One-Shot Signatures and Applications

Aggelos Kiayias
University of Edinburgh and IOG
Joint work with Ryan Amos, Marios Georgiou, Mark Zhandry
Slides credits: Marios Georgiou & AK
One-Shot QSig

1. Quantum Retrieval Games
2. Tokenized Signatures
3. One-Shot Signatures

Common syntax
- \( \text{Gen}(1^n): (sk, vk) \)
- \( \text{Sign}(sk, m): \sigma \)
- \( \text{Ver}(vk, m, \sigma): b \)

Correctness

If \((sk, vk) \leftarrow \text{Gen}(1^n)\) then \(\text{Ver}(vk, m, \text{Sign}(sk, m)) = 1\) for any message \(m\).

Security

High level: \(sk\) can sign only a single message. It collapses.
Quantum Retrieval Games

\[(sk, vk) \leftarrow \text{Gen}(1^n)\]

A wins if:
- \(\text{Ver}(vk, 0, \sigma_0) = 1\)
- \(\text{Ver}(vk, 1, \sigma_1) = 1\)

**Construction:** Hidden Matching Problem [Gavinsky 2012]

**Security:** Unconditional
Tokenized Signatures

\[ (sk, vk) \leftarrow \text{Gen}(1^n) \]

A wins if:
- \[ \text{Ver}(vk, 0, \sigma_0) = 1 \]
- \[ \text{Ver}(vk, 1, \sigma_1) = 1 \]

**Construction:** Hidden Cosets [CLLZ21]

**Security:** iO + OWF
One-Shot Signatures

\[ (sk, vk) \leftarrow \text{Gen}(1^n) \]

\[ \sigma_0, \sigma_1, vk \]

A wins if:

- \( \text{Ver}(vk, 0, \sigma_0) = 1 \)
- \( \text{Ver}(vk, 1, \sigma_1) = 1 \)

Constructions:

- Partition \( Z_2^n \) into subsets and access partition via
  - Quantum oracles
  - Classical oracles
- Plain Model: Candidate constructions by obfuscating the oracles
One-Shot Signatures from One-Shot Chameleon

Syntax

- \( \text{Gen}(1^n): (sk, y) \)
- \( \text{Invert}(sk, x): r \)
- \( H(x, r): y \)

Significance

- Signing key
- Verification key
- Message
- Signature

Correctness

If \( (sk, y) \leftarrow \text{Gen}(1^n) \) then \( H(x, r) = y \) for any input \( x \), where \( r \leftarrow \text{Invert}(sk, x) \).

Security

\( H \) is collision resistant.
Equivocal Hash Functions

Syntax

- $\text{Gen}(1^n): (sk, y, Q(.))$
- $\text{Equiv}(sk, b): x$
- $H(x): y$

Correctness/Equivocality

If $(sk, y) \leftarrow \text{Gen}(1^n)$ then $H(x) = y$ and $Q(x)=b$ for any input $x$, where $x \leftarrow \text{Invert}(sk, b)$.

Security

$H$ is collision resistant.
# OS Chameleon and Equivocal Hash Functions

**Theorem.** Equivocal hash functions $\iff$ One-shot Chameleon

- $\text{Gen}(1^n) : (sk, y, Q(.))$
- $\text{Equiv}(sk, b) : x$
- $H(x) : y$
- $\text{Gen'}(1^n) : (sk, y')$
- $\text{Invert}(sk, x') : r'$
- $H'(x', r') : y$

$x'$ is in $\{0,1\}$

Proof $\leq$ we set $Q$ to be the first bit. Define $H(x) = H'(x_0, x_1 \ldots x_{k-1})$.

For Equiv, given $sk, b$, run $\text{Invert}(sk, b)$ to obtain $r'$ such that $H'(b, r') = y$, it follows that $b || r'$ satisfies the $Q$ predicate and is a preimage.

$\Rightarrow$ Define $H'(x', r') = H(r') || Q(r') \ XOR \ x'$ and $y' = y || 0$

For invert($sk, x'$) : run Equiv($sk, x'$) to obtain $r'$ such that $Q(r') = x'$ and $H(r') = y$. Observe that $H'(x', r') = H(r') || Q(r') \ XOR \ x' = y \ || \ x' \ XOR \ x' = y \ || \ 0 = y'$
The quantum flavours of collision resistance

- **Collision resistant.**
  - Hard to find distinct $x_0$, $x_1$ such that $H(x_0) = H(x_1)$

- **Unequivocal**
  - No efficient adversary can find a hash $y$, and predicate $Q$ such that later given $b$, it is possible to find a pre-image $x$ with $Q(x) = b$ and $H(x) = y$

- **Collapsing**
  - Let $A(y)$ be the preimage set of any $y$. Having access to a superposition of $A(y)$ is no more useful than having a random element of $A(y)$.

**Observe:** Collapsing $\Rightarrow$ Unequivocal $\Rightarrow$ Collision resistant
Hash functions with quantum capabilities

- Initially non-collapsing introduced as a problem.
  - Observe that it is feasible to be CR and non-collapsing: due to no-cloning, despite the uncertainty in a pre-image set $A(y)$ this state cannot be cloned and measured twice to break collision resistance.
  - **What is the potential problem**: non-committing hashing - [U16]
  - **However it can also be a good thing**

- **(Classically) Collision Resistant & non-collapsing**
  - **Application**: quantum lightning [Z17] (a stronger version of quantum money)
    - Gen => bolt
    - Verify(bolt) => bolt, serial number.
    - Not possible to create two distinct bolts with the same serial.

- **(Classically) Collision resistant & equivocal**
  - More applications! Quantum lightning => decentralized cryptocurrency, but one-shot signatures can do more: e.g., decentralized smart contracts without PoW/erasures/VDFs
Constructing One-Shot Signatures

Approach via one-shot Chameleon for one-bit messages

- \textbf{Gen}(1^n): (sk, y)
- \textbf{Invert}(sk, b): r
- H(b, r): y

Key question: how do we implement \textbf{Invert}

**Pick-one trick** from [ARU14]. Grover’s search is adapted appropriately.

Suppose \( A(y) \) the preimage set of \( y \)

- Superposition of \( A(y) \); \( \Rightarrow \) elements of the form \((0, r)\) and \((1, r)\)
- Apply phase shift \(|(-1)^H(b, r)(b, r)>\) and diffusion \( I -2^*|A(y)> < A(y)|\)
- Measure & repeat until a suitable solution is produced.
  - (note: for efficiency, there should be sufficiently many \( r \) choices \(|A(y)| / |A(y) : H(b, r)=y| \) is polynomial)
One-Shot Chameleon From Oracles

Quantum Oracle

- Partition \{0,1\}^n into 2^{n/2} sets \{U_y\} of size 2^{n/2}
- Oracles
  1. \(H(x) = y\) if \(x \in U_y\)
  2. \(\text{Reflect}(\text{state}, y) =\)
     - If \(\text{state} = |y, U_y\rangle\) return \(-|y, U_y\rangle\)
     - If \(\text{state} = |y, \perp U_y\rangle\) return \(|y, \perp U_y\rangle\)
- Gen:
  1. Evaluate \(H\) on uniform superposition
  2. Measure output register to get \((sk, y)\)
     - Input register collapses to uniform superposition of \(y\)’s preimages
- \(\text{Invert}(sk, b):\)
  1. Run Grover’s search using \(\text{Reflect}\).
  2. Retrieve uniform superposition of preimages starting with \(b\).
  3. Measure.

Classical Oracle

- Partition \{0,1\}^n into 2^{n/2} cosets \{U\} of size 2^{n/2}
- Oracles
  1. \(H(x) = y\) if \(x \in U_y\)
  2. \(H^\perp(x, y) =\)
     - If \(x \in U_y^\perp\), accept
     - If \(x \notin U_y^\perp\), reject
- Gen:
  1. Evaluate \(H\) on uniform superposition
  2. Measure output register to get \((sk, y)\)
     - Input register collapses to uniform superposition of \(y\)’s preimages
- \(\text{Invert}(sk, b):\)
  1. Run Grover’s search using \(\text{QFT} \cdot H^\perp(\cdot, y) \cdot \text{QFT}\).
  2. Retrieve uniform superposition of preimages starting with \(b\).
  3. Measure.
Applications
Applications: Budget Signatures

\[\sigma \leftarrow \text{Sign}(sk, (vk_0, vk_1, 0.4))\]
PoW Coins with Classical Communication

- **To mine a new coin:**
  - \((sk_0, vk_0) \leftarrow \text{Gen}\)
  - Run a proof of work on \(vk_0\) to generate proof \(\pi\).
  - \((sk_0, vk_0, \pi)\) is the coin

- **To send the coin:**
  - Receiver generates new \((sk_1, vk_1) \leftarrow \text{Gen}\) and sends \(vk_1\) to the sender.
  - Sender signs \(vk_1\): \(\sigma \leftarrow \text{Sign}(sk_0, vk_1)\) and sends \((vk_0, \pi, \sigma)\) to the receiver.
Pow Coins: Improvements

- **Succinctness:**
  - Compress \((vk_0, \pi), (vk_1, \sigma_1), \ldots, (vk_n, \sigma_n)\) into succinct proof

- **Protecting privacy:**
  - Use Zero-Knowledge, (as above)
Ordered Signatures

- Each signature is associated with a time tag
- **Security**: One cannot sign a message with a “past” tag

\[
(vk_0, \sigma_0) \rightarrow (vk_1, \sigma_1) \rightarrow (vk_2, \sigma_2)
\]

\[
\sigma_0 \leftarrow \text{Sign}(sk_0, (vk_1, m_1, t_1)) \quad \sigma_1 \leftarrow \text{Sign}(sk_1, (vk_2, m_2, t_2))
\]

- Verification:
  1. Verify all signatures
  2. Verify that \( t_i > t_{i-1} \)
- Proof of burn by signing at \( t = \infty \)
Public-coin Proofs of Quantumness

OSS => PPQ

How can a prover convince a verifier he is quantum

\[(sk, vk) \leftarrow \text{Gen}\]

\[vk\]

\[m \leftarrow \{0,1\}^n\]

\[\sigma \leftarrow \text{Sign}(sk,m)\]

\[\sigma\]

\[\text{Ver}(vk,m,\sigma) = 1\]

If a classical P is convincing then by rewinding we could get signatures of two messages => breaks one-shot
One-Shot Signatures with Trapdoors

\[
\begin{align*}
\text{sk, vk} & \xleftarrow{} \text{Gen}(1^n) \\
\sigma_0, \sigma_1, \text{vk} & \quad \text{A wins if:} \\
& \quad \text{Ver(tr, vk, 0, } \sigma_0) = 1 \\
& \quad \text{Ver(tr, vk, 1, } \sigma_1) = 1
\end{align*}
\]

Construction: [BCMVV18]
Assumption: Learning with Errors
## Summary QSig

<table>
<thead>
<tr>
<th></th>
<th>Honest Key Generation</th>
<th>Dishonest Key Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Verification</strong></td>
<td>Quantum Retrieval Games (unconditional)</td>
<td>Trapdoor One-Shot Signatures (LWE)</td>
</tr>
<tr>
<td><strong>Public Verification</strong></td>
<td>Tokenized Signatures (iO + OWF)</td>
<td>One-Shot Signatures (only w.r.t. an oracle)</td>
</tr>
</tbody>
</table>
Directions

1. Constructing Equivocal hash functions
   a. E.g., via light weight hash function approach as in [Z19], other techniques?

2. OSS design approaches and security
   a. Lower bounds for finding collisions against the partition oracle $H$, given $H^\perp(x,y)$
   b. We based the construction on one-shot Chameleon; other design techniques?

3. OSS without oracles
   a. E.g., from iO, similarly to tokenized signatures.
      Apply obfuscation to the classical oracle construction from one-shot Chameleon, is the
      resulting non-oracle construction secure?

4. Succinct and ZK proofs for chains of signatures for OSS applications.