Design of Key Management Protocols for Internet of Things

Cungang Yang and Celia Li
(Corresponding author: Cungang Yang)
Department of Electrical and Computer Engineering, Ryerson University
350 Victoria Street, Toronto, Canada
(Email: cungang@gmail.com)
(Received June 2, 2019; Revised and Accepted Dec. 16, 2019; First Online Feb. 28, 2020)

Abstract

The Internet of Things (IoT) provides transparent and seamless incorporation of heterogeneous and different end systems. It has been widely used in many applications such as smart homes. However, people may resist the IOT as long as there is no public confidence that it will not cause any serious threats to their privacy. Effective secure key management for things authentication is the prerequisite of security operations. In this paper, we present an interactive key management protocol and a non-interactive key management protocol to minimize the communication cost of the things. The security analysis, numerical analysis and simulation results show that the proposed schemes are efficient and resilient to various types of attacks.

Keywords: IoT Security; Key Management; Ticket Based Authentication

1 Introduction

The Internet of Things (IoT) comprises of billions of devices that can sense, communicate, compute and potentially actuate [1, 3, 4]. IoT involves accessing, monitoring and controlling various sensors and devices over the internet. A great example of the IoT application is smart homes. Household systems like smart smoke-alarms, air quality sensors, smart doorbells, and home monitoring devices can now communicate with smart watches, and activity trackers. After an activity tracker assessed your sleep – determining when you are in light sleep – it can tell your alarm clock to go off. Your alarm clock in unison with your phone will check the weather – just before you wake up (based on your preference and sleep cycle) and tell air conditioners in your car and your home to change the temperature accordingly. Navigation apps on your smart phone – after gathering information from your weather app – can predict how the weather will affect traffic congestion, and plan a route to your work. As the communication between IoT devices may include sensitive and critical data, the security requirements for any IoT-based system are high. To set up a security channel between different devices such as an air quality sensor and a smart watch, a number of security operations (authentication, authorisation, and data integrity) are needed [8]. Since key management is the prerequisite of these security operations, the motivation of this research is thus to develop pairwise key generation and rekeying schemes for IoT devices [6]. Generally, the design of IoT key management protocols has the following security requirements.

1) Secrecy and authenticity: The protocol needs to guarantee that only the intended party learn the key management and that this key is unique and fresh. Security and authenticity need to be protected against attacks such as impersonation, DoS, *etc.* Another security goal is to minimize the negative effects of a comprised key. Keys maybe exposed regardless of the security of the key management protocol that generates them, *e.g.*, by break-ins to a device, poor secure storage for keys, *etc.* Mechanisms like independence between different keys in a system, frequently refreshment, and perfect forward secrecy, as discussed below, address this goal.

2) Key refreshment: The key management protocol must provide automatic mechanism to periodically refresh keys: when a cryptographic key is used actively, the amount of data encrypted with it grows and it becomes easier to perform attacks on the encryption algorithm. To prevent breaking of the security, every key has to be replaced after a time interval.

3) Perfect forward secrecy: Perfect forward secrecy (denoted PFS) refers to the property that disclosure of long-term key does not comprise the session keys from earlier runs. If one encryption key is compromised, only the data encrypted by that specific key is compromised. Some cryptosystems allow session keys to be derived from long term keys, so that if the long term key is compromised, an attacker might

have enough information to figure out session keys and/or decrypt data encrypted using those keys.

4) Key Separation: Different cryptographic functions should use different and independent keys (namely the exposure of one key should not compromise the other). This applies to different functions used in the key exchange protocol as well as the cryptographic functions applied to data during the subsequently sessions. In particular, one has to careful not to reuse the session key for different functions.

Also, new key management protocols should fit the features of IoT and avoid the challenges such as limited bandwidth and vulnerable to attacks. So far, the research on the secure key issues of the IoT is focused on homogenous and heterogeneous wireless sensor networks. Perrig [7] presented a suite of security protocols optimized for sensor networks that they called 'SPINS'. The suite is built upon two secure building blocks, each performing individual required work: SNEP and TESLA. SNEP offers data confidentiality, authentication, integrity, and freshness, while TESLA offers broadcast data authentication. The TESLA protocol, used on regular networks, is modified as a SPINS for use in resource-constrained wireless sensor networks. Disadvantages of this scheme include TESLA overhead from releasing keys after a certain delay and possible message delay. A non-interactive key management approach is introduced in the article "self-certified keys - concepts and applications" [11]. This scheme allows the computation of a session key in a non-interactive manner. Non-interactive key management protocol involve minimal interaction among the nodes of the network which requires global clock. In a key pre-distribution scheme [12,15,16], some keys are preloaded into each sensor before sensor deployment. After deployment, sensor nodes undergo a discovery process to set up shared keys for secure communications. This scheme ensures to some probability that any two sensor nodes can communicate using a pairwise key. This scheme does not, however, ensure that two nodes always are able to compute a pairwise key to use for secure communication. The key management scheme of 802.11i in WLAN is helpful to develop key management protocols in IoT. However, 802.11i has the following weaknesses:

1) The authentication server (AS) works as key distribution center that may not be reachable.

2) More communication costs on the network due to the involvement of the AS.

3) Single point failure of AS.

4) 802.11s does not support Perfect Forward Secrecy. If the primary master key (PMK) is exposed, the session keys will be compromised.

5) The 4-way handshake is vulnerable to DOS attack.

Our work is based on the key management scheme of 802.11i. The motivation of our work is trying to enhance 802.11i with new interactive and non-interactive key management protocols whose design should be able to fit the features of IoT and solve the weaknesses of the key management scheme in 802.11i. The contribution of this paper is developing pairwise key generation and rekey schemes for IoT devices. In particular, we bring in a novel interactive key management protocol which is resilient to attacks and save communication cost. Moreover, we propose a secure non-interactive key management protocol which further reduces the communication cost close to zero. The rest of the paper is organized as follows. Section 2 presents our proposed interactive key management scheme. The non-interactive key management scheme is explained in Section 3. The numerical analysis on the performance of the interactive and non-interactive key management schemes are explained in Section 4. Finally, Section 5 concludes the paper.

2 An Interactive Key Management Scheme

The interactive key management scheme between device A and device S is comprised of two phases that is shown in Figure 1. Notations used in the rest of the paper is summarized in Table 1. In Phase 1, A requests to communicate with S. They mutually authenticate each other with a Ticket-based authentication protocol and generate a Pairwise Master Secret (PMK). In Phase 2, following the establishment of the PMK, a session key rekey protocol is executed to confirm the existence of the PMK and the liveliness of the peers; the session key rekey protocol is resilient to DoS attack and supports Perfect Forward Secrecy (denoted PFS) which refers to the property that disclosure of long-term PMK does not comprise the session keys from earlier runs.

Table 1: Notations

Notation	Description
I_x	ID of node X
P_x	Public key of x
T_x	Ticket issued to x
T_{exp}	Expiry date of a ticket
N_x	Nonce of node x
Sig_x	Digital signature of node x
D_x	Domain name of x
$E_{pub_A}(m)$	Encrypt m using A's public key
V_k	Message authentication code resulting from the application of a MAC key k on a message m

2.1 Phase 1: Ticket-based Authentication and PMK Generation

Tickets are used to establish the trust relationships among entities. For example, devise A will trust devise S if the ticket of S is valid and issued by the ticket agent it trusts. A ticket agent is defined as an authority who issues and manages various types of tickets and can be trusted by various entities in IoT. Before deployment of IoT devices, the network operator, denoted by OP, requests tickets from a ticket agent, one per device, and preinstall the ticket for each node. The OP is also responsible for requesting and distributing new tickets before the current tickets expire.

Following is the structure of a ticket for device R:

$$T_R = \{I_R, I_A, T_{exp}, P_R, D_R, Sig_A\}$$

- T_R: Ticket issued by ticket agent I_A.
- I_R: ID number of the device R that is given this ticket.
- I_A: ID number of the ticket agent who issued ticket T_R to I_R.
- T_{exp}: Expiry date and time of ticket T_R.
- P_R: Public key of I_R, which is used to verify the signature of messages sent by I_R.
- D_R : Domain name of the network that the device is located.
- Sig_A: Digital signature of ticket agent I_A.

With the design of tickets in the design of the key management protocol, the key generation and negotiation of IoT devises do not need the involvement of the third party, such as the key distributed center or authentication server. The messages exchange only between the pair of devices dramatically reduce the communication cost of the network. Following is the messages to be exchanged according to the order of the protocol as shown in Phase 1 of Figure 1.

1) Device A broadcasts its ticket periodically. This message allows device S to detect its presence in order to join the negotiation process. S verifies the digital signature of the ticket agent who issued A's ticket T_A using the ticket agent's public key. We assume that the tickets of all nodes are issued by the same ticket agent and the public key of the agent has been preinstalled in each node. S verifies the domain name of the ticket and ensure that the device it associated is from the same network. S also verifies other information in the ticket such as the ID of the ticket agent and the ticket expiry date.

2) If the above verifications are successful, S extracts A's public key from T_A and generates a message MS which contains S's ticket T_S and two nonce N_{S1} and N_{S2}. S then encrypts the message using the A's public key and sends the encrypted message to the A. Upon receiving the message, A decrypts it using its private key, and verifies the digital signature of the ticket agent who issued the ticket T_S. A verifies the domain name of the ticket and ensure that the device it associated is from the same network. A then verifies other information recorded in ticket T_S such as the ID of the ticket agent who issued T_S and the ticket expiry date.

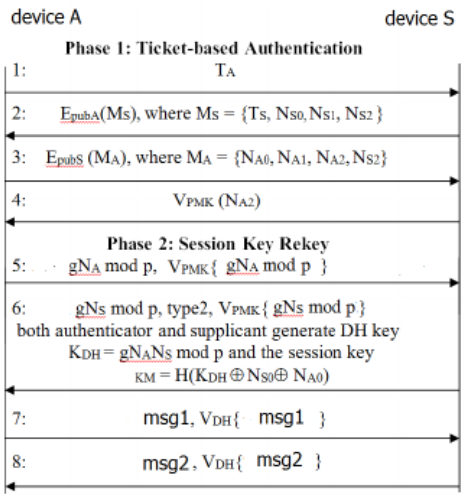

Figure 1: Interactive key management protocol

3) If the above verifications succeed, A retrieves S's public key from ticket T_S, and generates a message MA containing random numbers N_{A1}, N_{A2} and N_{S2}. A then encrypts message MA using S's public key, and sends the encrypted message to S. S will decrypt the message using its private key to retrieve N_{A1}, N_{A2} and N_{S2}. A authenticates S if N_{S2} is correct. Both devices A and S then calculate their shared PMK by applying a hash function H to the message $N_{S1}||N_{A1}$, and N_{S1} and N_{A1} are the random numbers generated in Steps 2 and 3 above. That is, PMK = H($N_{S1}||N_{A1}$).

4) S then uses the key PMK and applies a (predetermined) MAC algorithm on N_{A2} to produce a message authentication code $V_{PMK}(N_{A2})$, which S then sends to A. Upon receiving this message authentication code, A performs the same computation as S just did to produce a message authentication code $V'_{PMK}(N_{A2})$. If $V'_{PMK}(N_{A2}) = V_{PMK}(N_{A2})$, then A has successfully authenticated S, because only S has the knowledge of the shared key PMK and N_{A2} .

In Phase 1, device A and S exchange their tickets and verify the validity of each other's tickets. The trust relationship between A and S from the same network is based on their exchanged tickets which should be issued by a same ticket agent. The results of the protocol are mutual authentication of the pair and the generation of

a shared PMK key which is the basis for the following process to create the session key for data confidentiality.

2.2 Session Key Rekey

The session key rekey protocol is shown in Phase 2 of Figure 1. Here, we assume g and p are public information known by both A and S.

1) In the first message, $g^{N_A} modp$, $V_{PMK} g^{N_A} modp$. Device A generates a random number N_A and calculate the MAC value of g^{N_A} mod p with the PMK key. Device S authenticates A.

2) S generates a random number N_S, $g^{N_S} modp$ and calculate the MAC value of $g^{N_S} modp$ with the PMK key. With this step, A authenticates S. Both A and S then calculate DH key $K_{DH} = g^{N_A N_S} modp$ and their shared session Key KM by applying a hash function H to the message $K_{DH} \oplus N_{S0} \oplus N_{A0}$ where N_{S0} and N_{A0} are the random numbers generated in Steps (1) and (2). That is, $KM = H(K_{DH} \oplus N_{S0} \oplus N_{A0})$.

3) A sends an acknowledgement message, msg1, $V_{KM} msg1$, to S. S authenticates A.

4) S sends an acknowledgement message, msg2, $V_{KM} msg2$, to A. A authenticates S.

The main reason of the DoS attack on the original 4-way handshake of 802.11i is due to the plaintext of message 1. In the new session key rekey protocol, we have generated a shared key to protect the first message so as to avoid blocking and the legitimate authenticator and the supplicant is not necessary to allocate memory to store all the received nonces and the derived PTKs. The interactive key management protocol is resilient to DoS. First, the attacker cannot impersonate device A and forge message 1 since he does not know the PMK and cannot generate the proper MAC value. Any change in the original message 1 cannot be successfully verified by S. Second, the PTK inconsistency in 802.11i 4-way handshake will not happy in the proposed interactive key management protocol. The nonce values of DH key $KDH = g^{N_A N_S} modp$ and the session key, $H(K_{DH} \oplus N_{S0} \oplus N_{A0})$ are all secret values. They both hide from the attackers. Without the knowledge of N_A, N_S, N_{S0}, and N_{A0}, the attacker is not possible to modify the session key or DH key. Thus, the session key inconsistency problem occurred in 802.11i 4-way handshake will not occur in our proposed interactive key management protocol. We consider PMK as the long term key and session key as a short term key. Within the lifetime of PMK, multiple session keys should be updated. our protocol supports PFS. In the scheme, a DH key is introduced and located between the PMK and the session key. PMK key securely transfer the public information g^{NA} mod p and g^{NS}mod p for mutually authenticity of A and S while hide their secret value N_A and N_S accordingly. The knowledge of PMK does not help to derive DH key $g^{N_A N_S}$ mod p because the secret values N_A or N_S are private information of A and S. Even the PMK is exposed, the attacker cannot derive the DH key that is current used, previously used or will be used by valid device A and S. In addition, DH key is the basis to retrieve the session key. For example, the session key is $H(KDH \oplus N_{S0} \oplus N_{A0})$. Hence, the attacker cannot compromise the session keys in case PMK is exposed.

3 The Non-interactive Key Management Protocol (Non-INT)

3.1 Overview

The authenticity of public keys in a public cryptosystem is gained in two different ways: either it is verified by its certificate, or it is verified implicitly during the use of the keys. The latter is introduced by Girault as self-certified keys [2]. Self-certified keys are not verified until it is used for cryptographic function such as signature verification. Public keys of each node are verified without the aid of its public key certificate or an online Certificate Authority (CA) [11]. The concept of self-certified keys is employed in this paper due to its simple non-interactive rekey mechanism. In this section, by coupling the ticket-based technique with the self-certified keys, we obtain a fully non-interactive key management protocol for IoT. In contrast with prior work [11], our techniques for session key update do not require any interaction and do not involve any reliable broadcast communications among devices. Here, we present a new scheme that offers both device A and S to compute or rekey a session key in a non-interactive manner. We achieve this result by using the user-controlled key progression. Compare with interactive key management schemes, the new non-interactive approach further reduce the communication cost of the session key generation and rekey to zero or close to zero.

3.2 Bootstrapping

The network is initialized by the network operator OP. OP chooses large primes p and q with $q|(p-1)$ (q is a prime actor of p-1). OP chooses a random number $K_A \in Zq*$ with order q and generates its (public, private) key pair (y_Z, x_Z). We assume that the public key y_Z, p, q and g are preinstalled to every node of the network. To issue the private key for a device A with identifier ID_A, OP computes the signature parameter $r_A = g^{k_A}(modp)$ and $s_A = x_Z \times h(ID_A, r_A) + k_A(modq)$. r_A is called the guarantee and $x_A = s_A$ is its private key. The public key of A can be computed by any node that has y_Z, ID_A and r_A using the following equation $y_A = y_Z^{h(ID_A, r_A)} \times r_A(modp)$. We denote this initial key pair as $(x_{A,0}, y_{A,0})$. We assume that each node has installed the initial pair of public and private key issued by the OP.

3.3 Self-Certification

The non-interactive key management protocol is comprised of two phases. Phase 1 in Figure 2 is in charge of the PMK key generation and rekey which is interactive. Phase 2 discuss the session key generation and rekey which is non-interactive. For the original non-interactive scheme, for each PMK update, the device A and S need to exchange $r_{A,t} = g^{K_A}, tmodp$ and $r_{S,t} = g^{Ks}, tmodp$ where $1 \leq t \leq n$. This scheme waste valuable bandwidth because each $r_{A,t}$ or $r_{S,t}$ could be as large as 2048 bits or 3072 bits and number n is uncertain since the number of session keys update within a PMK rekey interval is unknown.

Phase 1. Ticket-based authentication and PMK generation.
In Phase 1 of the non-interactive key management protocol.

1) First message T_A includes R and $V_{PMK}R$. Device A generates a random number R and calculate the MAC value of R with the PMK key. Device S authenticates A because only A has the shared PMK to generate the MAC value.
2) Upon receiving the second message, A decrypts it using its private key, and verifies the digital signature of the ticket agent who issued the ticket T_S using the ticket agent's public key. A receives three random numbers N_{S0}, N_{S1}, N_{S2} and $g^{N_S} modp$ where N_S is the secret value generated and hold by S, A verifies other information of ticket T_S such as the ID of the ticket agent who issued T_S and the ticket expiry date.
3) If the above verifications succeed, A retrieves S's public key from ticket T_S, and generates a messageM_A containing $g^{N_A} modp, l, \delta T, F$ and three random numbers $N_{A0}, N_{A1} and N_{A2}$. N_A is the secret value generated and hold by A. A then encrypts message MA using S's public key, and sends the encrypted message to S. S will decrypt the message using its private key and retrieve $g_{N_A} modp$, the length of the one-way hash chain l, session key progression interval δT, lifetime of the PMK F and three random numbers $N_{A0}, N_{A1} and N_{A2}$. Again, S authenticates A in this message.
4) In message 4, S verified A's authenticity. Finally, both A and S calculate the DH key as $K_{DH} = g^{N_A N_S}$ mod p and derive the initial $V_{A,1}$ and $V_{S,1}$ value as $H(K_{DH} \oplus N_{A0} \oplus N_{S0})$. In Phase 1, whenever generate or rekey the PMK, A and S generate their new secret values N_A and N_S which are the basis to derive new session keys in the second phase. After Phase 1, both A and S know their common secret value V as well.

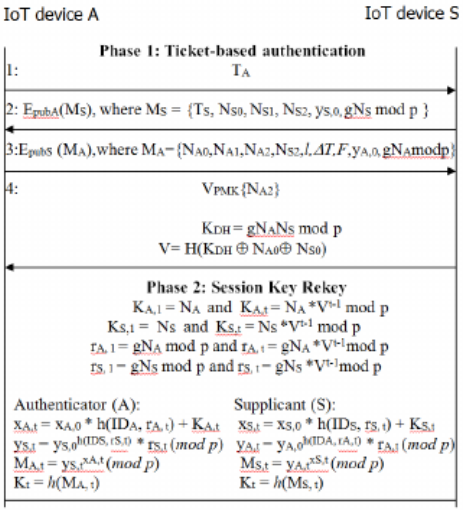

Figure 2: Non-interactive key management protocol

Phase 2. Session key generation and rekey.
$x_{A,0}, x_{S,0}, y_{S,0}$ and $y_{A,0}$ are assigned by the OP. $y_{S,0}$ and $y_{A,0}$ are exchanged by A and S with the second and third messages of Phase 1. We define that

$$\begin{aligned} K_{A,t} &= K_{A,t-1} \times V \bmod p \\ &= K_{A,t-1} \times H(K_{DH} \oplus N_{A0} \oplus N_{S0}) \bmod p \\ &= K_{A,1} \times (H(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \bmod p \\ K_{A,1} &= N_A. \end{aligned}$$

Thus,

$$\begin{aligned} K_{A,t} &= N_A \times H(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \bmod p \\ &= N_A \times V_{t-1} \bmod p. \\ r_{A,1} &= g_{A,1}^{K} \bmod p \\ &= g^{N_A} \bmod p. \\ r_{A,t} &= g_{A,t}^{K} \bmod p \\ &= g^{N_A} \times (H(K_{DH} \oplus N_{A0} \oplus N_{S0}))^{t-1} \bmod p \\ &= g^{N_A} \times V_{t-1} \bmod p. \end{aligned}$$

For devise S,

$$\begin{aligned} K_{S,1} &= N_S \\ K_{S,t} &= N_S \times V_{t-1} \bmod p, \\ r_{S,1} &= g^{N_S} \bmod p \\ r_{S,t} &= g^{S} \times V_{t-1} \bmod p. \end{aligned}$$

In Phase 2, A keeps its secret value $K_{A,1} = N_A$ and derives $K_{A,t} = N_A \times V_{t-1} modp$ for the following sessions. S keeps $K_{S,1} = N_S$ and derives $K_{S,t} = N_S \times V_{t-1} modp$ for the following sessions. On the other hand, to derive the public key of the S, A needs to know $r_{S,1}$ and $r_{S,t}$. $r_{S,1} = g^{N_S} modp$ is transferred to A in message 2 of layer 1 while $r_{S,t} = g_{N_S} \times V_{t-1} modp$ can be derived for each session because A know g^{N_S} and V. Each r value we derived will be $\in Zq*$ because q is a prime and all r value are modular p and its value must be in Zq*.

The initial scheme [11] is not a pure non-interactive key management scheme because in their approach the set of $r_{A,t} = g_{V_t} modp$ is shared through message exchange. Compare with the scheme, our protocol allows A and S to generate the $r_{A,t}$ by themselves, and thus no message exchange are involved.

3.4 Security Analysis

For our proposed scheme, the security of the $V_{A,t}$ values depends on the public key algorithm we used in Phase 1 which is safe. The non-interactive has no PFS problem because the PMK has no relationship with the values of $V_{A,t}$ and $V_{S,t}$. If the PMK exposed, it will not compromise the session key.

1) Key security. In the non-interactive key management protocol, the security of the session rekey procedure of Phase 2 depends on the Schnorr signature scheme whose security is based on the intractability of discrete logarithm problems. The Schnorr signature scheme has been provably secure in a random oracle model [5, 13]. To derive the value of the session key, the attacker has to figure out $x_{A,t}$ and $y_{S,t}$.

$$\begin{aligned} x_{A,t} &= x_{A,0} \times h(ID_A, r_{A,t}) + K_{A,t} \\ &= x_{A,0} \times h(ID_A, g^{N_A} \times V_{t-1} modp) \\ &\quad + K_{A,t} \bmod p \\ &= x_{A,0} \times h(ID_A, g^{N_A} \times V_{t-1} modp) \\ &\quad + N_A \times V_{t-1} \bmod p. \\ y_{S,t} &= y_{S,0}^{h(ID_S, r_{S,t})} \times r_{S,t} \bmod p \\ &= y_{S,0}^{h(ID_S, g^{N_S})} \times V_{t-1} \bmod p) \times r_{S,t} \bmod p \\ &= y_{S,0}^{h(ID_S, g^{N_S})} \times V_{t-1} \bmod p) \times g^{N_S} \\ &\quad \times V_{t-1} \bmod p, \end{aligned}$$

where only the ID of A and S, p and g are public known. Other parameters are hiding from the attackers. Thus the session keys cannot be disclosed to attackers.

2) Key refreshment. For the non-interactive key management protocol, the update of PMK is carried out in Phase 1 while the session key rekey is automatically implemented by device A and S. Whenever the session key needs rekeying, the Phase 2 of each protocol will be carried out.

3) Perfect forward secrecy. The only value in Phase 1 relating to the generation of session key is V. $V = H(K_{DH} \oplus N_{A0} \oplus N_{S0})$. If the PMK is exposed, it cannot derive DH key. Thus, we can say that the attacker cannot compromise the session key if PMK is exposed.

4) Key separation.

   a. PMK and Session key: The PFS analysis shows that PMK is independent from the session key. That is, if PMK is exposed, the session key will not be compromised. Due to the same reason, if a session key is exposed, the PMK cannot be compromised either.

   b. PMK and DH key: In the non-interactive key management protocol, DH key $K_{DH} = g^{N^A N^S} modp$, the N^A and N^S are secret random numbers that only known by the authenticator and supplicant. The PMK and session key are independent: if PMK is exposed, it does not help to figure out the DH key. On the other hand, if DH key is exposed, the PMK will not be compromised.

   c. DH and Session key: The session key $K_t = h(MA, t) = y_{S,t}^{x-A,t} modp$. To derive the session key, we have to know $x_{A,t}$ and $y_{S,t}$

$$\begin{aligned} x_{A,t} &= x_{A,0} \times h(ID_A, r_{A,t}) + K_{A,t} \\ &= x_{A,0} \times h(ID_A, g^{N_A} \times V_{t-1} \bmod p) \\ &\quad + K_{A,t} \bmod p \\ &= x_{A,0} \times h(ID_A, g^{N_A} \times V_{t-1} \bmod p) \\ &\quad + N_A \times V_{t-1} \bmod p \\ &= x_{A,0} \times h(ID_A, g^{N_A} \\ &\quad \times h(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \bmod p) \\ &\quad + N_A \times h(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \\ &\quad \bmod p \\ y_{S,t} &= y_{S,0}^{h(ID_S, r_{S,t})} \times r_{S,t} \bmod p \\ &= y_{S,0}^{h}(ID_S, g^{N_S} \times V^{t-1} \bmod p) \times r_{S,t} \\ &\quad \bmod p \\ &= y_{S,0}^{h}(ID_S, g^{N_S} \times V^{t-1} \bmod p) \times g^{N_S} \\ &\quad \times V_{t-1} \bmod p \\ &= y_{S,0}^{h}(ID_S, g^{N_S} \\ &\quad \times h(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \bmod p \\ &\quad \times g_{N_S} \times h(K_{DH} \oplus N_{A0} \oplus N_{S0})^{t-1} \\ &\quad \bmod p. \end{aligned}$$

If DH key is exposed, the session key of non-interactive protocol cannot be compromised since only g, p, KDH and IDs of authenticator and supplicant are know. Other parameters are hiding from the attackers. Due to the same reason, if the session key is exposed, the attacker still cannot derive the DH key.

4 Performance Analysis and Simulation

4.1 Numerical Analysis

We compare our proposed interactive key management protocol (INT), and the non-interactive protocol (Non-INT) with the EAP-TLS and 4-way handshake protocol.

We choose EAP-TLS and 4-way handshake protocol for comparison because EAP-TLS and 4-way handshake is the authentication protocol in IEEE 802.11i. 4-way handshake protocol is vulnerable to DoS attack while our proposed protocols do not have. The performance is measured in terms of Latency of the key generation protocol, which is defined as the summation of the computation cost and communication cost.

- Computation costs, which are the latencies (in milliseconds) incurred by the security operations such as encryption, decryption and hashing [9];
- Communication costs, which indicate the number of messages exchanged between the neighbouring devices to complete an key generation session.

Computation costs. Table 2 lists the security operations, the current state-of-the-art algorithms implementing the operations, and the computation time each of these algorithms incurs. Since the encryption operation of RSA is a modular exponentiation, we assume that the cost of modular exponentiation is the same as that of RSA encryption. The original EAP-TLS and 4-way handshake protocol performs one public-key encryption, one public-key decryption, one signature generation, three signature verifications, five MAC operation and two hash function (assuming that A and S compute the MAC key KMAC in parallel). The fourth column of Table 2 records the above numbers of operations. By multiplying the computation cost of each operation (from the third column) and the number of times it is executed, and summing up the costs of all operations the EAP-TLS and 4-way handshake protocol performs, we obtain a total computation cost of 97.9645ms, as shown in the third last row of the fourth column.

Similarly, the fifth and sixth columns of Table 3 list the numbers of security operations the proposed INT and non-INT perform, respectively. Applying similar calculations as above, we obtain the computation costs of the proposed INT and non-interactive protocol, which are 108.09ms and 110.94ms, respectively. The Non-INT protocol includes an interactive PMK generation and a non-interactive session key generation. The latency of PMK and session key generation in non-INT protocol includes two times Epub, two times Dpub, two times Vsig, one time MAC and five times modular exponentiation operations. Two devices pre-compute their session keys before the session key is expired. Thus, its computation cost for the latency of session key generation in non-interactive protocol is zero.

Communication costs. For the PMK generation, Table 2 lists the number of messages involved in each of the three protocols we compare. The proposed INT and Non-INT require less messages to be exchanged than EAP-TLS and 4-way handshake. For the session key generation, Table 3 lists the number of messages involved in each of the three protocols we compare. The proposed INT has the same number of messages to be exchanged as EAP-TLS and 4-way handshake. There is no message exchange between the two devices to negotiate session key in the non-interactive protocol, and thus their communication cost is zero. In summary, considering both computation and communication costs, the latency of EAP-TLS, INT and Non-INT are 385.16ms, 327.77ms and 182.74ms, respectively.

4.2 Simulation Results

We further evaluate and compare the performance of EAP-TLS, INT and Non-INT protocols under realistic network settings using simulations. The 600m x 600m network has one home device, which is placed in the center area of the square. We assume a number of neighbouring devices could directly communicate with the home device to illustrate the overhead of the key generation approach used by EAP-TLS, INT and Non-INP. We varied the number of neighbors from 1 to 30. Each data point in the graphs is the average of 10 runs using different random seeds. The graphs are plotted with a confidence interval of 95%. We conducted two experiments as function of:

1) Number of neighboring devices: We measure the key generation latency as function of the number of the neighboring devices. We assume that up to 30 neighboring devices implement the EPH-TLS, INT or Non-INT protocol with the home device simultaneously. We calculate the average key generation delay, averaged over all neighbors participating in the experiment. We also keep track of the maximum key generation delay, the maximum value among all neighbors of the home device. The messages of the key generation protocols may get lost. We measure the success rate of key distribution for 10 neighbors. The success rate is defined as follows: if the home device has m neighbors and we consider eight messages of INT as an example, the number of key generation messages for all neighbor's key generation request is m * 8. Assume each experiment run 10 times with different seeds, the total messages regarding to a client's request is 10 * m * 8. If the simulation result shows that s messages are lost, the success rate of m neighbors is (10 * m * 8 - s)/ (10 * m * 8).

2) Background traffic load: We calculate the average and maximum key generation latency of 10, 20 and 30 neighbors as a function of background traffic. The data rate for both scenarios is varied from 10 Mbits/s to 50 Mbits/s. Data rate is 0 means that there is no background traffic. We also measure the success rate of key generation messages as the function of background traffic. The data rates various from 10Mbits/s to 50MBits/s. Here, we assume the home

Table 2: Cost of PMK

Operations	Algorithm	Time(ms)	EAP-TLS	INT	Non-INT
Epub	RSA	1.42	1	2	2
Dpub	RSA	33.3	1	2	2
Gsig	ECDSA	11.6	1	0	0
Vsig	ECDSA	17.2	3	2	2
MAC	HMAC	0.0015	5	5	1
Hashing	SHA-1	0.009	2	1	0
Modular Exponentiation		1.42	0	3	5
Total computational cost			97.9645	108.09	110.94
# of messages			12	8	4
Latency of PMK key			313.36	251.69	182.74

Table 3: Cost of session key generation

Operations	Algorithm	Time(ms)	4-way Handshake	INT	Non-INT
MAC	HMAC	0.0015	3	4	0
Hashing	SHA-1	0.009	0	1	0
Modular Exponentiation		1.42	0	3	0
Total computational cost			0.0045	4.275	0
# of messages			4	4	0
Latency of session key			71.8	76.08	0

device has 10 neighbors. Following is a detailed discussion of the experimental results.

Experiment 1. Function of number of neighboring BMAPs. The graph in Figure 3 and Figure 4 show the average latency and maximum latency as function of the home BMAP's neighbors. As the number of neighboring devices increases from 1 to 30, the average latency of EAP-TLS, INT and Non-INT increases as expected, by approximately 69.7%, 81.1% and 76.3% respectively. The maximum latency of the protocols increase by approximately 93.1%, 89.5% and 98.7%. More clients imply more key distribution requests to be processed by the home device, and more channel contention around the home device, resulting in longer delay Figure 5 shows the success rate as the function of neighbors. According to the formula we provided in section IV, the success rate of key distribution messages of 10 neighboring devices in EAP-TLS, INT and Non-INT are at the range of 98.3% and 99.6%. We observe that the number of neighboring devices does not have a big impact on its success rate, which is a positive attribute of the key generation scheme.

Experiment 2. Function of background traffic load We examine how background traffic may affect the average latency and maximum latency if 10 neighboring devices request key generation

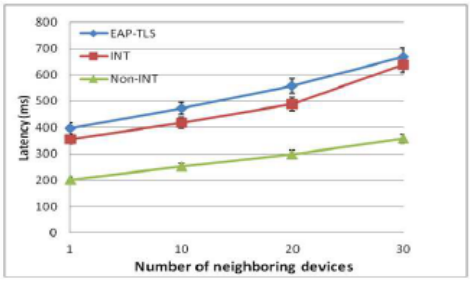

Figure 3: Average latency as function of number of neighbors

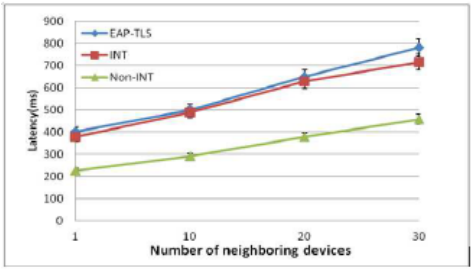

Figure 4: Maximum latency as function of number of neighbors

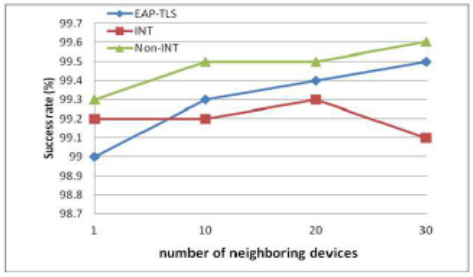

Figure 5: Success rate as function of number of neighbors

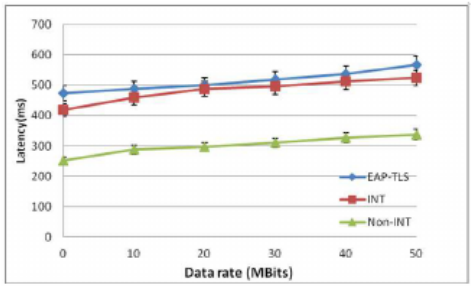

Figure 6: Average latency as function of background traffic

from the home device. Figure 6 shows average latency as function of data rate, which is varied from 10Mbits/s to 50Mbits/s. Data rate is 0 means that there is no background traffic. As the data rate increases, average latency of neighboring devices is enlarged. Higher data rate implies more background traffic to be processed by the home device, and more channel contention around the home device, resulting in longer delay. Figure 7 shows the maximum latency of 10 neighboring devices. As the data rate increases from 0 to 50MBits/s, the maximum latency of EAP-TLS, INT and Non-INT increases as expected, by approximately 34.3%, 19.9% and 18.2% respectively.

Figure 8 shows the success rate as the function of data rate. The success rate of key generation messages of 10 neighboring devices is at the range of 96.8% and 99.7%. We observe the success rate is higher if there is no background traffic (data rate is 0). However, the data rate does not have a big impact on success rate, which is a positive attribute of the key generation scheme.

5 Conclusion

Security has become the central issue for IoT and key management plays a critical role to ensure data confidentiality and integrity. A new design of ticket-based authentication protocol, an interactive key management protocol and a non-interactive key management protocol en-

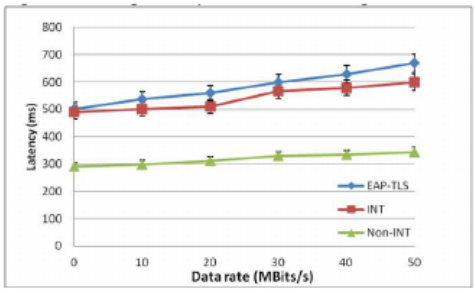

Figure 7: Maximum latency as function of background traffic

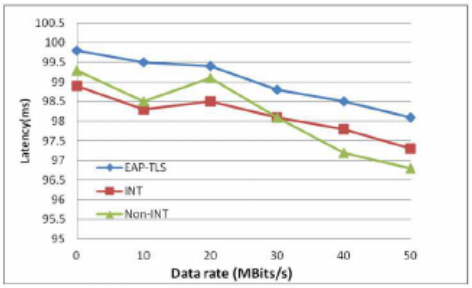

Figure 8: Success rate as function of background traffic

hanced the security of 4-way handshake and at the same time have lower latency than that of EAP-TLS and 4-way handshake. Unlike EAP-TLS and 4-way handshake. the interactive key management protocols support PFS and resists DoS attack,

References

[1] M. Frustaci, P. Pace, G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, 2018.

[2] M. Girault, "Self-certified public keys," in *Workshop on the Theory and Application of of Cryptographic Techniques*, vol. 547, pp. 490-497, 2001.

[3] L. Gutiérrez-Madroñal, M. F. Wagner, I. Medina-Bulo, "Test event generation for a fall-detection IoT system," *IEEE Internet of Things Journal*, vol. 6, no. 4, 2019.

[4] M. S. Henriques and N. K. Vernekar, "Using symmetric and asymmetric cryptography to secure communication between devices in IoT," in *International Conference on IoT and Application*, May 2017. DOI:10.1109/ICIOTA.2017.8073643.

[5] P. Horster, M. Michels, H. Peterson, "Meta-ElGamal signature schemes," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 96-107, 1994.

[6] C. T. Li, M. S. Hwang and Y. P. Chu, "An efficient sensor-to-sensor authenticated path-key establishment scheme for secure communications in wireless sensor networks", *International Journal of Inno-*

vative Computing, Information and Control, vol. 5, no. 8, pp. 2107-2124, Aug. 2009.

[7] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, pp. 52-61, 2003.

[8] L. Liu, Z. Cao, O. Markowitch, "A note on design flaws in one aggregated-proof based hierarchical authentication scheme for the internet of things," *International Journal of Electronics and Information Engineering*, vol. 5, no. 2, pp. 88–92, 2016.

[9] M. Long, "Energy-efficient and intrusion resilient authentication for ubiquitous access to factory floor information," *IEEE Transaction on Industrial Informatics*, vol. 2, no. 1, pp. 40-47, 2006.

[10] D. Manz, J. Alves-Foss and S. Zheng, "Network simulation of group key management protocols," *Journal of Information Assurance and Security*, pp. 67-79, 2008.

[11] H. Petersen, P. Horster, "Self-certified keys - concepts and applications," in *Communications and Multimedia Security*, pp. 102-116, 1997.

[12] M. Saikia, Md. A. Hussain, "Combinatorial group based approach for key pre-distribution scheme in wireless sensor network," in *International Conference on Computing, Communication and Automation (ICCCA'17)*, 2017. DOI: 10.1109/CCAA.2017.8229851.

[13] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp.161-174, 1994.

[14] S. Vanstone, *Deployments of Elliptic Curve Cryptography*, 2005. (`http://www.cacr.math.uwaterloo.ca/conferences/2005/ecc2005/vanstone.pdf`)

[15] O. Yagan, A. M. Makowski, "Wireless sensor networks under the random pairwise key predistribution scheme: Can resiliency be achieved with small key rings?," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, 2016.

[16] J. Zhao, "Probabilistic key predistribution in mobile networks resilient to node-capture attacks," *IEEE Transactions on Information Theory*, vol. 63, no. 10, 2017.

Biography

Cungang Yang completed his Ph.D degree in computer science in 2003 at University of Regina, Canada. In 2003, he joined the Ryerson University as an assistant professor in the Department of Electrical and Computer Engineering. His research areas include security and privacy, enhanced role-based access control model, information flow control, web security and secure wireless networks.

Celia Li completed her Ph.D degree in electrical engineering and computer science department in 2015 at York University. Her research is focused on security and privacy, role-based access control and wireless mesh network security.