SECURE MESSAGING: MODERN THEORY AND PRACTICE

Nadim Kobeissi, INRIA, CNAM
Technische Universität Harburg/Hamburg
December 7, 2015

About me

- PhD student at INRIA Paris and CNAM, team PROSECCO (**Pro**gramming **Sec**urely with **C**rypt**o**graphy) and CEDRIC.
- Worked on many open source projects projects: Cryptocat, miniLock, Peerio…
- Interested in secure messaging, verifiable protocols.

Today's discussion

- Real-world examples and applications.
- Differences in expectations (versus email, etc.)
- Protocol designs and flaws (OTR, Axolotl.)
- Future of secure messaging.

Who needs secure messaging?

- More and more, we talk with friends and colleagues through short messages rather than email:
  - *Facebook Messenger,*
  - *WhatsApp,*
  - *iMessage…*
- Over time, the **aggregated data** in these conversations becomes quite large.
  - *Go check out your Facebook Messenger chat history!*

Email has PGP…

- Extremely tedious encryption system for email:
  - *Manual key exchange and verification.*
  - *Manual identity management.*
  - *Manual plugin installation into desktop mail client.*
  - *Keys difficult to transfer and manage across devices.*
  - *Difficult to set up and use for regular computer users.*

Secure messaging in 2004

- First serious effort for secure messaging was **OTR**, a plugin for Gaim (now Pidgin).
- Integrates secure messaging into desktop client for AIM, Yahoo Chat, Google Talk, etc.

✅ *Innovative protocol.*

✅ *Improvements in usability.*

☹️ *Still requires external plugin.*

☹️ *Limited to desktop use-case.*

Secure messaging in 2012

- 2012: Cryptocat (my project as an undergrad) attempted to bring secure messaging to the web as a browser app.
- Logic: we need to make secure messaging **accessible** if we want it to become **ubiquitous**.
- Result: many bugs at first (unprecedented use of browser crypto), today has long-lasting effects on secure messaging design and web cryptography.

Secure messaging today

- On mobile, many applications: Signal, ChatSecure, Threema, Silent Circle, Telegram…
- Mobile requires improvements in **usability** and to address the **asynchronous use-case**:
  - *Users need to have a frictionless session setup experience (like WhatsApp or iMessage),*
  - *Users need to be able to message each other while the other party is offline…*

Secure messaging today

- Even commercial applications are implementing end-to-end encryption **(only against passive attacker):**
  - *iMessage, but without the ability for key verification, keys managed by Apple.*
  - *WhatsApp, but without the ability for key management, only between Android devices, needs to be approved by WhatsApp on server-side.*
- Signal (previously known as **TextSecure**) is worth investigating due to their security guarantees and open source engineering.

OTR: A closer look

- Opportunistic encrypted Instant Messaging protocol.
- Works as a layer over XMPP, other transports…
- Prioritizes **forward secrecy**, as opposed to PGP.
- Another interesting property: **plausible deniability**.

OTR: A closer look

- Long-term signing (identity) keys. Ephemeral encryption + MAC keys.
- Authenticated key exchange based on SIGMA (SIGn-and-MAC).
- Message exchange in which old MAC keys are published and new encryption + MAC keys are derived immediately after every message.
- SMP for authentication.

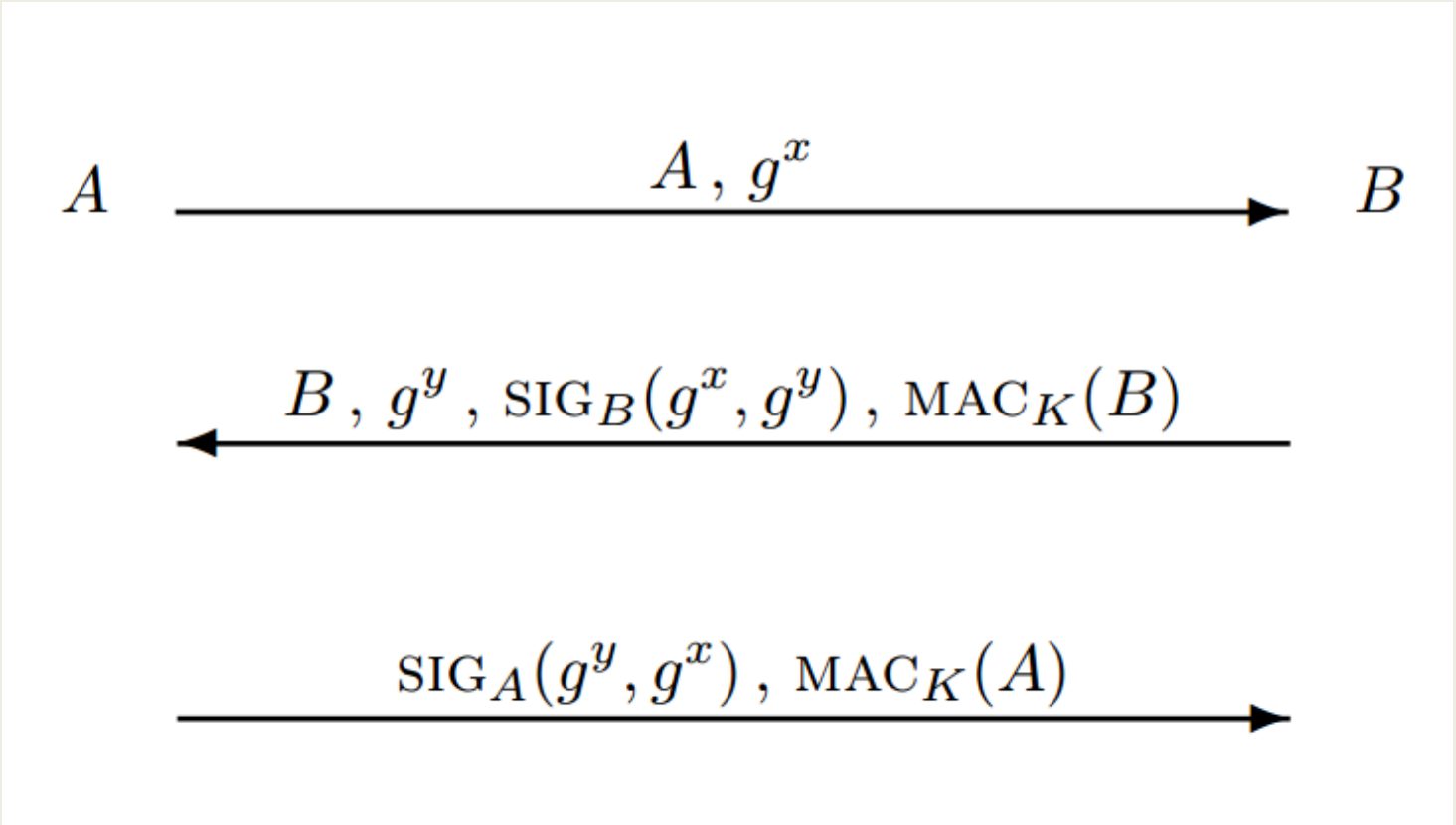
OTR: Primitives

- AES-CTR for encryption.
- Diffie-Hellman (with 1536-bit fixed Prime) for public key agreement.
- DSA for digital signatures.
- SHA({1,256})-HMAC for message authentication.

OTR: Guarantees

- Formally, OTR wants to guarantee, against an **active adversary**:
  - *Secrecy:* if A sends a message M to B, nobody can obtain the message except A and B.
  - *Authenticity:* if B receives a message M from A, then A sent message M to B.
  - *Forward Secrecy:* if A sends a message M to B and her long-term keys are later compromised, M remains secret.

OTR: SIGMA AKE



$K = KDF(g^{xy})$

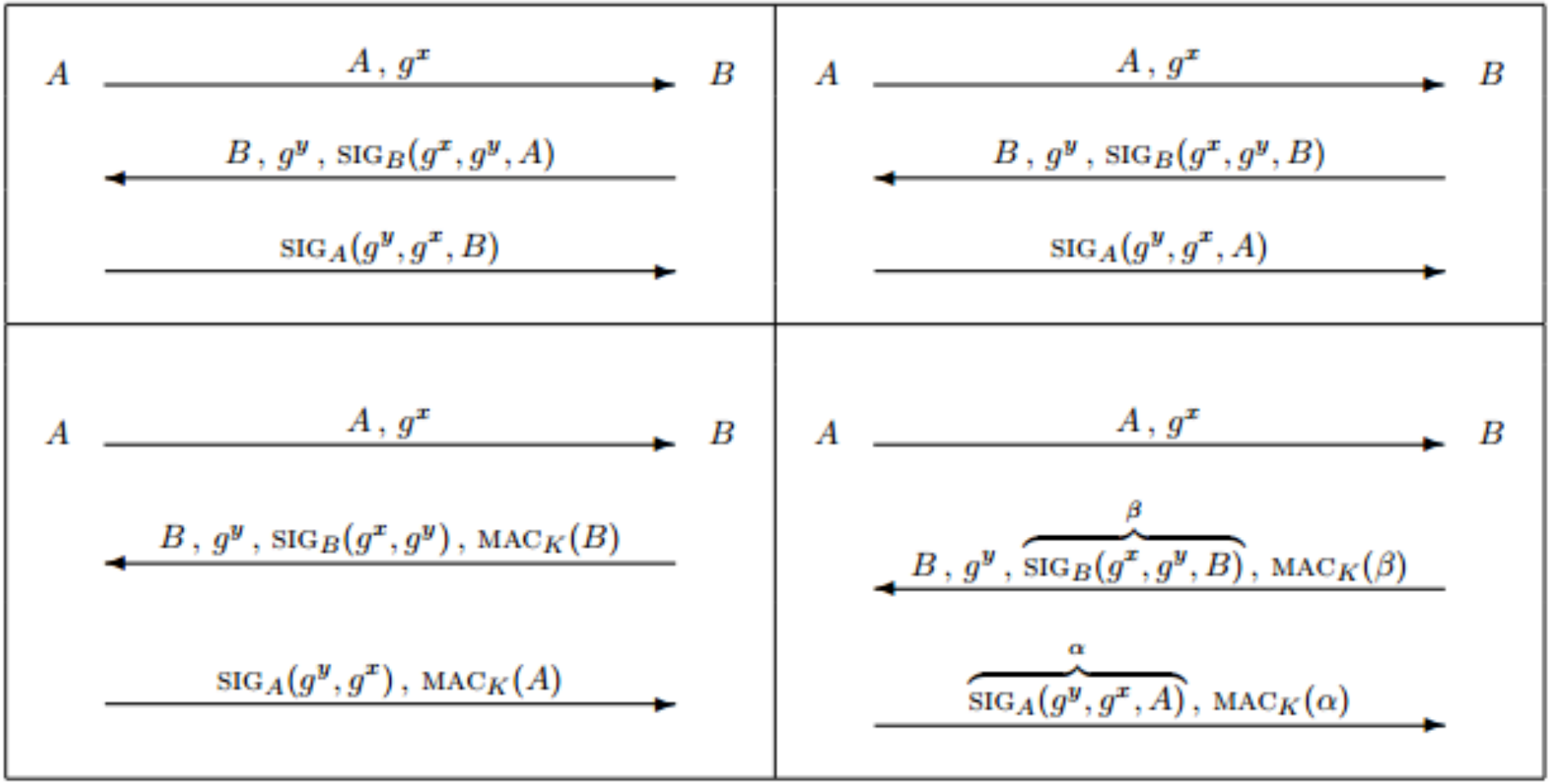
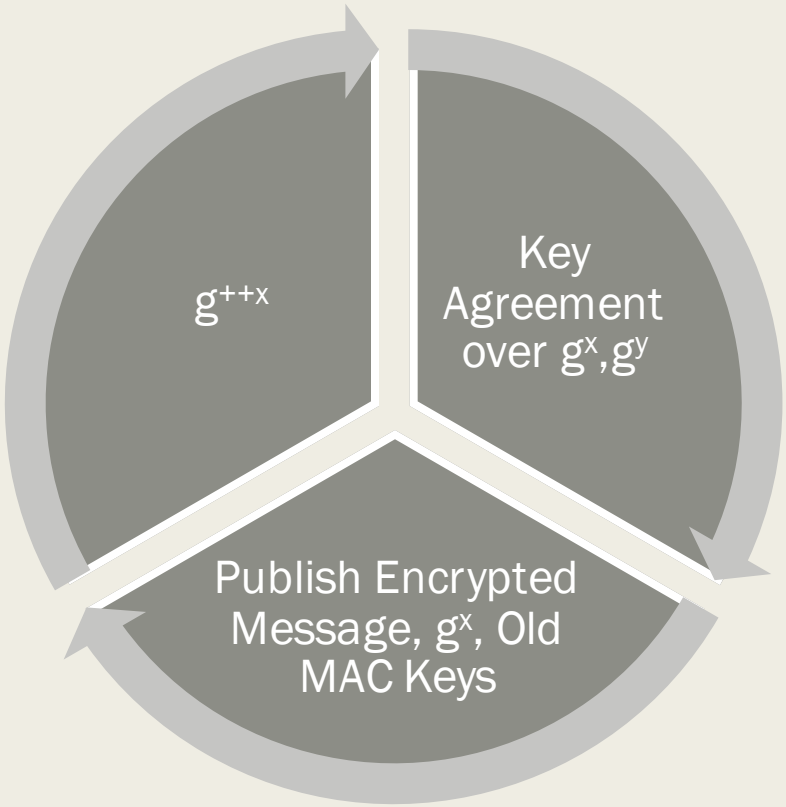


Figure 1: Test your intuition: which of these four authenticated Diffie-Hellman exchanges constitute secure key-exchange protocols (without identity protection)? Answers are provided within the paper. The notation SIG_X represents a signature by participant X; MAC_K represents a message authentication function computed using a key K derived from the Diffie-Hellman key g^{xy}. The output session key in all cases is derived from g^{xy} independently of K.

OTR: Forward-secure ratchet



OTR: Authentication

- Alice has **two** ways to authenticate the long-term key of Bob:
  - *Key fingerprint:* Bob reads a hash of his long-term public key and Alice checks if she has the same hash on her screen.
  - *SMP:* Zero-knowledge protocol that allows Alice to ask Bob a secret question and determine if he has given the right response.

OTR: “Plausible Deniability”

- If a judge J obtains a copy of an OTR transcript between A and B, they cannot be *“convinced”* that A and B necessarily produced that transcript.
- Accomplished by publishing MAC keys of previous messages in next messages.
- Is it a convincing property?

OTR: Weaknesses

(Bonneau et al.)

- **Version Rollback**: Attacker can force obsolete OTR protocol version.
  - *Fix by integrating version number into AKE.*
- **Authentication Failure**: Mallory can convince Alice of a successful AKE with Bob, even if Bob has no idea what is going on.
  - *Only applies to convincing of successful AKE. No MITM, no other consequences.*
- **Message Integrity Attack**: Actually interesting!

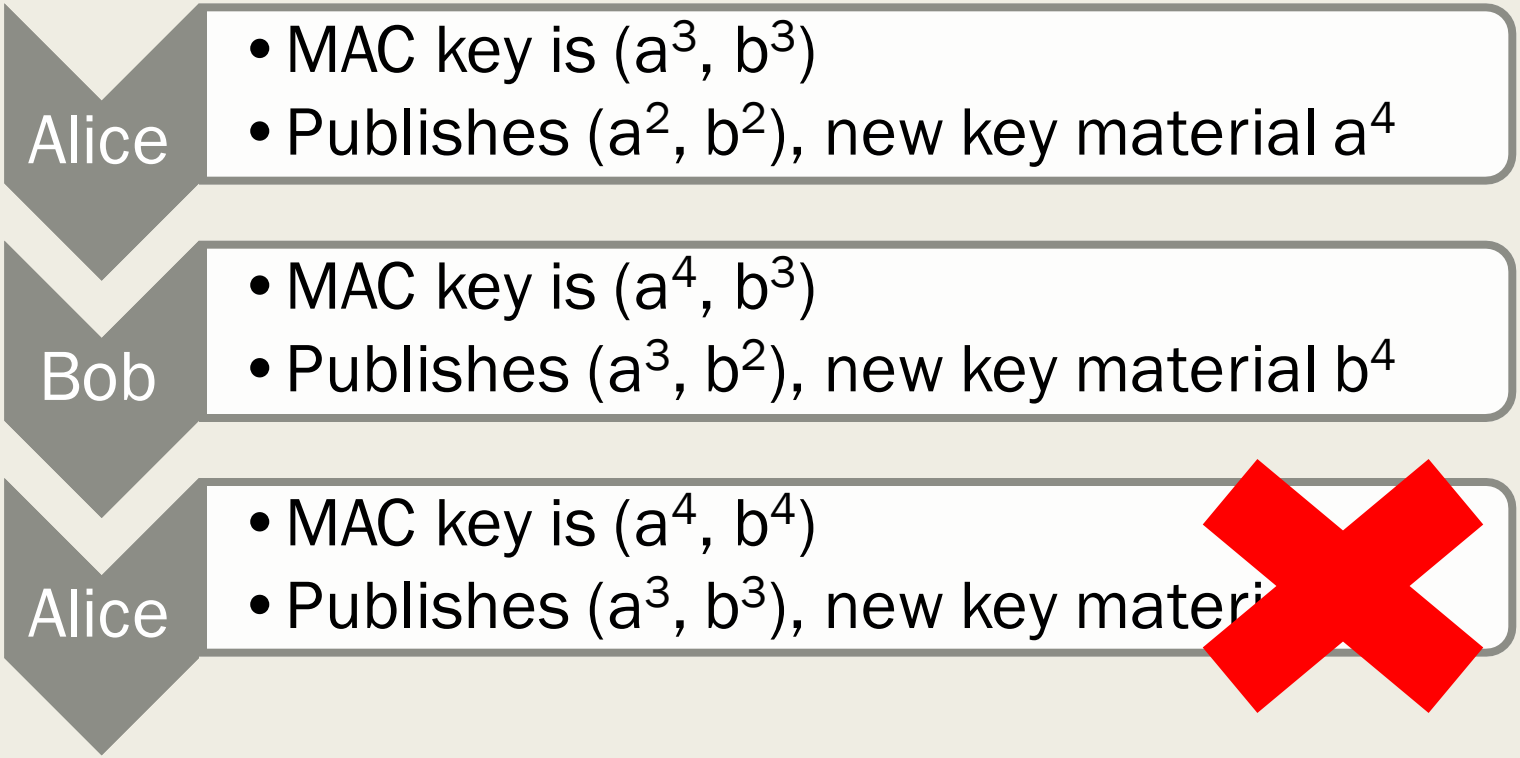
OTR: Message Integrity Attack

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

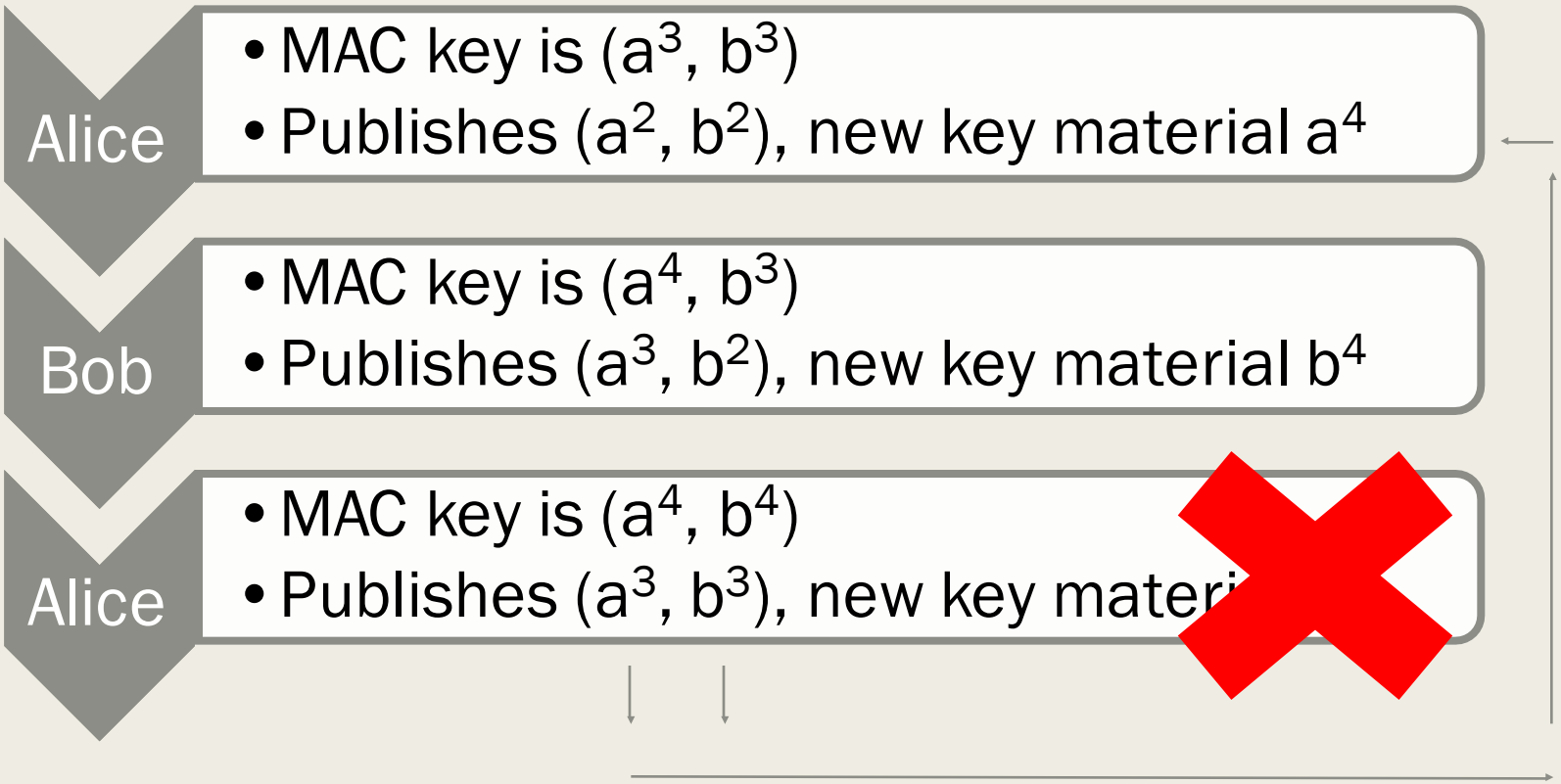
Bob
- MAC key is (a^4, b^3)
- Publishes (a^3, b^2), new key material b^4

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

OTR: Message Integrity Attack



OTR: Message Integrity Attack



TextSecure: A closer look

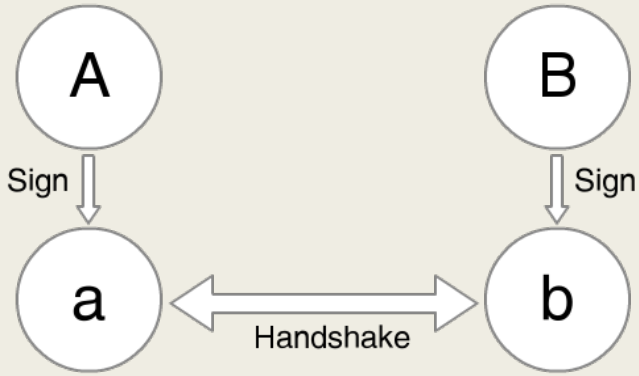
- Signal wants to have a **WhatsApp-like user experience**:
  - *Message users when they are offline.*
  - *Automatic session setup.*
  - *Semi-automatic key authentication.*
- As such, we need something better than SIGMA.

TextSecure: Security properties

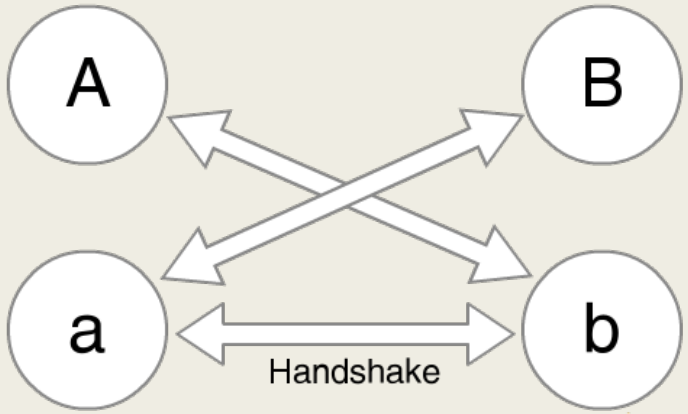
- TextSecure also guarantees secrecy, authenticity, forward secrecy, but also:
  - ***Future secrecy:*** *If A sends message M to B, receives message N and then sends M', the compromise of M and N does not compromise M'.*

TextSecure: Triple Diffie-Hellman

SIGMA-like AKE



Axolotl AKE

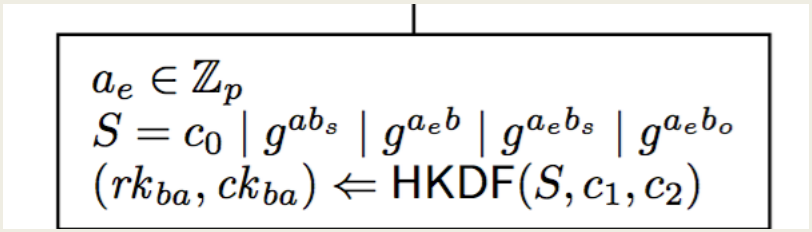


Goals:

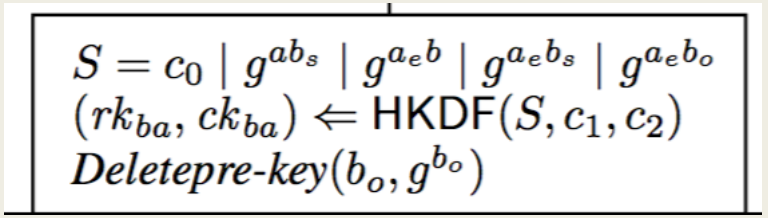
- Reduce number of rounds.
- Reduce dependence on signing primitive.

TextSecure: Triple Diffie-Hellman

- Loosely inspired by Microsoft Research's NAXOS AKE.
- One-round AKE, with ephemeral key shares pre-generated, signed and stored on the server:
  - *Signed pre-key*: Stored on the server and can be re-used,
  - *One-time pre-key:* Stored on the server, not signed, used only once for "freshness" of session AKE.



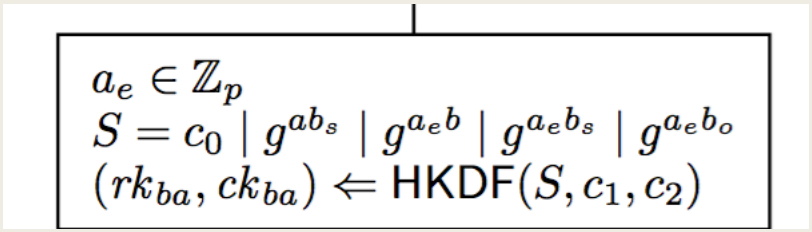
Alice



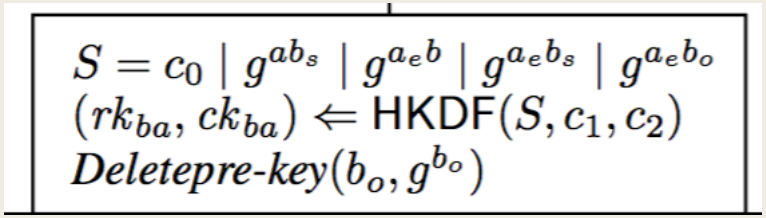
Bob

TextSecure: Triple Diffie-Hellman

- One-round AKE, with ephemeral key shares pre-generated, signed and stored on the server:
  - *Signed pre-key*: Stored on the server and can be re-used,
  - *One-time pre-key:* Stored on the server, not signed, used only once for "freshness" of session AKE (introduced in Version 3)
- What happens if the **pre-key isn't signed**? Or if there is no one-time prekey?



Alice



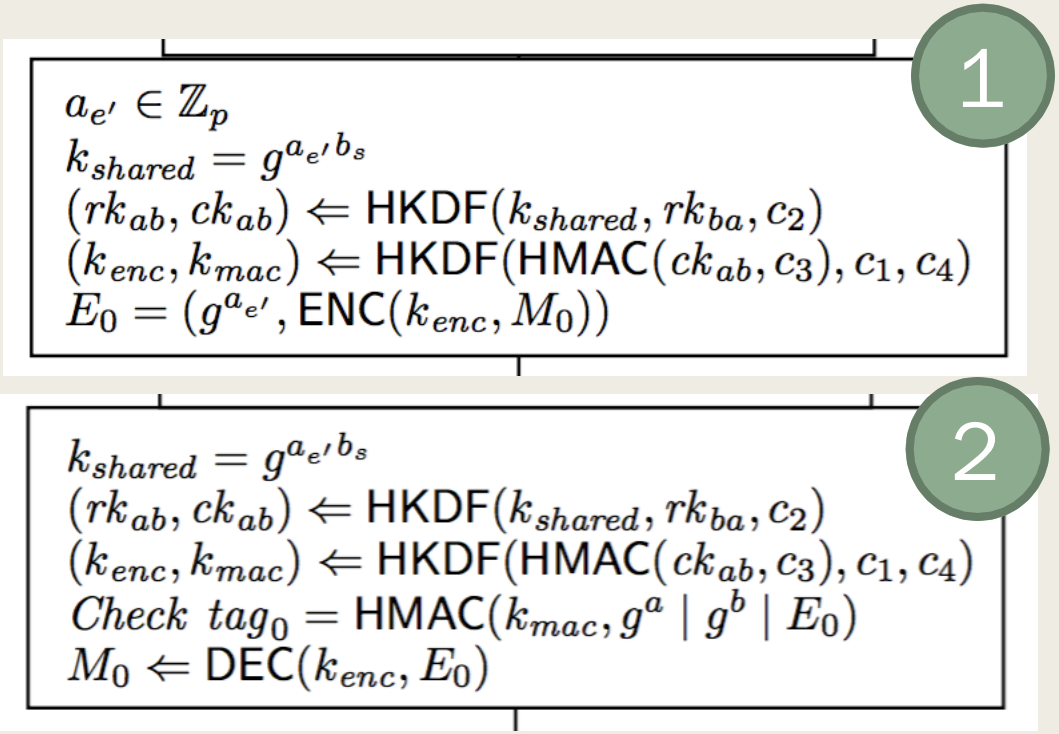
Bob

TextSecure: Axolotl ratchet

- **Stronger forward secrecy**: No need to wait for reply message to initiate new message keys.
- **Future secrecy**: Message keys are automatically refreshed for each message.

TextSecure: Axolotl ratchet

- From your handout:

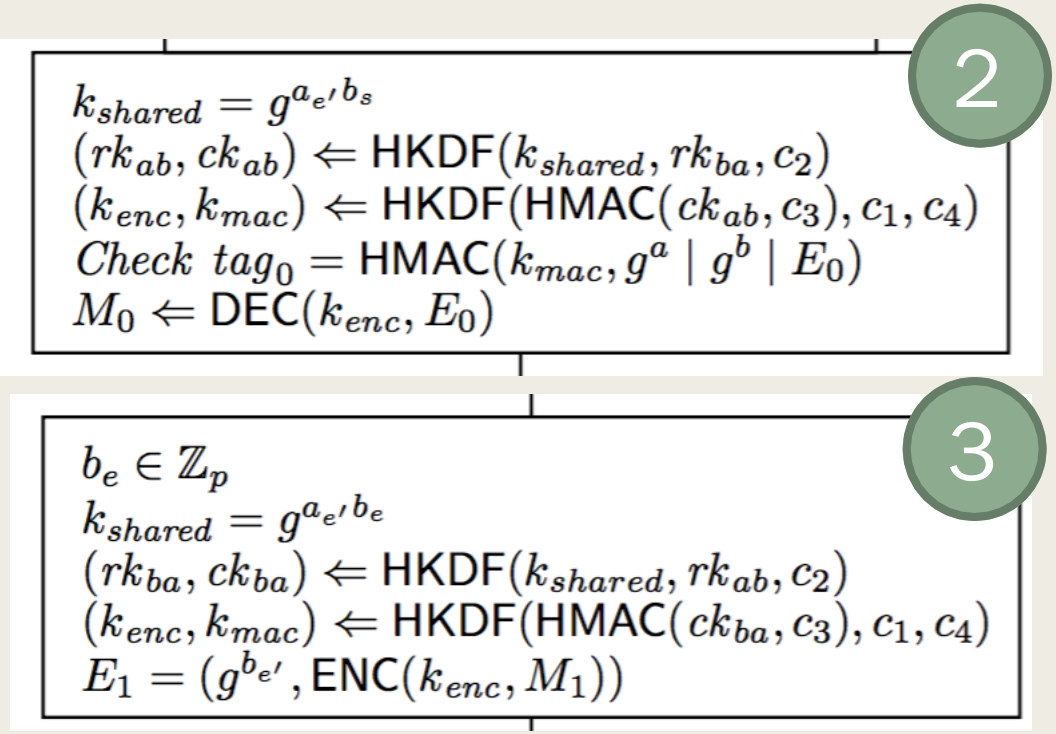


Alice sends a message.

Bob receives a message.

TextSecure: Axolotl ratchet

- From your handout:

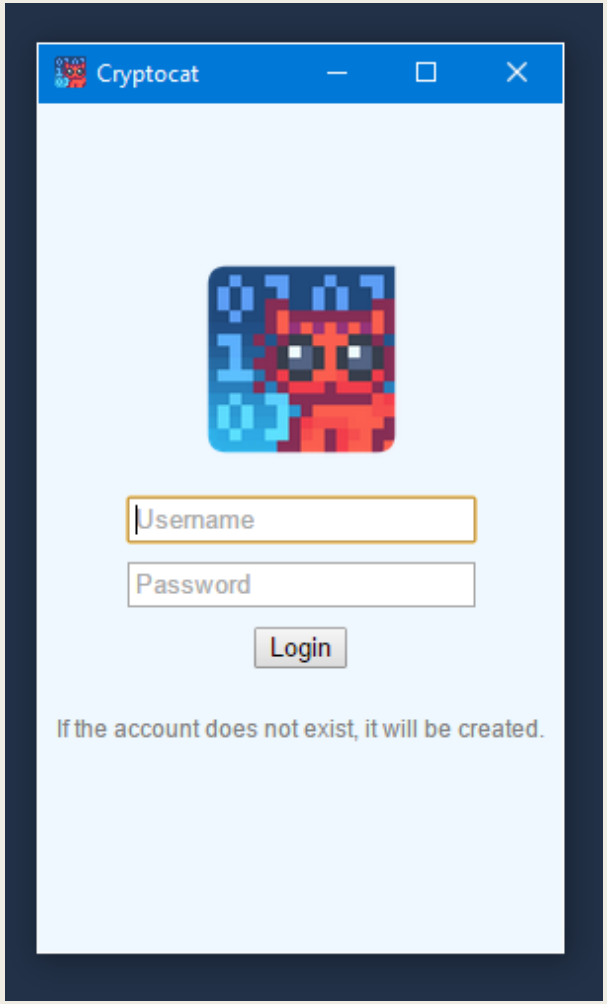


Bob receives a message.

Bob sends a reply.

Future of secure messaging

- Signal ChatSecure, etc. are pushing forward open source secure messaging on mobile. Victory!
- My next project: rewrite Cryptocat to replace Pidgin as a **desktop client** with built-in encryption, easy account creation… the synchronous use-case is still relevant.



I hope you learned something!

- Thanks!
- Our team website: http://prosecco.gforge.inria.fr
- My website: https://nadim.computer