Internet of Things: Security in the Keys

Sabrina Sicari
Universita' degli Studi dell'Insubria
Via G. Mazzini 5 - 21100
Varese, Italy
sabrina.sicari@uninsubria.it

Alessandra Rizzardi
Universita' degli Studi dell'Insubria
Via G. Mazzini 5 - 21100
Varese, Italy
a.rizzardi@uninsubria.it

Daniele Miorandi
U-Hopper
Via A. da Trento 8/2 - 38122
Trento, Italy
daniele.miorandi@u-hopper.com

Alberto Coen-Porisini
Universita' degli Studi dell'Insubria
Via G. Mazzini 5 - 21100
Varese, Italy
alberto.coenporisini@uninsubria.it

ABSTRACT

Security threats may hinder the large scale adoption of the emerging Internet of Things (IoT) technologies. Besides efforts have already been made in the direction of data integrity preservation, confidentiality and privacy, several issues are still open. The existing solutions are mainly based on encryption techniques, but no attention is actually paid to key management. A clever key distribution system, along with a key replacement mechanism, are essentials for assuring a secure approach. In this paper, two popular key management systems, conceived for wireless sensor networks, are integrated in a real IoT middleware and compared in order to evaluate their performance in terms of overhead, delay and robustness towards malicious attacks.

Keywords

Internet of Things; Security; Key Management; Prototype

1. INTRODUCTION

Since the last decades, Internet of Things (IoT) emerged in the form of innovative and customized services provided to individuals and businesses in order to improve their everyday experience. Such services are made available by heterogeneous technologies, which cooperate over a global network infrastructure in different application domains. Lots of critical issues raise, due to the amount of involved entities and generated information, such as scalability, security&privacy, data quality preservation [6]. A scalable infrastructure, able to deal with diverse data sources would enable the realization of an efficient and reliable IoT system. In this direction, we proposed a flexible and dynamic middleware architecture for IoT, named *NetwOrked Smart object* (NOS). NOS acts as a distributed processing and storage layer for the data collected by IoT deployments and provides a mechanism for the automatic assessment of data quality and security by means of well-defined algorithms [7]. A level of robustness is assigned to each data source regarding the following security features: integrity, confidentiality, authentication system, privacy; while, as regards data quality, the following requirements are considered: accuracy, precision, timeliness, completeness. In this way, the user will be aware of the security and quality level of the data being accessed in order to take informed decisions about their usage. Although this represents a further advancement in securing an IoT system, a fundamental aspect is still missing. In fact, most security systems are based on the use of keys for encrypting the transmitted data, in order to protect them from malicious attacks. However, also the keys may be compromised, thus hindering the correctness of the information. To cope with such an issue, NOSs should include a key management system, in order to directly deal with key distribution and replacement and improving the robustness of the adopted authentication and encryption mechanisms. In this paper, two popular key management systems by Dini et al. [3] and Di Pietro et al. [2], originally conceived for Wireless Sensor Networks (WSN), are integrated in the real NOS prototypical platform.

Q2SWinet'16, November 13-17, 2016, Malta, Malta

DOI: http://dx.doi.org/10.1145/2988272.2988280

2. RELATED WORKS

One of the main challenge, currently addressed in literature and in EU projects, is the development of an interoperable and secure platform, in order to provide services for the users. Several middleware layers have been proposed to enforce the integration and the security of devices and data within the same information network, taking into account the heterogeneity of devices and communication technologies of IoT [6]. By the authors, an attempt to provide a lightweight and flexible middleware for IoT applications is at the heart of the work reported in [7], where a prototypical implementation of NOS, able to evaluate the data provided by the IoT system both in terms of security and quality, is presented. As a step beyond [7], in this paper we aim at improving the actual NOS prototype with a robust key management system, in order to further secure the communica-

tions with data sources. To this end, the analysis of the state of the art started from the available approaches designed for WSN scenarios. Many solutions have been proposed aiming to overcome traditional pre-distribution approaches [8]. Among them, the algorithms proposed by Dini et al. [3] and Di Pietro et al. [2] gained popularity due to their efficiency in terms of resource consumptions and resilience towards malicious attacks. For such reasons we decided to integrate them within NOS platform. As regards IoT context, a limited contribution in the key management field actually exists. [5] only provides a classification of existing protocols relying on key distribution mechanisms to establish a secure communication channel among the sources. It concludes that symmetric approaches are still not the default choice for IoT. Instead, public key cryptography is likely to be increasingly recommended, provided that the associated asymmetric techniques are properly optimized. In [5] opinion, a trusted third party will take a more active role to secure IoT and to adapt to its heterogeneous nature. [1] proposes a lightweight key management protocol for e-health applications. It is based on collaboration to establish a secure end-to-end communication channel among resource constrained sources. Third parties are in charge of executing the cryptographic primitives. However, the evaluation showed a high overhead and the use of third parties and of a certificate authority heavily impacts on scalability and efficiency. [9] and [10] adopt a group key management distribution scheme for WSN in a IoT scenario in which the nodes are organized in a hierarchical structure. Note that, in our approach we do not infer about a hierarchy among the sources, since we consider each device independent from the others.

3. SYSTEM ARCHITECTURE

The nodes and the users are the principal actors in an IoT scenario: the former acts as data sources; while the latter may be individuals, or businesses, or other kinds of application, interested in accessing services provided by NOSs exploiting the IoT data. NOSs aim to act as a distributed middleware bringing the data processing closer to the sources than a centralized platform. The southbound NOS interfaces use HTTP as network protocol; they include both the handling of the transmissions by nodes and a service for source registration. In fact, NOS supports either registered sources or anonymous ones. Registered ones are associated with an identifier, and, optionally, with an encryption scheme, including the proper keys for interactions with NOSs. The management of such keys will be investigated in Sec. 4. As regards transmissions, for each incoming data, NOSs extract: the kind of data source, the communication mode, the data type, and a timestamp. Since received data are highly heterogeneous, NOS initially puts them in the *Raw Data* storage unit and, periodically, elaborates them according to a two-phase structure, which includes *Data Normalization* and *Analysis*. Then, data are uniformly represented, and enriched with relevant metadata in the form of a score for the robustness of each security and data quality property considered: data confidentiality, data integrity, source privacy and source authentication (*Security Analysis*); data accuracy, data precision, information timeliness and completeness (*Data Quality Analysis*). The rules executed for the assessment of security and quality scores are not object of this work, since the results are in [7]. The data thus processed are inserted in the *Processed Data* collection and are used for providing services by means of a publish/-subscribe mechanism, based on *Message Queue Telemetry Transport* (MQTT) protocol (northbound interfaces).

4. KEY MANAGEMENT SYSTEMS

NOSs allow the registration of the sources interested in providing data to the IoT system, as said in Sec. 3. An unique identifier is assigned to each source by NOS and, if the source wants to exchange its data in a secure way, NOS assigns the proper keys for encrypting the transmitted information. As emerged in Sec. 2, no existing solution specifically addresses the key management in an IoT platform. Therefore, we aim to integrate two popular and robust key management systems, named Dini et al. [3] and Di Pietro et al. [2], conceived for WSN in our distributed NOS.

4.1 Key Management by Dini et. al

The distribution system of the cryptographic keys presented by Dini et al. [3] responds to the requirements of dynamism and mobility of IoT, since it is not conceived for a fixed network topology, but let the nodes to dynamically join and leave the network. Also the resource constraints of the nodes are taken into account, thus reducing the network traffic and the processing operations. This approach represents a node-to-node distributed key agreement, supporting the creation of secure communication channels, which connect sequences of adjacent nodes sharing the same keys. This result is achieved by propagating the key connecting the start node and the first subsequent node to all the other nodes in the channel. Once a secure channel has been established, no hop-by-hop encryption/decryption is required. Two set of keys are stored in each node: the global and the local. The former is assigned by NOS during the registration phase and is used to legitimate the encrypted communications among the sources within the IoT system. The latters are directly assigned by NOS to a source or indirectly by a source to another one (just owning the same global key) and are used for encrypting the transmitted data. Global keys are only aimed at the propagation of the local keys, as specified in the following. Each key is denoted by a name and a value. The name K of a local key consists of two components: (i) K_{node}, which is codified in the d most significant bits of K, and is equal to the name of the NOS which generated the key; (ii) an incremental number K_{incr}, which is codified in the least significant bits of K. In a similar way, the global keys have a value and a name, which is the order number of that key in the pool of key values generated by each NOS by means of the algorithm in [4]. Keeping in mind that multiple NOSs could act in the IoT environment, each NOS N stores three tables aimed at preserving information about keys and connections with registered sources: (i) the connection table CT_N, which contains one entry for each source S connected to NOS N; more in detail, the entry $CT_{N,S}$ contains the list of the local keys K_i generated by N (K_{node} is equal to N) shared with S, which allows multiple connections $(N, S)K_i$; (ii) the global key table GT_N, including one entry for each global key generated by NOS N (K_{node} is equal to N); (iii) the local key table KT_N, containing an entry for each local key K generated by another NOS (K_{node} is not equal to N). The information included in KT_N are required in case a source begins to send its data to another NOS, which does not corresponds to the NOS to

which it was registered before. In case of only one NOS, no local key table is required. Instead, since data sources do not generate local keys by themselves, they store only two local tables: the connection table, including their active connections with NOS and other sources, and the global key table. The number of global keys stored by a source depends on the specific application domain, since a source could provide data belonging to different contexts and, therefore, register to NOS with multiple credentials. Such a case is not considered in this paper. Algorithm 1 outlines the steps of the key generation and distribution proposed by Dini et al. It also provides a mechanism for the local key replacement. The new key value has to be updated into the connection key table of each node which shares this key. This is achieved by propagating a key replacement message to all the sources directly connected to NOS. Such a procedure is periodically started by NOSs, to improve the system robustness, or in case of malicious node detection. For further detail we refer to [3]. If, otherwise, a global key has been compromised, then it should be discarded from the network and further updated. In this case, NOS will send an invalidation message containing the name of the compromised key; therefore, the sources will discard such a key and they could not use it to establish other connections (e.g., with possible malicious nodes).

4.2 Key Management by Di Pietro et. al

Di Pietro et al. [2] scheme is composed of two main phases. In the first one, the new session key is autonomously generated by each source, while in the second one the new session key is synchronized among the registered sources. In fact, the algorithm guarantees that each source generates and shares the same key. MSG denotes a message that a source wants to send, while $E_k(MSG)$ indicates the encryption algorithm E employing the key k. $E_{k^{-1}}(MSG)$ represents the decryption of MSG. The length in bits of key k is denoted by q. H and G are one-way hash functions, which do not need to be kept secret, as well as E. Each source is provided by NOS, during the registration phase, with two random seeds, *S1* and *S2*, each q bits long. Each source can store an integer counter representing the sequence number of the current session key. The final value of the key is represented by the XOR boolean function applied to the results of the hash operations on the two seeds. Algorithm 2 outlines the steps of the solution proposed by Di Pietro et al. for the key generation and distribution; while Algorithm 3 defines the key update function. There exist two different scenarios for the application of this scheme. The former requires a single central entity in charge of acts as a synchronizer for performing the re-keying; while the latter is a distributed approach, in which the sources should rely on themselves to achieve synchronization in the re-keying process. We adopt the second one, where multiple NOSs manage the configuration activities among the registered sources, which, in turn, do not directly initiate any re-keying process. This differs from the original Di Pietro et al. approach, in which each node could initiate the re-keying activity. NOS invokes re-keying commands, encrypted with the last key generated by the sources, periodically. When a source receives by NOS a message that requires to update the current session key, it first saves the current value of the key, and then updates the session key. The source must save the value of the pre-

Algorithm 1 Dini et al. Key Distribution

if $(N, S)K$ *exists in* CT_S **then**
  S sends D to N encr with K
  N decrypts D with K
else
  G = findGlobalKey(N, S)
  if (G == “) **then**
    S communicates with N through adjacent node M with connection $(S, M)K$
    S sends D to M encr with K
    if $(M, N)K$ *exists* **then**
      M sends D to N encr with K
      N decrypts D with K
      S and N update their CT
      connection $(S, M, N)K$ set
    else
      if $(M, N)K'$ *exists, where* $K != K'$ **then**
        M sends D to N with couple $K = (K_{node}, K_{incr})$ received by S encr with K'
        N sends an ack to M
        M, S and N update their CT
        connection $(S, M, N)K$ set
      else
        if $(M, N)G$ *exists* **then**
          M sends D to N with couple $K = (K_{node}, K_{incr})$ received by S encr with G
          N sends an ack to M
          M, S and N update their CT
          connection $(S, M, N)K$ set
        else
          does not exist any G shared by M and N
          M communicates with N through adjacent nodes
          while connection with N set or adjacent nodes available **do**
            execute *searchAdjacentNodes()*
          end while
          comment: if such procedure fails, S is disconnected from the network
        end if
      end if
    end if
  else
    N generates a new local key K
    S sends to N the couple $K = (K_{node}, K_{incr})$ encr with G
    S sends an ack to N
    N and S update their CT
    S can send D, encr with K, to N and can decrypt messages from N with G
  end if
end if

Algorithm 2 Di Pietro et al. Key Generation

```
    Initialization
 2: currentSession = 0
    S1' = H(S1)
 4: S2' = G(S2)
    k = S1' XOR S2'
 6: while true do
        MSG reception
 8:     if MSG.session == currentSession then
            M = E_{k^-1}(MSG)
10:     else
            M = E_{oldk^-1}(MSG)
12:     end if
        if MSG.type == updateSession then
14:         execute updateKey() see Algorithm 3
        end if
16:     continue with normal execution
    end while
```

Algorithm 3 Di Pietro et al. Key Update Function

```
    updateKey()
    S1' = H(S1)
 3: S2' = G(S2)
    k_old = k;
    k = S1' XOR S2'
 6: currentSession = currentSession + 1
```

vious key, since it could receive messages that have been previously encrypted with the old key.

5. EVALUATION AND COMPARISON

NOS has been implemented as a real prototype in a modular architecture, by means of *Node.JS*, *MongoDB* for storage management, and *Mosquitto* for MQTT mechanism. Such features allowed us to easily integrate the new modules related to the two key management systems to the previous version of NOS [7]. In our experimental setup, a NOS runs on a Raspberry Pi. To simulate the behavior in a real-world setting, it is connected to open data feeds, provided in real time from six sensors at the meteorological station in Campodenno (Trentino, Italy). The sources are distinguished in registered and no registered ones in two different scenarios. The former includes 2 registered and 4 no registered sources; while the latter includes 3 registered sources and 3 no registered ones.

5.1 Overhead

The overhead analysis takes into account three metrics: the execution time, the storage capacity and the processing effort required by the algorithms. As regards the average execution time *T(s, z)*, s is the number of sources and z represents the number of parallelizable operations to be performed:

$$T(s,z) = T_{seq} + \frac{T_c(z)}{s} + T_0(s) \quad (1)$$

Where: (i) T_{seq} is the execution time of the non-parallelizable sections of the algorithm; (ii) $\frac{T_c(z)}{s}$ is the execution time of the parallelizable ones; (iii) $T_0(s)$ represents the communication time spent among the sources. In the first scenario, s is equal to 2. In Dini et al. there is no parallelizable operations, therefore z is equal to 0. Hence: (i) T_{seq} has been evaluated to be equal to 28ms; (ii) $\frac{T_c(0)}{2}$ is equal to 0; (iii) $T_0(2)$ is 4ms. The final execution time *T(2, 0)* is 32ms. A similar result is obtained for the second scenario, with s equal to 3. Such an overhead depends on the hop number required for the key propagation from NOS towards the most distant source. Instead, regarding the storage capacity required from the sources, we have that the information stored in CT and GT tables are: (i) 2 bytes for the global key name; (ii) 8 bytes for the global key value; (iii) 1 byte for K_{incr} for each local key; (iv) in the first scenario, K_{node} is equal to 1 byte, therefore the name of the local key requires 2 bytes; while, in the second scenario, the name of the local key requires 3 bytes; (v) 8 bytes for the local key value. Summarizing, with 2 and 3 registered sources, 20 and 21 bytes of storage, respectively, are required. Concerning the dimension of the transmitted message, each packet for the initial local key generation is 10 bytes, for the first scenario, and 11 bytes for the second one; while, each packet generated by the re-keying operation requires 18 and 19 bytes, respectively. The execution of the re-keying is expected to generate a response message of 1 byte. As regards the computational overhead, NOS performs: (i) 2 concatenations to establish the name of the key; (ii) the computation of the key value. While, each source, upon receiving a new local key, has to perform: (i) 2 decryption operations to know the encrypted content of the message including the new key sent by NOS. Now we refer to the same overhead metrics for Di Pietro et al. The session key generation is a parallelizable operation, therefore: (i) T_{seq} is 4ms and 8ms for the first and second scenario, respectively; (ii) $\frac{T_c(1)}{2}$ is 3ms for both the scenarios; (iii) $T_0(2)$ is equal to 0, since no message exchange is performed. The final execution time are, then 7ms and 11ms. Concerning the storage, the 2 seeds (arbitrary long q bits) and the 2 keys of 8 bytes has to be stored. At the computational level, the required operation are: (i) 1 decryption, in order to know the encrypted content of the initialization message sent by NOS; (ii) 2 hash functions, in order to generate the key value; (iii) 1 XOR operation. We can conclude that the algorithm of Di Pietro et al. presents better perfomances with respect to Dini et al., because: (i) some operations are parallelizable; (ii) no message is exchanged among the sources; (iii) the storage required by Dini et al. is influenced by the number of registered sources, therefore it may be more affected by scalability issues then Di Pietro et al.

5.2 Delay

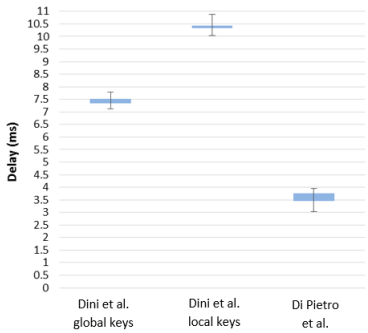

Figure 1: Delay of key generation

The delay introduced by the two algorithm has been evaluated as the time required for the key generation, propagation and update (Fig. 1). In Dini et al. the key generation is more expensive (about 7ms for the global keys, 10ms for the local keys) than in Di Pietro et al. (about 3ms), due to

the more complex operations to be executed. However, such a delay, for Dini et al., will remain unchanged with respect to the increase of the number of sources. As regards the

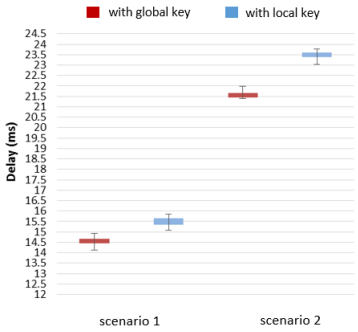

Figure 2: Delay of key propagation in Dini et al. with direct connections

key propagation, we consider only the Dini et al. approach, since Di Pietro et al. does not include any key communication among sources. The results are shown in Fig. 2 and 3concern three situations: (i) the key propagation happens with a direct connection between the sources, by means of a shared global or local key; (ii) the key propagation happens through adjacent nodes, also by means of a shared global or local key; (iii) the case of unreachable sources. Finally, the delay of key replacement has been studied. In both the algorithms, the results depend on the number of registered sources (Fig. 4). Dini et al. approach presents an higher delay with respect to Di Pietro et al. due to the time required for key propagation.

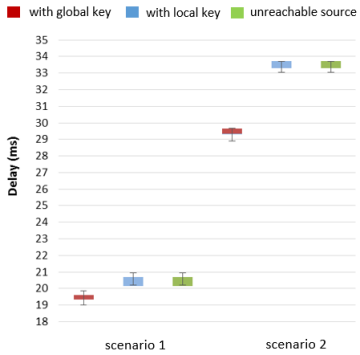

Figure 3: Delay of key propagation in Dini et al. through adjacent nodes

5.3 Robustness towards malicious attacks

In order to compare the robustness of the two key management methods towards malicious attacks, it is important to distinguish between external attacks, performed by entities acting outside the IoT secure network and internal attacks, performed by registered sources. Concerning external attacks, adopting the algorithm of Di Pietro et al., an external node cannot discover the session key, since it is not exchanged among the registered sources and NOS; the only way to succeed with such a kind of attack is to discover the seeds and the current session key. The seeds are sent in clear by NOS during the registration phase, and this represents the main drawback of Di Pietro et al. approach. Instead the session key continuously changes over the time.

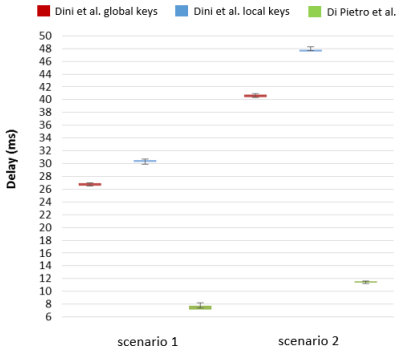

Figure 4: Delay of key replacement

More robust is the approach of Dini et al., which adopts the algorithm in [4] for the random key pre-generation process. The value of a global key cannot be easily derived from the value of another global key, but if one of them is eavesdropped, a part of the network may be compromised. To reduce the risk, Dini et al. also introduces the local keys as a second level of security. An external source could eavesdrop a packet containing the frequence of re-keying and the name of the local key. However, in order to gain the value of the local key, it should know the encryption algorithm, by compromising NOS itself. Both the approaches do not consider possible physical attacks to the sources, which would allow to directly steal the seeds, the session numbers or the global/local keys. As regards the resilience towards internal attacks, Dini et al. and Di Pietro et al. do not directly refer to key revocation due to an insider attack, and also in our implementation we do not consider this case, which could be accounted as a future extenstion.

6. CONCLUSIONS

The paper has presented an integration of two popular key management systems in the existing IoT middleware NOS. The adoption of robust key distribution and replacement algorithms has the scope to improve the resilience of the IoT system, thus enhancing the reliability of the services provided to users. The overhead, the delay and the robustness towards malicious attacks of both the solutions have been compared by means of a real NOS prototype. Future extensions include the evaluation of the presented solutions in a wider scenario in presence of multiple NOSs.

7. REFERENCES

[1] M. R. Abdmeziem and D. Tandjaoui. An end-to-end secure key management protocol for e-health applications. *Computers&Electrical Engineering*, 44, 2015.

[2] R. Di Pietro, L. Mancini, and S. Jajodia. Providing secrecy in key management protocols for large wireless sensors networks. *Ad Hoc Netw.*, 1.

[3] G. Dini and L. Lopriore. Key propagation in wireless sensor networks. *Computers&Electrical Engineering*, 41, 2015.

[4] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *9th ACM Conference on Computer and Communications Security*, 2002.

[5] K. T. Nguyen, M. Laurent, and N. Oualha. Survey on secure communication protocols for the internet of things. *Ad Hoc Netw.*, 32, 2015.

[6] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Comp. Netw.*, 76, 2015.

[7] S. Sicari, A. Rizzardi, D. Miorandi, C. Cappiello, and A. Coen-Porisini. A secure and quality-aware prototypical architecture for the internet of things. *Information Systems*, 58, 2016.

[8] M. A. Simplício, P. S. Barreto, C. B. Margi, and T. C. Carvalho. A survey on key management mechanisms for distributed wireless sensor networks. *Comp. Netw.*, 54.

[9] L. Veltri, S. Cirani, S. Busanelli, and G. Ferrari. A novel batch-based group key management protocol applied to the internet of things. *Ad Hoc Netw.*, 11.

[10] H. Yu, J. He, T. Zhang, and P. Xiao. A group key distribution scheme for wireless sensor networks in the internet of things scenario. *International Journal of Distributed Sensor Networks*, 2012, 2012.