ALIKE: Authenticated Lightweight Key Exchange

Sandrine Agagliate, GEMALTO Security Labs
Outline:

- Context

Description of ALIKE
- Generic description
- Full specification

Security properties
- Chip Unforgeability and Channel Secrecy
- Underlying PK-scheme security

Benchmark

Conclusion
CONTEXT: Contact-less cards (1)

Create a Secure Channel, using a key exchange protocol
- With no authentication: PACE (with password), DH
- Mutual authentication: Symmetric solutions like MiFare
  - Requires embedded dedicated HW circuit for both card and reader
  - Requires a common secret to be shared between the two parties
- Card authentication: ALIKE

Why an asymmetrical solution?
- When readers don’t necessarily need authentication:
  - Examples: access control, public transportation
- Allows facilitating interoperability
  - With secret key, each system derives the keys of its cards from its own master key
  - With public key, each system chooses to trust a CA
- Allows low-cost SAM-less reader
What challenge for an asymmetrical solution?

- Very strong time limitations:
  - Our target: The global transaction should not exceed 150 ms
  - Example: Tests on public transportation in London => traffic fluidity up to 450 ms
- Memory is limited on smart cards
- Pre-computation pose a number of practical problems

ALIKE = Authenticated Lightweight Key Exchange protocol [Coron, Gouget, Paillier, Villegas, 2010]

- Provides lightweight transactions in contact-less applications
- Increases the security level compared to classical asymmetrical authentication scheme like RSA (80-bit security)
- Based on the public key encryption scheme “RSA for paranoids” [Shamir, CryptoBytes, 1995] and on a block cipher
  - RSAP allows very fast decryption (performed inside the smart-card, where a cryptographic coprocessor is commonly available )
  - Contact-less cards commonly embed a coprocessor for a block cipher such as DES or AES
On-going Standardization

ISO/IEC 29192 (Draft in progress) : Lightweight cryptographic mechanisms targeted for constrained environments
  - Part 1: General
  - Part 2: Block ciphers
  - Part 3: Stream ciphers
  - Part 4: Mechanisms using asymmetric techniques

Committee Draft 29192-4 (in progress):
  - identification scheme cryptoGPS
  - authenticated key exchange protocol ALIKE
  - ID-based signature scheme I2R-IBS
Objective

ALIKE is a very fast protocol for contactless applications such that:
- A verifier PCD (e.g. a reader) authenticates a prover PICC (e.g. a contact-less card) relative to a certification authority CA
- Additionally, PCD and PICC establish a session key used for secure messaging

- There is no authentication of the PCD by the PICC
- Main target applications:
  - Access control, contact-less transport

PCD = Proximity Coupling Device
PICC = Proximity Integrated Circuit Card
Security requirements for ALIKE

Chip unforgeability under active attacks

- It should be “impossible” for an attacker to authenticate as a PICC without knowing that PICC’s private key

Channel secrecy under passive attacks

- It should be “impossible” for an attacker to recover the session key K of an eavesdropped transaction

Since there is no authentication of the PCD, «channel secrecy» cannot be secure under active attacks
ALIKE protocol: generic construction

Primitives:
- A block-cipher: $E : \{0,1\}^\alpha \times \{0,1\}^\beta \rightarrow \{0,1\}^\beta$, $\alpha \leq \beta$
- A public-key encryption scheme $\mathcal{E}$

[KeyGen] : key pair $(sk, pk)$, certificate $\sigma$ on $pk$ from CA

[Challenge-Response-Verification]:

**PICC: Card** Priv. key $sk$

choose $k$ in $\{0,1\}^\alpha$
compute $y = E_k(0)$

recover $r = D_{sk}(c)$
compute $res = E_r(k)$

**PCD: Reader** Pub. key $pk$

choose $r$ in $\{0,1\}^\alpha$
compute $c = E_{pk}(r)$

check $\sigma$
recover $k = E_{r^{-1}}(res)$
verify $y = E_k(0)$

$K = k$ XOR $r$

$K = k$ XOR $r$
Choice for the public-key encryption scheme

- We revisit «RSA for paranoids» RSAP [Shamir, CryptoBytes, 1995]
  - Unbalanced modulus $N = p.q$
  - Decryption of ciphertexts is done only modulo the smallest prime $p$
  - Possibly use moduli with fixed common part, without degrading security

- [KeyGen]
  - Given the security parameter $\kappa$ and a public exponent $e$:
    - prime $p$ with $|p| = \kappa$ such that $\gcd(e, p-1) = 1$
    - prime $q$ such that $|p| < |q|$, and modulus $N = p.q$
    - private exponent $d = e^{-1}$ mod $(p-1)$

- [Encryption]
  - Given $m$ in $\{0,1\}^\alpha$, with $\alpha + t \leq \kappa - 1$, compute $c = (m || H(m))^e$ mod $N$
    - $H : \{0,1\}^\alpha \rightarrow \{0,1\}^t$ is a hash function such that $\alpha + t \leq \kappa - 1$

- [Decryption]
  - Given $c$, compute $x = c^d$ mod $p$
  - Then parse $x$ as $m || h$ where $m$ is in $\{0,1\}^\alpha$ and $h$ is in $\{0,1\}^t$. If the parsing fails or if $h \neq H(m)$ return error. Otherwise return $m$. 
ALIKE protocol: full description

Primitives:
- A block-cipher: $E: \{0,1\}^\alpha \times \{0,1\}^\beta \rightarrow \{0,1\}^\beta$, $\alpha \leq \beta$: AES ($\alpha=\beta=128$)
- A public-key encryption scheme $E$ = variant of RSA for paranoids
  - small prime factor $p$ + moduli with fixed common part + $E_1||.(0)$ as hash function

[KeyGen] : key pair $(sk,pk)$, certificate $\sigma$ on $pk$ from CA

[Challenge-Response-Verification]:

**PICC: Card** Priv. key $sk$

choose $k$ in $\{0,1\}^{\alpha-1}$
compute $y = E_{0||k}(0)$
recover $r = D_{sk}(c) = c^d \mod p$
compute $res = E_{0||r}(k)$

**PCD: Reader** Pub. key $pk$

choose $r$ in $\{0,1\}^{\alpha-1}$
compute $c = E_{pk}(r) = (r||h)^e \mod N$, with $h = E_{1||r}(0)$
check $\sigma$
recover $k = E_{0||r^{-1}}(res)$
verify $y = E_{0||k}(0)$

$K = k \ XOR \ r$

$K = k \ XOR \ r$
Security assumptions

- Ideal Cipher Model (ICM)
  - Block-cipher is replaced with a publicly accessible ideal cipher, i.e. a family of random permutations parametrized by a key.
  - The attacker must query the encryption or decryption oracles attached to the IC.

- ICM has been shown to be equivalent to the Random Oracle Model (ROM) [Coron, Patarin, Seurin, Crypto’2008]
  - ICM is not a stronger assumption than the ROM.

- Viewing E as an ideal cipher, we proved that our construction is secure under appropriate security assumptions on E.
Security assumptions (2)

- [Bellare, Desai, Pointcheval and Rogoway, Crypto’1998]

- **OW-CPA:**
  - A public-key encryption scheme $\mathcal{E}$ is said to be $(t, \epsilon)$-OW-CPA if no adversary running in time $t$, given a random public key $pk$ and $c = \mathcal{E}_{pk}(m)$ where $m$ is generated at random in the message space, can output $m$ with probability better than $\epsilon$

- **OW-CCA:**
  - Same as OW-CPA, but with access to a decryption oracle for any $c' \neq c$

- **P-OW-CPA:** (partially OW-CPA)
  - Same as OW-CPA, but with $c = \mathcal{E}_{pk}(m)$ where $m=m1\|m2$ is generated at random in the message space, can output $m1$ with probability better than $\epsilon$
Security theorems: on underlying PK-scheme assumption

Theorem 1 (Active Unforgeability)
- ALIKE is \((t, \varepsilon)\)-secure against unforgeability under active attacks, in the ideal cipher model, assuming that \(E\) is \((t', \varepsilon')\)-OW-CCA secure.

Theorem 2 (Passive Secrecy)
- ALIKE is \((t, \varepsilon)\)-passively secure against secrecy, in the ideal cipher model, assuming that \(E\) is \((t', \varepsilon')\)-OW-CPA secure.
Security of underlying PK-scheme

- RSAP is partially OW-CPA secure [Shamir, CryptoBytes, 1995]

- Chosen Ciphertext attack on RSAP (RSAP is not OW-CCA secure):
  - Generate a random $c$ in $\mathbb{Z}_N$
  - Request its decryption $m = c^d \mod p$
  - Compute $c' = m^e \mod N$
  - Then $\gcd(c-c', N)$ disclose $p$ with overwhelming probability

- Other Known attacks on RSAP are related to the size of the message to encrypt/decrypt
  - Known countermeasure: message size strictly $<$ smallest prime size
  - Taken into account in ALIKE

Theorem 3 (Underlying Public Key Encryption Scheme)

- $\mathcal{E} = \text{RSAP-H}$ is $(t, \epsilon)$-OW-CCA secure, assuming that RSAP is $(t', \epsilon')$-P-OW-CPA secure
Target: at least 80-bit security

Tuning the size of $N$ and $p$:
- Factoring algorithms whose running time depends on the size of $N$;
  The fastest such algorithm is the General Number Field Sieve (GNFS) [Lenstra, Lenstra, 1993]
- Factoring algorithms whose running time depends on the size of $p$;
  The fastest such algorithm is the Elliptic Curve Method (ECM) [Lenstra, 1987]

Tuning public exponent $e$:
- Coppersmith’s attack
  Attack based on Coppersmith’s Theorem for finding small roots of polynomial equations. The attack applies when a small public exponent $e$ is used.
- Shamir’s bound
  Take $e$ such that $m^e$ size before the modular reduction is at least twice $N$ size
Real-life implementation of ALIKE (2)

Tuning the number $\lambda$ of non-predetermined bits in $N$

- [Shamir, CryptoBytes, 1995]: RSA moduli with a fixed common part can be used without degrading the overall system security
- allows to reduce transmissions

Example of settings

- $\lambda = \text{nb of non-predetermined bits in } N$;
- $t = \text{output size of the redundancy (hash size) used in ALIKE with RSAP-H}$

| ALIKE Security | $|N|$ 1248 | $|p|$ 352 | $\lambda$ 403 | $e$ 11 | Block Cipher AES-128 | $\alpha$ 128 | $\beta$ 128 | $t$ 128 |
|----------------|--------|-------|--------|-----|----------------|------|------|------|
| 80 bits        |        |       |        |     |                 |      |      |      |
| 100 bits       | 2048   | 560   | 611    | 17  | AES-128         | 128  | 128  | 128  |
ALIKE – benchmark (source Sec Lab’s)

- Based on NXP’s SmartMX P5CT072 platform
  - FameXE cryptoprocessor
  - DES processor

- PCD simulated on a PC via a transparent contact-less reader
  - Modular exponentiation + DES block-cipher

- Code size of our ALIKE library = 1.6 KB

- Estimation for $|p| = 352$, $|N| = 1248$ and $|\sigma| = 1280$ (80-bit security if DES is replaced by AES)
  - Total transaction time is close to 156 milliseconds
  - RAM consumption : 900 bytes
  - Non-volatile memory : 248 bytes
ALIKE (80-bit Security) - estimation
## Summary

<table>
<thead>
<tr>
<th>Security Level [bits]</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Crypto-coprocessor functionalities</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required for Modular multiplication</td>
<td>Not required</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Functions required</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>A random number generation.</td>
<td>- A random number generation.</td>
<td>- A random number generation.</td>
</tr>
<tr>
<td>Two blocks cipher executions without specific side channel and fault attacks countermeasures.</td>
<td>- Two blocks cipher executions</td>
<td>- Two blocks cipher executions</td>
</tr>
<tr>
<td>A modular exponentiation with small modulus (</td>
<td>p</td>
<td>= 352 bits)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non Volatile memory</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>To store RSA keys for ALIKE (88 bytes to compare to 400 bytes for classical RSA) and certificates</td>
<td>To store public of CA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code size</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6 kbytes on 8051 core</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data transferred with communication speed at 106 kb/s</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming data 160 bytes  15.40 ms</td>
<td>Incoming data 192 bytes  18.8 ms</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal Process</th>
<th>PICC process</th>
<th>PCD process</th>
</tr>
</thead>
<tbody>
<tr>
<td>- From 4 to 15 faster than classical RSA according to component</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- As example for 8051 core: 80 ms at 31MHz for CPU and 48 MHz for crypto-coprocessor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion:

- ALIKE is a new key exchange protocol allowing to
  - Authenticate the smartcard relatively to a CA
  - Establish a session key (to create a secure channel between smartcard and reader)

- ALIKE specificities:
  - Allows possible interoperability
  - Requires limited hardware resources
  - Very fast: 156ms for total transaction  \(\rightarrow\) RSAP is much faster than RSA
  - Secure: 80-bit security

- ALIKE is proven secure
- Proof of concept / prototype
- In right way to be standardized
References

- **ALIKE previously called SPAKE:**


