COMPUTATIONAL ANALYSIS OF THE EDHOC PROTOCOL

Baptiste COTTIER
## Symbolic vs Computational Security

<table>
<thead>
<tr>
<th></th>
<th>Symbolic</th>
<th>Computational</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PRIMITIVES</strong></td>
<td>Treated as blackboxes</td>
<td>Functions on bitstrings</td>
</tr>
<tr>
<td><strong>MESSAGES</strong></td>
<td>(typed) Terms</td>
<td>Bitstrings (1001101101)</td>
</tr>
<tr>
<td><strong>ATTACKER</strong></td>
<td>Restricted to compute only using these primitives</td>
<td>Any probabilistic polynomial-time algorithm</td>
</tr>
<tr>
<td><strong>SECRECY</strong></td>
<td>Attacker can not distinguish when the value of the secret changes</td>
<td>Attacker can not distinguish the secret from a random value</td>
</tr>
</tbody>
</table>
EDHOC constraints:
• Small number of messages (ideally 3, or 4 with key-confirmation)
• Small message size (~100 bytes in total)
• Minimize code and memory footprint

Analysis done in the **static-static** setting using:
• 128 bits-security **Elliptic Curve DH**
• 64-bits security **MAC** (trade-off to reduce communication)

<table>
<thead>
<tr>
<th>Id</th>
<th>AEAD</th>
<th>Hash</th>
<th>MAC len</th>
<th>ECDH curve</th>
<th>Signature</th>
<th>Application AEAD</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>AES-CCM-16-64-128</td>
<td>SHA-256</td>
<td>8</td>
<td>X25519</td>
<td>EdDSA</td>
<td>AES-CCM-16-64-128</td>
<td>constrained</td>
</tr>
<tr>
<td>1</td>
<td>AES-CCM-16-128-128</td>
<td>SHA-256</td>
<td>16</td>
<td>X25519</td>
<td>EdDSA</td>
<td>AES-CCM-16-64-128</td>
<td>constrained</td>
</tr>
<tr>
<td>2</td>
<td>AES-CCM-16-64-128</td>
<td>SHA-256</td>
<td>8</td>
<td>P-256</td>
<td>ES256</td>
<td>AES-CCM-16-64-128</td>
<td>constrained</td>
</tr>
<tr>
<td>3</td>
<td>AES-CCM-16-128-128</td>
<td>SHA-256</td>
<td>16</td>
<td>P-256</td>
<td>ES256</td>
<td>AES-CCM-16-64-128</td>
<td>constrained</td>
</tr>
<tr>
<td>4</td>
<td>ChaCha20/Poly1305</td>
<td>SHA-256</td>
<td>16</td>
<td>X25519</td>
<td>EdDSA</td>
<td>ChaCha20/Poly1305</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ChaCha20/Poly1305</td>
<td>SHA-256</td>
<td>16</td>
<td>P-256</td>
<td>ES256</td>
<td>ChaCha20/Poly1305</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>A128GCM</td>
<td>SHA-256</td>
<td>16</td>
<td>X25519</td>
<td>ES256</td>
<td>A128GCM</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>A256GCM</td>
<td>SHA-384</td>
<td>16</td>
<td>P-384</td>
<td>ES384</td>
<td>A256GCM</td>
<td>high-security</td>
</tr>
<tr>
<td>25</td>
<td>ChaCha20/Poly1305</td>
<td>SHAKE256</td>
<td>16</td>
<td>X448</td>
<td>EdDSA</td>
<td>ChaCha20/Poly1305</td>
<td>high-security</td>
</tr>
</tbody>
</table>
SECURITY GOALS

• Security Level of 128 bits: Minimum expected time needed to attack the protocol.

  With $T$ the execution time of the protocol and $\varepsilon$ the success probability of the attack, we have:

  $$\frac{T}{\varepsilon} \cong 2^{128}$$

• Applicative data confidentiality:

  • **Key-Privacy**: At most both participants know the final session key. By compromising the long-term credential of either peer, an attacker shall not be able to compute past session keys

  • **Mutual Authentication**: Exactly both participants have the material to compute the final session key

  • **Identity Protection**
Equivalent to **Implicit Authentication**

Relies on the **Computational Diffie-Hellman** assumption

- Depends on the group size where Diffie-Hellman is considered

Indistinguishability in the **Find-Then-Guess model**. The adversary is given access to oracles:

- **Send**: models an active attack, in which the adversary may intercept a message and then either modify it, create a new one, or simply forward it to the intended participant.

- **Reveal**: models the misuse of session keys by a user

- **Test**: tries to capture the adversary’s ability (or inability) to tell apart a real session key from a random one

  - Given several accesses to the **Send** and **Reveal** oracles, and only one access to the **Test** oracle, the attacker succeeds if he can distinguish the session key from a random value
• Equivalent to Explicit Authentication

• Ends when both parties activate the following flags (initialized at 0):
  • **Accept**: asserts that we have the required material
  • **Terminate**: asserts that other party has the required material

• Relies on MAC security:
  • 64 bits MAC provides 128-bits security
  • *To check*: Is 128 bits security reached after few AEAD messages?
The protocol should protect the identity of the parties:

- against active attackers for the Initiator
- against passive attackers for the Responder

Security games:

- Given two identities, an active attacker should not distinguish the Initiator
- Given two identities, a passive attacker should not distinguish the Responder
**PROTOCOL DECOMPOSITION**

**KEY EXCHANGE**

- Initiator:
  - Suites_I, G_X, C_I, EAD_1
  - message_1
    - G_Y, Enc (ID_R, MAC_2, EAD_2), C_R
    - message_2
      - AEAD (ID_I, MAC_3, EAD_3)
      - message_3
        - AEAD (EAD_4)
        - message_4

- Responder:
  - Accept = 1
  - Terminate = 1

**Key-Privacy**

**Identity Protection**

**Mutual authentication**