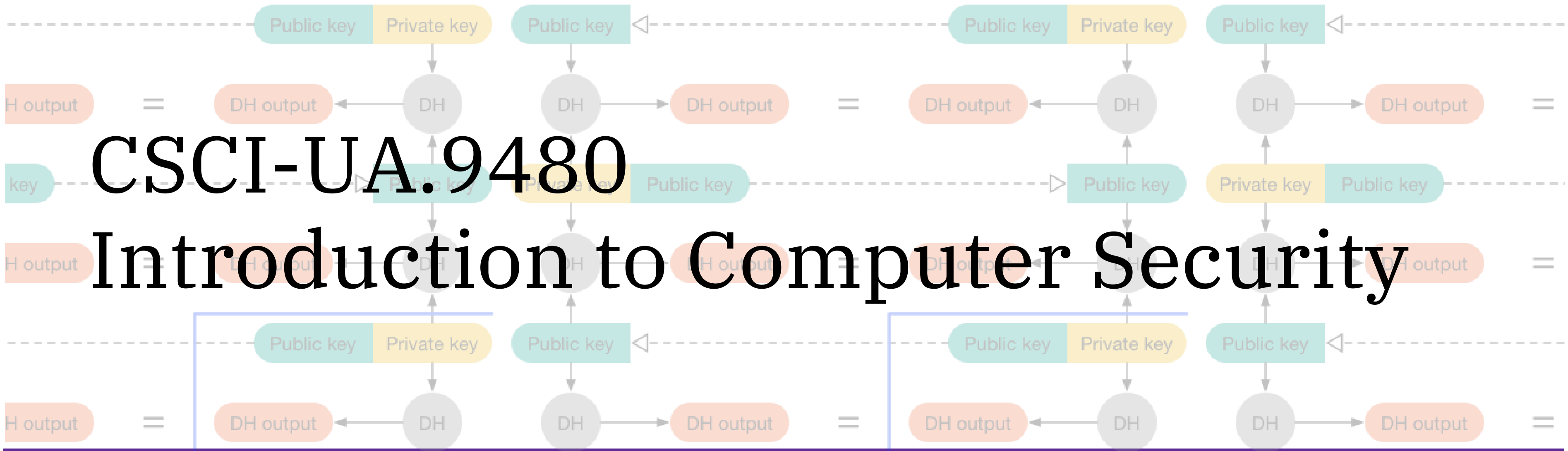


CSCI-UA.9480
Introduction to Computer Security

Session 1.5
Usable Security and Secure Messaging

Prof. Nadim Kobeissi

Usable Security: Then and Now

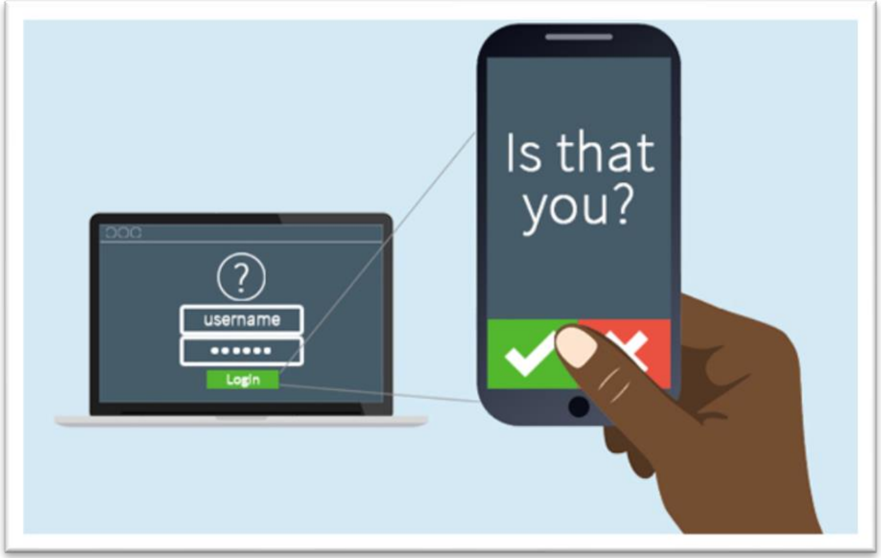
1.5a

“Humans are incapable of securely storing high-quality
cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations.”
– *Kaufmann, Perlman and Speciner.*

The last word on your identity: you.

But this isn't the case in computer security.

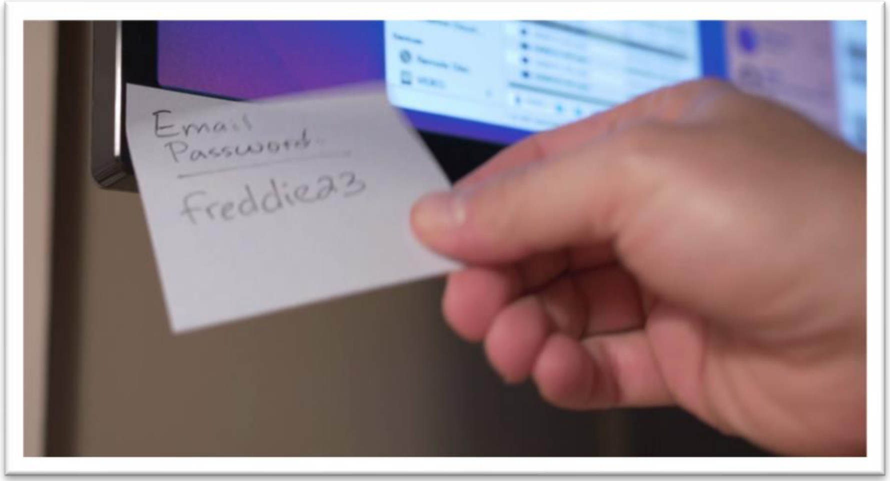
- Two-factor authentication? Attacker can manipulate a trusted party while you're away.
- Trusted internal network? Attacker breaks into mail room employee's email and sends a bugged PDF to the CEO.



We know humans are fallible.

So we need security to be easy.

- If humans had only 1KB of resilient storage, we'd be fine!
- If secure systems aren't easy, they either *fail open*, or they lead to forced compromises on behalf of the user.



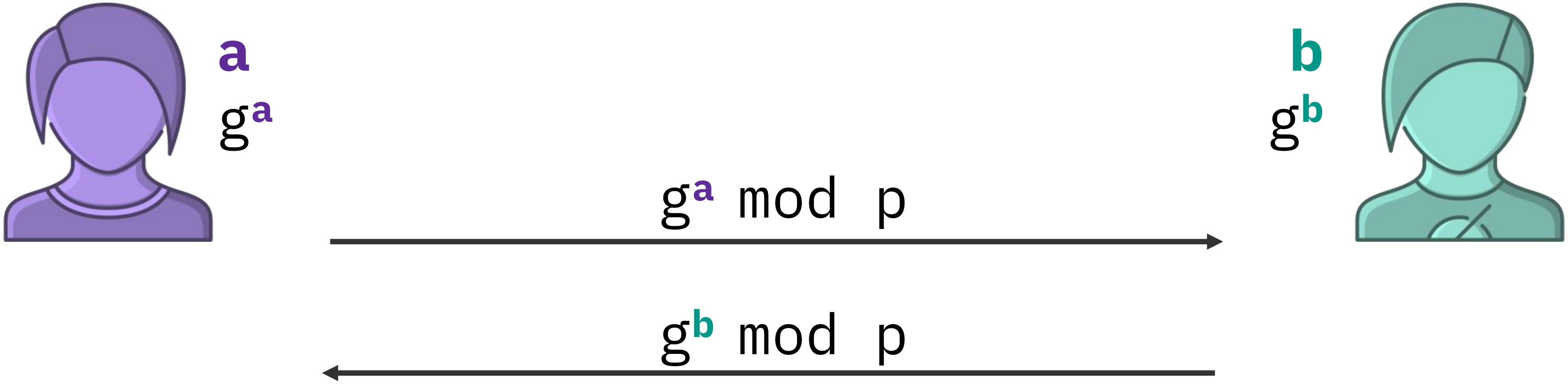
Email encryption: PGP.

- “Pretty Good Privacy” (1990s.)
- Created for email encryption:
    - Asynchronous (no online handshake necessary.)
    - Non-repudiable (binding signatures.)

Did you know?

PGP’s author, Phil Zimmermann, was criminally investigated in 1991 because PGP allegedly violated the Arms Export Control Act and was supposed to be classified as a munition.

Remember: Diffie-Hellman.



a

g^a

b

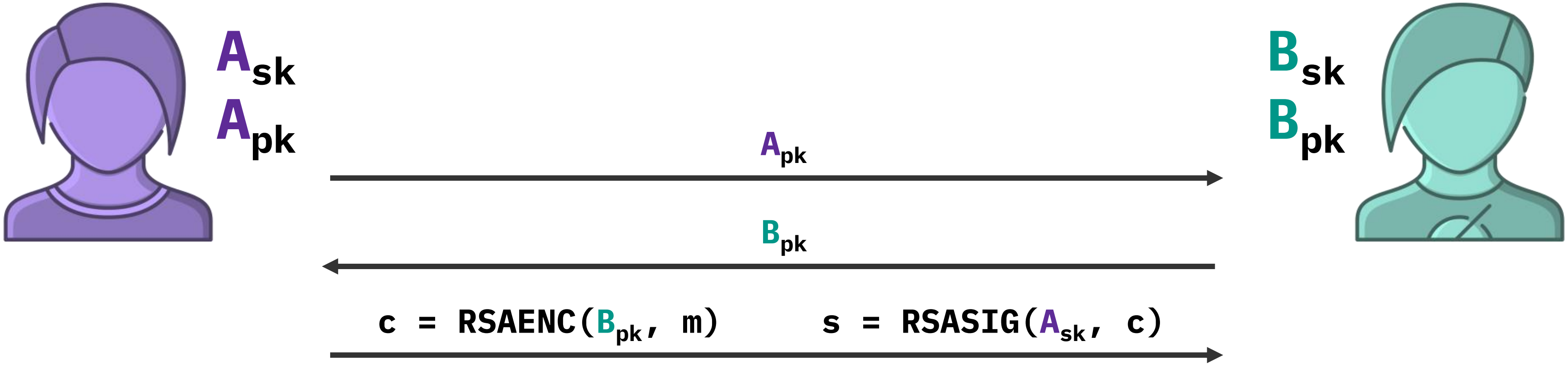
g^b

$g^a \bmod p$

$g^b \bmod p$

Public values:	g, p
Private keys:	a, b
Public keys:	g^a, g^b
Shared secret:	$g^{ab} \bmod p$

PGP works in a similar way (but with RSA.)



A_{sk}

A_{pk}

B_{sk}

B_{pk}

A_{pk}

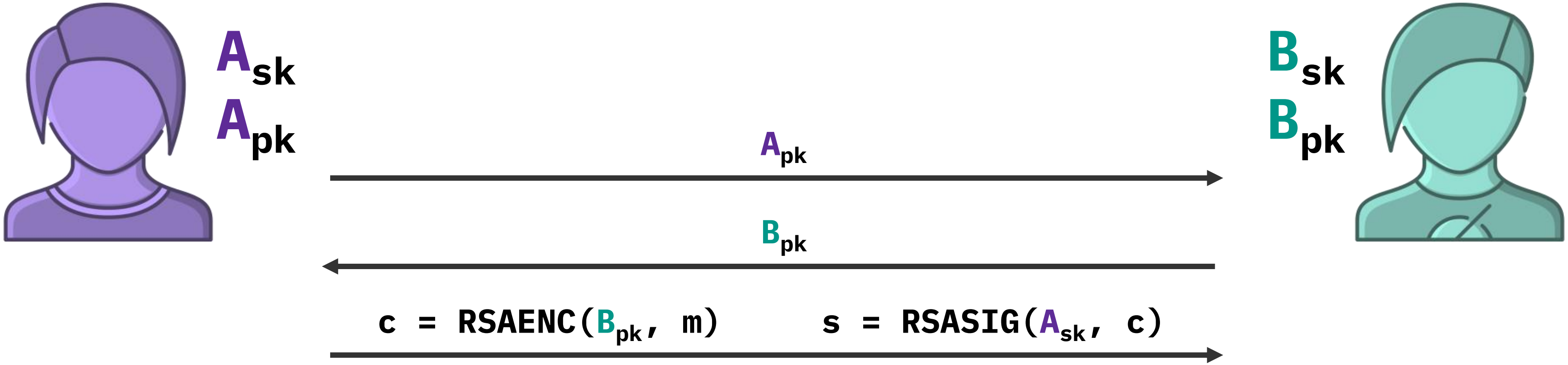
B_{pk}

$c = \mathtt{RSAENC}(B_{pk}, m)$ $s = \mathtt{RSASIG}(A_{sk}, c)$

$(\mathtt{true}|\mathtt{false}) = \mathtt{RSAVER}(A_{pk}, c)$ $m = \mathtt{RSADEC}(B_{sk}, c)$

RSA can be used for both *public key encryption* and for *public key signatures*.

What’s a possible attack for this scheme?



A_{sk}

A_{pk}

B_{sk}

B_{pk}

A_{pk}

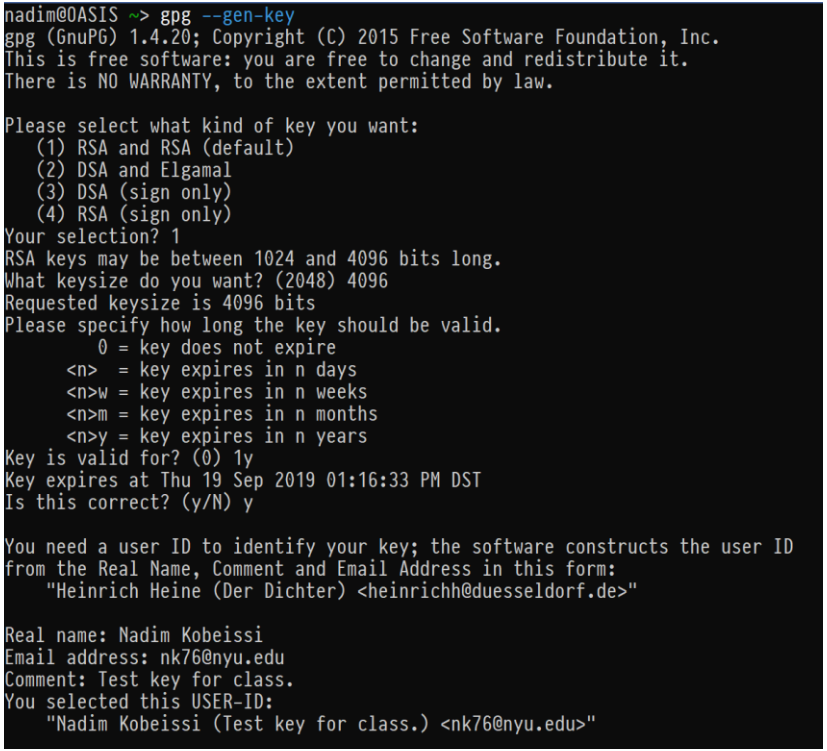
B_{pk}

$c = \mathrm{RSAENC}(B_{pk}, m) \qquad s = \mathrm{RSASIG}(A_{sk}, c)$

$(\mathrm{true}|\mathrm{false}) = \mathrm{RSAVER}(A_{pk}, c) \qquad m = \mathrm{RSADEC}(B_{sk}, c)$

RSA can be used for both *public key encryption* and for *public key signatures*.

PGP Step 1: Generate a key pair.

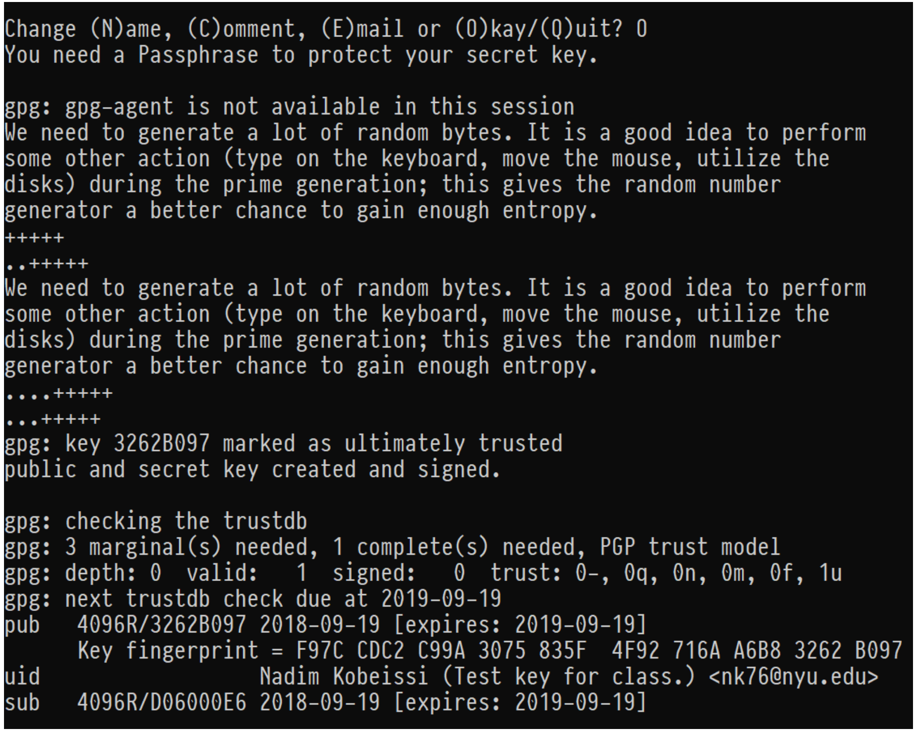


```
nadim@OASIS ~> gpg --gen-key
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 1y
Key expires at Thu 19 Sep 2019 01:16:33 PM DST
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Nadim Kobeissi
Email address: nk76@nyu.edu
Comment: Test key for class.
You selected this USER-ID:
    "Nadim Kobeissi (Test key for class.) <nk76@nyu.edu>"
```



```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

gpg: gpg-agent is not available in this session
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
+++++
..+++++
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
....+++++
...+++++
gpg: key 3262B097 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2019-09-19
pub   4096R/3262B097 2018-09-19 [expires: 2019-09-19]
      Key fingerprint = F97C CDC2 C99A 3075 835F  4F92 716A A6B8 3262 B097
uid                  Nadim Kobeissi (Test key for class.) <nk76@nyu.edu>
sub   4096R/D06000E6 2018-09-19 [expires: 2019-09-19]
```

PGP Step 2: export your public key.

- PGP public keys contain metadata, encryption public keys, signing public keys, etc.
- Public keys are uploaded to “key servers.”
- Other party must then fetch this key (some mail clients do this automatically.)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQINBFuiMDMBEACtolKCi+6PipgggL4LjBfWXq8G4bviAPVJSl0kyE9YdHZ++51u
23sJT4vgNat/sJGLHC9v8eEqwlhuQyoSeXYELChoxFsVxrDD3vSqdgALyx2cu9vM
QR+Q8MTfJlnzpqeW9wzbnmb8ciCRTguBJnHHylye1w6A9X57VtjZVu7/13WiWR1v
Sy83SvjayA1x0g3ioX9ENCbBGC0IPVMTvpvzq1MwqUK3g4geclov8mHC1ad0DqJt
HdjvKD1C1U/lZkRdo0wS7edSJd0n1hfXW4emhUiZbViYbaoMjOTExJftDTR05hC+
eYa3W0wlvYHNi7NuXbrzHB5vN5JLeSBMzH5dQ3+ytD8Nilk6b18zrZ0jRj628uon
QSkbl5hD9QaE9rUa+ie0bOUsZ1e4qoDizwkesKu/rqQwXISP3MieHkx2LzFsFI6A
0WFftNOt787xkptjuNXNxYK3gR2pfKJEEqW9PbCRG8BT6sMBEN5pNXzWXp9d5ikB
FIR8i7UriHxIfYq48GjtzK6dq8c5LXFlSrEg1A8XOf9KE9ccrBDcKC9GggF7/1yH
ExPciPvCq1XjCdCbj2HGzsn+ZpmOlM+zW6nOnTCpcCJw/nreHHD53aA6kcBshsf1
GNDorHI8gTestduMmz7oya2nstEmAaiH3CI/9J2Un1JTmF46Y14dt7VFWQARAQAB
tDNOYWRpbSBLb2JlaXNzaSAoVGVzdCBrZXkgZm9yIGNsYXNzLikgPG5rNzZAbnl1
LmVkdT6JAj4EEwECACgFAluiMDMCGwMFCQHhM4AGCwkIBwMCBhUIAgkKCwQWAgMB
Ah4BAheAAAoJEHFqprgyYrCXt4sP+QFlPztNTyFZIycnahTfeRSYipkcq9ND20sz
NiHNu53uTkGDt6fPUydyuMkm6M2xCqHy63VNmXtwThYoQpCpvwV2yZ6bULn7dCjh
usBmJuBl2aQVjFE8ZyXFi5V1mmkoiRqAOWrdvgy3ACqk3WSapeFWAZlYEJgVFWSY
Jk3nt2Twz3OJb4+LsKo6J9/kWCqp/7nRRJ8/iIsOTEvBjrwBL98acFbuxGrers6+
MGNdpLdkj4uDDgmsr+/Z30fgtk6cTIWevUKzOyJNB4Dkzhyy8QvVxRjCR2FGsLtU
qXoFTqoK6wBwedROAwBRRwmVO3t07jogDu+RiXCMM4IROhzZhL9MPPbkESmg0CLY
USFXYh5d1/BN2SWj3Z3ExFGtf6YS0MhKDk0FEGcqfDuQoJ99ffiM/o6mpoXSCJdS
Hc7yGnt4FfYk+yCwdg3F7tjxt0GT6aDtl4d40hNeJQJtKEFJB06IQPmmLCKYkXdB
5j/Ii0VwW6olq/UNiWpAy6IPZ3MMjobz0f9GsIpyXCD3UMJ5nrYm5rhn530eAEMh
ZkjyTd1izTBRtFNLsNL6Fwet55afa0X7Zv8wcIK0GDMue1ANxfzSRdtUvIyz3h3j
cQLl03yDOB8xtc5Q3PnS5D5jHgWB1Nt8AesazOp0pEKVl0t2r9G0tx7iWz4AA+v8
uJsoK81GuQINBFuiMDMBEACpZHp4cMT7nBaAZAjJDlXOFSRZuGkAf5UIAKxZMQC/
Ym3Z6yB6/uDW0tuaKeaeUKbFKPmFAHUAKIAMQG0WenvxH4Ftyuc7psiJQevQyouR
KUDb/WqRHsYMFm5cCaBmBe2zSKAOMLRRSAJp8Yxa3eQZ6XvDmBRoegFKC7g/AA0t
hZ5/rxgLUQaCYhz9qaz87luYuKos6+EPDpku21HX7nfMcYwZ+jfsgcnVrtxu+s7t
bSHd2unrfTS1jwTVCuBdFSYNrUMv4EUWxUFEhJw+yId74aeB/ENTyAJn6B+6hlU5
KbO4aRlcngDsgxC9fRqsiW+FtLK4TsO6KomgBwt1WjhqQiTPxpXIMcbssshwYjk3
9lej9j35o2b2dES2mg2yndrRJNyvj7hDYz5p/xJOu4cEy33jNk9CigzkiWm1Kfv4
50W6fq8ZOVhf44UjZ5H0oSwzrriMvPyzaUxjIoluQNErir0nwv3r4v41nfI81A+L
RMAszMLEVta2g7dy1zYxdUT5ZcMxpA8R/k+B9J6QZd7cu4s9k7FIPynU5JFfHyah
JBnDNYt9T1UoyQujyXPjITZqEaYpG4Q6vW0oLdDSRfT27gWWyI3hf4eXFVe96Ekh
+XdNPw55usULy8+2U3hLbIT2yMyQvAYJshHUMX2Mla1VAnNqmCFGX1OwA5eqXhdT
ewARAQABiQIlBBgBAgAPBQJbojAzAhsMBQkB4TOAAAoJEHFqprgyYrCXU/AQAJJD
0XRXLIVOd6TRIgrhi+8TEfzWK71KKfXDtzaWsCqBueHdX/q6dq9skieunPufNspQ
dhMGzlaJiuI50oC9OC3fy9wT08G1Gt2L2lKbCrmsQ6yOSpWNW3g7Gn1jiJSmf/Z4
S1ENgnRi2zsU4BVyLWkeosyzquBEeGrg3uKhI2FxOSSEVQJROMXQbRiRKHGCButx
GvvUWBumgbt50gkLo5yXXXhJUILiRJdVVCBTcU40E8pT7wSa2decBpj94LTu5Exv
B3TXXAycHgUxUcNyvzNAYc5GpaE5ldxVkapgFM8uTta299uUbPzSLvt5AMAB7KQ8
Z7u02WeyfmstUiOpPM5/06Nof95dXijNnUK4nIbRhcRZyjqW8uM7MeLhgJzUAx8D
Qxd1ZRx/XLktAvHSKHA+eziVrlcHYiFPJtR6hE/rZsxy43adsKMdunhl2IjI9Ofk
YoDfPb5TEQHO6mXVtFE4WkX0YkOn2LVe340jDN60i4pcvKIznOrKaxX2p+jnBOJc
8rEQK9US4r+noiP4JFSqgTYf4PmC9sAUpYzu4STz+luknaWxTZvp7yo6izfb3jq0
mg7OHqf6uZbL+5cy2hSCV/hJrxAR8iA9OQYUvtk8dA69XWlgJvOu9MsFRmbNUwSb
95AgRCY+hQWlDItVDdcsksEtk3w3sKvDKzLP27o8
=ndd4
-----END PGP PUBLIC KEY BLOCK-----
```

PGP Step 3: verify public key authenticity.

- To prevent man-in-the-middle attacks, Alice and Bob must verify a “key fingerprint,” which is essentially a hash of the public key.
- This is done out of band, sometimes even at “key parties” (the saddest kind of party.)

- New efforts: Autocrypt.
  - Automates key exchange (as we will see in secure messaging apps like WhatsApp.)
  - Does not yet support out-of-band auth.

PGP Step 4: set up a PGP-enabled mail client.

- Mozilla Thunderbird (desktop application.)
- Mailvelope (Gmail browser plugin.)
- K-9 Mail (Android phones.)

- Step 5: install PGP plugin.
- Step 6: import public keys.
- Step 7: send email.

Test your knowledge!

Does PGP provide *message integrity*?

□ **A:** Yes.

□ **B:** No.

Test your knowledge!

Does PGP provide *message integrity*?

☑ **A:** Yes.

□ **B:** No.

Test your knowledge!

Does PGP provide *forward secrecy*?

□ **A:** Yes.

□ **B:** No.

Test your knowledge!

Does PGP provide *forward secrecy*?

☐ **A:** Yes.

☑ **B:** No.

Test your knowledge!

Does PGP provide *ease of use*?

□ **A**: No.

□ **B**: No.

Test your knowledge!

Does PGP provide *ease of use*?

☑ **A:** No.

☑ **B:** No.

From PGP to Usable Systems

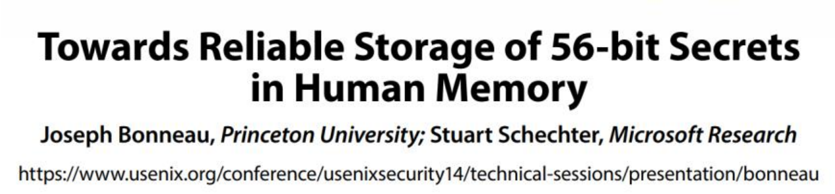
1.5b

Reasons not to use PGP.

- Very high likelihood of user error.
- Sending or forwarding a single plaintext email: leak entire thread.
- Downgrade attacks.
- Lack of obfuscation or traffic masking.
- No forward secrecy.
- Conflating authentication with non-repudiation.
- Complexity.
- Targeted attacks.

Usability patterns exist.

- Passphrases instead of random bytes.
- Two-factor, hardware-based authentication.
- Security by default.
- “Failing closed” instead of “failing open.”
- Upgrading user security with minimal changes to user behavior.



Towards Reliable Storage of 56-bit Secrets in Human Memory

Joseph Bonneau, *Princeton University;* Stuart Schechter, *Microsoft Research*

https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/bonneau

Examples of usable security systems.

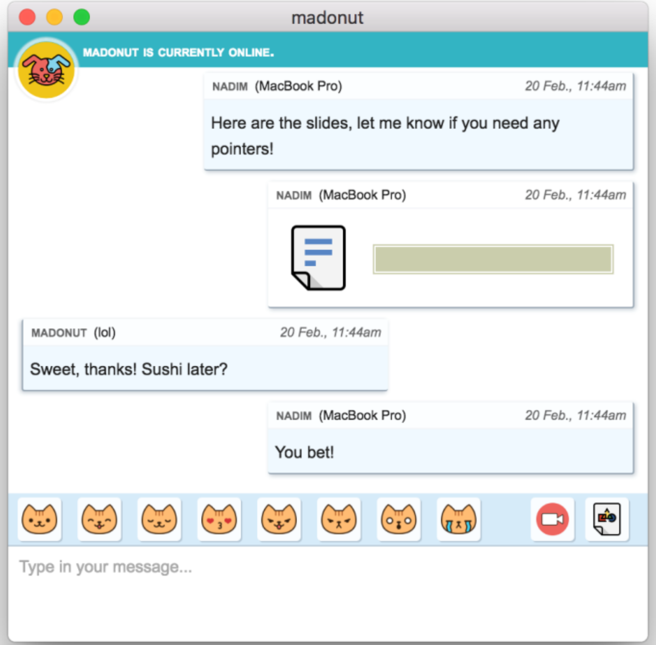
- Touch ID, Face ID.
- Apple Pay, Android Pay, Samsung Pay.
- YubiKey and two-factor authentication.
- HTTPS and TLS.
- Let’s Encrypt.
- Secure messaging.
- ATMs and more.

What do these systems have in common?

- Fail closed, not open.
- Minimal memorization of user secrets.
- High availability.
- Resilience to user error.

My usable security contribution in 2011.

- Cryptocat: my pet project (pun intended), First end-to-end encrypted secure messenger to work in any web browser!
- Sadly, many security flaws and programming errors...
- Re-release in 2016 as the first formally verified secure messenger! Now secure.
- Small user base due to no mobile apps.



NADIM (MacBook Pro) 20 Feb., 11:44am
Here are the slides, let me know if you need any pointers!

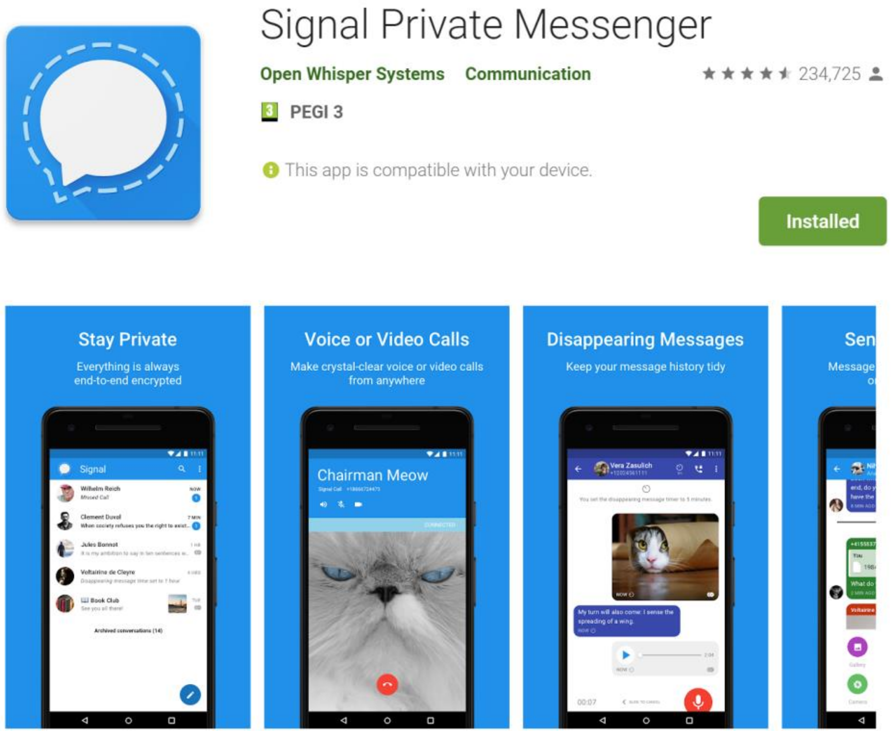
NADIM (MacBook Pro) 20 Feb., 11:44am

MADONUT (lol) 20 Feb., 11:44am
Sweet, thanks! Sushi later?

NADIM (MacBook Pro) 20 Feb., 11:44am
You bet!

Signal and usable security in messaging.

- Signal brought asynchronous, modern secure messaging to mobile devices.
  - Was based on OTR (synchronous, slower, less secure.)
  - *Future secrecy, Trust on First Use...*
  - Separates authentication from non-repudiation.
- Licensed by WhatsApp, Google, etc.
- Today faces strong competition from Wire, iMessage.



Signal Private Messenger

Open Whisper Systems Communication ★★★★☆ 234,725

PEGI 3

This app is compatible with your device.

Installed

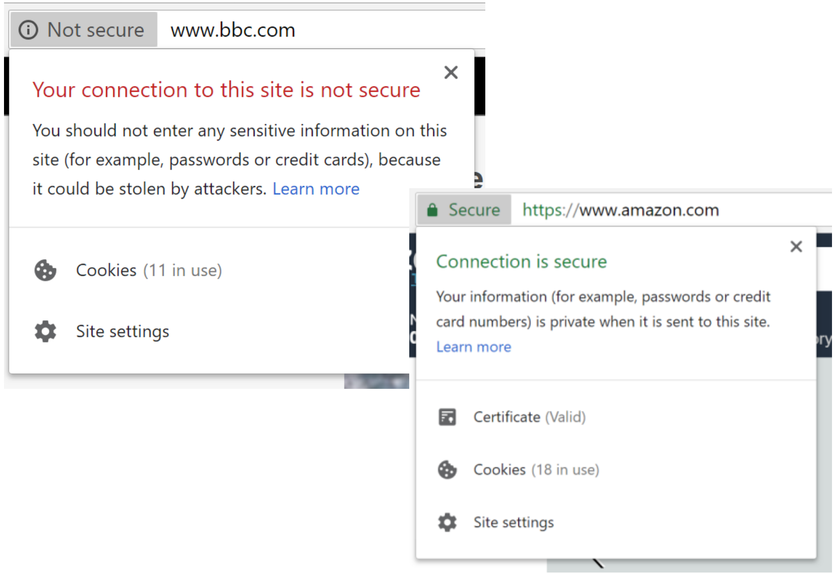
Stay Private
Everything is always end-to-end encrypted
Voice or Video Calls
Make crystal-clear voice or video calls from anywhere
Disappearing Messages
Keep your message history tidy
Sen
Message
Signal
Chairman Meow
Vera Zasulich

Example of a usable security design.

- *Trust on first use*: instead of mandating out-of-band public key verification, trust the first received key but throw alarm bells in case changes detected.
- *Mimicking traditional usage patterns*: Signal allowed messaging offline users like SMS, thereby not requiring changes in user behavior.

Security engineers are always adapting.

- Since July 2018, Google Chrome has been marking all non-HTTPS websites as "Not secure." This only makes sense due to recent huge increase in HTTPS adoption.
- Security engineers are always experimenting with new trends and methodologies.
- Research in usable security tends to be "softer" and more subjective than other computer security but is still important.



ⓘ Not secure www.bbc.com

Your connection to this site is not secure

You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers. Learn more

Secure https://www.amazon.com

Connection is secure

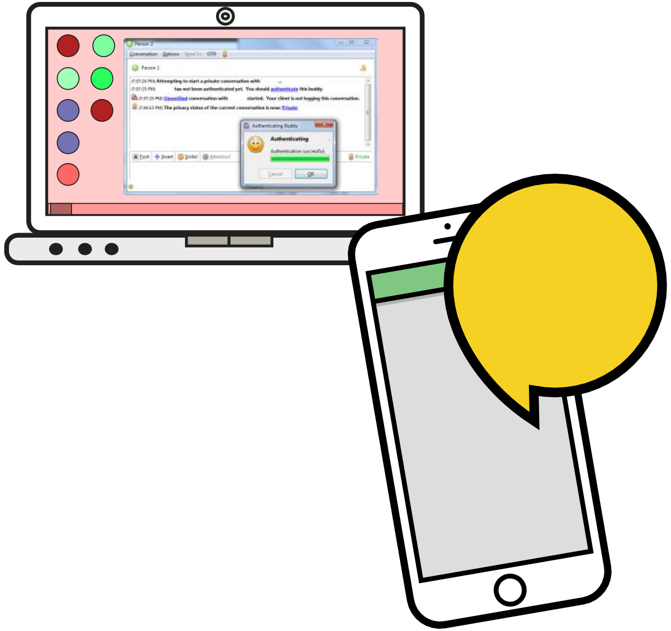
Your information (for example, passwords or credit card numbers) is private when it is sent to this site. Learn more

Off-the-Record Messaging (2004)

1.5c

An increasingly central use case.

- Before smartphones (early 2000s), instant messaging was a laptop/desktop affair: Google Chat, MSN Messenger, Yahoo Messenger, AIM...
- After the iPhone (2007), we had powerful computers in our pockets and instant messaging rocketed to arguably the most important user communications use case.



(7:37:24 PM) Attempting to start a private conversation with

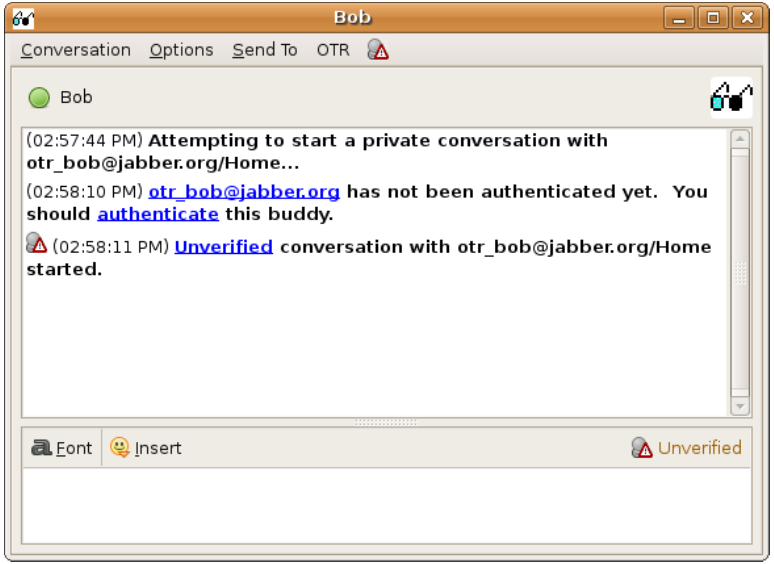
(7:37:25 PM) has not been authenticated yet. You should authenticate this buddy.

(7:37:25 PM) Unverified conversation with started. Your client is not logging this conversation.

(7:44:13 PM) The privacy status of the current conversation is now: Private

Off-the-record messaging.

- Presented by Nikita Borisov, Ian Goldberg and Eric A. Brewer in 2004.
- Plugin for then-popular IM clients (mainly Pidgin and Adium.)
- Opportunistic, platform-agnostic encryption.



(02:57:44 PM) Attempting to start a private conversation with otr_bob@jabber.org/Home...

(02:58:10 PM) otr_bob@jabber.org has not been authenticated yet. You should authenticate this buddy.

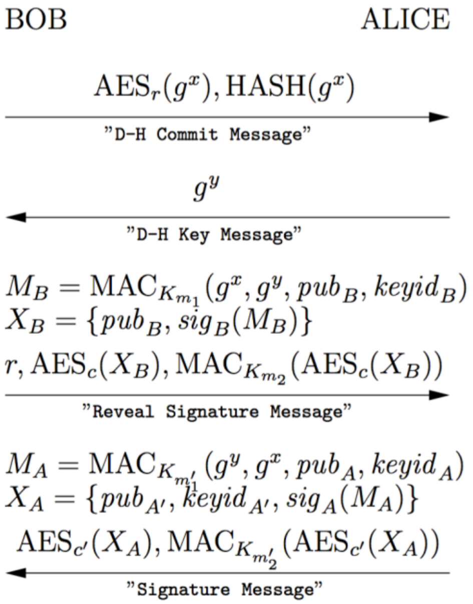
(02:58:11 PM) Unverified conversation with otr_bob@jabber.org/Home started.

Did you know?

The OTR paper was titled *“Off-the-Record Communication, or, Why Not To Use PGP”*, giving a clear hint as to an impetus behind the project.

Off-the-record messaging.

- Unlike PGP, OTR is a synchronous protocol: both parties must be online.
- Also unlike PGP, compromising OTR long-term signing/encryption keys did not lead to message decryption!
- A new targeted property: deniability.



BOB ALICE

$\text{AES}_r(g^x), \text{HASH}(g^x)$

”D-H Commit Message”

g^y

”D-H Key Message”

$M_B = \text{MAC}_{K_{m_1}}(g^x, g^y, pub_B, keyid_B)$

$X_B = \{pub_B, sig_B(M_B)\}$

$r, \text{AES}_c(X_B), \text{MAC}_{K_{m_2}}(\text{AES}_c(X_B))$

”Reveal Signature Message”

$M_A = \text{MAC}_{K_{m'_1}}(g^y, g^x, pub_A, keyid_A)$

$X_A = \{pub_{A'}, keyid_{A'}, sig_A(M_A)\}$

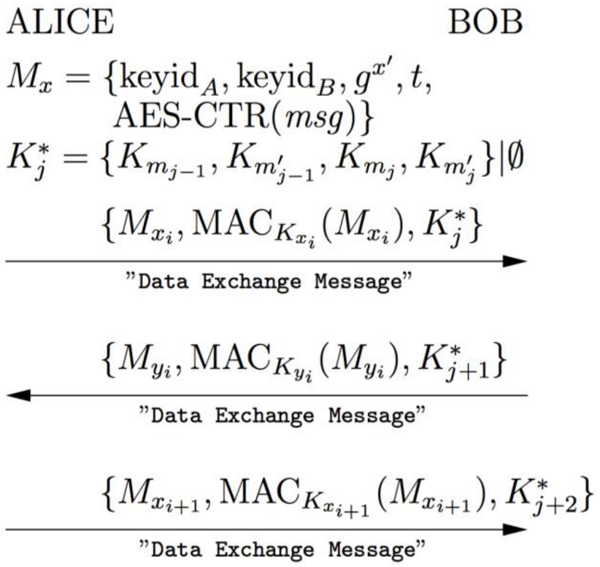
$\text{AES}_{c'}(X_A), \text{MAC}_{K_{m'_2}}(\text{AES}_{c'}(X_A))$

”Signature Message”

Figure 1: OTR authenticated key exchange protocol

Off-the-record messaging.

- Unlike PGP, OTR is a synchronous protocol: both parties must be online.
- Also unlike PGP, compromising OTR long-term signing/encryption keys did not lead to message decryption!
- Both parties contribute to “freshness” of upcoming keys.



$$M_x = \{\text{keyid}_A, \text{keyid}_B, g^{x'}, t, \text{AES-CTR}(msg)\}$$

$$K_j^* = \{K_{m_{j-1}}, K_{m'_{j-1}}, K_{m_j}, K_{m'_j}\}|\emptyset$$

$$\{M_{x_i}, \text{MAC}_{K_{x_i}}(M_{x_i}), K_j^*\}$$

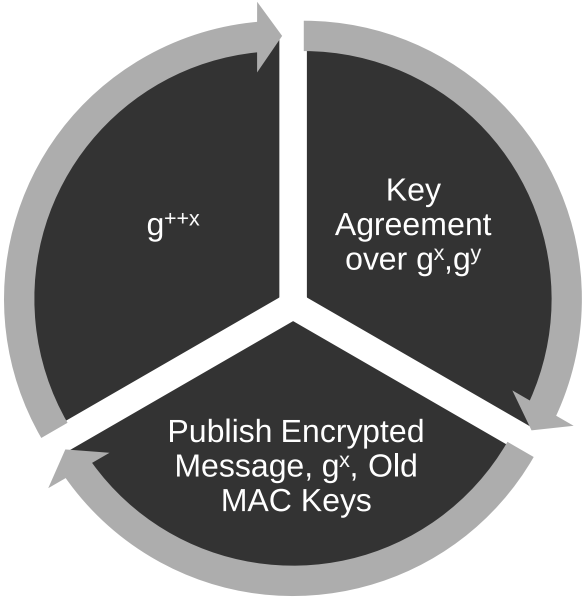
$$\{M_{y_i}, \text{MAC}_{K_{y_i}}(M_{y_i}), K_{j+1}^*\}$$

$$\{M_{x_{i+1}}, \text{MAC}_{K_{x_{i+1}}}(M_{x_{i+1}}), K_{j+2}^*\}$$

Figure 2: OTR data exchange protocol

Off-the-record messaging.

- Unlike PGP, OTR is a synchronous protocol: both parties must be online.
- Also unlike PGP, compromising OTR long-term signing/encryption keys did not lead to message decryption!
- Both parties contribute to “freshness” of upcoming keys.


g^{++x}
Key Agreement over g^x,g^y
Publish Encrypted Message, g^x, Old MAC Keys

Test your knowledge!

What is the correct term for the OTR security property we just discussed?

□ **A**: Confidentiality.

□ **B**: Integrity.

□ **C**: Forward secrecy.

Test your knowledge!

What is the correct term for the OTR security property we just discussed?

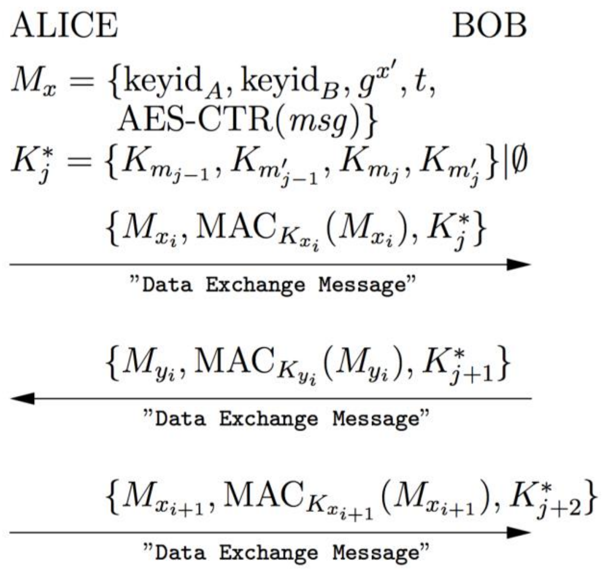
□ **A:** Confidentiality.

□ **B:** Integrity.

☑ **C:** Forward secrecy.

Finite State Analysis of OTR (Bonneau et al)

- *Version Rollback*: Attacker can force obsolete OTR protocol version. Fix by integrating version number into AKE.
- *Attack on Strong Deniability*: Possible via strong network attacker.
- *Authentication Failure*: Mallory can convince Alice of a successful AKE with Bob, even if Bob has no idea what is going on.



ALICE BOB

$$M_x = \{\text{keyid}_A, \text{keyid}_B, g^{x'}, t, \text{AES-CTR}(msg)\}$$

$$K_j^* = \{K_{m_{j-1}}, K_{m'_{j-1}}, K_{m_j}, K_{m'_j}\}|\emptyset$$

$$\xrightarrow{\{M_{x_i}, \text{MAC}_{K_{x_i}}(M_{x_i}), K_j^*\}}$$

”Data Exchange Message”

$$\xleftarrow{\{M_{y_i}, \text{MAC}_{K_{y_i}}(M_{y_i}), K_{j+1}^*\}}$$

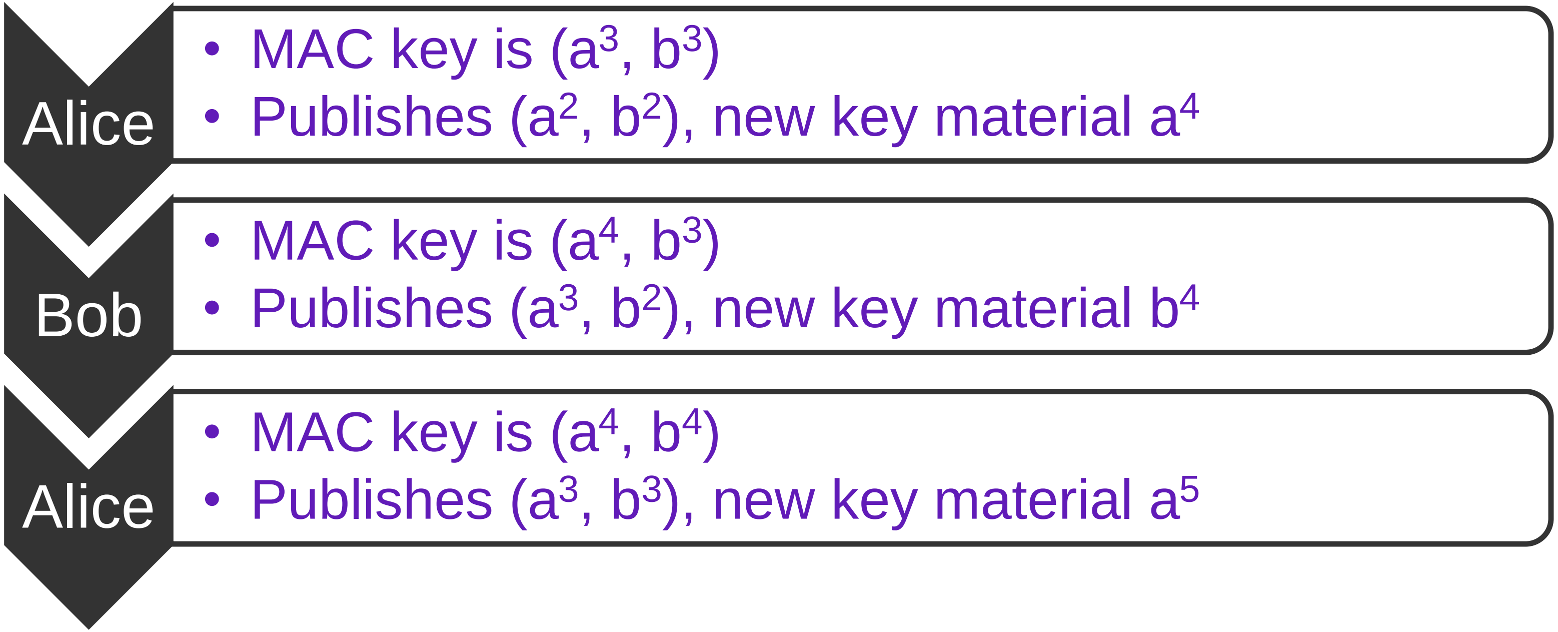
”Data Exchange Message”

$$\xrightarrow{\{M_{x_{i+1}}, \text{MAC}_{K_{x_{i+1}}}(M_{x_{i+1}}), K_{j+2}^*\}}$$

”Data Exchange Message”

Figure 2: OTR data exchange protocol

OTR: message integrity attack.



Alice

- MAC key is (a^3, b^3)
- Publishes (a^2, b^2), new key material a^4

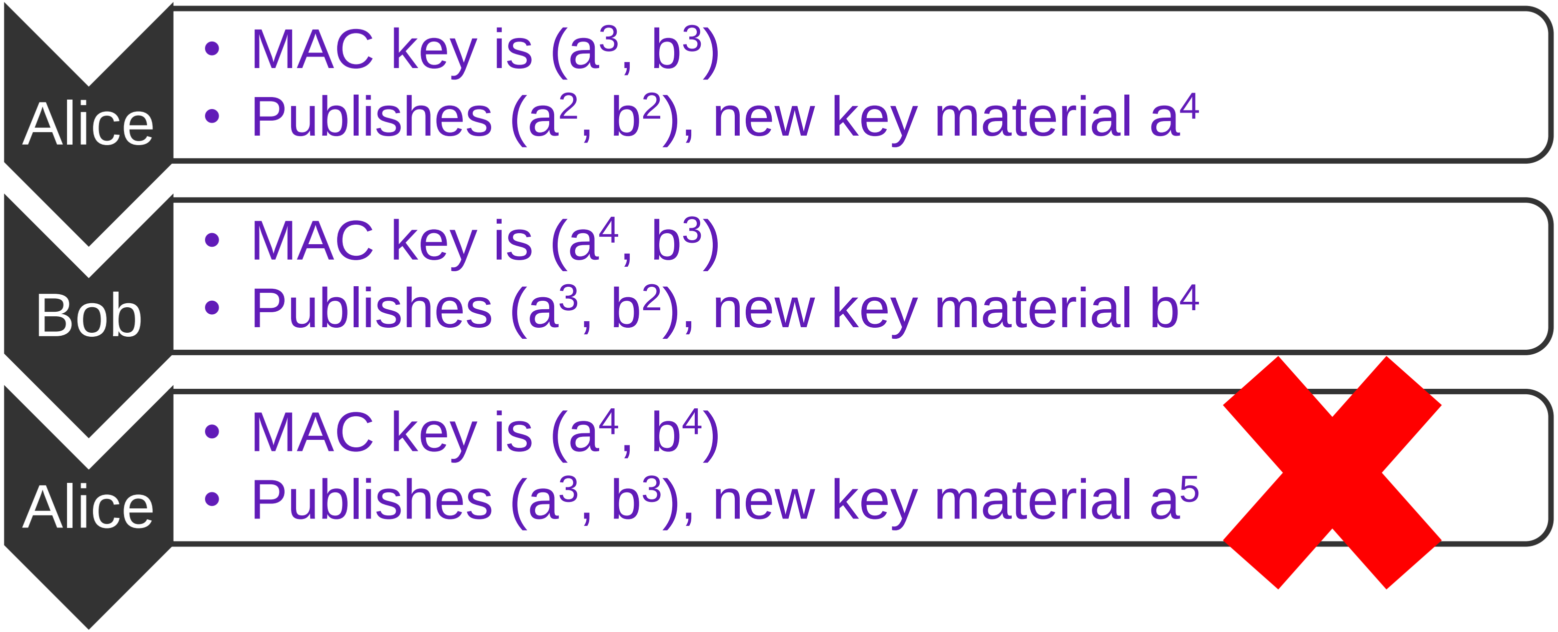
Bob

- MAC key is (a^4, b^3)
- Publishes (a^3, b^2), new key material b^4

Alice

- MAC key is (a^4, b^4)
- Publishes (a^3, b^3), new key material a^5

OTR: message integrity attack.



Alice

- MAC key is (a^3, b^3)
- Publishes (a^2, b^2), new key material a^4

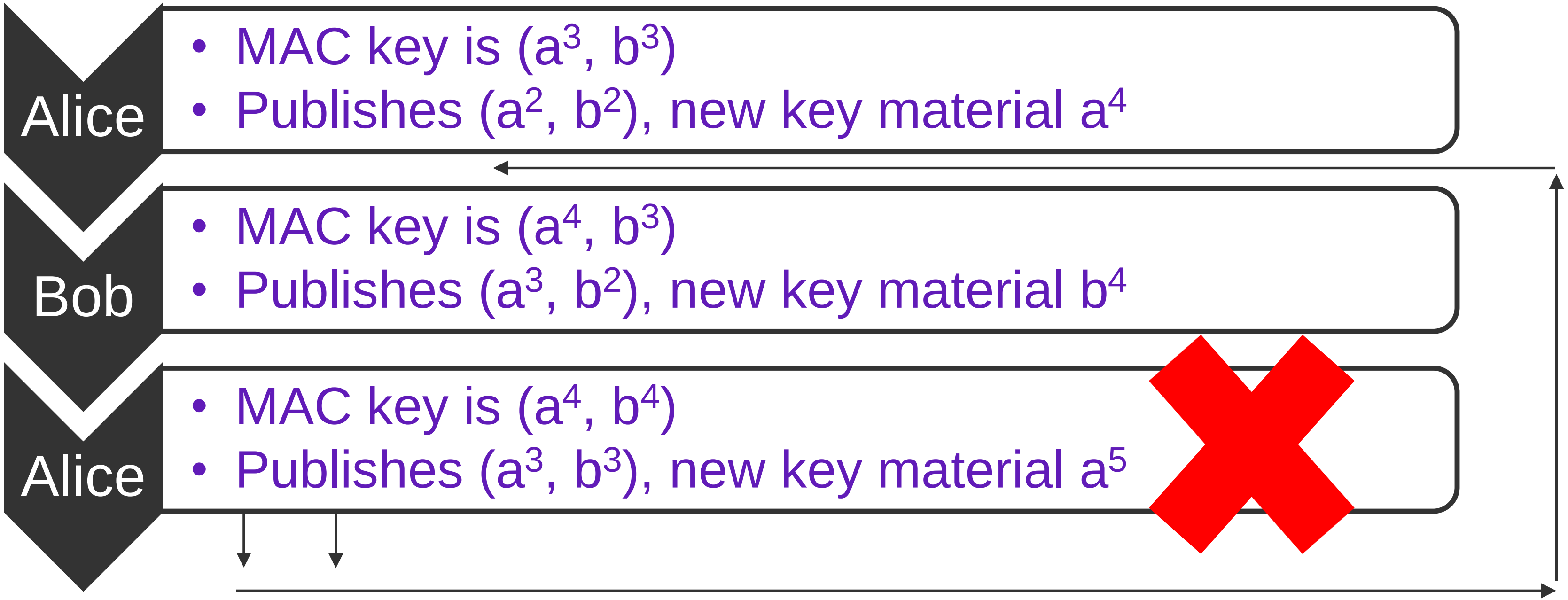
Bob

- MAC key is (a^4, b^3)
- Publishes (a^3, b^2), new key material b^4

Alice

- MAC key is (a^4, b^4)
- Publishes (a^3, b^3), new key material a^5

OTR: message integrity attack.



Alice

- MAC key is (a^3, b^3)
- Publishes (a^2, b^2), new key material a^4

Bob

- MAC key is (a^4, b^3)
- Publishes (a^3, b^2), new key material b^4

Alice

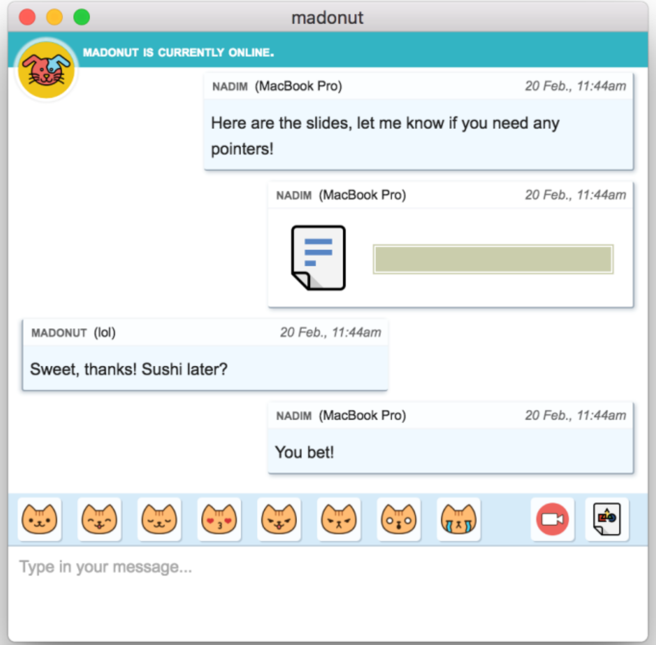
- MAC key is (a^4, b^4)
- Publishes (a^3, b^3), new key material a^5

Signal Protocol (2013)

1.5d

My usable security contribution in 2011.

- Cryptocat: my pet project (pun intended), First end-to-end encrypted secure messenger to work in any web browser!
- Sadly, many security flaws and programming errors…
- Re-release in 2016 as the first formally verified secure messenger! Now secure.
- Small user base due to no mobile apps.



NADIM (MacBook Pro) 20 Feb., 11:44am

Here are the slides, let me know if you need any pointers!

NADIM (MacBook Pro) 20 Feb., 11:44am

MADONUT (lol) 20 Feb., 11:44am

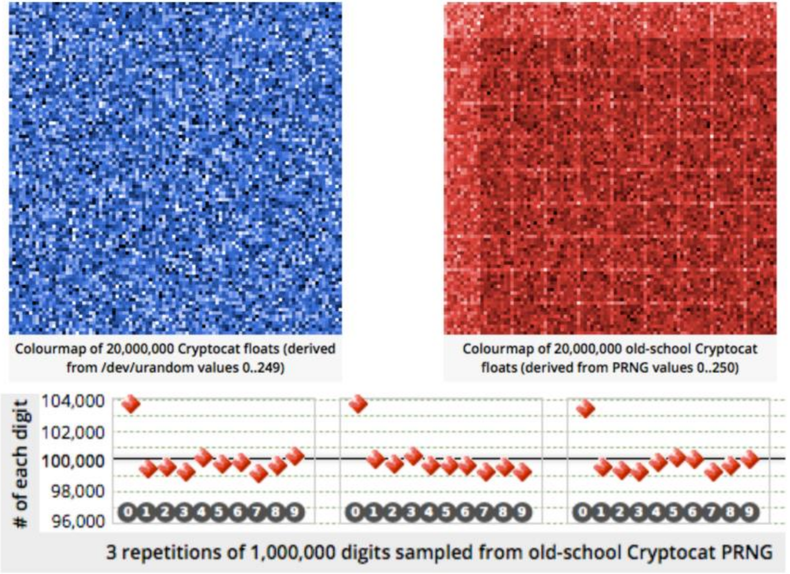
Sweet, thanks! Sushi later?

NADIM (MacBook Pro) 20 Feb., 11:44am

You bet!

Cryptocat: bugs in 2011-2012.

- *Weak private keys:* Instead of generating 16 bytes (0-255), Cryptocat generated 16 bytes (0-10) by mistake.
- *Nonce-reuse*: AES-CTR encryption was used but with matching nonce counters on both sides.
- *Biased PRNG:* Generating values were not indistinguishable from random.



Colourmap of 20,000,000 Cryptocat floats (derived from /dev/urandom values 0..249)

Colourmap of 20,000,000 old-school Cryptocat floats (derived from PRNG values 0..250)

3 repetitions of 1,000,000 digits sampled from old-school Cryptocat PRNG

Test your knowledge!

Why is it a problem if Cryptocat generates private keys with each byte being in the range 0-10?

Test your knowledge!

Why is it a problem if Cryptocat generates private keys with each byte being in the range 0-10?

Because the overall entropy of that private key would be $10^{16} = 2^{53}$. Instead of $255^{16} = 2^{128}$.

Test your knowledge!

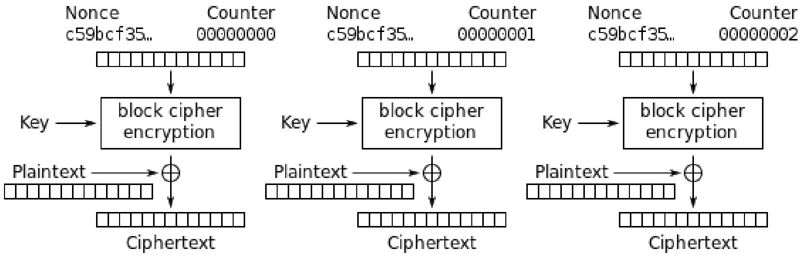
Why is it a problem if AES-CTR reuses nonces?

Isn’t AES a block cipher?

Test your knowledge!

Why is it a problem if AES-CTR reuses nonces? Isn't AES a block cipher?

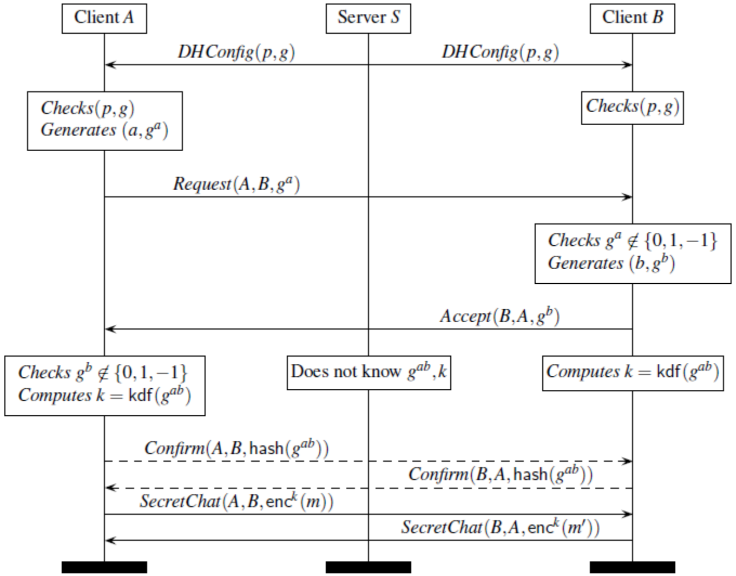
Counter mode makes block ciphers work like stream ciphers. $c_1 \oplus c_2 = m_1 \oplus m_2$



Counter (CTR) mode encryption

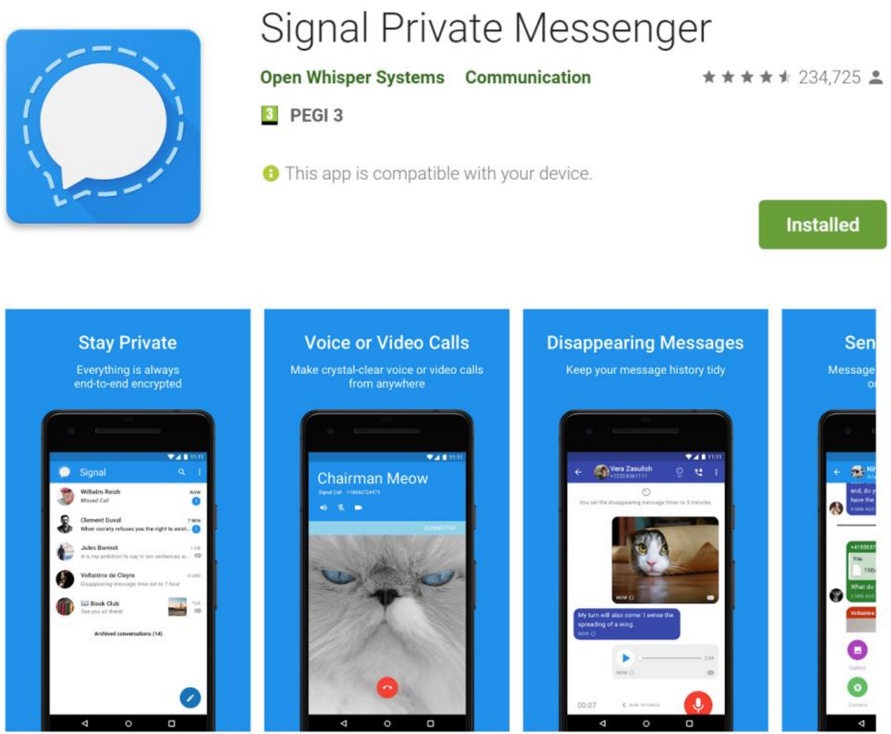
Telegram “Secret Chats”: quick overview.

- Diffie-Hellman values sent dynamically.
- Naïve Diffie-Hellman used for authentication and encryption.
- No re-keying.


Client A
Server S
Client B
$DHConfig(p,g)$
$DHConfig(p,g)$
$Checks(p,g)$
$Generates\ (a,g^a)$
$Checks(p,g)$
$Request(A,B,g^a)$
$Checks\ g^a \notin \{0,1,-1\}$
$Generates\ (b,g^b)$
$Accept(B,A,g^b)$
$Checks\ g^b \notin \{0,1,-1\}$
$Computes\ k = \mathsf{kdf}(g^{ab})$
Does not know g^{ab},k
$Computes\ k = \mathsf{kdf}(g^{ab})$
$Confirm(A,B,\mathsf{hash}(g^{ab}))$
$Confirm(B,A,\mathsf{hash}(g^{ab}))$
$SecretChat(A,B,\mathsf{enc}^k(m))$
$SecretChat(B,A,\mathsf{enc}^k(m'))$

Signal and usable security in messaging.

- Signal brought asynchronous, modern secure messaging to mobile devices.
  - Was based on OTR (synchronous, slower, less secure.)
  - *Future secrecy, Trust on First Use...*
  - Separates authentication from non-repudiation.
- Licensed by WhatsApp, Google, etc.
- Today faces strong competition from Wire, iMessage.



Signal Private Messenger

Open Whisper Systems Communication ★★★★☆ 234,725

PEGI 3

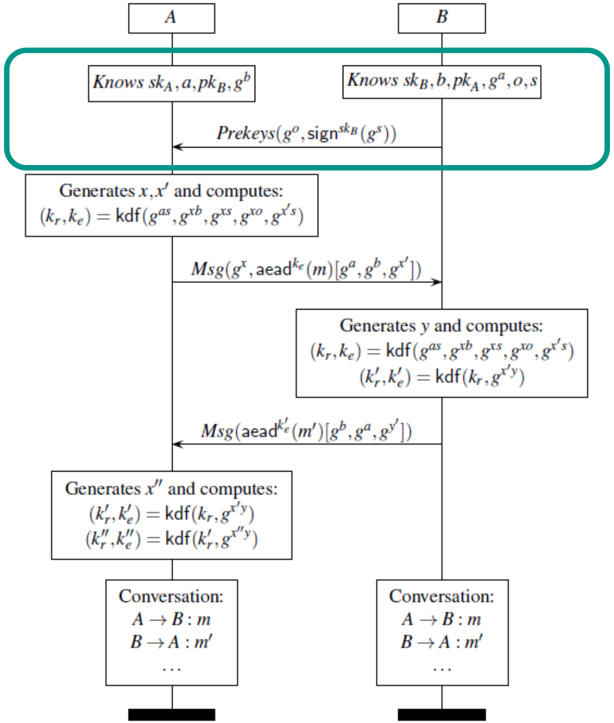
This app is compatible with your device.

Installed

Stay Private
Everything is always end-to-end encrypted
Voice or Video Calls
Make crystal-clear voice or video calls from anywhere
Disappearing Messages
Keep your message history tidy
Sen
Message
Signal
Chairman Meow
Vera Zasulich

Signal Protocol: overview.

- Four-way Diffie-Hellman in AKE step.
- Complex key schedule for ratcheting between messages.
- Offers offline messaging (due to zero-round-trip AKE.)
- Each party uploads 100 one-time ephemerals to the server, so Alice can send message without Bob being online.
- Re-keying like OTR, but each party uses its key immediately.



A

B

Knows sk_A, a, pk_B, g^b

Knows sk_B, b, pk_A, g^a, o, s

$Prekeys(g^o, \mathsf{sign}^{sk_B}(g^s))$

Generates x, x' and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$

$Msg(g^x, \mathsf{aead}^{k_e}(m)[g^a, g^b, g^{x'}])$

Generates y and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$

$Msg(\mathsf{aead}^{k'_e}(m')[g^b, g^a, g^{y'}])$

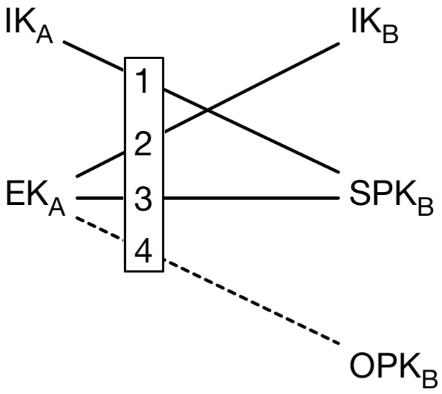
Generates x'' and computes:
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$(k''_r, k''_e) = \mathsf{kdf}(k'_r, g^{x''y})$

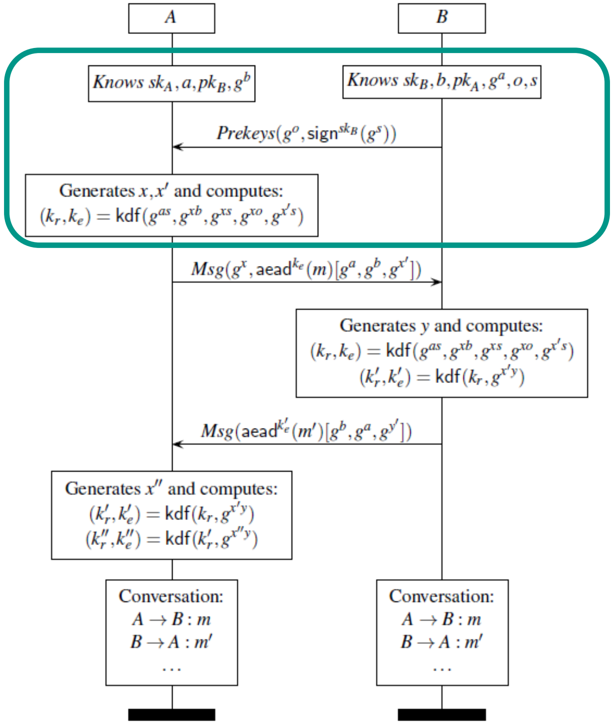
Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\dots$

Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\dots$

Signal Protocol: Asynchronous AKE (X3DH.)

- Each party uploads 100 one-time ephemerals to the server, so Alice can send message without Bob being online.
- Re-keying like OTR, but each party uses its key immediately.





A

B

Knows sk_A, a, pk_B, g^b

Knows sk_B, b, pk_A, g^a, o, s

$Prekeys(g^o, \mathsf{sign}^{sk_B}(g^s))$

Generates x, x' and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$

$Msg(g^x, \mathsf{aead}^{k_e}(m)[g^a, g^b, g^{x'}])$

Generates y and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$

$Msg(\mathsf{aead}^{k'_e}(m')[g^b, g^a, g^{y'}])$

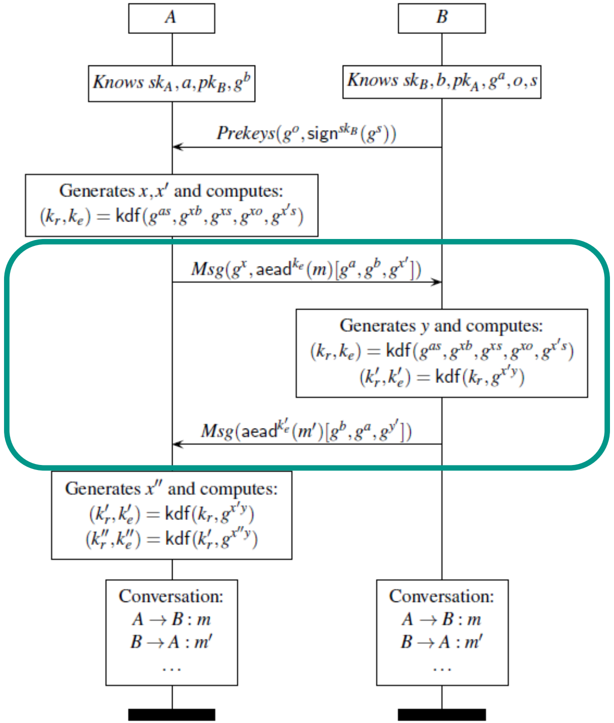
Generates x'' and computes:
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$(k''_r, k''_e) = \mathsf{kdf}(k'_r, g^{x''y})$

Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\ldots$

Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\ldots$

Signal Protocol: messaging.

- Alice can now send her message (after performing an HKDF on the obtained AKE master secret.)
- Later, Bob can go online and do his share of the AKE to decrypt the message and establish the session.



A

B

$Knows\ sk_A, a, pk_B, g^b$

$Knows\ sk_B, b, pk_A, g^a, o, s$

$Prekeys(g^o, \mathsf{sign}^{sk_B}(g^s))$

Generates x, x' and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$

$Msg(g^x, \mathsf{aead}^{k_e}(m)[g^a, g^b, g^{x'}])$

Generates y and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$

$Msg(\mathsf{aead}^{k'_e}(m')[g^b, g^a, g^{y'}])$

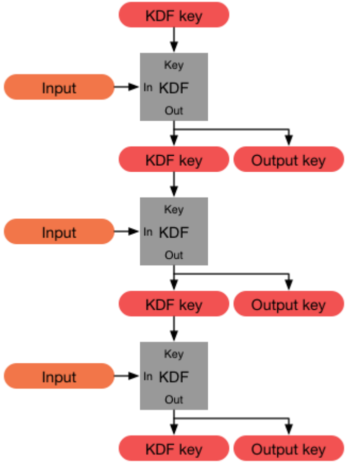
Generates x'' and computes:
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$(k''_r, k''_e) = \mathsf{kdf}(k'_r, g^{x''y})$

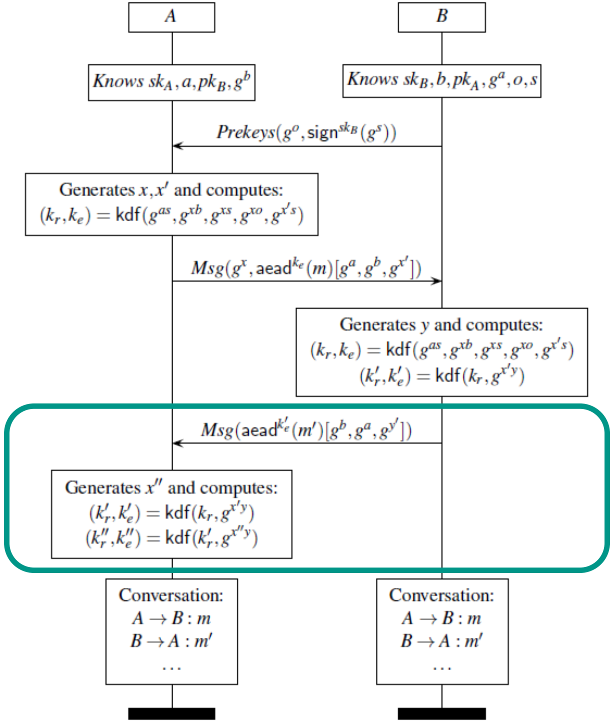
Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\ldots$

Conversation:
$A \rightarrow B : m$
$B \rightarrow A : m'$
$\ldots$

Signal Protocol: “double ratchet.”

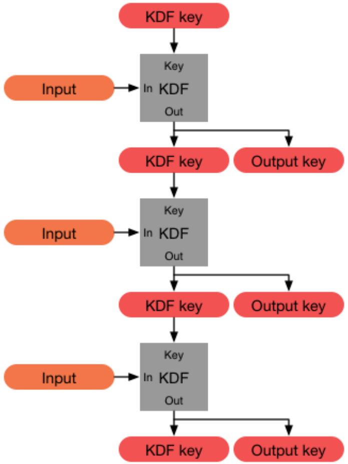
- Every message has a new ephemeral value $g^{x'}$.
- New keys derived from old keys + $g^{yx'}$ (new x, old y.)

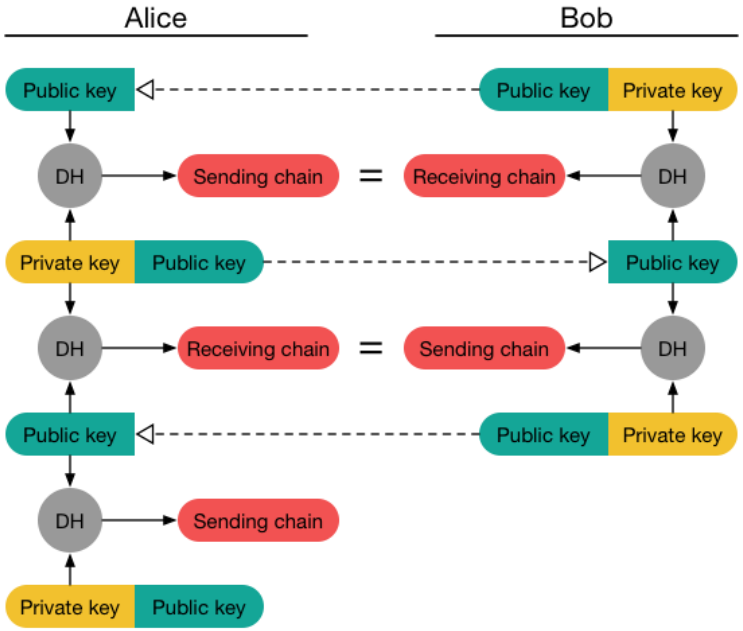



A
B
$Knows\ sk_A, a, pk_B, g^b$
$Knows\ sk_B, b, pk_A, g^a, o, s$
$Prekeys(g^o, \mathsf{sign}^{sk_B}(g^s))$
Generates x, x' and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$Msg(g^x, \mathsf{aead}^{k_e}(m)[g^a, g^b, g^{x'}])$
Generates y and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$Msg(\mathsf{aead}^{k'_e}(m')[g^b, g^a, g^{y'}])$
Generates x'' and computes:
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$(k''_r, k''_e) = \mathsf{kdf}(k'_r, g^{x''y})$
Conversation:
$A \to B : m$
$B \to A : m'$
$\dots$
Conversation:
$A \to B : m$
$B \to A : m'$
$\dots$

Signal Protocol: “double ratchet.”

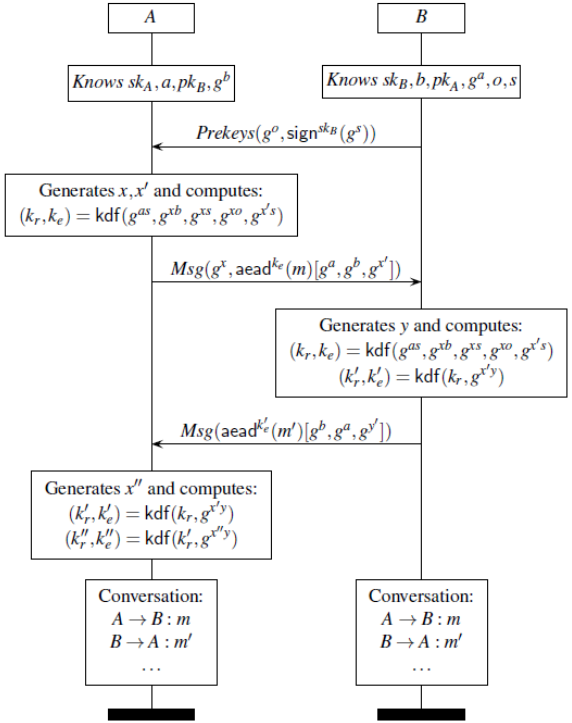
- Every message has a new ephemeral value $g^{x'}$.
- New keys derived from old keys + $g^{yx'}$ (new x, old y.)





Signal Protocol: weaknesses found.

- Bob may have used Carol’s ephemerals (unknown key share attack).
- Attacker can exhaust B’s one-time keys, first message can be replayed.
- Key compromise impersonation: if I compromise Bob’s private keys, I can impersonate Alice *to* Bob.



A

B

$Knows\ sk_A, a, pk_B, g^b$

$Knows\ sk_B, b, pk_A, g^a, o, s$

$Prekeys(g^o, \mathsf{sign}^{sk_B}(g^s))$

Generates x, x' and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$

$Msg(g^x, \mathsf{aead}^{k_e}(m)[g^a, g^b, g^{x'}])$

Generates y and computes:
$(k_r, k_e) = \mathsf{kdf}(g^{as}, g^{xb}, g^{xs}, g^{xo}, g^{x's})$
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$

$Msg(\mathsf{aead}^{k'_e}(m')[g^b, g^a, g^{y'}])$

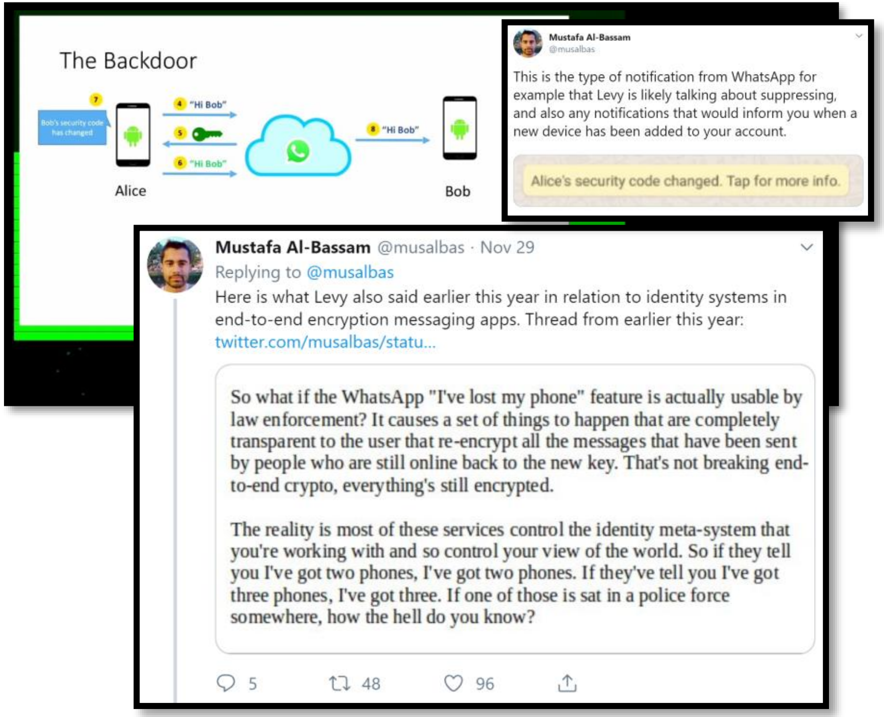
Generates x'' and computes:
$(k'_r, k'_e) = \mathsf{kdf}(k_r, g^{x'y})$
$(k''_r, k''_e) = \mathsf{kdf}(k'_r, g^{x''y})$

Conversation:
$A \to B : m$
$B \to A : m'$
$\dots$

Conversation:
$A \to B : m$
$B \to A : m'$
$\dots$

Controversies in WhatsApp implementation.

- Backups not encrypted.
- Vulnerable to man-in-the-middle by an attacker that controls GSM network!
    - I can deregister your device/phone number and register a new device.
    - WhatsApp will **automatically re-encrypt previous messages sent to you and send them to the new device.**
    - Not a backdoor, but still problematic. Can be mitigated by adding an “account PIN”.
    - GCHQ official stated on the record that this is an attack vector they would use for surveillance.



The Backdoor

Mustafa Al-Bassam
@musalbas

This is the type of notification from WhatsApp for example that Levy is likely talking about suppressing, and also any notifications that would inform you when a new device has been added to your account.

Alice's security code changed. Tap for more info.

Mustafa Al-Bassam @musalbas · Nov 29

Replying to @musalbas

Here is what Levy also said earlier this year in relation to identity systems in end-to-end encryption messaging apps. Thread from earlier this year: twitter.com/musalbas/statu...

So what if the WhatsApp "I've lost my phone" feature is actually usable by law enforcement? It causes a set of things to happen that are completely transparent to the user that re-encrypt all the messages that have been sent by people who are still online back to the new key. That's not breaking end-to-end crypto, everything's still encrypted.

The reality is most of these services control the identity meta-system that you're working with and so control your view of the world. So if they tell you I've got two phones, I've got two phones. If they've tell you I've got three phones, I've got three. If one of those is sat in a police force somewhere, how the hell do you know?

Signal Protocol: group chat.

- Alice creates a Signal session with Bob, Carol, etc.
- Individually re-encrypts message to each session.
- Excellent analysis which finds many attacks on this and similar approaches: *“More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema.”*
    - Available as part of your readings.

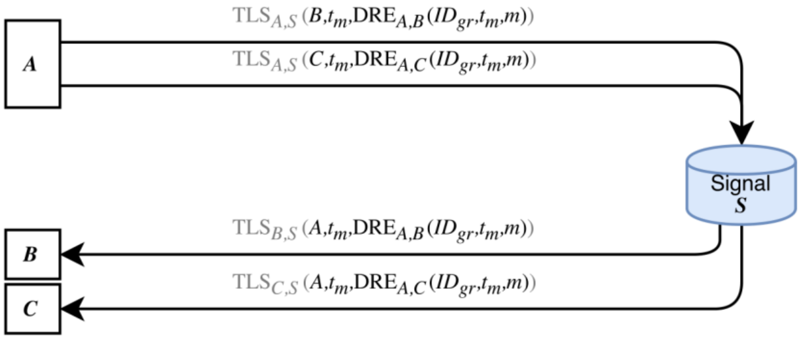
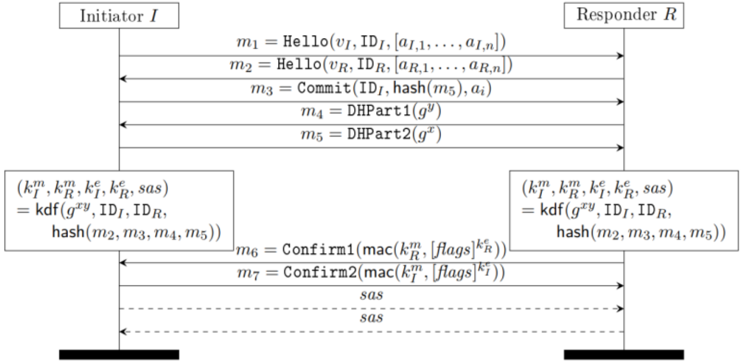

$\mathrm{TLS}_{A,S}(B,t_m,\mathrm{DRE}_{A,B}(ID_{gr},t_m,m))$
$\mathrm{TLS}_{A,S}(C,t_m,\mathrm{DRE}_{A,C}(ID_{gr},t_m,m))$
A
Signal
S
$\mathrm{TLS}_{B,S}(A,t_m,\mathrm{DRE}_{A,B}(ID_{gr},t_m,m))$
$\mathrm{TLS}_{C,S}(A,t_m,\mathrm{DRE}_{A,C}(ID_{gr},t_m,m))$
B
C

Figure 3. Schematic depiction of Signal’s traffic, generated for a message m from sender A to receivers B and C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$. Transport layer protection is not in the analysis scope (gray).

Encrypting voice calls: ZRTP/SRTP.

- ZRTP is sometimes used as the AKE step for an SRTP connection.
- Signal and WhatsApp don’t need to use ZRTP.
- SRTP sets up an encrypted, authenticated stream using a data format that is suitable for voice calls.



Initiator I

Responder R

$m_1 = \texttt{Hello}(v_I, \texttt{ID}_I, [a_{I,1}, \ldots, a_{I,n}])$

$m_2 = \texttt{Hello}(v_R, \texttt{ID}_R, [a_{R,1}, \ldots, a_{R,n}])$

$m_3 = \texttt{Commit}(\texttt{ID}_I, \mathsf{hash}(m_5), a_i)$

$m_4 = \texttt{DHPart1}(g^y)$

$m_5 = \texttt{DHPart2}(g^x)$

$(k_I^m, k_R^m, k_I^e, k_R^e, sas)$
$= \mathsf{kdf}(g^{xy}, \texttt{ID}_I, \texttt{ID}_R,$
$\mathsf{hash}(m_2, m_3, m_4, m_5))$

$(k_I^m, k_R^m, k_I^e, k_R^e, sas)$
$= \mathsf{kdf}(g^{xy}, \texttt{ID}_I, \texttt{ID}_R,$
$\mathsf{hash}(m_2, m_3, m_4, m_5))$

$m_6 = \texttt{Confirm1}(\mathsf{mac}(k_R^m, [\mathit{flags}]^{k_R^e}))$

$m_7 = \texttt{Confirm2}(\mathsf{mac}(k_I^m, [\mathit{flags}]^{k_I^e}))$

sas

sas

Test your knowledge!

Why do secure messengers like Signal and WhatsApp not need a ZRTP handshake?

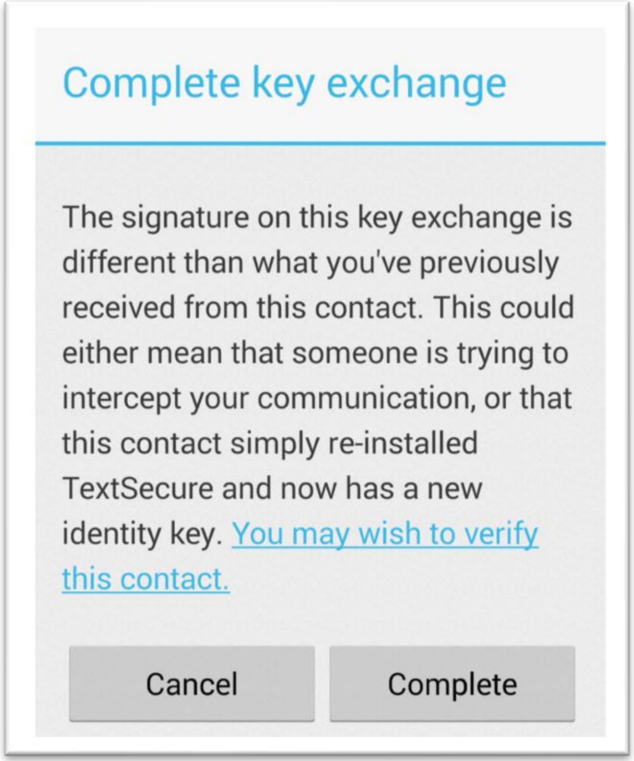
Test your knowledge!

Why do secure messengers like Signal and WhatsApp not need a ZRTP handshake?

Because they can simply send the SRTP shared secret across the existing Signal session.

Usability properties (Unger et al.)

- *Automatic Key Initialization*: No additional user effort is required to create a long-term key pair.
- *Low Key Maintenance*: Key maintenance encompasses recurring effort users have to invest into maintaining keys. Some systems require that users sign other keys or renew expired keys. Usable systems require no key maintenance tasks.
- *Easy Key Discovery*: When new contacts are added, no additional effort is needed to retrieve key material.
- *Easy Key Recovery*: When users lose long-term key material, it is easy to revoke old keys and initialize new keys (e.g., simply reinstalling the app or regenerating keys is sufficient).

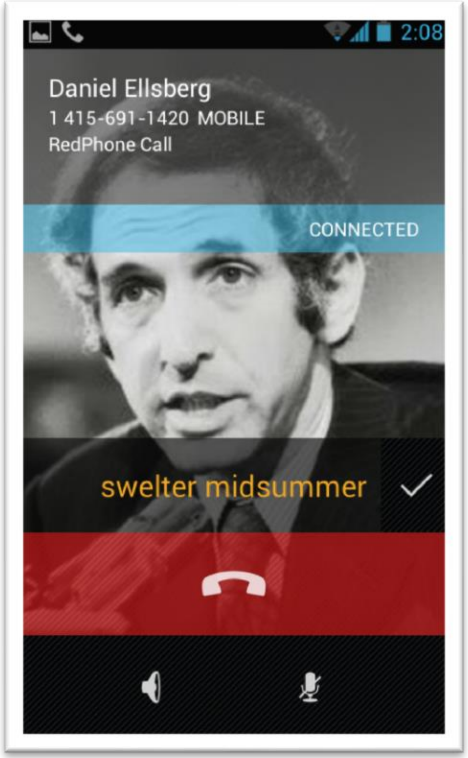


Complete key exchange

The signature on this key exchange is different than what you've previously received from this contact. This could either mean that someone is trying to intercept your communication, or that this contact simply re-installed TextSecure and now has a new identity key. You may wish to verify this contact.

Usability properties (Unger et al.)

- *In-band*: No out-of-band channels are needed that require users to invest additional effort to establish.
- *No Shared Secrets*: Shared secrets require existing social relationships. This limits the usability of a system, as not all communication partners are able to devise shared secrets.
- *Alert-less Key Renewal*: If other participants renew their long-term keys, a user can proceed without errors or warnings.
- *Immediate Enrollment*: When keys are (re-)initialized, other participants are able to verify and use them immediately.
- *Inattentive User Resistant*: Users do not need to carefully inspect information (e.g., key fingerprints) to achieve security.

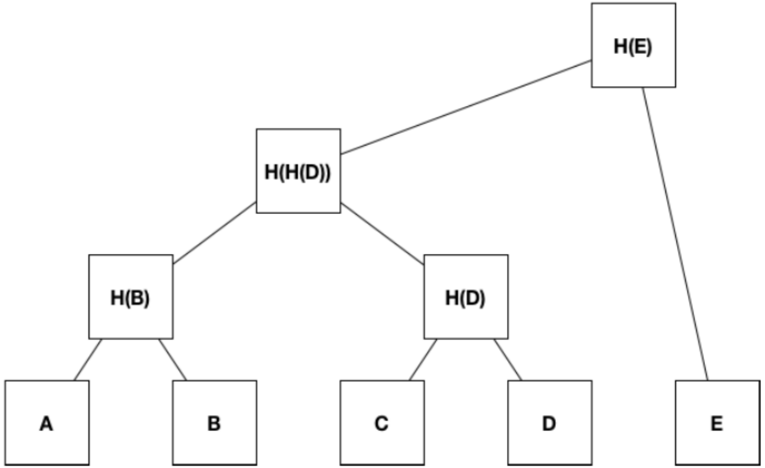


The Future (2017+)

1.5e

Message Layer Security.

- IETF standard, similar to TLS.
- Tries to standardize secure messaging while also providing a solution to scalability problems with group chats.
- Currently being specified with support from major tech players (Google, Facebook, Mozilla, etc.)



Next time: Attacking Cryptographic Systems.

1.6