A Key Distribution Protocol for Wireless Sensor Networks

Adrian Herrera*
C3I Division
Defence Science & Technology Organisation
Adelaide, Australia
Email: Adrian.Herrera@dsto.defence.gov.au
*The author worked on this project while at the CSIRO and the University of Wollongong

Wen Hu
Autonomous Systems Laboratory
CSIRO ICT Centre
Brisbane, Australia
Email: Wen.Hu@csiro.au

***Abstract*—This paper presents the design, implementation and evaluation of an automated method for distributing symmetric cryptographic keys in a Wireless Sensor Network (WSN). Unlike previous methods for key distribution in WSNs, we do not rely on sensitive knowledge to be stored in program memory prior to network deployment. Additionally, the protocol proposed uses dominant security primitives to ensure strong security and interoperability with existing networks (such as the Internet), while operating independent of the network layer protocol. Through both hardware experimentation and simulation, we show that this protocol can provide strong confidentiality, integrity and authenticity protection to the symmetric keys as they are distributed throughout a network, while maintaining the ability to scale to large-size networks and remain energy efficient.**

I. Introduction

Previous work in WSN security has often relied on sensitive cryptographic knowledge preloaded into a mote's CPU prior to network deployment. This results in increased risk to network security, as the capture of a single node by an adversary can lead to the compromise of the shared secret and hence all communication within the network [9].

To overcome this problem, we propose a key distribution protocol that does not rely on preshared cryptographic knowledge, is simple to use and is highly automated. Our work is primarily based around motes equipped with a Trusted Platform Module (TPM), such as the CSIRO Opal mote [6]. For those motes not equipped with a TPM, we utilise the TinyECC library [7] to provide the equivalent cryptographic functions (using ECC PKC rather than RSA). From our work we have identified the following properties that we believe a key distribution protocol should provide: security, simplicity, syndication, scalability, efficiency and interoperability.

II. Motivation and System Assumptions

The simplest approach for distributing a shared symmetric key amongst nodes in a WSN is to preload each sensor node with a network-wide fixed key prior to deployment (for example, in program memory). This approach has the following advantages [2]: there is no associated communication overhead because nodes are not required to communicate with each other to initialise the key; storage requirements are minimal because each node is only required to store a single key; and the approach is highly scalable because no communication between nodes is required upon deployment.

The main disadvantage of this technique is that the compromise of a single sensor node results in the compromise of the entire network [8]. Additionally, a single network-wide preshared key is unable to ensure authenticity at the sensor node level.

Our aim is thus to design a protocol for distributing symmetric cryptographic keys amongst resource-constrained (computation, memory and energy) motes within a WSN. In order to make the proposed protocol effective in real-world sensor network deployments, it must adhere to the properties listed in §I.

A. Assumptions

For the purposes of this key distribution protocol, we first assume a single-tier hierarchical WSN, where a node is either a sensor node or a base station. If the node is a base station, it is assumed to be attached to a more powerful and secure workstation for storing cryptographic keys, or for offloading the required cryptographic operations.

Secondly, the key distribution protocol proposed provides a method for distributing cryptographic keys throughout a WSN to allow for secure sensor node-to-base station communication only. This suits our target applications (for example, control in industrial environments and remote sensing on the battle field), where data is collected by sensor notes and required to be securely forwarded through the network to a central base station.

Finally, after the key distribution procedure has completed, we assume that the motes will use a symmetric encryption technique for further communication of data to the base station.

III. Protocol Design

In this section we describe the proposed key distribution protocol in detail. This is followed by the justification of our design choices against the required properties listed in §I.

A. Protocol Description

Table I outlines the notation used to describe our key distribution protocol. The procedure for obtaining a shared key to allow a sensor node to communicate with the base station securely is illustrated in Fig. 1.

Symbol	Description
$K_{x,y}$	Symmetric key shared between x and y
K_x^+	x's public key
K_x^-	x's private key
$H(\cdot)$	Cryptographic hash operation
$\{\cdot\}_K$	Encrypt/sign operation using key K

TABLE I
DESCRIPTION OF CRYPTOGRAPHIC NOTATION

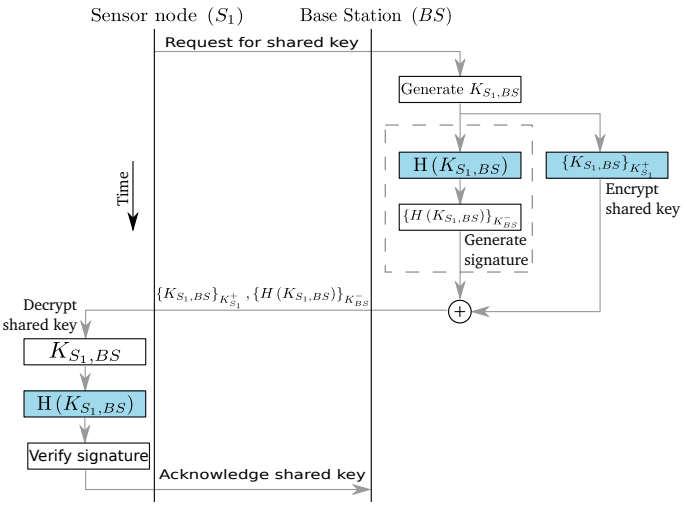


Fig. 1. Key distribution protocol flowchart

During deployment, the base station's (BS) public key is loaded on to all nodes within the network. Similarly, all sensor node's public keys are stored in BS. For large-size networks, the public keys would be stored in a database on the base station's attached workstation.

A sensor node (S_1) periodically sends a request for a new shared key to BS. To defend against replay attacks, a nonce is included with this request. This nonce is randomly generated by the sensor node for each key request, and also serves as the initial sequence number for that session. After receiving the request, BS produces the shared key $K_{S_1,BS}$. The length of $K_{S_1,BS}$ is dependant on the symmetric encryption algorithm intended for securing future communication.

Following the generation of $K_{S_1,BS}$, BS encrypts $K_{S_1,BS}$ with S_1's public key and calculates a signature using a SHA1 hash of $K_{S_1,BS}$. The encrypted key and signature pair are transmitted to S_1.

Upon reception of the encrypted shared key and signature pair, S_1 can use it's private key to decrypt $K_{S_1,BS}$. The signature is then verified based on the SHA1 value of $K_{S_1,BS}$ and the public key of BS. If the signature is verified correctly, an acknowledgement message is sent to BS. BS can then associate $K_{S_1,BS}$ with S_1 for securing future communication.

B. Reliable Communication

No assumptions were made on the underlying network layer protocol; our key distribution protocol has been designed to function independent of the network layer protocol used.

Although the TPM's 2048-bit RSA engine provides interoperability with existing cryptography standards, one downside is the relatively large size of the encrypted message and signature produced. Because the MTU of low-power networks is typically limited to 128 bytes, packet fragmentation is required. To ensure reliable end-to-end communication between a sensor node and the base station, a stop-and-wait ARQ scheme was implemented [4].

For motes without a TPM, we use 160-bit ECC. While the TPM's MTU limitations do not exist when ECC is used, reliable end-to-end communication is still required in the case where packets are lost.

A basic form of network congestion control was also implemented. This included exponential timeouts for dropped packets and packet snooping on neighboring nodes [4].

C. Protocol Properties

In this section we analyse the proposed approach against the desired properties listed in §I.

Security When the TPM is utilised, the proposed key distribution protocol does not rely on sensitive cryptographic knowledge to be preloaded into each mote's CPU prior to network deployment. Sensitive information is instead stored within the tamper-resistant TPM. Note that this is not the case when ECC is used, because the private keys must be stored within the CPU's memory (which is not tamper-resistant). Additionally, both RSA and ECC PKC ensure the necessary levels of communication confidentiality and authenticity.

Simplicity The proposed key distribution protocol does not require any human intervention once the network has been deployed. This makes it highly automated.

Syndication The proposed key distribution protocol makes no assumption on the underlying network layer protocol, and has been designed to ensure independence. We have implemented and evaluated the proposed key distribution protocol on top of a number of popular network layer protocols available in TinyOS; Collection Tree and Dissemination. We will discuss this property further in §IV.

Scalability We will show that the proposed protocol can scale to large-size networks with small key distribution times in §V-C.

Efficiency While PKC is not traditionally used in WSNs due to the high overheads, the inclusion of the TPM provides an efficient mechanism to perform these operations at less than 5% of the financial cost of the mote hardware [5]. We also implemented the cryptographic

primitives using the TinyECC library. A comparison of the two PKC methods is provided in §V.

Interoperability The proposed protocol uses dominant security primitives (namely, RSA PKC) to ensure interoperability with traditional networks such as the Internet. While ECC offers smaller key sizes, it is not as widely used in the Internet and thus limits interoperability.

IV. Protocol Implementation

Our key distribution protocol has been developed for the CSIRO Opal mote [6] running the TinyOS 2.x operating system [1]. Two interfaces are provided: an application-level interface to start, stop and set the renewal frequency of the key distribution procedure; and a network layer interface to wrap the required network layer protocol and provide an end-to-end communication service to the application-level interface [4].

A. Parallel Key Distribution

Because we assume the base station is typically resource-rich compared to a sensor node (i.e. it has access to a workstation), the key distribution protocol was extended to allow multiple concurrent key distribution transactions in the base station. This was achieved via a multithreaded Java application that executes on the base station, with a new thread allocated for each key request. Interoperability can be ensured by using standard cryptography packages and classes (e.g. those from the `javax.crypto` and `java.security` packages).

V. Evaluation

We evaluated our key distribution protocol using both a small-scale network testbed and large-scale network simulation. Due to our development platform featuring an integrated TPM we mainly focus on these results, and offer our TinyECC results as a comparison. Further details are available in [4].

A. Execution Times

A pair of Opal motes was used to measure the time taken for a single sensor node to request and receive a shared key from the base station. The execution times of the various operations required by the key distribution protocol were measured and are given in Fig. 2.

Fig. 2 shows that the sensor node spent a total of $1,277$ ms on cryptographic operations when the TPM was used. This included RSA decryption (972 ms) and signature verification (305 ms). In comparison, the base station spent 1,239 ms. This included symmetric key generation (19 ms), RSA encryption (257 ms) and signature generation (963 ms).

The TPM also required 836 ms to execute the various initialisation routines [3]. In total, the time for a sensor node to request, receive and verify a shared key was approximately 3.452 seconds.

The TinyECC operations were on average four times slower than the equivalent RSA-based operations executed on the TPM (Fig. 2). Note that the TinyECC library required its own initialisation to generate the required elliptic curve parameters and public keys. The total time for the TinyECC-based key distribution protocol to execute was approximately 10.728 seconds.

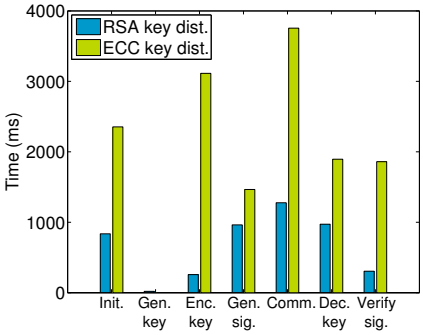
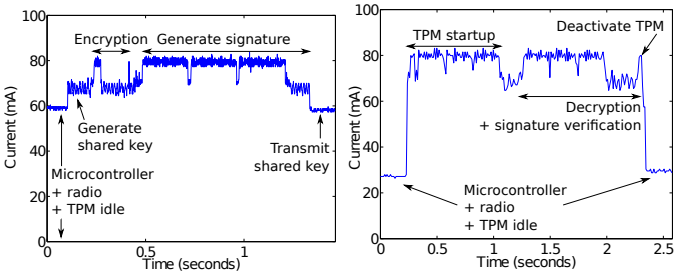


Fig. 2. Execution times of the different stages of the key distribution protocol

B. Energy Requirements

Fig. 3 shows the current consumption over time for both the sensor node and base station as they completed the key distribution procedure (with the TPM). Similarly to §V-A, the base station mote (rather than the attached workstation) performed all the required cryptographic operations.



(a) Base station (b) Sensor node

Fig. 3. Current draw over time for two motes performing the RSA-based key distribution procedure

Because the base station mote is typically connected to a permanent power supply, its TPM remains active. This avoided the delays associated with TPM startup.

To minimise current consumption and energy requirements, the sensor node's TPM was only activated when required (i.e. when a cryptographic operation was required).

When the TPM was activated but idle, it consumed approximately 60 mA. When performing a computationally-intense operation (e.g. RSA encryption/decryption or RSA signature generation/verification), the current consumption of the mote increased to approximately 80 mA. When less computationally-expensive operations were executed (e.g. random number generation or an operation on the I^2C bus) current consumption decreased to approximately 70 mA.

When performing computationally-expensive ECC operations (e.g. encryption, decryption, signature generation and signature verification), the current draw of the Opal mote increased by 20 mA to 47 mA. Note that the Opal mote consumed approximately 27 mA when both the microcontroller and radio were idle.

The energy consumption of each operation is given in Table II (based on the results here and §V-A).

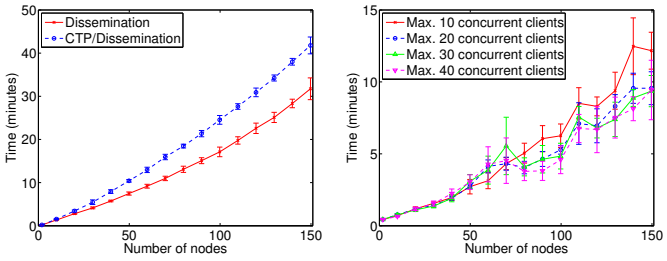
Operation	Energy (mJ) TPM	TinyECC
Initialisation	273.87	442.55
Generate key	5.7	N/A
Encrypt key	84.3	585.43
Generate signature	323.57	275.42
Communication	125.33	405.54
Decrypt key	326.59	356.26
Verify signature	100.04	349.68
Total	1,239.4	2,414.88

TABLE II
ENERGY CONSUMPTION OF THE DIFFERENT OPERATIONS IN THE KEY DISTRIBUTION PROTOCOL

C. Scalability

The results in §V-A were used as the basis for investigating the scalability of the key distribution protocol in large-scale WSNs. Using the execution times shown in Fig. 2, the various TPM operations were replaced by timer components to emulate the TPM's functionality. This allowed the key distribution protocol to be simulated in the TinyOS SIMulator (TOSSIM).

To measure scalability, the TOSSIM simulation was repeatedly run over a number of varying-sized network topologies. For each simulation, the total time taken for the base station to distribute the key to each of the sensor nodes in the network was recorded. This was done for both the CTP and Dissemination network layer protocols.



(a) Single-threaded server (b) Multi-threaded server

Fig. 4. Time taken to complete the key distribution procedure

Fig. 4a shows the total time for both the Dissemination and CTP network protocols increased exponentially with the size of the network. Using only the TinyOS Dissemination protocol, the key distribution protocol was able to complete in 25% less time than when a combination of CTP and Dissemination was used.

If the key renewal frequency is once per day at 1% duty cycle (which allows $1\% \times 24 \times 60 = 14.4$ minutes to complete the key distribution procedure for all nodes within the network), then the Dissemination protocol can operate in a network of up to 80 nodes. In comparison, the combination of CTP and Dissemination can operate with up to 60 nodes only.

D. Parallel Key Distribution

To further improve the efficiency of the key distribution protocol, the base station mote's cryptographic operations were offloaded to the attached workstation to allow the processing of multiple sensor nodes concurrently. As introduced in §IV-A, a multithreaded Java application was developed to achieve this parallelisation. The number of concurrent clients was limited to between 10 and 40 nodes, while the Dissemination protocol was used as the underlying network protocol.

Fig. 4b shows that parallelising the key distribution protocol introduced a significant 74% improvement in the total time to distribute a shared key to 150 sensor nodes while serving a maximum of 20 to 40 concurrent clients. This improvement makes the proposed key distribution protocol practical in real-world sensor network deployments. For example, if the key distribution procedure requires less than 10 minutes to complete in a 150-node network (while serving a maximum of 20 to 40 concurrent sensor nodes) and is performed once per day, this represents a duty cycle of only $\frac{10}{60\times24}\times100\% = 0.7\%$.

VI. CONCLUSION

We have introduced a key distribution protocol that provides an efficient, secure and automated method for providing nodes within a WSN with a symmetric cryptographic key that can be used for securing data communication. The choice of RSA or ECC encryption primitives (through an integrated TPM and the TinyECC library respectively) is offered, however we favour the functionality provided by the TPM. Our evaluation shows that this key distribution protocol is able to provide a high level of security (through tamper-resistant hardware and RSA PKC) and interoperability (through RSA PKC), while maintaining efficiency (in both energy and memory requirements) and the ability to scale to large-size networks.

REFERENCES

[1] T. T. 2.x Working Group, "Tinyos 2.0," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005.

[2] H. Chan, A. Perrig, and D. Song, *Key distribution techniques for sensor networks*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[3] T. C. Group, *TPM Main - Part 3 Commands*, Trusted Computing Group, July 2007, specification Version 1.2.

[4] A. Herrera, "A key distribution protocol for wireless sensor networks," University of Wollongong, undergraduate thesis, 2011.

[5] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward trusted wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 7, no. 1, Aug. 2010.

[6] R. Jurdak, K. Klues, B. Kusy, C. Richter, K. Langendoen, and M. Brünig, "Opal: A multi-radio platform for high throughput wireless sensor networks," *Embedded Systems Letters*, vol. 3, Issue 4, November 2011.

[7] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, april 2008.

[8] A.-N. Shen, S. Guo, and H.-Y. Chien, "An efficient and scalable key distribution mechanism for hierarchical wireless sensor networks," in *Sarnoff Symposium, 2009. SARNOFF '09. IEEE*, 30 2009-april 1 2009.

[9] A. Wood and J. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, oct 2002.