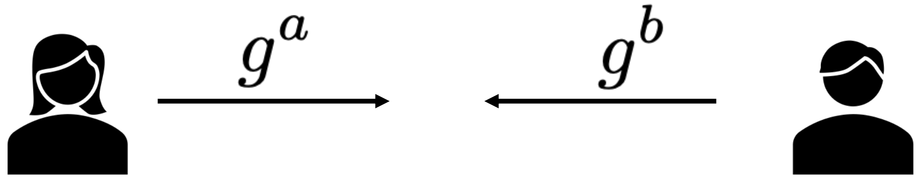
CS 258: Quantum Cryptography

Mark Zhandry

Previously…

Recall Diffie-Hellman

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



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

$$K = g^{ab}$$

Recall Shor's Algorithm

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

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

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

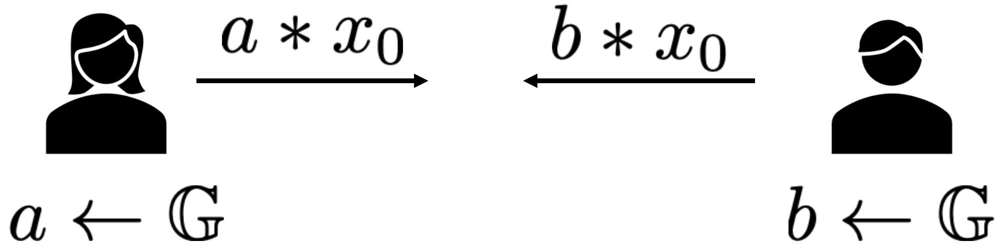
An (abelian) group action is a triple $(\mathbb{G}, \mathcal{X}, *)$ where:

- $\mathbb{G}$ 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



$$K = (ab) * x_0$$

Hashing from Groups

Let g, h be two group elements

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

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

Hashing from Group Actions?

Let x_0, x_1 be two set elements

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

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

Today: Kuperberg’s Algorithm

To make Kuperberg's algorithm easier to follow, we will switch to an additive notation for group actions

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

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

Nothing changes except notation

We are given $x_0, x_1 = a * x_0$, our goal is to find a

Let’s Try Attacking Group Actions Anyway

- Prepare $$\frac{1}{\sqrt{2|\mathbb{G}|}} \sum_{g\in\mathbb{G},b\in\{0,1\}} |g,b\rangle_{\mathcal{A}} |0^m\rangle_{\mathcal{B}}$$
- Apply U_H to obtain $$\frac{1}{\sqrt{2|\mathbb{G}|}} \sum_{g\in\mathbb{G},b\in\{0,1\}} |g,b\rangle_{\mathcal{A}} |g * x_b\rangle_{\mathcal{B}}$$
- Measure $\mathcal{B}$ → Measurement outcome z
  State collapses to $$\frac{1}{\sqrt{2}}|g,0\rangle + \frac{1}{\sqrt{2}}|g-a,1\rangle$$

$$g * x_0 = (g-a) * x_1 = z$$

$$\frac{1}{\sqrt{2}}|g,0\rangle + \frac{1}{\sqrt{2}}|g-a,1\rangle$$

Now apply QFT_q to the first register

$$\Downarrow$$

$$\frac{1}{\sqrt{2q}}\sum_h |h\rangle \left(e^{i2\pi gh/qgh}|0\rangle + e^{i2\pi(g-a)h/q}|1\rangle\right) = \frac{1}{\sqrt{2q}}\sum_h |h\rangle e^{i2\pi gh/q}\left(|0\rangle + e^{-i2\pi ah/q}|1\rangle\right)$$

$$\Downarrow$$ Measure first register

random $\rightarrow$

$$h,\ |\psi_h\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{-i2\pi ah/q}|1\rangle$$

Turns out a single sample doesn't contain enough information to find a

With polynomially-many samples, it turns out there is an *inefficient* algorithm to recover a

Kuperberg's algorithm: a tradeoff between samples and computation time

For simplicity, we will take q to be a power of 2, $q = 2^k$

Notice that if $k = 1$, $|\psi_h\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^{ah}|1\rangle$

$$\Big\downarrow$$

Apply $\mathbf{H}$: $|ah \bmod 2\rangle$

If h odd, learn least significant bit of a

Our goal then is to gradually reduce k until it's 1

Combining Samples

$$|\psi^k_{h_0}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{-i2\pi a h_0/2^k}|1\rangle \qquad |\psi^k_{h_1}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{-i2\pi a h_1/2^k}|1\rangle$$

$$\Big\Downarrow$$

We will show this momentarily

$$|\psi^k_{h_0 \pm h_1}\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{-i2\pi a (h_0 \pm h_1)/2^k}|1\rangle$$

If h_0, h_1 have same lsb, then $|\psi^k_{h_0 \pm h_1}\rangle = |\psi^{k-1}_{(h_0 \pm h_1)/2}\rangle$

$$|\psi^k_{h_0}\rangle|\psi^k_{h_1}\rangle = \frac{1}{2}\sum_{b_0,b_1}|b_0,b_1\rangle e^{-i2\pi a(b_0h_0+b_1h_1)/2^k}$$

Step 1: $$\text{CNOT}|\psi^k_{h_0}\rangle|\psi^k_{h_1}\rangle = \frac{1}{2}\sum_{b_0,b_1}|b_0,b_1\oplus b_1\rangle e^{-i2\pi a(b_0h_0+b_1h_1)/2^k}$$

$$= \frac{1}{2}\sum_{b_0,b_1}|b_0,c\rangle e^{-i2\pi a(b_0h_0+(c\oplus b_0)h_1)/2^k}$$

$$= \frac{1}{2}\sum_{b_0,b_1}|b_0,c\rangle e^{-i2\pi a(b_0h_0+(c+(-1)^c b_0)h_1)/2^k}$$

$$= \frac{1}{2}\sum_{b_0,b_1}|b_0,c\rangle(-1)^{-i2\pi ac/2^k}e^{-i2\pi ab_0(h_0+(-1)^c h_1)/2^k}$$

$$= \frac{1}{\sqrt{2}}\sum_{c}|\psi^k_{h_0+(-1)^c h_1}\rangle|c\rangle(-1)^{-i2\pi ac/2^k}$$

$$\frac{1}{\sqrt{2}}\sum_{c}|\psi^{k}_{h_0+(-1)^c h_1}\rangle|c\rangle(-1)^{-i2\pi ac/2^k}$$

Step 2: Measure c

$$\Downarrow$$

Random choice of $|\psi^{k}_{h_0 \pm h_1}\rangle$

Attempt 1

- Generate T random samples $h_i, |\psi^k_{h_i}\rangle$
- Partition samples into odd and even parities

Even

Odd

- Even's are already good

$$|\psi^k_{h_i}\rangle = |\psi^{k-1}_{h_i/2}\rangle$$

$$= |\psi^{k-1}_{(h_i \pm h_j)/2}\rangle$$

- Pair off odds and combine

$$(h_i|\psi^k_{h_i}\rangle), (h_j, |\psi^k_{h_j}\rangle) \mapsto h_i \pm h_j, |\psi^k_{h_i \pm h_j}\rangle$$

Attempt 1

Evens and odds will have $\approx T/2$ samples each

End samples will include all odds and half of evens

➡ $\approx 3T/4$ Samples after first iteration

Repeat k times → final number of samples $\approx (3/4)^k T$

Total running time: $\approx T \approx (4/3)^k$

Slightly better than Grover search, but still exponential

Kuperberg's algorithm

- Generate T random samples $h_i, |\psi^{\boldsymbol{k}}_{h_i}\rangle$
- Partition samples based on lowest r bits

Ends in 000 Ends in 001 Ends in 010 …

- Pair off within each set and combine

$$(h_i|\psi^{\boldsymbol{k}}_{h_i}\rangle), (h_j, |\psi^{\boldsymbol{k}}_{h_j}\rangle) \mapsto h_i \pm h_j, |\psi^{\boldsymbol{k}}_{h_i \pm h_j}\rangle$$

- Keep only the $h_i - h_j$ terms

Kuperberg’s algorithm

Since h_i, h_j agree on lowest r bits, $h_i - h_j$ ends in r zeros

$$|\psi^{k}_{h_i - h_j}\rangle = |\psi^{k-r}_{(h_i - h_j)/2^r}\rangle$$

After one step, have $\approx T/4$ samples

- Lose half because of merging
- Lose another half for the $+$ outcomes

How many samples do we need?

In each iteration, we need to fill each set with at least 2 samples

In particular, for the very last iteration, need $\approx 2^r$ samples

In each iteration, we only keep $1/4$ of samples

Each iteration reduces k by r → k/r iterations

Need to start with $T \approx 2^r \times 4^{k/r} = 2^{r+2k/r}$

Set $r = \sqrt{k}$ → $T \approx 2^{3\sqrt{k}} = 2^{O(\sqrt{\log q})}$

Thm [Kuperberg]: Dlog in (abelian) group actions can be solved in time $2^{O(\sqrt{\log q})}$, where q is the group order

Known as "subexponential" time

Impact on cryptography

Recall: want security against attacks running in time 2^{128}

Classical groups: can in principle set group size 2^{256}

Find collision in $f(x,y) = g^x h^y$ in time $\sqrt{q}$

by birthday paradox

Post-quantum group actions: need groups at least $2^{128^2} \approx 2^{16384}$

Results in much less efficient schemes

Tension between structure and security

Groups have lots of structure → lots of application, but also Shor's algorithm

Group actions have less structure, but some → fewer applications, resistant to Shor's, but subject to Kuperberg's

Other concepts may have even less structure → even fewer applications, but also fewer attacks

The Case of SIKE (Supersingular Isogeny Key Exchange)

Motivated to block Kuperberg’s algorithm, SIKE was proposed using isogenies over “supersingular” elliptic curves

These curves are not abelian group actions, so plausible exponential security $\rightarrow$ improved efficiency

But, since there is no commutation, also not clear how to do key exchange. Solution was for Alice,Bob to give each other extra hints

Turns out these hints completely broke the protocol

Broader Picture: Hidden Shifts

Kuperberg actually solves a much more general problem called hidden shift

Given $f_0, f_1 : \mathbb{G} \to \mathcal{X}$ injective, such that $f_1(g) = f_0(a+g)$, find a

($\mathbb{G}$ written additively)

Group action Dlog is a special case of hidden shift where

$$f_0(g) = g * x_0 \qquad f_1(g) = g * x_1 = (g+a) * x_0$$

An even broader picture: Non-abelian groups

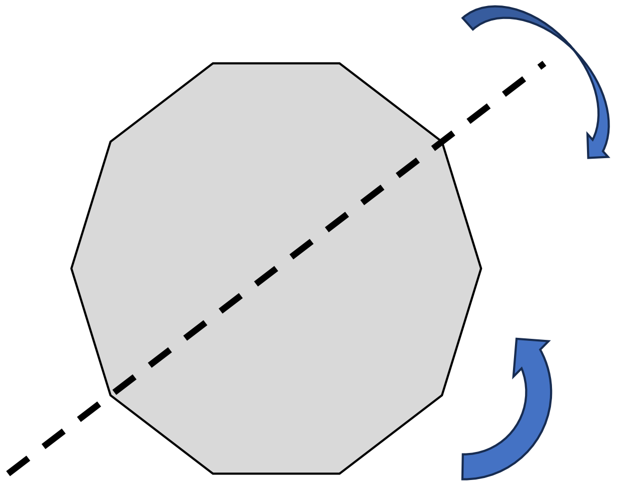
Recall: Shor’s algorithm solves the period-finding / hidden subgroup problem in abelian groups

Natural question: what about non-abelian groups?

Turns out that non-abelian groups seem much harder

An even broader picture: Non-abelian groups

Hidden shift can be seen as hidden subgroup for the **dihedral** group



$g * x_0$ $g * x_1$ is a rotation + reflection

An even broader picture: Non-abelian groups

Period-finding hardness:

Class of groups	Hardness
Abelian	Polynomial time (Shor)
Dihedral	Sub-exponential-time (Kuperberg)
General	Exponential

Proven upper bounds (attacks)
Conjectured lower bounds

A central challenge

Recall: the classical black box *query* complexity of abelian hidden subgroup *provably* was exponential

→ Gives exponential lower bound for “generic” classical algorithms

Not a proof of actual hardness, but a good heuristic to justify security

A central challenge

Want: a black box query lower-bound for group actions

Unfortunately, this is impossible as the query complexity is actually polynomial

A central challenge

$$h, \ |\psi_h\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}e^{-i2\pi ah/q}|1\rangle$$

Apply Hadamard and measure

$$\left.\begin{array}{l} \text{Prob 1} = \cos(ah\pi/q)^2 \\ \text{Prob 0} = \sin(ah\pi/q)^2 \end{array}\right\}$$

Noisily learn whether $ah \bmod q$ Is closer to 0 or $q/2$

With $\mathsf{polylog}(q)$ samples, a information-theoretically fixed

So we will never know for sure if group actions are even generically secure. But despite tons of interest, still no polynomial-time attack

Next time: lattices