The devil is in the detail: designing and implementing the 4th version of the Off-the-Record messaging protocol

Sofía Celi

A little bit of context…

Why we need secure communication?

“An especially problematic excision of the political is the marginalization within the cryptographic community of the secure-messaging problem, an instance of which was the problem addressed by Chaum. Secure-messaging is the most fundamental privacy problem in cryptography: **how can parties communicate in such a way that nobody knows who said what**. More than a decade after the problem was introduced, Racko and Simon would comment on the near-absence of attention being paid to the it. Another 20-plus years later, the situation is this: there is now a mountain of work on secure-messaging, but it's unclear what most of it actually does.“

-Rogaway, P. (2015), *The Moral Character of Cryptographic Work*, University of California, Davis, USA

- We need options that work
- We need full specifications
- We need properties, limitations and requirements
- We need protocols that update existing definitions: vague terms get better defined
- We need reviews and verifications
- We need ideas from different places
- We need implementations

What are ‘real-world’ conversations?

- People use the “digital world” for communication

On ‘casual real-world’ conversations, we know:

- who participates in it
- what is said
- who is listening to it
- how long it lasts

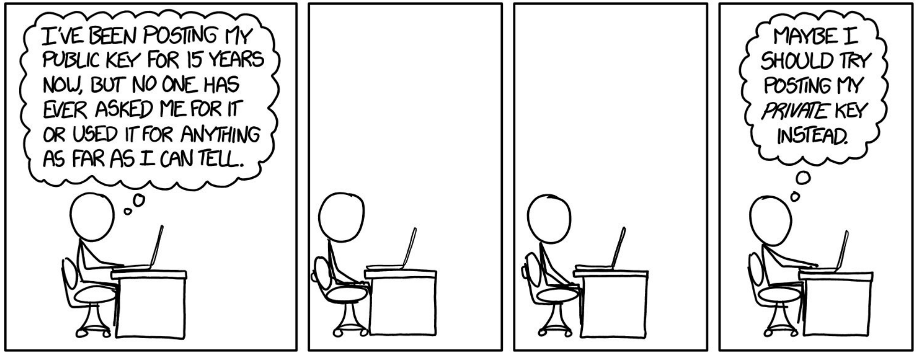
Properties:

- You can deny having participated in it
- You can choose who listens to it
- You can choose how long it will last
- You know something of the identity of whom you communicate with

In the beginning…

Why OTR was created?

- Paper in 2004 by *Ian Goldberg*, *Nikita Borisov* and *Eric Brewer*
- Conversations in the "digital" world should mimic casual real world conversations
- PGP: protect communications. Sign messages and encrypt them.
- Problems: there is a record, there is a ‘proof’ of authorship



https://xkcd.com/1553/

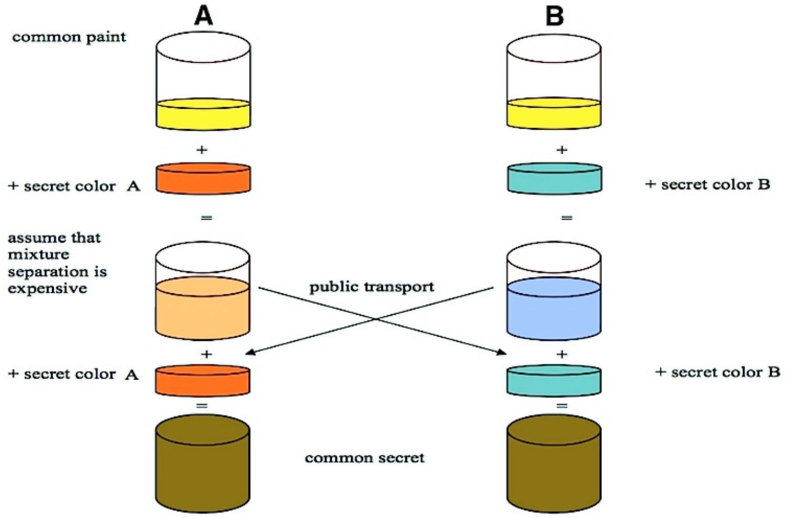
Let’s start with properties

- Forward secrecy:
  - Usage of unique keys for the encryption of each message

  - “The idea of perfect forward secrecy (sometimes called break-backward protection) is that previous traffic is locked securely in the past.” (Menezes, A., Oorschot, P., Vanstone, S. (1997), *Handbook of Applied Cryptography*, CRC Pres.)
  - “A classical adversary that compromises the long-term secret keys of both parties cannot retroactively compromise past session keys” (Bellare, M., Pointcheval, D., & Rogaway, P. (2000). *Authenticated Key Exchange Secure Against Dictionary Attacks*. In Advances in Cryptology–EUROCRYPT)

- Usage of Diffie-Hellman key exchange:
  - Generate *a,* perform DH exchange
  - Use the shared secret *K ((g^b)^a)* to generate *MK*
  - Encrypt messages with *MK*
  - Forget *a* after key exchange; forget *MK* after session

- But there are problems with this…



- Post-compromise security (sometimes referred as backward secrecy):
  - Even if a message key gets compromised, no future messages can be decrypted
  - “A protocol between Alice and Bob provides Post-Compromise Security (PCS) if Alice has a security guarantee about communication with Bob, even if Bob’s secrets have already been compromised” (Cohn-Gordon, K., Cremers, C., & Garrat, L. (2016). *On Post-Compromise Security*. Department of Computer Science, University of Oxford)

Double Ratchet Algorithm

- Happens after an *AKE*

Alice:

- Has a shared secret *K*
- Bob's public key: *bob_dh_pub_0*

Bob:

- Has a shared secret *K*
- Bob's private key: *bob_dh_priv_0*

- Generates:
  - alice_dh_priv_0, alice_dh_pub_0 = generateDH()
- Calculates:
  - shared_secret_1 = DH(alice_dh_priv_0, bob_dh_pub_0)

Alice:

- Derives:
  - RK_0, CKs_0 = KDF(K, shared_secret_1)
- Wants to send message 1 “Hello”
- Derives
  - CKs_1, MK_0 = KDF(CKs_0)
- Encrypts:
  - c_1 = ENC(MK_0, “Hello”)
- Sends: c_1 || alice_dh_pub_0

Bob:

- Calculates:
  - shared_secret_1 = (bob_dh_priv_0, alice_dh_pub_0)
- Derives:
  - RK_0, CKr_0 = KDF(K, shared_secret_1)
- Derives
  - CKr_1, MK_0 = KDF(CKr_0)
- Decrypts
  - “Hello” = DEC(MK_0, c_1)

- If, at that point, Bob wants to send messages, he:

- Generates:
    - bob_dh_priv_1, bob_dh_pub_1 = generateDH()
- Calculates:
    - shared_secret_1 = DH(bob_dh_priv_1, alice_dh_pub_1)

- Double-ratchet algorithm: “Ping-pong” mechanism
- Post-compromise in the sense of giving a timeframe (aka channel healing)
- Alwen, Coretti and Dodis: Immediate Decryption and Message-loss Resilience

Deniability

- Types: online, offline, message, participation
  “We can distinguish between message repudiation, in which Alice denies sending a specific message, and participation repudiation in which Alice denies communicating with Bob at all.”
  - Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M. (2015), *SoK: Secure Messaging*, 2015 IEEE Symposium on Security and Privacy

“A protocol is strongly deniable if transcripts provide **no evidence** even if long-term key material is compromised (offline deniability) and no outsider can obtain evidence even if an insider interactively colludes with them (online deniability).”

- Unger, N. & Goldberg, I. (2015), *Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging*, University of Waterloo, Waterloo, Canada.

Offline and Online Deniability

- Offline Deniability: anyone can forge a transcript using the long-term public keys
  - Achieved by using MAC keys derived from a shared secret and revealing them
  - Achieved by using a DAKE
- Online Deniability: Participants in a OTRv4 exchange cannot provide proof of participation to third parties without making themselves vulnerable to KCI attacks.
  - Achieved by using a DAKE, that uses ring signatures

- Usage of MAC. Every MAC key is “revealed” after been used.
- Usage of DAKEs: usage of ring signatures

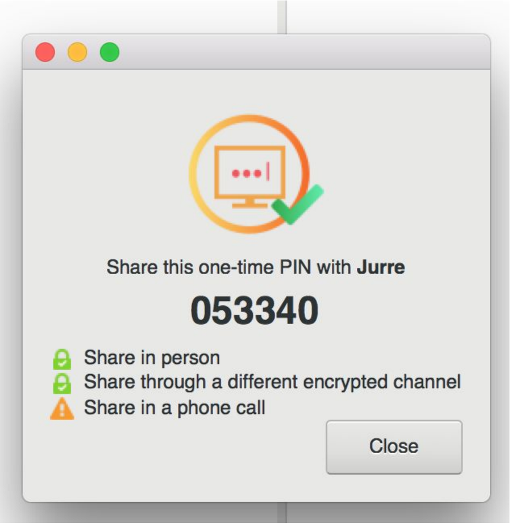
- “Ring signatures are similar to ordinary digital signatures, except that messages are signed by a set of potential signers called a ring. Anyone with knowledge of a private key corresponding to any public key in this ring can produce the ring signature, and it is not possible to determine which key was used”.

- Unger, N. & Goldberg, I. (2015), *Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging*, University of Waterloo, Waterloo, Canada.

- Lachlan J. Gunn, Ricardo Vieitez Parra, and N. Asokan: “Circumventing Cryptographic Deniability with Remote Attestation”
- “Deniability depends upon the ability of an adversary to lie: cryptographic deniability means nothing if a verifier can trust your communications partner to truthfully reveal what you said. Remote attestation allows even manifestly untrustworthy actors such as criminal organizations or hostile intelligence agencies to reach such a level of trustworthiness by piggybacking on a verifier’s trust in a hardware vendor; such an adversary can compromise your partner’s device, and use attestation to prove to a skeptical audience that the messages you sent to that device were not fabricated”

Verification

- Fingerprint verification: key change?
- Socialist Millionaires Protocol: use a shared secret.
  - Alice and Bob learn whether they share the same secret or not
  - They learn nothing else



The state of the art

	OTRv3	OTRv4	Signal	OMEMO	Olm/Megolm	Telegram
Forward secrecy	Weak	Interactive: full Non- interactive: weak	Weak	Weak	None	Weak*
Post-compromise secrecy	Full	Full	Full	Full	Full	Full*
Online Deniability	○	● ◐	○	○	○	○
Offline Deniability	◐	●	◐	◐	●	●

●	provides property
◐	partially provides property
○	does not provide property

- Signal, Wire, Riot, OMEMO, Whatsapp
- MLS

- People moving on from: desktop clients, XMPP
- Too many apps to install
- No clear privacy and security properties given
- No good synchronization between devices
- No mapping of security/privacy properties into the UI
- Deniability in the UI?

Version 4

Why a version 4 of OTR?

- We want deniability: participation, message, online and offline
- We want forward secrecy and post-compromise secrecy
- We want a higher security level
- We want to update the cryptographic primitives
- We want additional protection against transcript decryption in the case of ECC compromise
- We want elliptic curves

New communication model

- We want in-order and out-of-order delivery of messages
- We want online and offline conversations
- We want different modes in which something can be implemented
- We don’t want to trust servers

- Do we need new versions?

Limitations and current issues

- Metadata protection
- Post-quantum algorithms
- Group chat support

Things to discuss:

- What about the synchronization and multi-device problem?
- Should messages disappear / no history?
- Impact of ‘top’ properties on the underlying protocol
- Can there be modes for deniability?
- Do we need new protocols or to update the existing ones?
- Do we need more apps?

Implementation problems

- Which language do we choose?
- Which library we choose?
- How do we correctly store/delete/change keys?
- How do we manage keys?
- Too many languages: problems with cryptographic libraries
- Should serves be trusted?
- Is the code audited? Is the protocol verified?
- How do the UI will look like?

Thanks to everyone involved

To the main collaborators (people in the current team or with more than 6000 lines of code/text contributed):

- Ian Goldberg
- Nik Unger
- Mike Hamburg
- Sofia Celi
- Reinaldo de Souza Jr
- Rosalie Tolentino
- Jurre van Bergen
- Iván Pazmiño
- Giovane Liberato
- Fan Jiang
- Mauro Velasco
- Pedro Palau
- Cristina Salcedo
- Others who have collaborated

Check out our repos!

The protocols:

https://github.com/otrv4/otrv4

https://github.com/otrv4/otrv4-prekey-server

The library:

https://github.com/otrv4/libotr-ng

The plugin:

https://github.com/otrv4/pidgin-otrng

The prekey server:

https://github.com/otrv4/otrng-prekey-server

https://github.com/otrv4/prekey-server-xmpp

The toolkit:

https://github.com/otrv4/libotr-ng-toolkit

Golang

https://github.com/otrv4/otr4

Java by Danny van Heumen

https://gitlab.com/cobratbq/otr4j

OTR.im

- Happy to host you and setup CI/CD

Time for references

1. Goldberg, I. and Unger, N. (2016). Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging, Waterloo, Canada: University of Waterloo. Available at: http://cacr.uwaterloo.ca/techreports/2016/cacr2016-06.pdf
2. Hamburg, M. (2015). Ed448-Goldilocks, a new elliptic curve, NIST ECC workshop. Available at: https://eprint.iacr.org/2015/625.pdf
3. Gunn, L. J., Vieitez Parra, R. and Asokan, N. (2018) On The Use of Remote Attestation to Break and Repair Deniability. Available at: https://eprint.iacr.org/2018/424.pdf

4. Rogaway, P. (2015), *The Moral Character of Cryptographic Work*, University of California, Davis, USA

5. Menezes, A., Oorschot, P., Vanstone, S. (1997), *Handbook of Applied Cryptography*, CRC Pres.)

6. Bellare, M., Pointcheval, D., & Rogaway, P. (2000). *Authenticated Key Exchange Secure Against Dictionary Attacks*. In Advances in Cryptology–EUROCRYPT

7. Cohn-Gordon, K., Cremers, C., & Garrat, L. (2016). *On Post-Compromise Security*. Department of Computer Science, University of Oxford

8. Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M. (2015), *SoK: Secure Messaging*, 2015 IEEE Symposium on Security and Privacy

Questions?

- Come us find us online, as well! (https://otr.im/)
- IRC: #otr at OFTC
- We have an assembly!

Thanks!

Sofía Celi
@claucece

OFF-THE-RECORD MESSAGING

@otr_im

You have unlocked the secret slides*

*Copyright to Nik Unger

Difference with Signal

- OTRv4 has better deniability properties and perfect forward secrecy
- OTRv4 has a well defined specification
- OTRv4 has different verification mechanisms
- OTRv4 supports different networks and is not centralized
- OTRv4 supports other features, such as symmetric keys

Difference with OMEMO

- OTRv4 is agnostic: can work over any protocol, even asynchronous
- OTRv4 has better deniability properties
- OTRv4 has a well defined specification
- OMEMO supports transcript synchronizing between devices

Difference with MLS

- OTRv4 is not for groups; MLS is
- OTRv4 has better deniability properties for a one-to-one conversation

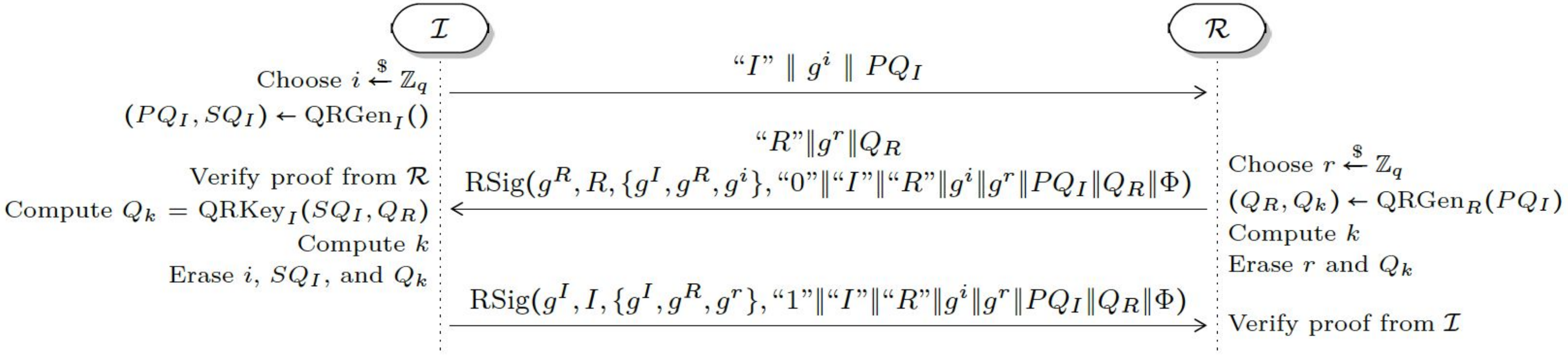
Why deniability matters

- It is a right in casual real-world conversations, even if you don’t think about it
- It is useful not only to you but to whom you are talking to
- It is resistance
- We shouldn’t make the situation worse than plaintext, by adding irrefutable proof of conversations

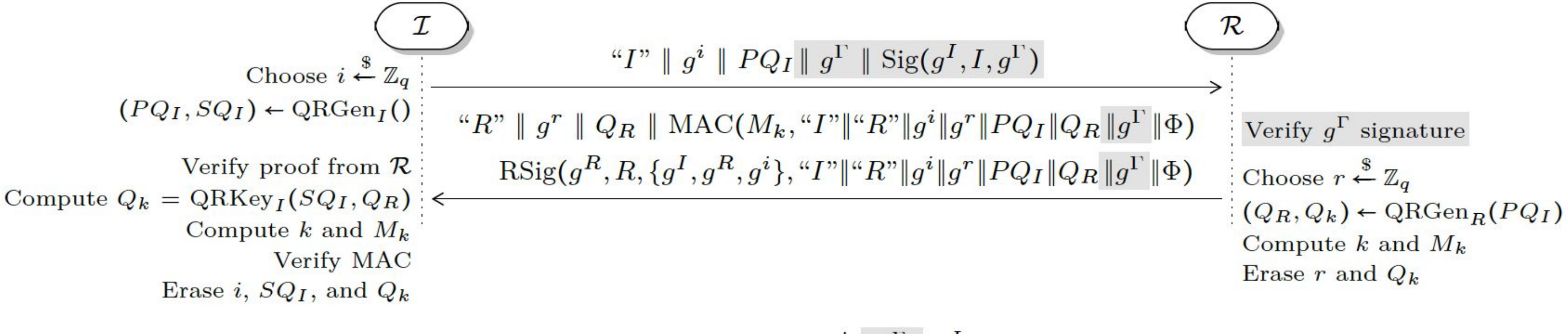
What is weak forward secrecy?

- Strong forward secrecy: protects the session key when at least one party completes the DAKE exchange
- Weak forward secrecy: protects the session key only when both parties complete the DAKE exchange

The DAKEs



DAKEZ -Unger, N. & Goldberg, I. (2015), *Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging*, University of Waterloo, Waterloo, Canada



XZDH -Unger, N. & Goldberg, I. (2015), *Improved Strongly Deniable Authenticated Key Exchanges for Secure Messaging*, University of Waterloo, Waterloo, Canada