Secure Data Collection with Cryptography for Wireless Sensor Network on a Simulator

2013SE265 EUGENE TAN Chuen Liang
Supervisor: GOTO Kunio

1 Introduction

Development of various areas of Information and Communication Technology (ICT) has led to an explosive growth in volume of data. The term Big data has become a popular word in the last few years. Organizations started to analyze data to make better decisions for business moves. One of the characteristics of Big Data, "variety" has led us to WSN (Wireless Sensor Network). Gathering the large volume and wide variety of sensed data, it is unable to ignore the security risk of information leakage. Security in WSN has always been a challenge[4]. Therefore the cryptography complexity needs to be minimize to fit these restrictions.

The goal of this paper is to construct WSN in OMNeT++ general simulator with encryption communication on Ubuntu 14.04.5 LTS Linux version 3.13.0 64bit. We will use an extension INET framework of OMNeT++ that provides protocol implementations to construct WSN and decide transmission details. For encryption we have choosed NTRU public key cryptography, a lattice-based public key cryptosystem for security innovation [2], because NTRU encryption implementations is a compact and low cost that is suitable for sensors. And for easy use of NTRU encryption we implemented C++ class definitions. We will establish a general method to implant encryption into OMNeT++. With this technique, future WSN simulation project to be more secure and efficient and can be applied with other cryptosystem.

2 Simulators for Wireless Sensor Network

In this section, we discuss about the network simulator to the base of this research.

2.1 OMNeT++ and INET Framework

OMNeT++ is a discrete event network simulator built in C++, a general network simulator. And OMNeT++ can support protocols and simulate power consumption in WSN using INET Framework extension.

INET Framework is an open-source model library for the OMNeT++ simulation environment. It provides protocol implementations and several application models for communication networks. We will use the following version; omnetpp-5.0, inet-3.4.0.

3 Secure data collection

In this section, we will discuss about the cryptography used for the simulation in OMNeT++.

WSN poses challenges such as strict resource constraints on each individual sensor, lack of processing capability, etc. Therefore the cryptography complexity needs to be minimized to fit these restriction.

3.1 NTRU public key cryptography

The NTRU cryptosystem is patented by NTRU CRYPTOSYSTEMS(http://www.ntru.com) is one of the fastest public-key encryption schemes known today [3]. The cryptographic strength is equivalent to RSA, and NTRU has a faster private key operation. NTRU security is based on the hardness of the Shortest Vector Problem (SVP) in a very high dimension lattice [1]. It still uses relatively large operands, but it reduces the overall asymptotic complexity of the encryption operation to $O(n^2)$ compared to RSA's $O(n^3)$ [2].

3.2 NTRU Program

To begin NTRU cryptography you would need to install the NTRU Library. We have downloaded the C++ library from "https://github.com/tbuktu/libntru". According to README.md in the source file, we used NTRU_DEFAULT_PARAMS_256_BITS for the program, encryption block size not more than 106 octets and encrypted output is a fixed size of 1022 octets.

Listing 1 NTRU encrypt and decrypt

```
1 static void generateRandomKey (std::string
      keyBaseName);
2 static uint8_t* encrypt(const uint8_t* input,
      int ilen, std::string keyBaseName, int*
      outLen);
3 static uint8_t* decrypt(const uint8_t* input,
      std::string keyBaseName, int* outLen);
```

The NTRU library implementations codes was complicated, therefore we have implemented an C++ class definitions for easy use listed in Listing 1. First, generateRandomKey will generate a binary file; public key and private key. The file name is decided with the inputted string in keyBaseName.

In encrypt, input is the input message in 8-bit unsigned integer, and ilen the length of the input message in bytes, keyBaseName the public key, outLen the length of the encrypted result in bytes.

As for decrypt, the process are similar, input, keyBaseName, outLen has the same function. The differences is that it does not have ilen, the length of the message in bytes. This is because, the length of the message before encryption is the same.

4 System Implementation

Figure 1 is the goal network image in GUI.

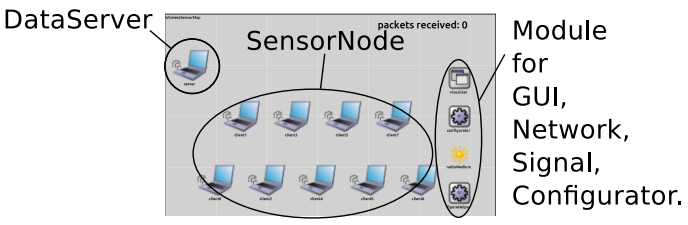

Figure 1 Simulation Network

To construct this network, the following files need to be created. (Only important files are listed; encryption and communication process.) All created files is placed in the same arbitrary directory.

1. WirelessSensorMap.ned 2. omnetpp.ini
3. DataServer.cc 4. Sensor.cc
5. UDPMessage.msg 6. PayloadFormat.h

First, ini file for network configuration, NED (Network Description) file is to describe structure of the simulation model. And cc files for communication process.

Listing 2 is the automatic generation command for Makefile and run command. MYPROG is the path in your computer from home to inet/src. And arbitrary directory is automatically generated with opp_makemake.

Listing 2 Makefile Generation and Run Command

```
1opp_makemake -f -I MYPROG/src(INET include path
    ) -lntru(External library) -L MYPROG/src(
    INET library path) -lINET
2
3./arbitrary directory -n MYPROG/inet/src:.
```

Figure 2 is a sequence diagram for the communication content in the simulation.

After all Sensor node and DataServer has initialized, DataServer will generate tickets and send out to all nodes and starts the ticket valid timer. When individual SensorNode receives the ticket, they will send back an acknowledgement and begin sending encrypted messages while the ticket is valid. Finally, DataServer will regenerate a new ticket and the process repeats.

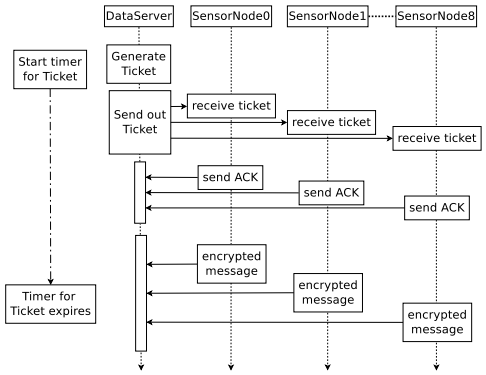

Figure 2 Communication Content Sequence Diagram

Algorithm 1 is the overall process of DataServer. Ticket generation, ticket encryption and delivery and decryption.

Algorithm 1 DataServer process

```
1: if timer1 then
2:     ticket generation and destination decision then
3:     ticket send process and encryption
4: end if
5: if timer2 then
6:     check for ticket arrival and re-transmit
7: end if
8: if receive UDP data then
9:     process receive and decrypt message
10: end if
```

Algorithm 2 is the overall process of Sensor. Receive encrypted packet from DataServer and respond back. In the encrypted message contains parameters; temperature, etc.

5 Results

Implementation according to Figure 2 have caused severe packet collisions. Therefore, we have re-implement

Algorithm 2 Sensor process

```
1: if receive packet from server then
2:     decrypt and store ticket
3:     send encrypted message(parameters; temperature, etc)
       with ticket
4:     update value(parameters; temperature, etc)
5: end if
```

to ticket and sending process to individually. And according to the reference implementation of NTRU, maximum encryption length is 104 octets. Any messages longer than this will be fragmented and reassemble on the receiving end.

Listing 3 is the result of the encryption process.

Listing 3 ticket and message transmission

```
-- to: client0:60000 ticket 825198428,
    expiration 22, encrypted in 1022 octets
client0 Received packet from server(10.0.0.1)
payloadlen 1022
server message(decrypted): plen 10, ticket
    825198428, expiration 22
client0 sending data with ticket 825198428
-- sendint to server, length(raw)/(enc) 47/1022
DataStr = t 13.415423, h 2.682374, e 32.081550
server Received packet from client0(10.0.0.2),
    plen 1022
-- decrypted len = 47
header:  totalfrags 1, plen 47
ticket 825198428, dataID 0, fragSeq 1
DataString t 13.415423, h 2.682374, e 32.081550
```

6 Conclusion

We have managed to implement WSN in OMNeT++ simulator with an extension of INET framework and using NTRU public key cryptography C++ library as encryption. We have successfully establish UDP encrypted communication with NTRU in OMNeT++.

Future challenges is to apply the method in this paper to other encryption such as DTLS protocol to further encryption strength. And the current transmission order for this network is based on simple acknowledgement transmission. Which is to verify NTRU public key encryption process on all transmission. Therefore this need to be reconsider for smother communication and error scenario for abnormal cases suited for wireless sensor network. A communication scenario more suited for wireless sensor network.

References

[1] Gaubatz, G., Kaps, J. P., Ozturk, E. and Sunar, B.: State of the Art in Ultra-Low power Public Key Cryptography for Wireless Sensor Network, *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 1–5 (2005).

[2] Gaubatz, G., Kaps, J. P. and Sunar, B.: Public Key Cryptography in Sensor Networks—Revisited, *Security in Ad-hoc and Sensor Networks, Lecture Notes in Computer Science*, Vol. 3313, pp. 2–18 (2004).

[3] Nguyen, P. Q. and Pointcheval, D.: Analysis and Improvements of NTRU Encryption Paddings, *Lecture Notes in Computer Science - Advance in Cryptology*, Vol. 2442, pp. 210–225 (2002).

[4] Perrig, A., Tankovic, J. and Wagner, D.: Security in wireless sensor networks, *Communications of the ACM*, Vol. 47, pp. 53–57 (2004).