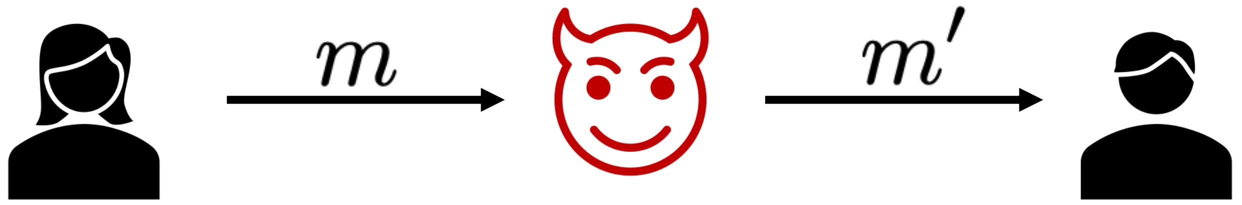
CS 258: Quantum Cryptography

Mark Zhandry

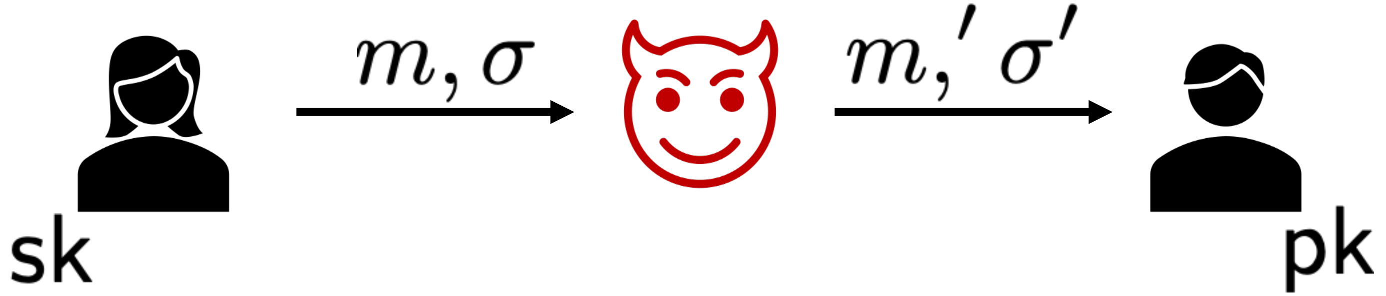
A long long time ago…

Signatures



Signatures

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}()$$



$$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m) \qquad 0/1 \leftarrow \mathsf{Ver}(\mathsf{pk}, m', \sigma')$$

Adversary who sees pk and a signed message m, σ cannot produce another signed message that verifies

Schnorr Signatures

Assume a "good" hash function H

$\mathsf{sk} = s$ $\mathsf{pk} = g^s$

$\mathsf{Sign}(\mathsf{sk}, m)$: choose random x

$$a = g^z$$
$$c = H(m, a)$$
$$r = z + cs$$
$$\sigma = (a, c, r)$$

$\mathsf{Ver}(\mathsf{pk}, m, (a, c, r))$: check:

$$c = H(m, a)$$
$$g^r = a \times \mathsf{pk}^c$$

Intuition for security

$a = g^z$ $c = H(m, a)$ $r = z + cs$

$$\sigma = (a, c, r)$$

The hash enforces that challenge formed after m, a

Otherwise, pick r, c, let $a = g^r \times \mathsf{pk},^{-c}$
find m s.t $c = H(m, a)$

by construction,
$g^r = a \times \mathsf{pk}^c$

Schnorr for Group Actions

Assume a “good” hash function H

$$\mathsf{sk} = s \qquad \mathsf{pk} = s * x_0$$

$\mathsf{Sign}(\mathsf{sk}, m)$: choose random $z_1, \cdots, z_\lambda$

$$a_i = z_i * x_0$$

$$c = H(m, a_1, \cdots, a_\lambda)$$

$$r_i = (1 - c_i)z_i \text{“}+\text{”} c_i s$$

$$\sigma = (a_1, \cdots, a_\lambda, c, r_1, \cdots, r_\lambda)$$

$c \in \{0,1\}^\lambda$

Question for today: How do we formally argue security?

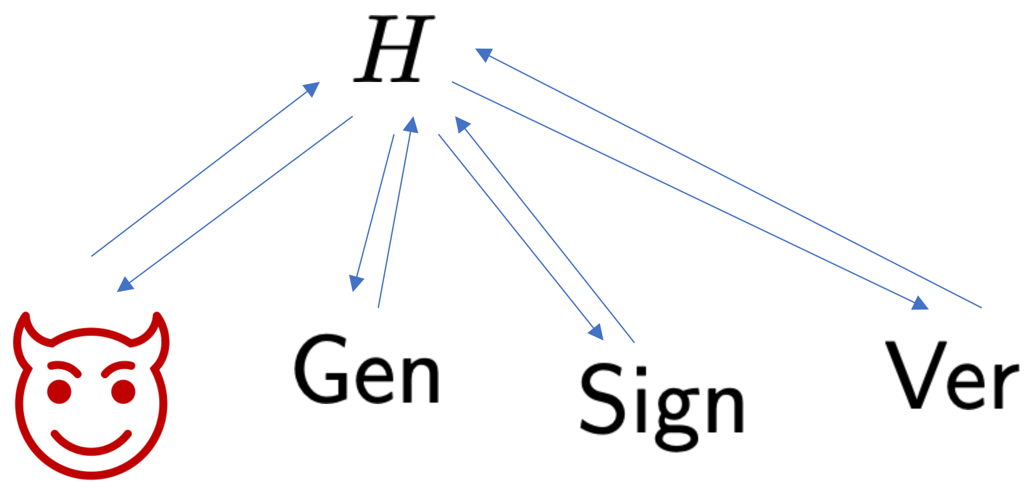
In particular, what does it mean for a hash function to be “good” in this context?

Problem: in general, we have no idea how to prove security of Schnorr from “typical” properties of hash functions

But in real world, nevertheless seems secure. So what do we do?

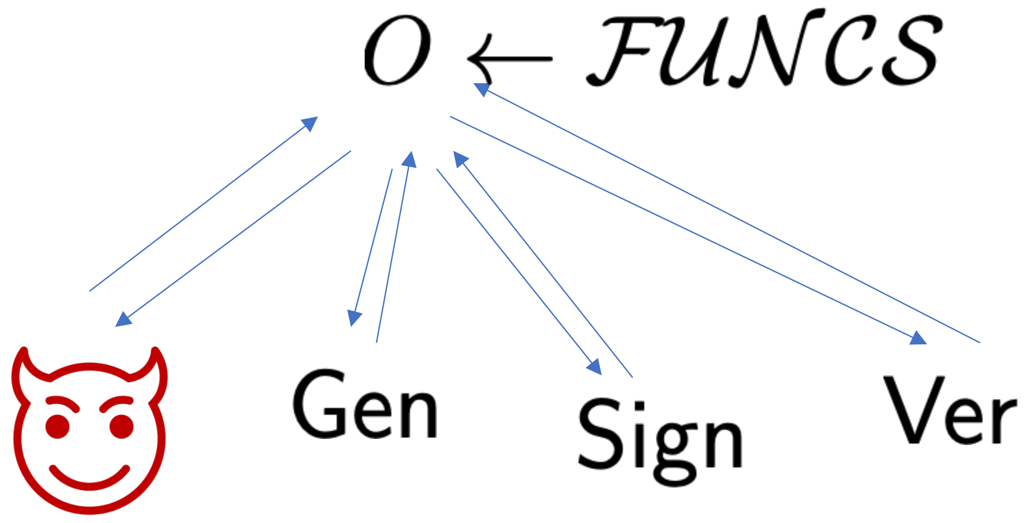
The (Classical) Random Oracle Model

In real world:



Adversary may do more, but in typical attacks all it does is evaluate H

Random oracle model:



Treat hash function as a truly random function that everyone queries

The random oracle model is **not** an assumption about H

H is clearly distinguishable from a random function, since we have efficient code that evaluates it

Instead, it’s an assumption that attacks don’t do anything which depends on the code except evaluate. Sometimes called a “heuristic”

The heuristic is known to fail in certain contrived settings. But for “practical” applications, seems to be reasonable

It turns out that the random oracle model is the only way we know how to justify the security of many of the cryptosystems we currently use today

This continues to be true in the quantum setting

Random oracles in a quantum world

Consider running Grover search on a hash function H

This requires applying the unitary U_H

Just given classical queries to H, there is no way to apply U_H

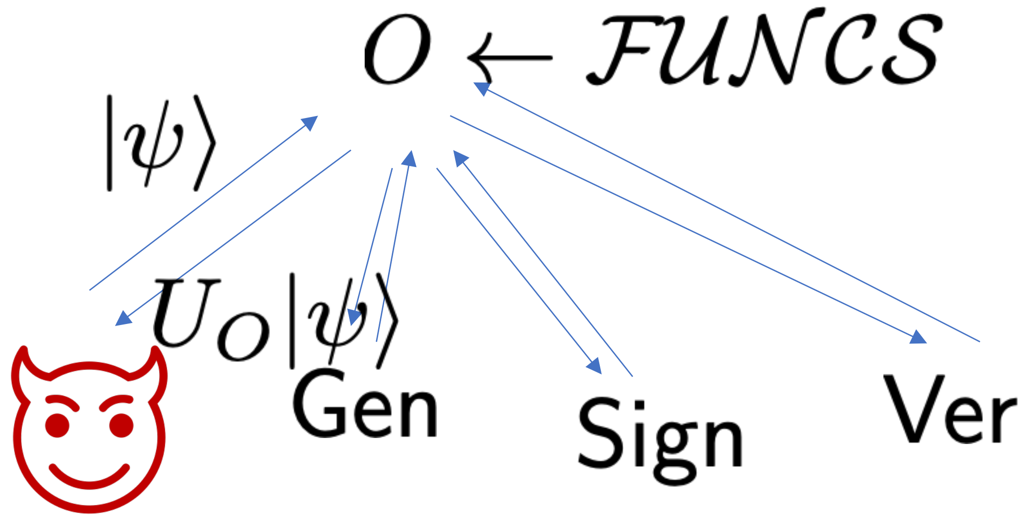
The classical ROM fails to capture standard quantum algorithms!

The Quantum Query Model

A quantum query to a function O just means that we get to press a button, and U_O will be applied to our state. No need to implement it ourselves

Grover search (and collision finding) work in this model, and are known to be optimal in this model

Quantum random oracle model

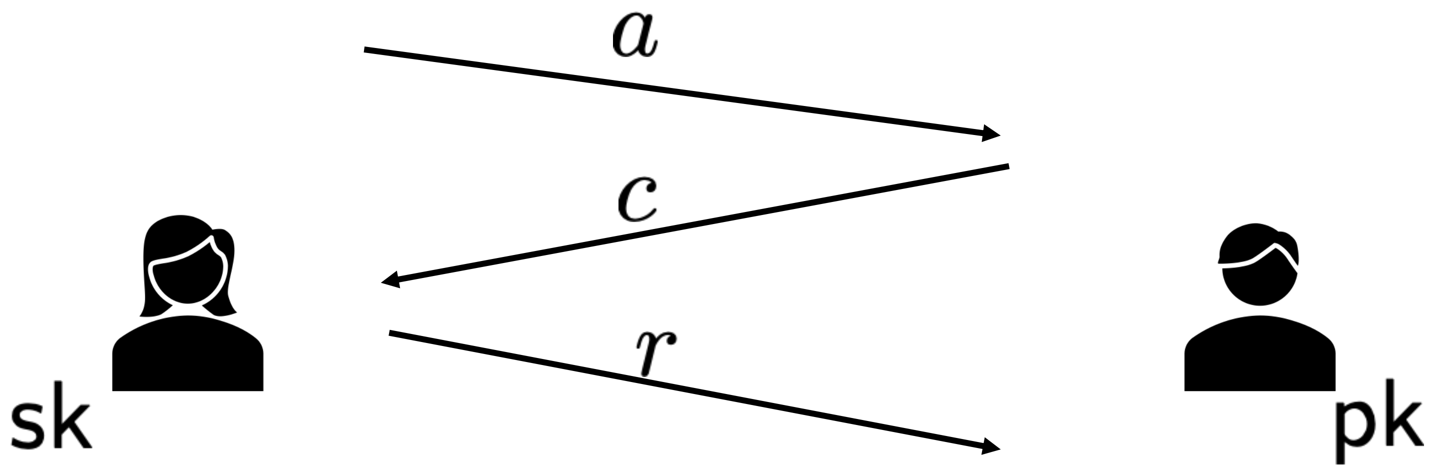


Typically, the "honest" algorithms will still only make classical queries

Now let’s see a classical security proof, and how it fails when we move to the quantum setting

Sigma Protocols

$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$



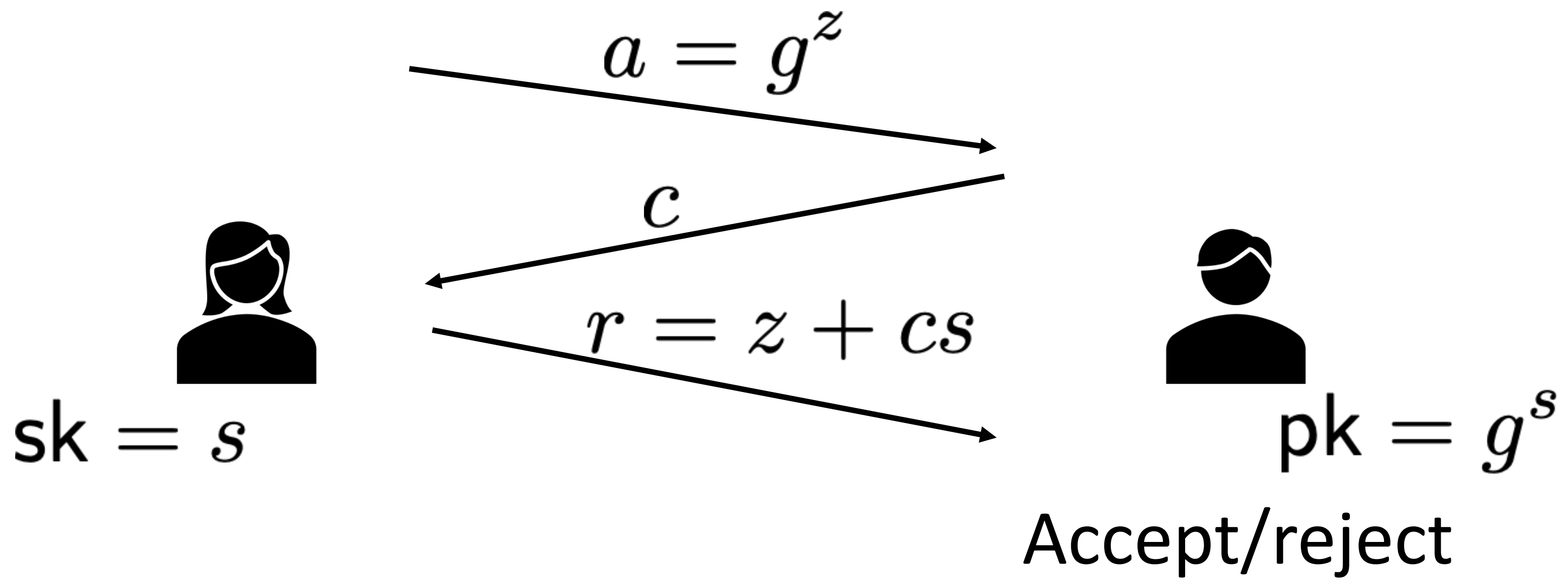
Accept/reject

Public coin: c is just a random string

Soundness: impossible for adversary to impersonate Alice

Zero-knowledge: Anyone can sample (a, c, r) for themselves

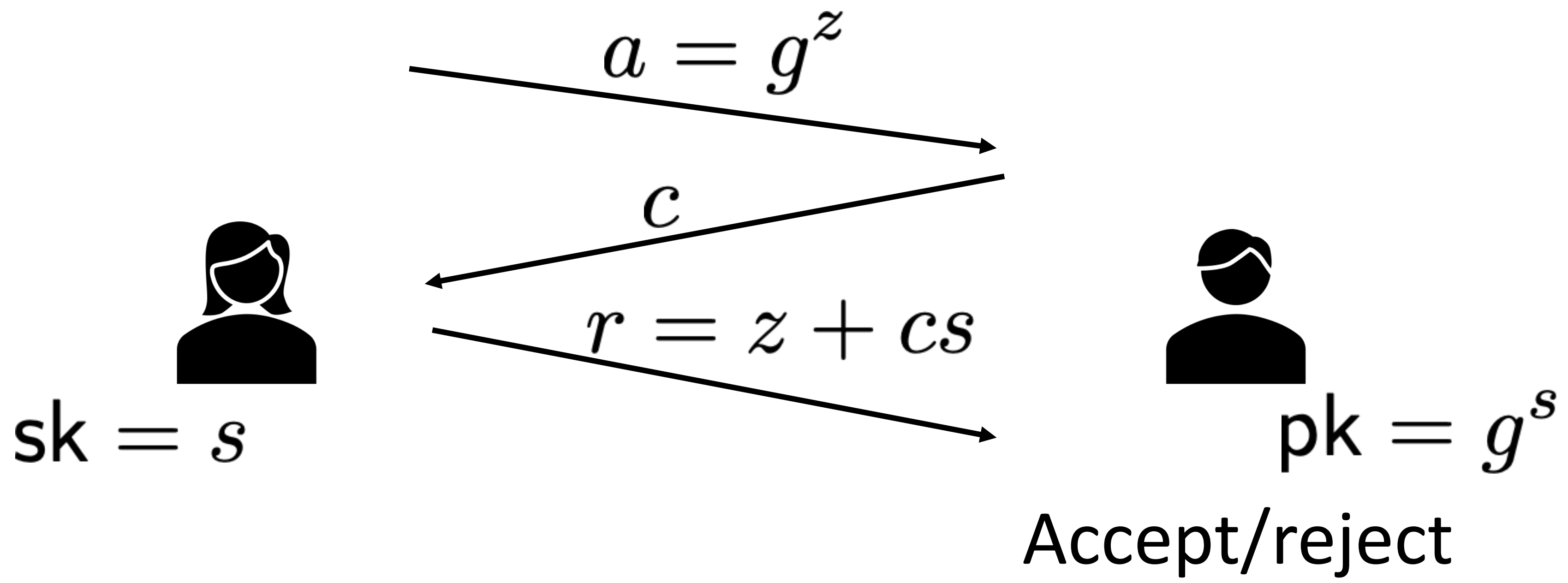
Schnorr Identification



Lemma: Under Dlog assumption, protocol is sound

Proof idea: Dlog implies hard to find $(a, c_1, r_1), (a, c_2, r_2)$ with $c_1 \neq c_2$. Using rewinding to extract such a “collision” from an impersonator

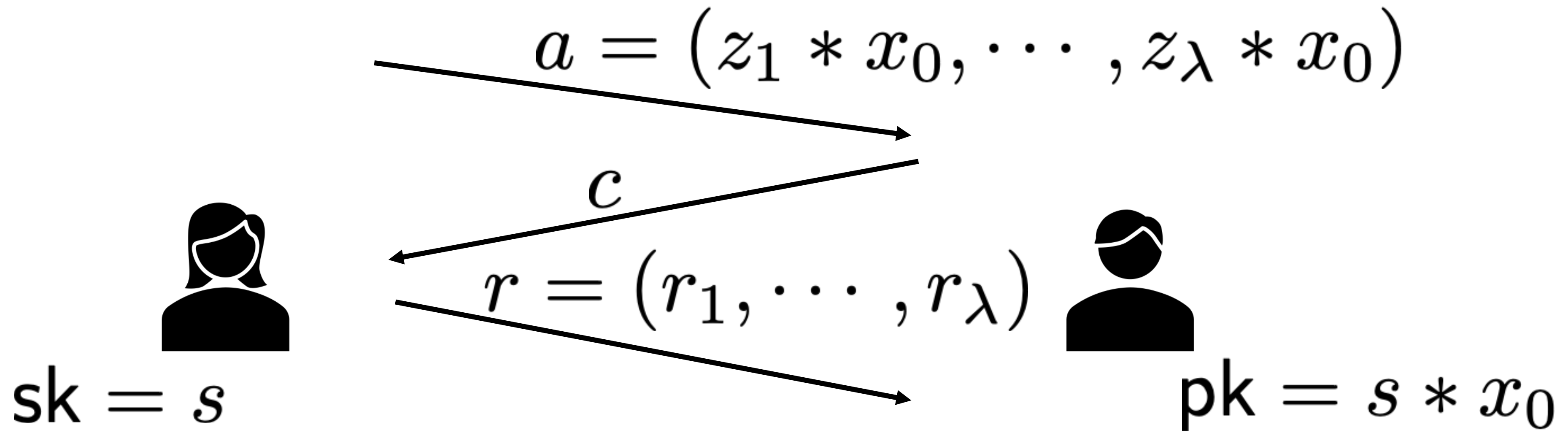
Schnorr Identification



Lemma: Protocol is zero knowledge

Proof: Choose random c, r, set $a = g^r \times \mathsf{pk}^{-c}$

Group Action-based Identification


$a = (z_1 * x_0, \cdots, z_\lambda * x_0)$
c
$r = (r_1, \cdots, r_\lambda)$
$\mathsf{sk} = s$
$\mathsf{pk} = s * x_0$

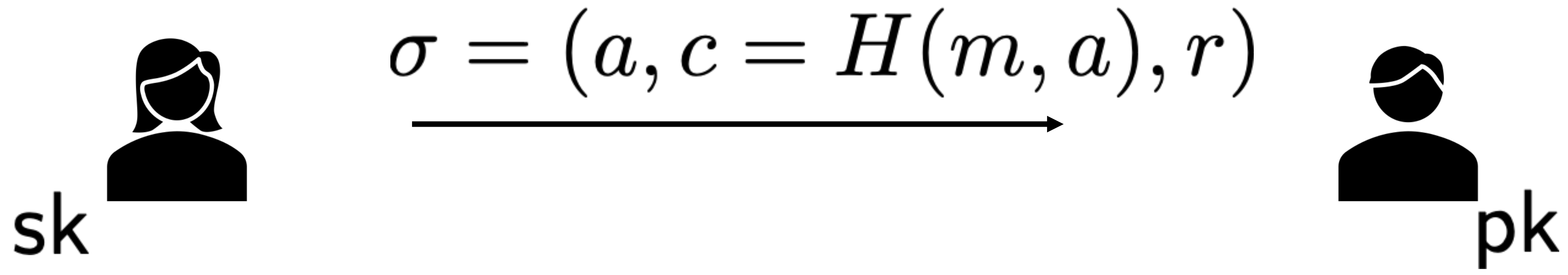
Accept/reject

$$r_i = (1 - c_i) z_i \text{``}+\text{''} c_i s$$

Lemma: Under GA-Dlog assumption, protocol is sound

Lemma: Protocol is zero knowledge

The Fiat-Shamir Transform



Shnorr signatures = Schnorr identification + Fiat-Shamir

GA-based signatures = GA-based identification + Fiat-Shamir

Thm (informal): Assuming Sigma protocol is sound, impossible for efficient **classical** adversary to forge a signature in ROM

Proof idea:

$$\begin{tikzcd} O \arrow[r, shift left=2, leftarrow, "{(m_i, a_i)}"] \arrow[r, shift right=2, "c_i"'] & \text{😈} \arrow[r, "{\sigma = (a, c = O(m,a), r)}"] & {} \end{tikzcd}$$

Assume without loss of generality that (m, a) was some query

Choose random query $i \in [1, q+1]$

Choose random function O'

For all queries except i, answer with

$$(m_j, a_j) \mapsto O'(m_j, a_j)$$

For $j = i$, send a_i, receive c_i

re-program $O'(m_i, a_i) = c_i$

answer with c_i

(future queries use re-programmed oracle)

Upon output $\sigma = (a, c, r)$, send r

$\xrightarrow{a_i}$

$\xleftarrow{c_i}$

$\xrightarrow{r}$

Thm (informal): Assuming Sigma protocol is sound, impossible for efficient **classical** adversary to forge a signature in ROM

Proof idea:

If we happen to correctly guess i to be the first time (m, a) was queried, then adversary sees truly random oracle O

$$O(m_i, a_i) = c_i$$

$$(m_j, a_j) = O'(m_j, a_j) \text{ for } j \neq i$$

Before query i, we were answering with $O'(m_i, a_i) \neq c_i$. But by assumption, no such query made

Thm (informal): Assuming Sigma protocol is sound, impossible for efficient **classical** adversary to forge a signature in ROM

Proof idea: Where to get O'? Problem: exponential-sized object

Solution: lazy sampling. Answer each query randomly, but keep track of previous queries to ensure same answer if same query made twice

Moving to quantum

Problem 1: how to simulate random oracle?

- "Keep track of previous queries" implies writing the queries down; observer effect $\rightarrow$ changes adversary's state

Problem 2: How to get a_i?

- Adversary's query is superposition. Perform a measurement?

Simulating Random Oracles

Lemma: A 2q-wise independent function is perfectly indistinguishable from a random function

Def: A family $\mathcal{H}$ of functions $h : \{0,1\}^n \to \{0,1\}^m$ is k-wise independent if, for all tuples of k *distinct* points

$$(x_1, \cdots, x_k) \in (\{0,1\}^n)^k$$

and all tuples of k (possibly non-distinct) points

$$(y_1, \cdots, y_k) \in (\{0,1\}^m)^k$$

We have that

$$\Pr_{h \leftarrow \mathcal{H}}[h(x_i) = y_i \forall i \in [k]] = 2^{-km}$$

Solving Problem 2

First idea: same as classical, but measure query i to get m_i, a_i

Choose random query $i \in [1, q+1]$

Choose random function O'

For all queries except i, answer with

$$U_{O'}$$

For $j = i$, **measure query to get** m_i, a_i

send a_i, receive c_i

re-program $O'(m_i, a_i) = c_i$

answer with reprogrammed $U_{O'}$

(future queries use re-programmed oracle)

Upon output $\sigma = (a, c, r)$, send r

a_i

c_i

r

Unfortunately, doesn’t work. By observer effect, measuring ith query messes up adversary’s state

Miraculously, however, a small change actually does work, despite observer effect

Choose random query $i \in [1, q+1]$, $\boxed{b \in \{0,1\}}$

Choose random function O'

For all queries except i, answer with

$$U_{O'}$$

For $j = i$, measure query to get m_i, a_i

send a_i, receive c_i $\xrightarrow{a_i}$ $\xleftarrow{c_i}$

$b = 0 \begin{cases} \text{re-program } O'(m_i, a_i) = c_i \\ \text{answer with re-programmed } U_{O'} \end{cases}$

(future queries use re-programmed oracle)

Upon output $\sigma = (a, c, r)$, send r $\xrightarrow{r}$

Choose random query $i \in [1, q+1]$, $\boxed{b \in \{0,1\}}$

Choose random function O'

For all queries except i, answer with

$$U_{O'}$$

For $j = i$, measure query to get m_i, a_i

send a_i, receive c_i $\quad \xrightarrow{a_i} \quad \xleftarrow{c_i}$

$$\boxed{b = 1 \begin{cases} \text{answer with existing } U_{O'} \\ \text{re-program } O'(m_i, a_i) = c_i \end{cases}}$$

(future queries use re-programmed oracle)

Upon output $\sigma = (a, c, r)$, send r $\quad \xrightarrow{r}$

Why on earth does this work?!

Possible to evaluate by direct calculation, but I’m going to attempt to give an intuitive explanation

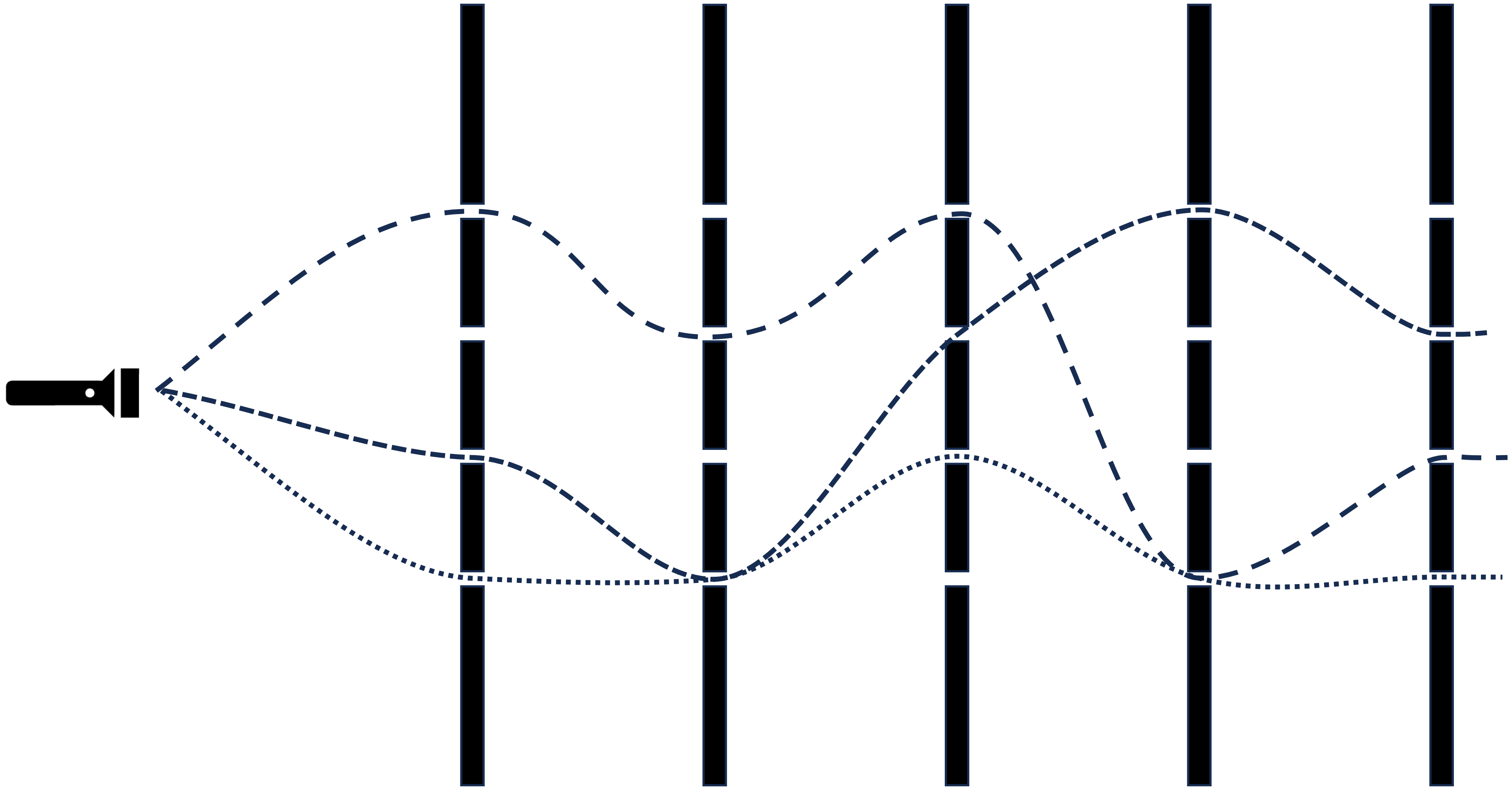
Phase queries

$$V_f|x,z\rangle = |x,z\rangle(-1)^{z\cdot f(x)}$$

$$\begin{aligned}
&(\mathbf{I}\otimes\mathbf{H}^{\otimes m})V_f(\mathbf{I}\otimes\mathbf{H}^{\otimes m})|x,y\rangle \\
&= (\mathbf{I}\otimes\mathbf{H}^{\otimes m})V_f|x\rangle\frac{1}{\sqrt{2^m}}\sum_z |x,z\rangle(-1)^{z\cdot y} \\
&= (\mathbf{I}\otimes\mathbf{H}^{\otimes m})\frac{1}{\sqrt{2^m}}\sum_z |x,z\rangle(-1)^{z\cdot(y\oplus f(x))} \\
&= |x, y\oplus f(x)\rangle = U_f|x,y\rangle
\end{aligned}$$

So we can assume adversary is making phase queries

A path view



Each wall corresponds to phase query

Each slit corresponds to basis state $|(m_j, a_j), z_j\rangle$

For a path P, define $\mathsf{Parity}(P)$ as function

$$\mathsf{Parity}(P)(m_j, a_j) = \bigoplus_{((m_j, a_j), z_j) \in P} z_j$$

General computation: $\mathbf{U}_q V_O \ \mathbf{U}_{q-1} \ V_O \ \cdots \ \mathbf{U}_2 \ V_O \ \mathbf{U}_1|0\rangle$

V_O imparts a phase to each path equal to

$$(-1)^{\sum_{m,a} O(m,a)\cdot \mathsf{Parity}(P)(m,a)}$$

Key observations:

- Initially, $\mathsf{Parity}(P)(m,a) = 0$ everywhere
- If $\mathsf{Parity}(P)(m,a) = 0$, adversary learns nothing about $O(m,a)$
- So we can assume at end, adversary's paths have
  $$\mathsf{Parity}(P)(m,a) \neq 0$$
- All paths must have transitioned from zero to non-zero at some query

Another Gentle Measurement Lemma: if an intermediate measurement gives only t possible outcomes,

$$\Pr[x|\text{with measurement}] \geq \Pr[x|\text{without measurement}]/t$$

$$\sum_y |\alpha_{x,y}|^2 \geq \left|\sum_y \alpha_{x,y}\right|^2 /t$$

Constructive interference can only amplify by the the number of different paths

Idea: if we measure a query and get (m, a), we are only separating paths by the query number that went from

$$\mathsf{Parity}(P)(m, a) = 0$$

to $$\mathsf{Parity}(P)(m, a) \neq 0$$

Intuitively, if this partitions the paths, then we can apply Gentle Measurements to show that success probability didn't decrease too much

Define set of paths Q_i = set of paths P where

$\text{Parity}(P)(m,a) = 0$ just prior to query i

$\text{Parity}(P)(m,a) \neq 0$ just after to query i

(m,a) = final output

Notice that for paths in Q_i, can re-program $O(m,a)$ at query i

This is essentially what the first attempt was doing

Problem: some paths may be in multiple Q_i

Basically, if path goes back and forth between $\mathsf{Parity}(P)(m,a) = 0$ and $\mathsf{Parity}(P)(m,a) \neq 0$

So not a partition, and therefore cannot use Gentle Measurements

Define set of paths R_i = set of paths P where

$\mathsf{Parity}(P)(m,a) \neq 0$ just prior to query i

$\mathsf{Parity}(P)(m,a) = 0$ just after to query i

Notice that for paths in R_i, can re-program $O(m,a)$ immediately **after** query i

Also note that these paths must transition back to $\mathsf{Parity}(P)(m,a) \neq 0$ at some point

Observe: for each path P

$$\#\{i : P \in Q_i\} = 1 + \#\{i : P \in R_i\}$$

Second, correct attempt is measuring a path in $\{Q_i\}_i \cup \{R_i\}_i$

This works because the R_i exactly capture overcounting in Q_i

Note: This analysis assumes that we get the actual (m, a) when we measure. That is, that we guessed the correct query i where we transitioned from $\mathsf{Parity}(P)(m, a) = 0$ to $\mathsf{Parity}(P)(m, a) \neq 0$.

In reality, only correctly guess with probability $1/q$

Overall success probability: $O(1/q^2)$

Where did we use Zero Knowledge?

We didn’t; ZK is used to prove security even against an adversary that sees many signed messages

Intuitively, why did we need a new proof? After all, the model of security for signatures didn’t change (signature security is still just PPT vs QPT)

But it did! In the ROM/QROM, the model includes the queries made to the RO. For QROM, not only do we have to worry about quantum computing, but we also have to worry about superposition queries

Next time: quantum protocols, applications and limitations