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
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
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)
Certificates

- TLS uses certificates
- We want something without a trusted third party
Manual key fingerprint verification

S. R. Verschoor

Key-authentication from KEMs

2021–09–09
Manual key fingerprint verification (cont.)

Silke Verschoor 📞
✓ Verified

Your safety number with Silke Verschoor · 06 [REDACTED]:

<table>
<thead>
<tr>
<th>23212</th>
<th>20924</th>
<th>03635</th>
<th>03660</th>
</tr>
</thead>
<tbody>
<tr>
<td>58522</td>
<td>28262</td>
<td>56010</td>
<td>36639</td>
</tr>
<tr>
<td>77483</td>
<td>78332</td>
<td>85453</td>
<td>56535</td>
</tr>
</tbody>
</table>

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
Usability issues lead to reduced security

- studies where only 13% of users are able to successfully 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
Authenticate zeroknowledge@xmpp.jp

Authenticating a buddy helps ensure that the person you are talking to is who he or she claims to be.

How would you like to authenticate your buddy?
- Shared secret
- Question and answer

To authenticate, pick a secret known only to you and your buddy. Enter this secret, then wait for your buddy to enter it too. If the secrets don’t match, then you may be talking to an imposter.

Enter secret here:

Enter question here:

Enter secret answer here (case sensitive):

Help  Cancel  Authenticate
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 \(x = \text{Hash}(pk_A, pk_B, ssid, pwd)\)
- Bob computes \(y = \text{Hash}(pk_A, pk_B, ssid, pwd)\)
- They run the SMP protocol over the OTR channel to compare if \(x = y\) \textit{in zero-knowledge}
  
  - If \(x \neq y\), Alice should not learn \textit{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
Proposed solution: KOP

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

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 $x = x_1 x_2 \ldots x_n$ (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 $x = x_1 x_2 \ldots x_n$ (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 $x = x_1x_2\ldots x_n$ (Alice) and $y = y_1y_2\ldots y_n$ (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 $x = x_1 x_2 \ldots x_n$ (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 $x = x_1 x_2 \ldots x_n$ (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 $x = x_1x_2\ldots x_n$ (Alice) and $y = y_1y_2\ldots y_n$ (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 $x = x_1x_2 \ldots x_n$ (Alice) and $y = y_1y_2 \ldots y_n$ (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 $\alpha(y) \oplus \beta(y)$
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 $s[j]$
Key encapsulation mechanism (KEM):

- \((pk, sk) \leftarrow \text{KeyGen}()\)
- \((k, ct) \leftarrow \text{Encaps}(pk)\)
- \(k \leftarrow \text{Decaps}(sk, ct)\)

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

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

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

\( ct_1, \ldots, ct_m \)

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

OT construction from KEMs [MR21]
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 $\alpha(y) \oplus \beta(y)$
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)
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
Simple Universal Composability

Real model (protocol $\pi$)
- 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

S. R. Verschoor
Key-authentication from KEMs
2021–09–09
Simple Universal Composability

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
Simple Universal Composability

\[ \text{Z output bit} \]
\[ \text{SUC-REAL}_{\pi, A, Z}(1^\lambda, z) \]

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

\[
\left| \Pr[\text{SUC-REAL} = 1] - \Pr[\text{SUC-IDEAL} = 1] \right| = \text{negl}(\lambda)
\]
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**

- $A \leftrightarrow P_1 \leftrightarrow P_2 \leftrightarrow P_3 \leftrightarrow P_4 \leftrightarrow P_5 \leftrightarrow \ldots \leftrightarrow P_m \leftrightarrow Z$
- $\mathcal{F} \mathcal{F}' \mathcal{F}'$"}

**Ideal model**

- $S \leftrightarrow P_1 \leftrightarrow P_2 \leftrightarrow P_3 \leftrightarrow P_4 \leftrightarrow P_5 \leftrightarrow \ldots \leftrightarrow P_m \leftrightarrow Z$
- $\mathcal{F}$

S. R. Verschoor

Key-authentication from KEMs

2021–09–09
Simple Universal Composability

Hybrid model: protocol \( \pi \) uses functionality \( \mathcal{F}' \)

- **SUC composition theorem:**
  - if \( \pi \) SUC-secure computes \( \mathcal{F} \) in the \( \mathcal{F}' \)-hybrid model, and \( \rho \) SUC-secure computes \( \mathcal{F}' \) in the \( \mathcal{F}'' \)-hybrid model, then \( \pi^\rho \) SUC-secure computes \( \mathcal{F} \) in the \( \mathcal{F}'' \)-hybrid model
  - \( \pi^\rho \): replace each invocation of \( \mathcal{F}' \) by executing \( \rho \)

- **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

\[ \text{Alice} \ (x) \quad \xrightarrow{x \text{ or } \emptyset} \quad \mathcal{F}_{\text{pec}} \quad \xrightarrow{y} \quad \text{Bob} \ (y) \]

\[ b \llbracket x = y \rrbracket \]

\[ b \llbracket x = y \rrbracket \]
PEC protocol

Alice \((x = x_1 \ldots x_n)\)

\[ \mathcal{F}_{OT} \]

Bob \((y = y_1 \ldots y_n)\)

Repeat \(n\) times

\[ m_A = G(\alpha(x)) \oplus \beta(x) \]

\[ m_B = \alpha(x) \oplus \beta(x) \]

\[ m_B = \alpha(x) \oplus \beta(x) \]

\[ m_A = G(\alpha(y)) \oplus \beta(y) \]

\[ m_B = \alpha(y) \oplus \beta(y) \]

else:

\[ m_B = \text{REJECT} \]

\[ m_B = \alpha(y) \oplus \beta(y) \]

if \( m_A = G(\alpha(y)) \oplus \beta(y) \):

\[ m_B = \alpha(y) \oplus \beta(y) \]

else:

\[ m_B = \text{REJECT} \]

\[ m_A = G(\alpha(y)) \oplus \beta(y) \]

\[ m_B = \alpha(y) \oplus \beta(y) \]
PEC protocol (simplified)

\[
\begin{align*}
    \text{Alice } (x) & \quad \text{FE} \quad \mathcal{F}_{OT} \quad \text{Bob } (y) \\
    x, \beta(x) & \quad \beta(\cdot) \\
    \beta(x) & \quad \alpha(\cdot) \\
    \alpha(\cdot) & \quad \alpha(y) \\
    m_A = G(\alpha(x)) \oplus \beta(x) & \\
    m_B & \\
    \text{if } m_A = G(\alpha(y)) \oplus \beta(y): \quad m_B = \alpha(y) \oplus \beta(y) \\
    \text{else:} \quad m_B = \text{REJECT} \\
    \text{[} m_B = \alpha(x) \oplus \beta(x) \text{]} & \quad \text{[} m_A = G(\alpha(y)) \oplus \beta(y) \text{]} 
\end{align*}
\]
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 $F_{pec}$)
  - show each change is indistinguishable

- Last hybrid is a valid simulator
SUC security of PEC (corrupt Alice)

\[ A(Alice) \xrightarrow{x', \beta(x')} S \xrightarrow{\beta(x')} F_{pec} \xrightarrow{\alpha(\cdot)} m_A = G(\alpha(x')) \oplus \beta(x') : x = x' \]

\[ m_B = \alpha(x') \oplus \beta(x') \]

\[ \text{else:} \]

\[ m_B = \text{REJECT} \]

\[ y \]

\[ x \]

\[ [x = y] \]
SUC security of PEC (corrupt Bob)

Alice (x) \[ \mathcal{F}_{pec} \] S

\( OT^n \) \( \beta(\cdot) \)

y, \( \alpha(y) \)

\( OT^n \) \( \alpha(y) \)

\[ [x = y] \]

if \( x = y \):

\[ m_A = G(\alpha(y)) \oplus \beta(y) \]

else:

\[ m_A \leftarrow \{0,1\}^\lambda \]

\[ m_A \]

\[ b = [m_B = \alpha(y) \oplus \beta(y)] \]

\[ b \]

b[\( x = y \)]

b[\( x = y \)]

A(Bob)

\( \mathcal{F}_{pec} \)

S

\( OT^n \) \( \beta(\cdot) \)

\( \beta(\cdot) \)

\( y, \alpha(y) \)

\( OT^n \) \( \alpha(y) \)

b[\( x = y \)]
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

- 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
libkop

- Hybrid KEM
  - Kyber (Round 3 CCA, NIST PQC lvl 5)
  - ECDH (Ed448 Goldilocks, Decaf)
    - with implicit failure on parsing error
- C99 (~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
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
References


Socialist Millionaire Protocol

Alice (x)

\(a_2, a_3 \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q\)

\(g_2, g_3 := (g_{b_2})^{a_2}, (g_{b_3})^{a_3}\)

\(s \leftarrow \mathbb{Z}_q\)

\(R_a := (Q_a/Q_b)^{a_3}\)

\(R_{a_3} = P_a/P_b\)

\([x = y]\)

Bob (y)

\(b_2, b_3 \leftarrow \mathbb{Z}_q \times \mathbb{Z}_q\)

\(g_{b_2}, g_{b_3}\)

\(g_2, g_3 := (g^{a_2})^{b_2}, (g^{a_3})^{b_3}\)

\(r \leftarrow \mathbb{Z}_q\)

\(P_a = g_3^s, Q_a = g^s g_2^x\)

\(P_b = g_3^r, Q_b = g^r g_2^y\)

\(R_b := (Q_a/Q_b)^{b_3}\)

\(R_{b_3} = P_a/P_b\)

\([x = y]\)
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 $D_{i+1}$