In-band key-authentication from post-quantum key encapsulation mechanisms

Sebastian Verschoor

David R. Cheriton School of Computer Science
University of Waterloo

September 9th, 2021

UNIVERSITY OF
WATERLOO

Outline

Key authentication
- Usability
- Socialist Millionaire Protocol

Post-quantum solution
- Intuition
- Oblivious transfer
- Private equality confirmation

Proof of security
- Simple Universal Composability
- Post-quantum security

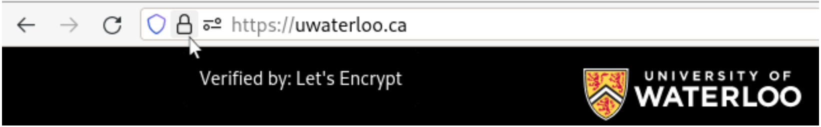
Implementation

Discussion

Key authentication

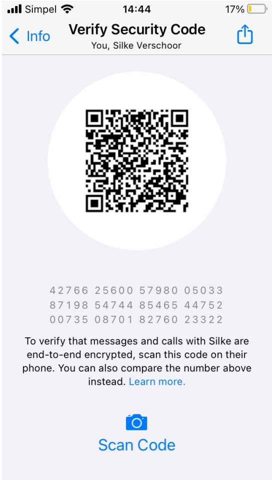
UNIVERSITY OF WATERLOO

- Secure messaging
  1. Trust establishment
     1.1 key exchange
     1.2 **key authentication**
  2. Conversation security
  3. Transport privacy
- Key authentication prevents Person-in-the-Middle attacks (and other impersonation attacks)



- TLS uses certificates
- We want something without a trusted third party

Manual key fingerprint verification



Silke Verschoor

✓ Verified

Your safety number with Silke Verschoor · 06 [redacted]

```
23212 20924 03635 03660
58522 28262 56010 36639
77483 78332 85453 56535
```

If you wish to verify the security of your end-to-end encryption with Silke Verschoor · 06 [redacted] compare the numbers above with the numbers on their device.

✓ You have verified your safety number with Silke Verschoor · 06 [redacted].

Mark as not verified

Key authentication: Usability

Usability issues lead to reduced security

- studies where only 13% of users are able to succesfully authenticate keys

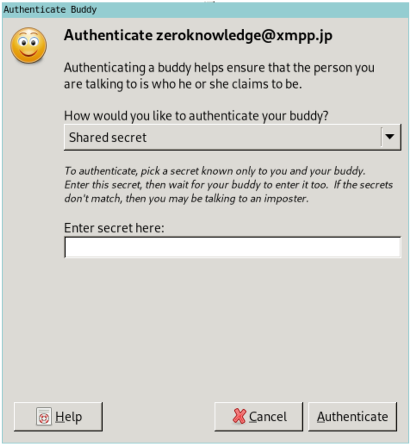
Observed problems with manual fingerprint comparison:

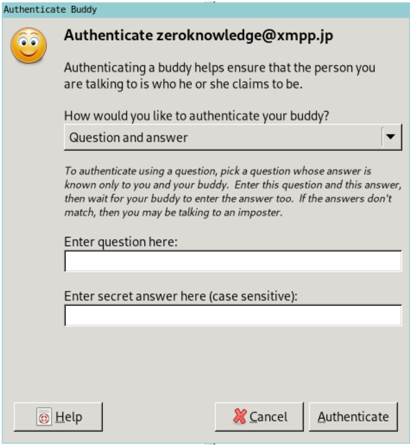
- compare fingerprints in-band (note that the share button lets you do this)
- compare only in one direction
- toggle “Mark as Verified” without actually verifying

Observed user behaviour:

- allowing in-band authentication increases usability
- users naturally rely on shared information

Secret-based Zero-Knowledge verification





Secret-based Zero-Knowledge verification

Implemented in OTR [AG07]

Two interfaces

- Shared secret (mutual authentication)
- Question/Answer

Pro's:

- In-band
- User sees no technical details (keys/fingerprints)

Con's:

- "Shared secrets require existing social relationships. This limits the usability of a system" [Ung+15]
- Synchronous

No user study to confirming improved usability

Private Equality Test (PET)

- Alice and Bob share a (low-entropy) secret *pwd*
- Alice and Bob have set up an OTR channel using pk_A and pk_B
- Alice computes $x = Hash(pk_A, pk_B, ssid, pwd)$
- Bob computes $y = Hash(pk_A, pk_B, ssid, pwd)$
- They run the SMP protocol over the OTR channel to compare if $x = y$ *in zero-knowledge*
  - If $x \neq y$, Alice should not learn *anything* about y (similarly Bob should not learn anything about x)

Socialist Millionaire Protocol

- Diffie-Hellman based protocol (not quantum-safe)
  - Shared secrets vulnerable to harvest-and-decrypt
- No direct translation to post-quantum primitives
- Fairness abandoned in the OTR implementation
  - One user can abort after getting output

Post-quantum solution

Proposed solution: KOP

- A (KEM-based Oblivious Transfer)-based Private Equality Confirmation

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \dots x_n$ (Alice) and $y = y_1 y_2 \dots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \dots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

UNIVERSITY OF WATERLOO

A solution using envelopes [FNW96]
Binary inputs $x = x_1 x_2 \ldots x_n$ (Alice) and $y = y_1 y_2 \ldots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \ldots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

Intuition

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \dots x_n$ (Alice) and $y = y_1 y_2 \dots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \dots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \ldots x_n$ (Alice) and $y = y_1 y_2 \ldots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \ldots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

Intuition

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \dots x_n$ (Alice) and $y = y_1 y_2 \dots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \dots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

Intuition

A solution using envelopes [FNW96]
Binary inputs $x = x_1 x_2 \dots x_n$ (Alice) and $y = y_1 y_2 \dots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \dots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

UNIVERSITY OF WATERLOO

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \ldots x_n$ (Alice) and $y = y_1 y_2 \ldots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \ldots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

A solution using envelopes [FNW96]

Binary inputs $x = x_1 x_2 \dots x_n$ (Alice) and $y = y_1 y_2 \dots y_n$ (Bob)

- Alice writes down n random pairs $(A_1[0], A_1[1]), \dots, (A_n[0], A_n[1])$
- Alice computes $\alpha(x) = A_1[x_1] \oplus \cdots \oplus A_n[x_n]$
- Bob learns $\alpha(y)$ as follows. Per pair:
  - Alice fills two envelopes, with $A_i[0]$ and $A_i[1]$
  - while Alice is not watching, Bob opens envelope $A_i[y_i]$
  - $A_i[1 - y_i]$ is destroyed
- Switch roles, so Alice learns $\beta(x)$
- They compare $\alpha(x) \oplus \beta(x)$ with $\alpha(y) \oplus \beta(y)$

Oblivious transfer

Envelopes are realized by Oblivious Transfer (OT)
Endemic 1-out-of-m OT (m envelopes)

- If both Sender and Receiver are honest:
  - Receiver input j
  - Let $s[1], \ldots, s[m]$ be random values
  - Receiver gets output $s[j]$
  - Sender gets output $s[1], \ldots, s[m]$
- Malicious parties choose their own output
  - Malicious Sender sets $s[1], \ldots, s[m]$
  - Malicious Receiver sets $s[j]$

OT from KEMs

- Key encapsulation mechanism (KEM):
  - $(pk, sk) \leftarrow KeyGen()$
  - $(k, ct) \leftarrow Encaps(pk)$
  - $k \leftarrow Decaps(sk, ct)$
- Public keys need to form a group $(\mathcal{G}, +)$
- Decapsulation *must not* fail explicitly
  - Nor leak (implicit) failure through side-channel
- m (local) random oracles $H_i : \mathcal{G}^{m-1} \rightarrow \mathcal{G}$

PQ KEMs have been under scrutiny by many cryptographers and can be instantiated as hybrid with pre-quantum primitives

OT from KEMs

Sender

Receiver (j)

Receiver:

$(pk, sk) \leftarrow \text{KeyGen}()$
for i in $\{1, \dots, m\} \setminus \{j\}$:
  $r_i \xleftarrow{\$} \mathcal{G}$
$r_j := pk - H_j((r_\ell)_{\ell \neq j})$

Receiver → Sender: $r_1, \dots, r_m$

Sender:

for i in $\{1, \dots, m\}$:
  $pk_i := r_i + H_i((r_\ell)_{\ell \neq i})$
  $(s[i], ct_i) \leftarrow \text{Enc}(pk_i)$

Sender → Receiver: $ct_1, \dots, ct_m$

Receiver:

$s[j] := \text{Dec}(sk, ct_j))$

OT construction from KEMs [MR21]

Output to both parties

The envelopes are only secure against semi-honest adversaries

- Simultaneous comparison (last step) is not possible
  - Bob can reflect Alice's last message to have her accept
  - Existing implementation [RR17]: only Bob gets output
- Use a cryptographic hash function G:
- Alice sends $G(\alpha(x)) \oplus \beta(x)$
- Bob rejects, or replies $\alpha(y) \oplus \beta(y)$

Output to both parties

Problem(?): Alice and/or Bob can send anything in the last message.

- A malicious party can force the other party to reject even when $x = y$
- Bob can even do this after having learned whether $x = y$
- In the context of key authentication this does not matter
- I call the resulting functionality Private Equality *Confirmation* (PEC)

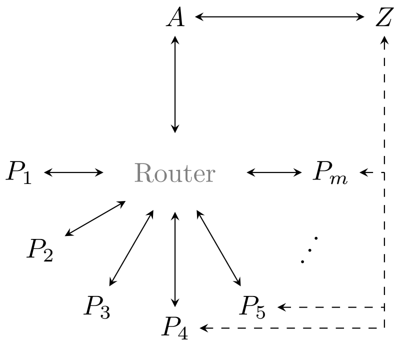
Output to both parties

Problem(?): Alice and/or Bob can send anything in the last message.

- A malicious party can force the other party to reject even when $x = y$
- Bob can even do this after having learned whether $x = y$
- In the context of key authentication this does not matter
- I call the resulting functionality Private Equality *Confirmation* (PEC)

Simple Universal Composability (SUC)

- Simulation paradigm (real/ideal)
- Environment Z
  - Wants to distinguish real model from ideal model
  - Chooses input and read outputs of parties P_i
  - Can corrupt parties
  - Interacts with the protocol (via the adversary interface)
- SUC-secure $\Leftrightarrow$ UC-secure
  - But SUC is less expressive than UC

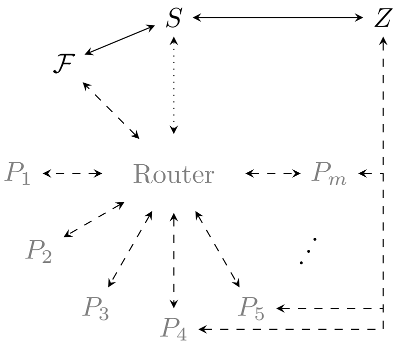


Real model (protocol π)

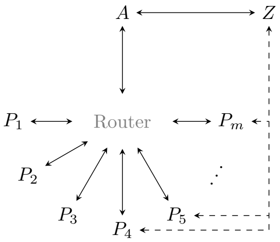
- Parties P_i send messages
  - Authenticated
  - Non-confidential
  - Scheduled by A
- Environment Z controls input/output
- Corrupt parties reveal state
- A can send messages for maliciously corrupted parties

Ideal model (functionality $\mathcal{F}$)

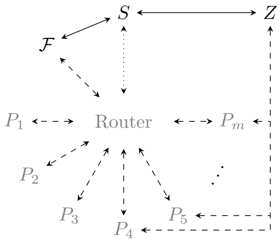
- Dummy parties P_i
  - Non-corrupted parties only forward input/output
  - Private messages
- Simulator S
  - Controls input/output of corrupted parties



UNIVERSITY OF WATERLOO



Z output bit
$\textsc{suc-real}_{\pi,A,Z}(1^\lambda, z)$



Z output bit
$\textsc{suc-ideal}_{\mathcal{F},S,Z}(1^\lambda, z)$

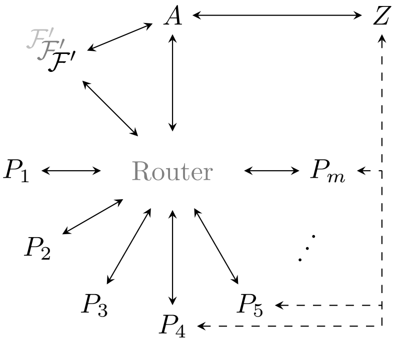
SUC-security: For every adversary A there must be a S such that for all environments Z on any advice z:

$$\left| \Pr[\textsc{suc-real} = 1] - \Pr[\textsc{suc-ideal} = 1] \right| = \mathit{negl}(\lambda)$$

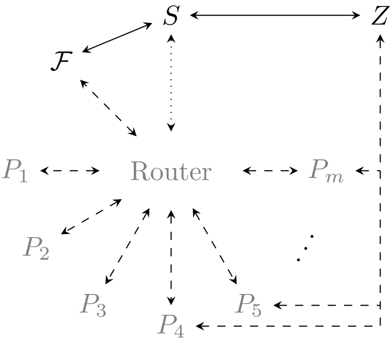
Simple Universal Composability

- Simulator S
  - Goal: generate identically distributed view for Z
  - S^A: defined relative to A
  - Z is external to S: no rewinding
  - S has to extract the effective input of the corrupted party to $\mathcal{F}$
  - Can run code of honest parties itself
  - Can see output of corrupted parties
- Hard to prove anything in this plain model
  - Replace the real model with a hybrid model

Simple Universal Composability



Hybrid model



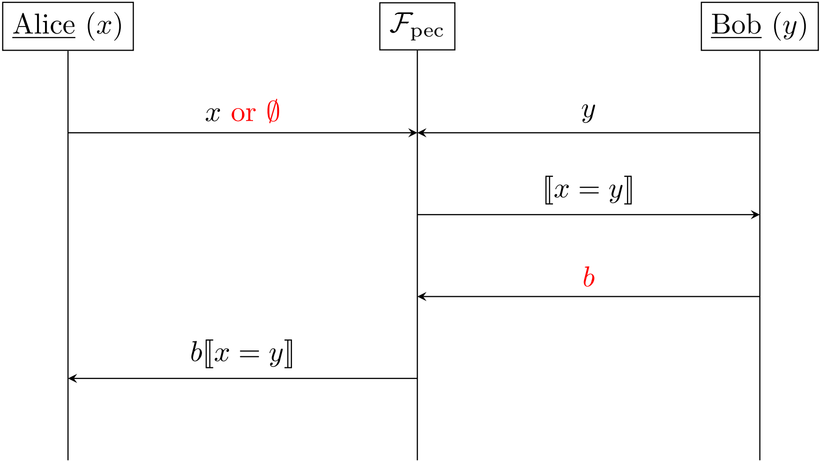
Ideal model

Simple Universal Composability

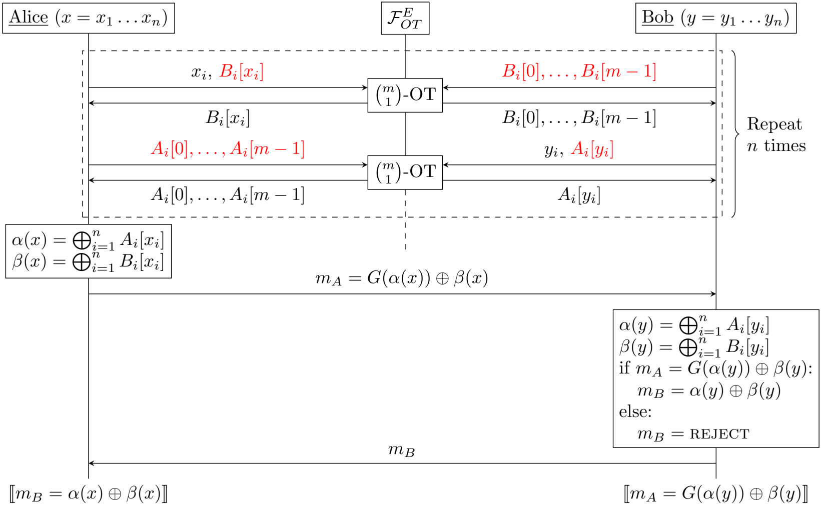
Hybrid model: protocol π uses functionality $\mathcal{F}'$

- SUC composition theorem:
  if π SUC-secure computes $\mathcal{F}$ in the $\mathcal{F}'$-hybrid model,
  and ρ SUC-secure computes $\mathcal{F}'$ in the $\mathcal{F}''$-hybrid model,
  then π^ρ SUC-secure computes $\mathcal{F}$ in the $\mathcal{F}''$-hybrid model
  - π^ρ: replace each invocation of $\mathcal{F}'$ by executing ρ
- S usually runs $\mathcal{F}'$ in the simulation
  - Can see adversary input
  - Can choose output (distributed similarly)
- Rarely go all the way to real model
  - In this case: the random oracle model is the lowest hybrid

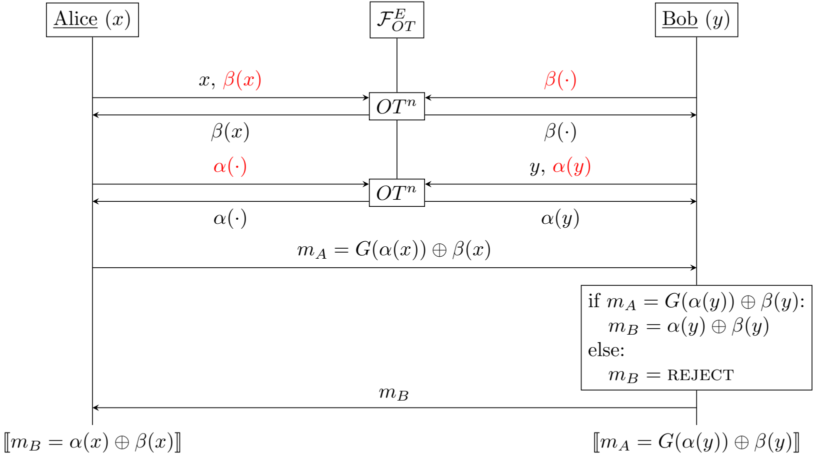
PEC functionality



PEC protocol



PEC protocol (simplified)

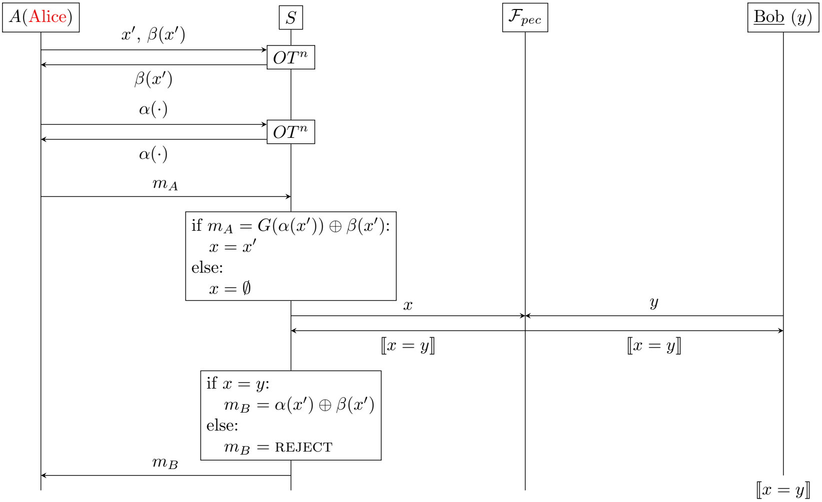


SUC security of PEC

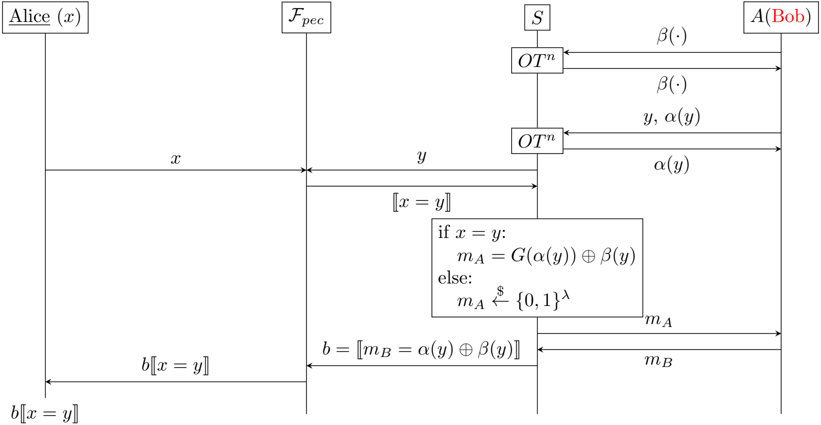
Hybrid argument to prove indistinguishability

- Start with a simulator that simply runs the honest party's code
  - trivially identical view for Z
  - invalid: requires knowledge of y
  - change it until it no longer requires y (but it will need $\mathcal{F}_{pec}$)
  - show each change is indistinguishable
- Last hybrid is a valid simulator

SUC security of PEC (corrupt Alice)



SUC security of PEC (corrupt Bob)



SUC security of PEC (corrupt Bob)

Two computational assumptions (in case $x \neq y$)

- random m_A should be indistinguishable from $G(\alpha(x)) \oplus \beta(x)$
  - note that $\alpha(x)$ is uniformly random
  - so this reduces to "G is pseudorandom"
- ideal model always rejects when $x \neq y$, real model might accept
  - real Alice sends $m_A = G(\alpha(x)) \oplus \beta(x)$
  - real Alice accepts $m_B = \alpha(x) \oplus \beta(x)$
  - so this reduces to "G is one-way"

Post-quantum security

- Post-quantum security
  - Environment is a quantum machine (with quantum advice)
  - Assume a PQ-secure OT
  - Assume a PQ-secure G (PQ one-way, PQ pseudorandom)
- The security argument can be lifted to quantum security
  - No internal rewinding
  - Lifting does not necessarily preserve tightness
    - but the proof was asymptotic and non-uniform anyway

Implementation

libkop

- Hybrid KEM
  - Kyber (Round 3 CCA, NIST PQC lvl 5)
  - ECDH (Ed448 Goldilocks, Decaf)
    - with implicit failure on parsing error
- C99 ($\sim$2000 LoC)
- Side channel protection
  - Constant time
  - No secret indices
- Domain separation ROMs

2-RTT protocol, 80-bit inputs ($m = 4$, $n = 40$)

- Message size
  - 254 KiB
  - 508 KiB
  - 254 KiB
  - 32 B
- Speed[1] (ms)
  - 22
  - 114
  - 106
  - 15

[1] measured without TurboBoost

Discussion

Key authentication from post-quantum KEMs (+ group structure)

Limitations

- OT security argument (despite claims) is not proven quantum-safe
  - any Post-Quantum UC-secure OT suffices
- Asymptotic, non-uniform proof
- Rather heavy machinery

Alternate solutions

- Use alternative key authentication ceremony
- Direct post-quantum replacement for SMP
- PAKE

Thank you

[AG07] Chris Alexander and Ian Goldberg. "Improved User Authentication in Off-the-Record Messaging". In: *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*. WPES '07. Alexandria, Virginia, USA: Association for Computing Machinery, Oct. 2007, pp. 41–47. ISBN: 9781595938831. DOI: 10.1145/1314333.1314340.

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. "Comparing Information Without Leaking It". In: *Communications of the ACM* 39.5 (1996), pp. 77–85. DOI: 10.1145/229459.229469.

[HSS11] Sean Hallgren, Adam Smith, and Fang Song. "Classical Cryptographic Protocols in a Quantum World". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Berlin, Heidelberg: Springer, 2011, pp. 411–428. ISBN: 978-3-642-22792-9. DOI: 10.1007/978-3-642-22792-9_23.

[MR21] Daniel Masny and Peter Rindal. *Endemic Oblivious Transfer*. July 2021. iacr: 2019/706.

References (cont.)

[RR17] Peter Rindal and Mike Rosulek. "Malicious-Secure Private Set Intersection via Dual Execution". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Ed. by Bhavani M. Thuraisingham et al. ACM, 2017, pp. 1229–1242. DOI: 10.1145/3133956.3134044.

[Ung+15] Nik Unger et al. "SoK: Secure Messaging". In: *2015 IEEE Symposium on Security and Privacy, SP 2015*. IEEE Computer Society, 2015, pp. 232–249. DOI: 10.1109/SP.2015.22.

Socialist Millionaire Protocol

Alice (x)		Bob (y)
$a_2, a_3 \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$		$b_2, b_3 \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q$
	g^{a_2}, g^{a_3} →	
	← g^{b_2}, g^{b_3}	
$g_2, g_3 := (g^{b_2})^{a_2}, (g^{b_3})^{a_3}$; $s \xleftarrow{\$} \mathbb{Z}_q$		$g_2, g_3 := (g^{a_2})^{b_2}, (g^{a_3})^{b_3}$; $r \xleftarrow{\$} \mathbb{Z}_q$
	$P_a = g_3^s, Q_a = g^s g_2^x$ →	
	← $P_b = g_3^r, Q_b = g^r g_2^y$	
$R_a := (Q_a/Q_b)^{a_3}$		$R_b := (Q_a/Q_b)^{b_3}$
	R_a →	
	← R_b	
$R_b^{a_3} = P_a/P_b$		$R_a^{b_3} = P_a/P_b$
$[\![x = y]\!]$		$[\![x = y]\!]$

Quantum Lifting

- A simple hybrid argument [HSS11]:
  For every adjacent hybrid H_i, H_{i+1}:
  - there is a machine M and classical distributions D_i, D_{i+1}
  - so that $M(D_i) = H_i$ and $M(D_{i+1}) = H_{i+1}$
  - and D_i is quantum indistinguishable from D_{i+1}