messages between control system devices. Communication is usually initiated by the master to the slave, and in some cases unsolicited responses from the slave to the master can be sent [16]. Secure DNP3 incorporated authentication into the protocol based on challenge-response betweeen master and outstation. A symmetric session key is used to perform authentication, and critical operations must be authenticated before it can be executed. The session keys can be updated using *Update Key Protocol* either using a symmetric or assymmetric scheme. DNP3 is vulnerable to MITM attacks to capture addresses, which can be used to manipulate the system. In particular, spoofing unsolicited responses to the Master to falsify events and trick an operator into taking inappropriate actions can be done [16].

2.5.3 *Profinet*

Profibus (Process fieldbus) is a Master/Slave protocol that supports multiple masters through token sharing. Profibus over ethernet is called Profinet [16]. When a master has the token in possession, it can communicate with its slaves. Typically, a master Profibus node is a PLC or RTU and a slave is a sensor, or some control device. Profinet also suffers from the lack of authentication, allowing a spoofed node to impersonate a master node. Stuxnet exploited the vulnerabilities of Profibus by compromising PLCs (master Profibus nodes) [16].

3. INDUSTRIAL CONTROL SYSTEMS

Figure 1 illustrates an example architecture of Industrial Control Systems (ICS). Field devices equipped with wireless transmission capabilities are deployed on site to monitor the operation of an industrial (e.g., power grid, gas, petrochemical, manufacturing) plant. The data collected are aggregated by the Field Wireless Management Station, e.g., PLC or RTU, and it is important that all the data collected must be reliably reported to the Field Wireless Management Station. At the same time, field devices, e.g., sensors and PLCs can be configured dynamically based on the contextual information in the plant, thus providing automation to the control operation. This is done by the Plant Resource Manager (PRM) [1] which makes automation decisions based on the data collected from the field devices.

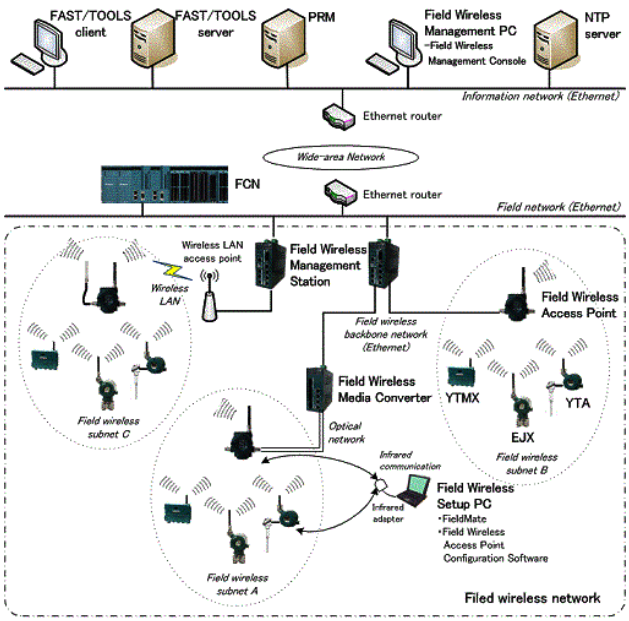

Figure 1: An example architecture of industrial control system [1]

As shown in Figure 1, the communication between the Field Wireless Management Station and PRM is via ethernet connection, which implies the use of TCP/IP protocol. End-to-end communication security can be guaranteed between them. However, as most of the field devices are not directly IP addressable, it is hence not possible to secure the sensory data from the field devices end-to-end, but to rely on the Field Wireless Management Station to protect its communication links with all the field devices. Nonetheless, application-level security can be used to protect the system end-to-end.

3.1 Threats and Security Attacks

In this paper, we do not consider physical jamming attacks and distributed denial-of-service attacks, but concentrating on the potential threats in the communication channel and data integrity.

- *Eavesdropping on communication channels* – Attackers can eavesdrop on the communication channel between the field devices and the field controllers, as well as the channel between the field controllers and the PRM in the backend to obtain sensor readings, commands and aggregated data.

- *Man-in-the-middle attack (MITM)* – In order to save bandwidth, sensor data are usually aggregated by the field controller. This implies that data received from the sensors are first decrypted, and subsequently aggregated before they are re-encrypted by the field controller to be sent to the central controller. In this case, the data protection would be split into two secure channels, one between the field device and the field controller, and another between the field controller and the central controller. Consequently, the man-in-the-middle, i.e., the field controller when compromised or misbehaved can tamper with the data, and selectively report data to the central controller as occurred in *Stuxnet*. In a more generalized form, the communication is vulnerable to MITM attacks when attackers are sitting in between any of the ICS entities.

- *Compromise of field controller* – This results in the ability of the field controllers to fully manipulate the field devices, and eventually destroy them by issuing instructions to perform out-of-norm operations, while reporting normal operational status to the central controller. In this scenario, the attacker does not need to compromise the sheer number field devices in order to destroy the whole system.

- *Compromise of both field controller and field devices* – This would render the whole system to be subverted. Since the attackers have obtained the cryptographic keys of all devices in the system, it is able to manipulate the devices, data and the communication channels between devices in the system. As a result, it makes intrusion detection system (IDS) useless.

3.2 Security Requirements

Based on the architecture of ICS, we derive three main security requirements as follows:

3.2.1 Data Integrity

The measurements performed by field devices must reflect the current state of the instruments in the plant. Therefore, any modification to the measurements can cause inappropriate decision to be made by the plant central controller. The data integrity is not only important for analysis, it is also useful inputs for intrusion detection system. When sensor readings appear to drift from the normal range, this indicates a potential fault in the devices in the field.

3.2.2 Data Origin Authentication

The data origin of the measurement is important to ensure that the measurement was taken using a designated or certified device. The sensor data collected in ICS must be traced back to its origin, in order to identify the field device that produced the measurements. This is crucial when ascertaining the status of a field device, and at the same time, such information would be useful when performing fault diagnostic to identify the faulty or misbehaving devices in the field.

3.2.3 Secure Data Aggregation

As most of the data are being aggregated, e.g., by the Field Wireless Management Station/field controller in an ICS, this shows that the original data have been transformed. Although this poses difficulty in ensuring the data integrity and data origin, a very important security requirement is to ensure that the end points, i.e., the Plant Resource Manager (PRM) or the central controller that receives the transformed or aggregated data must have the ability to check the integrity and data origin. In addition, any unauthorized data modification by the field controllers, the Field Wireless Management Station or any other intermediaries must be detected, so that any incidents of intrusion can be detected and acted upon swiftly.

4. IMPLEMENTING END-TO-END DELAYED INTEGRITY VERIFICATION

This section describes the implementation and realization of end-to-end data integrity protection for ICS based on our previous work [15]. It allows the field controller (or the Field Wireless Management Station) to securely aggregate the data collected from field devices, while enabling the central controller to verify the aggregated data, ensuring its integrity and data originality from the field devices.

Our protection scheme [15] is an adaptation of Chameleon Hashing [9] and Chameleon Signatures [18, 17] in that it allows the field controller to aggregate the sensor data, compute a Chameleon Hash Value (CHV) of the aggregated data, and then sign it using a traditional digital signature. The aggregated data together with the signature are sent to the central controller, thus establishing the authenticity of the data from the field controllers. The field devices which know the trapdoor function of the Chameleon Hash Function can *periodically* compute a different message, m_i' and a random value r_i', where m_i' consists of all the previous sensor data logged by the field device, fd_i within a time period. The combination of (m_i', r_i') and its Chameleon Hash Value, CHV_{verify} are then forwarded to the central controller for verification via any routes available in the ICS network. It serves as an evidence to prove that the recorded sensor data are truly originated from the field device itself. The central

controller is then convinced of the integrity of the sensor data and its data origin because (m'_i, r'_i) when hashed, the resulting Chameleon Hash Value is equivalent to the CHV sent by the field controller, and that the recorded sensor data matches the corresponding values sent by the field devices. This protocol and the salient property of Chameleon Hashing prove that the field controllers have not been compromised.

4.1 Commissioning and Key Generation

Prior to the deployment of the end-to-end integrity verification scheme, field devices and the corresponding field controller must be commissioned with the relevant cryptographic keying materials. When the operating plant is first deployed with sensors in the field, the sensors are grouped together and then bound to a field controller. This can be done through various device discovery protocol to enable the field controller to discover the sensors in its proximity.

4.1.1 Generation of Chameleon Hash Function Keys

Our scheme uses the Krawczyk and Rabin's discrete logarithm construction [17] to generate Chameleon Hashing keys. Two prime numbers p and q are randomly generated such that $p = kq + 1$ where q is a large enough prime factor. An element g of order q in $\mathbb{Z}_p^*$ is chosen so that the private key, K' is $x \in \mathbb{Z}_p^*$. The public key K is generated as:

$$y = g^x \mod p \tag{1}$$

As shown in Figure 3, all sensors in each group are commissioned with a *Trapdoor Chameleon Hash Function*, where they share a key, K' among themselves, while the responsible field controller is configured with the corresponding *Chameleon Hash Function* which is public (i.e., knowing K), and this is also known to the central controller.

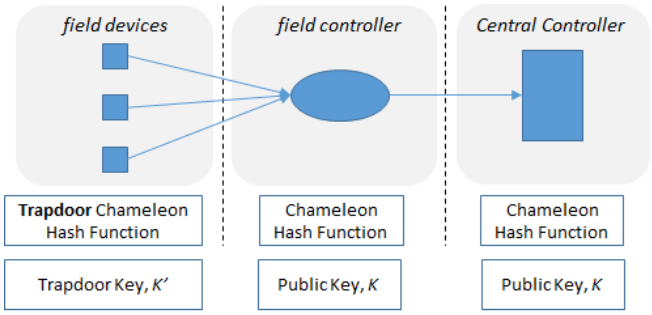

Figure 3: Commissioning of cryptographic keying materials

In order to generate a Chameleon Hash, given a message $m \in \mathbb{Z}_p^*$, choose a random value $r \in \mathbb{Z}_p^*$, the Chameleon Hash denoted as CHV can be computed as:

$$CHA\_HASH(m, r) = g^m y^r \mod p \tag{2}$$

Such a configuration allows anyone to compute a Chameleon Hash given a known message, while only the field devices have the ability to produce the same chameleon hash value using a different message, m' such that CHA_HASH(m, r) = CHA_HASH(m', r'). This is done by solving r' in the following equation using K', i.e., x:

$$m + xr = m' + xr' \mod p \tag{3}$$

Apart from the field devices in the group, no other entity in the system should know the *Trapdoor Chameleon Hash Function* and K' (i.e., x).

4.2 Operation

Figure 2 shows the cryptographic operations to be performed by the field controller and the field devices to ensure the integrity and authenticity of the data. The details of each operation is described in the following sections.

4.2.1 Transmission of Sensor Data

For a group of field devices fd_1, fd_2, fd_3,..., fd_n where n is the number of sensors bound to a field controller fc, each field device periodically reports sensor reading to the field controller. The reporting of sensor reading can be done through a standard secure communication channel, e.g., using DTLS between the field device and the field controller. For example, fd_i sends m_{ij} to the field controller, where i is the device identifier, and j is the message number. This message can be protected using DTLS Record Layer, thus providing message freshness guarantee and protection against message replay.

Operation: **Transmit Data**
$fd_i \rightarrow fc$: $\{fd_i, m_{ij}\}$

4.2.2 Data Aggregation

As the field controller, fc receives sensor data from multiple field devices it manages, it is responsible for aggregating the data received before forwarding them to the central controller. The field controller collects n messages from the field devices, i.e., m_{1j}, m_{2j},..., m_{nj} where n is the number of field devices, and j is the message number.

Operation: **Aggregate Data**
$AggData_j$: $\{m_{1j}, m_{2j}, ..., m_{nj}\}$
where j is the aggregated message number and $j > 0$

For instance, when the field controller first receives data from the field devices, it formulates $AggData_1$ to be sent to the central controller. The aggregated data is hashed using the field controller's Chameleon Hash Function. Once the hash is computed, a standard digital signature scheme can be used by the field controller to sign the aggregated data before it is sent to the central controller.

Operation: **Hash Aggregated Data**
Let $m = AggData_j$ and a random number r_j is generated. The Chameleon Hash Value, $CHV_{fc,j}$ is computed using CHA_HASH(m, r_j) as follows:

$$CHV_{fc,j} = g^{AggData_j} y^{r_j} \mod p \tag{4}$$

The $CHV_{fc,j}$ is then signed using a digital signature scheme, e.g., ECDSA.

Operation: **Sign** $CHV_{fc,j}$
$SEC\_MSG_{fc,j}$ = SIGN$(Priv_{fc}, CHV_{fc,j})$

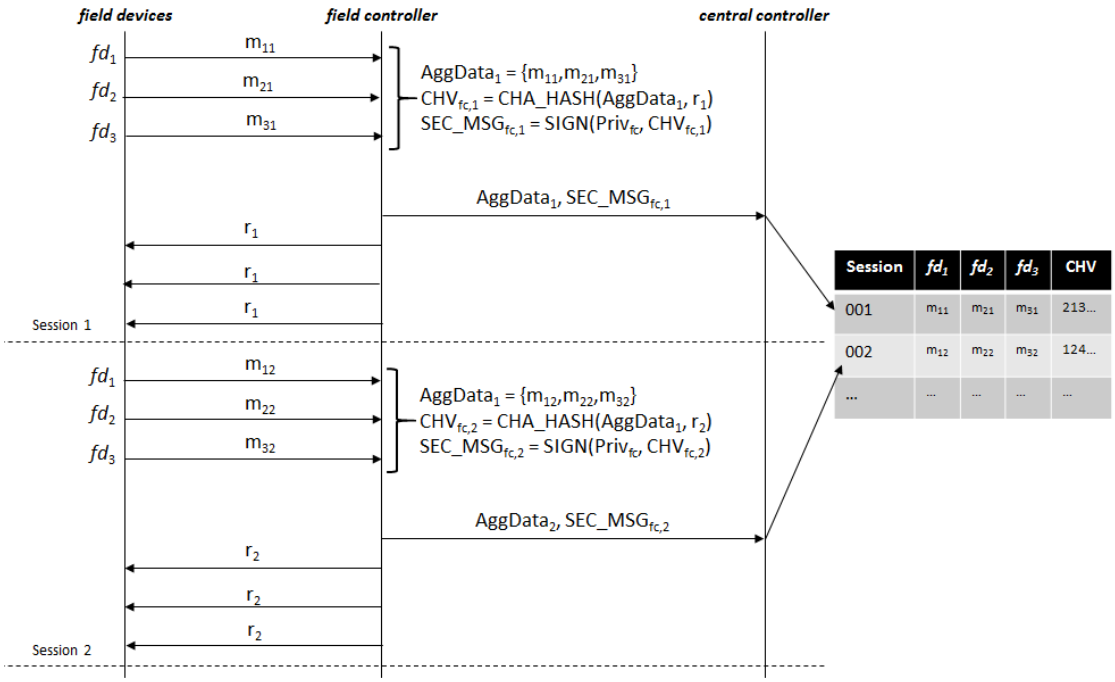

Figure 2: Protocol for reporting field device readings periodically using Chameleon Hashing

Finally, the aggregated data, and the signature are sent to the central controller for verification. At the same time, the resulting random number r_j for generating the CHV is sent to all the field devices managed by the field controller as a means to acknowledge receipt of the sensor data transmitted by the field device.

Operation: **Transmit to Utility and Acknowledge** fd
$fc \rightarrow$ *central controller*: $SEC\_MSG_{fc,j}$, $\{AggData_j\}$, r_j
$fc \rightarrow fd_{1,2,...,n}$: r_j

4.2.3 Verification by the Central Controller

Upon receipt of the aggregated data from the field controller, the central controller verifies the digital signature using the its public-key, Pub_{fc}. During the signature verification process, the central controller uses the field controller's Chameleon Hash Function to compute the $CHV_{fc,j}$ as the Chameleon Hash Public-Key, K (i.e., y) is also known to the central controller. If the verification of digital signature is successful, the central controller accepts the integrity and authenticity of the data received, and stores the $CHV_{fc,j}$ and the data for *end-to-end* verification later on.

Operation: **Verify Signature**
VERIFY(Pub_{fc}, $AggData_j$, $SEC\_MSG_{fc,j}$)

4.3 Periodic End-to-End Integrity Verification

Since the signature is generated by the field controller, the central controller can only believe that the data are originated from the field controllers and have not been tampered with during transmission. However, the central controller does not have the guarantee that the field controller had not maliciously tampered with the aggregated data itself. Therefore, our scheme relies on the field devices to corroborate the integrity and authenticity of the aggregated data by transmitting an evidence, known as *the commitment* (described in the next section) to the central controller for end-to-end integrity verification.

4.3.1 Transmission of Evidence

As the field devices continuously send sensor readings to the field controller, our scheme divides time into intervals where each interval allows the field device to send up to t messages to the field controller. At the end of each interval, each field device uses any of the received r_v from the field controller, fc where $1 \leq v \leq t$ and the corresponding *Trapdoor Chameleon Hash Function* to compute a value, r'_i such that the concatenation of its device identifier with all the respective data it had sent for that period, $\{Id_{fd,i}, m_{i,1}, m_{i,2},..., m_{i,t}\}$ when hashed together with r'_i is equivalent $CHV_{fc,v}$.

In essence, each field controller, fd_i computes the following: Let m'_i denotes $\{Id_{fd,i}, m_{i,1}, m_{i,2}, ..., m_{i,t}\}$ recorded by the field device, fd_i. The field device, fd_i computes CHA_HASH(m'_i, r'_i) such that its Chameleon Hash Value, CHV_{verify} is equivalent to $CHV_{fc,v}$. As mentioned previously, $CHV_{fc,v}$ is computed based on CHA_HASH ($AggData_v$, r_v). This means that we need to solve r'_i, where i is the field device identifier, in order to generate a collision using Equation 3. In order to compute r'_i, we must solve the following:

$$r'_i \mod p = (AggData_v + xr_v - m'_i)x^{-1} \mod p \quad (5)$$

Although the field device has the trapdoor key K', essentially x, it does not have a copy of $AggData_v$, i.e., the aggregated data provided by the field controller. Hence, it is not able to derive r'_i. Instead, it can provide a *commitment* that will allow the central controller to verify the integrity of the data reported by the field device and field controller.

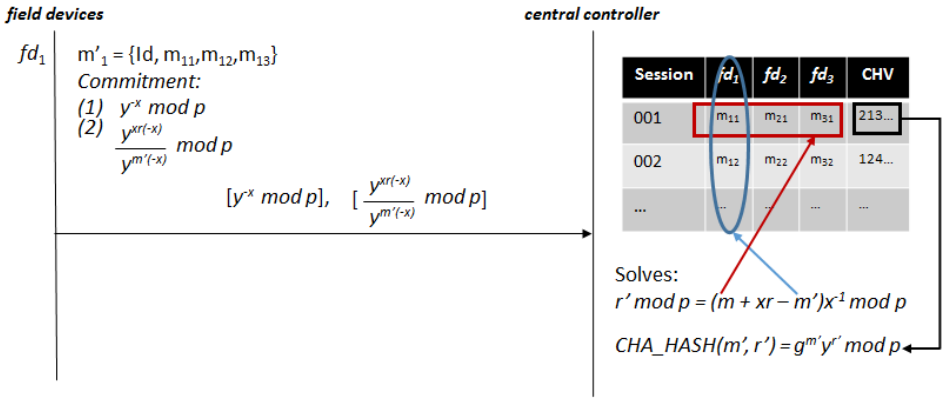

Figure 4: End-to-End *Delayed-Integrity-Verification*

Operation: **Field device, fd_i computes a commitment**

$$[y^{-x} \mod p], [\frac{y^{xr_v(-x)}}{y^{m'_i(-x)}} \mod p] \quad (6)$$

The *commitment* as shown above is sent to the central controller to perform end-to-end integrity verification.

Since only the field devices know the *Trapdoor Key*, K' or x, no other entity can produce a different (m'_i, r'_i) when hashed, is equivalent to $CHV_{fc,v}$. The field device's unique device key, $Id_{fd,i}$ is also concatenated with the sensor data for identification purpose. It ensures that no one can impersonate the field device. In this scheme, the field device is only required to perform the *Trapdoor Chameleon Hash* operation to produce a *commitment*, without needing to generate any signatures in order to sign the sensor data recorded in the past time period. Even though the *commitment* could be routed via the field controller, if it was dropped, the central controller would be able to detect this easily and suspect misbehaviour of the field controller.

4.3.2 *Integrity Verification*

As shown in Figure 4, the central controller stores all the data received from the field controllers, including the aggregated data ($AggData$), as well as all the Chameleon Hash Values ($CHVs$) and their corresponding r values.

When the central controller receives the *commitment* from each field device, it can verify the integrity and authenticity of the message by ensuring that CHA_HASH(m_v, r_v) = CHA_HASH(m'_i, r'_i). Given Equation 5, and the *commitment* received, as well as the sensor data stored, the central controller can compute $y^{r'_i}$ as follows:

$$y^{r'_i} \mod p = (y^{-x(m_v)}) \times (\frac{y^{xr_v(-x)}}{y^{m'_i(-x)}}) \mod p \quad (7)$$

Once $y^{r'_i}$ is computed, the central controller can perform a final check on the derived CHV_{verify} by computing:

$$CHV_{verify} = g^{m'_i} y^{r'_i} \mod p \quad (8)$$

The derived CHV_{verify} must be equivalent to $CHV_{fc,v}$ as indicated in Figure 4. This ensures that all the reported readings, whether they are from the field device or the field controller are consistent with each other, and that they have not been tampered with. If there's an anomaly in terms of message values, the field controller is suspected to be faulty or compromised. Consequently, attacks on the field controllers can be detected swiftly through this scheme.

The integrity verification can take place as frequently as required depending on the severity of the application. The verification frequency can be dynamically configured to suit the needs of ICS applications. A close to real-time integrity verification would require a higher number of message transmissions and an increase in the computation costs.

5. IMPLEMENTATION AND EVALUATION

This section describes the prototype implementation in order to validate the feasibility of the protocol to provide *delayed-integrity-verification*.

5.1 Prototype Implementation

The protocol was implemented using Java in order to simulate periodic reporting of sensor readings in an ICS network. We used a Raspberry Pi Model B+ to act as a field device, while a computer is used to simulate the field controller. The Raspberry Pi has a modest computational capability, with a CPU of 700 MHz Low Power ARM1176JZFS Applications Processor, and 512 MB of SDRAM.

We relied on Java Cryptography Architecture (JCA) to implement our security protocol. The mathematical opreations performed by the field controller include the generation of Chameleon Hashing (Eq. 4) and digital signature on the CHV. As for the field device, it has to generate a commitment (Eq. 6) to be sent to the central controller for verification. Finally, the central controller has to verify (1) the digital signature generated by the field controller, (2) the commitments and the CHVs whether the sensor data are integrity-preserved as shown in Eq. 7 and 8.

5.2 Preliminary Results

As shown in Table 1, using the proposed *delayed-integrity-verification* scheme, the field device fd is required to generate a commitment to facilitate the integrity verification. If we compare this with the time required to generate a digital signature on fd in order to guarantee data integrity, we

observed that our scheme incurred less computation and is more efficient. This is because the time taken to generate a digital signature on fd took an average of 6000 ms, while the generation of commitment only took 111 ms.

Device	Operation	Time (ms)
Controller (fc)	Generate CHV	0.955955
Device (fd)	Generate Commitment	111.6
Controller (cc)	Verify Data Integrity	2.288591
Device (fd)	Generate Signature	5830 ms

Table 1: Performance measurements

As the field controller, fc and the central controller, cc are assumed to be more computationally capable than the field devices fd, we deployed them on a PC or laptop. Consequently, the time taken to generate a Chameleon Hash and to verify data integrity on fc and cc respectively were a lot faster than those cryptographic operations performed on the fd. Specifically, the time taken to generate a Chameleon Hash on fc took approximately 1 ms, while the data integrity verification on cc took roughly 2.3 ms.

6. SECURITY ANALYSIS AND DISCUSSION

This section provides an informal security analysis on the security protocol presented in this paper.

6.1 End-to-End Integrity

The use of Chameleon Commitment Scheme allows the entity which possesses the *Trapdoor Chameleon Hash Function* to produce a different message that yields the same Chameleon Hash Value enables the field devices to "replace" the message content signed by the field controller with only the messages originated by themselves. As a result, the central controller can easily verify the message content of the field devices by just computing the Chameleon Hashing, since the signature verification had been done while validating the messages sent by the field controllers.

Assuming that the field devices are trusted and have not been compromised, the central controller must accept the data (m_i', r_i') produced by the field device, as they are the only entities in the system which are configured with the *Trapdoor Chameleon Hash Function*. The field controller on the other hand does not have the capability to do so, all previous messages and the corresponding Chameleon Hash Values (CHVs) it generated cannot be later modified by itself, and hence serve as a commitment to the security of the system.

6.2 Detection of Data Inconsistency

There could be discrepancies when the central controller compares messages received from the field devices, with the aggregated data signed by the field controller. This signifies that there is a potential device compromise or fault in either the field controller or the field device. The central controller can demand all field devices to send (m_i', r_i') periodically, so that the central controller can cross check the sensor readings and data received from the field controllers. If the central controller did not receive any information from the field devices for an extended period of time, it could well be that the field devices have been compromised. The discrepancy in the data can serve as a pre-cursor to a system intrusion, and would be a significant event to trigger a full intrusion detection to identify compromise devices.

6.3 Mitigating Field Controller Compromise

In case that the field controller has been compromised, it is able to take over control of all the field devices that are bound to it. The attacker will have access to (1) the encrypted channel between the field devices and itself, (2) data reported by field devices, (3) private-key of the field controller to sign messages to the central controller, (4) commands and invocations to be issued to the field devices, and (5) communication between the field devices and the central controller.

While issuing malicious commands to the field devices, the attacker can continue to report "normal" readings to the central controller although the real data reported the field devices show otherwise. Prior to signing the aggregated data, the attacker can modify the data to show that operation in the plant are normal so that the central controller is not able to detect any abnormal operation.

With our scheme, the field devices have to periodically send their respective (aggregated) sensor readings to the central controller, so that the controller can perform a cross-check on the data sent by the field devices and the one sent by the field controller. As only the field devices have the Trapdoor Chameleon Hash Function, no one else is able to compute the *commitment* which allows the central controller to verify the resulting CHV_{verify} that must be equivalent to the one sent earlier by the field controller. The central controller can detect device compromise if there is a discrepancy between the two data sets.

6.4 Mitigating Field Device Compromise

In case that a field device has been compromised, the attacker will have access to the (1) Trapdoor Chameleon Hash Function which is also known to other field devices under the administration of the same field controller, (2) the encrypted communication channel between the compromised field device and the field controller, the communication channel of other field devices are not compromised.

The attacker can take down the compromised sensor, while continuously send "normal" readings to the field controller. Although the Trapdoor Chameleon Hash Function is the same across all field devices within a group, this does not mean that by compromising one field device, the whole group will be compromised. This is because each field device has its own unique secure communication channel with the field controller, and all messages are encrypted in the channel. As the attacker does not have access to these communication channels, it does not know the content of the messages (m_i') and therefore even if it knows the Trapdoor Chameleon Hash Function, it is not able to derive the required *commitment*, i.e., $[(\frac{y^{xr_v(-x)}}{y^{m_i'(-x)}}) \mod p]$ for the respective field device. Consequently, the attacker would need to compromise each individual field device if it aims to take down the whole system.

7. CONCLUSIONS AND FUTURE WORK

We have provided a novel use of Chameleon Signatures other than its traditional usage, to detect misbehavior of field controllers in ICS. Using the proposed end-to-end *integrity verification* scheme, we can ensure that the data aggregated by the field controller is protected end-to-end in that the integrity, authenticity and data originality of the sensor data can be guaranteed. This would be beneficial to the protection of critical infrastructures. The advantage of this scheme is that the digital signatures generated by the field controllers can be used to verify both the data reported by the field devices and the field controllers themselves. The field devices are not required to use any public-key based signature scheme, but merely executing a (Chameleon) hash operation.

We have also provided a security analysis to first show the feasibility and robustness of the proposed scheme. The natural next step is to simulate the protocol in an ICS test bed to assess the performance of the protocol in a larger scale.

Another future direction is to integrate the protocol with an intrusion detection system. Machine learning techniques can be used to detect any anomaly in the sensor readings.

8. REFERENCES

[1] Industrial Control Systems, http://www.yokogawa.com/.

[2] IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, 2006.

[3] Kaspersky Lab provides its insights on Stuxnet worm, Sept 2010.

[4] Z. Alliance. Zigbee Features Set and ZigBee PRO Specifications, 2007.

[5] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik. Sanitizable signatures. In *Proceedings of the 10th European Conference on Research in Computer Security*, ESORICS'05, pages 159–177, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] M. Bellare and G. Neven. Transitive signatures: new schemes and proofs. *Information Theory, IEEE Transactions on*, 51(6):2133–2151, 2005.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'03, pages 416–432, Berlin, Heidelberg, 2003. Springer-Verlag.

[8] M. Brachmann, S. L. Keoh, O. Morchon, and S. Kumar. End-to-end transport security in the ip-based internet of things. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–5, 2012.

[9] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, Oct. 1988.

[10] D. Chaum. Zero-knowledge undeniable signatures. In *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 458–464. Springer, 1990.

[11] D. Chaum and H. Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology (CRYPTO'89) Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer New York, 1990.

[12] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.

[13] DNP. A DNP Protocol Primer, 2005.

[14] C. Johnson. Cybersafety: on the interactions between cybersecurity and the software engineering of safety-critical systems. In *Twentieth Safety-Critical Systems Symposium, Bristol, UK*. Springer Verlag, February 2012.

[15] S. L. Keoh and Z. Tang. Towards secure end-to-end data aggregation in ami through delayed integrity verification. In *10th International Conference on Information Assurance and Security (IAS), Okinawa, Japan*, Nov 2014.

[16] E. Knapp. Chapter 4 - industrial network protocols. In E. Knapp, editor, *Industrial Network Security*, pages 55 – 87. Syngress, Boston, 2011.

[17] H. Krawczyk and T. Rabin. Chameleon hashing and signatures, 1997.

[18] H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS*. The Internet Society, 2000.

[19] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919 (Informational), Aug. 2007.

[20] R. Langner. To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve. Technical report, The Langner Group, Nov 2013.

[21] G.-Y. Liao, Y.-J. Chen, W.-C. Lu, and T.-C. Cheng. Toward authenticating the master in the modbus protocol. *Power Delivery, IEEE Transactions on*, 23(4):2628–2629, Oct 2008.

[22] S. Micali and R. Rivest. Transitive signature schemes. In B. Preneel, editor, *Topics in Cryptology âĂŤ CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 236–243. Springer Berlin Heidelberg, 2002.

[23] Modbus. Modbus Application Protocol Specification V1.1b3, April 2012.

[24] R.-W. Phan. Authenticated modbus protocol for critical infrastructure protection. *Power Delivery, IEEE Transactions on*, 27(3):1687–1689, July 2012.

[25] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), Apr. 2006. Obsoleted by RFC 6347, updated by RFC 5746.

[26] Z. Shelby, K. Hartke, and C. Bormann. Constrained Application Protocol (CoAP). Internet Draft draft-ietf-core-coap-18, IETF, 2013.