KEMF: Key Management for Federated Sensor Networks

Piers O'Hanlon[1(✉)], Joss Wright[1], Ian Brown[1], and Tulio de Souza[2]

[1] Oxford Internet Institute, University of Oxford, Oxford, UK
{piers.ohanlon,joss.wright,ian.brown}@oii.ox.ac.uk
[2] Department of Computer Science, University of Oxford, Oxford, UK
tulio.de.Souza@cs.ox.ac.uk

Abstract. We present a lightweight key management protocol that provides secured device registration and communication in federated sensor networks. The protocol is designed for zero configuration and use in small packet low power wireless networks; protocol messages may fit into single packets. We use the Casper security protocol analyser to examine the behaviour and security properties of the protocol model. Within the assumptions of the model, we demonstrate forward secrecy, security against man-in-the-middle attacks, and local network key protection, comparing favourably with related protocols. Our experimental analysis shows that the protocol may feasibly be deployed on current sensor platforms with 256-bit elliptic curve cryptography.

Keywords: Privacy · Security · Sensor networks · Key management · IoT

1 Introduction

There are countless uses for devices that form the fabric of the Internet of Things (IoT) and sensor networks. Wireless sensor networks occur with varying degrees of complexity, the simplest wireless sensor networks have tended to be standalone systems running a single application that defines both the constituent nodes and all other aspects of the network. Increasingly, however, wireless sensor networks are being deployed in a multi-application structure comprising nodes running a common middleware that allows one or more applications to run on the same infrastructure operated by a single provider. The use of middleware, such as Senshare [6], offers a flexible abstraction of the low-level characteristics of the hardware, allowing data from each node to serve a number of applications.

The sharing model can be extended further by allowing *federation* of the infrastructure. A federated network is composed from many devices, which may be sourced from a number of different providers, and allows different entities to deploy nodes into the network and potentially run multiple applications across a common middleware. Federated sensor networking provides an economic benefit, and can lead to longer-term deployments offering a range of sensing options, but also raises privacy concerns for those individuals in the sensing environment [4].

J. Tian et al. (Eds.): SecureComm 2014, Part II, LNICST 153, pp. 17–24, 2015.
DOI: 10.1007/978-3-319-23802-9_3

We propose a novel protocol, KEMF, for secured device registration and key management for federated sensor and IoT networks. The protocol allows a newly connected device to automatically and securely obtain a network access key from the local gateway to enable secure and private communication. This is facilitated by a pre-arranged agreement between the entity which hosts the gateway and the organisation that provides the devices. However, KEMF does not require the exchange of any device specific secrets before the node enters the network.

We utilise the Casper [8] security protocol analyser to model its behaviour and examine its security properties. We also model another related protocol to provide for a comparison, and to highlight the benefits provided by KEMF.

2 Related Work

Whilst the core security primitives utilised by sensor and IoT networks are drawn from existing techniques and protocols, there are still new mechanisms required. There is a large body of research work in this field [2,12], though less of it has an emphasis on federated operation. Despite all the research work, the Internet Engineering Task Force (IETF) is has only recently formed the new ACE working group to tackle security issues in IoT and sensor networks.

There are many sensor systems, though there are less that aim to support federated operation with multi-application and multi-party ownership capabilities. One such system is SenShare [6], which is a platform that attempts to address the technical challenges in transforming sensor networks into open access infrastructures capable of supporting multiple applications.

The origins of modern day authentication and key management go back to the late 1970s when the Needham-Schroeder protocol was invented. In 1995 Gavin Lowe developed his advanced security analysis techniques based on Hoare's Communicating Sequential Processes (CSP) [3] and its model checker Failures Divergences Refinement (FDR2) [11], finding an attack [7] on the Needham-Schroeder protocol, and proposing the fixed Needham-Schroeder-Lowe protocol.

There have been many key management schemes developed for use with existing sensor networks [2,12], but few that aim to tackle the area of federated deployments [5,10]. MASY [10] is one of the few schemes that is aimed at federated network deployments, but it has its problems as we show in Sect. 4.

3 KEMF Security Protocol

The protocol provides for secure delivery of a network access key, from a local access control gateway, on to a new smart device in a federated network. It is assumed that the device has been distributed with a unique pre-installed asymmetric key, imprinted by its provider. This key is used only to provide for a secured request of its appropriate network access key. The delivery of the actual network access key is secured using a separate locally generated symmetric key. Whilst the protocol does involve communication with the device's provider server

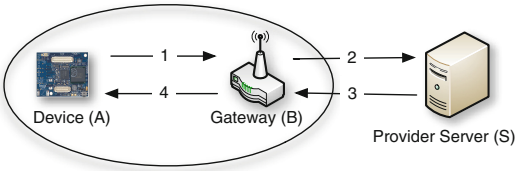

Fig. 1. System Actors and protocol steps

the local network key is not shared with the provider thus not compromising the privacy and security of the local network and allaying pervasive monitoring.

We employ asymmetric cryptography in an atypical fashion by requiring both halves of the keypair to be kept secret, rather than one key being made public and the other kept private. This approach allows the protocol to provide for improved security properties, and in particular forward secrecy. As the public key allows only for encryption of messages, subsequent compromise of the public key by an intruder does not allow compromise of previously sent messages. This approach therefore provides an improvement over the use of a single symmetric key, which could potentially lead to decryption of any previous or future messages. It should also be noted that, whilst we propose that both keys are kept secret, they are still used in the conventional manner whereby the 'public' key is used for encryption and the 'private' key is used for decryption.

Typically sensor and IoT devices are low power and minimally resourced entities. Thus for normal communications it is only feasible to use symmetric encryption, such as the Advanced Encryption Standard (AES). Whilst most modern devices are capable of performing the more onerous computation required for conventional asymmetric cryptography, they can be prohibitively slow even for basic operations. However Elliptic Curve Cryptography (ECC) which was developed by Koblitz and Miller, provides for asymmetric keys with significantly reduced computational requirements, and smaller key sizes. Specifically we employ the Elliptic Curve Integrated Encryption Scheme (ECIES).

We now introduce the details of the KEMF protocol. The three actors in the protocol and their interactions are outlined in Fig. 1, with the key terms defined in Table 1. We begin by defining the roles of the three actors:

Device(A): The sensor device to be registered on, and securely connected to, the network with its imprinted secret asymmetric 'public' key $K^{+}_{S(A)}$.

Gateway(B): The local gateway which provides for local key distribution and liaison with the provider server.

Provider Server(S): The cloud-based server that provides for authentication and authorised use of its devices on registered networks.

Device(A) is imprinted with its secret asymmetric public key, $K^{+}_{S(A)}$, by the provider allowing it to securely communicate with the provider server(S), which shares the corresponding asymmetric private key $K^{-}_{S(A)}$. The protocol

Table 1. KEMF Protocol key terms

$K^{+}_{S(A)}$	S(A)'s 'public' asymmetric imprinted key
$K^{-}_{S(A)}$	S(A)'s 'private' asymmetric key
k_A	A's generated symmetric session key
k_{BS}	Symmetric key agreed between B and S
k^{net}_A	A's local network access symmetric key

Table 2. KEMF Protocol steps and message sizes (in Bytes)

Step	Message	Size
1.	$A \rightarrow B : S, \{A, k_A\}_{K^{+}_{S(A)}}$	48 B
2.	$B \rightarrow S : \{A, \{A, k_A\}_{K^{+}_{S(A)}}\}_{k_{BS}}$	48 B
3.	$S \rightarrow B : \{A, B, k_A, K^{-}_{S(A)}\}_{k_{BS}}$	64 B
4	$B \rightarrow A : \{A, B, k^{net}_A\}_{k_A}$	32 B

assumes that a preconfigured trust relationship exists where the provider server has securely agreed, over a conventional secure channel (e.g. D/TLS), a session key, k_{BS}, with the gateway(B).

The steps of the protocol are outlined in Table 2, which we detail here:

1. The device(A) is switched on within range of a trusted gateway(B). At this stage B's address is unknown to A so it broadcasts out the initial registration message. The registration message is encrypted with A's public key, $K^{+}_{S(A)}$, and contains A's identifier (e.g. EUI-64 address), and its dynamically generated symmetric key k_A.
2. The gateway(B) receives the registration message, and after verifying that it has a pre-established relationship with provider server(S), it forwards the message to S using their agreed symmetric session key k_{BS}.
3. The provider server(S) receives the registration request and after verifying that device(A) belongs to gateway(B) it then decrypts the message and sends A's private key, $K^{-}_{S(A)}$, and generated symmetric key, k_A, to B using k_{BS}.
4. Once gateway(B) receives the message from provider server(S) it encrypts device(A)'s local network access key, k^{net}_A, using A's symmetric key, k_A. In this way B has the option to provide a separate network key to each device.

The protocol may also be broken down into two phases. Firstly the *registration* phase, which involves steps 1–3, only needs to occur once for each device after which B has obtained A's private key, $K^{-}_{S(A)}$, which then allows it to verify subsequent messages from A. Secondly there is the *(re)keying* phase which, after the registration phase has completed, only requires steps 1 and 4, thus freeing the provider server of further work, providing for reduced latency operation, and allowing autonomy for the local gateway to manage keys. To protect against

replay attacks in the rekeying phase the device retains all previous keys (k_A^{net}) so to avoid replays of step 4 containing old, potentially compromised, keys. The number of keys retained may be controlled by having the devices periodically utilise step 1 with a new k_A. We have kept the device message sizes within the limits of common low power wireless protocols such as IEEE 802.15.4, without making any compromises on addressing or key sizes. Typically the maximum payload size is around 90 bytes and if one utilises UDP over 6LowPAN then this can go down to 50–60 bytes.

4 Security Analysis

We take a theoretical approach to analysing KEMF by representing the protocol in a security protocol analyser to examine it for potential weaknesses and attacks. Our threat model, which is standard for wireless systems such as IoT and sensor networks, is largely similar to the commonly assumed Dolev-Yao model [1], where the attacker is assumed to have complete control over the communications channel, and may attempt to read any message, remove and change existing messages, or inject new messages. Although not strictly within the Dolev-Yao model we consider some situations with end system compromise.

We assume that the local gateway and provider server are trusted, whilst the sensor devices are potentially susceptible to compromise. Furthermore both the gateway and provider server trust one another, though the provider server has ultimate trust. The trust relationship between the server and gateway means that the server will only provide services for a predefined range of devices, thus limiting damage in the case of gateway compromise. The main threat to the system is for an intruder to obtain A's secret public key, $K^+_{S(A)}$ as the sensor nodes are typically more vulnerable as they are harder to physically secure.

We modelled KEMF using Gavin Lowe's Casper (Compiler for the Analysis of Security Protocols) [8]. Casper is implemented in Haskell and employs the process algebra CSP [3], in conjunction with its model checker FDR2 [11]. Lowe originally utilised CSP and FDR2 to develop a novel method for analysing security protocols, which proved to be remarkably successful in finding attacks upon a number of well known protocols [7,9]. Casper may be used not only model each actor in the protocol but to also explicitly model a malicious agent or *intruder*.

We firstly model the attacker without any knowledge of the device keys and Casper does not find any attacks on KEMF. Secondly we see that the protocol resists some attacks when A's public key, $K^+_{S(A)}$, is compromised. The KEMF protocol can resist a passive attack as an intruder cannot read the contents of the messages from A due to the use of asymmetric cryptography, nor can he read the messages from B due to the protection by A's session key, k_A. This compares well against MASY [10] which cannot resist these attacks.

However an active intruder that has effectively taken control of device(A) and obtained its public key, $K^+_{S(A)}$, may pose as A and obtain A's key to the network. Provided the attack does not occur before a non-compromised registration phase, an intruder node cannot maliciously inject a new key to be used by that or any

other node. Furthermore due to the fact that A's public key, $K^+_{S(A)}$ protects the transport of its internally generated key, k_A, any later compromise of $K^+_{S(A)}$ does not allow an intruder to decrypt any previous messages thus affording the node forward secrecy. Once such a key compromise has been detected it can be excluded from the network without affecting the future security of other devices on the network. Finally when both the device's private and public keys are compromised then the system is open to a number of attacks.

We also modelled MASY in Casper to understand its behaviour both with and without key compromise. The MASY protocol utilises an approach where the device is imprinted, by a 'company', with its IP address and symmetric key to provide for secured registration and enrollment on a network. Whilst the MASY provides for a key management solution it suffers from a number of problems. The compromise of the company symmetric key, from such a device, would lead to a general failure of the protocol as an intruder entity could both inject and read any messages from the past or future.

In summary, the KEMF protocol has the following security properties:

Forward Secrecy: The protocol provides for partial forward secrecy of the messages exchanged between the device and the gateway, which holds if either the 'public' asymmetric key, or one of A's existing session keys, is compromised. This is due to the protection of A's session key, k_A, by its public key, $K^+_{S(A)}$, and the use of a suitable KDF for A's session key. However if A's private asymmetric key, $K^-_{S(A)}$, is compromised then forward secrecy fails.

MitM Prevention: A Man in the Middle attack, between A and B, is prevented as the communication from A, via B, to S (or directly to B in a rekeying phase) is encrypted using A's imprinted asymmetric public key, $K^+_{S(A)}$, and communication from B to A is encrypted using A's session key k_A.

Local Network Key Protection: The local network access symmetric key, k^{net}_A, sent from B to A, is protected by A's session key k_A, which in turn is encrypted using A's public asymmetric key $K^+_{S(A)}$.

Replay Attack Protection: With no key compromises there is protection against replay attacks as an adversary cannot maliciously reuse subsections of the protocol due to the retention and non-reuse of old k_A's by A.

5 Experimental Analysis

We analysed the KEMF protocol on an embedded sensor node platform, the iMote2 from MEMSIC Inc, which runs linux-2.6 on a 419 MHz ARMv5. For ECIES, which is a hybrid elliptic curve asymmetric key algorithm, we utilised an implementation, based upon the OpenSSL library, of the SECG standard, using the 'secp256k1' elliptic curve in combination with 128-bit AES in cipher-block chain (CBC) mode, and the SHA512 hash function. The initial message (1), from Table 2, is 48 bytes long, which consists of a 32 byte encrypted payload plus the 16 bytes server ID. With this configuration the iMote2 encrypted the 32 byte

payload in an average of 41.5 ms, which means that the protocol is quite feasible on such a platform. The final message (4) sent to the device, to be decrypted by the iMote2, consists of an encrypted payload of the node ID (8 bytes), the gateway ID (8 bytes), and the node's network key (16 bytes). The 32 byte payload is decrypted using the 128-bit AES key, which is far quicker than the ECIES encrypted initiation message, taking an average of 3.9 µs.

Our implementation analysis shows that the protocol is feasible on current sensor platforms using asymmetric ECC based cryptography. The messages sent between the gateway and provider server are also small but are of less concern as they often use link layer technologies with larger MTUs.

6 Conclusions

We have detailed and evaluated KEMF a new device registration and key management protocol for federated networks that provides for low overhead operation. We have utilised the Casper security protocol analyser to show KEMF to be secure within our threat model. It provides for forward secrecy, even if the device's public key is compromised, protection against man in the middle attacks, and local network key security and privacy. It provides for better protection against key compromise than MASY, another similar protocol.

Our experimental analysis has shown that the protocol may feasibly be deployed on a current sensor platform, providing for good performance when using asymmetric elliptic curve cryptography. We also show that the protocol is suited for use in LowPANs where message sizes are very limited.

Acknowledgments. We would like to acknowledge funding from the EPSRC for the FRESNEL (EP/G070687/1), and Being There (EP/L00416X/1) projects.

References

1. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
2. Simplicio, M.A., Barreto, P.S.L.M., Margi, C.B., Carvalho, T.M.B.: A survey on key management mechanisms for distributed wireless sensor networks. Comput. Netw. **54**(15), 2591–2612 (2010)
3. Hoare, C.: Communicating Sequential Processes. Prentice-Hall, Upper Saddle River (1985)
4. Huygens, C., Joosen, W.: Federated and shared use of sensor networks through security middleware. In: ITNG, pp. 1005–1011 (2009)
5. Khan, S.U., Lavagno, L., Pastrone, C., Spirito, M.: An effective key management scheme for mobile heterogeneous sensor networks. In: i-Society (2011)
6. Leontiadis, I., Efstratiou, C., Mascolo, C., Crowcroft, J.: SenShare: transforming sensor networks into multi-application sensing infrastructures. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 65–81. Springer, Heidelberg (2012)

7. Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996)
8. Lowe, G.: Casper: a compiler for the analysis of security protocols. In: Computer Security Foundations Workshop, pp. 18–30 (1997)
9. Lowe, G., Roscoe, A.W.: Using CSP to detect errors in the TMN protocol. IEEE Trans. Softw. Eng. **23**, 659–669 (1997)
10. Maerien, J., Michiels, S., Huygens, C., Joosen, W.: MASY: MAnagement of Secret keYs for federated mobile wireless sensor networks. In: IEEE WiMob (2010)
11. Roscoe, A.W.: Model-checking CSP, Chap. 21. Prentice-Hall, Englewood Cliffs (1994)
12. Xiao, Y., Rayi, V.K., et al.: A survey of key management schemes in wireless sensor networks. Comput. Comm. **30**, 2314–2341 (2007)