SSL-based Lightweight Security of IP-based Wireless Sensor Networks

Wooyoung Jung, Sungmin Hong, Minkeun Ha, Young-Joo Kim, Daeyoung Kim†

Korea Advanced Institute of Science and Technology
sida37@icu.ac.kr, {ghitsh, minkeun.ha, yjkim73, kimdl}@kaist.ac.kr

Abstract—IP-based wireless sensor networks (IP-WSN) are popularly recognized as a global sensor networks infrastructure by combining IPv6 technology with wireless sensor networks (WSN). However, IP-WSN, currently, does not provide security because it is commonly recognized that it is a big burden for IP-WSN to adapt Internet’s dominant security protocol. We suggest a security protocol for IP-WSN based on Secure Sockets Layer (SSL). We have developed IP-WSN sensor node hardware with 16-bit MSP430 MCU including 116 KB Flash and 8 KB RAM, and implemented IP-WSN software stack and lightweight SSL components. The resource consumption is around 64 KB of Flash and 7 KB of RAM when it is used that ECC for key-exchange and authentication, RC4 for data encryption, and MD5 for hashing. And it takes total 2 s for a full SSL handshake and a 127 bytes packet transfer. It can make IP-WSN secure and be used for the public and private services such as military observation, healthcare, and home-network.

Keywords-6LoWPAN; IP-WSN; SSL; Sensor Networks Security

I. INTRODUCTION

In the last several years, although many WSN technologies have been developed in the world, they were based on local networks. Therefore it is hard to merge and reuse those pre-constructed sensor networks for various services with different protocols as a larger scale network due to its heterogeneity. IP-WSN is popularly recognized as a global sensor networks infrastructure by combining IPv6 technology with WSN so that all the sensor nodes, equipped with IPv6 address and TCP/IP stack, can communicate with any peer through the Internet and take advantages of IPv6 technology like security, mobility, and so force. Also it shares a basic network stack and can be used irrespective of applications for services and is interoperable with other IP-WSN.

However, for the majority of public and private services of WSN such as military, home-network, and healthcare, security is important and should be supported. The global accessibility of IP-WSN through the Internet causes additional security issues, so IP-WSN needs a more reliable security protocol to prevent it from potential threats using more powerful devices having abundant resources like PC, laptop, and PDA. IETF 6LoWPAN working group [16] also recognizes the importance of security and actively discusses based on a draft for the security consideration [3], but still no solution has been suggested.

We suggest lightweight Secure Sockets Layer (SSL) [11] for IP-WSN security using ECC [5] instead of RSA for key-exchange and authentication, RC4 for data encryption, and SHA1 and MD5 hashing algorithms and present it in this paper. This lightweight SSL is small enough to be operated in the highly restricted device resources including 16-bit MCU, 116 KB Flash memory, and 8 KB RAM with embedded OS, IEEE 802.15.4 PHY/MAC, lightweight TCP/IP stacks and 6LoWPAN adaptation layer. It also provides basic security functions like privacy, integrity, authentication, access control, and non-repudiation; it can be interoperable with existing SSL protocol used in the Internet.

SSL is the most commonly used and verified security protocol on the Internet today. It is used for many popular services and applications including all kinds of well-known web browsers, e-mail, on-line banking, stock trading, and e-commerce [12]. SSL is a hybrid cryptography combining public-key cryptography for key-exchange and authentication with symmetric-key cryptography for data encryption and integrity. Public-key cryptography is widely believed to be beyond the capabilities of WSN devices. However, we have implemented IP-WSN and lightweight SSL software using around 64 KB Flash and 7 KB of RAM. And it takes total 2 s for a full SSL handshake and a 127 bytes packet transfer.

In this paper, we describe the results of our IP-WSN project named Sensor Networks for All-IP worLd (SNAIL), and present the Security of SNAIL (SSNAIL) using lightweight SSL. The remainder of this paper is organized as follows. Section II reviews related work. Section III introduces SNAIL. Section IV particularly describes SSNAIL. In Section V, we measure the resource consumption and evaluate the performance. Finally, we conclude with future work in Section VI.

II. RELATED WORK

A. 6LoWPAN security analysis

6LoWPAN security analysis draft [3] discusses possible threats and security options for 6LoWPAN. This document summarizes attacks and threats against user and data security in 6LoWPAN caused by its wireless radio access and connectivity to the Internet. However, it does not provide any solution although several real services were implemented using 6LoWPAN technology in the world. This paper may contribute to 6LoWPAN technology by suggesting a lightweight SSL which can be operated over 6LoWPAN network stack.

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R0A-2007-000-10038-0) and the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2009-(C1090-0902-0047))

†Corresponding author: College of Information Science & Technology, Korea Advanced Institute of Science and Technology, Associate Professor
B. OpenSSL

The OpenSSL [18] is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) security protocols and related cryptography standards required by them. It has been widely used for many related projects implementing the SSL.

However, RSA public key cryptography of OpenSSL is not suitable for IP-WSN since its resource is constrained and the size of OpenSSL library is over 1 MB.

C. MarixSSL

PeerSec Networks MatrixSSL [15] is an embedded SSL implementation designed for small footprint applications and devices. It has a around 50-70 KB footprint, but still is not small enough to be adapted into IP-WSN devices; targets 32-bit and 64-bit CPUs, for instance, ARM processor family, MIPS, PowerPC, i386, and x86-64. It also supports only RSA-based key-exchange, and on the IPC@CHIP platform, RSA decryption takes nearly 45s [14].

D. GoAhead WebServer

GoAhead WebServer [19] is a small footprint to help put a web server in every device under an "open source license". It supports embedded web technologies including Active Server Pages (ASP), embedded JavaScript, and in-process CGI. It also includes support for SSL, Digest Access Authentication (DAA). However, it only supports RSA and targets somewhat powerful platforms which can operate Windows XP, Windows CE, Linux, and other popular OS. In addition, 60 KB memory footprint is still quite large to run in IP-WSN devices.

E. Sizzle

Sizzle [12] is a SSL-based end-to-end security architecture for the embedded Internet. This shows that ECC makes public-key cryptography feasible on highly constrained embedded devices. It is a small footprint implementation of a HTTPS stack that brings the well-established security properties of SSL and a relatively small set of cryptographic algorithms to the embedded Internet. It uses MD5 and SHA1 hashing algorithms which is mandatory for SSL; RC4 for bulk encryption because of its high speed, compact code size, widespread support in existing SSL deployments; and the Elliptic Curve Diffie Hellman (ECDH) for key exchange [6] and the Elliptic Curve Digital Signature Algorithm (DCDSA) [7]. The idea using ECC with ECDH-ECDSA-RC4-MD5 is similar to this paper. Sizzle has proxy based architecture; TCP/IP connection terminates at the gateway; and sensor network using Sizzle uses its own protocol instead of IP-WSN standard so that Sizzle has no additional resource load by IP-WSN network stack including lightweight TCP/IP stack and 6LoWPAN adaptation layer.

F. IP-WSN

IP-WSN is a low-power wireless personal area network equipped with IP connectivity between sensor nodes and any IP-enabled devices including mobile handsets, PCs, servers, and other devices deployed in different networks as well as same networks using standard Internet protocols. A sensor node acquires a unique IPv6 address and connects to the Internet via IP-WSN gateway. IP-WSN is based on 6LoWPAN standards by IETF 6LoWPAN working group who standardized RFC4944 [1] and RFC4919 [2] and has been proposing various drafts. In addition, a new working group, ROLL (Routing Over Low power and Lossy networks) [17], is making efforts to provide an independent IP-based routing solution for low-power and lossy networks. Recently, many researchers and companies including ArchRock, Cisco, Dust Network, and other organizations are actively participating in standardization activities. Those working groups’ efforts are based on the fact that the size of a network stack, compressed packet length, and energy overhead are reasonable compared to ZigBee [21]. However, those working groups still have not solved the remaining security issues.

III. SNAIL

Sensor Networks for All-IP world (SNAIL) is a project to implement IP-WSN platform, make a widespread test-bed, and build real services in the planet. SNAIL includes compact IP-WSN software which can be used in highly constrained sensor node hardware with 16-bit MCU, 8 KB RAM, and 116 KB Flash. In this section, we describe our SNAIL platform in detail.

A. SNAIL sensor node

Our SNAIL sensor node runs on two different platforms which are OSAL (Operating System Abstraction Layer) [23] of TI solution and ANTS EOS [20] of RESL (Real-time and Embedded Systems Laboratory) in ICU. This paper handles implementation on the former platform.

1) Software:

SNAIL sensor node software meets RFC4944 whose subjects are "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [1]. As shown in the (a) of Figure 1, a 6LoWPAN adaptation layer locates between a network manager and a lightweight TCP/IP stack. The stack (a) in Figure 1 depicts the SNAIL sensor node network stack and boxes in white mean unimplemented components in this paper.

Regarding routing protocol, we have implemented HiLow [4] routing protocol based on width-first tree based algorithm due to its salability, convenient address structure to manage the large-scale sensor networks by using a 16-bit short address, and no need to maintain a routing table. The auto-configuration is performed in the process of so-called bootstrapping. The optimization of ND (Neighbor Discovery) of IPv6 has been discussed in the 6LoWPAN working group; though, it has not been standardized yet, neither has bootstrapping. We have designed our own
bootstrapping and network prefix advertisement mechanism. As shown in Figure 2, auto-configuration is performed just after an association process between two nodes and the joining node then registers itself to a PAN Coordinator so that the PAN coordinator can manage all the registered nodes in a PAN and advertise the network prefix periodically. From this method, the overhead of multicast messages of ND can be considerably reduced since only unicast messages are utilized in the process.

The components of our embedded TCP/IP stack support a lightweight IPv6, ICMPv6, TCP and UDP which are in gray in the (a) of Figure 1. Especially, ICMPv6 supports echo request/reply messages and neighbor discovery messages. For the security of SNAIL platform, we implemented a lightweight SSL protocol which supports MD5 and SHA1 hashing algorithms (those are mandatory in SSL standard); RC4 for data encryption because of its high speed, compact code size, and widespread support in existing SSL deployments; and ECDH-ECDSA for key-exchange. Lightweight SSL is located between TCP/IP stack and an application layer as can be seen in the (a) of Figure 1.

2) Hardware:

We have developed SNAIL sensor node hardware shown in the (a) of Figure 3. The sensor node hardware includes MSP430FG4618 MCU, 8KB RAM, 116KB Flash memory, and CC2420 RF. It is powered by two AA-sized batteries and contains one USB port for communication between coordinator and gateway, and debugging.

B. SNAIL gateway

The implemented gateway was designed to handle basic requirements of IP-WSN and further research topics with rich resources. We describe the details as follows.

1) Software:

The SNAIL gateway has three major roles: 1) IPv6 packet delivery from the Internet to the corresponding sensor node and vice versa; 2) network prefix handling; 3) network management. For the first role, the gateway supports an IPv6 stack at least as well as IPv4 stack since IPv-WSN is based on IPv6. Because IPv6 networks are not fully deployed throughout the world, IPv6 packets should also pass through the IPv4 networks. For this purpose, the gateway network stack is equipped with both IPv4 and IPv6 stack and facilitates 6to4 tunneling to support an IPv6 packet delivery via IPv4 networks. In addition, the gateway uses the same adaptation layer as the sensor node as shown in the (b) of Figure 1. Its objective is to compress and decompress the incoming and outgoing packets for transparent packet delivery.

The gateway helps the resource limited PAN coordinator node to manage the entire PAN in performing the second and third role. The coordinator is connected to the gateway with a serial cable to communicate each other so that it can request the gateway to handle work on behalf of it. The gateway as a first device retrieving a network prefix in both cases of IPv6 networks and 6to4 tunneling shares the network prefix with the PAN coordinator and periodically advertises a network prefix and notifies all the sensor nodes of its aliveness by using unicast messages. Likewise, maintaining all information of sensor nodes in the gateway makes it much easier to cope with network management works. For instance, development of network management applications and tools is far more convenient than retrieving node information from the PAN coordinator and information residing in the gateway is useful for numerous applications such as commissioning, neighbor discovery, time synchronization, SNMP, security, and other research subjects.

2) Hardware:

The developed SNAIL gateway hardware is shown in the (b) of Figure 3. The gateway hardware includes Embedded Intel Pentium processor, 512MB RAM, and 8GB CF memory to run Fedora core 7 and two USB ports for a serial communication between the gateway and the PAN coordinator and WLAN interface. The PAN coordinator is stacked on the gateway board to reduce volume.

<table>
<thead>
<tr>
<th>Type</th>
<th>Sensor node</th>
<th>Gateway</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>8MHz TI MSP430</td>
<td>Embedded Intel Pentium M/Celeron M</td>
</tr>
<tr>
<td>RAM</td>
<td>8 KB</td>
<td>512 MB Memory</td>
</tr>
<tr>
<td>Storage</td>
<td>116 KB Flash</td>
<td>8 GB Memory</td>
</tr>
<tr>
<td>Network</td>
<td>IEEE 802.15.4 RF</td>
<td>2 X 10/100 Mbps Ethernet</td>
</tr>
<tr>
<td>I/O</td>
<td>RS232C</td>
<td>RS232C, LPT</td>
</tr>
<tr>
<td>USB</td>
<td>USB 2.0 compliant Port</td>
<td>2 X USB 2.0 compliant Port</td>
</tr>
<tr>
<td>Power</td>
<td>AA size battery</td>
<td>ACPI support</td>
</tr>
</tbody>
</table>
IV. SSNAIL

The security technology for IP-WSN should be small and operated in the highly restricted device resources including 8 or 16-bit MCU, around 100 KB Flash memory, and less than 10 KB RAM with embedded OS, IEEE 802.15.4 PHY/MAC, lightweight TCP/IP stacks and 6LoWPAN adaptation layer. It also must provide basic functions like privacy, integrity, authentication, access control, and non-repudiation; and it can be interoperable with existing security technologies used in the Internet.

As shown in Table 1, SNAIL sensor node hardware has 16-bit MCU, 8 KB RAM, and 116 KB Flash. The size of SNAIL software is 48 KB and it uses around 7 KB RAM with 1280 bytes full TCP/IP fragment and reassembly. We have tried to implement lightweight SSL whose code size is under 30 KB and memory occupation is under 500 bytes to be operated in this SNAIL platform because SSL is the most commonly used and verified security protocol in the Internet today and used for many popular services and applications including all kinds of well-known web browsers, e-mail, on-line banking, stock trading, and e-commerce. SSL is a hybrid cryptography combining public-key cryptography for key-exchange and authentication with symmetric-key cryptography for data encryption and integrity. Public-key cryptography is widely believed to be beyond the capabilities of WSN devices. However, we have designed lightweight SSL using ECC instead of RSA for key-exchange and authentication, RC4 for data encryption, and SHA1 and MD5 hashing algorithms.

A. SSNAIL architecture

While SSL has been verified and identified as a good security solution in the Internet, it has been recognized as being too "heavy-weight" for highly restricted sensor networks devices due to its public-key cryptography [12]. IP-WSN, by contrast, is designed to make global sensor networks equipped with IPv6, so that it has more load by the technology to harness IPv6 to WSN. SSNAIL is a part of SNAIL project to implement lightweight SSL which can be operated in the highly constrained devices with 30 KB Flash and 500 bytes RAM usage.

Figure 4 shows the SSNAIL architecture we have implemented. It is an IP-WSN which provides end-to-end secure communication to any peer in the Internet. The sensor node has a globally unique IPv6 address and IP-WSN stacks including ECC based lightweight SSL which can be worked as both server and client side SSL. Therefore, sensor node can securely communicate with others in the intra-PAN (Personal Area Network) and inter-PAN through the Internet.

The gateway plays the same role as the SNAIL gateway described in Section III. It delivers IPv6 packets including SSL protocol messages from the Internet to the corresponding sensor node and vice versa.

The PAN coordinator is connected to the gateway with a serial cable to communicate with each other so that all the peers in the IP-WSN and Internet can securely communicate via the gateway using TCP/IP and SSL.

The host and monitoring system are in the Internet, but usually applications used in the Internet like well-known web browsers do not support ECC based SSL today, so that we have developed a simple monitoring program on which ECC based SSL is enabled.

B. ECC-based Lightweight SSL

SSL allows applications to communicate across a network in a way designed to prevent eavesdropping, tampering, and message forgery. SSL provides endpoint authentication and communications privacy over the Internet using cryptography. It is flexible enough to accommodate different cryptographic algorithms for key-exchange, encryption and hashing.

We have implemented a relatively small set of cryptographic algorithms. We chose the ECDH-ECDSA based ECC for key-exchange and authentication due to its resource efficiency and speed. We selected RC4 for data encryption due to its compact code size and speed. The MD5 and SHA1 hashing algorithms which are mandatory for SSL are implemented for the derivation of the master secret and symmetric keys. In addition, we implemented RSA, DES, and 3DES cipher suits for the code size, memory usage, and performance comparison. It can be chosen and built at the compile time. Our lightweight SSL meets SSL 3.0 standard [11].

1) Lightweight SSL components

Figure 5 shows the software components of our lightweight SSL included in the SNAIL stack. Its record protocol involves using SSL in a secure manner and with message integrity ensured. To this end it is used by upper layer protocols. The purpose of the record protocol is to take an application message to be transmitted, fragment the data which needs to be sent, encapsulate it with appropriate headers and create an object just called a record, which is encrypted and can be forwarded for sending under the TCP protocol. It takes intensive time and power to encrypt and hash messages so that we selected RC4 for encryption and MD5 for hashing.

The alert and cipher change protocol are simple message handling protocols meeting SSL standard. The first one is to convey session messages related with data exchange and functioning of the protocol. And second one is to cause the pending session state to be established as a fixed state. These messages and any other SSL messages are transferred.
The handshake protocol constitutes the most complex part of the SSL protocol. It is used to initiate a session between the server and the client, authenticate the parties to each other, and negotiate cryptographic algorithms and keys. For the key-exchange and authentication, we chose ECDH-ECDSA operation of ECC using the 160-bit curve secp160rl which provides the same level of security as the RSA using 1024-bit key for the faster computations and lower memory and power consumption. According to Gura et al. [9], 160-bit ECC key-exchange operations are 13 times faster than 1024-bit RSA decryption operations on the "mote".

2) ECC-based full handshake

The (a) of Figure 6 shows the full handshake operation based on ECDH-ECDSA, as specified in [8]. The client sends the server a ClientHello (a-1) message containing cipher suite including ECDH-ECDSA, RC4, and MD5 as well as version, random, and session ID. The server, in response to the ClientHello message sends a ServerHello (a-2) message, containing the same set of fields as the ClientHello message so that the parties negotiate a cipher suite. The server’s Certificate (a-3) message contains the server’s ECDH public-key signed by a certificate authority using ECDSA. If the server’s certificate is correctly verified with the ECDSA signature validation the server is considered authenticated, and then the client sends the server ECDH public-key using the ClientKeyExchange (a-4) message. Next, the client sends a ChangeCipherSpec (a-5) message to confirm the messages so far received and to verify whether the pending data is correct, and then sets up the pending set of algorithm parameters and keys into the current set of the same. Then the client sends the finished message, which is first protected with just negotiated algorithms, keys and secrets. This is to confirm that the negotiated parameters and data are correct. The server in response to the client sends the same message sequence. If the finished message is correctly read by either party this confirms that the transmitted data, negotiated algorithms and the session key are correct. After that, each party uses its own ECDH private-key and the other’s public-key to perform an ECDH operation and arrive at a shared premaster secret. Both entities then use the premaster secret to create a master secret which is used to derive cipher keys, initial vector, and Message Authentication Code (MAC) keys for data encryption and authentication by the record protocol.

3) Abbreviated handshake

As shown in (b) of Figure 6, the client sends a ClientHello (b-1) message including the non-zero session ID of the session to be resumed. If the server then has that session information cached and is willing to re-establish the connection under the specified session state, it will send a ServerHello (b-2) message with the same Session ID value. If a session ID match is not found, the server generates a new session ID and a full handshake is performed.

V. EVALUATION

The SSNAIL test-bed has been built in the KOREN [22] which is a non-profit research network providing development environment for IPv6 research. To evaluate the feasibility of deployment both in IPv4/IPv6 networks, we located a remote test host in IPv4 networks outside the KOREN and IP-WSN in IPv6 networks of KOREN. 30 sensor nodes were deployed at intervals of more than 5 meters in an open area with no obstacles.

A. Memory Usage

We used IAR Embedded Workbench IDE to debug and determine memory usage of SSNAIL software. IAR shows the code memory and static data memory at the build time. We also calculated allocated heap at the code level without considering dynamic situations caused by packet sizes and frequencies of applications. We experimented with just SNAIL sensor nodes. And we fixed static RAM size as 7 KB and heap size as 1KB for SNAIL implementation. SNAIL basically uses around 7 KB static RAM with 1280 full TCP/IP fragment and reassembly and 422 bytes of heap. Table 2 presents memory usage measurements for our SSNAIL sensor node software depending on the cipher suites enabled. These numbers include memory consumed by OSAL, IEEE 802.15.4 PHY/MAC, and IP-WSN software stack; and also include cryptographic algorithms and lightweight SSL components excepting the numbers in the first and second rows for SNAIL. The maximum code size with RSA-3DES-MD5-SHA1 is roughly 64 KB, so
there is plenty of room in Flash memory to implement additional and even more complex software. However, the maximum heap usage with same cipher suite is over 1 KB. Therefore, we changed TCP/IP fragmentation and reassembly buffer size to 512 bytes and reduced static RAM limit to 6.5 KB and extended heap size to 1.5 KB. It is sure that the TCP/IP fragmentation and reassembly with reduced buffer decreases the packet processing capability. ECC-RC4-MD5-SHA1 cipher suite takes the least RAM consumption, which is 6,936 bytes. If the TCP/IP buffer size is set to 1280 bytes, it takes a total 7,704 bytes.

B. Performance test of SNAIL

The performance of SNAIL is measured by taking average round-trip time in hops for 100 ICMPv6 and UDP echo request/reply packets with which full test data is filled up to 127 bytes; i.e. the maximum frame size of IEEE 802.15.4. As shown in the (a) of Figure 7, the average round-trip time in hops for both ICMPv6 and UDP slightly increases as the number of hops grows. According to our analysis, two major factors affecting the average round-trip time were an asynchronous I/O operation to multiplex both serial and virtual tunneling interface inputs and the sum of transmission rate between the KOREN and a 6to4 relay server and between the remote host and the relay server.

C. Performance test of SSNAIL

The performance of SSNAIL is measured by taking average time in hops for a SSL handshake and 100 round-trip TCP packets with which full test data is filled up to 127 bytes. As shown in the (b) of Figure 7, the average time for an ECC based full handshake and 127 bytes packet round-trip is 2 s. The experiment using an abbreviated handshake takes around 1 s. And, as expected, an RSA case takes 8.5 s.

VI. CONCLUSION AND FUTURE WORK

We have presented a lightweight security protocol for IP-WSN. We have developed ECC based lightweight SSL for the security of IP-WSN in addition to SNAIL platform with OSAL, IEEE 802.15.4 PHY/MAC, lightweight TCP/IP, and 6LoWPAN adaptation layer. The minimum resource consumption is around 64 KB of Flash and 7 KB of RAM when it uses ECC for key-exchange and authentication, RC4 for data encryption, and MD5 for hashing. In addition, it takes 2 s for a SSL full handshake and 127 bytes packet round-trip. This paper contributes to the 6LoWPAN and IP-WSN related research and extends IP-WSN service areas to more critical ones such as military, health-care, home-network. For the future work, we will design a low-power mechanism for our SNAIL platform and analyze the result when the lightweight SSL is operated over UDP because TCP increases the traffic and the power consumption and UDP is enough for the majority of WSN services.

REFERENCES