Introduction

Silent Circle instant messaging protocol (SCimp)
  - Key negotiation
  - Rekeying
  - Sending data
  - Progressive encryption
  - Group conversation
  - File transfer

Proverif results

Silent Text

Comparison with other protocols

Conclusions
Introduction

- Silent Circle instant messaging protocol (SCimp)
- Built on XMPP
- Mobile environment
  - Asynchronous
- Security properties
  - Confidentiality/Integrity/Availability
  - Authentication of other party
  - Deniability
  - Key erasure
  - Future secrecy
  - (Privacy Protection)
SCimp: Key negotiation

Alice

- genKeyPair()
- DH($sk_a, pk_b$)

Bob

- hash($pk_a$)
- $pk_b$
- $pk_a, mac_a$
- $mac_b$
- genKeyPair()
- DH($sk_b, pk_a$)

Keys derived from DH

Out of band

- Confirm SAS

Authenticated
ECDHE gives shared secret $Z$, from which are derived:
- $k_{snd,0}, k_{rcv,0}, i_{snd,0}, i_{rcv,0}$; for message encryption and authentication
- $mac_a, mac_b$; to confirm knowledge of $Z$
- SAS; for authentication of identity
- $cs$; for rekeying

User messages can be sent after four key exchange messages

SAS confirms identity all previous communication
  - Requires commitment to $pk_a$ to prevent collision attack

Completely ephemeral
  - Deniable
Alice and Bob share $cs$

hash($pk_a$), MAC$_{cs}$($pk_a$)

pk$_b$, MAC$_{cs}$($pk_b$)

pk$_a$, mac$_a$

mac$_b$

Keys derived from DH and $cs$

Verify MAC$_{cs}$
SCimp: Rekeying

- First: store old decryption key (messages might arrive out of order)
- Optional: SAS comparison only after several rekeyings
- Rekeying ensures future secrecy
- It is not specified when to rekey
- Protocol aborts on error
  - Keys are discarded, including cs
SCimp: Sending data

- Encrypt
  - $\text{ciphertext} = \text{AES}_{k_j}(i_j, \text{plaintext})$
SCimp: Sending data

- Encrypt
  - ciphertext = AES_{k_j}(i_j, plaintext)
- Update keys (ratchet)
  - k_{j+1} = MAC_{k_j}(i_j)
  - i_{j+1} = i_j + 1
SCimp: Sending data

- Encrypt
  - ciphertext = AES_{k_j}(i_j, plaintext)

- Update keys (ratchet)
  - $k_{j+1} = MAC_{k_j}(i_j)$
  - $i_{j+1} = i_j + 1$

- Send message:
  - $i_j$
  - ciphertext

No message signatures: deniable

Ratchet enables key erasure, but:
- Out of order messages require you to store old keys
- Old keys compromise future keys

S. R. Verschoor (TU/e)  Secure Messaging in Mobile Environments  December 7th, 2015  8 / 17
SCimp: Sending data

- **Encrypt**
  - ciphertext = AES_{k_j}(i_j, plaintext)

- **Update keys (ratchet)**
  - k_{j+1} = MAC_{k_j}(i_j)
  - i_{j+1} = i_j + 1

- **Send message:**
  - i_j
  - ciphertext

- **No message signatures: deniable**

- **Ratchet enables key erasure, but:**
  - Out of order messages require you to store old keys
  - Old keys compromise future keys
genKeyPairs(): $(sk_0, pk_0), (sk_a, pk_a)$

$Z_0 = DH(sk_0, pk_B)$

$ct = AES_{k_0}(i_0, pt)$

$(pk_0, ct; \text{ hash}(pk_a))$
SCimp v2: Progressive encryption

- SAS confirmation after “regular” key negotiation
  - Confirms entire conversation

Vulnerable to Man In The Middle (MITM) attack
- MITM re-encrypts and forwards user messages
- MITM blocks keying messages
SCimp v2: Progressive encryption

- SAS confirmation after “regular” key negotiation
  - Confirms entire conversation
- Vulnerable to Man In The Middle (MITM) attack
  - MITM re-encrypts and forwards user messages
  - MITM blocks keying messages
SCimp v2: Group conversation

- Everything encrypted with a single symmetric key
SCimp v2: Group conversation

- Everything encrypted with a single symmetric key
- Group initiator generates a random symmetric key $k$
SCimp v2: Group conversation

- Everything encrypted with a single symmetric key
- Group initiator generates a random symmetric key $k$
- Generate random session key $ks$
- ciphertext = $AES_{ks}(k)$
- $eks = ECC\_Encrypt_{pk_B}(ks)$
- Send: $eks$, ciphertext
- Decrypt
SCimp v2: Group conversation

- Everything encrypted with a single symmetric key
- Group initiator generates a random symmetric key $k$
- Generate random session key $ks$
- ciphertext = $\text{AES}_{ks}(k)$
- $eks = \text{ECC Encrypt}_{pk_B}(ks)$
- Send: $eks$, ciphertext
- Decrypt
- Derive group key from $k$
SCimp v2: Group conversation

- Everything encrypted with a single symmetric key
- Group initiator generates a random symmetric key $k$
- Generate random session key $ks$
- ciphertext = $\text{AES}_{ks}(k)$
- $eks = \text{ECC\_Encrypt}_{pk_B}(ks)$
- Send: $eks$, ciphertext
- Decrypt
- Derive group key from $k$

- No authentication possible
  - Relies on trust in the server
  - Trivial MITM
SCimp v2: File transfer

- Files are encrypted and uploaded to the cloud
- Keys are exchanged using regular SCimp messages
- Convergent encryption
  - key = hash(file)
  - Missing a salt/secret
  - Vulnerable to file confirmation attack
SCimp v2: File transfer

- Files are encrypted and uploaded to the cloud
- Keys are exchanged using regular SCimp messages
- Convergent encryption
  - key = hash(file)
  - Missing a salt/secret
  - Vulnerable to file confirmation attack
  - Vulnerable to file swapping attack
Proverif results

- First key negotiation (if SAS confirmed over authenticated channel)
  - ✓ Confidentiality of keys
  - ✓ Authenticity of keys and other party identity

- Rekeying
  - ✓ Confidentiality of keys
  - ✓ Authenticity of keys and other party identity
  - Future secrecy
    - ✓ When attacker misses first rekeying after compromise
    - ✓ When users reconfirm the SAS

- Sending user message
  - ✓ Confidentiality of keys
  - ✓ Strong secrecy of messages
  - ✓ Authenticity of messages and keys
  - ✓ Forward secrecy (if keys can be erased)
  - ✓ Deniability

- Progressive encryption
  - × Confidentiality/authenticity of first message
  - ✓ Confidentiality/authenticity of all messages and keys (after SAS is confirmed over an authenticated channel)
Silent Text: SCimp implementation

- CCM implementation does not validate authentication tag
  - Problem in LibTomCrypt (fixed)
- Timing side-channel vulnerability
  - All secrets compared with memcmp
- Race condition in message parsing queue
- Message keys are deleted before received messages are validated
- Returned error codes are not checked
- Memory allocation is not checked
- State machine contains bugs and is often bypassed
- Style issues
## Comparison with other protocols

<table>
<thead>
<tr>
<th>Feature</th>
<th>SCimp v1</th>
<th>SCimp v2</th>
<th>OTR</th>
<th>TextSecure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data in first message</td>
<td>✗</td>
<td>✓ (✗)</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Key erasure</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Preshare public keys</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rekey on each reply</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Ratchet every message</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>ECC</td>
<td>✓</td>
<td>✓</td>
<td>✗ (✓)</td>
<td>✓</td>
</tr>
</tbody>
</table>
Conclusions

- SCimp version 1 is secure (proven by Proverif)
  - ...but does not solve problems of a mobile environment
- SCimp version 2 solves problems of a mobile environment
  - ...but is insecure
- SCimp implementation has a lot of problems
  - ...lowering both security and user experience
- OTR is secure and good for synchronous environment
- TextSecure is secure and good for mobile environment