Post-Quantum EPID Signatures from Symmetric Primitives

Dan Boneh    Saba Eskandarian    Ben Fisch
Hardware Enclaves

A trusted component in an untrusted system

- Protected memory isolates enclave from compromised OS

Untrusted System

Enclave
- Data
- Secrets

Adversary who controls OS still can’t see inside enclave
Hardware Enclaves

A trusted component in an untrusted system

- Protected memory isolates enclave from compromised OS
- Proves authenticity via a process called *attestation*

Untrusted System

<table>
<thead>
<tr>
<th>Enclave</th>
<th>Secure Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Data</td>
<td>Attestation/Communication</td>
</tr>
<tr>
<td>- Secrets</td>
<td>Adversary who controls OS still can’t see inside enclave</td>
</tr>
</tbody>
</table>
Hardware Enclaves

A trusted component in an untrusted system

- Protected memory isolates enclave from compromised OS
- Proves authenticity via a process called *attestation*
  - Is it “post-quantum” secure?
EPID Signatures [BL09]

Group signature-like primitive that provides two properties:

1. Signatures from any member of a group are indistinguishable from each other

2. Users can have their credentials revoked either by a blacklisted key or a blacklisted signature

Intel’s EPID signature scheme relies on pairings and is not post-quantum secure
EPID Signatures [BL09]

$sk_i, cert_i \leftarrow \text{Join}(\ldots)$ - interactive protocol between group member and manager to join group

$\sigma \leftarrow \text{Sign}(gpk, sk_i, cert_i, m, \text{SIG-RL})$ - any user who has joined can sign a message anonymously as a group member

$1/0 \leftarrow \text{Verify}(gpk, m, \text{KEY-RL}, \text{SIG-RL}, \sigma)$ - signatures only verify if signed by a valid, unrevoked group member

$\text{KEY-RL}' \leftarrow \text{RevokeKey}(\text{KEY-RL}, sk_i)$ - revoke a group member by key

$\text{SIG-RL}' \leftarrow \text{RevokeSig}(\text{SIG-RL}, \sigma)$ - revoke a group member by signature

Security properties: **Anonymity** and **Unforgeability**
sk_i, cert_i ← Join(...) - interactive protocol between group member and manager to join group

σ ← Sign(gpk, sk_i, cert_i, m, SIG-RL) - any user who has joined can sign a message anonymously as a group member

1/0 ← Verify(gpk, m, KEY-RL, SIG-RL, σ) - signatures only verify if signed by a valid, unrevoked group member

KEY-RL' ← RevokeKey(KEY-RL, sk_i) - revoke a group member by key

SIG-RL' ← RevokeSig(SIG-RL, σ) - revoke a group member by signature

Security properties: **Anonymity** and **Unforgeability**

Our design goal: post-quantum security from **symmetric primitives only**
Picnic Signatures [CDGORRSZ17]

Uses ZKB++ MPC-in-the-head type proof system [IKOS07, GMO16]
i.e. proof of knowledge from symmetric primitives

High-level idea: Signature is proof of knowledge of preimage of a one-way function
e.g. I know sk such that f(sk)=y
Our Basic Approach [BMW03,CG04]

Join
User generates pk, sk
Group manager signs pk to form cert

Sign
User signs message with sk
User publishes proof of knowledge of signature as $\sigma$

Additionally need to support revocation
Our Basic Approach [BMW03, CG04]

Join
User

\[ \text{Sign} \]
\[ s = \text{Sign}(sk_i, m) \]

Proof of Knowledge: I have a certificate on a key \( sk^* \) and a signature \( s \) on message \( m \) signed with \( sk^* \)
Post-Quantum EPID Signature

Join

User $sk_i$

Manager $gsk, gpk$
Post-Quantum EPID Signature

Join

User

sk_i

c

Manager
gsk, gpk
Post-Quantum EPID Signature

Join

User

Manager

gsk, gpk

sk_i

t_{i}^{\text{join}} = f(sk_i, c)

c
Post-Quantum EPID Signature

\[ t_{\text{join}} = f(sk_i, c) \]
Post-Quantum EPID Signature

Sign
\[ r \leftarrow \{0,1\}^\lambda \]
\[ t = f(s_{k_i}, r), r \]
Post-Quantum EPID Signature

Sign
\[ r \leftarrow \{0,1\}^\lambda \]
\[ t = f(sk_i, r), r \]

Proof of Knowledge:
1. I know a valid certificate for \( t^{\text{join}} \), c
Post-Quantum EPID Signature

\textbf{Sign}
\[ r \leftarrow \{0, 1\}^\lambda \]
\[ t = f(sk_i, r), r \]

\textbf{Proof of Knowledge:}
1. I know a valid certificate for \( t^{\text{join}}, c \)
2. I know \( sk_i \) such that \( t = f(sk_i, r) \) and \( t^{\text{join}} = f(sk_i, c) \)
Post-Quantum EPID Signature

Sign

\[ r \leftarrow \{0,1\}^\lambda \]
\[ t = f(sk_i, r), r \]

Proof of Knowledge:

1. I know a valid certificate for \( t^\text{Join}, c \)
2. I know \( sk_i \) such that \( t = f(sk_i, r) \) and \( t^\text{Join} = f(sk_i, c) \)
3. There is no signature in SIG-RL such that \( f(sk_i, r') = t' \)

publish proof and \( t \) as signature
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td>ZKB++, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td></td>
</tr>
</tbody>
</table>
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td>ZKB++, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td></td>
</tr>
</tbody>
</table>
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td><strong>ZKB++</strong>, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, LowMC</td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td></td>
</tr>
</tbody>
</table>
### Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td><strong>ZKB++</strong>, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, <strong>LowMC</strong></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td></td>
</tr>
</tbody>
</table>
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td><strong>ZKB++</strong>, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, <strong>LowMC</strong></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td>Tree-based, SPHINCS, Fish</td>
</tr>
</tbody>
</table>
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td>ZKB++, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, <strong>LowMC</strong></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td><strong>Tree-based</strong>, SPHINCS, Fish</td>
</tr>
</tbody>
</table>
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td><strong>ZKB++</strong>, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, <strong>LowMC</strong></td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td><strong>Tree-based</strong>, SPHINCS, Fish</td>
</tr>
</tbody>
</table>

Post-quantum EPID signature size (group size $2^{30}$):
## Instantiation

<table>
<thead>
<tr>
<th>Need</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero Knowledge PoK</td>
<td>ZKB++, Ligero, zk-STARK</td>
</tr>
<tr>
<td>PRF/CRHF</td>
<td>AES, MiMC, LowMC</td>
</tr>
<tr>
<td>Post-Quantum Signature from symmetric primitives</td>
<td>Tree-based, SPHINCS, Fish</td>
</tr>
</tbody>
</table>

Post-quantum EPID signature size (group size $2^{30}$): **217MB**

Way too big!! Culprit: signature verification inside PoK
Post-Quantum EPID Signature

Sign
\[ r \leftarrow \{0,1\}^\lambda \]
\[ t = f(sk_i, r), r \]

Proof of Knowledge:
1. I know a valid certificate for \( t^{join}, c \)
2. I know \( sk_i \) such that \( t = f(sk_i, r) \) and \( t^{join} = f(sk_i, c) \)
3. There is no signature in SIG-RL such that \( f(sk_i, r') = t' \)

publish proof and \( t \) as signature

Requires signature verification!
How can we remove this?
The Attestation Setting

Each Intel SGX attestation involves contacting Intel, who verifies the attestation for you.

How can we leverage this to reduce signature sizes?
The Attestation Setting

Each Intel SGX attestation involves contacting Intel, who verifies the attestation for you.

How can we leverage this to reduce signature sizes?

Idea: If group manager has to be online, maybe it can update users’ certificates
User anonymity sets relative to last certificate update
Signatures for Attestation

Manager puts user credentials in a Merkle tree and signs root

Users get newest Merkle root/inclusion proof when they connect to the manager
Signatures for Attestation

Manager puts user credentials in a Merkle tree and signs root

Users get newest Merkle root/inclusion proof when they connect to the manager

Signature on Merkle tree root can be verified outside PoK

Only need much smaller Merkle inclusion proof inside PoK
Signatures for Attestation

\[ r \leftarrow \{0,1\}^\lambda \]
\[ t = f(sk_i, r), r \]

Proof of Knowledge:

1. I know an inclusion proof for \( t^{\text{join}}, c \)
2. I know \( sk_i \) such that \( t = f(sk_i, r) \) and \( t^{\text{join}} = f(sk_i, c) \)
3. There is no signature in SIG-RL such that \( f(sk_i, r') = t' \)

Publish proof, t, and signed Merkle root as signature

Similar to post-quantum Ring signatures of Derler et al [DRS17]
# Signature Sizes

<table>
<thead>
<tr>
<th>Group Size</th>
<th>RO Model*</th>
<th>QRO Model*</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^7$</td>
<td>1.37MB</td>
<td>2.64MB</td>
</tr>
<tr>
<td>$2^{10}$</td>
<td>1.85MB</td>
<td>3.59MB</td>
</tr>
<tr>
<td>$2^{20}$</td>
<td>3.45MB</td>
<td>6.74MB</td>
</tr>
<tr>
<td>$2^{30}$</td>
<td>5.05MB</td>
<td>9.89MB</td>
</tr>
<tr>
<td>$2^{40}$</td>
<td>6.65MB</td>
<td>13.0MB</td>
</tr>
</tbody>
</table>

Potential application: large data transfer, e.g. streaming movies

*under ideal cipher assumption on LowMC