CS 258: Quantum Cryptography

Mark Zhandry

Previously…

Thm: There exists a quantum algorithm that performs $O(\sqrt{2^n})$ evaluations of U_f, and finds an x such that $f(x) = 1$ with probability $1 - O(2^{-n})$

Quantum Period Finding

Thm: There exists a QPT algorithm making only a polynomial number of queries which solves the period-finding problem with overwhelming probability

Includes in particular Discrete Log and Factoring

Main tool: the Quantum Fourier Transform (QFT)

$$\mathsf{QFT}_q|x\rangle = \frac{1}{\sqrt{q}}\sum_{y=0}^{q-1} e^{i2\pi xy/q}|y\rangle$$

$$\mathsf{QFT}_q = \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & e^{i2\pi 1/q} & e^{i2\pi 2/q} & e^{i2\pi 3/q} & \cdots \\ 1 & e^{i2\pi 2/q} & e^{i2\pi 4/q} & e^{i2\pi 6/q} & \cdots \\ 1 & e^{i2\pi 3/q} & e^{i2\pi 6/q} & e^{i2\pi 9/q} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$$\begin{aligned}\mathsf{QFT}^{\otimes n}q\frac{1}{\sqrt{|\mathbb{H}|}}\sum_{g\in\mathbb{H}}|y+g \bmod q\rangle &= \frac{1}{\sqrt{|\mathbb{H}|q^n}}\sum_{w}|w\rangle\left(\sum_{g\in\mathbb{H}}e^{i2\pi(y+g)\cdot w/q}\right)\\ &= \frac{1}{\sqrt{|\mathbb{H}|q^n}}\sum_{w}|w\rangle e^{i2\pi y\cdot w/q}\left(\sum_{g\in\mathbb{H}}e^{i2\pi g\cdot w/q}\right)\\ &= \sqrt{\frac{|\mathbb{H}|}{q^n}}\sum_{w\in\mathbb{Z}_q^n/\mathbb{H}}|w\rangle e^{i2\pi y\cdot w/q}\end{aligned}$$

Unfortunately, all currently deployed public key cryptosystems rely on the hardness of either Discrete Log or Factoring

⬇

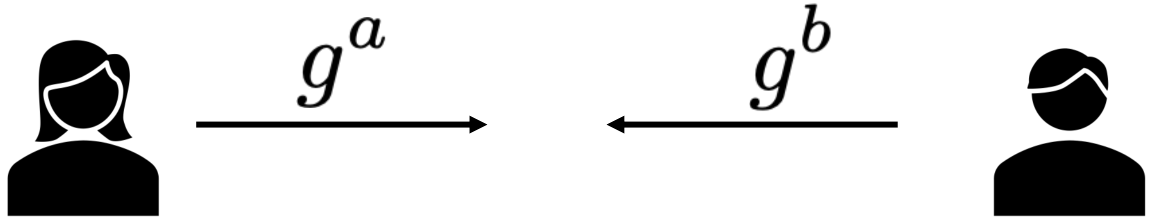
Basically all of our communication is broken once quantum computers are able to run Shor’s algorithm

Starting Today: Candidate Post-Quantum Cryptosystems

This Week: Group Actions

Recall Diffie-Hellman

$\mathbb{G}$ a cyclic group of order q with generator g



g^a $\longrightarrow$ $\longleftarrow$ g^b

$a \in \mathbb{Z}_q$ $b \in \mathbb{Z}_q$

$$K = g^{ab}$$

Recall Shor's Algorithm

Given $h = g^a$, define $f : \mathbb{Z}_q^2 \to \mathbb{G}$ $\quad f(x,y) = g^x h^y$

$$\begin{aligned} f(\ (x,y) + (ra, -r)\ ) &= g^{x+ra} h^{y-r} = g^{x+ra} g^{ay-ar} \\ &= g^{x+ay} = g^x h^y = f(x,y) \end{aligned}$$

Finding periods in f reveals a

Shor vs Diffie-Hellman

Shor requires group multiplication and discrete exponentiation ($x \mapsto g^x$) to compute $f(x,y) = g^x h^y$

Typically, discrete exponentiation is obtained from multiplication by repeated squaring

$$g^{13} = g \times (g^6)^2 = g \times ((g^3)^2)^2 = g \times ((g \times g^2)^2)^2$$

Diffie-Hellman “only” requires discrete exponentiation

What if we had a group where we could perform exponentiation but not multiplication?

Group Action

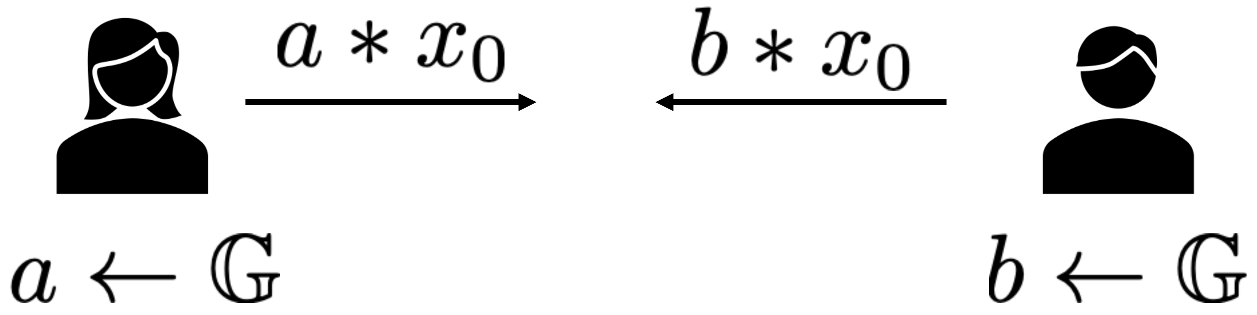
An (abelian) group action is a triple $(\mathbb{G}, \mathcal{X}, *)$ where:

- $\mathbb{G}$ is an (abelian) group
- $\mathcal{X}$ is a set
- $* : \mathbb{G} \times \mathcal{X} \to \mathcal{X}$ is an efficient binary operation satisfying

$$g * (h * x) = (gh) * x$$

- There is some element $x_0 \in \mathcal{X}$ that can be efficiently computed
- Usually ask that for each $x, y \in \mathcal{X}$, there exists a unique $g \in \mathbb{G}$ such that $y = g * x$
- Also usually ask that it is possible to efficiently identify elements of $\mathcal{X}$

Diffie-Hellman over Group Actions



$$K = (ab) * x_0$$

Supposedly Hard Problems on Group Actions

Discrete Log: computing a from $y = a * x_0$

Computational Diffie-Hellman: computing $(ab) * x_0$ from $a * x_0$ and $b * x_0$

Decisional Diffie-Hellman: Distinguishing $(ab) * x_0$ from $c * x_0$, given $a * x_0$ and $b * x_0$

Cyclic Groups as Group Actions

Given a cyclic group $\mathbb{G}$ of order q with generator g

Let $\mathbb{G}' = \mathbb{Z}_q$, $\mathcal{X}' = \mathbb{G}$, $x_0 = g$, and $a * x = x^a$

Then Group Diffie-Hellman ⬌ Group Action Diffie-Hellman

discrete log in $\mathbb{G}$ ⬌ discrete log in $(\mathbb{G}', \mathcal{X}', *)$

CDH in $\mathbb{G}$ ⬌ CDH in $(\mathbb{G}', \mathcal{X}', *)$

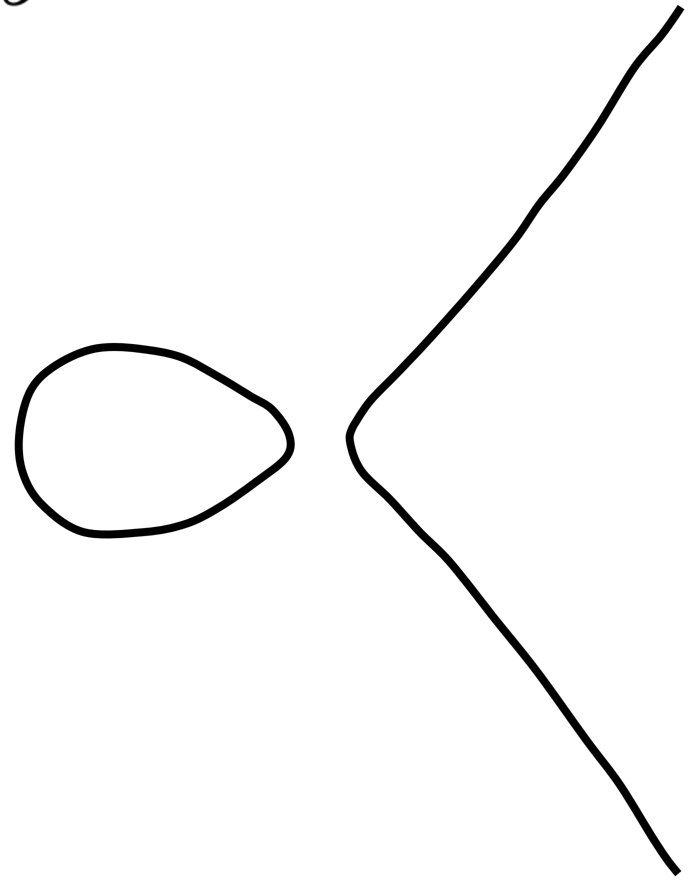
DDH in $\mathbb{G}$ ⬌ DDH in $(\mathbb{G}', \mathcal{X}', *)$

But group actions based on cyclic groups have quantumly-easy discrete logs

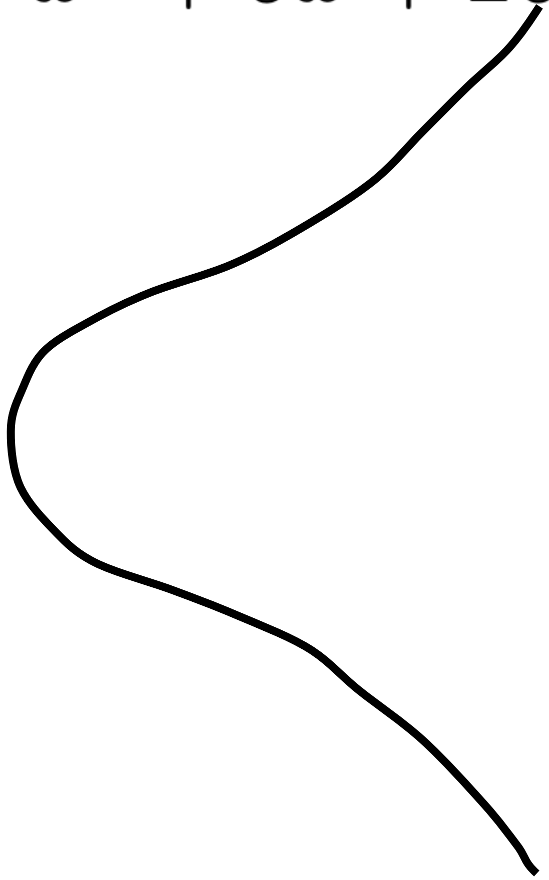
Instead, we use isogenies over elliptic curves

Elliptic curves

$y^2 = x^3 - 6x + 5$



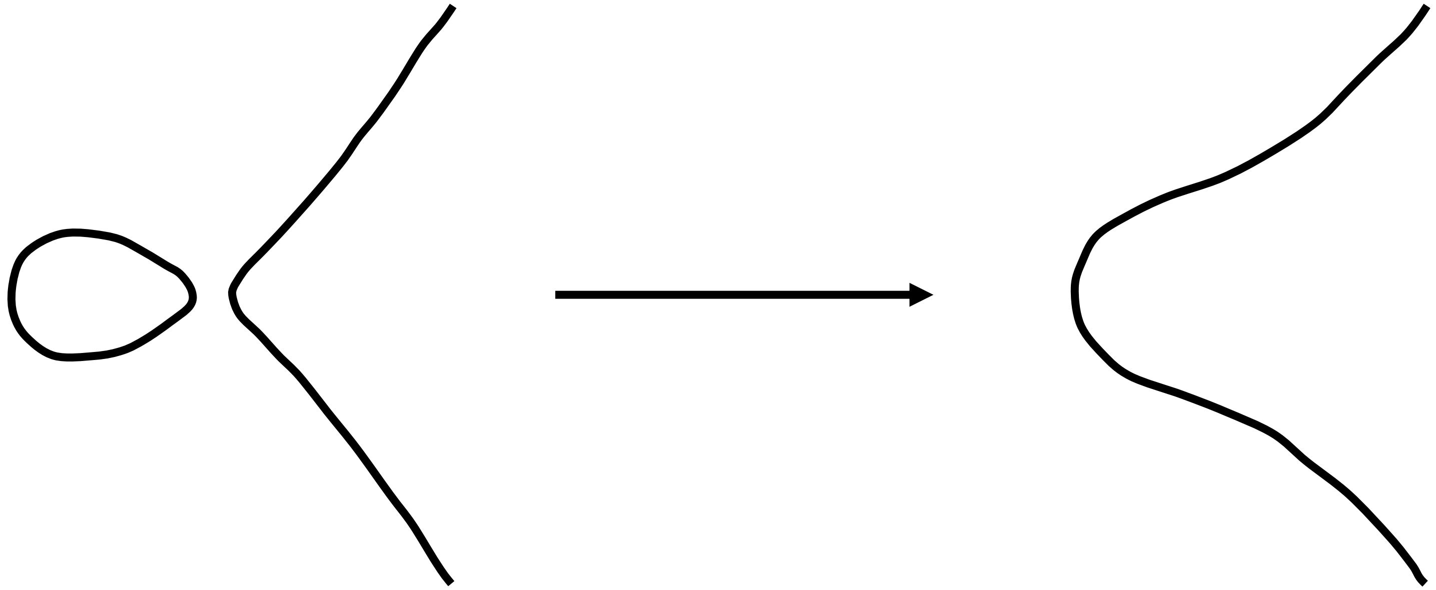
$y^2 = x^3 + 9x + 26$



For crypto, defined over $\mathbb{Z}_p$ or finite field $\mathbb{F}$

Isogenies

Rational maps between elliptic curves



$$(x, y) \mapsto (X, Y) = \left(\frac{x^2 - x - 3}{x - 1}, y \times \frac{x^2 - 2x + 4}{x^2 - 2x + 1} \right)$$

It turns out that these rational maps have a group structure (Ideal Class Group)

For “ordinary” elliptic curves, this group is abelian

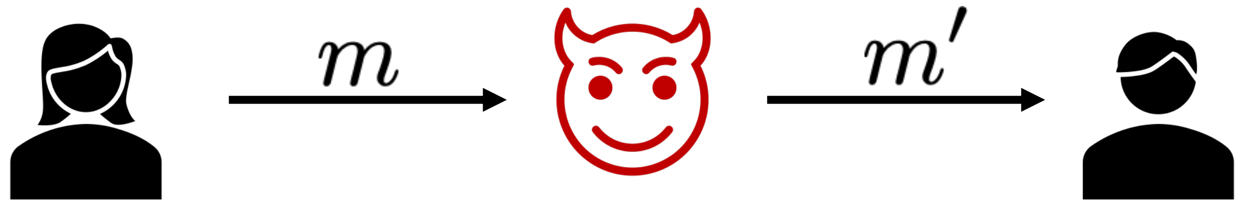
The Challenge with Group Actions

Group exponentiation	Group action
Group multiplication	✗ Combining set elements

The presumed post-quantum security of group actions derives from the inability to meaningfully combine set elements

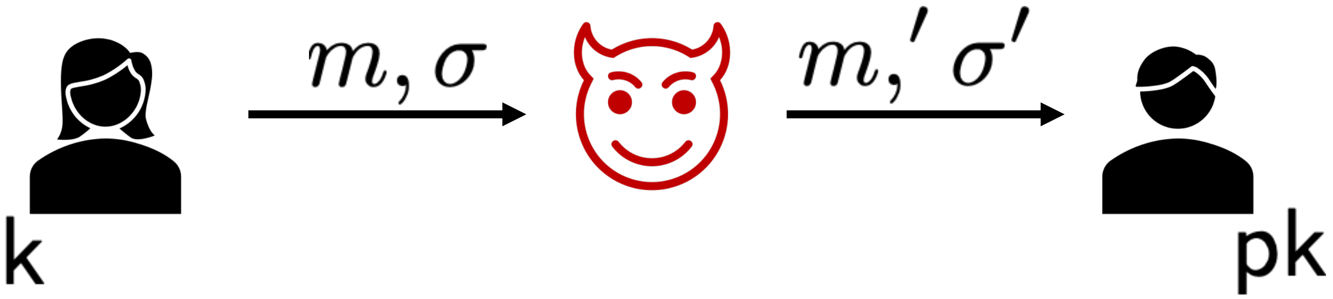
But, the inability to combine set elements also limits the kinds of cryptographic protocols we can build

Example: Signatures



Example: Signatures

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



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

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

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

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

$$h = g^r$$
$$c = H(m, h)$$
$$s = r + ca$$
$$\sigma = (h, c, s)$$

$\mathsf{Ver}(\mathsf{pk}, m, (h, c, s))$: check:

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

Intuition for security

$$h = g^r \qquad c = H(m, h) \qquad s = r + ca$$

$$\sigma = (h, c, s)$$

Challenge c determines linear function in unknowns (a, r)

a, r hidden in the exponent

One equation in two unknowns cannot be solved “in clear”

But can be verified “in the exponent” using group operations

Intuition for security

$$h = g^r \qquad c = H(m, h) \qquad s = r + ca$$

The hash enforces that challenge formed after m, h

Otherwise, pick s, c, let $h = g^s \mathsf{pk}^{-c}$,

find m s.t $c = H(m, h)$

by construction,
$g^s = h \times \mathsf{pk}^c$

Schnorr Signatures for Group Actions?

Assume a "good" hash function H

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

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

$$y = r * x_0$$

$$c = H(m, y)$$

$$s = r + ca$$???

$$\sigma = (y, c, s)$$

$\mathsf{Ver}(\mathsf{pk}, m, (y, c, s))$: check:

$$c = H(m, h)$$

$$s * x_0 = y \times (c * \mathsf{pk})$$

???

Muxing: A simple way to “combine” set elements

For a bit b, $x^b \text{“}\times\text{”} y^{1-b} = \begin{cases} y & \text{if } b = 0 \\ x & \text{if } b = 1 \end{cases}$

$$(g * x_0)^b \text{“}\times\text{”} (h * x_0)^{1-b} = (bg \text{“}+\text{”} (1-b)h) * x_0$$

Similar, define muxing for additive operations

Schnorr Signatures for Group Actions?

Assume a “good” hash function H

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

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

$$y = r * x_0$$

$$c = H(m, y)$$

$$c \in \{0,1\} \xrightarrow{} \quad s = (1-c)r \text{“}+\text{”} ca$$

$$\sigma = (y, c, s)$$

$\mathsf{Ver}(\mathsf{pk}, m, (y, c, s))$: check:

$$c = H(m, y)$$

$$s * x_0 = y^{1-c} \text{“}\times\text{”} \mathsf{pk}^c$$

Insecure! Recall Schnorr intuition

$$h = g^r \qquad c = H(m, h) \qquad s = r + ca$$

The hash enforces that challenge formed after m, h

Otherwise, pick s, c, let $h = g^s \mathsf{pk}^{-c}$,

find m s.t $c = H(m, h)$

by construction,
$g^s = h \times \mathsf{pk}^c$

Time to brute-force an m at most $\#\{c\}$

Increasing the muxing index

Assume a "good" hash function H

$$\mathsf{sk} = (a_1, \cdots, a_\lambda) \qquad \mathsf{pk} = (a_1 * x_0, \cdots, a_\lambda * x_0)$$

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

$$y = r * x_0$$

$$c = H(m, y)$$

$$s = r / a_i \quad (a_0 = 1)$$

$$\sigma = (y, c, s)$$

$c \in \{0, 1, 2, \cdots, \lambda\}$

$\mathsf{Ver}(\mathsf{pk}, m, (y, c, s))$: check:

$$c = H(m, y)$$

$$s * \mathsf{pk}_c = y$$

Parallel Repetition

Assume a “good” hash function H

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

$\mathsf{Sign}(\mathsf{sk}, m)$: choose random $r_1, \cdots, r_\lambda$

$$y_i = r_i * x_0$$

$$c = H(m, y_1, \cdots, y_\lambda)$$

$$s_i = (1 - c_i) r_i \text{“}+\text{”} c_i a$$

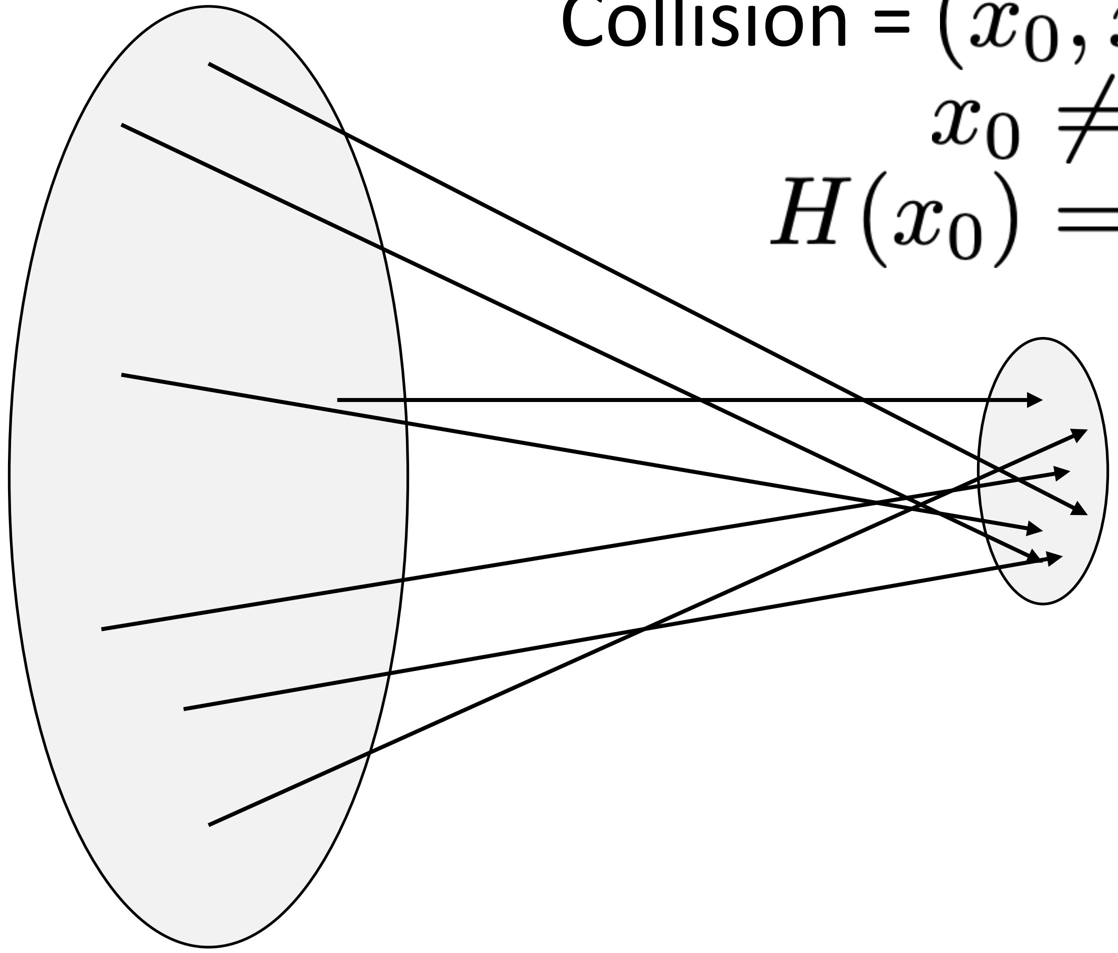
$$\sigma = (y_1 \cdots, y_\lambda, c, s_1, \cdots, s_\lambda)$$

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

Can combine both ideas to get a signature containing $O(\lambda/\log\lambda)$ set elements to have a challenge from a set of size 2^λ. Some formal evidence that this may be optimal, but still open

Compare to Schnorr, which only needs $O(1)$ elements

Example: Hashing



Collision = (x_0, x_1) s.t.

$$x_0 \neq x_1$$
$$H(x_0) = H(x_1)$$

Collision resistance: collisions exist in abundance, but impossible to find in polynomial time

Hashing from Groups

Let g, h be two group elements

$$H : \mathbb{Z}_q^2 \to \mathbb{G}$$

$$H(x, y) = g^x h^y$$

Claim: collision for H → a s.t. $g^a = h$

Proof: suppose $(x_0, y_0) \neq (x_1, x_1)$ but $g^{x_0} h^{y_0} = g^{x_1} h^{y_1}$

→ $g^{x_0 - x_1} = h^{y_1 - y_0}$ → $x_0 \neq x_1$ (why?)

→ $g^{\frac{x_0 - x_1}{y_1 - y_0}} = h$

Hashing from Groups

Let g, h be two group elements

$$H : \mathbb{Z}_q^2 \to \mathbb{G}$$

$$H(x, y) = g^x h^y$$

Let ℓ be bit-length used to represent group elements

H will be shrinking provided $\ell \ll 2 \log q$

For larger ℓ, can generalize to $H : \mathbb{Z}_q^k \to \mathbb{G}$

Hashing from Group Actions?

Let x_0, x_1 be two set elements

$$H : \mathbb{G} \times \{0,1\} \to \mathcal{X}$$

$$H(g,b) = g * x_b$$

polynomial

Can generalize to $H : \mathbb{G} \times \{0, 1, \cdots, k\} \to \mathcal{X}$

Claim: collision for H → a s.t. $a * x_0 = x_1$

Proof: suppose $(g_0, b_0) \neq (g_1, b_1)$ but $g_0 * x_{b_0} = g_1 * x_{b_1}$
→ $b_0 \neq b_1$ (why?) → can take $b_0 = 0, b_1 = 1$
→ $(g_0/g_1) * x_0 = x_1$

Hashing from Group Actions?

$$H : \mathbb{G} \times \{0, 1, \cdots, k\} \to \mathcal{X}$$

Let ℓ be bit-length used to represent set elements

Problem: in group actions based on isogenies, typically $\ell \approx 2 \log |\mathbb{G}|$

So H has input length $\log |\mathbb{G}| + \log k \ll \ell$

Thus, H is not compressing!

Hashing from Group Actions?

Some weak formal evidence that collision resistance is not possible from group action discrete log, but still very much open

Note: there are other hash function proposals using supersingular isogenies (not abelian), but based on different favor of hard computational problem

Next time: quantum algorithms for group actions