

On the Security of the Sensor Cloud Security Library (SCSLib)

¹Mohamed Rasslan, ¹Mahmoud Nasreldin, ¹Ghada Elkabbany and ²Aya Elshobaky

¹Electronics Research Institute, Ministry of Higher Education and Scientific Research, Cairo, Egypt

²Alexandria University, Alexandria, Egypt

Article history

Received: 08-02-2018

Revised: 02-05-2018

Accepted: 21-05-2018

Corresponding Author:

Mohamed Rasslan
Electronics Research
Institute/Ministry of Higher
Education and Scientific
Research, Cairo, Egypt
Email: mohamed@eri.sci.eg

Abstract: Nowadays, cloud computing has experienced remarkable growth. Indeed, one of the main challenges of cloud computing is to protect data from different sensor devices during processing and storing. In 2014, Henze *et al.* proposed the Sensor Cloud Security library (SCSLib) which allows cloud developer to access encrypted sensor data stored in the cloud. SCSLib drawback is that it allows a dishonest referee, dealing with a dispute, to decrypt all the future and past authenticated cipher-text between parties. In this study, we proposed an improved scheme for the SCSLib that prevent the users of the library from such attacks. In addition, the proposed scheme fulfils the following security requirements: Integrity, authenticity, confidentiality and availability. Moreover, security analysis in the cloud-computing environment is presented.

Keywords: Cloud Computing, Sensor Networks, Sensor Cloud Security Library (SCSLib), Integrity, Authenticity, Confidentiality

Introduction

Recently, Wireless Sensor Networks (WSNs) are known by their ability to enable new solutions in many applications such as health-care, environmental monitoring, transportation business and industrial automation. With the growth of sensor networks, many challenges have appeared in terms of flexibility, scalability and heterogeneous information services. The integration of WSNs with cloud provides better flexibility, unlimited resources, immense processing power and the capability to provide quick response to the user (Calik *et al.*, 2004). Cloud computing has a distributed design and encloses a computational model which enables it to improve availability, scalability, agility, collaboration and adaptability of the system. Cloud computing which is an extension of grid computing is a highly developing field in IT industry. This has led to significant advantages to other fields such as: Research, education, banking, medicine and entertainment. Cloud computing technology allows reducing the rates spent on computing infrastructure, increasing performance and rising efficiency of an organization (Hu *et al.*, 2017). With the rapid development of the Internet of Things (IoT) and its integration with the cloud and due to their powerful processing and storage abilities of cloud computing for sensing data, they have received significant interest from industry and academia fields (Khan and Al-Yasiri, 2016). In the IoT-cloud, physical sensors are represented

as virtual sensors. The main functions of these sensors are to sense and forward their sensing data to the cloud. The IoT-cloud offers sensing services directly to various nodes/applications through virtual sensors. Nodes and applications request on demand sensing services from the IoT-cloud (Khan and Al-Yasiri, 2016).

Cloud computing main objective is to provide secure, quick and convenient data storage with all services delivered over the internet. There are several advantages and benefits of sensor cloud infrastructure such as, better analysis of data, scalability, collaboration, visualization, free provisioning of increased data storage, processing power, dynamic provisioning of services, automation, flexibility, agility of services, resource optimization and quick response time. There are many applications of sensor networks such as, emergency response information (sensor networks will collect information about the status of buildings, people and transportation pathways), energy management (energy distribution will be better managed when we begin to use remote sensors), medical monitoring (instant release of emergency medication to the bloodstream), logistics and inventory management and battlefield management (remote sensors can help eliminate some of the confusion associated with combat). In this study, we give a detailed description of Sensor Cloud Security library (SCSLib) proposed by Henze *et al.*, which allows cloud developer to access encrypted sensor data stored in the cloud. We show that SCSLib do not protect sensor data in the cloud from authenticity attacks such as, replay attack.

The remainder of this paper is organized as follows. In the next section, we briefly review the details of security requirements and types of authentication attacks in cloud environment. Then, the details of Burrows-Abadi-Needham logic (BAN logic) main rules are presented. In addition, we describe SCSLib library and its expected attacks. Then, in the next section, the improved scheme of SCSLib to protect data in the cloud is proposed. Latter, logical analysis of SCSLib and its proposed modification using BAN-logic are presented. Finally, the paper concludes in the last section.

Background

Wireless Sensor Networks Security

Availability, data integrity, data confidentiality, data freshness, time synchronization, secure localization, authentication and self-organization are the security requirements in Wireless Sensor Networks (WSN) (Finogeev and Finogeev, 2017). Availability guarantees that services are functioning even under attacks. Data integrity means that received message is identical to the sent message. Data confidentiality implies that legitimate nodes only understand messages. Data freshness deploys nonce or time-stamp in order to resist replay attacks. Secure mechanisms in WSNs need time synchronization. Secure localization aims to accurately and automatically locate each sensor node in a WSNs (e.g., locate a fault). Authentication ensures that communicating node is the one that it claims to be. In WSNs, nodes have to be self-organizing and self-healing. Cryptographic mechanisms protect the confidentiality and authenticity of communication channels in WSN against outsider attacks such as, eavesdropping, replay, modification, or spoofing of packets. Denial-of-Service (DoS) attacks against networks (disrupt, subvert, or destroy a network) availability in physical layer (jamming attack) can be prevented by deploying spread-spectrum communication such as, frequency hopping and code spreading. Moreover, DoS in data link layer can be prevented by using error-correcting codes. Message Authentication Code (MAC) is used against spoofing, alteration and stealthy attack (in which the attacker compromises a sensor node and injects false data through that sensor node). DoS attacks against WSNs could damage the safety of people.

Security in Cloud

Cloud security is one of the main inhibitors for cloud adoption nowadays: Transferring resources and data to the cloud has the result of a loss of control that makes risk analysis and mitigation more difficult and avoids potential customers from using cloud computing in their applications. In order to overcome this problem, cloud-application's developers should take into account the

possible security problems from the beginning and should try to make most benefits of the flexibility offered by the cloud standard. Also, they should consider the security restrictions by cloud customers (Dinh *et al.*, 2017). There is a great concern about legality and confidentiality of data while developing a secure cloud (Casola *et al.*, 2016). According to Potey *et al.* (2016) cloud computing security can be classified as the following main objectives: Availability, confidentiality, integrity, authenticity and accountability:

- Availability ensures that data and services are always available for its users at any time, at any place
- Confidentiality means that user's data are only available to authorized users, which keeps their data secret
- Integrity assures that data has not been changed during storage and transmission over the network
- Authenticity is a core security requirement in cloud computing systems, which requires mutual trust between the parties (Abbas and Khan, 2014; Abbas *et al.*, 2017). In addition, it is important for authenticity to validate that both parties involved are who they claim to be
- Accountability assures that no entity can deny its participation in a data transfer between them

Examples on WSN Security in Cloud Environment

To securely control sensor data storage and process sensor data effectively in the cloud environment, there are different techniques such as, OAuth-protocol (establish a secure connection between the data owner/cloud-platform and Trust Point), Object security from trust point to Service (provides security to the individual data fields and integrity checksum covering the complete data item), Service assurance and Data access Granting (the data owner looks up a service and its service description in the cloud service marketplace, then instructs her trust point to encrypt her data for the selected service). Furthermore, some of the important security concepts in sensor clouds are classified follow:

Secure Sensor Information System

Cloud must ensure the strengthen of the security issue in sensor-cloud while sensing data under sensing as a service, the secure cloud architecture has a dedicated layer to deal the security issue, the security management layer (has three parts: Identity authentication unit, resource access control and data encryption unit).

Enhance Security in Multicasting

Secure multicasting method is not only ensuring secure data transmission but also enhance the resistance of the system against introducers, by using a key

exchange algorithm while transferring data between tiers in a secure manner.

Multilevel Authentication System

Users should successfully own passwords at all levels to access the cloud services (Saha *et al.*, 2016).

Types of Authentication Attacks

Authentication is a major challenge in cloud computing services. This is due to the fact that it is frequently targeted by an attacker (Sumitra *et al.*, 2014; Pawar and Anuradha, 2015; Chouhan and Singh, 2016). A primitive way to provide authentication is the use of simple username and password. Other advanced methods use various forms of secondary authentication such as: Shared secret questions, site keys and virtual keyboards. According to (Chouhan and Singh, 2016) some of the authentication attacks are:

- *Brute force attacks*: This attack is conducted by trying all possible combinations of password to break it. This attack is generally applied to crack the encrypted passwords
- *Dictionary attack*: This attack is relatively faster than brute force attack. While brute force attack checks all encrypted passwords, the dictionary attack tries to match the password with most occurring words
- *Shoulder surfing/spying attack*: In this type of attack, the attacker spies the user's movements to deduce the password
- *Phishing attacks*: In this attack, the attacker uses the web to redirect the user to the fake website to obtain its passwords/pin codes
- *Key loggers*: They are software programs which observe the user actions by recording each key pressed by the user
- *Replay/Reflection attacks*: It is a type of network attack where a valid data is repeated or delayed maliciously

Replay attack is a major violation of security where information is stored without authorization. Then, stored information is retransmitted to force the receiver into unauthorized operations such as: False identification, false authentication, or a duplicate transaction. Although the replay messages may be ciphered, in addition, the attacker may not know the actual keys and passwords, the retransmission of valid logon messages is sufficient to gain access to the network.

Burrows-Abadi-Needham logic (BAN logic)

Authentication protocols are the main security component in cloud computing and it is necessary to ensure the correctness of these protocols. In literature, many protocols contain redundancies or security flaws. Burrows *et al.* (1989) proposed a logic method called BAN logic to analyze authentication protocols. With the logic, all public - and shared key primitives are

formalized. BAN logic is a set of rules that define and analyze information exchange protocols. In particular, BAN logic aids its users to determine the trustworthiness and the security of the exchanged information. BAN logic begins with the assumption that the exchanged information is on media exposed to tampering and public monitoring. The BAN logic makes it possible to reason in a simple way over cryptographic protocols in a formal way. The basis for the BAN logic is the belief of a party in the truth of a formula. A formula does not necessarily be true in the general sense of truth (Wessels, 2001).

BAN logic uses the logic to describe the authentication protocols. They transformed each message into a logical formula which is an idealized version of the original message. For a successful verification, the belief state of communicating parties should satisfy the protocol goals. They consider that the authentication check is completed between principals Alice and Bob, if there is a data packet "X" which the recipient Bob believes is sent by the sender (signer), Alice. Thus, authentication between Alice and Bob will be completed if $Bob \models Alice \models X$ and $Bob \models X$, where the symbol \models means believe.

First, the basic rules of the BAN logic are listed below:

The Interpretation Rule:

$$\frac{Bob \models (Alice \sim (X, Y))}{Bob \models (Alice \sim X), Bob \models (Alice \sim Y)}$$

The above rule means that if Bob believes that Alice once said a message containing both X and Y, therefore he believes that Alice once said each statement separately.

Message Meaning Rule:

$$\frac{Bob \models \frac{Q - Alice}{Alice} \rightarrow Alice, Bob \triangleleft [X]_{S-Alice}, Alice \neq Bob}{Bob \models Alice \sim X}$$

This means that if Bob believes that Q-Alice is the public key of Alice and Bob sees a message X signed by Alice's secret key S-Alice, this implies that Bob believes that Alice once said X.

Nonce Verification Rule:

$$\frac{Bob \models \#(X), Bob \models Alice \sim X}{Bob \models Alice \models X}$$

The above rule means that if Bob believes that X is a recent message and Alice once said X, therefore it believes that Alice believes in X.

Jurisdiction Rule:

$$\frac{Bob \models Alice \Rightarrow X, Bob \models Alice \models X}{Bob \models X}$$

This rule means that if Bob believes that Alice has jurisdiction over X and Bob believes that Alice believes in X , then Bob believes in X .

Freshness Rule:

$$\frac{Bob \models \#(X)}{Bob \models \#(X, Y)}$$

The above rule means that if Bob believes in the freshness of X and Y , therefore it believes in the freshness of each statement separately. The analysis is undertaken for the message exchanged between the sender, Alice and recipient, Bob. In this study, BAN logic is used to check the correctness of SCSlib library and its proposed modification.

Overview of SCSlib Library Henze et al. (2014)

In their paper, Henze *et al.* (2014) proposed a library that permits cloud service developers to manipulate secure sensor information in the cloud without the need to deal with cryptographic operations details. The library is called Sensor Cloud Security Library (SCSlib). SCSlib is implemented as a “C” library which uses the encryption algorithms of the OpenSSL library. The main functions of SCSlib are given below:

- Interfacing with the cloud
- Processing of sensor data items (verification and decryption)
- Caching of cryptographic keys for performance improvement

At the receiver side, which is the sensor cloud, SCSlib library main objectives are to check the integrity and authenticity of the sensor data. SCSlib gets the data source public key using the corresponding callback function `Sc_process_data_item()`. Then, it verifies the digital signature in the JSON Web Signature (JWS) using the retrieved public key. For decryption of sensor information, SCSlib searches JSON Web Encryption

(JWE) objects which represent the encrypted sensor data as shown in Fig. 1. This is done through recursive iterations over the JSON-serialized object. For each JWE, the decryption key is obtained using the above-mentioned call back function. Finally, to get the original information, the encrypted sensor value is decrypted using the decryption key.

Methods

Henze *et al.* (2014) proposed an “encryption-then-blind signature with designated verifier” scheme to prove the authenticity and integrity of the evidence in cloud environment. Encrypt-then-sign scenario has a potential draw back which is the susceptibility to replay attack, as shown in Fig. 2.

In the next sub-section, the BAN logic security analysis of Henze *et al.*’ scheme used in SCSlib is presented. Then, the scenario of the proposed attack is given. Next, the proposed modification to SCSlib library to overcome the replay attack problem is illustrated. Finally, our proposed modification is analyzed using BAN logic.

Logical Analysis of SCSlib Library using BAN Logic

Assume that Alice has discovered a breakthrough business idea and wants to inform her boss, Bob, about her discovery. Then, Alice will encrypt the message “ M ” using Bob’s public key (Q-Bob) and then sign the result using her secret key (S-Alice). Next, Alice sends the following message: $[\{M\}_{Q-Bob}]_{S-Alice}$ to Bob. However, Eve can set herself as a man-in-the middle and intercept the messages from Alice to Bob. Eve can then use Alice’s public key to compute $\{M\}_{Q-Bob}$. Then, Eve signs using S-Eve it and sends $[\{M\}_{Q-Bob}]_{S-Eve}$ to Bob. When Bob receives $[\{M\}_{Q-Bob}]_{S-Eve}$ and verifies Eve’s signature on it, Bob will assume that Eve has made this astonishing discovery and Alice cannot disprove Eve’s claim, as shown in Fig. 3 and 4.

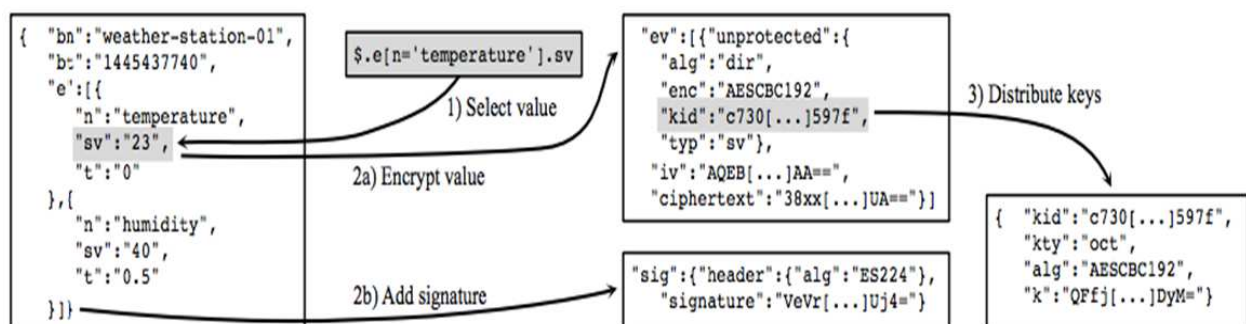


Fig. 1: Overview of the process of protecting a sensor data item according to (Henze *et al.*, 2014)

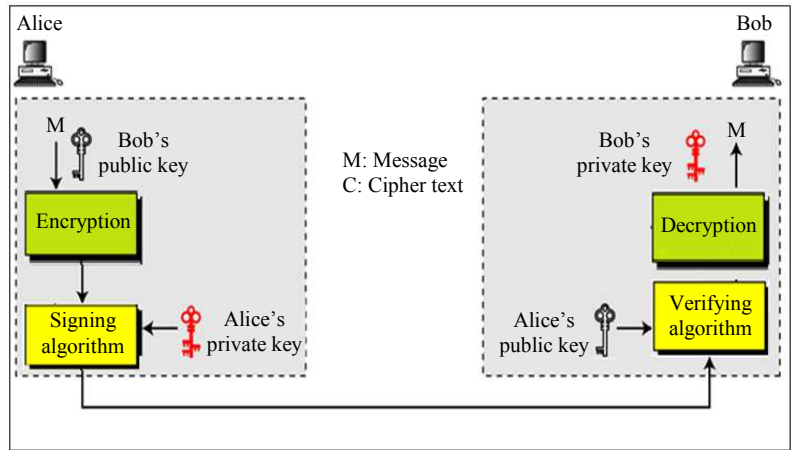


Fig. 2: Encrypt-then-sign scheme

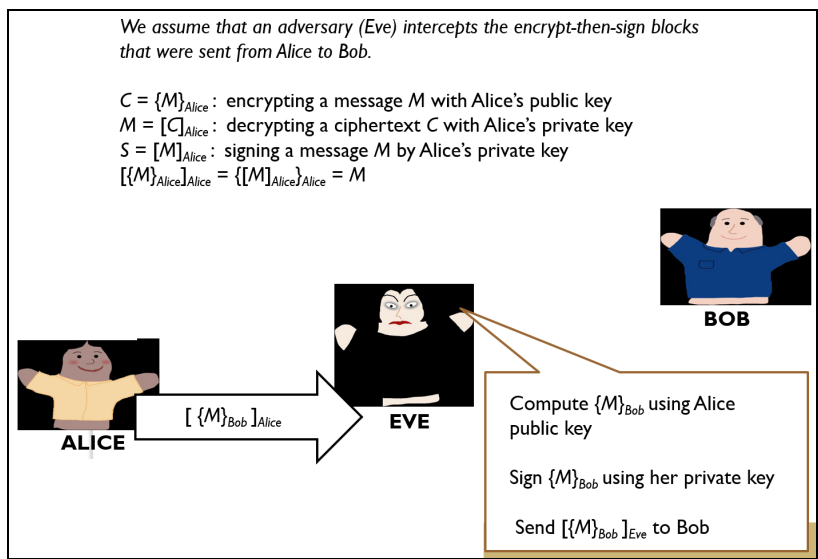


Fig. 3: Cryptanalysis - Step 1

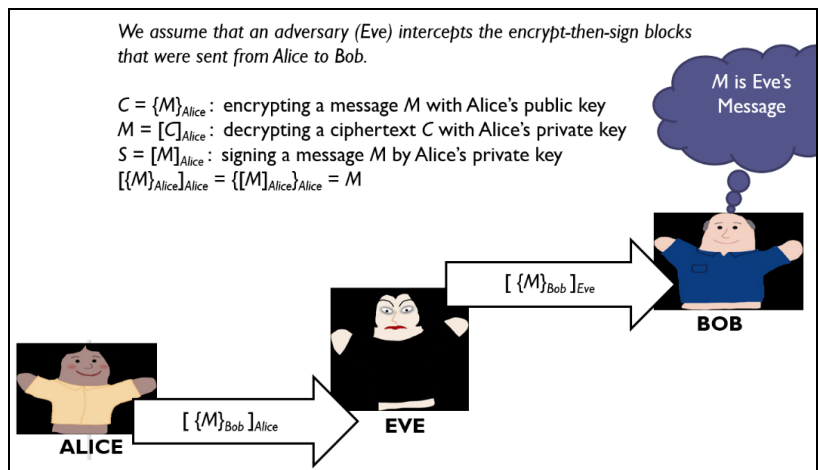


Fig. 4: Cryptanalysis-step 2

To achieve complete authentication between Alice and Bob, the following goals must be achieved:

$$\text{Goal 1: } Bob \models Alice \models M_i$$

$$\text{Goal 2: } Bob \models M_i$$

where, M_i represents the message sent by Alice.

In order to complete the analysis, the following assumptions are made:

$$Alice \models \xrightarrow{Q - Bob} Bob \quad (1)$$

$$Bob \models \xrightarrow{Q - Alice} Alice \quad (2)$$

$$Bob \models Alice \Rightarrow M_i \quad (3)$$

Equation 1 indicates that Alice believes that Q-Bob is the public key of Bob. Then, Equation 2 indicates that Bob believes that Q-Alice is the public key of Alice. Equation 3 indicates that Bob believes that Alice has jurisdiction over M_i . After making these assumptions, the messages exchanged in the initial phase are transformed into logical formulas. Finally, the basic rules of BAN logic will be applied to these formulas. Following is the transformation of the proposed attack into logical formulas:

$$Alice \ Bob : \left[\{M_i\}_{Q-Bob} \right]_{S-Alice} \quad (4)$$

The analysis of the protocol can now be performed. By applying message meaning rule to Equation 4 and using Equation 1, the following can be deduced:

$$Bob \models Alice \sim M_i \quad (5)$$

But, there is no timestamp or nonce. Thus, Bob does not believe in the freshness of M_i (Equation 4). From Equation 5, one can deduce that Henze *et al.* (2014) do not achieve the goals of authenticity and that the sent message could be intercepted by a man-in-the-middle.

The Proposed Attack

Here we will focus on the encryption/signature at the sensor node and decryption/verification at the sensor cloud. Since the invention of the public key cryptography, the cryptographers have known that the combination of encryption and signature tends to insecure results. The simple ‘‘Encrypt and Sign’’ recipient knows only who wrote the message and has no assurance about who encrypted it.

In this section, we introduce the suggested attack on SCSlib library. The proposed attack shows that man-in-the-middle can intercept the traffic between the sensor-node and the cloud as shown in Fig. 5:

- The attacker intercepts the traffic between sensor-node and the cloud
- The attacker verifies the digital signature of the sensor-node (SN_{sig}) on the encrypted data using the sensor-node public-key (SN_{pub})
- The attacker signs the encrypted data using the attacker's private-key ($Attacker_{priv}$)
- The attacker sends it again to the cloud but with her/his signature ($Attacker_{sig}$)

As demonstrated above, the attacker gets a copy of the signed-encrypted data and could forge all future and past traffic between the contented parties. This scenario shows the absence of integrity and authenticity properties in SCSlib library.

The Proposed Modification of the SCSlib Library

In this section, the security mitigation to SCSlib library is proposed. This modification overcomes the drawback mentioned earlier. In order to detect the replay attacks, SCSlib needs to add additional information to the message, which enables the receiver to verify the freshness of the message. We propose an enhanced SCSlib library that protects the cloud from replay attacks. There are two possible scenarios: Sign-Encrypt-Sign and Encrypt-Sign-Encrypt (shown in Fig. 6). In this study, we use the ‘‘Sign-Encrypt-Sign’’ approach (Nasreldin *et al.*, 2015).

The following tasks are done at the sensor side:

- Generate a hash value of the plaintext, then sign it with the sensor private-key (SN_{priv}) as the first signature and include it with the plain text
- Encrypt both the plain text and the first signature using the cloud public key (C_{pub})
- Generate a hash value for the encrypted data and sign it using the private-key of the sensor node (SN_{priv}) as the second signature and send it to the cloud

The following tasks are done at the sensor cloud side:

- Verify the second signature of the protected data
- Decrypt the protected signed data in order to generate the original signed plain text
- Verify the first signature with the original plain text. If it verifies, the integrity, authentication, confidentiality and availability features of SCSlib are achieved. The Block diagram of the modified library is shown in Fig. 7

Logical Analysis of the Modified SCSlib Library using BAN Logic

There are two possible scenarios: Sign-Encrypt-Sign and Encrypt-Sign-Encrypt. Both scenarios can

resist the cipher-text forwarding attack and its consequences. To ensure confidentiality and chain of custody at the sender side, another signature step is needed before the encryption step. Moreover, at the recipient side, another encryption step is executed

after the message decryption. That is to say, these are two extra steps (one block for signature at the sender side and one block for encryption at the recipient side) are added to overcome the plaintext-subsection and cipher text stealing attacks.

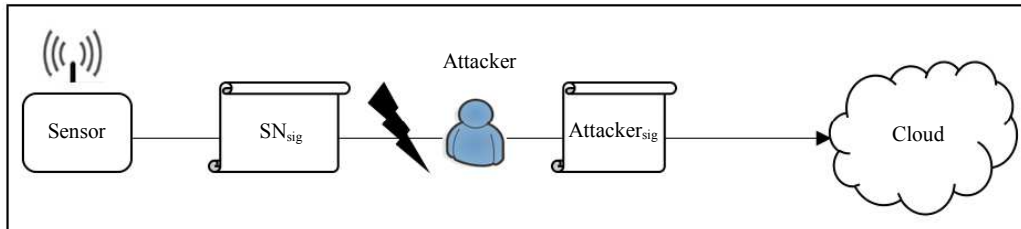


Fig. 5: Cloud security attack

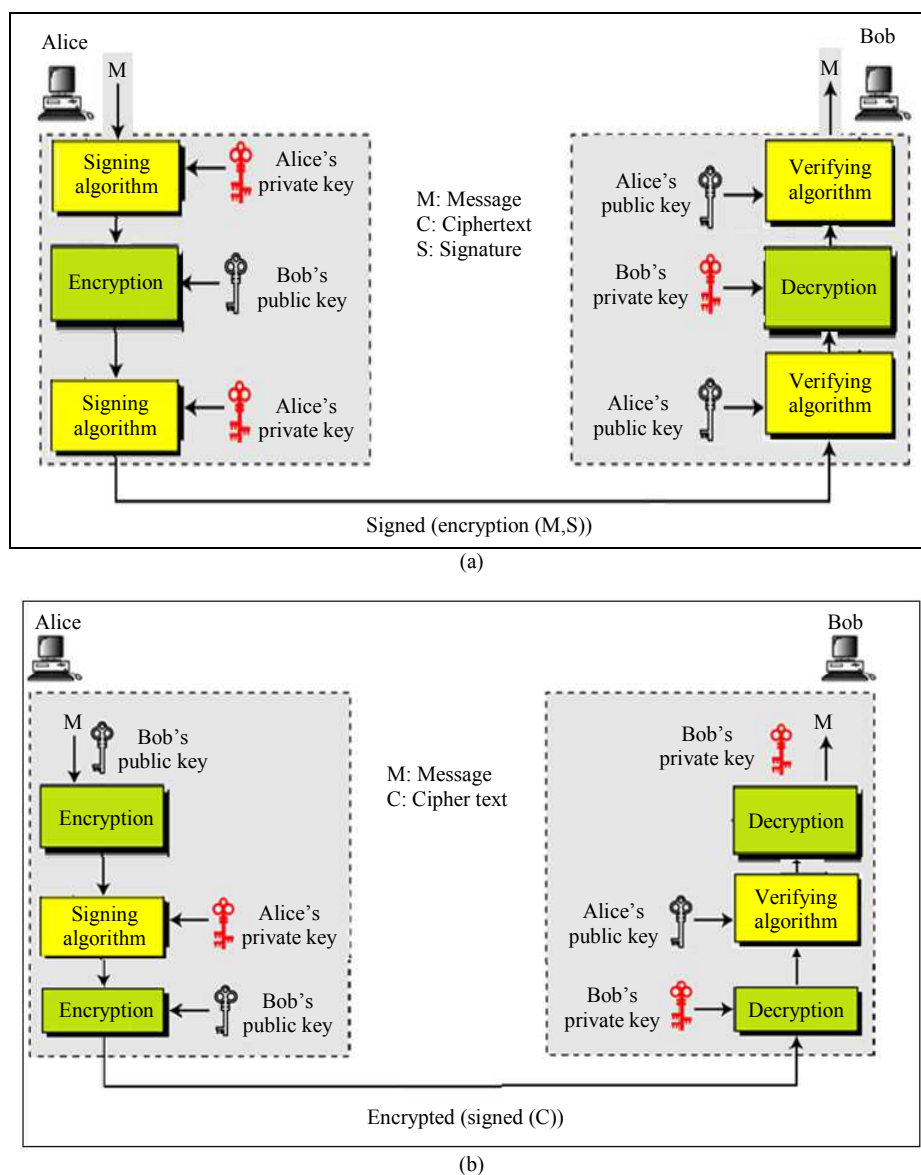


Fig. 6: Sign-encrypt-sign and encrypt-sign-encrypt (a) sign-encrypt-sig (b) encrypt-sign-encrypt

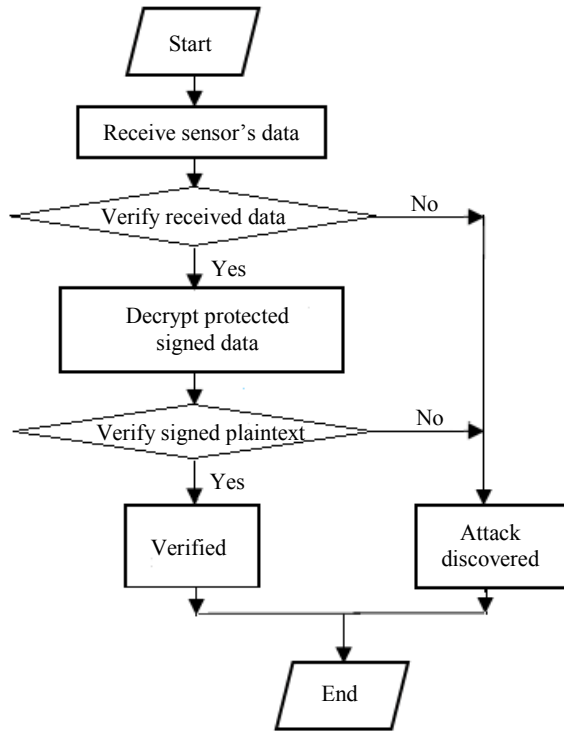


Fig. 7: Flowchart verify/decrypt/verify of data at the cloud

In the following, we present the analysis for the message exchanged between the sender, Alice and recipient, Bob. The authentication is considered complete between Alice and Bob, if the following goals are achieved:

- Goal 1: $Bob \models Alice \models M_i$
- Goal 2: $Bob \models M_i$

where, M_i represents the message sent by Alice:

- Alice, first, signs the message M_i using its secret key. Then, it encrypts both the message and the signature using Bob's public key. Finally, the output of encryption is signed using her secret key.
- Alice sends $\{\{M_i, T\}_{S-Alice}\}_{Q-Bob}\}_{S-Alice}$ to Bob, where T is the timestamp generated by Alice.

In order to complete the analysis, the following assumptions are made:

$$Alice \models \xrightarrow{Q-Bob} Bob \quad (6)$$

$$Alice \models \xrightarrow{Q-Alice} Alice \quad (7)$$

$$Bob \models \xrightarrow{Q-Alice} Alice \quad (8)$$

$$Bob \models \xrightarrow{Q-Bob} Bob \quad (9)$$

$$Bob \models Alice \Rightarrow M_i \quad (10)$$

$$Alice \models \#T \quad (11)$$

$$Bob \models \#T \quad (12)$$

Equation 6 indicates that Alice believes that Q-Bob is the public key of Bob. Equation 7 indicates that Alice believes that Q-Alice is the public key of herself. Then, Equation 8 indicates that Bob believes that Q-Alice is the public key of Alice. Equation 9 indicates that Bob believes that Q-Bob is the public key of himself. Then, Equation 10 indicates that Bob believes that Alice has jurisdiction over the block sent. Finally, Equation 11 and 12 indicate that Alice and Bob believe in the freshness of T (since it is changed for each message). After making the assumptions, the messages transferred in the initial phase are transformed into logical formulas. Finally, the basic rules of the BAN logic will be applied to the logical formulas.

Using the message $\{\{M_i, T\}_{S-Alice}\}_{Q-Bob}\}_{S-Alice}$, Equation 8 and message meaning rule:

$$Bob \models Alice \sim (T, M_i)$$

But, Alice and Bob believe in the freshness of T_i (Equation 11 and 12). Thus, applying nonce verification rule, the following is obtained:

$$Bob \models Alice \models M_i \quad (13)$$

Then, by applying jurisdiction rule using Equation 10, the following is obtained:

$$Bob \models M_i \quad (14)$$

From Equation 13 and 14, one can deduce that the proposed modification achieves the goals of authentication without bugs or redundancies.

Conclusion

SCSlib library that enables cloud service developers to transparently access protected sensor data was

proposed by Henze *et al.* (2014). This library fails to simultaneously satisfy both the integrity and authenticity properties that are required for the sensor clouds. A third party can intercept the traffic between the sensor node and sensor cloud. Replay attack is a major trouble to the authenticity of the cloud. In this study, an improved scheme for SCSlib library that could prevent the users of the library from such attacks is proposed. This is done by using the “Sign-Encrypt-Sign” approach. The proposed modification fulfills integrity, authenticity, confidentiality and availability at the same time. In addition, BAN logic is used in order to perform the security analysis of both SCSlib and our modified scheme. The analysis shows that SCSlib library does not fulfill the claimed security goals. On the other hand, the analysis shows that the proposed modification to SCSlib library achieves goals of authentication and integrity without bugs.

Acknowledgement

We thank Prof. Heba Aslan for assistance with reviewing and comments that greatly improved the manuscript.

Author’s Contributions

The author prepared the study, elaborated the methodology, performed the analysis and wrote the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that no ethical issues involved.

References

- Abbas, A. and S. Khan, 2014. A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. *IEEE J. Biomed. Health Informat.*, 18: 1431-1441. DOI: 10.1109/JBHI.2014.2300846
- Abbas, H., O. Maennel and S. Assar, 2017. Security and privacy issues in cloud computing. *Ann. Telecommun.*, 72: 233-235. DOI: 10.1007/s12243-017-0578-3
- Burrows, M., M. Abadi and R. Needham, 1989. A logic of authentication. *Math. Phys. Eng. Sci.*, 426: 233-271. DOI: 10.1098/rspa.1989.0125
- Calik, P., P. Yilgora, P. Ayhanb and S. Demir, 2004. Oxygen transfer effects on recombinant benzaldehydelyase production. *Chem. Eng. Sci.*, 59: 5075-5083. DOI: 10.1016/j.ces.2004.07.070

- Casola, V., A. De Benedictis, M. Rak and E. Rios, 2016. Security-by-design in clouds: A security-SLA driven methodology to build secure cloud applications. *Proc. Comput. Sci.*, 97: 53-62. DOI: 10.1016/j.procs.2016.08.280
- Chouhan, P. and R. Singh, 2016. Security attacks on cloud computing with possible solution. *Int. J. Adv. Res. Comput. Sci. Software Eng.*, 6: 92-96.
- Dinh, T., Y. Kim and H. Lee, 2017. A location-based interactive model of Internet of Things and cloud (IoT-cloud) for mobile cloud computing applications. *J. Sensors*, 2017: 489-489. DOI: 10.3390/s17030489
- Finogeev, A. and A. Finogeev, 2017. Information attacks and security in wireless sensor networks of industrial SCADA systems. *J. Industrial Inform. Integrat.*, 5: 6-16. DOI: 10.1016/j.jii.2017.02.002
- Henze, M., S. Bereda, R. Hummen and K. Wehrle, 2014. SCSlib: Transparently accessing protected sensor data in the cloud. *Proc. Comput. Sci.*, 37: 370-375. DOI: 10.1016/j.procs.2014.08.055
- Hu, J., C. Chen, C. Fan and K. Wang, 2017. An intelligent and secure health monitoring scheme using IoT sensor based on cloud computing. *J. Sensors*, 2017: 1-11. DOI: 10.1155/2017/3734764
- Khan, N. and A. Al-Yasiri, 2016. Identifying cloud security threats to strengthen cloud computing adoption framework. *Proc. Comput. Sci.*, 94: 485-490. DOI: 10.1016/j.procs.2016.08.075
- Nasreldin, M., M. El-Hennawy, H. Aslan and A. El-Hennawy, 2015. Digital forensics evidence acquisition and chain of custody in cloud computing. *Int. J. Comput. Sci.*, 12: 153-160.
- Pawar, M. and J. Anuradha, 2015. Network security and types of attacks in network. *Proc. Comput. Sci.*, 48: 503-506. DOI: 10.1016/j.procs.2015.04.126
- Potey, M., C. Dhote and D. Sharma, 2016. Homomorphic encryption for security of cloud data. *Proc. Comput. Sci.*, 79: 175-181. DOI: 10.1016/j.procs.2016.03.023
- Saha, S., Das, R., Datta, S. and Neogy, S., 2016. A cloud security framework for a data centric WSN application. *Proceedings of the 17th International Conference on Distributed Computing and Networking*, Jan. 04-07, ACM, Singapore, pp: 1-6. DOI: 10.1145/2833312.2849559
- Sumitra, B., C.Pethuru and M. Misbahuddin, 2014. A survey of cloud authentication attacks and solution approaches. *Int. J. Innovat. Res. Comput. Commun. Eng.*, 2: 6245-6253.
- Wessels, J., 2001. Application of BAN-Logic. Technical report, CMG Public Sector B.V.

Appendix 1: The Description of Both SCSlib and the Modified SCSlib Libraries

SCSlib and the modified SCSlib libraries have the same configuration and setup processes. The differences between these libraries are:

1. At sender side (the sensor networks): The way of creating Json file that will be sent to the cloud
2. At the receiver side (sensor cloud): The way, the library will decrypt and verify the received Json file

Setup Process of the Library

For both libraries, the following packages are needed to be installed, through the respective packet manager in order to successfully build the library:

- Libjansson (Version 2.5 was used for development)
- Libssl
- Automake
- Autoreconf
- Make
- Gcc
- Libtool
- Autotools

Installation

Both libraries use automated tools as building environments. The following steps have to be performed in order to build and install the library:

- Autoreconf -fi
- ./configure
- Make
- Sudo make install

Data Format at the Nodes

The different parts in the Json file are: "Ciphertext" and "sig" tags. In SCSlib, these tags present the encrypted data and the outer signature, while in the modified SCSlib they present the encrypted-text (data + inner signature) and the outer signature. That is to say, at SCSlib: The Ciphertext → is the encrypted data only. While, at the modified SCSlib the ciphertext → is the encrypted (data + inner signature)

Data Format at the Cloud

SCSlib verifies the received data, by verifying the outer signature. Then, it generates the original data by decrypting the cipher text. On the other hand, modified SCSlib verifies the received data at the first time by verifying the outer signature. Then, it generates the

original signed data by decrypting the cipher text. Later, it verifies the generated original message by verifying the inner signature. Finally, it compares the outer and the inner signature.

Appendix 2: Building an Example

In order to test the minimal example:

- Make sure that the library is installed correctly.
- Compile the example using the Make file by calling "make" (make sure that the library header:#include "headers/sensorcloud_crypto_library.h" is included)
- Initialize the library: Initialization routine requires four parameters:
 - The service's RSA private key as a PEM-encoded string
 - A public-key callback to handle the event of missing public keys
 - A symmetric-key callback to handle the event of missing symmetric data keys
 - A pointer to a SCErrorstruct that is filled by the library in case of errors

The initialization function has the following signature:

```
intsc_init(  
const char*pem_private_key,  
SCPublicKeyCallbackpublic_key_cb,  
SCSymmetricKeyCallbacksym_key_cb,  
SCError* sc_error);
```

The public-key callback must have the following signature:

```
JWK_PUBLIC_KEY scwi_public_key_callback(  
const char* key_id,  
PublicKeyTypekey_type);
```

where, The key_id is a string containing the hexadecimal representation of thepublic key's SHA-1 message digest. The key_type denotes whether the missingpublic key is an RSA key (PUBLIC_KEY_RSA) or an ECDSA key (PUBLIC_KEY_ECDSA).

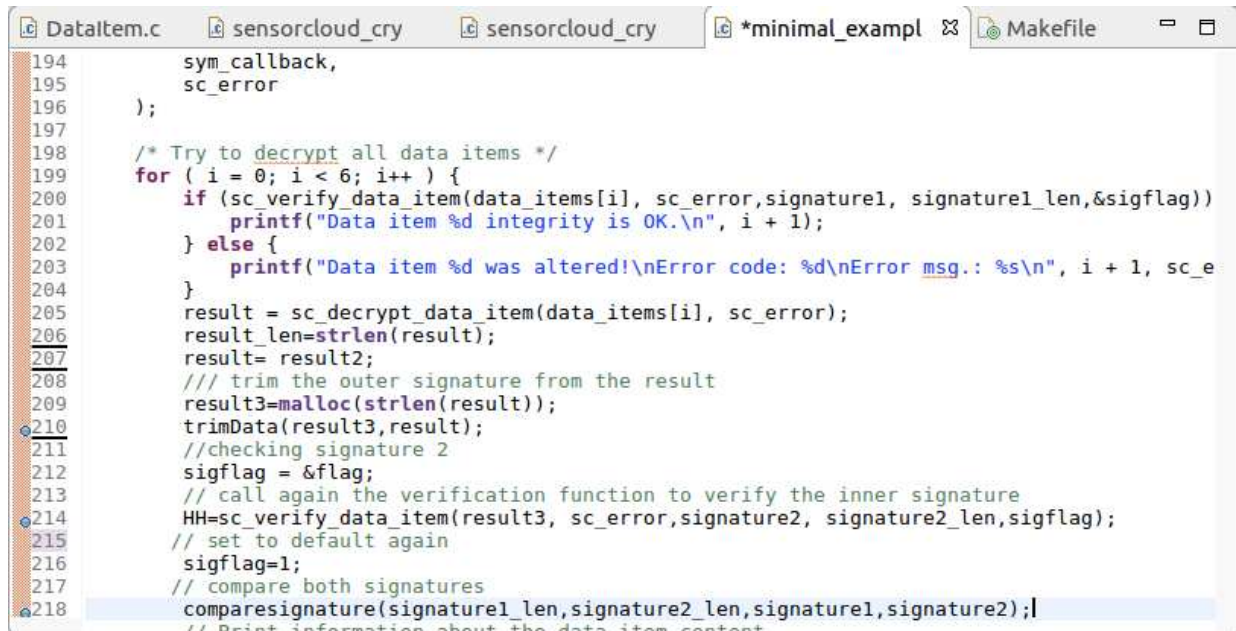
- The callback must return a JWK string containing the missing public key.

Similarly, the symmetric-key callback must have the following signature:

```
JWK_SYMMETRIC_KEY  
scwi_symmetric_key_callback(const char* key_id);
```

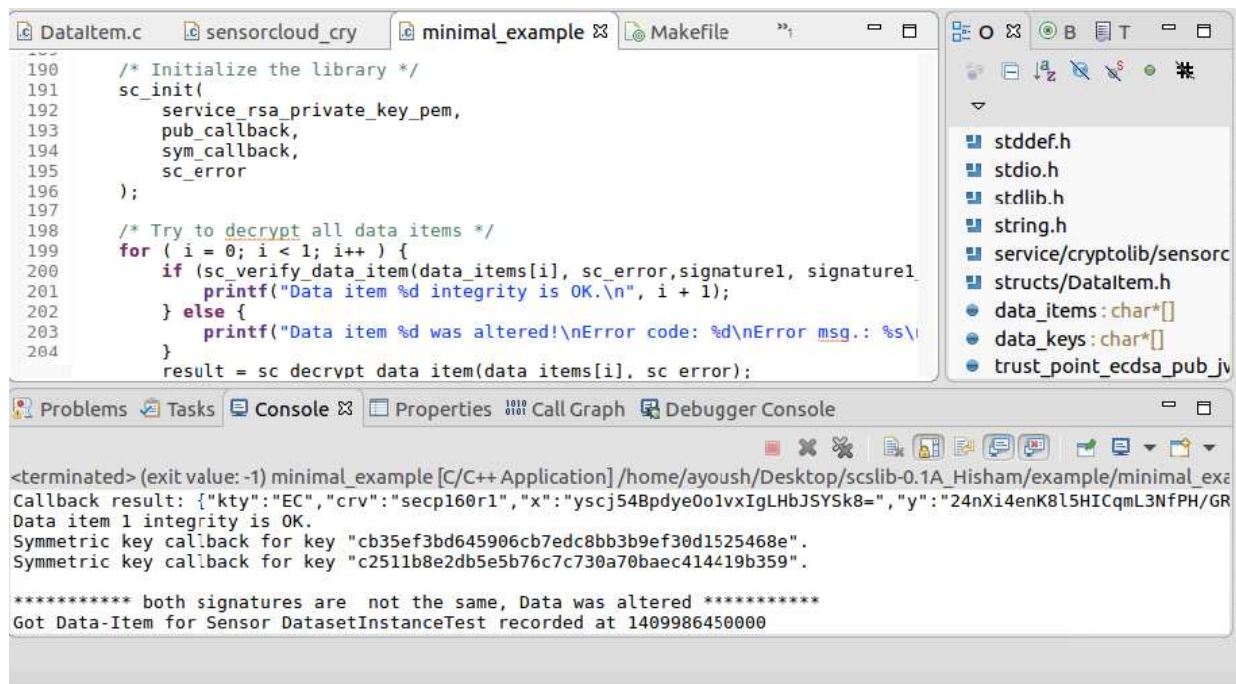
Only the key_id of the missing symmetric key has to be passed to the callback and, analogously to the previous

callback, it must return a JWK string containing the missing symmetric key.



```
194     sym_callback,  
195     sc_error  
196 );  
197  
198 /* Try to decrypt all data items */  
199 for ( i = 0; i < 6; i++ ) {  
200     if (sc_verify_data_item(data_items[i], sc_error,signature1, signature1_len,&sigflag))  
201         printf("Data item %d integrity is OK.\n", i + 1);  
202     } else {  
203         printf("Data item %d was altered!\nError code: %d\nError msg.: %s\n", i + 1, sc_e  
204     }  
205     result = sc_decrypt_data_item(data_items[i], sc_error);  
206     result_len=strlen(result);  
207     result= result2;  
208     /// trim the outer signature from the result  
209     result3=malloc(strlen(result));  
210     trimData(result3,result);  
211     //checking signature 2  
212     sigflag = &flag;  
213     // call again the verification function to verify the inner signature  
214     HH=sc_verify_data_item(result3, sc_error,signature2, signature2_len,sigflag);  
215     // set to default again  
216     sigflag=1;  
217     // compare both signatures  
218     comparesignature(signature1_len,signature2_len,signature1,signature2);  
    // Print information about the data item content
```

The results for running the minimal example are:



```
190 /* Initialize the library */  
191 sc_init(  
192     service_rsa_private_key_pem,  
193     pub_callback,  
194     sym_callback,  
195     sc_error  
196 );  
197  
198 /* Try to decrypt all data items */  
199 for ( i = 0; i < 1; i++ ) {  
200     if (sc_verify_data_item(data_items[i], sc_error,signature1, signature1  
201         printf("Data item %d integrity is OK.\n", i + 1);  
202     } else {  
203         printf("Data item %d was altered!\nError code: %d\nError msg.: %s\  
204     }  
    result = sc_decrypt_data_item(data_items[i], sc_error);
```

Problems Tasks Console Properties Call Graph Debugger Console

```
<terminated> (exit value: -1) minimal_example [C/C++ Application] /home/ayoush/Desktop/scslib-0.1A_Hisham/example/minimal_ex  
Callback result: {"kty":"EC","crv":"secp160r1","x":"yscj54Bpdy0o1vxIgLHbJSYSk8=","y":"24nXi4enK8l5HICqML3NFPH/GR  
Data item 1 integrity is OK.  
Symmetric key callback for key "cb35ef3bd645906cb7edc8bb3b9ef30d1525468e".  
Symmetric key callback for key "c2511b8e2db5e5b76c7c730a70baec414419b359".  
  
***** both signatures are not the same, Data was altered *****  
Got Data-Item for Sensor DatasetInstanceTest recorded at 1409986450000
```