Lecture 11: Secure Messaging

CPSC 4440/5440: Real World Cryptography

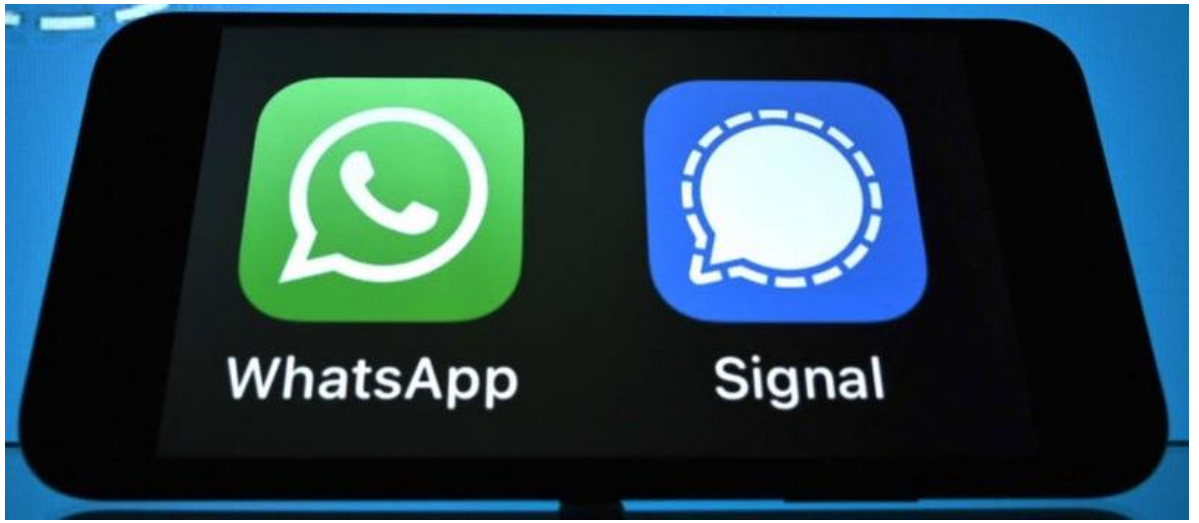
Instructor: Prof. Fan Zhang

Yale University

Spring 2026

Yale

Computer Science



- Have you used any of these?
- Do you know how they work?

The Problem of Secure Messaging

- So far: Metadata hiding against **traffic analysis**
- Today we are going to tackle another threat: **device compromise**.

- Let's make generous assumptions
  - They know **how to use public key cryptography**
  - They know each other's **public keys**
  - They don't want to hide metadata, **just content**

Doesn't PKC solve the problem?

Alice

sk_A, pk_A

$\text{Enc}(\text{pk}_B, m), \text{Sig}_{\text{Alice}}(m)$

Bob

sk_B, pk_B

Seems *Pretty Good Privacy* for Alice.

Doesn’t PKC solve the problem?

Pretty Good Privacy

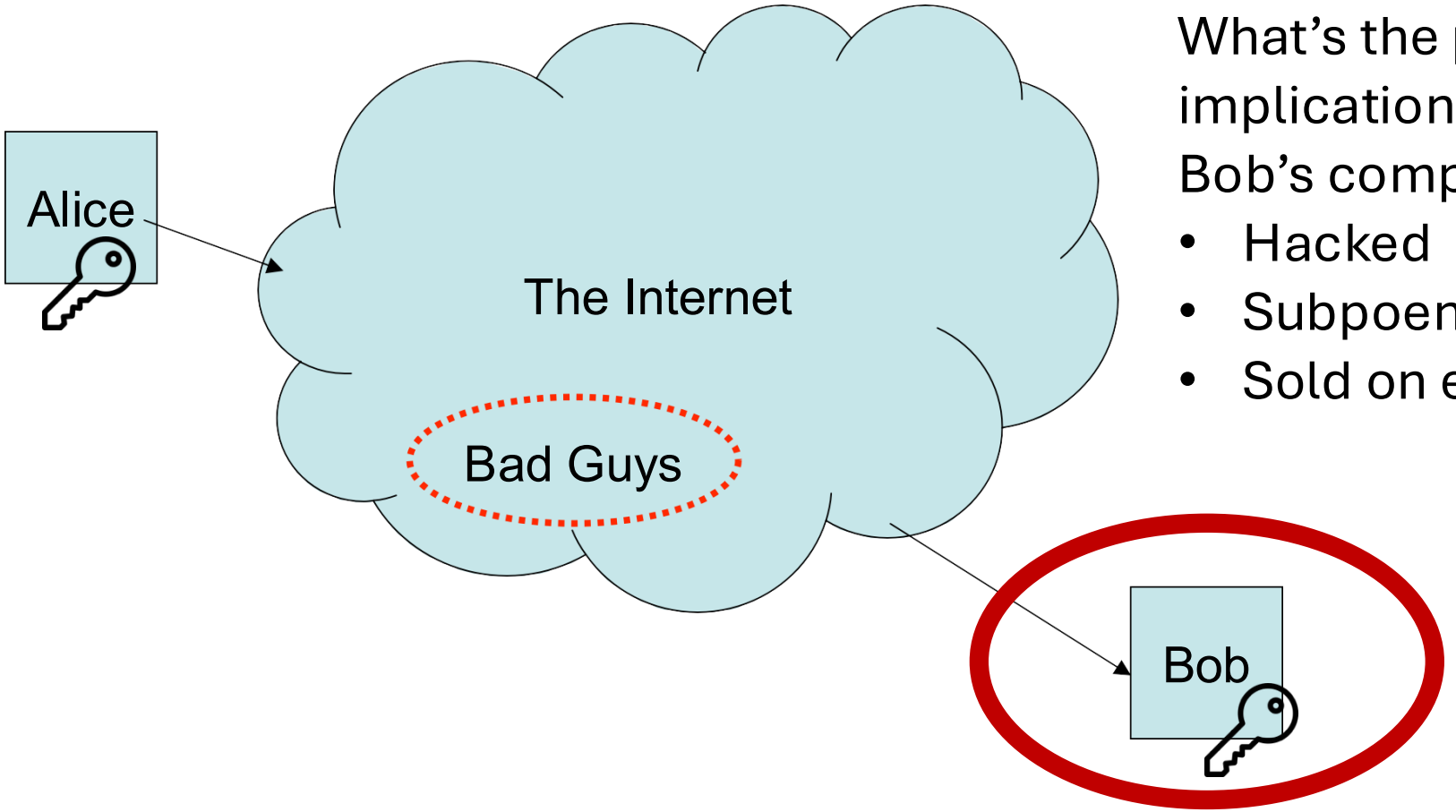
Original author(s)	Phil Zimmermann PGP Inc. Network Associates PGP Corp.[1]
Developer(s)	Broadcom Inc.
Initial release	1991; 34 years ago
Stable release	11.4.0 Maintenance Pack 2 / May 23, 2023; 21 months ago[2]

OpenPGP

That’s basically how *Pretty Good Privacy* (PGP) works for email encryption.

Why not good enough?

Threat Model



What's the privacy implications for **Alice** if Bob's computer is

- Hacked
- Subpoenaed
- Sold on eBay?

Bad guys who got the keys can…

- Decrypt past messages
- Learn their content
- Learn that Alice sent them
- Obtain a proof about what Alice said.
  - Alice's signatures
- Alice better watches what she says to Bob
  - Her privacy depends on Bob's actions!

What went wrong?

- Bob lets his computer get stolen?
- Have you…
  - left your laptop unattended?
  - delayed installation of security patches?
  - run other’s software with sudo?
- What about grandpa?

What Really Went Wrong

- The threat model does not anticipate breaches
- Symptoms in PGP
  - ⇒ creates incriminating records of what a party has said
  - ⇒ key breach leaks *past* messages

- What are ideal privacy properties for messaging?

Ideal: Casual Conversations

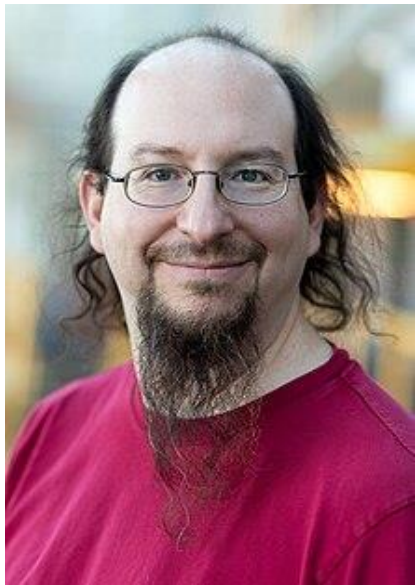
- Alice and Bob talk in a room
- No one else knows what they say
  - Unless Alice or Bob tell them
- No one can prove what was said
  - Not even Alice or Bob
  - Illegal to record conversations without notification
- These conversations are “off-the-record”
- Can we have off-the-record messaging on the Internet?

Off-the-Record Communication, or, Why Not To Use PGP

Nikita Borisov
UC Berkeley
nikitab@cs.berkeley.edu

Ian Goldberg
Zero-Knowledge Systems
ian@cypherpunks.ca

Eric Brewer
UC Berkeley
brewer@cs.berkeley.edu



Crypto Tools

- What we need is
  - **Perfect forward secrecy (PFS)**
  - **Deniability**

Perfect Forward Secrecy

- We have seen PFS in AKEs: If a server is compromised in the future, *past* communication remain confidential.
- General idea:
  - Use short-lived **encryption keys**
  - **Securely erase** retired keys from memory
  - Use long-term **signature keys** to help distribute & authenticate the short-lived encryption key

Deniability [New]

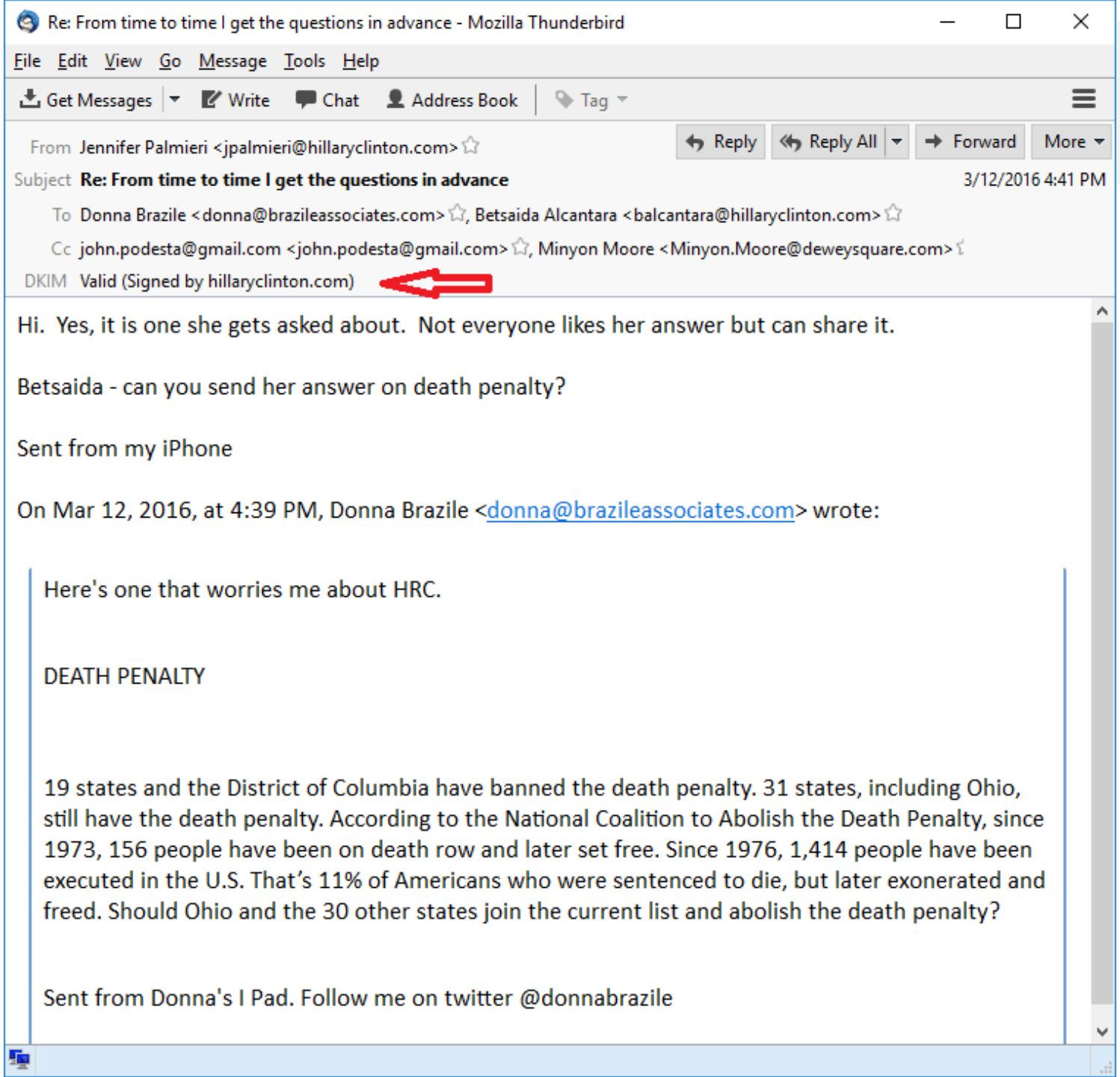
- There are *many* flavors of deniability.
- E.g., deniability against an eavesdropper:
  - Eve intercepts a communication transcript T between Alice and Bob
  - Eve presents T to a judge
  - Alice and Bob can deny their association with T, by saying “Eve created the T all by herself”
- Judge only considers *verifiable evidence*.
- Example of non-deniable protocol: PGP

Deniability [New]

- Stronger definition: Bob is malicious.
  - Alice said X to Bob.
  - Bob records the transcript T.
  - Bob presents T to a judge, trying to prove “Alice said X” or even “Alice spoke with me”
  - Deniability for Alice: “Bob could have generated T all by himself.”
- Judge can be **online** or **offline**
  - Online: judge interacts with Bob while Alice and Bob talk
  - Offline: judge appears *after* the conversation

Deniability probably have changed history

- In October 2016, the personal emails of campaign chairman were leaked. These revealed inner campaign strategy and excerpts from Clinton's private speeches to Wall Street firms.



changed history

- Emails are digitally signed by providers to prevent spams
- Called **Domain Keys Identified Mail (DKIM)**
- DKIM signatures undermined deniability by democrats.

Deniability probably have changed history

- 2016: DKIM signatures proved authenticity of some (but not all) emails from Hillary Clinton breach
- 2020: Republican released one email (with DKIM signature) claimed to have been from Hunter Biden's laptop.

- If emails were *off-the-record*, history might look different… Not hard to implement.

Off-the-Record Protocol

- **Security goals**: Confidentiality, Authentication, Perfect Forward secrecy, Deniability
- Assume Alice and Bob know each other's public keys
  - We of course need to deal with that later

Off-the-Record Communication, or, Why Not To Use PGP

Nikita Borisov
UC Berkeley
nikitab@cs.berkeley.edu

Ian Goldberg
Zero-Knowledge Systems
ian@cypherpunks.ca

Eric Brewer
UC Berkeley
brewer@cs.berkeley.edu

Step 1: Deniable AKE

- OTR original paper proposed *insecure* Diffie-Hellman KE
- Alice and Bob pick random x, y resp.
- A → B: g^x, $Sign_{Alice}(g^x)$
- B → A: g^y, $Sign_{Bob}(g^y)$
- Both: use $SS=g^{xy}$ as the shared secret

- Do you spot the security bug?
- **Deniable**: Bob can't prove to a judge that Alice talked with him.

Deniable Authentication

- Need to authenticate msg authorship
- Do not want to sign messages (non-repudiation)
- What do use?

- Answer: Message Authentication Codes (MACs)

MAC allows Deniability

- Alice and Bob have the same mac key
- Bob cannot prove that Alice generated the MAC
  - He could have done it, too
  - Anyone who can verify can also forge

- Encryption also must be perfect malleable (e.g., AES-CTR mode)

Step 2: Message Transmission

- Both derive Encryption Key (`EK`) and MAC Key (`MK`)
  - `EK, MK ⇐ KDF(SS)`
- A ⇦⇨ B: (Enc_{EK}(M), MAC(Enc_{EK}(M), `MK`))

Step 3: Re-key

- How about PFS?
- To get PFS, Alice and Bob perform a new AKE after every message.

Step 3: Re-key

Can **piggyback DHKE on regular messages** (MACs are omitted)

Initial KE

$$
\begin{aligned}
A \to B: &\quad g^{x_1} \\
B \to A: &\quad g^{y_1} \\
A \to B: &\quad g^{x_2}, E(M_1, k_{11}) \\
B \to A: &\quad g^{y_2}, E(M_2, k_{21}) \\
A \to B: &\quad g^{x_3}, E(M_3, k_{22})
\end{aligned}
$$

Alice: write to me using a new key

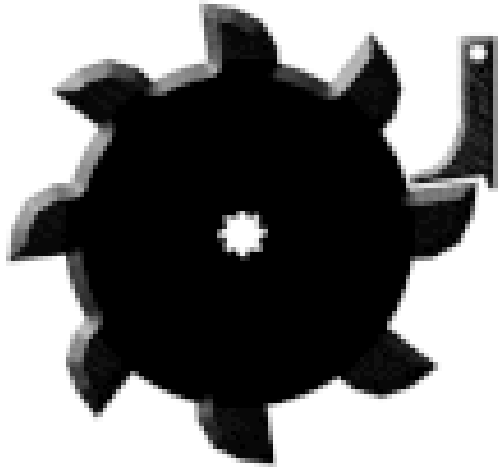
Bob derives $k_{21} = \mathrm{DH}(g^{x_2}, g^{y_1})$

and so on…

where $k_{ij} = H(g^{x_i y_j})$, the result of a 128-bit hash function

- Alice securely erases x_1 after Bob started to use k_{2*}
  - Practical consideration: messages may arrive out-of-order
- If Alice is compromised after that, still can't decrypt k_{1*}

Concept: Key Ratcheting



Property 1: Old keys can’t be recovered from new keys

Diffie-Hellman Key Ratcheting

$$
\begin{aligned}
A \to B: &\quad g^{x_1} \\
B \to A: &\quad g^{y_1} \\
A \to B: &\quad g^{x_2}, E(M_1, k_{11}) \\
B \to A: &\quad g^{y_2}, E(M_2, k_{21}) \\
A \to B: &\quad g^{x_3}, E(M_3, k_{22})
\end{aligned}
$$

Suppose Alice's compromised at *this* point: leaks k_{2*}

Alice's anti virus killed the trojan at *this* point: secrecy sealed by k_{3*}

where $k_{ij} = H(g^{x_i y_j})$, the result of a 128-bit hash function

- **Property 2 (Self-healing)**: After the attacker is removed, compromise of x_n is *healed* by x_{n+1}. (Same for y.)

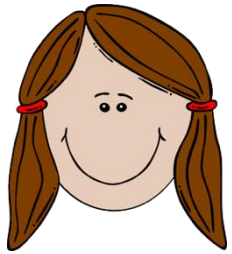
Step 4: Publish MK

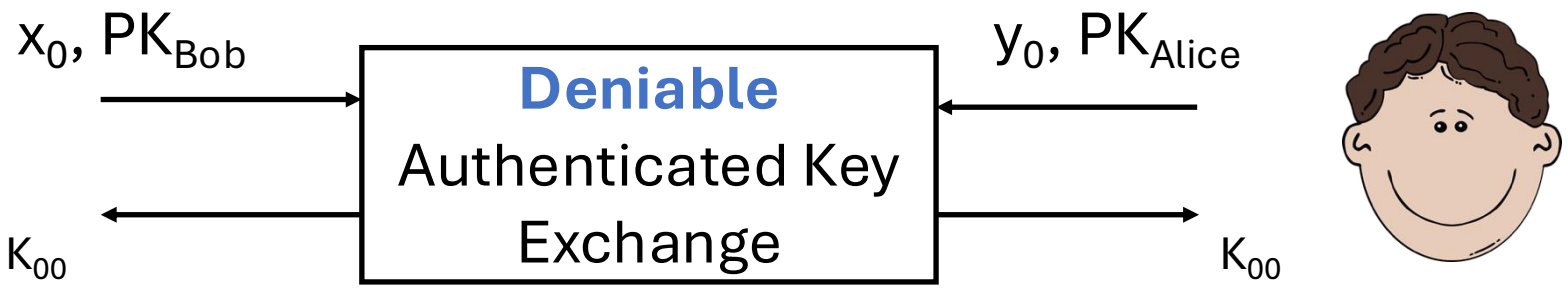
- For *extra* deniability, Alice and Bob publish old MAC keys.

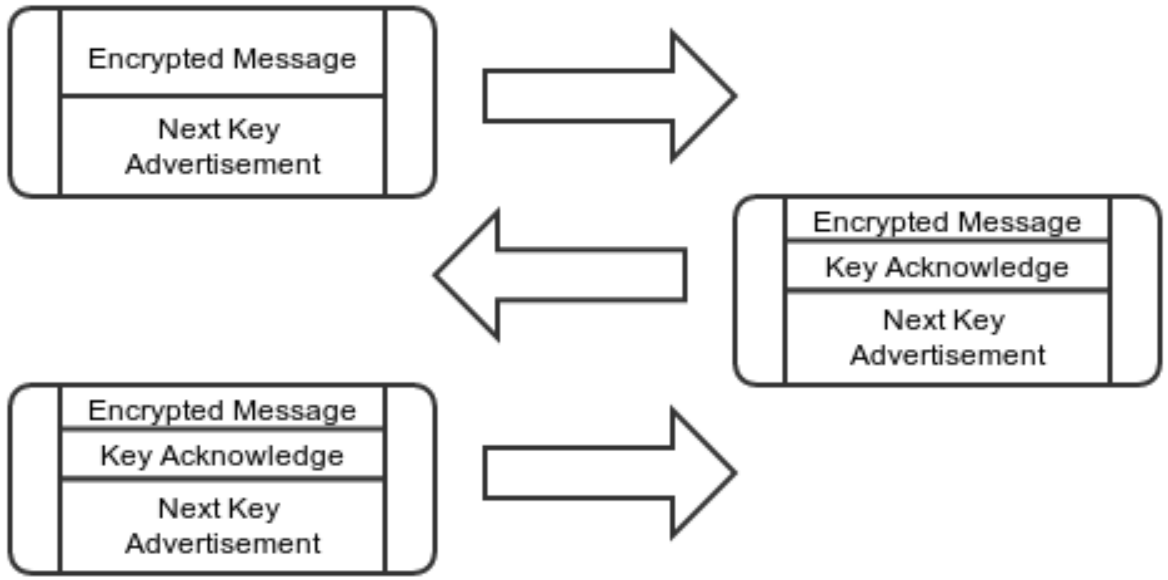
- How?
  - E.g., Alice sends (MK_{n-1}, g^{x_n}, $E(msg_n)$) to Bob
  - Since the channel is insecure, MK_{n-1} is plausibly leaked

- This lets anyone forge messages
  - Provides extra deniability at message level
- Could be added to DKIM, which rotates keys every 3 months

Putting everything together: Recipe for E2EE







Key Ratcheting

Publishing old MAC keys

OTR

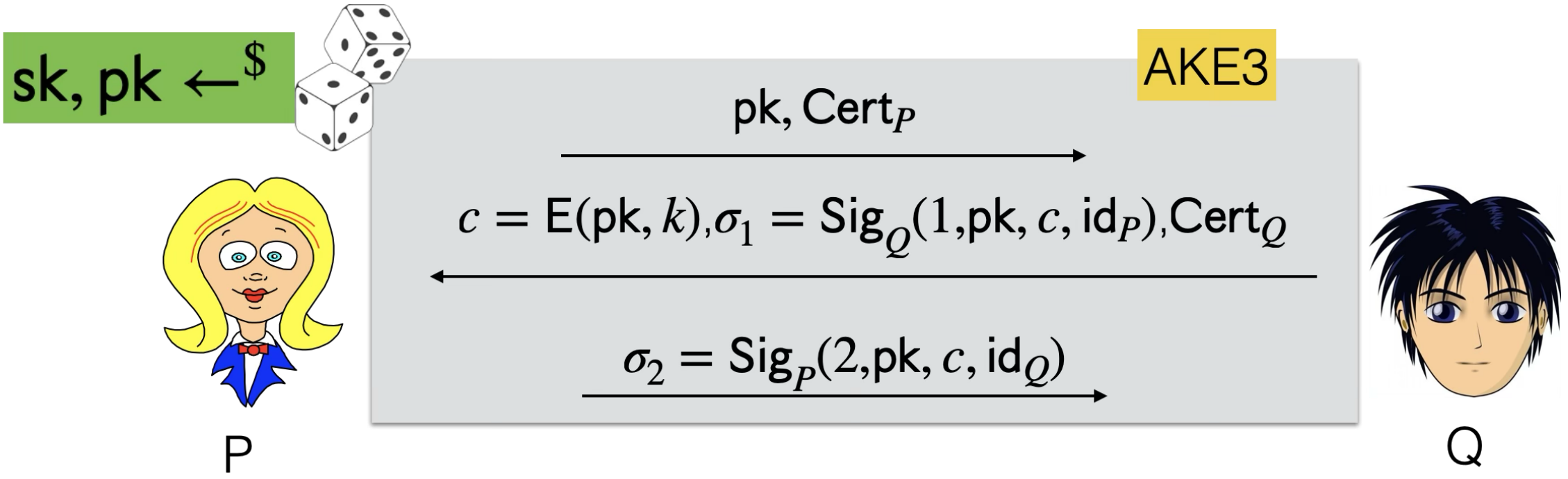
- OTR (2004) initiates the study of **secure messaging**
- Features: deniability and forward secrecy
- Modern versions and variants of OTR
  - OTR v4
  - Signal, WhatsApp, Facebook Messenger

Deniable
Authenticated Key Exchange

Techniques to achieve Deniability

- Deniability for P is ensured by a proof that when P engages in the protocol with Q, whatever evidence an attacker A is able to gather that might implicate P, A could have generated on its own.
- Technically: Show a simulator S who can generate transcripts indistinguishable from real conversation
  - without talking to P,
  - or only involving P.
- Conceptually similar to how we show zero-knowledge of Schnorr ID protocol.

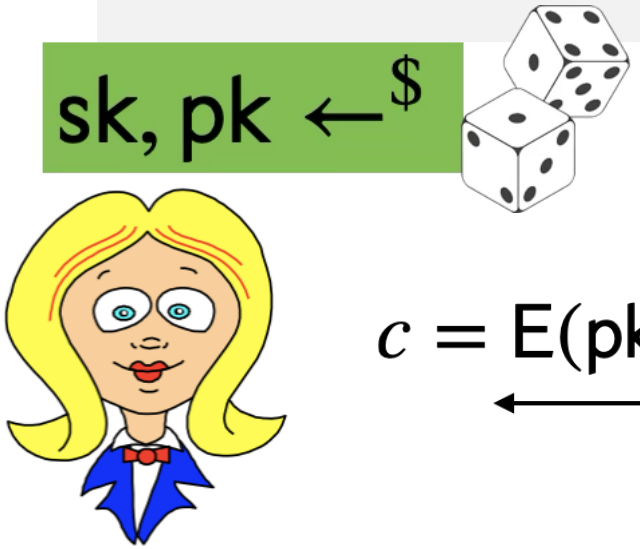
Warm-up: AKE3 from Lecture 3



Warmup: Is AKE3 deniable by P (against Eavesdropper or malicious Q)?

What about Q?

AKE4



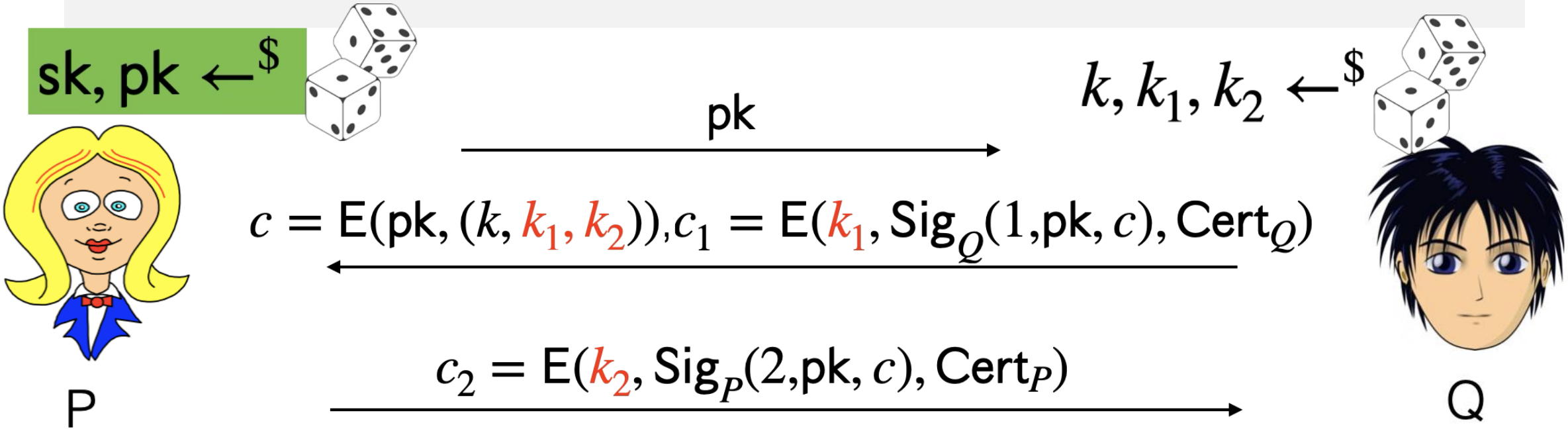
$$\text{pk} \longrightarrow$$

$$c = \mathsf{E}(\text{pk}, (k, k_1, k_2)), c_1 = \mathsf{E}(k_1, \text{Sig}_Q(1, \text{pk}, c), \text{Cert}_Q) \longleftarrow$$

$$c_2 = \mathsf{E}(k_2, \text{Sig}_P(2, \text{pk}, c), \text{Cert}_P) \longrightarrow$$

- Is AKE4 deniable by P against an eavesdropper?

AKE4



- Is AKE4 deniable by P against a malicious Q?
- Proof: P can say: “I was talking with someone else.”
- Q’s attack: choose $k = H(Sig_Q(pk))$.

AKE5 (Nobody signs anything)

P

Public Key $A = g^{\alpha}$

$u \leftarrow^{\$} Z_q$

$U = g^u, Cert_P$ →

Q

Public Key $B = g^{\beta}$

- $v \leftarrow^{\$} Z_q$
- $k_1, k_2, k_3 = H((AU)^{\beta+v}, \dots, id_P, id_Q)$

← $V = g^v, k_1, Cert_Q$

$k_1, k_2, k_3 = H((BV)^{\alpha+u}, \dots, id_P, id_Q)$

Check against k_1 received

k_2 →

Check against own k_2

Use k_3 as session key

Use k_3 as session key

Deniability argument (intuition)

Public Key g^{α} (Q doesn't have this secret key)

Public Key g^{β}

$u \leftarrow^{\$} Z_q$

$U = g^u, Cert_P$

Q

$v \leftarrow^{\$} Z_q$

$k_1, k_2, k_3 = H((AU)^{\beta+v}, \dots)$

$g^v, k_1, Cert_Q$

k_2

Q can generate an identically distributed transcript by himself without talking to P.

Highlights of Signal Protocol

A bit history about Signal

- Introduced in 2010 (then TextSecure)
- Acquired by Twitter in 2011
- In 2013, key people left Twitter and founded Open Whisper Systems. Fully open-sourced TextSecure.
- Introduced ***Double Ratcheting***
- Renamed to Signal in 2014
- The **Signal *protocol*** is used by WhatsApp in 2016 for its billions of users.
- As of January 2022, the Signal app had approximately 40 million monthly active users [Wikipedia]

Signal's X3DH

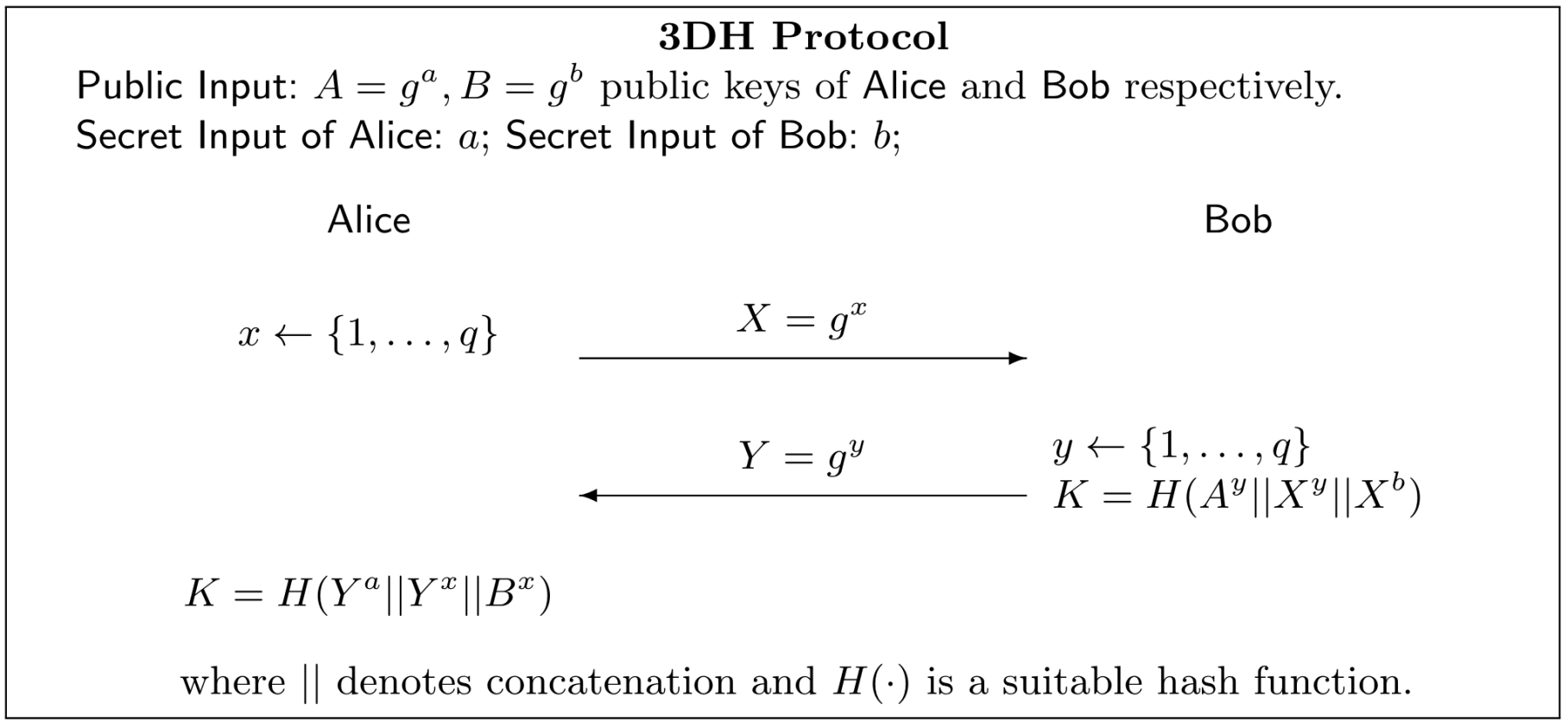


Fig. 2. Triple Diffie-Hellman key exchange protocol

Signal's X3DH

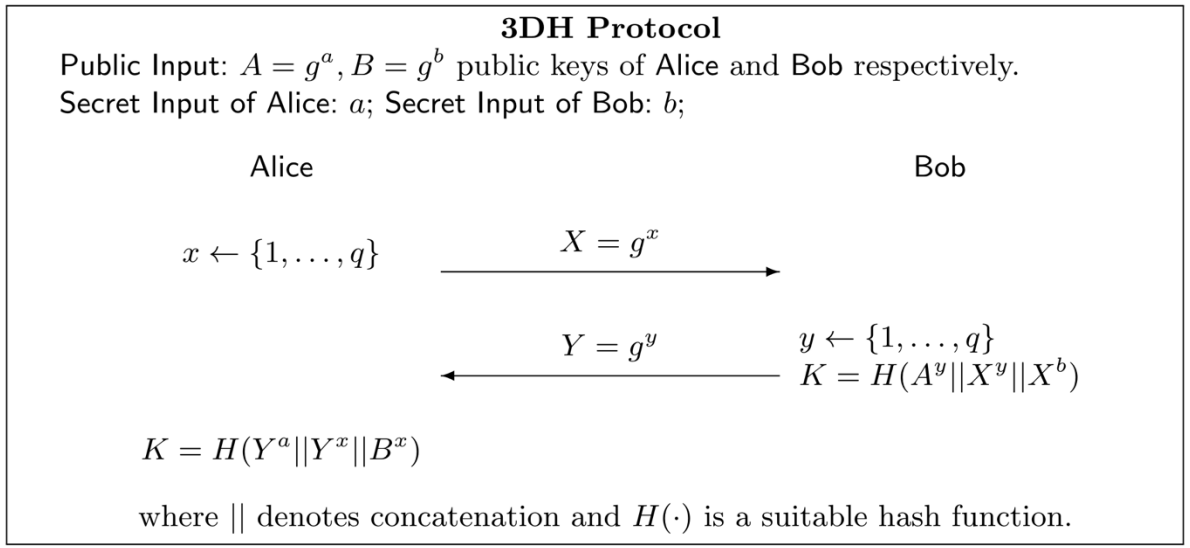


Fig. 2. Triple Diffie-Hellman key exchange protocol

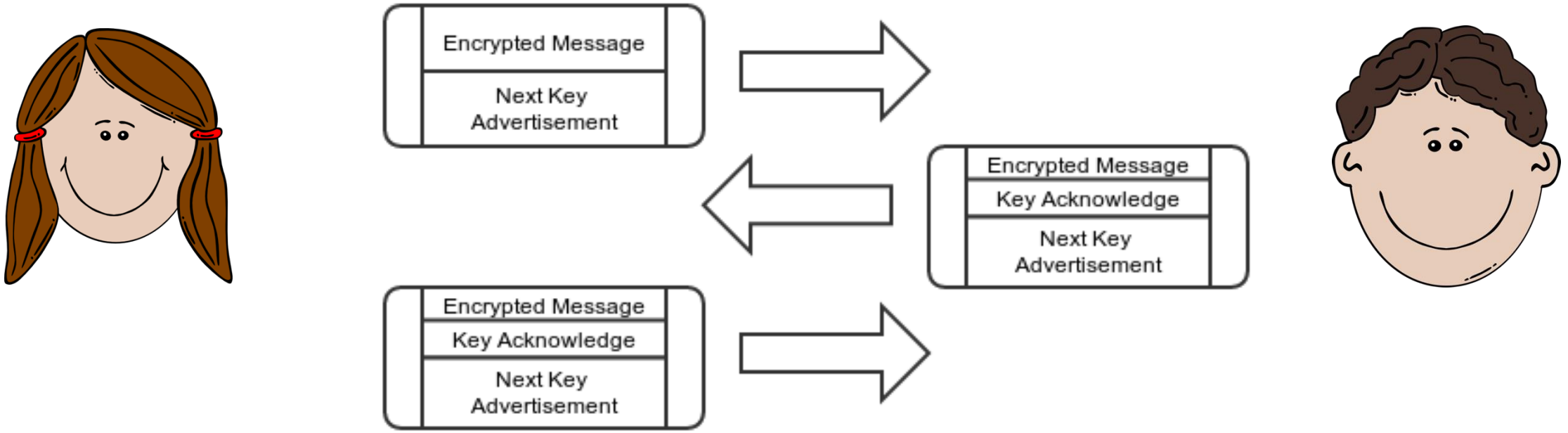
- Alice and Bob have long term keys g^a, g^b and short-term keys g^x, g^y.
- $K = H(g^{ya} \,||\, g^{xy} \,||\, g^{xb})$

- **Authenticity**: an MiTM attacker must know **a and x**, or **b and y,** or **x and y**

- ***Full* Participation Repudiation**: Intuitively, anyone can forge a session between Alice and Bob (generate random x, y)
- Proof that any transcript can be *simulated* is non-trivial [ANCS20].

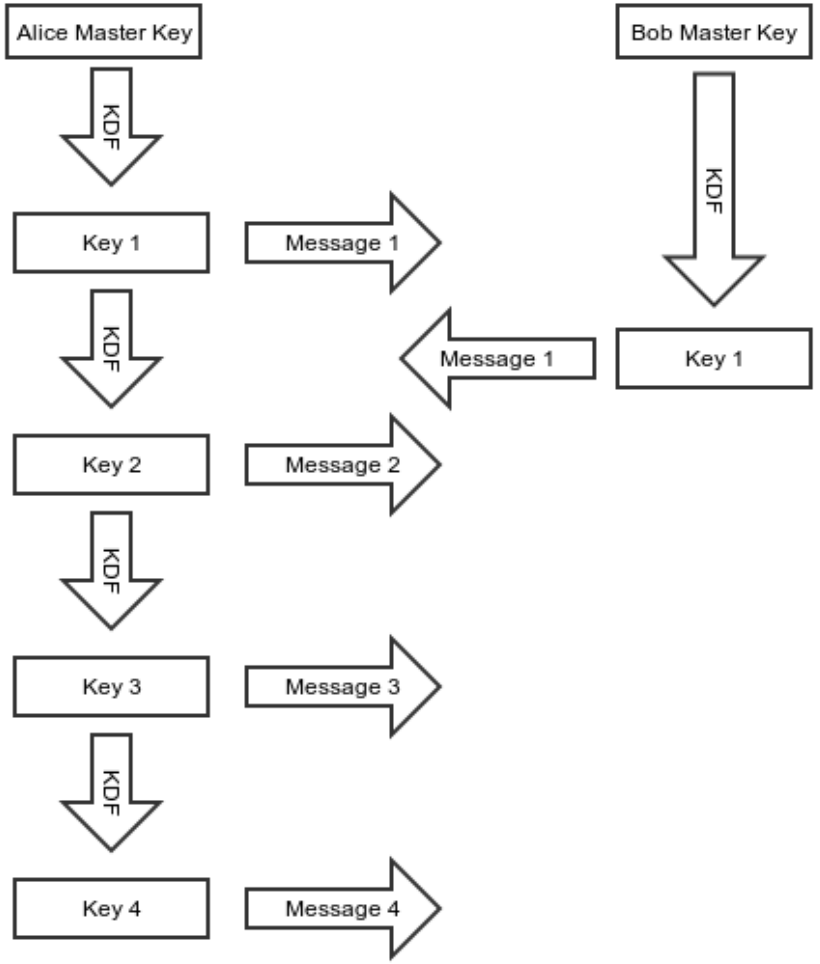
Full proof can be found in https://eprint.iacr.org/2021/642

Signal's observations about DH Ratcheting



- Seems an **overkill** to redo AKE between every message (key compromise unlikely)
- Make sense to reduce cadence (e.g., every change of sender, or first message after long silence)
- How to refresh keys in between DH Ratcheting?

Alternative: hash chain



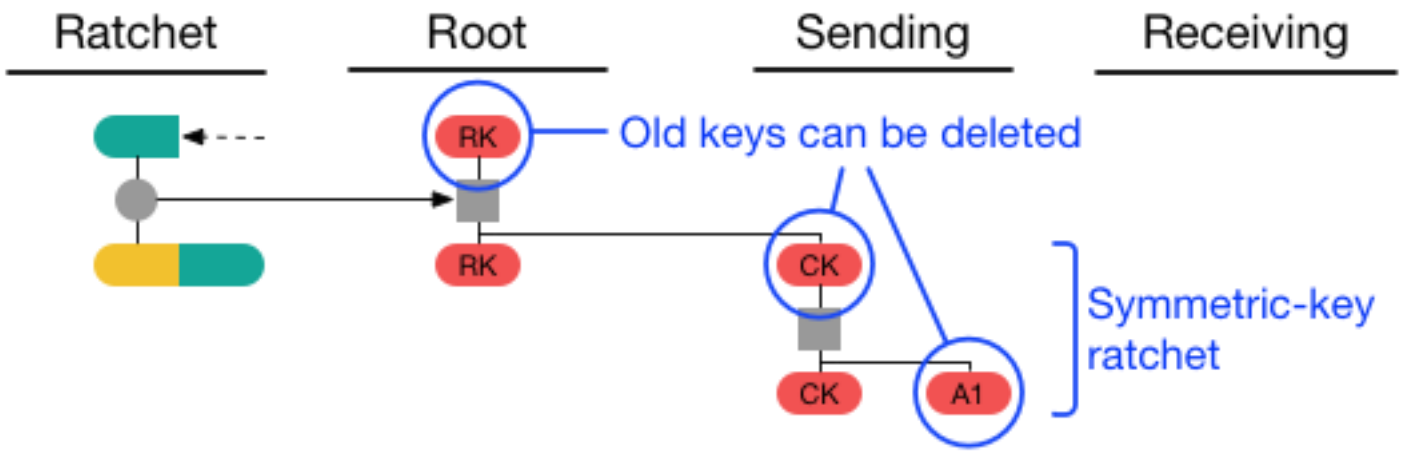
- KDF ratchet:

  $$Key_k = hash^k(Key_0)$$

- I.e., repeated hash
- Upsides:
  - Forward secrecy (can't invert hash func.)
  - Computationally light weight
  - non-interactive
  - Handling of out-of-order delivery is easier: msgs are attached with seq# (can "skip")
- Downsides: not self-healing

Signal's Double Ratcheting Algorithm

- Combines **hash chain** and **DH ratchets**
- For Alice: DH ratchet is advanced every time Bob replies; before that, each msg advances the hash chain ratchet.
- Two ratchets: send/receive
- Handles out of order msg



A problem hidden under the rug: Trust establishment

- How do Alice know that she is communicating with Bob?
- Signal uses

Safety Numbers

- We will study *Key Transparency Logs* in the next chapter.

Summary: Secure Messaging

- Important concepts pioneered by OTR: **forward secrecy**, **post-compromise security (self-healing)**, **deniability**
- Secure messaging protocol typically involves
  - Deniable Authenticated Key Exchange
  - Key ratcheting algorithms
  - Releasing the MAC key if necessary
- Signal is inspired by ORT, with various features
  - Asynchronous msgs
  - out-of-order msgs
  - Verification via Safety Numbers
  - Social discovery
- Under the rug: trust establishment