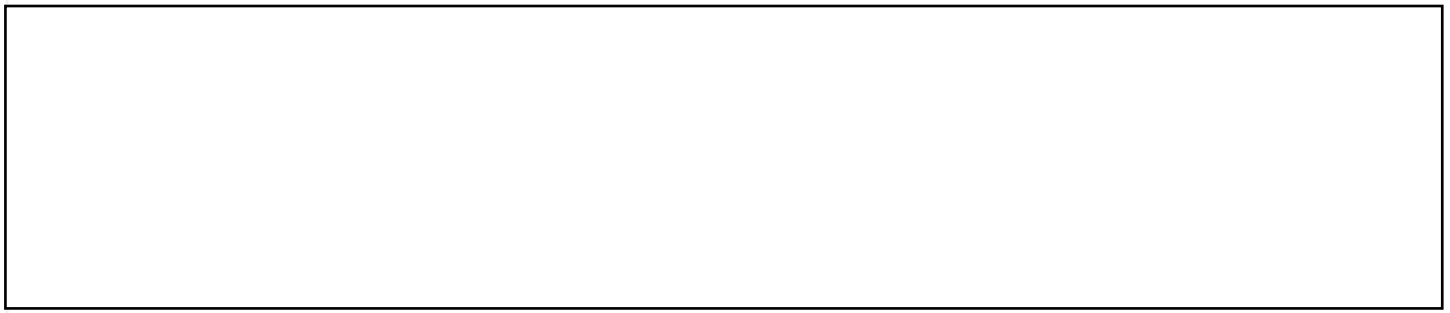
QKD Oracles for Authenticated Key Exchange

ia.cr/2025/1671 arxiv.org/abs/2509.12478

Kathrin Hövelmanns[1], Daan Planken[2], Christian Schaffner[2], **Sebastian Verschoor**[2]

2025-10-16

[1]TU/e, [2]UvA (QuSoft)



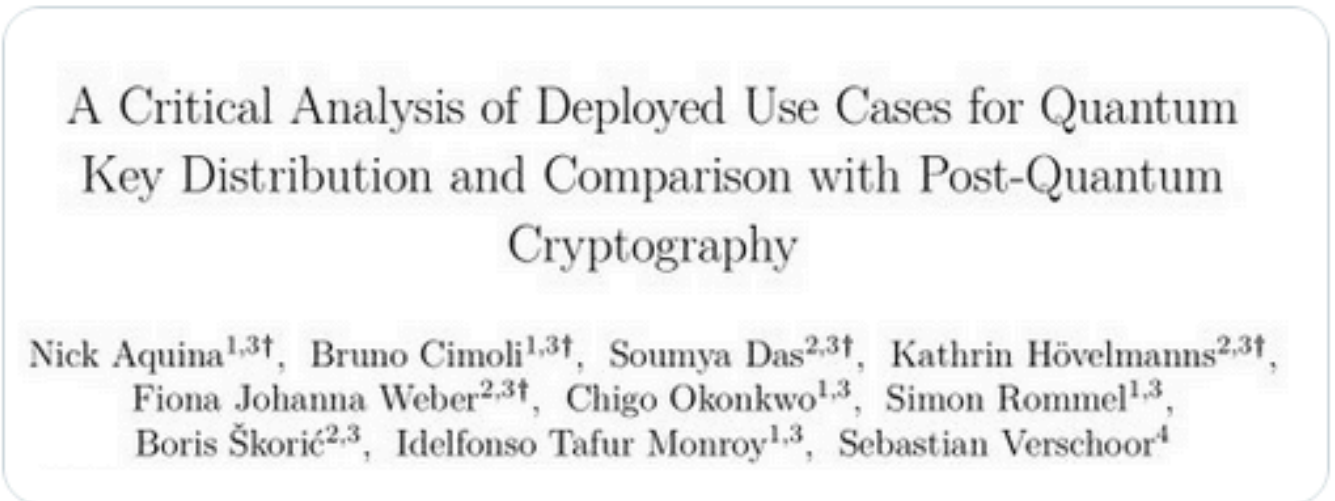
A Critical Analysis of Deployed Use Cases for Quantum Key Distribution and Comparison with Post-Quantum Cryptography

Nick Aquina[1,3†], Bruno Cimoli[1,3†], Soumya Das[2,3†], Kathrin Hövelmanns[2,3†], Fiona Johanna Weber[2,3†], Chigo Okonkwo[1,3], Simon Rommel[1,3], Boris Škorić[2,3], Idelfonso Tafur Monroy[1,3], Sebastian Verschoor[4]

Post

Martin Shkreli
@MartinShkreli

QKD, quantum networking, etc. it's not a thing. $IONQ will not make any money off of this, ever.



A Critical Analysis of Deployed Use Cases for Quantum Key Distribution and Comparison with Post-Quantum Cryptography

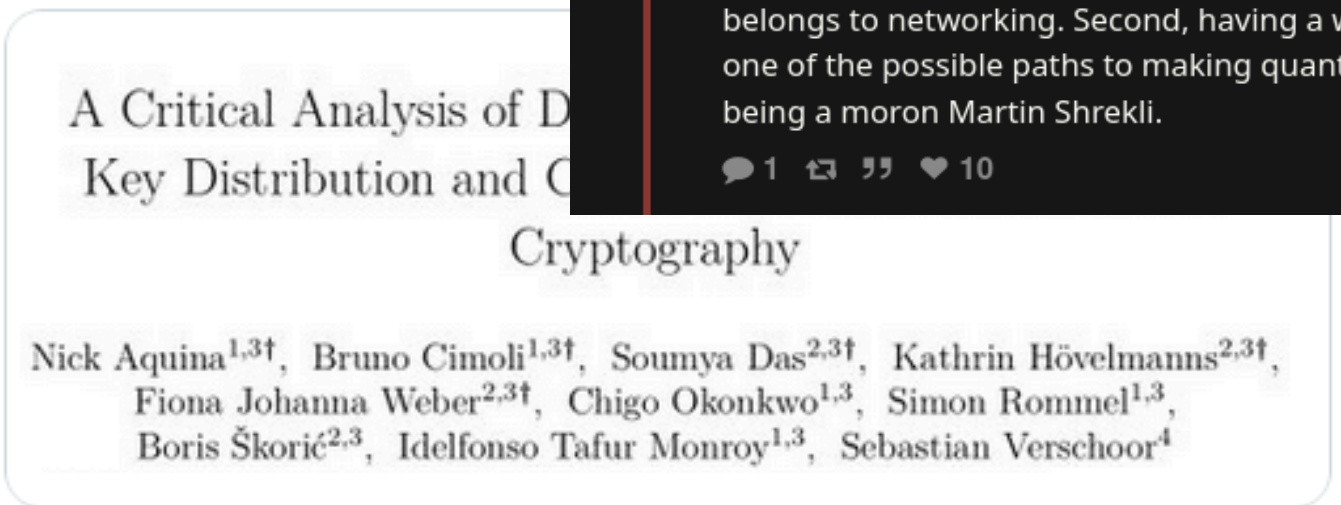
Nick Aquina[1,3†], Bruno Cimoli[1,3†], Soumya Das[2,3†], Kathrin Hövelmanns[2,3†], Fiona Johanna Weber[2,3†], Chigo Okonkwo[1,3], Simon Rommel[1,3], Boris Škorić[2,3], Idelfonso Tafur Monroy[1,3], Sebastian Verschoor[4]

8:35 PM · Sep 17, 2025 · **104.8K** Views

Post

Martin Shkreli ✓
@MartinShkreli

QKD, quantum networking, etc
money off of this, ever.


A Critical Analysis of D
Key Distribution and C
Cryptography

Nick Aquina[1,3†], Bruno Cimoli[1,3†], Soumya Das[2,3†], Kathrin Hövelmanns[2,3†], Fiona Johanna Weber[2,3†], Chigo Okonkwo[1,3], Simon Rommel[1,3], Boris Škorić[2,3], Idelfonso Tafur Monroy[1,3], Sebastian Verschoor[4]

8:35 PM · Sep 17, 2025 · **104.8K** Views

H @QcHst17 Sep 17
Replying to @MartinShkreli
First of all, QKD and networking are not kindred areas of research; QKD belongs to networking. Second, having a well-developed quantum network is one of the possible paths to making quantum computing feasible. Third stop being a moron Martin Shrekli.
1 10

← **Post**

Martin Shkreli ✓
@MartinShkreli

QKD, quantum networking, etc
money off of this, ever.


A Critical Analysis of D
Key Distribution and C
Cr
Nick Aquina[1,3†], Bruno Cimoli[1,3†],
Fiona Johanna Weber[2,3†],
Boris Škorić[2,3], Idelfonso Tafur Monroy[ ], Sebastian Verschoor

H @QcHst17 Sep 17
Replying to @MartinShkreli
First of all, QKD and networking are not kindred areas of research; QKD belongs to networking. Second, having a well-developed quantum network is one of the possible paths to making quantum computing feasible. Third stop being a moron Martin Shrekli.
1 10

Martin Shkreli ✓ @MartinShkreli Sep 17
i invented this field and none of it works sorry
4 19

8:35 PM · Sep 17, 2025 · **104.8K** Views

Quantum Key Distribution

QKD vs PQC

Quantum Key Distribution

- Information theoretically secure
- Provably secure
- Requires hardware updates

Post Quantum Cryptography

- Computationally secure
  - Attacker is computationally bounded
- Hardness assumptions
  - (Module) lattices / SHA3 / AES
- Requires software update (mostly)

Both approaches share security assumptions

- Key authentication, side channels, supply chains

Combined QKD and PQC

Can we combine QKD and PQC keys...

- ...such that we have security if either implementation is secure?
- ...such that we maintain ITS if QKD is secure?

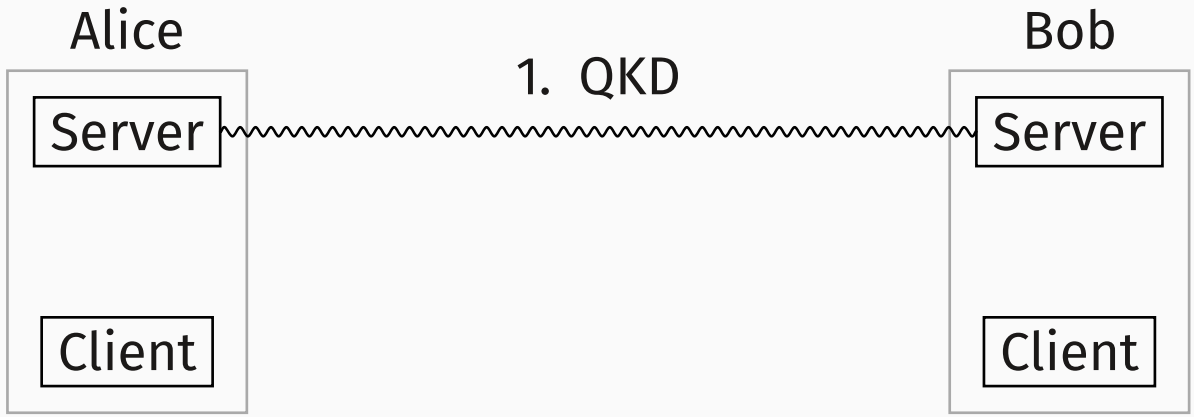
Related work

We identify gaps in the existing literature. Existing combiners...

- ...directly use KEM combiners
  - This is unsound: QKD is not a KEM
- ...combine keys using a cryptographic hash
  - This is only computationally secure, eliminating ITS property of QKD
- ...do not include the key ID in the protocol
  - We show that this leads to Dependent Key Attacks

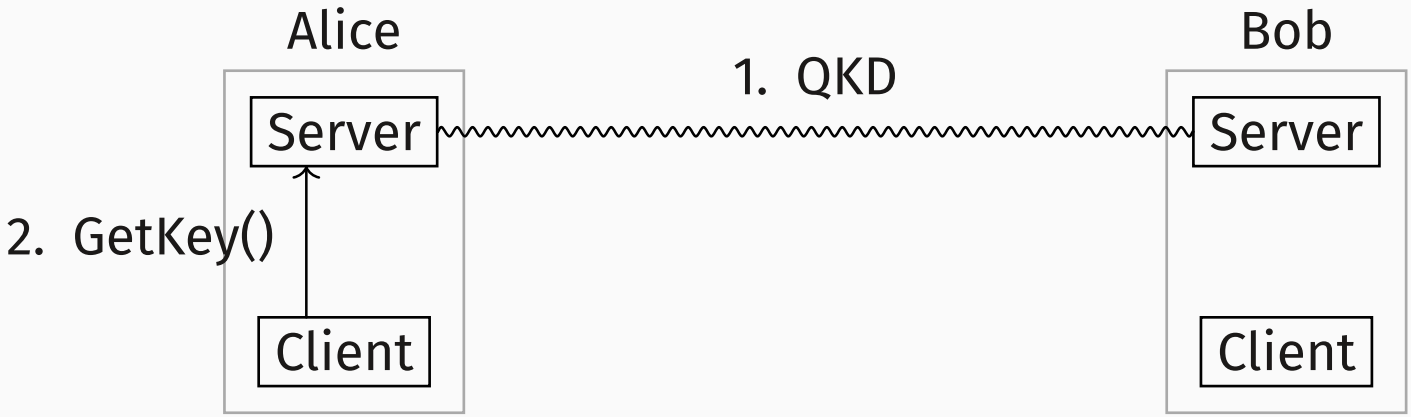
QKD interface: ETSI 014

Alice and Bob want shared key k



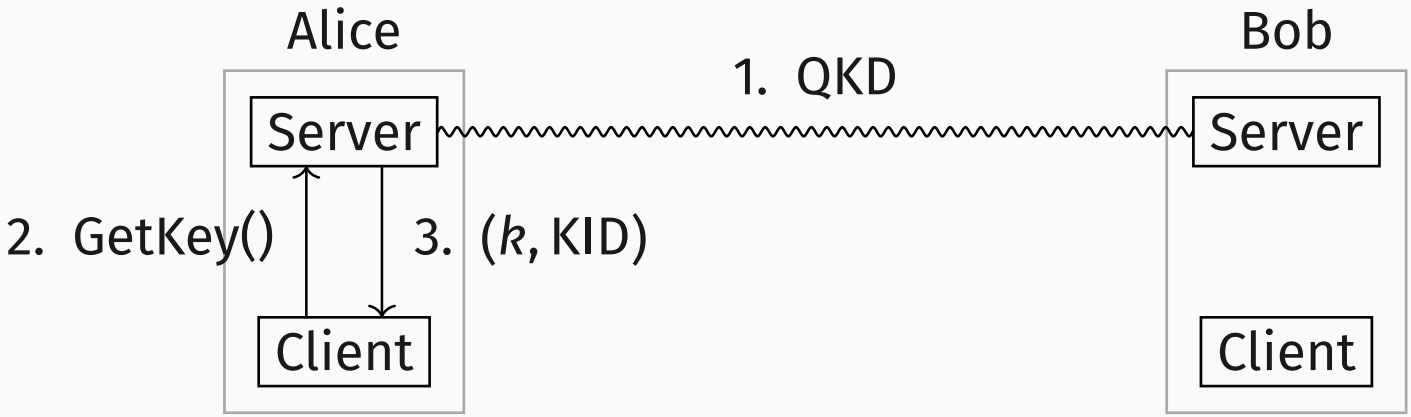
QKD interface: ETSI 014

Alice and Bob want shared key k



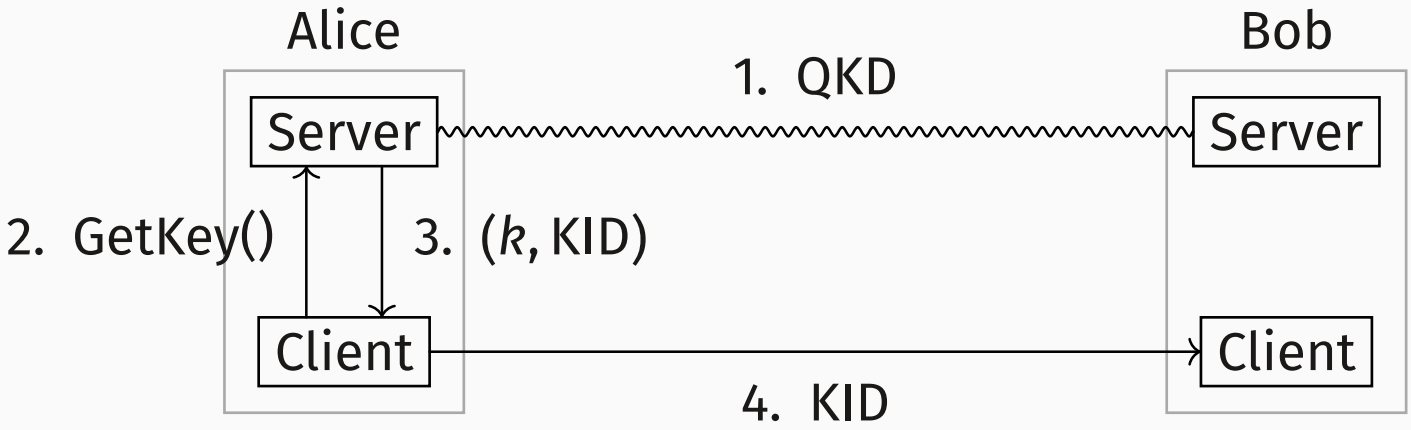
QKD interface: ETSI 014

Alice and Bob want shared key k



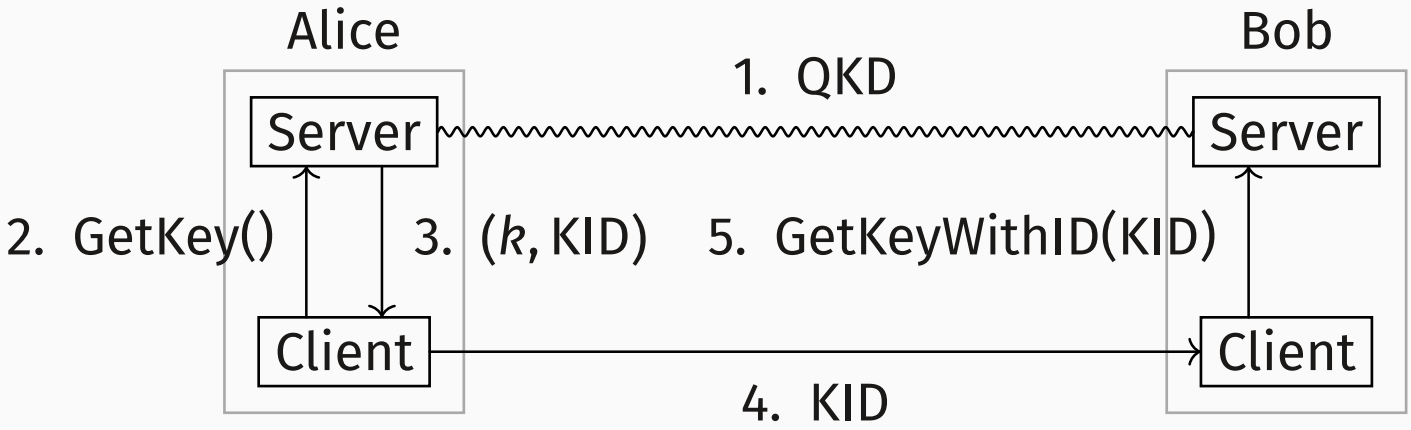
QKD interface: ETSI 014

Alice and Bob want shared key k



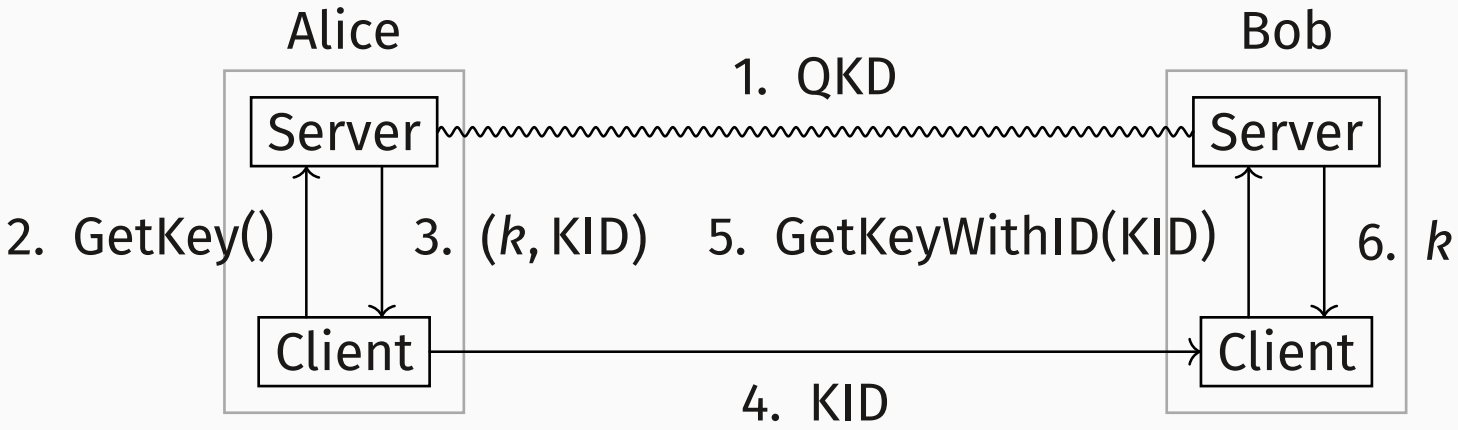
QKD interface: ETSI 014

Alice and Bob want shared key k



QKD interface: ETSI 014

Alice and Bob want shared key k



QKD interface: ETSI 014

Setting:

- E2E QKD (no relaying)
- Server/Client connection is assumed secure

Is the ETSI014 interface fundamental?

QKD interface: ETSI 014

Setting:

- E2E QKD (no relaying)
- Server/Client connection is assumed secure

Is the ETSI014 interface fundamental?

- No, but
  - It is what is used in practice
    - Finite size effects and low QKD key rates require QKD to run continuously
  - Modelling QKD directly adds (even more) complexity
    - In practice users don't have access to the QKD protocol transcript

Dependent Key Attack

Key Encapsulation Mechanism

A **KEM** is a tuple (KeyGen, Encaps, Decaps)

Key generation generates a *keypair*

$$(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}()$$

We *encapsulate* to a public key

$$(c, k) \xleftarrow{\$} \text{Encaps}(\text{pk})$$

to get a *ciphertext* and *key*

Decapsulation requires the secret key:

$$k \leftarrow \text{Decaps}(\text{sk}, c)$$

Key Encapsulation Mechanism

Active security: **Ind**istiguishability under **c**hosen-**c**iphertext **a**ttacks

Defined via a game between challenger and attacker (A)

GAME IND-CCA_b^A:

- $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}()$
- $(c, k_0) \xleftarrow{\$} \text{Encaps}(\text{pk})$
- $k_1 \xleftarrow{\$} \{0,1\}^\ell$
- $b' \leftarrow A^{\text{Dec}(\cdot)}(\text{pk}, c, k_b)$
- **return** b'

Oracle $\text{Dec}(c')$:

- **if** $c' = c$:
  - **return** $\perp$
- **return** $\text{Decaps}(\text{sk}, c')$

Key Encapsulation Mechanism

Active security: **Ind**istiguishability under **c**hosen-**c**iphertext **a**ttacks

Defined via a game between challenger and attacker (A)

GAME IND-CCA_b^A:

- $(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}()$
- $(c, k_0) \xleftarrow{\$} \text{Encaps}(\text{pk})$
- $k_1 \xleftarrow{\$} \{0,1\}^\ell$
- $b' \leftarrow A^{\text{Dec}(\cdot)}(\text{pk}, c, k_b)$
- **return** b'

Oracle $\text{Dec}(c')$:

- **if** $c' = c$:
  - **return** $\perp$
- **return** $\text{Decaps}(\text{sk}, c')$

No attacker should be able to distinguish IND-CCA_0 from IND-CCA_1

$$\text{Adv}(A) = \left|\Pr\left[\text{IND-CCA}_0^A = 1\right] - \Pr\left[\text{IND-CCA}_1^A = 1\right]\right|$$

Key Encapsulation Mechanism

Is this too strong?

- Historically CCA security was seen as a theoretical concern[citation needed]
- Daniel Bleichenbacher came with an attack in 1998 [Ble98]
  - Attack against RSA with PKCS#1 v1.5 padding
  - Send ciphertext to a server
  - Server reveals if the padding was invalid
    - side-channel leakage
  - Server acts as a decryption oracle
    - We may as well make that oracle as leaky as possible
- CCA security protects against the above

KEM Combiners

Can we combine KEM_1 and KEM_2 such that we strictly **add** security:

- the combination is secure even if only one component is secure?

Relevant for combining pre- and post-quantum KEMs

KEM Combiners

Can we combine KEM_1 and KEM_2 such that we strictly **add** security:

- the combination is secure even if only one component is secure?

Relevant for combining pre- and post-quantum KEMs

$$
\begin{array}{l}
\mathsf{Encaps}((\mathsf{pk}_1, \mathsf{pk}_2)): \\
\quad (c_1, k_1) \xleftarrow{\$} \mathsf{Encaps}_1(\mathsf{pk}_1) \\
\quad (c_2, k_2) \xleftarrow{\$} \mathsf{Encaps}_2(\mathsf{pk}_2) \\
\quad \textbf{return } ((c_1, c_2), k_1 \oplus k_2)
\end{array}
$$

Is this IND-CCA secure?

KEM Combiners

Mix-and-Match attack [Bin+19]

$$
\begin{aligned}
&A^{\mathsf{Dec}(\cdot)}\big((\mathsf{pk}_1, \mathsf{pk}_2), (c_1, c_2), k_b\big): \\
&\quad (c_1', k_1') \xleftarrow{\$} \mathsf{Encaps}_1(\mathsf{pk}_1) \\
&\quad (c_2', k_2') \xleftarrow{\$} \mathsf{Encaps}_2(\mathsf{pk}_2) \\
&\quad k_1^* \leftarrow \mathsf{Dec}\big((c_1', c_2)\big) \;//\; (c_1', c_2) \neq (c_1, c_2) \\
&\quad k_2^* \leftarrow \mathsf{Dec}\big((c_1, c_2')\big) \;//\; (c_1, c_2') \neq (c_1, c_2) \\
&\quad \textbf{return } k_b = k_1^* \oplus k_1' \oplus k_2^* \oplus k_2'
\end{aligned}
$$

KEM Combiners

Dual-PRF Combiner [Bin+19]

$$k \leftarrow \mathrm{PRF}\big(\mathrm{dualPRF}\big(k_1, k_2\big), \big(c_1, c_2\big)\big)$$

IND-CCA secure

- but relies on computational assumptions

KEM Combiners

XOR-then-MAC (XtM) Combiner [Bin+19]

For $i \in \{1, 2\}$:

$$\left(k_{\text{kem},i}, k_{\text{mac},i}\right) \leftarrow k_i$$

$$\tau_i \leftarrow \text{MAC}\left(k_{\text{mac},i}, \left(c_1, c_2\right)\right)$$

and then

$$c \leftarrow \left(c_1, c_2, \tau_1, \tau_2\right)$$

$$k \leftarrow k_{\text{kem},1} \oplus k_{\text{kem},2}$$

KEM Combiners

XOR-then-MAC (XtM) Combiner [Bin+19]

For $i \in \{1, 2\}$:

$$\left(k_{\text{kem},i}, k_{\text{mac},i}\right) \leftarrow k_i$$

$$\tau_i \leftarrow \text{MAC}\left(k_{\text{mac},i}, \left(c_1, c_2\right)\right)$$

and then

$$c \leftarrow \left(c_1, c_2, \tau_1, \tau_2\right)$$

$$k \leftarrow k_{\text{kem},1} \oplus k_{\text{kem},2}$$

But... we show that the proof for XtM is broken

- we don't know a MAC that is strong enough
- it's unclear if XtM is insecure or if its proof just needs repair

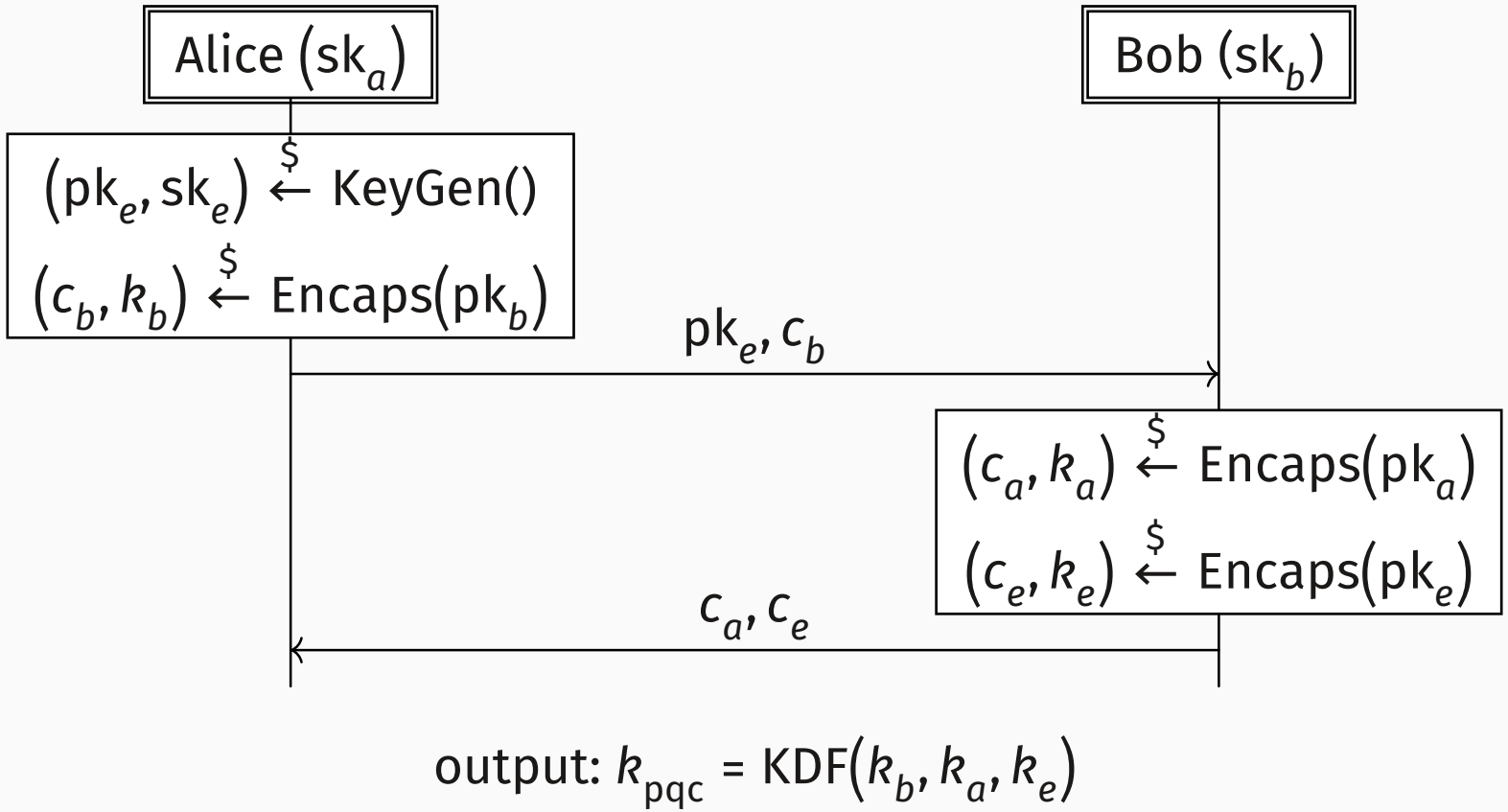
Authenticated Key Exchange

Authenticated Key Exchange

Establish a shared key between two parties

- parties know each others public key (*key authentication* is out of scope)
- or (in case of QKD) parties have a pre-shared symmetric key

AKE - Example: Triple KEM



output: $k_{\mathrm{pqc}} = \mathrm{KDF}(k_b, k_a, k_e)$

This is secure...*

Modelling AKE

Bellare-Rogaway 1993 [BR93]

- many variations
  - not always formally compatible
- we work with CK$^+$ [Kra05]

Security: **session key** should be indistinguishable from random

- even if the attacker tampers with messages
  - (the protocol may abort if it detects tampering)
- even if involved static keys are revealed (modelling forward secrecy)
  - *or* if local state is revealed (modelling partial device compromise)
- even if other session keys are revealed(**!**)

Modelling AKE

Bellare-Rogaway 1993 [BR93]

- many variations
  - not always formally compatible
- we work with CK$^+$ [Kra05]

Security: **session key** should be indistinguishable from random

- even if the attacker tampers with messages
  - (the protocol may abort if it detects tampering)
- even if involved static keys are revealed (modelling forward secrecy)
  - *or* if local state is revealed (modelling partial device compromise)
- even if other session keys are revealed(**!**)
  - for any session, except the **matching session**:
    - the session with the same protocol transcript

Modelling AKE

Execution model:

- Game between a challenger and an attacker
- Attacker *establishes* many **sessions** with an intended *peer*
  - Challenger maintains state per session
- Attacker can send protocol messages to a session
  - Challenger executes protocol on the input message and state
    - updates state
    - returns output message to the attacker
  - If a session accepts, it computes a **session key**
- Attacker selects a **test session**
  - Challenger reveals the session key ($b = 0$) or a random key ($b = 1$)
  - Attacker outputs b' and *wins* iff $b' = b$

Mixing in QKD

QKD oracle

- KIDs are generated with a global counter
- Honest party access
  - **GetKey**: SID $\mapsto \left(k_{\text{qkd}}, \text{KID}\right)$, where $k_{\text{qkd}} \xleftarrow{\$} \{0,1\}^{\ell}$
  - **GetKeyWithId**: (SID, KID) $\mapsto k_{\text{qkd}}$
    - oracle tracks calling SIDs
    - oracle checks that the SIDs are *peers*
    - oracle deletes k_{qkd}

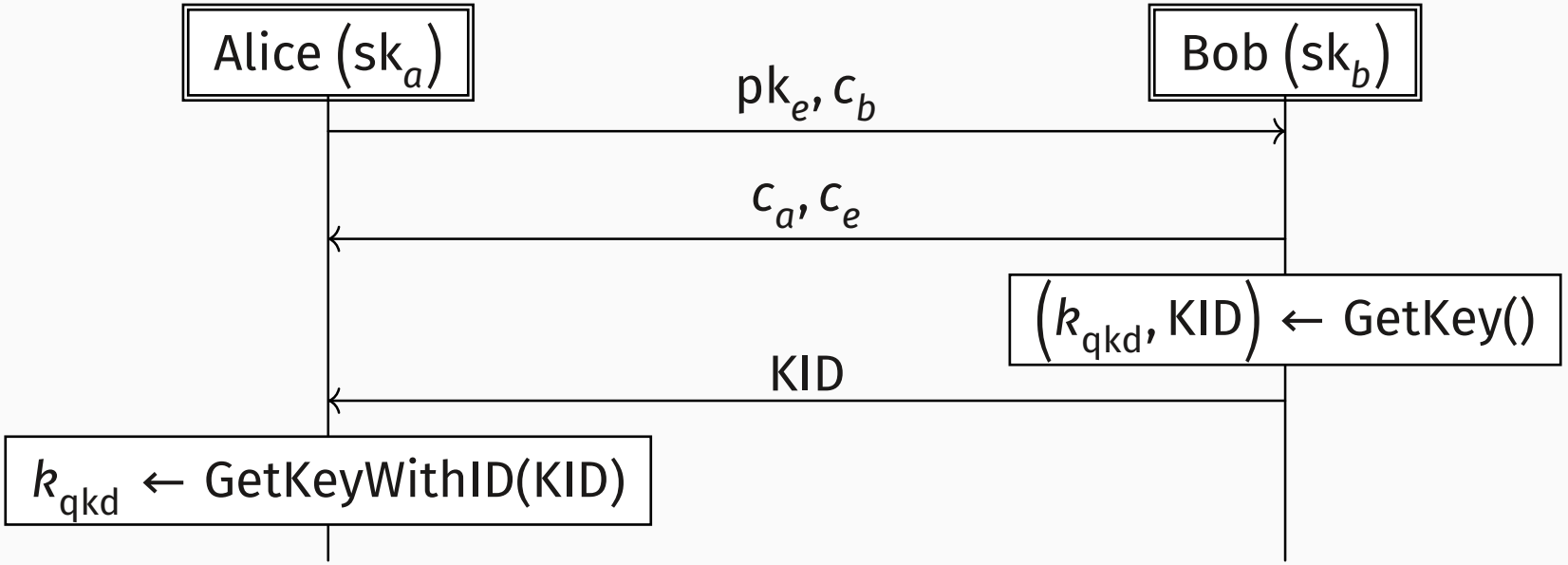
Mixing in QKD

QKD oracle

- KIDs are generated with a global counter
- Honest party access
  - **GetKey**: SID $\mapsto \left(k_{\text{qkd}}, \text{KID}\right)$, where $k_{\text{qkd}} \xleftarrow{\$} \{0,1\}^{\ell}$
  - **GetKeyWithId**: (SID, KID) $\mapsto k_{\text{qkd}}$
    - oracle tracks calling SIDs
    - oracle checks that the SIDs are *peers*
    - oracle deletes k_{qkd}
- Attacker access
  - **Leak**: KID $\mapsto k_{\text{qkd}}$
  - **Override**: $\left(\text{KID}, k'_{\text{qkd}}\right)$ sets $k_{\text{qkd}} \leftarrow k'_{\text{qkd}}$
    - oracle tracks which keys were leaked/overridden

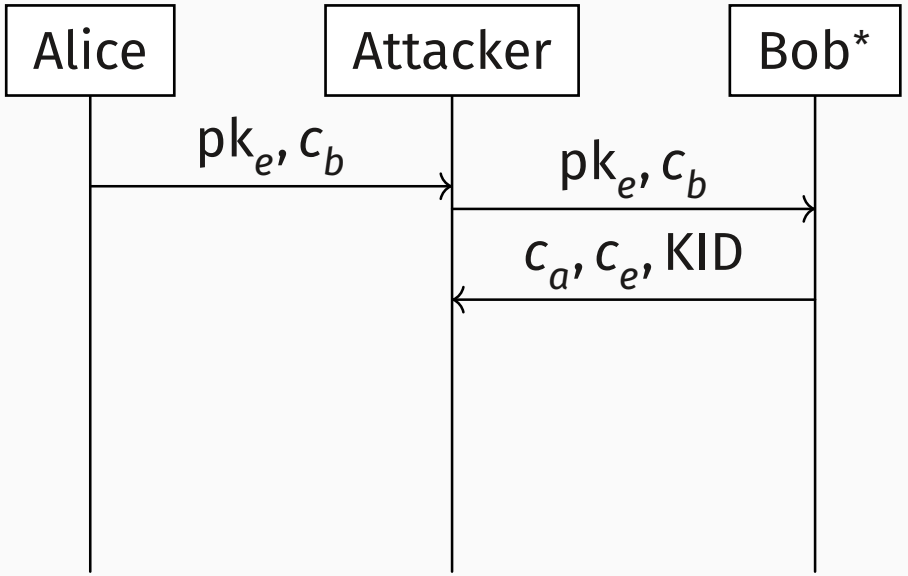
A QKD key can provide security if it was not leaked/overridden

Combined PQC and QKD protocol - Naive protocol

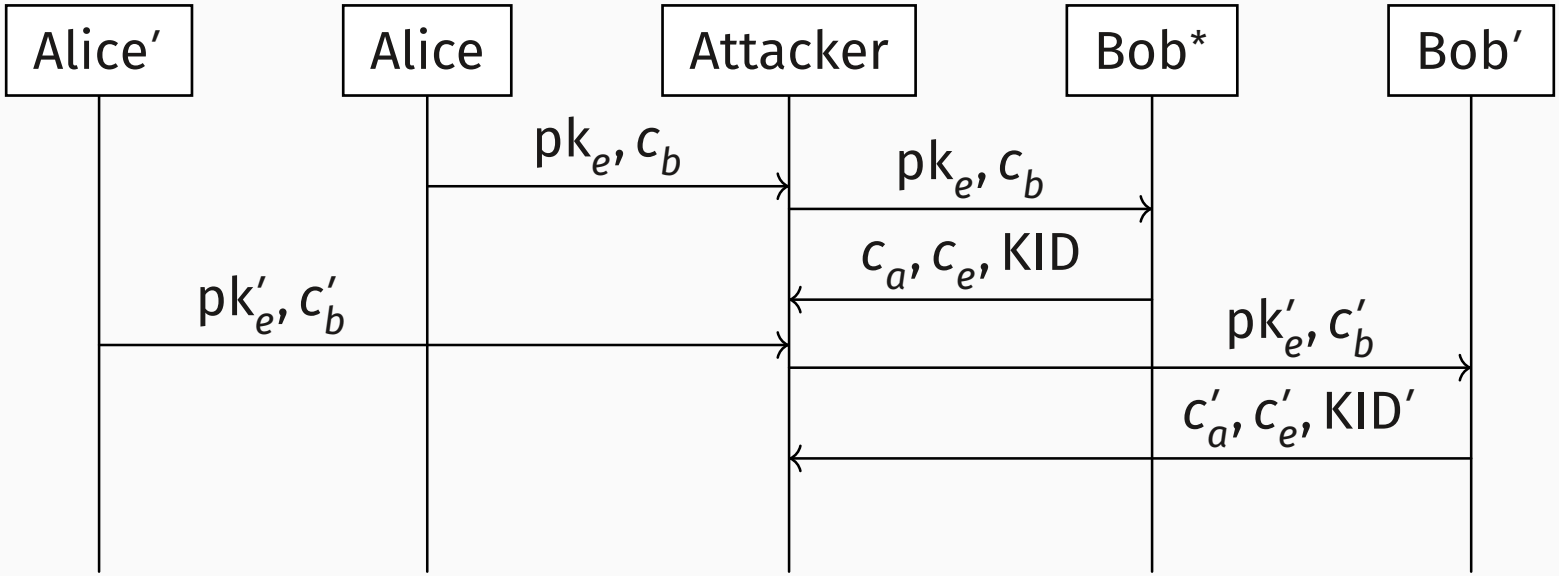


output: $k_{\mathrm{sess}} = k_{\mathrm{pqc}} \oplus k_{\mathrm{qkd}}$

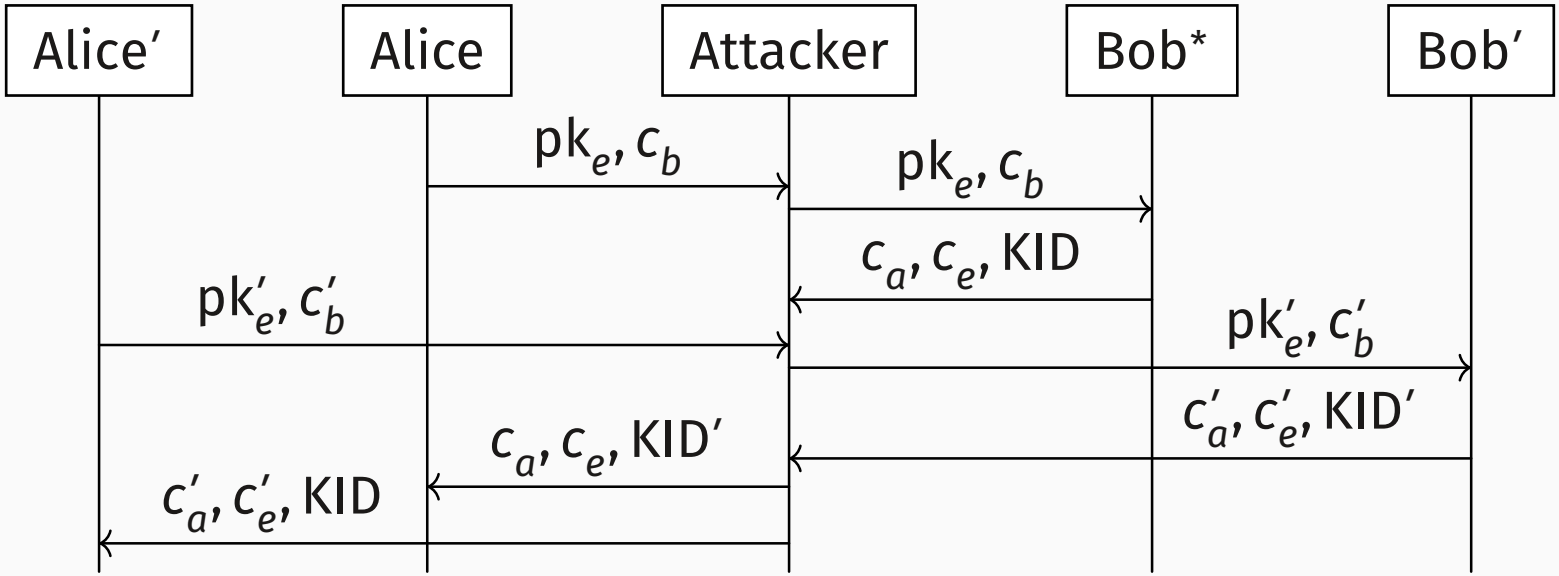
Combined PQC and QKD protocol - Naive protocol: Dependent Key Attack



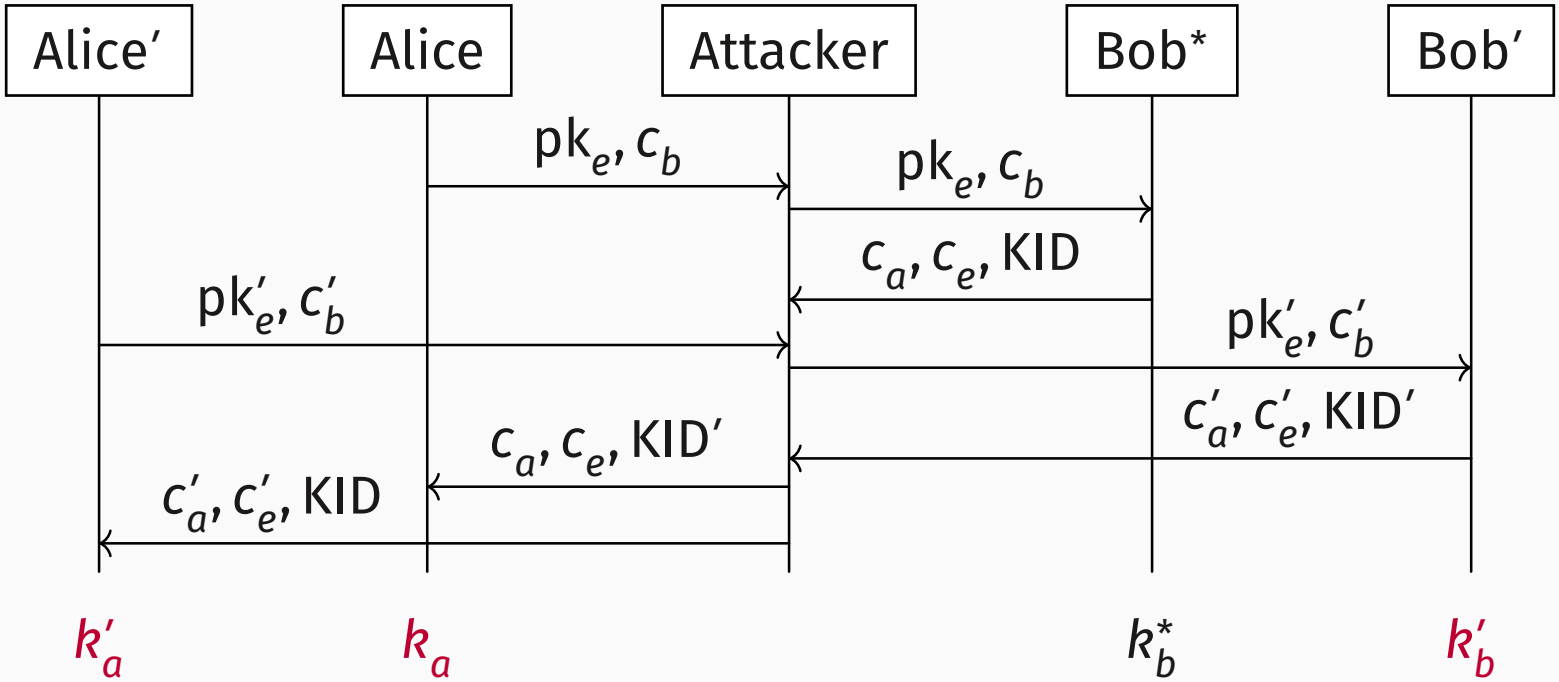
Combined PQC and QKD protocol - Naive protocol: Dependent Key Attack



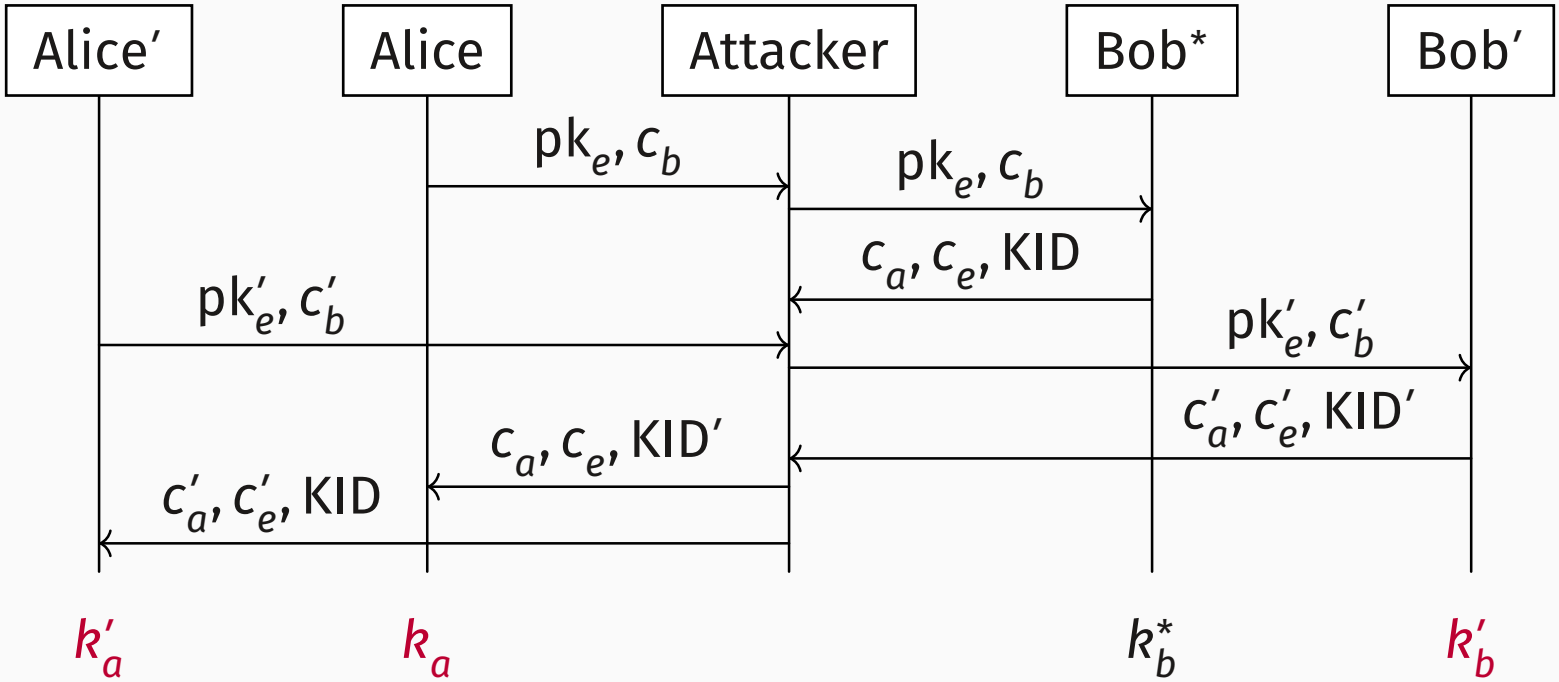
Combined PQC and QKD protocol - Naive protocol: Dependent Key Attack



Combined PQC and QKD protocol - Naive protocol: Dependent Key Attack



Combined PQC and QKD protocol - Naive protocol: Dependent Key Attack

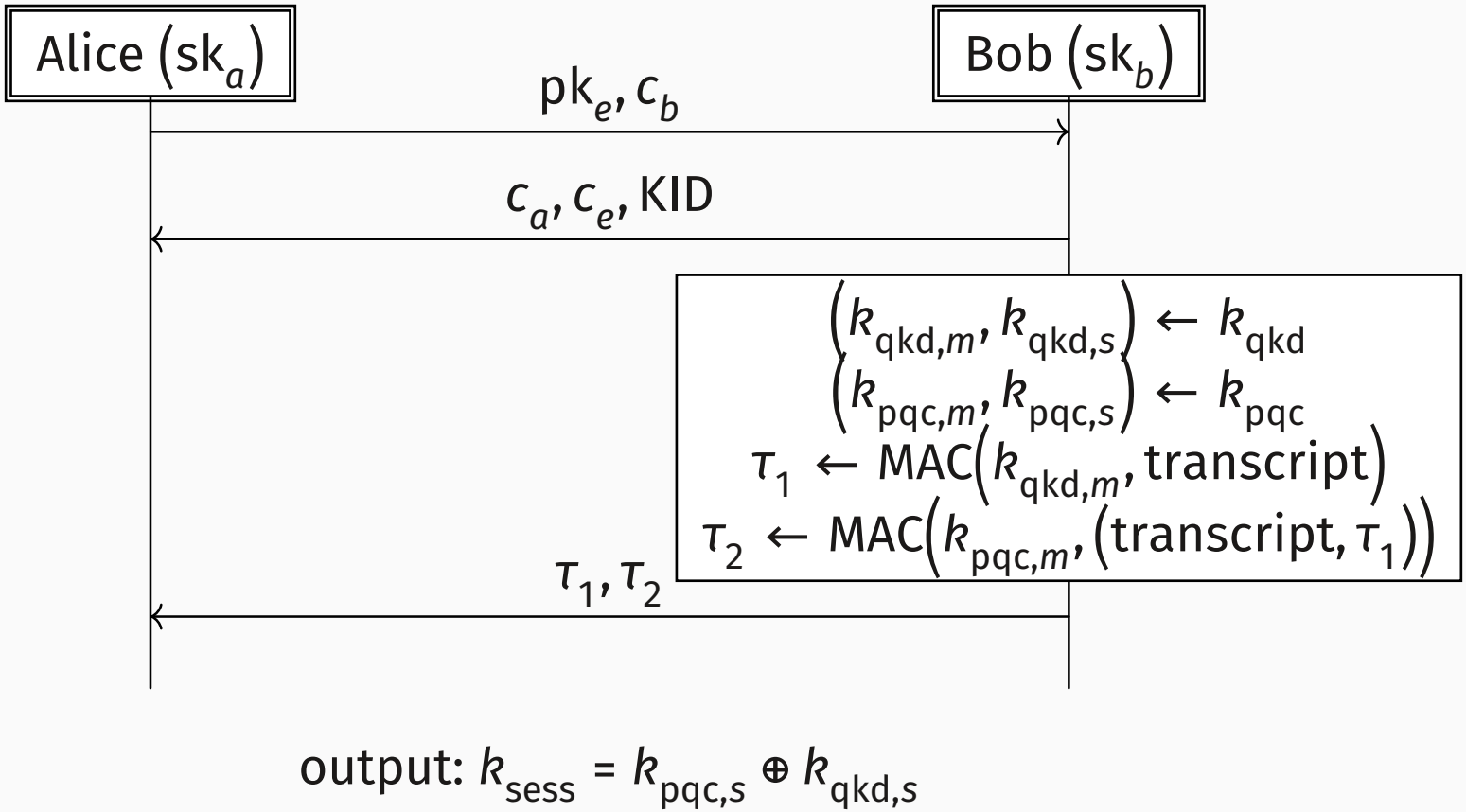


$$k^*_b = k'_a \oplus k_a \oplus k'_b$$

$$= \left(k'_{\mathrm{pqc}} \oplus k_{\mathrm{qkd}}\right) \oplus \left(k_{\mathrm{pqc}} \oplus k'_{\mathrm{qkd}}\right) \oplus \left(k'_{\mathrm{pqc}} \oplus k'_{\mathrm{qkd}}\right)$$

Our protocol

Combined PQC and QKD protocol - Our protocol



output: $k_{\mathrm{sess}} = k_{\mathrm{pqc},s} \oplus k_{\mathrm{qkd},s}$

Combined PQC and QKD protocol - Our protocol

Main theorems (very informal):

- If some PQC secrets are not revealed, our protocol is computationally secure...*
  - as secure as its components
  - proof in the QROM
- If the used QKD key is not leaked/overridden, our protocol is statistically secure
  - requires an ITS MAC (one-time MAC)
  - requires correctness of the KEMs(!)

Combined PQC and QKD protocol - Our protocol

Consider a session that uses the same k_{pqc} as the test session

- (we prove this session is unique)
- if it is a matching session, it may not be revealed
- otherwise, the MAC is invalid and Alice aborts

Similar for the session that uses the same k_{qkd} as the test session

The order of the MACs matters:

- with our oracle, the attacker can change k_{qkd} without changing KID
- the attacker cannot change an honestly generated k_{pqc} without changing the transcript

Conclusion

Conclusion

Combined cryptography is non-trivial

- Especially if we cannot rely on cryptographic hash functions

It is essential to include the QKD key ID

- In the protocol
- In the model (for example using our oracle)

Future work

- replace Triple KEM with a protocol resistant to state-reveal attacks
- replace perfectly random QKD keys with statistically secure ones
- model post-compromise security (the self-healing property)
- model relaying
- composability of the ETSI014 interface

Conclusion

Combined cryptography is non-trivial

- Especially if we cannot rely on cryptographic hash functions

It is essential to include the QKD key ID

- In the protocol
- In the model (for example using our oracle)

zeroknowledge.me/
talks/
#qkd-for-ake-tcs

Future work

- replace Triple KEM with a protocol resistant to state-reveal attacks
- replace perfectly random QKD keys with statistically secure ones
- model post-compromise security (the self-healing property)
- model relaying
- composability of the ETSI014 interface

Thank you for your attention

Bibliography

[Ble98] D. Bleichenbacher, “Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS,” in *CRYPTO98*, H. Krawczyk, Ed., Berlin, Heidelberg: Springer, 1998, pp. 1–12. doi: 10.1007/BFb0055716.

[Bin+19] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, “Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange,” in *Post-Quantum Cryptography*, Jintai Ding and Rainer Steinwandt, Eds., Cham: Springer International Publishing, Jul. 2019, pp. 206–226. doi: 10.1007/978-3-030-25510-7_12.

[BR93] M. Bellare and P. Rogaway, “Entity Authentication and Key Distribution,” in *CRYPTO93*, D. R. Stinson, Ed., Berlin, Heidelberg: Springer, 1993, pp. 232–249. doi: 10.1007/3-540-48329-2_21.

[Kra05] H. Krawczyk, “HMQC: A High-Performance Secure Diffie-Hellman Protocol,” in *CRYPTO05*, V. Shoup, Ed., Berlin, Heidelberg: Springer, 2005, pp. 546–566. doi: 10.1007/11535218_33.

Formalizing

```
QkdInit()
01  kID_ctr := 0
02  qKey := [ ]
03  qStatus := [ ]
04  qSentSid := [ ]
05  qRecvSid := [ ]

QKD-KEY-HOLDERS(kID)
06  return (qSentSid[kID], qRecvSid[kID])

QKD-LEAK(kID)
07  if qStatus[kID] = HONEST
08     qStatus[kID] := REVEALED
09  return qKey[kID]

QKD-OVERRIDE(kID, k)
10  qKey[kID] := k
11  qStatus[kID] := CORRUPT

QKD-GET-KEY(sID, ℓ)
12  kID_ctr := kID_ctr + 1
13  qSentSid[kID_ctr] := sID
14  if qKey[kID_ctr] = ⊥
15     qKey[kID_ctr] ←$ {0,1}^ℓ
16     qStatus[kID_ctr] := HONEST
17  k := qKey[kID_ctr]
18  return (kID_ctr, k)

QKD-GET-KEY-WITH-ID(sID, kID)
19  sID_qs := qSentSid[kID]
20  if (owner[sID_qs], peer[sID_qs]) ≠
                          (peer[sID], owner[sID])
21     return ⊥
22  if qKey[kID] = ⊥
23     return ⊥
24  qRecvSid[kID_ctr] += [sID]
25  k := qKey[kID]
26  qKey[kID] := ⊥
27  return k
```

Fig. 6: QKD oracle model. Honest sessions have access to the QKD-GET-KEY$(sID, \cdot)$ and QKD-GET-KEY-WITH-ID$(sID, \cdot)$ oracles (in purple), while the adversary may query QKD-KEY-HOLDERS, QKD-LEAK, and QKD-OVERRIDE (in red). The party calling QKD-GET-KEY specifies the key length ℓ. kID is a (predictable) key identifier. Besides the functionality, we add some bookkeeping values (in green) that are used in the proofs, but are not available to either the honest user or the adversary. Keys are removed in Line 26 from the oracle when they are requested by the receiver (the party calling QKD-GET-KEY-WITH-ID).

Formalizing

$\underline{\mathsf{Init}(pk_j)}$

01 $(c_b, k_b) \leftarrow \mathsf{Encaps}_{stat}(pk_j)$
02 $(pk_e, sk_e) \leftarrow \mathsf{KeyGen}_{eph}()$
03 $m_1 := (c_b, pk_e)$
04 $s' := (k_b, sk_e, m_1)$
05 **return** (m_1, s')

$\underline{\mathsf{SendM1}^{\text{QKD-GET-KEY}(sID,\cdot)}(i, sk_i, j, pk_j, m_1)}$

06 $(c_b, pk_e) := m_1$ // or abort
07 $k'_b := \mathsf{Decaps}_{stat}(sk_i, c_b)$
08 $(c_a, k_a) \leftarrow \mathsf{Encaps}_{stat}(pk_j)$
09 $(c_e, k_e) \leftarrow \mathsf{Encaps}_{eph}(pk_e)$
10 $k_{pqc} := \mathsf{KDF}(k'_b, k_a, k_e)$
11 $(kID, k_{qkd}) \leftarrow \text{QKD-GET-KEY}(sID, \ell_{\mathtt{qkd}})$
12 $t := (m_1, (c_a, c_e, kID))$
13 $(k_{sess}, \tau_1, \tau_2) := \mathsf{Combine}(j, i, t, k_{qkd}, k_{pqc})$
14 $m_2 := (c_a, c_e, kID, \tau_1, \tau_2)$
15 **return** (m_2, k_{sess})

$\underline{\mathsf{SendM2}^{\text{QKD-GET-KEY-WITH-ID}(sID,\cdot)}(i, sk_i, j, s, m_2)}$

16 $(c_a, c_e, kID, \tau'_1, \tau'_2) := m_2$ // or abort
17 $(k_b, sk_e, m_1) := s$ // or abort
18 $k'_a := \mathsf{Decaps}_{stat}(sk_i, c_a)$
19 $k'_e := \mathsf{Decaps}_{eph}(sk_e, c_e)$
20 $k_{pqc} := \mathsf{KDF}(k_b, k'_a, k'_e)$
21 $k_{qkd} :=$ QKD-GET-KEY-WITH-ID(sID, kID)
22 **if** $k_{qkd} = \perp$: **abort**
23 $t := (m_1, (c_a, c_e, kID))$
24 $(k_{sess}, \tau_1, \tau_2) := \mathsf{Combine}(i, j, t, k_{qkd}, k_{pqc})$
25 **if** $\tau_1 \neq \tau'_1$ **or** $\tau_2 \neq \tau'_2$: **abort**
26 **return** k_{sess}

$\underline{\mathsf{Combine}(i_I, i_R, t, k_{qkd}, k_{pqc})}$

27 $(k_{qkd,m} \parallel k_{qkd,s}) := k_{qkd}$
28 $(k_{pqc,m} \parallel k_{pqc,s}) := k_{pqc}$
29 $k_{sess} := k_{qkd,s} \oplus k_{pqc,s}$
30 $\tau_1 := \mathsf{MAC}^{(qkd)}_{k_{qkd,m}}((t, i_I, i_R))$
31 $\tau_2 := \mathsf{MAC}^{(pqc)}_{k_{pqc,m}}((t, \tau_1, i_I, i_R))$
32 **return** $(k_{sess}, \tau_1, \tau_2)$

Fig. 9: A formal implementation of our AKE protocol $\Pi = (\mathsf{Init}, \mathsf{SendM1}, \mathsf{SendM2})$, using the QKD oracle (QKD-GET-KEY, QKD-GET-KEY-WITH-ID), in the case of end-to-end QKD.

Theorem 5.2 (PQC-based security). *Let Π be the protocol of Fig. 9, where KEM_{eph} is δ_{eph}-correct and has key length ℓ_{eph}, and KEM_{stat} is δ_{stat}-correct, has collision probabilities $\mu(\mathsf{Encaps}_{stat})$ and $\mu(\mathsf{Secret}_{stat})$, and has key length ℓ_{stat}. Let A be an* IND-StAA-PQC *adversary executing Π with n_{pt} parties that establishes n_s sessions that makes at most q_H calls to the key-derivation function* KDF *(modelled as a QROM). Then there exist* IND-CPA *adversary B_1 against KEM_{eph},* IND-CPA *adversary B_2 against KEM_{stat},* IND-CCA *adversary B_3 against KEM_{stat}, and* OT-sEUF-CMA *adversary B_4 against $\mathsf{MAC}^{(pqc)}$ such that*

$$
\begin{aligned}
&\mathrm{Adv}^{\text{IND-StAA-PQC}}(\Pi, A) \leq \\
&\quad 2{n_s}^2 \left(\delta_{eph} + \mathrm{Adv}^{\text{IND-CPA}}_{\mathsf{KEM}_{eph}}(B_1) + 2q_H \frac{1}{\sqrt{2^{\ell_{eph}}}} \right) \\
&\quad + 2{n_s}^2 n_{pt} \left(\delta_{stat} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_{stat}}(B_2) + 2q_H \frac{1}{\sqrt{2^{\ell_{stat}}}} \right) + 2 \cdot n_s \cdot \mu(\mathsf{Encaps}_{stat}) \\
&\quad + 8 n_s n_{pt} \left(\delta_{stat} + \mathrm{Adv}^{\text{IND-CCA}}_{\mathsf{KEM}_{stat}}(B_3) + {n_s}^2 \mu(\mathsf{Secret}_{stat}) + n_s \frac{2q_H}{\sqrt{2^{\ell_{stat}}}} + n_s \mathrm{Adv}^{\text{OT-sEUF-CMA}}_{\mathsf{MAC}^{(pqc)}}(B_4) \right)
\end{aligned}
$$

where the runtime of B_1, B_2, B_3, B_4 is about that of A.

Theorem 5.5 (QKD-based security). *Let Π be the protocol of Fig. 9, where KEM_{eph} is δ_{eph}-correct and KEM_{stat} is δ_{stat}-correct. Let A be an* IND-AA-QKD *adversary executing Π that establishes n_s sessions. Then there exist and* OT-sEUF-CMA *adversary B_7 against $\mathsf{MAC}^{(qkd)}$ such that*

$$\mathrm{Adv}^{\text{IND-AA-QKD}}(\Pi, A) \leq 2n_s \left(n_s(2\delta_{stat} + \delta_{eph}) + \mathrm{Adv}^{\text{OT-sEUF-CMA}}_{\mathsf{MAC}^{(qkd)}}(B_7) \right)$$

where the runtime of B_7 is potentially unbounded.

Trivial

An attack is trivial if **no** protocol can protect against it:

- if the adversary reveals the session key directly
- if the adversary reveals both static key and state
  - (for both properties:) via the test session or the matching session
- if the adversary compromised the peer's static key and impersonated them

Exception: an attack is non-trivial it creates multiple matching sessions

- a proper AKE shares the session key with only one other session

Or in case of QKD:

- if all used QKD keys were leaked/overridden