

Some Security Considerations over Contiki-based Sensor Network

Yan Yan

December 15, 2015

Contents

1	Introduction	2
1.1	Related Work	2
1.2	Experiment Setup	2
1.3	Adversary Power	3
1.4	Types of Packets	3
2	Link Layer Security	5
2.1	802.15.4 Security: <i>noncoresec</i>	5
2.2	Weak IV	6
2.2.1	Reset Problem	7
2.3	Distinctive packet length for RPL packets	8
2.4	Performance issue	8
3	DTLS	9
3.1	Implementation Issues	9
3.2	No multicast support	10
3.3	Overloading DTLS with LLSEC	10
4	Application Detection	11
4.1	Network Protocol Headers	11
4.2	Packet Length	11
4.3	Timing Packet Response	11
4.4	PINGLOAD: PING side-channel for Payload	13
4.4.1	Hypothesis	13
4.4.2	Experiment Result	15
4.4.3	General Hypothetical Model	15
5	Fulture Work	16

Chapter 1

Introduction

This paper discusses two security measurements, namely Link Layer Security (LLSEC) and Datagram TLS (DTLS), within Contiki OS.

1.1 Related Work

[1] discusses some security concerns in 802.15.4. LLSEC[2] is the implementation of 802.15.4 security in Contiki.

tinydtls[3] is the implementation of DTLS we used in DTLS related experiments.

1.2 Experiment Setup

All experiments are done within the Cooja simulator.

The setup is as described in Figure 1.1.

- **Adversary** is the malicious party that tries to recover information from the encrypted traffic.
- **Border Router**, or BR, is a device that connects the adversary to the sensor network. However, **BR is not allowed when LLSEC is enabled** as the adversary does not have the key and hence cannot connect into the network.
- **Sniffer** is a device that passively captures all traffics in the sensor network.
- **Target** and **Nodes** are sensors deployed in the sensor network. They communicates to each other through encrypted channels.
- **Sensor Network** discussed in this paper is a 6LowPAN network.

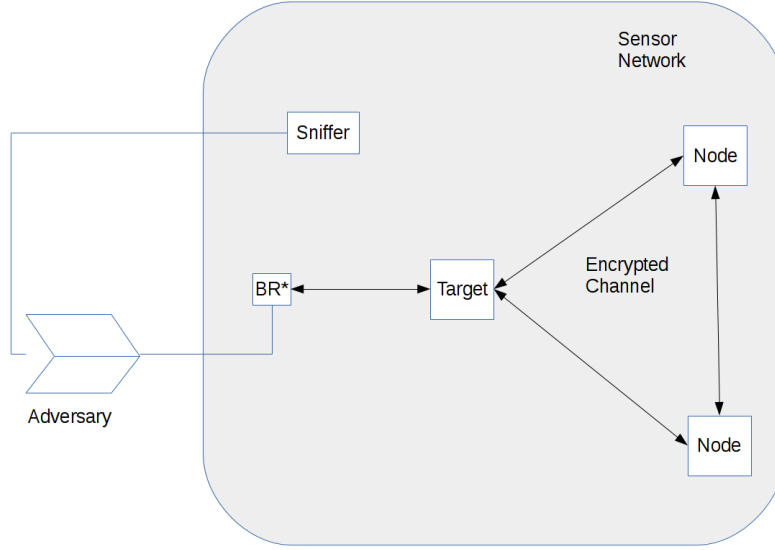


Figure 1.1: Experiment setup

1.3 Adversary Power

The powers assumed in the experiments are considered to be practical in real life.

When LLSEC is enabled, all traffic, including RPL¹ messages, are encrypted; therefore no external nodes can connect to the network. The only power for the adversary is to passively sniff all the traffic.

In other cases where LLSEC is disabled, the adversary will be enabled to join the sensor network through a BR and hence is also capable to send ICMP messages to the target(s).

1.4 Types of Packets

We simply categorise the packets into two types:

- **Network Management Packets:** These are the packets generated by the protocols those maintains the network, such as MAC ACKs, RPL messages or ICMP messages.
- **Data Packets:** These are those packets generated by the applications running on the nodes., such as a CoAP packet.

¹Routing Procol for Low-power and Lossy Networks

This is only a subjective rough categorisation and may not be precise. For example an TCP data packet may also serves as an ACK, or DTLS handshake packets could fall into both categories. However, we ignore this ambiguity as it is not our focus.

Chapter 2

Link Layer Security

Link Layer Security, or LLSEC, is a security measure that implements cryptography just above the physical layer.

Introducing cryptography at a lower level has several benefits. Firstly, more data being encrypted reduces the observable packet features to an adversary, such as SRC¹ and DST² field in the IP header which are very likely to be exploited by the adversary. Secondly, authentication at lower level also prevents an active adversary from joining the network which therefore weakens his power.

Imposing cryptography at a lower level also brings more challenge to the design of sensor network architecture. The first problem is its overhead. Even for a node that only tries to retransmits the packet to its next hop, it must decrypt the whole packet to extract its routing information, and then re-encrypt it before retransmission. This is particularly problematic in a mesh wireless sensor network as it could potentially lead to performance and energy consumption problems. Key management is also challenging due to the lossy and power optimised nature of wireless sensor network.

It is also noticeable that some packet features are not hidden even with LLSEC enabled, such as packet length, timing information and part of the MAC header.

2.1 802.15.4 Security: *noncoresec*

noncoresec[2] is the current implementation of LLSEC in Contiki. It corresponds to the AES_CCM_16 ciphersuite in 802.15.4 standard. This section briefly describes how it works.

- **Key Management:** All nodes share a network wide AES key for both encryption and authentication. The key is hardcoded during the setup stage.

¹Source Address

²Destination Address

Flags(1)	Addresses(8)	Frame Counter(4)	Security Level(1)	Block Counter(2)
----------	--------------	------------------	-------------------	------------------

Table 2.1: IV of 802.15.4 Frame with Security

- **AEAD**³: *noncoresec* implements AES_CCM_16⁴ as described in 802.15.4[4]. CCM mode turns AES into a stream cipher. The same key is used for both encryption and authentication.
- **Initial Vector (IV, or nonce)**: The IV for each packet is constructed from certain fields of unencrypted MAC frame header and therefore is public.

An adversary without the knowledge cannot join the sensor network as he cannot sent out a valid RPL message.

2.2 Weak IV

One problem within the *noncoresec* implementation is the low variance of IV. The IV is a 16 byte bit-string constitutes of the following fields(Table 2.1):

- **Flags (1 byte)**: This field contains part of the MAC frame header. It is identical to most (basically all) of the data packets.
- **Source Address (8 bytes)**: This is mapped from the source address field of the frame.
- **Frame Counter (4 bytes)**: This field increases by 1 for each frame sent.
- **Security Level (1 byte)**: This field indicates which ciphersuite to be used for this frame. In the case of AES_CCM_16, this is constantly 0x7.
- **Block Counter (2 bytes)**: This field begins from 0x0 and increases by 0x1 for each block in CCM mode. The block length for AES-128 is 16 bytes. The 2 bytes counter is sufficient as it supports up to 2^{32} bytes of data whereas the minimum MTU⁵ required by 6lowPAN standard[5] is 127 bytes.

In the current *noncresec* implementation, **Flags** and **Security Level** are constant. **Block Counter** always begins from 0x0 and the **Source Address** is also constant for a specific device. Such design leaves the 4 bytes **Frame Counter** the only field that is variable. This indicates that only 2^{32} messages are allowed which is cryptographically considered to be inappropriate.

³Authenticated Encryption with Associated Data

⁴CCM mode of AES-128 with 16 bytes MAC

⁵Maximum Transmit Unit, simply speaking this is the maximum length of a packet.

As we can see, the difference of ciphertext is exactly the difference of plaintext:

$$\Delta p = [00127401]_{16} \oplus [12262269]_{16} = [12345678]_{16} \quad (2.1)$$

2.3 Distinctive packet length for RPL packets

Some RPL packets are shorter than the minimum length of data packets which can be used to distinguish the packets.

2.4 Performance issue

The header overhead with LLSEC enabled is 20 bytes which is relatively a large overhead comparing to the 127 bytes MTU requirement of 6LowPAN standard[5].

Chapter 3

DTLS

DTLS has potentially the best interoperability as it is an variation of the widely used TLS in Internet. However, its design might not fit into the nature of WSN for practical reasons.

3.1 Implementation Issues

tinydtls[3] currently supports two ciphersuites, namely TLS_PSK_WITH_AES_128_CCM_8 and TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8.

However, we encountered several difficulties when trying to set up a network with DTLS.

Low Computational Power

Curve computation requires relatively a large amount of computational power. Even using a relatively power platform (CC2538), it still takes minutes to complete a DTLS handshake with TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8.

Low Bandwidth

The 6LowPAN standard specifies that the minimum MTU is 127 bytes whilst 67 (87 with LLSEC) bytes are occupied by protocol headers until UDP, which leaves 60 (40 with LLSEC) bytes available for UDP layer payload. This value can be easily exceeded even during the handshake, such like using a longer key. Some attempts have been made to solve this issue, e.g. CoDTLS[7]¹.

Code Size

The tinydtls fails to fit into some devices, e.g. skymote, as its size of code is too large.

¹This draft has been abandoned for some reason we do not know.

Therefore although TLS_PSK_WITH_AES_128_CCM_8 is less flexible (and probably less secure) as it uses a pre-shared master secret than TLS_ECDHE_ECDSA_WITH_AES_128_CCM it is still considered to be a relatively practical security measure as it requires less resources.

3.2 No multicast support

Some application protocols, such as CoAP, utilises the multicast feature of 6LowPAN whilst TLS is a protocol designated for securing communications between two parties, so is DTLS.

3.3 Overloading DTLS with LLSEC

Adopting both security measures at the same time is possible as they are implemented at different layers. However, it is questionable whether this will bring more security, as both *noncoresec* and TLS_PSK_WITH_AES_128_CCM_8 are using 128 bit AES with CCM mode as their cryptographic primitive.

Chapter 4

Application Detection

Similar to website fingerprinting, we try to identify the application running on target node by its traffic. This chapter discusses some general idea without a specific application.

4.1 Network Protocol Headers

Since most information in MAC¹, IP and UDP headers are related to routing and network maintenance and thus independent except the length fields and CRC².

4.2 Packet Length

Packet length is usually the most interested target in traffic analysis. However, packet lengths are also highly application dependant; thus we are not pursuing this topic further without a specific application.

4.3 Timing Packet Response

Unlike web applications where the client and server are usually physically distant, sensor networks can sometimes be located in a concentrated area, such as a house where its radius can be less than 10 meters.

These features theoretically enable one to capture all traffic in such a sensor network. As opposed to the case of Internet where packets are usually timed on

¹Media Access Control, not to be confused with the cryptographic term Message Authenticate Code.

²Cyclic Redundancy Check, a code to detect or correct transmission error

```

1961196 1 - 108: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|ICMPv6 ECHO REQ 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
353637
1961200 1 - 108: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|ICMPv6 ECHO REQ 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
353637
1961205 1 - 108: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|ICMPv6 ECHO REQ 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
353637
1961209 1 - 108: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|ICMPv6 ECHO REQ 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
353637
1961213 1 - 108: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|ICMPv6 ECHO REQ 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
353637
1961222 2 1 5: 15.4 A
1961230 2 1 107: 15.4 D 00:00:00:00:00:00:00:00:02 00:00:00:00:00:00:00:01|IPHC|IPv6|ICMPv6 ECHO RPLY 0|276D04A7
21793B56 00000000 34D40600 00000000 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34
4353637
1961234 1 2 5: 15.4 A
1961861 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961864 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961868 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961872 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961875 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961879 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961882 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
1961886 1 - 85: 15.4 D 00:00:00:00:00:00:00:00:01 00:00:00:00:00:00:00:02|IPHC|IPv6|UDP 45338 20220|0029EE70 1
7FEFD00 01000000 0004D400 14000100 00000004 D42B9B78 CC274913 1BD578FF A1
-- VISUAL --
3 53401,236-248 99%

```

Figure 4.1: Capture of a ping packet

the client’s side and thus network latency (RTT^3) must be concerned, being able to capture all traffic in the network provides much more accurate timing information and hence may be exploited to develop more efficient attacks.

Definition 4.3.1. In a Request-Response application model, **RI, Response Interval**, is defined as the interval between a request packet being received and its response being sent.

Example 4.3.1. Three packets are marked out in Figure 4.1 which forms an instance of ICMP ECHO[8] (also known as PING) session. The extracted packet features are displayed in Table 4.1.

Explanation of the Packets:

The first packet is an ICMP ECHO request and the third packet being its

³Round-Trip Time

Time (ms)	From (ID)	To (ID)	Length (bytes)	Type
1961218	1	2	108	ICMP ECHO
1961222	2	1	5	802.15.4 ACK
1961230	2	1	107	ICMP ECHO

Table 4.1: Packet Features of an ICMP ECHO request and response

response. The second packet is a 802.15.4 ACK⁴ and is thus transparent to the upper ICMP protocol.

From this example we can see that the RI for this PING session is:

$$1961230 - 1961218 = 12(\text{ms})$$

Timing information can be exploited by several attacks, such as [9] and [10].

We have experimentally measured a RNG⁵ call on Wismote platform in the Cooja simulator is roughly 0.3 ms.

4.4 PINGLOAD: PING side-channel for Payload

Support of ICMP ECHO is required by [8] and is also enabled in Contiki OS by default. However, our experimental results shows that the response time of these ping packets could potentially be exploited to reveal the application running on target sensor node.

We call such technique **Application Fingerprinting**.

4.4.1 Hypothesis

A phenomenon we realised is that when a ping packet arrived while the target node is executing some payload, say reading a sensor or processing data, the PING RI begins to vary comparing to a stable value when no there is no payload.

Example 4.4.1. Figure 4.2 shows RIs of PING collected in two experiments. The left half are collected with the target is constantly idle whilst the right half occasionally receives a request which triggers the target to call RNG. We can see that the PING RI varies alongside the target is given some payload.

⁴This is an acknowledgement from the receiver that notifies the sender that the packet has been received.

⁵Random Number Generator

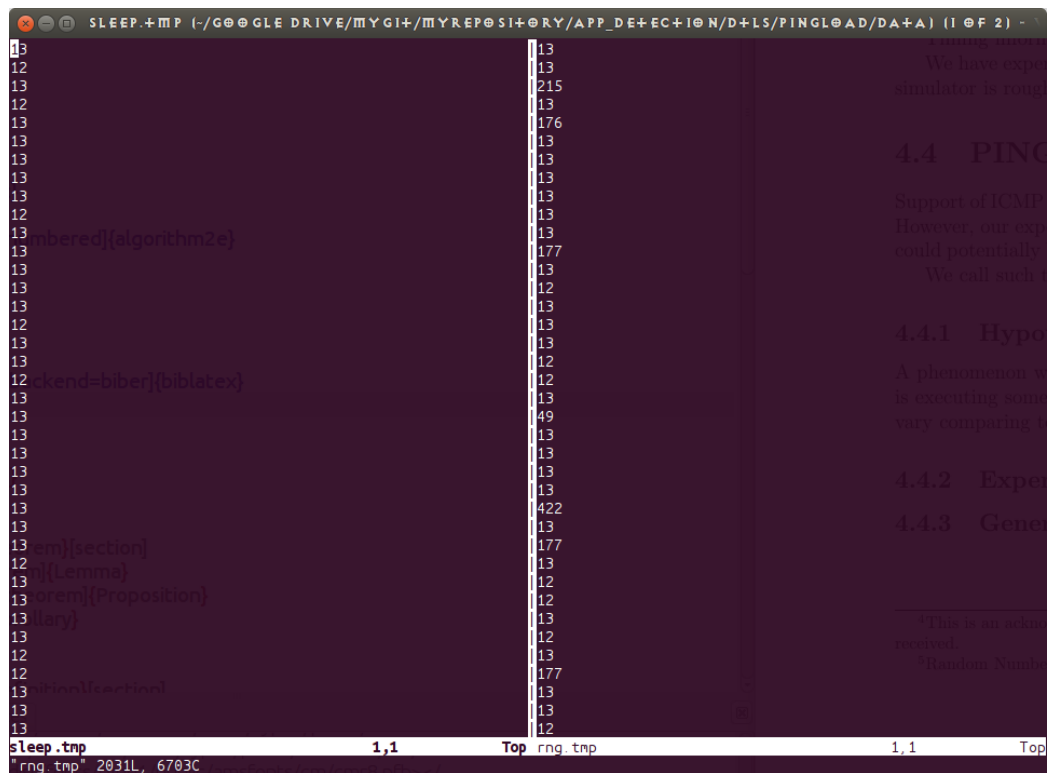


Figure 4.2: An example of PING’s RIs with different payload

The data shown in Figure 4.2 suggests that the “plain”, that is without any interference, PING RI is 12 to 13 ms. Further more, those variations of PING RI is very likely caused by the payload of the target.

This experiment inspirits that the distributions of PING RIs might vary according to the payload of target and could possibly considered as an fingerprint of the target’s application. In other word, an adversary could possibly tell whether the target is running a specific application by looking at its PING RIs distribution.

DESCRIBE HOW TO DO THE ATTACK HERE

4.4.2 Experiment Result

4.4.3 General Hypothetical Model

Chapter 5

Fulture Work

Bibliography

- [1] Naveen Sastry and David Wagner. “Security Considerations for IEEE 802.15.4 Networks”. In: *Proceedings of the 3rd ACM Workshop on Wireless Security*. WiSe ’04. Philadelphia, PA, USA: ACM, 2004, pp. 32–42. ISBN: 1-58113-925-X. DOI: 10.1145/1023646.1023654. URL: <http://doi.acm.org/10.1145/1023646.1023654>.
- [2] URL: <https://github.com/kkrentz/contiki/wiki>.
- [3] URL: <http://sourceforge.net/projects/tinydtls/>.
- [4] *IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. Tech. rep. 2006, 0.1–305. DOI: 10.1109/ieeestd.2006.232110. URL: <http://dx.doi.org/10.1109/ieeestd.2006.232110>.
- [5] G. Montenegro et al. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944 (Proposed Standard). Updated by RFCs 6282, 6775. Internet Engineering Task Force, Sept. 2007. URL: <http://www.ietf.org/rfc/rfc4944.txt>.
- [6] Adam Dunkels. *The ContikiMAC Radio Duty Cycling Protocol*. 2011.
- [7] Carsten Bormann Lars Schmertmann Klaus Hartke. *CoDTLS: DTLS handshakes over CoAP*. 2015. URL: <https://tools.ietf.org/html/draft-schmertmann-dice-codtls-01>.
- [8] R. Braden. *Requirements for Internet Hosts - Communication Layers*. RFC 1122 (INTERNET STANDARD). Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864. Internet Engineering Task Force, Oct. 1989. URL: <http://www.ietf.org/rfc/rfc1122.txt>.
- [9] K.P. Dyer et al. “Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail”. In: *Security and Privacy (SP), 2012 IEEE Symposium on*. May 2012, pp. 332–346. DOI: 10.1109/SP.2012.28.

- [10] Paul C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '96. London, UK, UK: Springer-Verlag, 1996, pp. 104–113. ISBN: 3-540-61512-1. URL: <http://dl.acm.org/citation.cfm?id=646761.706156>.