
Volkan Cambazoglu
Profun Research

- Security in Profun Demonstrator

- Formalizing a Wireless Sensor Network (WSN) security protocol
Security in Profun Demonstrator
Wireless Sensor Nodes

- MSP430 micro-controllers (MCUs) from Texas Instruments (TI) are 16-bit, RISC-based, mixed-signal processors designed for ultra-low power
- 8-16 MHz, 10 Kbyte RAM, 48 KByte Flash
- Zolertia Z1
- TelosB/Tmote sky
Security Protocols

- Confidentiality
  - Symmetric Key Crypto (3DES, RC5, AES, Skipjack, Twofish, XTEA)
  - Asymmetric Key Crypto (ElGamal, RSA, ECC)

- Integrity
  - MDCs (MD5, SHA-1)

- Authentication
  - MACs (CBC-MAC, OCB and a subset of previous)
Cryptographic Libraries for WSNs

- TinyOS
  - TinySec
  - SenSec
  - MiniSec
  - TinyECC

- Contiki
  - Relic toolkit
  - ContikiSec
  - LibTomCrypt
Security protocols for TinyOS

- SPINS (first attempt, not fully implemented)
  - Secure Network Encryption Protocol (SNEP) and µTesla
- TinySec (fully implemented link layer security)
  - Skipjack (80 bit key) block cipher with CBC-CS and CBC-MAC
  - adds less than 10% energy, latency and bandwidth overhead
- SenSec (inspired by TinySec)
  - Skipjack-X (more secure than Skipjack, not vulnerable to brute force attacks)
  - Provides resilient keying mechanism
- MiniSec
  - 2 modes of operation: single source and multi-source broadcast communication
  - Skipjack, OCB shared key encryption
- TinyECC
  - Elliptic Curve Crypto based Public Key Crypto software package
  - Supports digital signature scheme (ECDSA), key exchange protocol (ECDH) and public key encryption (ECIES)
Relic-toolkit

- Specially written for use in sensor networks
- Efficient and flexible
- A large set of modern cryptographic functions
- Ongoing project and updated

- No documentation and tricky to understand...
Relic-toolkit

- Multiple-precision integer arithmetic;
- Prime and Binary field arithmetic (and preliminary ternary field arithmetic);
- Elliptic curves over prime and binary fields (NIST curves and pairing-friendly curves);
- Bilinear maps (Tate pairing over supersingular binary curves and optimal pairings over BN curves) and related extension fields;
- Cryptographic protocols (RSA, Rabin, ECDSA, ECMQV, ECSS (Schnorr), Sakai-Ohgishi-Kasahara ID-based authenticated key agreement, Boneh-Lynn-Schacham and Boneh-Boyen short signatures).
ContikiSec

- The study considered AES, RC5, Skipjack, Triple-DES, Twofish, XTEA
- ContikiSec-Enc (confidentiality only)
  - AES CBC-CS (all nodes with a single 128 bit key)
- ContikiSec-Auth (authentication only)
- CMAC
- ContikiSec-AE (authentication with confidentiality)
  - OCB mode with AES using a single shared key for conf. and auth.

<table>
<thead>
<tr>
<th>Application</th>
<th>uIP</th>
<th>Rime</th>
<th>X-MAC</th>
<th>LLP</th>
<th>NULLMAC</th>
<th>ContikiSec</th>
<th>Radio</th>
</tr>
</thead>
</table>

Fig. 1. The Contiki stack with ContikiSec.

Contiki packet
ContikiSec-Enc
ContikiSec-AE
ContikiSec follow-up

• Asymmetric key cryptography
  • ElGamal (Discrete logarithm problem)
  • RSA (Factoring large prime numbers modulo n)
  • ECC (Elliptic curve discrete logarithm problem)
• ECC is an attractive option for WSNs
  • Smaller key sizes than RSA
  • Higher throughput
  • Higher security
  • Possible to import into Contiki using Relic
• Still, ECC optimizations are very important, otherwise it will be resource heavy
Student thesis at Linköping University

- Atmel AT91SAM7XC256
- 32 bit processor max 55 MHz, 256 KB of flash and 32 KB of SRAM
- 2 crytographic blocks
  - AES128 encryption accelerator
  - Triple-DES
- LibTomCrypt as the cryptographic library
- AES performs the best among DES, 3DES and Twofish with the help of hardware acceleration (0.57 vs 9.2 ms/KiB encryption)
- ECC outperforms RSA as public key crypto. solution (at least 38% faster) (RSA-1024 vs ECC-160)
Next step

• Implementing the front end
• Scanning RFID from phone
• Securely propagating this info to the central server
• Implementing certain cryptographic function blocks (to be used until a stable library is released)
  • ECC, AES, RC5, OCB, CBC-MAC
• Key management...
Key Management

Individual key

Pairwise shared key

Group key

Cluster key

Fig. 2 LEAP keying mechanisms

Fig. 3 Sensor network: cases A and B.

The properties we have to analyse are the following:

– Authentication of Data1 and Data2, i.e., the node Ni and the base station (BS) share the same value for Data1 and Data2 and both execute the same session of the protocol. This property allows us to prove that bilateral authentication is achieved by using the MAC, and the integrity of the message is guaranteed.

– Confidentiality of Data1 and Data2, i.e., Data1 and Data2 are secret values shared between Ni and BS, and they are not known by an intruder or third parties.

The verification with AVISPA finds only the replay attack shown in Table 1, where IBS represents an intruder playing the role of the base station. Nevertheless, as was said before, TinySec does not manage replay attacks, which are left to higher layers of the protocol stack. Apart from this attack, the protocol is secure, even when a node is compromised by the intruder.

Case B. In this case we use the same scenario than in the previous case (see Fig. 3), but we consider TinySec-Auth messages instead of TinySec-AE messages. The protocol for TinySec-AE packets using AVISPA syntax is as follows:

   {MAC(Ni.AM1.Size1.Data1)} H(Km.Ni)

   {MAC(BS.AM2.Size2.Data2)} H(Km.Ni)

As we mentioned before, TinySec-Auth does not provide any confidentiality mechanisms. Thus, we can only analyse the authentication of Data1 and Data2, i.e., we can prove the bilateral authentication between BS and Ni, by means of the MAC messages, and also the integrity of messages. As in the previous case, we have found a replay attack that we omit.
Key Management

- Key Pre-distribution
  - Store all the keys on nodes
  - Memory consumption?
- Random Key Pre-distribution
  - Pool of keys to be used randomly
  - Two nodes sharing the same key?
- Without Key Pre-distribution?
Profun Research

• Security in Profun Demonstrator

• Formalizing a Wireless Sensor Network (WSN) security protocol
Formalizing a WSN security protocol

- **Problem: Efficient Secure data aggregation**
  - Secure Hierarchical In-Network Aggregation in Sensor Networks \([O(\Delta \log^2 n)]\)
  - An Efficient Integrity-Preserving Scheme for Hierarchical Sensor Aggregation \([O(\Delta \log n)]\)
  - Lower trees with fixed degrees: a recipe for efficient secure hierarchical aggregation in WSNs \([O(\log n) \text{ by making } \Delta \text{ a constant}]\)

Secure Hierarchical In-Network Aggregation in Sensor Networks


(b) Naive commitment tree, showing derivations of some of the vertices. For each sensor node $X$, $X_0$ is its leaf vertex, while $X_1$ is the internal vertex representing the aggregate computation at $X$ (if any). On the right we list the labels of the vertices on the path of node $G$ to the root.
Secure Hierarchical In-Network Aggregation in Sensor Networks

- Query dissemination
- Aggregation commit
- Result checking

Arrows: Aggregation tree.
An Efficient Integrity-Preserving Scheme for Hierarchical Sensor Aggregation

- Improvement based on local information
- Nodes close in the aggregation tree also close in the commitment tree
- Make commitment trees balanced binary trees by introducing dummy nodes
- Communication overhead is significantly reduced (1/10 of previous)
Lower trees with fixed degrees: a recipe for efficient secure hierarchical aggregation in WSNs

- Improvement based on
  - limiting node degree $\Delta$ to constant (at most $k$ children of a node)
  - introducing virtual connections by dividing the whole commitment tree into $d$ subtrees

- Node congestion is reduced 30% for all network sizes even the small sized networks
Next step

- Formalize secure aggregation using Pi/Psi calculus
- Proving that the proposed protocol is indeed “optimally secure”
- Re-verifying the protocol after adding security features
- Building formal models of networking and/or security concepts that can be re-used