

(In-)secure messaging with SCimp

Sebastian R. Verschoor and Tanja Lange
(with many slides and pictures by Sebastian)

University of Waterloo / Eindhoven University of Technology

CryptoAction Symposium 2017
28 March 2017

<https://eprint.iacr.org/2016/703>

Secure Messaging protocols



TO THE TECHNOLOGY COMMUNITY:

Your threat model just changed.

Incoming President Donald Trump made campaign promises that, if carried out, threaten the free web and the rights of millions of people. He has praised attempts to undermine digital security, supported mass surveillance, and threatened net neutrality. He promised to identify and deport millions of your friends and neighbors, track people based on their religious beliefs, and suppress freedom of the press.

And he wants to use your servers to do it.

EFF ad in Wired magazine ([source](#))

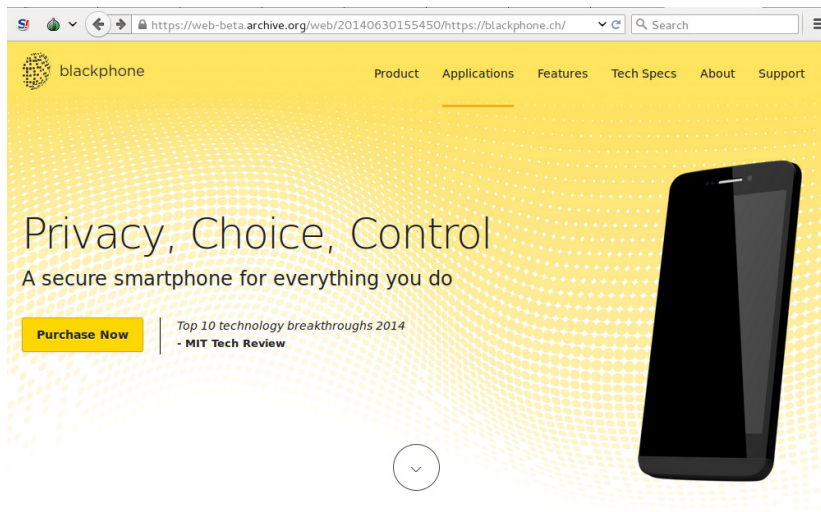
History of online secure messaging (1/2)

- ▶ 1991: Phil Zimmermann creates PGP
- ▶ 2004: Nikita Borisov, Ian Goldberg and Eric Brewer create OTR
 - ▶ Secure, but requires synchronous environment
- ▶ 2011: Gary Belvin introduces SecureSMS (master's thesis)
- ▶ 2012: SCimp (Silent Circle instant messaging protocol)
 - ▶ By Vinnie Moscaritolo, Gary Belvin and Phil Zimmermann
 - ▶ SecureSMS for XMPP
 - ▶ Even copies variable names and equation numbering from Belvin's thesis (despite creating internal inconsistencies)
- ▶ February 2014: Open Whisper Systems releases TextSecure v2
 - ▶ Allows offline initial user message
 - ▶ Later renamed to Signal

History of online secure messaging (2/2)

- ▶ May 2014: Silent Circle updates to SCimp v2
 - ▶ Allows offline initial user message
- ▶ August 2015: Silent Circle releases code for SCimp v2
 - ▶ Adds more inconsistencies between code and documentation
- ▶ September 2015: Silent Circle discontinues SCimp, switches to Signal-based protocol
- ▶ October 2015: Andreas Straub proposes OMEMO
 - ▶ Multi-device Signal for XMPP
- ▶ Oct-Nov 2016: Trevor Perrin and Moxie Marlinspike release official specification for the Signal protocol
- ▶ Dec 7th 2016: OMEMO gets standardized by the XMPP Standard Foundation: XEP-0384 (experimental)

Blackphone by Silent Circle



The screenshot shows a web browser window with the address bar containing the URL <https://web-beta.archive.org/web/20140630155450/https://blackphone.ch/>. The website has a yellow background with a pattern of small white dots. The Blackphone logo is in the top left, and a navigation menu with links for Product, Applications, Features, Tech Specs, About, and Support is in the top right. The main heading reads "Privacy, Choice, Control" with the subtext "A secure smartphone for everything you do". A yellow "Purchase Now" button is on the left, and a quote from MIT Tech Review is on the right. A black smartphone is shown on the right side of the page. A circular arrow icon is at the bottom center.

blackphone


Product Applications Features Tech Specs About Support

Privacy, Choice, Control

A secure smartphone for everything you do

Purchase Now

Top 10 technology breakthroughs 2014
- MIT Tech Review



<https://web-beta.archive.org/web/20140630155450/https://blackphone.ch/>

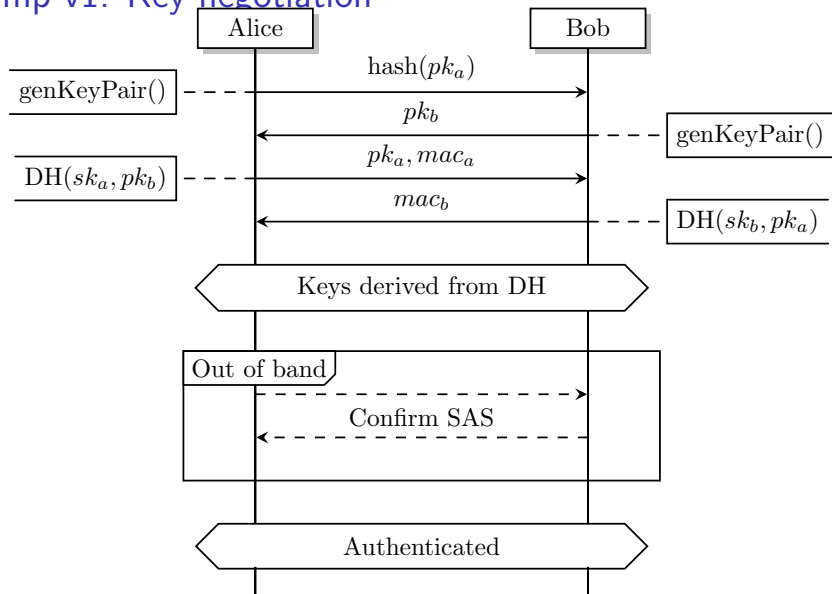
Blackphone by Silent Circle

- ▶ Android-based smart phone running Silent OS operating system.
- ▶ Rather expensive (2016: 799 USD); now 599 USD.
- ▶ Promise of higher security and privacy: “Blackphone puts privacy first” .
- ▶ The Blackphone 1 (summer 2014) came with service subscriptions for encrypted phone (Silent Phone) and chat (Silent Text).
- ▶ Silent Text was available as free Apple and Android apps but subscription fee for servers.
- ▶ No public security analysis or code; after intense online discussion (Wilcox-O’Hearn’s open letter to Phil Zimmermann and Jon Callas) some source code for on GitHub.
- ▶ Callas suggested analysis of SCimp as topic for Verschoor’s thesis.

Our involvement

- ▶ December 2015: Verschoor's Master's thesis (at TU/e) on SCimp
 - ▶ SCimp v1 is formally verified by ProVerif to be secure
 - ▶ SCimp v2 contains cryptographic flaws
 - ▶ the implementation contains many security bugs
- ▶ June 2016: Verschoor's cryptographic report on OMEMO
 - ▶ Minor bug found in multidevice setting
 - ▶ Developer patches it the same day as reported
- ▶ July 2016: Sebastian Verschoor and I release SCimp preprint paper
 - ▶ Some of Silent Circle's code (copied from SCimp implementation) still contains bugs that were reported in Verschoor's thesis
 - ▶ Bugs got patched a few days later
 - ▶ Initial bug report: **September 2015**

SCimp v1: Key negotiation



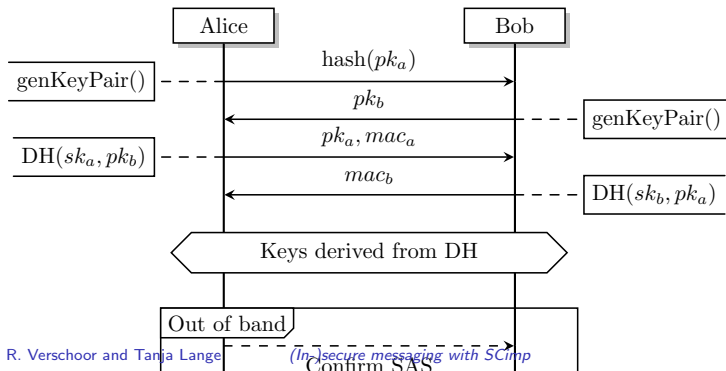
SAS: short authentication string

SCimp v1: Key negotiation

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

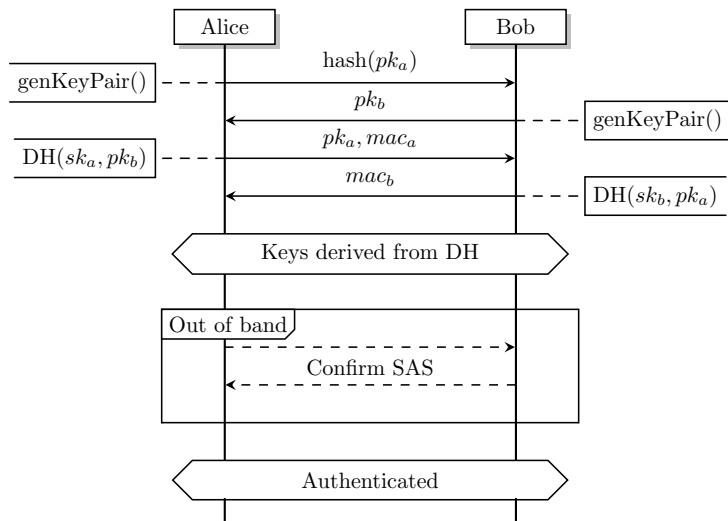
User messages can be sent after four key exchange messages



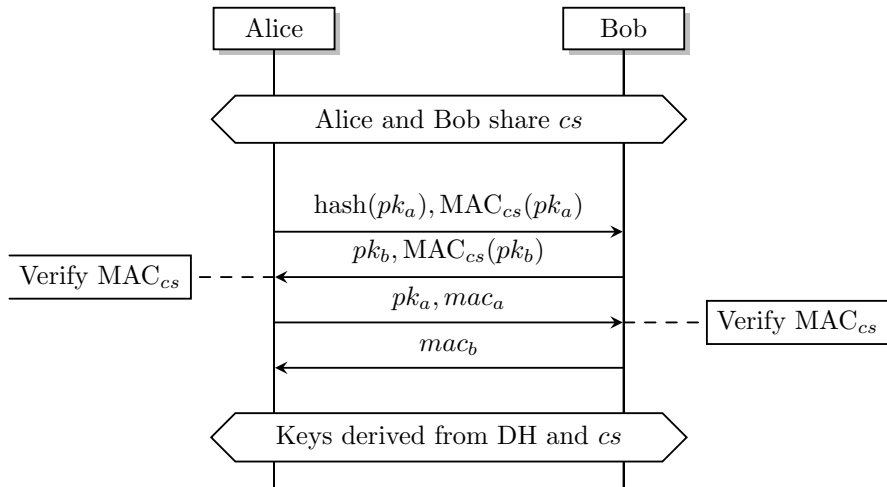
SCimp v1: Key negotiation

SAS confirms identity all previous communication

- Requires commitment to pk_a to prevent collision attack



SCimp v1: Rekeying



cs used for key continuity; gets updated at end up rekeying.
Description speaks of "self-healing".

SCimp v1: 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

SCimp v1: Sending user messages

- ▶ Encrypt
 - ▶ $\text{ciphertext} = \text{AES}_{k_j}(i_j, \text{plaintext})$

SCimp v1: Sending user messages

- ▶ Encrypt
 - ▶ ciphertext = $\text{AES}_{k_j}(i_j, \text{plaintext})$
- ▶ Update keys (ratchet)
 - ▶ $k_{j+1} = \text{MAC}_{k_j}(i_j)$
 - ▶ $i_{j+1} = i_j + 1$

SCimp v1: Sending user messages

- ▶ Encrypt
 - ▶ $\text{ciphertext} = \text{AES}_{k_j}(i_j, \text{plaintext})$
- ▶ Update keys (ratchet)
 - ▶ $k_{j+1} = \text{MAC}_{k_j}(i_j)$
 - ▶ $i_{j+1} = i_j + 1$
- ▶ Send message:
 - ▶ i_j
 - ▶ ciphertext

SCimp v1: Sending user messages

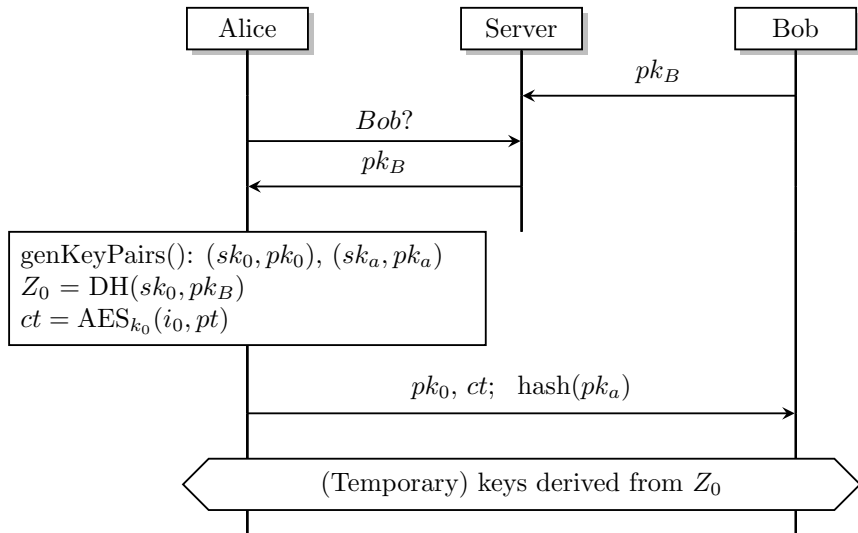
- ▶ Encrypt
 - ▶ $\text{ciphertext} = \text{AES}_{k_j}(i_j, \text{plaintext})$
- ▶ Update keys (ratchet)
 - ▶ $k_{j+1} = \text{MAC}_{k_j}(i_j)$
 - ▶ $i_{j+1} = i_j + 1$
- ▶ Send message:
 - ▶ i_j
 - ▶ ciphertext

- ▶ No message signatures: deniable
- ▶ Ratchet enables key erasure, but:
 - ▶ Out of order messages require you to store old keys
 - ▶ Old keys compromise future keys

Proverif results for SCimp v1

- ▶ 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

SCimp v2: Progressive encryption



From here on: regular key negotiation in parallel to user messages

Proverif results for SCimp v2

- ▶ Progressive encryption
 - ✗ Confidentiality/authenticity of first message
 - ✓ Confidentiality/authenticity of all messages and keys (**after** SAS)

SCimp

- ▶ ProVerif reports that the initial message of SCimp v2 is not confidential
 - ▶ This does not impact SCimp v1
- ▶ But, ProVerif also verifies that users can detect this when confirming the SAS later
- ▶ To determine the impact of this vulnerability, we had to look at the source code

SCimp

- ▶ ProVerif reports that the initial message of SCimp v2 is not confidential
 - ▶ This does not impact SCimp v1
- ▶ But, ProVerif also verifies that users can detect this when confirming the SAS later
- ▶ To determine the impact of this vulnerability, we had to look at the source code

We, that means Sebastian.



SCimp

A short example to give a flavor of the code

```
unsigned long ctxStrLen = 0;  
size_t kdkLen;  
int keyLen = scSCimpCipherBits(ctx->cipherSuite);
```

- ▶ **ctxStrLen** length in bytes (computing function returns size_t)
- ▶ **kdkLen** length in bytes
- ▶ **keyLen** function name suggests bit-length, but k_{snd} is $2 * \text{keyLen}$ bits long

Stealth MitM in SCimp

- ▶ Each message has a plaintext tag identifying the type:
 - ▶ keying message; or
 - ▶ user message
- ▶ The adversary can block the key negotiation using just this tag, thereby having set up a succesful MitM
- ▶ The vigilant user might detect this

Stealth MitM in SCimp

- ▶ Each message has a plaintext tag identifying the type:
 - ▶ keying message; or
 - ▶ user message
- ▶ The adversary can block the key negotiation using just this tag, thereby having set up a succesful MitM
- ▶ The vigilant user might detect this
- ▶ But the code contains another bug: the receiver overreacts when a keying message is received out of order

Stealth MitM in SCimp

- ▶ Each message has a plaintext tag identifying the type:
 - ▶ keying message; or
 - ▶ user message
- ▶ The adversary can block the key negotiation using just this tag, thereby having set up a succesful MitM
- ▶ The vigilant user might detect this
- ▶ But the code contains another bug: the receiver overreacts when a keying message is received out of order
 - ▶ only the message tag is inspected
 - ▶ the receiver deletes all local key material
 - ▶ thereby annulling any security set up in the past
- ▶ The adversary can desynchronize any secure session with a single out-of-order key message and set up a MitM undetected

More on SCimp

Other discrepancies between the model and the implementation

- ▶ Group messages have a single symmetric key
 - ▶ Relies on trust in the SC server
 - ▶ Subject to a trivial MitM attack
- ▶ CCM-mode implementation did not validate authentication tags
 - ▶ Problem in LibTomCrypt (fixed)
- ▶ Code contains many timing side-channel vulnerabilities
- ▶ The message parsing queue has a race condition
- ▶ Unchecked function error codes
 - ▶ Including memory allocations
- ▶ State machine based design: good coding style
 - ▶ and helps in making a model of the code
 - ▶ in case of SCimp: helps find where specs and code differ
 - ▶ seemed to have been an add on – state machine not actually used in SCimp implementation

More on SCimp extra features

SCimp file transfer

- ▶ Convergent encryption
 - ▶ $\text{key} = \text{hash}(\text{file})$
 - ▶ send as SCimp message
 - ▶ $\text{ciphertext} = \text{AES_CCM}_{\text{key}}(\text{file})$
 - ▶ upload to cloud
- ▶ Known vulnerabilities of CE:
 - ▶ confirmation of a file
 - ▶ learn the remaining information

More on SCimp extra features

SCimp file transfer

- ▶ Convergent encryption
 - ▶ $\text{key} = \text{hash}(\text{file})$
 - ▶ send as SCimp message
 - ▶ $\text{ciphertext} = \text{AES_CCM}_{\text{key}}(\text{file})$
 - ▶ upload to cloud
- ▶ Known vulnerabilities of CE:
 - ▶ confirmation of a file
 - ▶ learn the remaining information
- ▶ SCimp: receiver does not check $\text{hash}(\text{file}) = \text{key}$
 - ▶ file injection attack

More on SCimp extra features

SCimp file transfer

- ▶ Convergent encryption
 - ▶ $\text{key} = \text{hash}(\text{file})$
 - ▶ send as SCimp message
 - ▶ $\text{ciphertext} = \text{AES_CCM}_{\text{key}}(\text{file})$
 - ▶ upload to cloud
- ▶ Known vulnerabilities of CE:
 - ▶ confirmation of a file
 - ▶ learn the remaining information
- ▶ SCimp: receiver does not check $\text{hash}(\text{file}) = \text{key}$
 - ▶ file injection attack
 - ▶ This attack remained in the code until July 2016, independent of the upgrade of the messaging protocol

Other features, e.g., group messaging, usually fewer security guarantees.

Further reading

- ▶ Sebastian's homepage including tutorial given at ShmooCon:
<https://www.zeroknowledge.me/>
- ▶ ProVerif models are available:
<https://github.com/sebastianv89/scimp-proverif>
- ▶ Sebastian's thesis about SCimp:
<http://repository.tue.nl/844313>
- ▶ Preprint about SCimp:
<https://eprint.iacr.org/2016/703>
- ▶ Get Signal (Android/iPhone):
<https://whispersystems.org/>
- ▶ Try OMEMO (on Android):
<https://conversations.im/>
- ▶ OMEMO audit report (by Sebastian):
<https://conversations.im/omemo/audit.pdf>