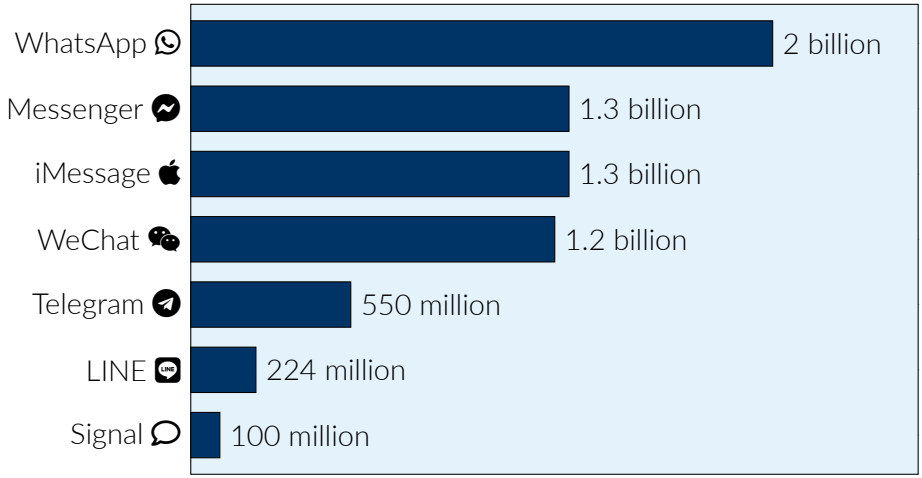
An Efficient and Generic Construction for Signal's Handshake (X3DH)

Post-Quantum, State Leakage Secure, and Deniable

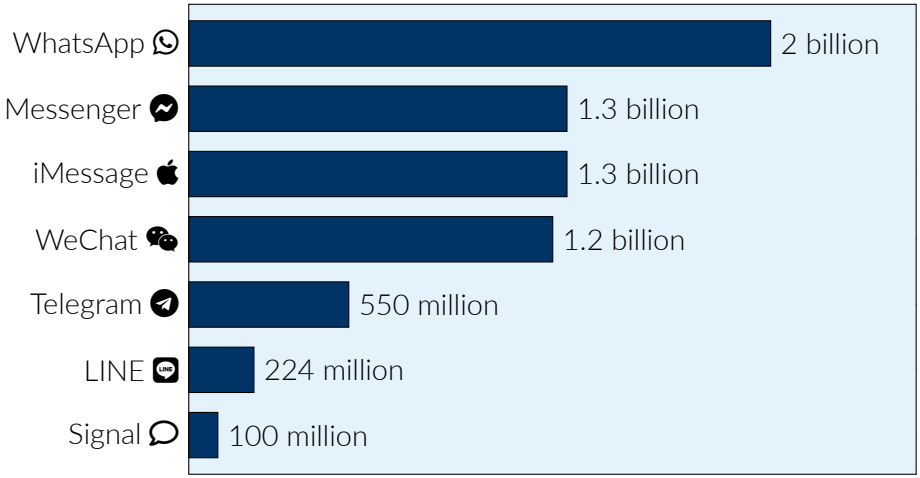
Keitaro Hashimoto[1] Shuichi Katsumata[2,3]
Kris Kwiatkowski[3] Thomas Prest[3]

Tokyo Institute of Technology AIST PQShield

Messaging apps

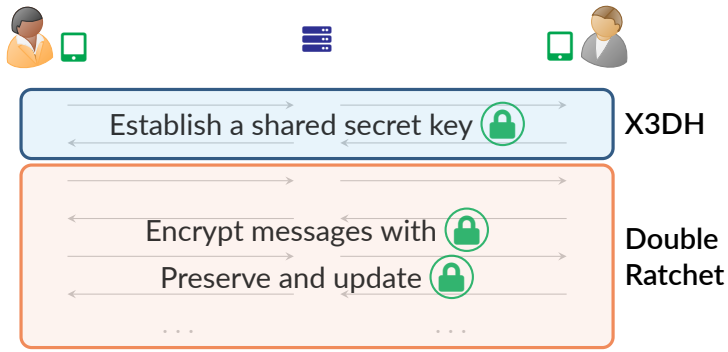


Messaging apps



Most of these apps use specific protocols.
We focus on variations of the Signal protocol (,,).

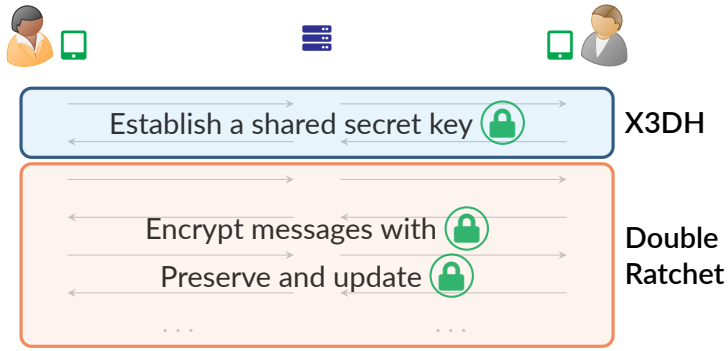
The Signal protocol



Two main threats

- Server compromise
- Device compromise

The Signal protocol

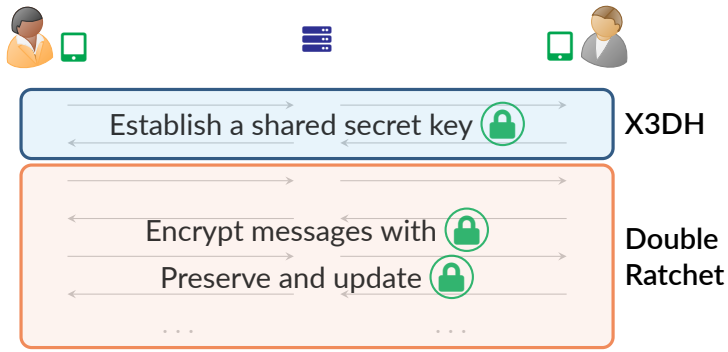


Pre-quantum world:

- **X3DH:**[1] Diffie-Hellman + XEdDSA + symmetric crypto (HKDF)
- **Double Ratchet:**[2] Diffie-Hellman + symmetric crypto (HKDF, HMAC, ...)

[1] https://signal.org/docs/specifications/x3dh/
[2] https://signal.org/docs/specifications/doubleratchet/

The Signal protocol



Post-quantum world:

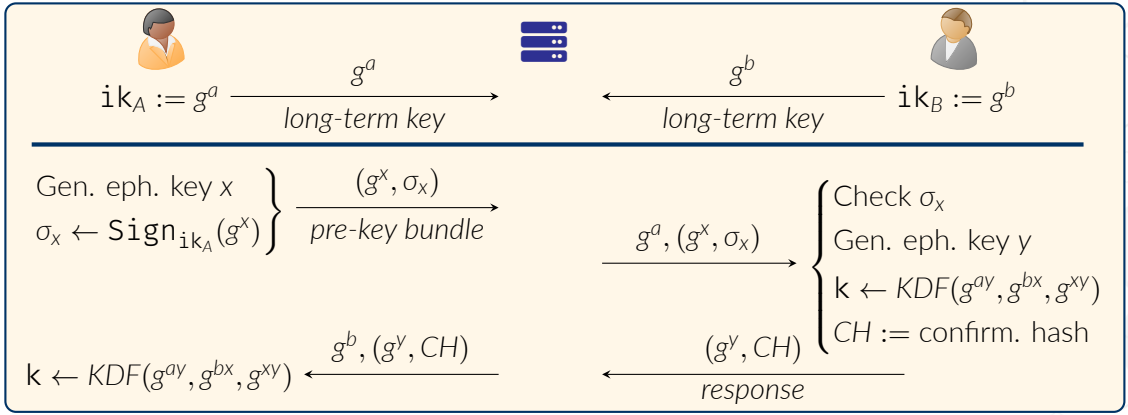
- **PQ X3DH:** KEM + (ring) signatures + symmetric crypto **[this work]**
- **PQ Double Ratchet:** KEM + symmetric crypto [ACD19]

Roadmap

1. Capture/formalize the properties expected from Signal's X3DH
2. Propose a generic construction that satisfies these properties

Understanding X3DH

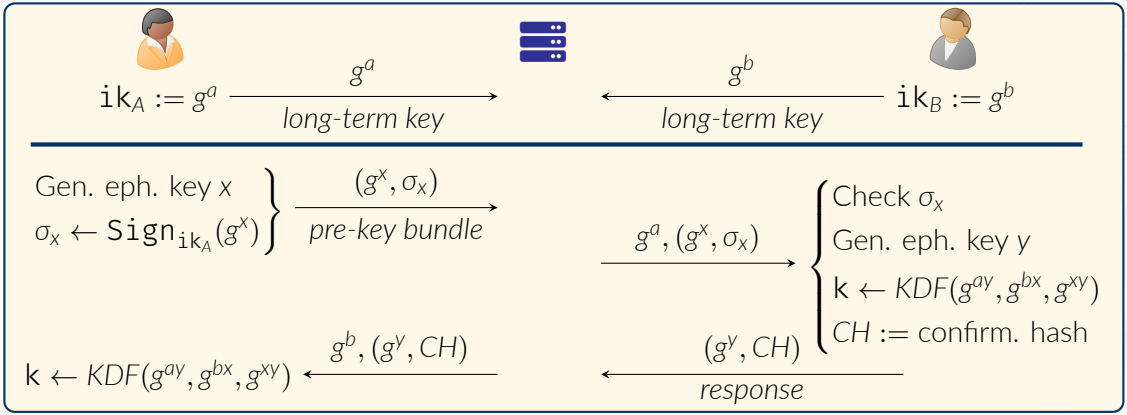
Reverse-engineering X3DH's spec [MP16]



Initial comments:

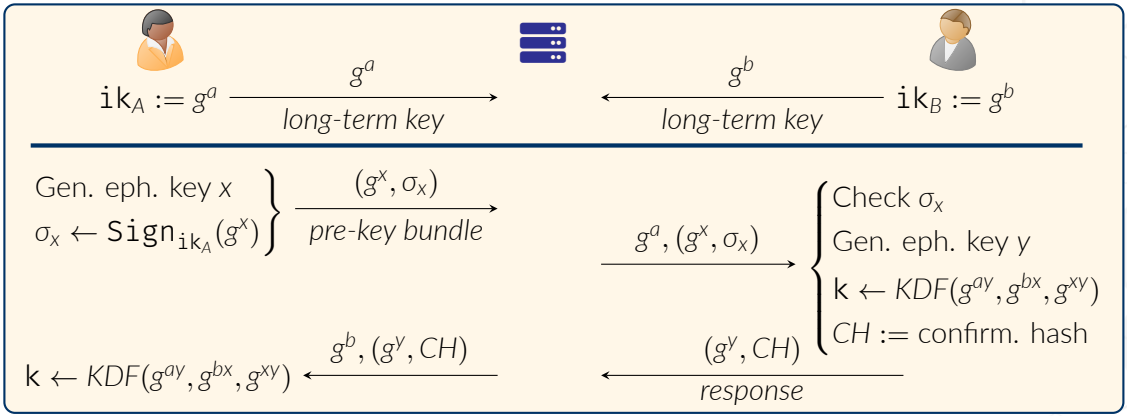
- Builds upon 3DH (same protocol without σ_x and CH).
- Note the two-message flow and the "*receiver-obliviousness*" of pre-key bundle.
- Parties delete ephemeral keys (x and y) as soon as they can (omitted in figure).

Authentication and forward secrecy



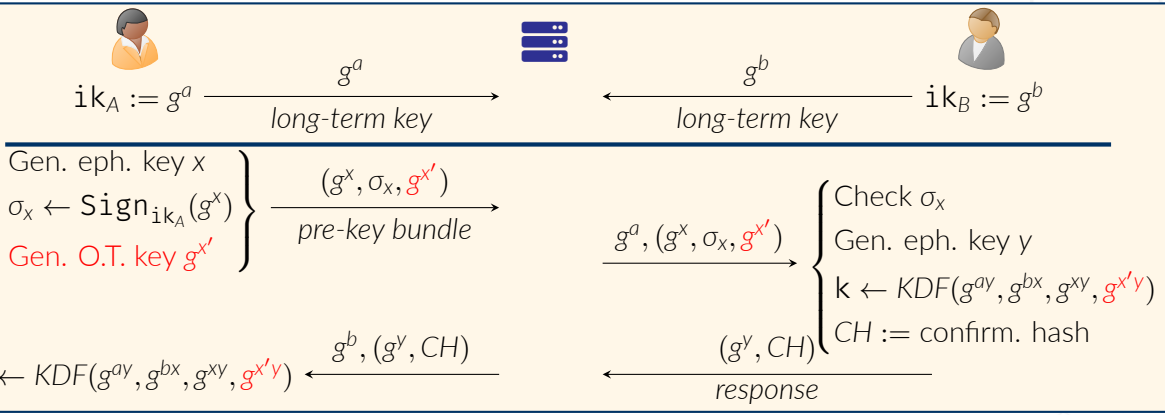
" Note that $[g^{ay}]$ and $[g^{bx}]$ provide mutual authentication, while $[g^{xy}$ provides] forward secrecy. "

Deniability



" X3DH doesn't give either Alice or Bob a publishable cryptographic proof of the contents of their communication or the fact that they communicated. "

One-time keys (optional)



X3DH allows to provide an optional one-time key.

- *Motivation:* (1) forward secrecy, and (2) protection against replay attacks
  - *Replay attacks:* there are simpler ways to thwart them
  - *Forward secrecy:* largely addressed by signed pre-keys and Double Ratchet

Wishlist

Functional properties:

- *Two-message flow:* Initiator → Responder → Initiator
- *Receiver-obliviousness:* first message is independent of the Responder[1]

Security properties:

- *Confidentiality:* session keys looks pseudorandom except for intended parties
- *Mutual authentication:* Initiator knows she talks to Responder, and reciprocally
  - Includes *Key-Compromise Impersonation* (KCI) attacks[2]
  - Includes *Unknown Key-Share* (UKS) attacks[3]
- *Deniability:* online/offline, against semi-honest/malicious adversaries

[1] Also known as *post-specified peers*.

[2] KCI: The adversary uses A's long-term key to impersonate other parties towards A.

[3] UKS: A and B compute the same key, but A believe he's talking to C.

Generic Constructions

Building blocks

Key establishment mechanism (KEM):

- $\mathtt{ek}$ = encapsulation key
- $\mathtt{dk}$ = decapsulation key
- $\mathtt{c}$, $\mathtt{k}$ = ciphertext, encapsulated key

- $\mathtt{Keygen}() \to (\mathtt{ek}, \mathtt{dk})$
- $\mathtt{Encaps}(\mathtt{ek}) \to (\mathtt{c}, \mathtt{k})$
- $\mathtt{Decaps}(\mathtt{dk}, \mathtt{c}) \to \mathtt{k}/\perp$

(Ring) signature scheme:

- $\mathtt{sk}$ = signing key
- $\mathtt{vk}$ = verification key
- σ = signature

- $\mathtt{Keygen}() \to (\mathtt{vk}, \mathtt{sk})$
- $\mathtt{Sign}_{\mathtt{sk}}(\mathtt{msg}) \to \sigma$
- $\mathtt{Verify}_{\mathtt{vk}}(\sigma, \mathtt{msg}) \to \top/\perp$

Pseudo-random function (PRF) $F_{\mathbf{k}}$.

V0: first generic construction

$\mathtt{lsk}_A := (\mathtt{sk}_A, \mathtt{dk}_A)$
$\mathtt{lpk}_A := (\mathtt{vk}_A, \mathtt{ek}_A)$

$\mathtt{lsk}_B := (\mathtt{sk}_B, \mathtt{dk}_B)$
$\mathtt{lpk}_B := (\mathtt{vk}_B, \mathtt{ek}_B)$

$(\mathtt{ek}_T, \mathtt{dk}_T) \leftarrow \mathsf{KEM.Keygen}()$
$\sigma_A \leftarrow \mathtt{Sign}_{\mathtt{sk}_A}(\mathtt{ek}_T)$

$\xrightarrow{\mathtt{lpk}_A, \mathtt{ek}_T, \sigma_A}$

$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_A, \mathtt{ek}_T)$
$(\mathtt{c}_A, \mathtt{k}_A) \leftarrow \mathtt{Encaps}(\mathtt{ek}_A)$
$(\mathtt{c}_T, \mathtt{k}_T) \leftarrow \mathtt{Encaps}(\mathtt{ek}_T)$
$\mathtt{k} \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathtt{Sign}_{\mathtt{sk}_B}(\mathtt{sid})$

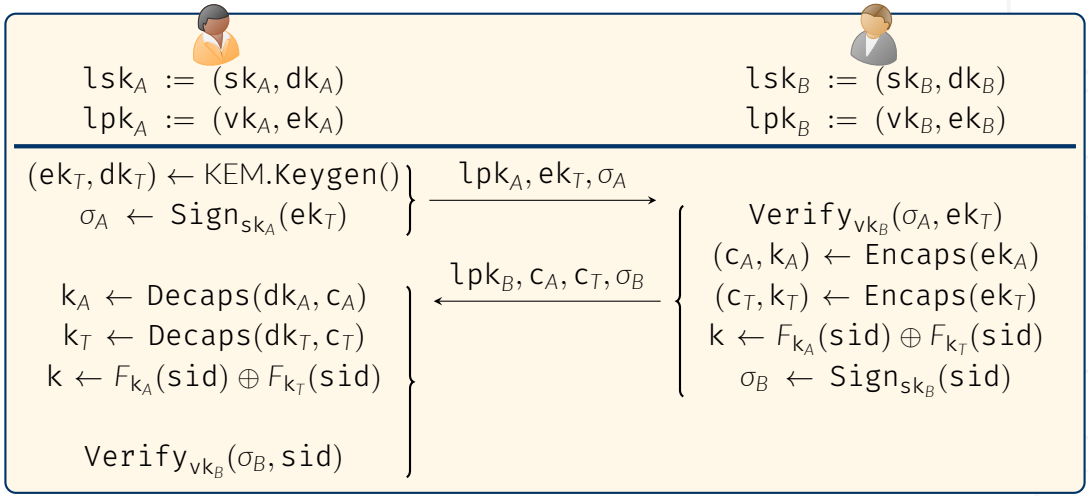
$\xleftarrow{\mathtt{lpk}_B, \mathtt{c}_A, \mathtt{c}_T, \sigma_B}$

$\mathtt{k}_A \leftarrow \mathtt{Decaps}(\mathtt{dk}_A, \mathtt{c}_A)$
$\mathtt{k}_T \leftarrow \mathtt{Decaps}(\mathtt{dk}_T, \mathtt{c}_T)$
$\mathtt{k} \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$

$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_B, \mathtt{sid})$

$\mathtt{sid} = (\mathtt{lpk}_A \| \mathtt{lpk}_B \| \mathtt{ek}_A \| \mathtt{c}_A \| \mathtt{c}_T)$ is the session identifier.

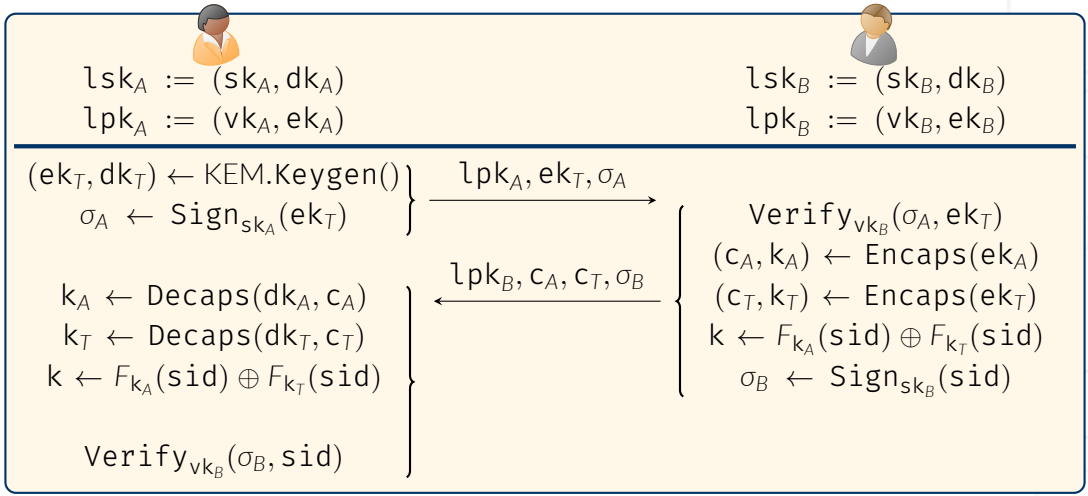
V0: first generic construction



Leakage:

- *Leakage of long-term keys:* secrecy of k is guaranteed by ek_T and c_T
- *State leakage (irrelevant for* *):* secrecy of k is guaranteed by c_A

V0: first generic construction



Mutual authentication:

- → is *explicitly* authenticated by σ_B.
- → is *implicitly* authenticated.

- → *KCI:* prevented by signatures σ_A and σ_B
- → *UKS:* prevented by $\mathtt{sid}$

V0: first generic construction

$\mathtt{lsk}_A := (\mathtt{sk}_A, \mathtt{dk}_A)$
$\mathtt{lpk}_A := (\mathtt{vk}_A, \mathtt{ek}_A)$

$\mathtt{lsk}_B := (\mathtt{sk}_B, \mathtt{dk}_B)$
$\mathtt{lpk}_B := (\mathtt{vk}_B, \mathtt{ek}_B)$

$(\mathtt{ek}_T, \mathtt{dk}_T) \leftarrow \mathsf{KEM.Keygen}()$
$\sigma_A \leftarrow \mathtt{Sign}_{\mathtt{sk}_A}(\mathtt{ek}_T)$

$\xrightarrow{\mathtt{lpk}_A, \mathtt{ek}_T, \sigma_A}$

$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_A, \mathtt{ek}_T)$
$(\mathtt{c}_A, \mathtt{k}_A) \leftarrow \mathtt{Encaps}(\mathtt{ek}_A)$
$(\mathtt{c}_T, \mathtt{k}_T) \leftarrow \mathtt{Encaps}(\mathtt{ek}_T)$
$\mathtt{k} \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathtt{Sign}_{\mathtt{sk}_B}(\mathtt{sid})$

$\xleftarrow{\mathtt{lpk}_B, \mathtt{c}_A, \mathtt{c}_T, \sigma_B}$

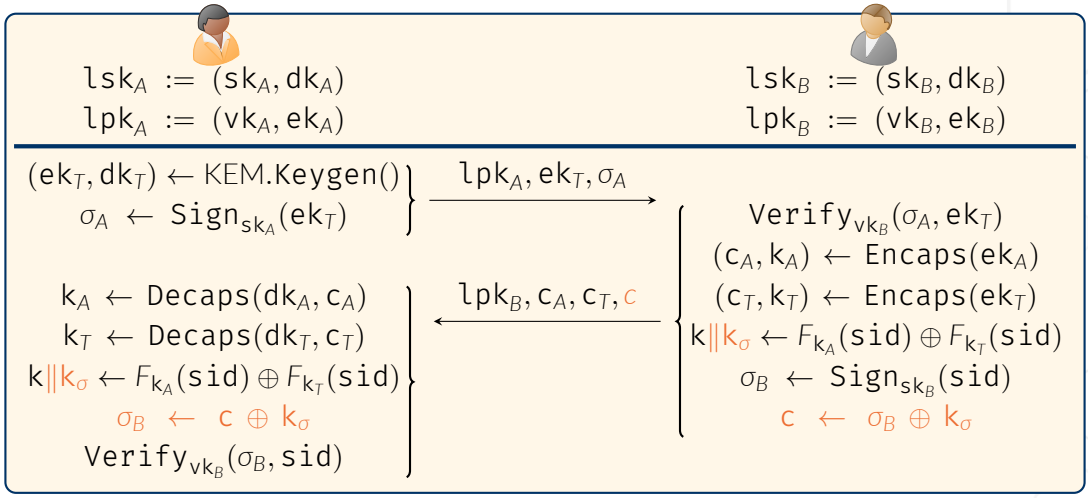
$\mathtt{k}_A \leftarrow \mathtt{Decaps}(\mathtt{dk}_A, \mathtt{c}_A)$
$\mathtt{k}_T \leftarrow \mathtt{Decaps}(\mathtt{dk}_T, \mathtt{c}_T)$
$\mathtt{k} \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$

$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_B, \mathtt{sid})$

Minor possible tweaks:

- *Omit* σ_A*:* Downgrades PFS to weak PFS (Double Ratchet then provides PFS again)
- *NAXOS trick:* mitigates leakage of 's randomness by combining it with $\mathtt{lsk}_B$

V1: very weakly deniable



We can tweak the protocol to achieve a weak flavour of deniability for free.

- We mask σ_B using a pseudorandom keystream derived from $\mathtt{k}_A, \mathtt{k}_T$ (tangerine).
- The transcript makes anonymous as long as doesn't cooperate.

V1.5: weakly deniable

$\mathtt{lsk}_A := (\mathtt{sk}_A, \mathtt{dk}_A)$
$\mathtt{lpk}_A := (\mathtt{vk}_A, \mathtt{ek}_A)$

$\mathtt{lsk}_B := (\mathtt{sk}_B, \mathtt{dk}_B)$
$\mathtt{lpk}_B := (\mathtt{vk}_B, \mathtt{ek}_B)$

$(\mathtt{ek}_T, \mathtt{dk}_T) \leftarrow \mathsf{KEM.Keygen}()$
$\sigma_A \leftarrow \mathtt{Sign}_{\mathtt{sk}_A}(\mathtt{ek}_T)$

$\mathtt{lpk}_A, \mathtt{ek}_T, \sigma_A$ →

$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_A, \mathtt{ek}_T)$
$(\mathtt{c}_A, \mathtt{k}_A) \leftarrow \mathsf{Encaps}(\mathtt{ek}_A)$
$(\mathtt{c}_T, \mathtt{k}_T) \leftarrow \mathsf{Encaps}(\mathtt{ek}_T)$
$\mathtt{k} \| \mathtt{k}_\sigma \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathsf{RS.Sign}_{\mathtt{sk}_B, \mathtt{ring}}(\mathtt{sid})$
$\mathtt{c} \leftarrow \sigma_B \oplus \mathtt{k}_\sigma$

← $\mathtt{lpk}_B, \mathtt{c}_A, \mathtt{c}_T, \mathtt{c}$

$\mathtt{k}_A \leftarrow \mathsf{Decaps}(\mathtt{dk}_A, \mathtt{c}_A)$
$\mathtt{k}_T \leftarrow \mathsf{Decaps}(\mathtt{dk}_T, \mathtt{c}_T)$
$\mathtt{k} \| \mathtt{k}_\sigma \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathtt{c} \oplus \mathtt{k}_\sigma$
$\mathsf{RS.Verify}_{\mathtt{ring}}(\sigma_B, \mathtt{sid})$

- Main idea: replace signatures by *ring* signatures (pine green).
- Setting $\mathtt{ring} = \{\mathtt{vk}_A, \mathtt{vk}_B\}$ only gives a weak flavour of deniability.

V2: deniable (offline, semi-honest)

$\mathtt{lsk}_A := (\mathtt{sk}_A, \mathtt{dk}_A)$
$\mathtt{lpk}_A := (\mathtt{vk}_A, \mathtt{ek}_A)$

$\mathtt{lsk}_B := (\mathtt{sk}_B, \mathtt{dk}_B)$
$\mathtt{lpk}_B := (\mathtt{vk}_B, \mathtt{ek}_B)$

$(\mathtt{ek}_T, \mathtt{dk}_T) \leftarrow \mathsf{KEM.Keygen}()$
$\sigma_A \leftarrow \mathtt{Sign}_{\mathtt{sk}_A}(\mathtt{ek}_T)$
$(\mathtt{sk}_T, \mathtt{vk}_T) \leftarrow \mathsf{RS.Keygen}()$

$\xrightarrow{\mathtt{lpk}_A, \mathtt{ek}_T, \sigma_A, \mathtt{vk}_T}$

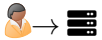
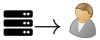
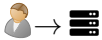
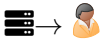
$\mathtt{Verify}_{\mathtt{vk}_B}(\sigma_A, \mathtt{ek}_T)$
$(\mathtt{c}_A, \mathtt{k}_A) \leftarrow \mathtt{Encaps}(\mathtt{ek}_A)$
$(\mathtt{c}_T, \mathtt{k}_T) \leftarrow \mathtt{Encaps}(\mathtt{ek}_T)$
$\mathtt{k} \| \mathtt{k}_\sigma \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathsf{RS.Sign}_{\mathtt{sk}_B, \mathtt{ring}}(\mathtt{sid})$
$\mathtt{c} \leftarrow \sigma_B \oplus \mathtt{k}_\sigma$

$\xleftarrow{\mathtt{lpk}_B, \mathtt{c}_A, \mathtt{c}_T, \mathtt{c}}$

$\mathtt{k}_A \leftarrow \mathtt{Decaps}(\mathtt{dk}_A, \mathtt{c}_A)$
$\mathtt{k}_T \leftarrow \mathtt{Decaps}(\mathtt{dk}_T, \mathtt{c}_T)$
$\mathtt{k} \| \mathtt{k}_\sigma \leftarrow F_{\mathtt{k}_A}(\mathtt{sid}) \oplus F_{\mathtt{k}_T}(\mathtt{sid})$
$\sigma_B \leftarrow \mathtt{c} \oplus \mathtt{k}_\sigma$
$\mathsf{RS.Verify}_{\mathtt{ring}}(\sigma_B, \mathtt{sid})$

- Main idea: replace signatures by *ring* signatures (pine green).
- Setting $\mathtt{ring} = \{\mathtt{vk}_A, \mathtt{vk}_B\}$ only gives a weak flavour of deniability.
- Setting $\mathtt{ring} = \{\mathtt{vk}_T, \mathtt{vk}_B\}$ with $\mathtt{vk}_T$ an ephemeral key makes (V2) truly deniable.

Bandwidth cost of V1 in bytes

Scheme	`lpk`				
Kyber512 + Falcon512	1697	1490	3187	2226	3923
Kyber512 + Dilithium2	2112	3220	5332	3956	6068
Kyber512 + SPHINCS$^+$	832	17888	18720	18624	19456

Conclusion and Further Reading

Further reading:

- Full paper
  https://ia.cr/2021/616
- Keitaro's presentation at PKC 2021 (video)
  https://www.youtube.com/watch?v=VO4fw0UHdMI
- Keitaro's presentation at PKC 2021 (slides)
  https://kaminomisosiru.github.io/assets/pdf/HKKP-PKC2021.pdf
- C implementation by Kris:
  https://github.com/post-quantum-cryptography/post-quantum-state-leakage-secure-ake

Related works:

- PQ X3DH:
  - Generic [BFG+22]
  - SIDH-based [DG21]
- PQ Double Ratchet
  - Generic [ACD19]

Questions?

https://ia.cr/2021/616

Joël Alwen, Sandro Coretti, and Yevgeniy Dodis.
The double ratchet: Security notions, proofs, and modularization for the Signal protocol.
In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 129–158. Springer, Heidelberg, May 2019.

Jacqueline Brendel, Rune Fiedler, Felix Günther, Christian Janson, and Douglas Stebila.
Post-quantum asynchronous deniable key exchange and the signal handshake.
In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *Public-Key Cryptography – PKC 2022*, pages 3–34, Cham, 2022. Springer International Publishing.

Samuel Dobson and Steven D. Galbraith.
Post-quantum signal key agreement with SIDH.
Cryptology ePrint Archive, Report 2021/1187, 2021.
https://eprint.iacr.org/2021/1187.

Moxie Marlinspike and Trevor Perrin.
The x3dh key agreement protocol, November 2016.
https://signal.org/docs/specifications/x3dh/.