SEABASS: Symmetric-keychain Encryption and Authentication for Building Automation Systems

Joshua Ng and Sye Loong Keoh
School of Computing Science
University of Glasgow
{2228124N, SyeLoong.Keoh}@glasgow.ac.uk

Zhaohui Tang
ICT Cluster
Singapore Institute of Technology
Zhaohui.Tang@SingaporeTech.edu.sg

Hajoon Ko
Department of Computer Science
Harvard University
hrko@g.harvard.edu

***Abstract*—There is an increasing security risk in Building Automation Systems (BAS) in that its communication is unprotected, resulting in the adversary having the capability to inject spurious commands to the actuators to alter the behaviour of BAS. The communication between the Human-Machine-Interface (HMI) and the controller (PLC) is vulnerable as there is no secret key being used to protect the authenticity, confidentiality and integrity of the sensor data and commands. We propose $SEABASS$, a lightweight key management scheme to distribute and manage session keys between HMI and PLCs, providing a secure communication channel between any two communicating devices in BAS through a symmetric-key based hash-chain encryption and authentication of message exchange. Our scheme facilitates automatic renewal of session keys periodically based on the use of a reversed hash-chain. A prototype was implemented using the BACnet/IP communication protocol and the preliminary results show that the symmetric keychain approach is lightweight and incurs low latency.**

I. Introduction

A Building Automation System (BAS) is a computerized, intelligent system that controls and monitors Heating, Ventilation, Air conditioning (HVAC), lighting devices, fire alarm systems, CCTV, access control systems, etc in a building. It is gaining popular as it has the capability to reduce maintenance cost and energy consumption, as well as to increase controllability, reliability, and usability for maintenance staff and tenants of the building. BAS can also be deployed in industrial infrastructures such as malls, enterprise buildings and factories [6] [16]. Typically, such a system performs supervisory control and monitoring of field devices such as sensors, actuators and controllers (Programmable logic controller (PLC), Remote terminal unit (RTU) and Intelligent Electronic Device (IED)), and this is realized through a Supervisory Control and Data Acquisition (SCADA) system. In the SCADA system, there is a Human-Machine-Interface (HMI) that can be used to control and visualize the field devices deployed in BAS. Therefore, the maintenance staff is able to quickly detect problems and subsequently perform the necessary adjustments.

Historically SCADA systems in industrial infrastructures have been used for monitoring and controlling critical infrastructures and manufacturing process operated in isolated environment. This forms an air gap, isolating the SCADA network from the outside world, so that the data and control commands remain secure inside the SCADA network. However, with the advent of Internet, there are increasing number of SCADA systems that are now connected to the outside world to allow for better decision making for enterprises by providing them with real-time remote updates at any time and any place. As a result, there is a significant security risks and threats to the SCADA networks that do not have sufficient security protection by default.

First of all, the communication between the field devices (i.e., sensors, PLCs, and Remote Terminal Units (RTU)) and the Human-Machine-Interface (HMI) in SCADA network is often unprotected. By using network sniffing tools, real time data can be eavesdropped and observed by adversary in order to understand the system behaviour. Active attacks could also be launched by injecting spurious messages into the network, altering the sensor data and commands issued to the actuators. In addition, it is susceptible to Man-In-the-Middle (MITM) and DoS attacks once the adversary gains access to the network. As a result, the compromise of SCADA network has a wider consequence that leads to losses of monies, lives, production, assets and reputation of the company.

In this paper, we propose $SEABASS$, a symmetric-key based key management scheme to distribute a session key between devices in SCADA network, i.e., HMI-PLC, PLC-PLC, or even PLC-Sensor, and to allow automatic renewal of the session key periodically based on the use of a reversed hash-chain and the authentication mechanism defined in Viot-SOC [12]. Any devices in the network can be paired to obtain a session key from the key management server, one acting as the client, while the other as the server; We assume that the devices use BACNet/IP as their communication medium in this work. A security protocol is defined between the three parties, namely the key management server, client and server to distribute and renew the session key.

This paper is organised as follows: Section II discusses the background of building automation system, the communication protocols used and related work. Section III presents the threat model, security requirements and outline the proposed key management scheme using lightweight hash-chain. Section IV describes the prototype implementation based on an environmental monitoring use case in a building, while Section V discusses the performance results and analysis. We conclude the paper with future work in Section VI.

II. Background and Related Work

A. SCADA System for BAS

A SCADA network consists of controller devices such as RTUs, PLCs and IEDs interconnected with sensing equipments and switch boxes or valve actuators spread in the process field. The data from sensing equipments is collected by these field devices and then sent to the MTU located at control network. The communication link between controller devices and the control network is typically by fibre cable, GPRS or other wireless technologies. The SCADA server or data server assembles the data, transfers alarms and events to the Human Machine Interface (HMI) and archives the collected data in a large database, called historian. The historian logs and archives time-based process data to allow for auditing, performance monitoring and trend analysis. The HMI provides an interface for the operating engineer to get a system overview [8], [13].

Increasingly, the top layer corporate network provides remote capabilities to allow for users to examine the state of the SCADA system. Consequently, this connection has become one of the main security concerns due to the exposure of this isolated SCADA network to the Internet. With a wrong configuration, anyone connected to the Internet can potentially connect to the control network and then compromising it [8].

B. Communication Protocols

1) ModBus: Modicon Communication Bus (Modbus) control protocol is one of the oldest and widely used communication protocols in BAS due to its simplicity and reliability [15]. Modbus was designed to allow for communication of industrial equipments such as computers, sensors, PLCs and other physical input/output devices over the IP/Ethernet network. It is the standard for industrial SCADA systems and uses Master/Slave protocol for data communication whereby a Master Terminal Unit (MTU) polls data as master from several slave devices like PLCs and IEDs, while supports data as slave to other master devices like ICS server and HMIs [11] [18]. Modbus provides mechanisms for both unicast and multicast transmissions between multiple MTUs and multiple RTUs. It is also able to support numerous Building Automation (BA) controllers, especially for HVAC controller-to-controller-communication (e.g., with chillers). However, there is no message verification checks that originated from legitimate devices (lack of authentication). This means that any compromised devices in Modbus network is able to manipulate and control slave devices to perform malicious actions [11]. Furthermore, anyone connected to the network can eavesdrop on the data transmitted from local slaves.

2) BACnet: BACnet is a standard data communication protocol for Building Automation and Control Networks which enables maximized interoperability across many products, building systems, and vendors in commercial buildings [3]. BACnet was developed by ASHRAE and released in 1995 as ASHRAE standard 135 and ISO standard in 2003 [5], [14]–[16]. It also supports IP network through BACNet/IP in which each BACnet/IP device understands how to use IP directly and constructs its own UDP message and sends it via IP addressing to the desired destination device. BACnet/IP uses UDP as its communication means and it heavily relies on IP broadcasting messages to locate other BACnet devices from separate subnets. BACnet suffers from *spoofing*, *device discovery attack* and *write-property attack* (c.f. Section III-A)

3) DNP3: Distributed Network Protocol (DNP) was first introduced in 1998 [10] as a serial protocol which is similar to Modbus [11]. It was initially designed for the communication between substation and their control station in an interconnected SCADA network [17]. It uses a standard asynchronous serial telecontrol channel (IEC 60870-5-101) and TCP/IP network (IEC 60870-5-104 extension of IEC 60870-5-101). DNP3 becomes standardized as IEEE 1815-2010 in 2010 and is still widely used in various SCADA domains such as power grids, waterways, and railways in North America and Asia [4]. DNP3 is widely used to interconnect RTU (control station) to IED (substation) in electric substations [11]. Though Secure DNP3 can be used to provide data integrity and authenticity, it can be compromised fairly easily due to the small number of well-defined function codes used in DNP messages.

4) LonWorks: LonWorks was developed by Echelon Corporation in 1991 and is an open standard in control networks [7] [19]. Each LonWorks device consists of a transceiver Chip called Neuron chip that is embedded with a firmware called LonTalk protocol, i.e., the communication protocol. LonTalk protocol implements all seven layers of the Open System Interconnection (OSI) standard Model to serve as an open system platform. LonWorks communicates via standard network variable types (SNVT) to facilitate interoperability by providing a well-defined interface. SNVT is similar to BACnet's context-driven based data types [2]. However, BACnet provides a greater provision of high-end functions, such as scheduling, alarm management, trending, etc whereas LonWorks seems to be focusing on interoperability at device level. [20] reported that the transmission of messages in LonTalk protocol is not encrypted and the identity of the sender cannot be verified when broadcasting messages.

III. Overview of $SEABASS$

This section first outlines the threat model in Building Automation Systems (BAS). Subsequently, we provide an overview of the proposed symmetric-key based hash-chain key management system to secure the communication in BAS.

A. Threat Model

- **Message Spoofing and MITM Attack** – As there is no message verification checks whether a message originates from a legitimate device in Modbus, an adversary can easily spoof control messages in order to trigger abnormal actions [11]. While in BACnet, an attack similar to ARP spoofing can be launched where a compromised device generates BACnet's "*I-Am-Router-To-Network*" messages with the fake content and forces other devices to send their messages via the attacker host [1]. With this, MITM

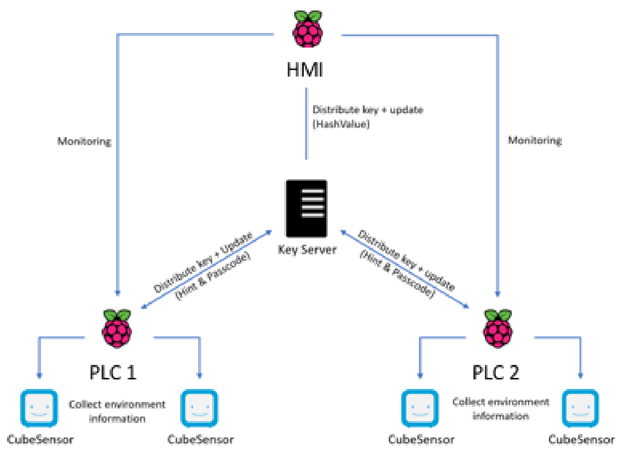

HMI
Distribute key + update (HashValue)
Monitoring
Monitoring
Key Server
Distribute key + Update (Hint & Passcode)
Distribute key + update (Hint & Passcode)
PLC 1
PLC 2
Collect environment information
Collect environment information
CubeSensor
CubeSensor
CubeSensor
CubeSensor

Fig. 1: Proposed Key Distribution Scheme for BAS

attack be easily launched to manipulate the data exchanged in the network.

- **WriteProperty Attack** – Property of field devices represents the current state of the environment. In BACnet, an adversary can manipulate the properties of the device by performing a *WriteProperty* attack, causing the device to switch on/off. Through changing the values in BACnet's object properties, the control and communications in the network can be easily disrupted [9].
- **Interception and Traffic Redirection** – An adversary can spoof "*I-Am-Router-To-Network*" or "*Router-Available-to-Network*" messages, thus tricking the other field devices into redirecting selective traffic messages to itself. Consequently, the adversary will be able to gain access to the traffic data and eavesdrops on the confidential monitoring data [9].

B. Security Requirements

In this paper, we specifically address three security requirements in BAS as described below:

- *Efficient Key Distribution* – In order to enable device and message encryption and authentication, BAS requires that all devices in the system be provisioned with a symmetric key in an efficient manner. As devices in BAS have scarce computation resources, certificate based and public-key approaches are not desirable.
- *Automated and regular Key Renewal* – One of the issues in existing BACnet protocol is that the key server handles too many keys to be distributed to field devices and these keys are required to be periodically updated by communicating with the key server. This implies that the key server has to be available at all times to facilitate key updates. An automated key management scheme is required, and with a very frequent key renewal rate, key revocation is not necessary.
- *Lightweight Communication Security* – There is a need for proper message authentication and integrity protection. A symmetric-key based scheme should be used, as public-key cryptography incurs high overheads if deployed these constrained devices in BAS.

C. Overview of Key Management Scheme for BAS

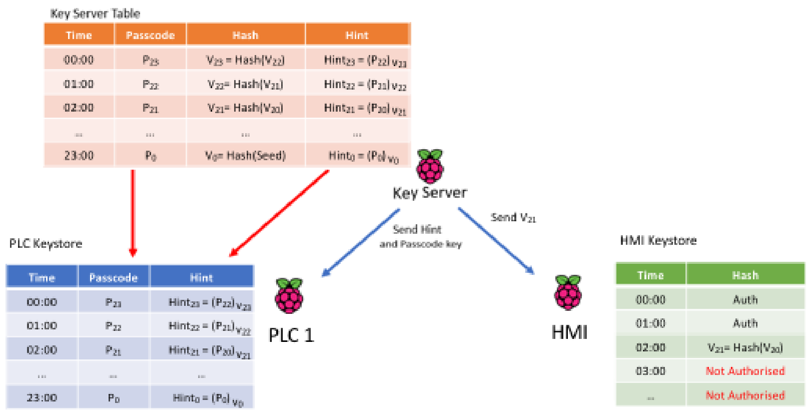
The main objective of the proposed key management scheme is to efficiently provision a secret communication key between two communicating devices (i.e., HMI-PLC and PLC-PLC) in BAS. Figure 1 shows that the key server is responsible for distributing security keys and security parameters to PLCs and HMI so that each PLC can establish a secure communication channel with the HMI. The communication between all devices in BAS is assumed to be BACnet/IP, as it allows them to discover other BACnet devices across different IP subnets and forms a private communication between them.

The PLC obtains environmental readings through the sensors and it can also trigger actions on actuators thus forming a feedback-control loop. The environmental state of the building can be visualized on HMI, thus enabling further optimization and adaptation of the building's energy consumption and operations. The proposed system is a generalized key management scheme that is able to distribute and manage session keys to enable HMI-PLC, PLC-sensor or PLC-PLC communication in a secure manner through encryption and authentication.

1) Key Generation and Distribution: As shown in Figure 2, the Key Server generates three security parameters, namely *Passcode*, *Hash Key* and *Hint*. The *Passcode* serves as a secret communication key to be used between the two communicating devices. The *Hash Key* is formed using a hash-chain by repeatedly applying a standard hash function such as SHA256 on a seed. This *Hash Key* is mapped to time and each time slot is assigned a *Hash Key* and a *Passcode*. The passcode for each time slot is encrypted with its corresponding *Hash Key*, thus forming the *Hint*. The Hash Keys comprising a hash chain are used in reversed order to provide fine-granular time-based access to the sensor data.

- *Passcode* – Key Server randomly generates j numbers of 128-bit keys, P_0 to P_{j-1}. j is a configurable parameter for the number of time slots in a day.
- *Hash Key* – A hash-chain is generated using the relation of $V_i = H(V_{i-1})$ where V_i is a *Hash Key* valid for time slot i and H is a hash function, i.e., SHA-256. V_0 is computed by using a random seed value. For example, the relation is based on $V_1 = H(\text{Seed}) \rightarrow V_2 = H(V_1) \rightarrow V_3 = H(V_2) \ldots V_i = H(V_{i-1})$.
- *Hint* – Protects the *Passcode* by computing $\text{Hint}_k = E(P_k)_{V_k}$, where Hint_k is the hint valid for time slot k, E is encryption of the Passcode, and V is a *Hash Key*. For example, the first Hint for time slot 23:00 – 00:00 can be formulated as $\text{Hint}_0 = E(P_0)_{V_0}$, and so on.

Similar to a client-server approach, the PLC which has the sensor data, is denoted as the server, while the HMI which accesses the sensor data from the PLC, is denoted as the client. The Key Server distributes the *Hint* and the *Passcode* to the PLC, while it sends a designated *Hash Key* to the HMI. As shown in the example in Figure 2, the *Hash Key* V_{21} is sent to HMI to allow for HMI to access the sensor data from



Key Server Table

Time	Passcode	Hash	Hint
00:00	P_{23}	V_{23} = Hash(V_{22})	$Hint_{23} = (P_{22})_{V_{23}}$
01:00	P_{22}	V_{22}= Hash(V_{21})	$Hint_{22} = (P_{21})_{V_{22}}$
02:00	P_{21}	V_{21}= Hash(V_{20})	$Hint_{21} = (P_{20})_{V_{21}}$
...	...	...	...
23:00	P_0	V_0= Hash(Seed)	$Hint_0 = (P_0)_{V_0}$

PLC Keystore

Time	Passcode	Hint
00:00	P_{23}	$Hint_{23} = (P_{22})_{V_{23}}$
01:00	P_{22}	$Hint_{22} = (P_{21})_{V_{22}}$
02:00	P_{21}	$Hint_{21} = (P_{20})_{V_{21}}$
...	...	...
23:00	P_0	$Hint_0 = (P_0)_{V_0}$

HMI Keystore

Time	Hash
00:00	Auth
01:00	Auth
02:00	V_{21}= Hash(V_{20})
03:00	Not Authorised
...	Not Authorised

Fig. 2: Key Distribution Process

time slot 00:00 – 02:00. It will not be able to access sensor data afterwards as it does not have the *Hash Key* to decrypt the *Passcodes* for the subsequent time slots. The length of each time slot is configurable and is determined by the key distribution server.

2) Authentication & Key Agreement: After the keys and security parameters have been distributed, the two communicating parties, i.e., the PLC and HMI must perform a key agreement protocol in order to establish a secure communication channel between them. For each time slot, a *Hash Key*, *V* has been allocated to the two communicating parties. The HMI can send a request to the PLC that it wishes to communicate with, in order to obtain the *Hint*. As HMI possesses the *Hash Key* for that particular time slot, it can decrypt the *Hint* to obtain the *Passcode* for the time slot. Authentication and key agreement between the PLC and HMI are successful when the *Hint* can be decrypted successfully.

As the *Hint* is encrypted with the *Hash Key*, and each time slot has a different *Hint*, if a HMI was not provisioned with the correct *Hash Key*, it will not be able to decrypt the *Hint* and subsequently recover the *Passcode*. Consequently, the sensor data recorded by the PLC will not be accessible to the HMI. The *Passcode* is automatically renewed when the time slot has expired. The HMI will need to request for a new *Hint* from the PLC in order to obtain the new *Passcode*. The renewal of *Passcode* does not require the Key Server to be available at all times, as it only involves the two communicating parties to relay the *Hint*.

D. Secure Communication Channel

Having both communicating parties agreed on the *Passcode* for a particular time slot, the sensor data or the actuator command can be encrypted, integrity protected and authenticated. The sensor data is encrypted with the *Passcode* using AES, and a MAC can be generated using SHA256. In particular, *Passcode* is used as the AES and MAC key for sensitive values in each BACnet object (see Section IV). As the PLC automatically changes the *Passcode* after the time slot has expired, the HMI may fail to decrypt the sensor reading when the *Passcode* has expired. In this case, the HMI will automatically request for the new *Hint* from the PLC in order to decrypt the encrypted sensor reading.

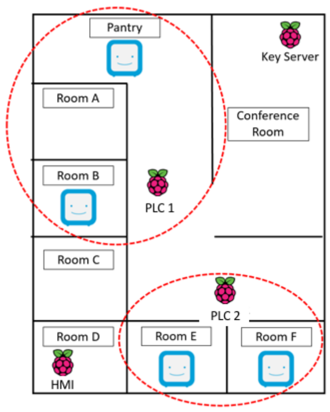


Fig. 3: Environmental Setup of BAS Prototype

IV. Prototype Implementation and Use Case

We have implemented a prototype to demonstrate the feasibility and security of the proposed lightweight key management scheme to secure the communications in BAS. Using environmental sensing and monitoring in a building as a use case, we deployed the prototype in an office space, to allow for the gathering of temperature, humidity, lighting, noise and air quality data and securely transmit them to the HMI.

As shown in Figure 3, the *CubeSensors* were used to monitor room conditions. A Raspberry Pi was used to simulate a PLC, gathering data from the *CubeSensors*. Sensor data were recorded by the PLC periodically every minute. The HMI obtains sensors data by communicating with the PLCs (i.e., Raspberry Pi devices) deployed, so that the state of the building can be visualized and appropriate control commands can be issued according to context changes in the building in a secure manner.

Each BACnet object represents the current status of a device, which can be sent and shared with other devices in the BACnet network. ANSI/ASHRAE 135-2016 specifies sixty standard objects. In order to comply with the BACnet/IP object standard, a new *Security Object* was defined to encapsulate the security parameters, i.e., the *Hash Key*, *Hint* and *Passcode*. The *Security object* contains three properties, namely *Name*, *Description*, *Ciphertext*.

```
         Security Object (ID)
    Name        : PLC-1-Passcodes
    Description : Encrypted Passcodes
    Ciphertext  : A139efJsw...
```

Fig. 4: Example of a Security Object defined for BACnet/IP

As shown in Figure 4, the Key Server hosts a set of *Security Objects* to distribute security parameters, allowing for PLCs and HMI to retrieve their respective *Security Objects* from the Key Server. Similarly, the PLCs also host a *Security Object* encapsulating the *Hint*, to allow for the HMI to retrieve the *Passcode* in every time slot.

As the communication protocol was BACnet/IP, each sensor was configured as a BACnet *Managed Object*. In this case, each PLC has a BACnet *Managed Object* to represent each sensor it manages and the object contains many properties, in particular it uses *Present Value* to indicate the sensor reading in plaintext, which is not desirable in a security context. Instead, using our proposed key distribution scheme, the sensor reading can now be encrypted and integrity protected, by defining a new property called *EncryptedPresentValue* in the *Managed Object*. Whenever a new sensor reading is obtained by the PLC, it encrypts the reading using the *Passcode* obtained from the Key Server and place it in the *EncryptedPresentValue* property. Subsequently, the HMI reads the *EncryptedPresentValue* from the PLC and decrypts it if it is authorised.

In this prototype implementation, there were 24 time slots defined for a day, where the *Passcode* was renewed on an hourly basis. As a result, all the passcodes will be used up after a day, and the key distribution server must distribute fresh *Hint*, *Passcode* and *Hash Key* to the PLCs and HMI on a daily basis. The duration of the time slot can be configured, for applications which require critical monitoring, the key renewal frequency can be increased, thus eliminating the need for key revocation.

V. Evaluation and Results

This section presents the evaluation and performance analysis of $SEABASS$.

A. Validation of Data Confidentiality

Wireshark was used to sniff the network traffic containing BACnet messages. It is trivial that prior to the deployment of the proposed scheme, the *present value* encoded in BACnet *Managed Object* can be eavesdropped and accessed in clear, thus revealing the sensor readings. With the deployment of our scheme, Wireshark no longer had the capability to sniff out the sensor readings, as they were encrypted by the respective *Passcode* using AES.

B. Performance Results

1) Key Server: Hash Key Generation: The deployed Key Server is responsible for generating *Hash Key* thus forming a hash chain to be used between two communicating devices. As there are many sensors and actuators in BAS, we evaluated the timing performance for generating the hash chain as a function of the number of paired devices. As shown in Table I, we first used a Raspberry Pi 2 device to act as the Key Server, and measured the time for generating a hash chain for 50, 500 and 5000 pairs of devices with varying frequencies (number of time slots per day). Our results show that there is no significant difference in terms of performance when generating a hash-chain consisting of 24 or 48 *Hash Keys*. However, when scaling the key renewal frequency to every 15 minutes, thus requiring 72 *Hash Keys* per pair of devices, there was a significant latency. It is noteworthy that the key generation process is taking place offline, and potentially can be offloaded to a more powerful server.

TABLE I: Performance of Hash Key Generation on Raspberry Pi 2

Number of pairs / Generate Hashes	50	500	5000
24 (Hourly)	2.8 s	15.7 s	231.6 s
48 (every 30 minutes)	3.7 s	20.0 s	264.2 s
72 (every 15 minutes)	4.0 s	26.1 s	742.2 s

It is also possible to partition the building by floor, to have a distributed Key Server per floor to be responsible for the generation of security parameters in order to enhance the scalability of the system. One possible consequences is that the system may end up having more attack surfaces, thus it is important that these Key Servers generates the security parameters offline and upon completion of the key distribution, all the generated security parameters are deleted and the Key Servers can be taken offline.

2) Key Server: Hint & Passcode Generation: Each *Passcode* is basically a random number, while the *Hint* is the encryption of the *Passcode* with the designated *Hash Key*. On a Raspberry Pi 2 device, the average time to generate a *Passcode* was approximately 1 ms, and it took 5 – 7 ms to generate a full set of 24 Passcodes.

As the for the *Hint*, the average time required to encrypt a *Passcode* was 1 – 2 ms. Encrypting all the *Passcodes* for a day, i.e., 24 time slots, took approximately 20 – 41 ms on average. Similarly, the generation of *Passcode* and *Hint* can be performed offline.

TABLE II: Summary of Performance Results

Device	**Tasks**	**Average time**
Key Server	Security Parameters generation	45 – 97 ms
Client	Decryption of sensor Reading	2.3 – 2.7 ms
Client	Renewal of Passcode	15 – 20 ms
Sensor	Encryption of Sensor Reading	6.5 – 10.5 ms

3) PLC: Encryption of Sensor Reading: Similar to the encryption of *Passcode*, the sensor reading is encrypted using the *Passcode* using AES. In the deployed system, two sensor readings were obtained, namely temperature and humidity. They were both encrypted separately, i.e., two encryption operations. The average time to perform AES encryption was approximately 6.5 – 10.5 ms on a Raspberry Pi 2.

4) HMI: Decryption of Sensor Reading: Decryption is slightly faster as compared to encryption. In our experiment, the average time to decrypt a sensor reading was between 2.3 – 2.7 ms. Note that for each time slot, in addition to the decryption of sensor readings, the HMI is required to decrypt the *Hint* obtained from the PLC in order to recover the *Passcode* to decrypt the sensor reading. Taking into consideration network access time in the simulated environment, it took approximately 15 – 20 ms to renew the *Passcode* after expiry.

VI. Discussion

Our scheme assumes that the communication between the key distribution server and PLC is protected by using a pre-shared key stored in each PLC's storage disk or ROM. If a particular PLC's pre-shared key is compromised, the human administrator can re-provision a new pre-shared key into the compromised PLC. Our paper's proposed key distribution scheme is orthogonal to that of a pre-shared key scheme: the former protects the communication between HMIs and PLCs, while the latter protects communication between the key distribution server and PLCs. The reason we do not use pre-shared keys for communication between HMIs and PLCs is because we assume key distribution servers are more robust and have a smaller attack surface than HMIs. If HMIs use pre-shared keys with PLCs and one HMI gets compromised, the human needs to manually change the pre-shared keys of all PLCs that was shared with the compromised HMI.

Meanwhile, using our key management scheme, a compromised HMI's passcode will be automatically expired after a certain period of time. During this period, PLCs should not provide data to this compromised HMI. One form of revocation that is similar to CRL can be employed, the key distribution server can distribute compromised *Passcodes* or *Hash Keys* to the PLCs and HMIs, in order to invalidate messages encrypted with the compromised *Passcodes*. Alternatively, a tree-based revocation scheme as described in [12] can be used, in which each server is responsible for maintaining a revocation tree to invalidate *Passcodes*.

VII. Conclusions and Future Work

This paper has provided a first attempt to deploy key distribution and management for Building Automation Systems, which in turn provides confidentiality and authenticity to sensor reading and commands in a building management scenario. The novelty of $SEABASS$ is that it is symmetric-key based, and hence lightweight compared to public-key based approach. The session keys to protect the communications in BAS are automatically and regularly renewed without requiring the Key Server to be available all the time.

We implemented and deployed $SEABASS$ using Raspberry Pi 2 to simulate a SCADA environment. The preliminary results look promising and we observed that the proposed system is fully deployable.

As a future work, we plan to further refine the solution based on real industry requirements and deploy $SEABASS$ on real PLCs or OpenPLCs in order to further validate the security, performance and reliability of the solution.

References

[1] D.G.Holmberg. BACnet wide area network security threat assessment. *National Institute of Standard and Technology*, NISTIR 7009, 2003.

[2] D. Fisher. BACnet ™and LONWORKS: Compared and Contrasted. *PolarSoft Inc*, May 2004.

[3] N. Georgios.L, Gilbert.C and Maher.K. Towards the next generation of intelligent building: An assessment study of current automation and future IoT based systems with a proposal for transitional design. *Sustainable Cities and Society*, 28:473–481, January 2017.

[4] Hyunguk.Y and Taeshik.S. Challenges and research directions for heterogeneous cyberâĂŞphysical system based on IEC 61850: Vulnerabilities, security requirements, and security architecture. *Future Generation Computer Systems*, 61:128–136, August 2016.

[5] B. International. *Introduction to BACnet For Building Owners and Engineers*, volume 1. 2014.

[6] C. S. Jan.H, Vlasios.T and David.B. *From Machine-To-Machine to the Internet of Things*. Elsevier Ltd, 2014.

[7] Jignesh.B and H. Verma. Design and Development of Wired Building Automation Systems. *Energy and Buildings*, 103:396–413, September 2015.

[8] A. Justyna.J.C and Boudewijn.R.H. In *WhatâĂŹs under the hood? Improving SCADA security with process awareness*, Cyber- Physical Security and Resilience in Smart Grids (CPSR-SG), Joint Workshop. IEEE, 2016.

[9] J. Kaur, J. Tonejc, S. Wendzel, and M. Meier. *Securing BACnet's Pitfalls*, pages 616–629. Springer International Publishing, Cham, 2015.

[10] B. P. S. Kieran.M, Ivo.F and Gavin.M. *Secure Communications in Smart Grid: Networking and Protocols*, volume 21. 2015.

[11] E. D. Knapp and J. T. Langill. *Chapter 6 - Industrial Network Protocols*, volume Industrial Network Security (Second Edition). 2015.

[12] H. Ko, J. Jin, and S. L. Keoh. Viotsoc: Controlling access to dynamically virtualized iot services using service object capability. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, CPSS '17, pages 69–80, New York, NY, USA, 2017. ACM.

[13] A. Kostas.M, Nick.F and Vangelis.G. Towards generic SCADA simulators: A survey of existing multi-purpose co-simulation platforms, best practices and use-cases. *Scientific Cooperations International Workshops in Electrical-Electronics Engineering*, pages 33–39, 2013.

[14] M. P. Michael N Johnstone and J. den Hartog. In *Timing attack detection on BACnet via a machine learning approach*, Lecture Notes in Computer Science, pages 57–64. Security Research Institute (SRI), Edith Cowan University 13th Australian Information Security Management Conference, 2015.

[15] D. Muhammad.W.A, Monjur.M, Mario.S, and Yacine.R. Building energy metering and environmental monitoring âĂŞ A state-of-the-art review and directions for future research. *Computer Standards and Interfaces*, 45:1–12, 2016.

[16] R. Pedro.D, Paulo.C and Wolfgang.K. Building automation systems: Concepts and technology review. *Computer Standards and Interfaces*, 45:1–12, 2016.

[17] S. Raphael.A and Ernest.F. Securing DNP3 Broadcast Communications in SCADA Systems. *IEEE Transactions on Industrial Informatics*, 2016.

[18] C. J. Sajal.B, Nishchal.K and Ernest.F. Practical Modbus Flooding Attack and Detection. In *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014),*, volume 149, pages 57–65, 2014.

[19] F. Wolfgang.G and I. Wolfgang.K, Member. Security in Building Automation Systems. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, 57, November 2010.

[20] X. Yan and W. Bo. A Security Extension to LonWorks/LonTalk Protocol. *International Journal of Digital Content Technology and its Applications (JDCTA)*, 7 Issue 6:790, March 2013.