Implementation of an RFID Key Management System for DASH7

Aparna Vegndla, Hwajeong Seo, Donggeon Lee, and Howon Kim*, *Member, KIICE*

Department of Electrical and Computer Engineering, Pusan National University, Busan 609-735, Korea

Abstract

The wireless sensor networking standard DASH7 operates in low-power communication with a better transmission quality in active RFID networks. The DASH7 security standard supports public key cryptography. At present, the DASH7 standard uses the message authentication code in the network layer for authentication and integrity. However, its security standard is still in an incubation stage with respect to the implementation of a crypto exchange over a DASH7 network. Effective key management is an important factor for privacy and security. If organizations are not careful about where and how keys are stored, they leave the encrypted data vulnerable to theft. In this regard, we present a key management system designed for efficient key management through public key infrastructure authentication as well as a non-repudiation feature for the DASH7 standard. We analyze the performance of the proposed system on a basis of various performance criteria such as latency and throughput.

Index Terms: DASH7, MAC, PKI, Public key cryptography, RFID

I. INTRODUCTION

DASH7 is a newly developed wireless sensor networking (WSN) and radio-frequency identification (RFID) standard. It is based on the ISO/IEC 18000-7 open standard for the license-free 433-MHz ISM band. This standard is designed for extremely low-power applications, active RFID, and WSNs. The DASH7 specification includes network protocols, architecture, and security [1].

Wireless sensor networks have been used to monitor some physical phenomena (e.g., temperature, pressure, humidity, and light) in the environment. In cases where the privacy of the collected data is a primary concern, secure networking and communication is very important. It is a great challenge to make data tamperproof due to the open

nature of wireless communication used in WSNs. This open nature causes maintenance problems with respect to the privacy, security, and reliability of the transmitted data.

The traditional WSN technology uses symmetric cryptography to protect the network from adversaries. However, it poses numerous problems, including those related to key distribution and the number of established keys. Therefore, key management becomes an intensive computational task for which WSNs are not suitable due to the limited computational power and storage capacity of the sensor nodes. On the other hand, public key cryptography simplifies key management by allowing secure communication by the use of “n” public keys and private key pairs in a network having “n” nodes. The DASH7 standard supports such a public key cryptography method. Hence, it can

Received 19 July 2013, Revised 14 October 2013, Accepted 31 October 2013

*Corresponding Author Howon Kim (E-mail: howonkim@pusan.ac.kr, Tel: +82-51-510-1010)

Department of Electrical and Computer Engineering, Pusan National University, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 609-735, Korea.

Open Access <http://dx.doi.org/10.6109/jicce.2014.12.1.019>

print ISSN: 2234-8255 online ISSN: 2234-8883

© This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright © The Korea Institute of Information and Communication Engineering

tackle the privacy issues head-on with its support of full public key encryption. The DASH7 security standard is implemented using the message authentication codes (MACs) in the network layer for the sake of authentication and integrity [2].

The central problem with the use of public key cryptography is the provision of the proof that a particular public key is authentic and the verification of whether the person who claims to have the public key is the correct person or not. This problem can be resolved by using the public key infrastructure (PKI) where the certification authority (CA) is the trusted third party who certifies the ownership of the public key. In this paper, we implement such an infrastructure through a key management system (KMS) that provides keys and digital certificates that are the best suited for DASH7-based applications. We have evaluated the performance of our system in terms of the latency and the throughput calculated by the signature generation and verification through the KMS in a personal computer (PC).

The rest of this paper is organized as follows: In Section I, we introduced DASH7. In Section II, we provide a background of DASH7. In Section III, we propose the implementation of the KMS. In Section IV, we evaluate and analyze the performance of the proposed KMS. Finally, in Section V, we conclude with a brief summary of our contributions.

II. BACKGROUND

We conducted an extensive study on DASH7. Since the security part of DASH7 is not yet fully defined, there is little work existing in this field. Previous work have contributed to DASH7 security by implementing network and data link layer security using an advanced encryption standard (AES) counter with CBC-MAC (CCM) over a CC430F5137 microcontroller [2]. This paper is an extensive work to improve the security of DASH7 through PKI authentication.

In this section, we will briefly introduce the DASH7 wireless networking standard and discuss its current status with respect to standardization.

A. DASH7 Wireless Networking Standard

DASH7 is an open-source standard for a WSN operating in the unlicensed ISM band at 433 MHz and follows the ISO/IEC 18000-7 standard. It has several advantages as compared to some of the regular wireless networking standards such as Wi-Fi, Bluetooth, and ZigBee. DASH7 has a maximum communication range of up to 2 km and can provide a relatively long battery life. Features like long-

range communication and an extremely low power requirement makes DASH7 an ideal choice for various application areas, such as defense, logistics, automotive control, and building automation. Its protocol stack is also small. It can support both private key (i.e., AES 128) and public key (elliptic curve cryptography [ECC] and RSA) techniques.

III. PROPOSED INFRASTRUCTURE (KMS)

PKI is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates [3]. Although DASH7 supports ECC, its PKI implementation is still under draft. To contribute to the DASH7 security specification, this paper proposes PKI for DASH7-based applications through KMS. We define the proposed KMS infrastructure to be as similar to PKIX as possible [4]. In this paper, we strictly focus our research on PKI for DASH7. The features of the proposed KMS are as follows:

- 1) Using 163-bit ECC for public key pair generation, the KMS consumes less CPU resources for signature generation and verification. This feature will be extremely useful for resource-constrained DASH7-based RFID devices.
- 2) For secure session management between DASH7 applications, a public key certificate (PKC) is used for session key establishment. This is in contrast to many of the currently available schemes that use asymmetric algorithms for encryption.
- 3) AES-CCM is used for data encryption between devices for data confidentiality. Currently, AES-CCM encryption is not available in DASH7 applications.

The proposed system is based on the existing certificate authority [5, 6]. The basic infrastructure of KMS supports several software components via Web interfaces. These components include CA, registration authority (RA), user, and repository.

A. KMS Architecture

Fig. 1 shows the relation between various interfaces. The user interface is for user access. The user cannot create private/public keys by him. CA creates keys for the user when it receives a request from the user.

In the request form, along with the basic information and contact details, the user also sends the user policy agreement that includes the level of assurance, the key generation mode, and the RA name. The RA interface consists of the

Fig. 1. Relation between component supports by key management system. CA: certification authority, CRL: certificate revocation list, RA: registration authority.

Fig. 2. Overview of experimental setup: key management system (KMS) generates and distributes certificates to the gateway and the tag. The reader collects the data and transfers them to the host and the middleware.

list of user requests; this interface has all functionalities including creating certificates and certificate revocation lists (CRLs), editing, approving requests, and deleting requests in the case when the requests come from an unauthorized entity. After RA approves a request, CA issues the certificates, and guarantees the authenticity of the entities. Finally, the issued certificates get stored in the repository interface that maintains the database of the certificates and CRLs.

B. Experimental Setup

A basic experimental setup for working with KMS consists of several software and hardware components. We worked on three computers: Host, KMS, and Middleware. We used three CC430 boards for the gateway (GW), tag, and reader, respectively. The major constituents of our experimental setup are described below in sequence:

- 1) **Middleware:** Middleware is connected to the reader and is used for collecting information from the reader.
- 2) **Host:** Host receives information from the middleware and data from the tag through the reader.

- 3) **OpenTag:** OpenTag [7] is an open-source stack actively used for developing DASH7-based applications. It is an embedded stack and is designed to run on DASH7 Mode 2 microcontrollers. OpenTag is written in C and can be used with a “bare metal” microcontroller unit, real-time operating systems, or POSIX-compliant systems. OpenTag can be used to build servers (DASH7 devices) or clients (user interfaces).
- 4) **Target board:** The Texas Instruments CC430 microcontroller family is an ultra-low power system on chip (SoC) series with an integrated RF core and several other peripherals. In our research, we have used the CC430F5137 microcontroller from the CC430 family. The target board is equipped with an AES accelerator module and can support the OpenTag software protocol stack; hence, it is an ideal choice for developing security and KMS applications for DASH7 [2]. We worked with CC430 boards for the GW, tag, and the reader. Fig. 2 shows how both the GW and the tag exchange their certificates over DASH7.

C. Certificate Generation for User

The sequence of steps involved in the generation of the certificate for the end-entity is as follows:

- 1) A user creates a public/private key pair when user requests a certificate from CA through the certificate signing request (CSR) and sends Pub_i , $SUBJECT_i$, PW_i , $Serial_i$, R , KEY_{size} , $KEY_{algorithm}$, and RA_{name} along with the request.
- 2) RA can verify the request to CA from the request list on its interface. RA approves the user request and signs it.
- 3) After checking whether RA has already approved the request, CA generates a hash table of the user request along with the user's public key. CA signs the hashed information with its private key and issues the certificate.
- 4) After the certificate has been issued, CA stores the user certificate in the repository and frequently updates the CRL in the repository.
- 5) CA sends the user the certificate, and then, the user installs this certificate on the user interface.

An overview of the certificate generation process is illustrated in Fig. 3.

D. Verification of Certificate by KMS

Verification is a key factor for the identity of a certificate. We have provided the verification facility through a KMS Web interface. During verification, the signature on the PKC of the user is verified using the PKC of the CA that issued the certificates to the user.

Fig. 3. Generation of a user certificate. U_i : user i , Pub_i : public key of i , Pri_i : private key of user i , $Cert_{U_i}$: certificate of user i , $SUBJECT_i$: user name, organization, etc., PW_i : password of user i , $ISSUER_i$: issuer i , $Serial_i$: serial number of the certificate of user i , R : role of the user, KEY_{size} : size of the key, $KEY_{algorithm}$: type of algorithm used for key generation, Pri_{RA} : private key of registration authority, Pri_{CA} : private key of certification authority, CSR_{U_i} : certificate signing request of user i , CRL : certificate revocation list of users, $h()$: hash function.

Fig. 4. Certificate exchange between the gateway and the tag. GW_{cert} : gateway's certificate, TAG_{cert} : tag's certificate.

E. Mutual Authentication among Devices

We worked with devices, GW, and tag implemented using OpenTag. Once the devices are mutually authenticated, they are capable of exchanging encrypted and authenticated data frames. The steps involved in mutual authentication are as follows.

1) GW/TAG Certificate Exchange

The first stage of the process for mutual authentication with public key cryptography is the exchange of certificates. The exchange of certificates between the GW and the tag is shown in Fig. 4.

CA generates the certificates for the GW and the tag. This is done by providing the signature on the digest of their identities using the elliptic curve digital signature algorithm (ECDSA)-with-SHA256. CA creates the keys for the GW and the tag and signs using the sect163r2 EC public key algorithm. First, message1 is sent by the GW with its

certificate and additional data to the tag over the DASH7 network. The tag verifies the signature on the certificate with the KMS CA public key, which is generated by the sect163r2 public key algorithm. Secondly, message2 is sent by the tag with its certificate and additional data to the GW over the DASH7 network. Further, the GW verifies the signature on the certificate with the KMS CA public key. By the end of these two messages, both the GW and the tag authenticate themselves.

2) Key Establishment

The process for key establishment is illustrated in Fig. 5. As CA generates a 163-bit public key, the sequence of steps involved in the key establishment between the GW and the tag is as follows:

- 1) GW sends a broadcast message (84 bytes) to the tag until it confirms the message. A message consists of the secret key (42 bytes) generated by the elliptic curve Diffie-Hellman (ECDH) value and the GW's signature on the secret key generated by ECDSA.
- 2) The tag verifies the GW message by using the certificate of GW, sent in response of the secret key to GW. Further, it creates a session key. The tag's key table consists of the public key of GW (42 bytes), private key (42 bytes), session key (16 bytes), and counter (1 byte). Hence, the size of the tag's key table is $42 + 42 + 16 + 1 = 101$ bytes.
- 3) The GW key table consists of the GW private key (42 bytes), the tag's ID (8 bytes), the session key (16 bytes), the tag's public key (42 bytes), and the counter (1 byte). In total, the size of the GW's key table is $42 + N \cdot (16 + 42 + 1) = 42 + 59N$ bytes. Here, "N" is defined as the number of tags.
- 4) Once the session key is established, both the GW and the tag can securely send encrypted and authenticated messages between them. AES-CCM was used for encryption. A message consists of an ID, encrypted data (in multiples of 16 bytes), and the authenticated message (8 bytes).

Fig. 5. Key exchange between the gateway and the tag. SK_{gw} : the gateway's secret key for calculating the elliptic curve Diffie-Hellman (ECDH) value, SK_{tag} : the tag's secret key for calculating the ECDH value, K_s : session key, E_{K_s} : encryption using the session key.

IV. EVALUATION AND PERFORMANCE ANALYSIS

In our KMS implementation, we have mainly addressed the implementation of PKI, which will improve the security standard of DASH7. Apart from this, another significant contribution is an enormous reduction in the amount of X.509v3 certificate storage by the omission of the extension fields from the X.509 certificate's format. This procedure requires that the extensions of the certificate be specified in the database configuration file of KMS. Thus, extension details can be saved in the server. Further, the reduced certificate is sent to the target device.

We used CC430 microcontrollers for the implementation of the GW and the tag. We have verified the certificate received by CC430 microcontrollers through serial port of CC430 microcontrollers, by accessing its memory in the debug mode by using the CCS tool.

A. Evaluation

With our KMS, we have improved several features of DASH7. An evaluation of our major contributions is as follows:

- 1) Security and reliability of DASH7: With our KMS, we have provided a non-repudiation facility through PKI by KMS CA. KMS CA provides digital signatures to the identities of the DASH7 applications by using optimal ECDSA. Although DASH7 supports ECC, earlier, the PKI implementation for DASH7 was under draft. With the proposed KMS's PKI, we have enhanced the security and the reliability of the DASH7 standard.
- 2) Data storage overhead: As part of our KMS implementation, we have reduced the original size of the certificate from 4.3 kB to 693 bytes. This large reduction of more than 85% in certificate size will have a considerable positive impact on the network congestion and the data storage overhead.
- 3) Computational cost: In our KMS implementation, the signature generation and verification is done on a PC by using ECC. Hence, virtually, no computation is done on the sensor node. In this modern era, computation on a PC is almost free. Hence, the proposed KMS reduced the computational cost on the sensor node by almost 100%.
- 4) Memory cost on target device: As the certificate size has been reduced by 85%, it will have a direct positive impact on the memory requirement of the target device. Our target board has a 32-kB flash memory and 4-kB RAM. In 26.5 kB of the flash memory, the source code can be uploaded. The application code size is 24.5 kB, and the data size is 1528 bytes. In total, the size of data

after including the certificate is $1528 + 693 = 2221$ bytes. If the original certificate size of 4.3 kB has to be used, then it will exceed the RAM capacity. Usually, sensor nodes are constrained by precious resources like memory. With our KMS implementation, the memory cost of target devices can be significantly reduced.

B. Performance Analysis

We used Intel Core 2 Duo CPU E8400 3.00 GHz for a performance analysis in terms of the latency and the throughput. The calculation of the latency and the throughput metric for the operations in the key agreement protocol using ECDH operations and the ECDSA algorithm is described below.

Tables 1 and 2 present the time taken by the 163-bit ECC from 12 signatures per second and 12 verifications per second.

To compute the latency and the throughput, the following metrics are used:

$$\text{Latency} = \text{Time taken for ECDH key generation} + \text{Time taken for signature generation} + (\text{Time taken for verification} * 2)$$

$$\text{Throughput of signature generation} = \text{Total time taken for signature generation} / 12$$

$$\text{Throughput of verification} = \text{Total time taken for verification} / 12$$

V. CONCLUSION

In this study, we implemented a KMS that could efficiently provide authentication, integrity, and confidentiality along with PKI authentication.

Table 1. Time taken by cryptographic operations for 163-bit ECC

	163 bits (ms)		
	Best case	Average case	Worst case
ECDH key generation	45.329	83.488	120.722
Sign	49.461	84.552	154.783
Verify	46.438	84.16	164.687

ECC: elliptic curve cryptography, ECDH: elliptic curve Diffie-Hellman.

Table 2. Throughput and latency for 163-bit ECC

	Throughput	Latency
Sign	84.552	187.666
Verify	84.16	

ECC: elliptic curve cryptography.

This KMS can act as an interface for providing public key-based certificates by generating digital signatures for the identities of DASH7 applications using optimal ECDSA. The evaluation shows that the KMS makes the DASH7 network standard more secure and more reliable; it considerably reduced the data storage overhead, reduced the certificate size by more than 85%, and almost nullified the need for the computational cost of the precious memory on the target device. The performance analysis of the KMS system is conducted on the basis of various performance criteria such as latency and throughput. A relatively low latency in the algorithm indicates less use of computational resources.

ACKNOWLEDGMENTS

This work was supported by the Industrial Strategic Technology Development Program (No. 10043907, Development of High-Performance IoT Device and Open Platform with Intelligent Software) funded by the Ministry of Science, ICT & Future Planning, Korea.

REFERENCES

- [1] J. P. Norair, "Introduction to DASH7 technologies," Dash7 Alliance White Paper, 2009.
- [2] H. Seo and H. Kim, "Network and data link layer security for DASH7," *Journal of Information and Communication Convergence Engineering*, vol. 10, no. 3, pp. 248-252, 2012.
- [3] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 5th ed. New York, NY: Prentice-Hall, 2011.
- [4] Internet Engineering Task Force, Public-key infrastructure (X.509) working group [Internet], Available: <http://datatracker.ietf.org/wg/pkix/charter/>.
- [5] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 public key infrastructure certificate and CRL profile," The Internet Engineering Task Force, Fremont, CA, *RFC 2459*, 1999.
- [6] C. Covell and M. Bell, OpenCA guides for 0.9.2+ [Internet], Available: <ftp://outside.ihg.uni-duisburg.de/security/openca-guide.pdf>.
- [7] J. P. Norair, Indigresso Wiki: OpenTag project [Internet], Available: <http://www.indigresso.com/wiki/doku.php>.

Aparna Vegendla

received the M.S. degree in Computer Engineering at Pusan National University in 2014. Her research interests include sensor networks, information security, key management system.

Hwajeong Seo

received the B.S.E.E. degree from Pusan National University, Busan, Korea in 2010, and the M.S. degree in Computer Engineering at Pusan National University. He is in Ph.D. degree in computer engineering from Pusan National University. His research interests include sensor networks, information security, Elliptic Curve Cryptography, and RFID security, IoT.

Donggeon Lee

received the B.S.E.E. degree from Pusan National University, Busan, Korea and the M.S. degree in Computer Engineering at Pusan National University. He is in Ph.D. degree in computer engineering from Pusan National University. His research interests include PUF, FPGA, side channel attack, IoT.

Howon Kim

received the B.S.E.E. degree from Kyungpook National University, Daegu, Korea, in 1993 and the M.S. and Ph.D. degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied with the COSY group at the Ruhr-University of Bochum, Germany. He was a senior member of the technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He is currently working as an associate professor with the Department of Computer Engineering, School of Computer Science and Engineering, Pusan National University, Busan, Korea. His research interests include RFID technology, sensor networks, information security, and computer architecture. Currently, his main research focus is on mobile RFID technology and sensor networks, public key cryptosystems, and their security issues. He is a member of the IEEE, and the International Association for Cryptologic Research (IACR).