Local Quantum Cryptography

Mark Zhandry (Princeton & NTT Research)

Based on joint work with Ryan Amos, Marios Georgiou, and Aggelos Kiayias
Quantum Background

Quantum Period Finding
[Simon’94, Shor’95]

Factoring: $N = pq \rightarrow p, q$

DLog: $g, g^a \rightarrow a$
Quantum Background

Observer Effect

QKD [Bennett-Brassard’84]

Observer effect → Eavesdropping detection
Quantum Background

Quantum Money [Wiesner’70]
Post-Quantum vs Quantum Crypto

Post-Quantum Crypto:

- Protect classical crypto from quantum attacks

Quantum Crypto:

- Use quantum effects to do new things
Emerging Area: Local Quantum Crypto

Local Quantum Crypto:

Everyone’s quantum, communication classical

Main Question: Is anything interesting possible?
Prior Work: (Verifiable) Delegation
Mahadev’18 x2, Brakerski-Christiano-Mahadev-Vazirani-Vidick’18

( I Don’t really count multi-device setting: requires entanglement )
Two Motivating Examples

**“Classical” Quantum Money**
Send quantum money over classical channels?

**Rate-Limited Decryption**
Can only decrypt single ciphertext every $T$ minutes

Send $c_1$, $c_2$, $c_3$ to Alice who decrypts $m_1$, $m_2$, $m_3$. 
This Work: Two Questions

**Q1:** Can quantum keys yield any interesting crypto?

**Q2:** Can quantum states be sent over classical channels?
Disclaimer

**Strong computational assumptions:**
- Obfuscation (VBB)
- *Extractable* witness encryption
- *Recursively composable* zk-SNARKs
- Post-quantum proofs of (sequential) work
Part 1: One-Shot Signatures and Applications
Tool: One-Shot Signatures

Syntax:

\((pk, sk) \leftarrow \text{Gen}()\)
\(\sigma \leftarrow \text{Sign}(sk, m)\)
\(0/1 \leftarrow \text{Ver}(pk, m, \sigma)\)

Security:

\((pk, m_0, m_1, \sigma_0, \sigma_1)\) s.t.
\(m_0 \neq m_1,\)
\(\text{Ver}(pk, m_0, \sigma_0) = 1,\) and
\(\text{Ver}(pk, m_1, \sigma_1) = 1\)
Impossibility of One-Shot Signatures?

**Attack?**
- $(pk, sk) \leftarrow \text{Gen}()$
- $\sigma_0 \leftarrow \text{Sign}(sk, m_0)$
- $\sigma_1 \leftarrow \text{Sign}(sk, m_1)$

**Idea!**
What if $sk$ is “used up” to produce $\sigma_0$?
- Makes no sense classically (rewinding)
- But quantumly, maybe?
One-Shot Signatures (Quantum)

**Syntax:**

\[(pk, \cdot sk) \leftarrow \text{Gen}() \]
\[
\sigma \leftarrow \text{Sign}(\cdot sk, m) \\
0/1 \leftarrow \text{Ver}(pk, m, \sigma)
\]

**Security:**

\[(pk, m_0, m_1, \sigma_0, \sigma_1) \text{ s.t.} \]
\[
m_0 \neq m_1, \\
\text{Ver}(pk, m_0, \sigma_0) = 1, \text{ and} \\
\text{Ver}(pk, m_1, \sigma_1) = 1
\]

For now, assume \(\exists\) OSS. Will construct later.
Goal: Prove that you destroyed your signing key
Solution: Signature Chaining
Solution: Signature Chaining

(assume message is part of sig)

\[ \sigma_{pk_0}(m_1 \parallel pk_1), \]
Solution: Signature Chaining

\[ \sigma_{pk_0}(m_1 \| pk_1), \sigma_{pk_1}(m_2 \| pk_2), \]

\[ pk_0 \]
Solution: Signature Chaining

\[ \sigma_{pk_0}(m_1 \| pk_1), \sigma_{pk_1}(m_2 \| pk_2), \sigma_{pk_2}(\bot) \]

Proof Idea:
- Valid post-burn signature
- Forked chain
- OSS Forgery
Caveats

- $|\text{signature}|$ grows with $\#$(messages)
  
  Fix: SNARKs

- $|\text{sk}|$ grows with $\#$(messages)
  
  Fix: Recursively Composable SNARKs

Stateful Signing

Natural for quantum keys
(reading key may disturb it)
OSS Apps: Burnable *Decryption*

**Goal:** Prove that you destroyed your decryption key

\[ \text{pk} \xrightarrow{c''} \text{sk} \xrightarrow{c} \pi \xrightarrow{c''} \text{pk} \]

\[ m, m' \xrightarrow{c''} m'', m''' \]
Burnable Sigs $\rightarrow$ Burnable Decryption

**Tool:** (Extractable) Witness Encryption

$$c \leftarrow \text{WE.Enc}(\text{NP statement } x, m)$$
$$m \leftarrow \text{WE-Dec}(x, \text{witness } w, c)$$

**Security:** $c$ hides $m$, unless you “know” a witness
Burnable Sigs $\rightarrow$ Burnable Decryption

$r \leftarrow \text{Message space}$
$c \leftarrow \text{WE.Enc}( \text{"r has a sig"}, m)$

Actually, OSS works directly
OSS Apps: Ordered Signatures

**Goal:** Only sign messages in increasing order

\[(m, \sigma) \quad \text{✓} \quad (m' > m, \sigma') \quad \text{✓} \]

\[(m' < m, \sigma) \quad \text{✗} \quad (m'', \sigma'') \quad \text{✓} \]

Same construction as burnable sigs, Ver checks message order
OSS Apps: Ordered Signatures

\[ m = (\text{timestamp, document}) \]

\[(T, \ast), \sigma\] \rightarrow \checkmark

\[(T' < T, D), \sigma\]

If Bob accepts, Alice must have “known” \( D \) at time \( T \)
OSS Apps: Single-Signer Signatures

Honest $sk$ can sign any number of messages
Ordered Sigs $\rightarrow$ Single-Signer Sigs

**Proof:** Used timestamped version

Secret keys must respect ordering, so can’t sign independently
OSS Apps: Single-Decryptor Encryption

Same as single-signer sigs, except now secret keys are for decrypting
Single-Signer $\rightarrow$ Single-Decryptor

$r \leftarrow \text{Message space}$
$c \leftarrow \text{WE.Enc(“r has a sig”, m)}$

Again, OSS works directly
Single-Decryptor App: Traitor Prevention

**Recall Traitor Tracing** [Chor-Fiat-Naor’94]:

- **Goal:** Given pirate decoder, can identify the traitor(s)

Diagram showing encrypted broadcast to multiple devices, indicating the process of identifying traitors.
Single-Decryptor App: Traitor Prevention

Traitor Prevention:

Encrypted broadcast

Only N individuals ever capable of decrypting
Single-Decryptor App: Traitor Prevention

Traitor Prevention:

Only $N$ individuals ever capable of decrypting
OSS Apps: Quantum Money*

Verification: check $\sigma_{pk_{mint}}(pk)$, that $sk_{mint}$ can sign random message

*Technically not “local” quantum crypto; will revisit later
OSS Apps: Cryptocurrency sans Blockchain

Mining entirely local!
OSS \rightarrow \text{Cryptocurrency w/o Blockchain}

\textbf{Tool: Proofs of Work (PoW)}

\(\pi \leftarrow \text{PoW}(ch, T), \text{ takes time } T\)

\(0/1 \leftarrow \text{Ver}(ch, T, \pi)\)

Time \ll T

\(ch \leftarrow \$\)

\(\pi \rightarrow \text{Ver}(ch, T, \pi) = 1\)
OSS → Cryptocurrency w/o Blockchain

Verification: check that can sign random message, PoW valid
OSS Apps: Delay Signatures

Can only sign single message every $T$ minutes

Application:
• Limit rate (quantum) money is created
OSS \rightarrow Delay Signatures

**Tool:** Proofs of Sequential Work (PoSW)

\[ \pi \leftarrow \text{PoSW}(ch, T), \text{ takes sequential time } T \]

\[ 0/1 \leftarrow \text{Ver}(ch, T, \pi) \]

Sequential time \(\ll T\)
OSS → Delay Signatures

\[ \pi_1 = \text{PoSW}(pk_0, T) \]

\[ \sigma_{pk_0}(m_1 \| pk_1 \| \pi_1), \]

\[ pk_0 \]
OSS $\rightarrow$ Delay Signatures

\[
\sigma_{pk_0}(m_1 \| pk_1 \| \pi_1), \\
\sigma_{pk_1}(m_2 \| pk_2 \| \pi_2), \\
\pi_2 = \text{PoSW}(\sigma_{pk_0}, T)
\]
OSS Apps: Delay Decryption

Can only decrypt single ciphertext every $T$ minutes

Application:
• Tiered broadcast subscriptions
Delay Sigs $\rightarrow$ Delay Decryption

\[ r \leftarrow \text{Message space} \]
\[ c \leftarrow \text{WE.Enc( "r has a sig", m)} \]
Part 2: Classically Sending Quantum States
Quantum States over Classical Channels?

**Rejected Solution:**
Send classical description of state

What if don’t know classical description?

**Rejected Solution:**
Use quantum teleportation

Requires quantum entanglement

**No In General:**
Could use to create entanglement via classical channel
Quantum States over Classical Channels?

Q2’: Can any *unclonable* state be sent over a classical channel?
Q2 Rephrased

Q2’: Can any unclonable state be sent over a classical channel?

No, if single message from Alice to Bob

No, if computationally unbounded

What if interaction + computational assumptions?
Signature Delegation with OSS
Signature Delegation with OSS
Signature Delegation with OSS

Alice effectively sent her unclonable state to Bob over classical channel.
Signature Delegation

Using recursively composable \textit{zk}-SNARKS, received state is computationally indistinguishable from original

Can apply to all of our schemes, to send quantum keys/money over classical channels
Part 3: Constructing OSS
Unequivocal Hash Functions
Closely related to concepts from [Ambainis-Rosmanis-Unruh’14, Unruh’16]

Classically:
col. resistance $\rightarrow$ unequiv. hash (rewinding)

Quantumly: maybe not
Equivocal Hash Functions

Equivocal Hash = Col. Resistance + ! Unequivocal

**Easy Thm:** Equivocal Hash $\rightarrow$ OSS

[Ambainis-Rosmanis-Unruh’14,Unruh’16]:
Construction relative to *quantum* oracle

But, no clear idea how to instantiate
Our Result

**Thm:** Equivocal hash relative to *classical* oracle
Can heuristically instantiate w/ iO
Simpler Goal: Non-Collapsing Hash

1

“collapses” to superposition over pre-images of $h$

0

“collapses” to single value $x$
A First (Broken) Attempt

\[ A \in \mathbb{Z}_2^{m \times n} \]

random, secret

\[ A \cdot x = b_1 \]
\[ A \cdot x = b_2 \]
\[ A \cdot x = b_3 \]

H: assign each “slice” a random output

\[ \mathbb{Z}_2^n \]
A First (Broken) Attempt

Inc. oracle $O$ which checks for membership in $\text{RowSpan}(A)$
Problem: Periodic $\Rightarrow$ Not Collision Resistant!

Simon’s Alg:

$x \xrightarrow{H(x)} (A_0, x=b) \xrightarrow{QFT} y \xrightarrow{?} y$

Recall: $y \in \text{RowSpan}(A)$

Repeat several times: reconstruct $A$
Our Construction

\( \mathbf{v}_\epsilon \in \mathbb{Z}_2^n \)
random, secret

\( \mathbf{v}_\epsilon \cdot \mathbf{x} = 0 \)

\( \mathbf{v}_\epsilon \cdot \mathbf{x} = 1 \)
Our Construction

\(v_0, v_1 \) random, secret

\(v_0 \cdot x = 0\)
\(v_1 \cdot x = 1\)

\(v_{\varepsilon} \cdot x = 0\)
\(v_0 \cdot x = 1\)

\(v_{\varepsilon} \cdot x = 1\)
\(v_1 \cdot x = 0\)
Our Construction

$v_{00}, v_{01}, v_{10}, v_{11}$ random, secret

$v_\varepsilon . x = 0$
$v_0 . x = 0$
$v_{00} . x = 0$
$v_\varepsilon . x = 0$
$v_0 . x = 0$
$v_{00} . x = 0$

$v_\varepsilon . x = 0$
$v_0 . x = 1$
$v_{01} . x = 0$
$v_\varepsilon . x = 0$
$v_0 . x = 1$
$v_{01} . x = 0$

$v_\varepsilon . x = 1$
$v_1 . x = 1$
$v_{11} . x = 0$
$v_\varepsilon . x = 1$
$v_1 . x = 1$
$v_{11} . x = 0$

$v_\varepsilon . x = 1$
$v_1 . x = 0$
$v_{10} . x = 0$
$v_\varepsilon . x = 1$
$v_1 . x = 0$
$v_{10} . x = 0$
Our Construction

\[ \text{Inc. oracle } O \text{ which checks for membership in } \text{RowSpan}(A_h) \]
Simon’s Algorithm?

Simon’s Alg:

\[ H(x) \xrightarrow{(A_h \cdot x = b_h)} QFT \xrightarrow{y} y \]

\[ y \in \text{RowSpan}(A_h) \]

Repeat \(\Rightarrow\) Different \(h\) \(\Rightarrow\) Different \(A_h\) \(\Rightarrow\) Never able to reconstruct \(A_h\)
Our Construction

**Thm:** If $H, O$ given as oracles, then collision resistant

With some extra work, can also equivocate
Future Directions?

Better assumptions?
• Even iO + LWE + LPN + Isogenies + ...?

More apps?
• Fancier crypto (e.g. functional enc)?
• Classically send copy-protected programs?
Thanks!