Diffie-Hellman, Cyclic Groups and Discrete Logarithms

CMSC 28400, Autumn 2021
University of Chicago

Instructor: David Cash

Today: Discrete-Log Based Crypto

RSA Encryption: Needs *factoring* to be hard. Large N required to defeat algorithms.

Diffie-Hellman (and others): Needs another problem called *discrete logarithms* to be hard. With right choice of group, can get much smaller parameters than RSA.

Second half of lecture (not on exam): Zero-Knowledge Proofs

Diffie-Hellman Protocol Parameters

Prime p, integers $g \in \mathbb{Z}_p^*$ and $q > 1$.

```
2.2.  2048-bit MODP Group with 224-bit Prime Order Subgroup

   The hexadecimal value of the prime is:

   p =  AD107E1E 9123A9D0 D660FAA7 9559C51F A20D64E5 683B9FD1
        B54B1597 B61D0A75 E6FA141D F95A56DB AF9A3C40 7BA1DF15
        EB3D688A 309C180E 1DE6B85A 1274A0A6 6D3F8152 AD6AC212
        9037C9ED EFDA4DF8 D91E8FEF 55B7394B 7AD5B7D0 B6C12207
        C9F98D11 ED34DBF6 C6BA0B2C 8BBC27BE 6A00E0A0 B9C49708
        B3BF8A31 70918836 81286130 BC8985DB 1602E714 415D9330
        278273C7 DE31EFDC 7310F712 1FD5A074 15987D9A DC0A486D
        CDF93ACC 44328387 315D75E1 98C641A4 80CD86A1 B9E587E8
        BE60E69C C928B2B9 C52172E4 13042E9B 23F10B0E 16E79763
        C9B53DCF 4BA80A29 E3FB73C1 6B8E75B9 7EF363E2 FFA31F71
        CF9DE538 4E71B81C 0AC4DFFE 0C10E64F
```

```
The hexadecimal value of the generator is:

   g =  AC4032EF 4F2D9AE3 9DF30B5C 8FFDAC50 6CDEBE7B 89998CAF
        74866A08 CFE4FFE3 A6824A4E 10B9A6F0 DD921F01 A70C4AFA
        AB739D77 00C29F52 C57DB17C 620A8652 BE5E9001 A8D66AD7
        C1766910 1999024A F4D02727 5AC1348B B8A762D0 521BC98A
        E2471504 22EA1ED4 09939D54 DA7460CD B5F6C6B2 50717CBE
        F180EB34 118E98D1 19529A45 D6F83456 6E3025E3 16A330EF
        BB77A86F 0C1AB15B 051AE3D4 28C8F8AC B70A8137 150B8EEB
        10E183ED D19963DD D9E263E4 770589EF 6AA21E7F 5F2FF381
        B539CCE3 409D13CD 566AFBB4 8D6C0191 81E1BCFE 94B30269
        EDFE72FE 9B6AA4BD 7B5A0F1C 71CFFF4C 19C418E1 F6EC0179
        81BC087F 2A7065B3 84B890D3 191F2BFA

   The generator generates a prime-order subgroup of size:

   q =  801C0D34 C58D93FE 99717710 1F80535A 4738CEBC BF389A99
        B36371EB
```

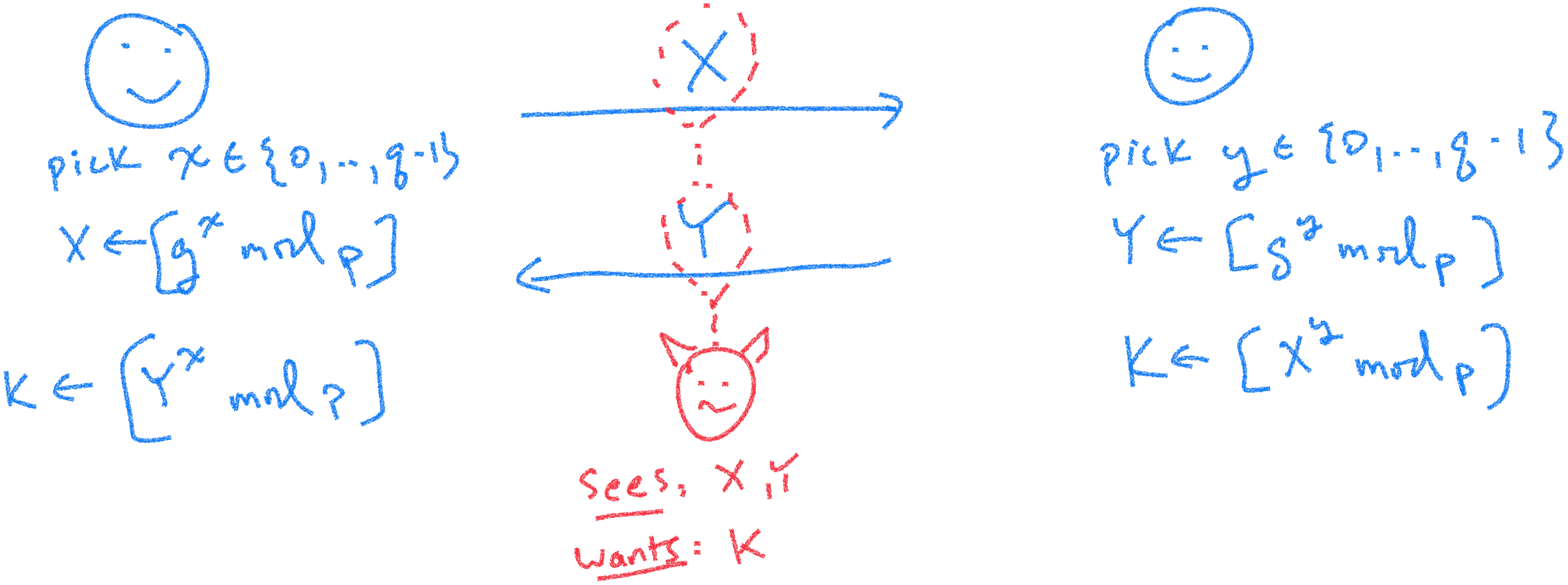
Diffie-Hellman Protocol

Prime p, integers $g \in \mathbb{Z}_p^*$ and $q > 1$.

pick $x \in \{0, \ldots, q-1\}$	$\xrightarrow{X}$	pick $y \in \{0, \ldots, q-1\}$
$X \leftarrow [g^x \bmod p]$	$\xleftarrow{Y}$	$Y \leftarrow [g^y \bmod p]$
$K \leftarrow [Y^x \bmod p]$		$K \leftarrow [X^y \bmod p]$

Diffie-Hellman Protocol

Prime p, integers $g \in \mathbb{Z}_p^*$ and $q > 1$.



Cyclic Groups Preliminaries

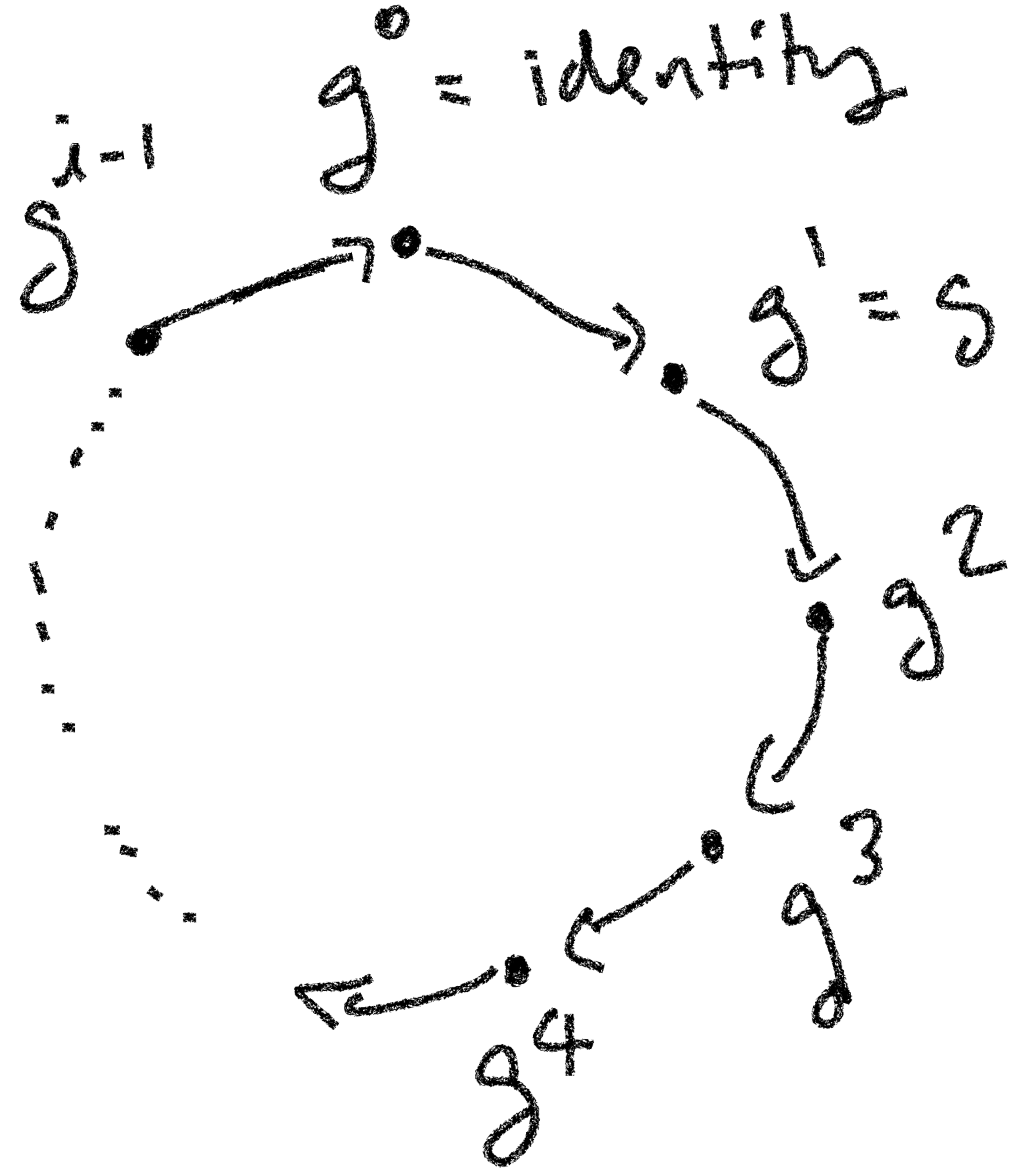
Definition: Let G be a finite group and let $g \in G$. Define

$$\langle g \rangle = \{g^0, g^1, g^2, \ldots\} \subseteq G.$$

Definition: Let G be a finite group and let $g \in G$. The smallest positive integer i such that $g^i = 1$ is called the *order of* g *(in* G*)*.

Cyclic Groups Preliminaries

$g^{[x \bmod i]} =$



Cyclic Groups Preliminaries

Lemma: Let G be a finite group and let $g \in G$ have order i. Then for all integers x,

$$g^x = g^{[x \mod i]}.$$

Proof:

Cyclic Groups Preliminaries

Lemma: Let G be a finite group of order m and let $g \in G$ have order i. Then $i \mid m$.

Proof:

Cyclic Groups Preliminaries

<u>Lemma</u>: Let G be a finite group of order m and let $g \in G$ have order i. Then $i \mid m$.

<u>Proof</u>: We show that $[m \bmod i] = 0$. Write $m = q \cdot i + [m \bmod i]$.

Then $e = g^m = g^{q \cdot i + [m \bmod i]} = (g^i)^q g^{[m \bmod i]} = g^{[m \bmod i]}$.

So $[m \bmod i]$ is a number in $\{0, \ldots, i-1\}$ such that $g^{[m \bmod i]} = e$.

But i is the smallest positive number such that $g^i = e$. So $[m \bmod i]$ must be zero

Cyclic Group Definition

<u>Definition</u>: Let G be a finite group. If $G = \langle g \rangle$ for some $g \in G$ then we say that G <u>*is cyclic*</u> and that g is a <u>*generator of G*</u> (or that <u>g *generates* G</u>).

<u>Examples</u>:

$\mathbb{Z}_5^*$ is cyclic with generator 2:

Prime-Order Groups are Cyclic

Theorem: Any prime-order group is cyclic.

Proof:

$\mathbb{Z}_p^*$ is Cyclic; Primitive Roots

Theorem: For every prime p, the group $\mathbb{Z}_p^*$ is cyclic.

(Note: $\mathbb{Z}_p^*$ has order $p - 1$, which is not prime for $p \neq 3$.)

Definition: Let p be prime. A *primitive root of p* is an integer x such that $[x \mod p]$ generates $\mathbb{Z}_p^*$.

Example: 2 is a primitive root of 5.

Discrete Logarithms

Claim: Let $G = \langle g \rangle$ be a finite cyclic group of order m. Then for any $h \in G$ there is a unique integer $x \in \{0,\ldots,m-1\}$ such that $h = g^x$.

Definition: In notation of the claim, define the function

$$\mathrm{dlog}_g : G \to \{0,\ldots,m-1\}$$

by $\mathrm{dlog}_g(h) = x$. This is called the *discrete logarithm of h with base g (in the group G).*

Example: In $\mathbb{Z}_9^*$, $\mathrm{dlog}_2(1) =$

The Discrete Logarithm Problem

<u>Informally</u>: Let $G = \langle g \rangle$ be a finite cyclic group. The *discrete logarithm problem in* G *with base* g is finding $\mathrm{dlog}_g(h)$, given g and h.

DLog Algorithm 1: Brute Force

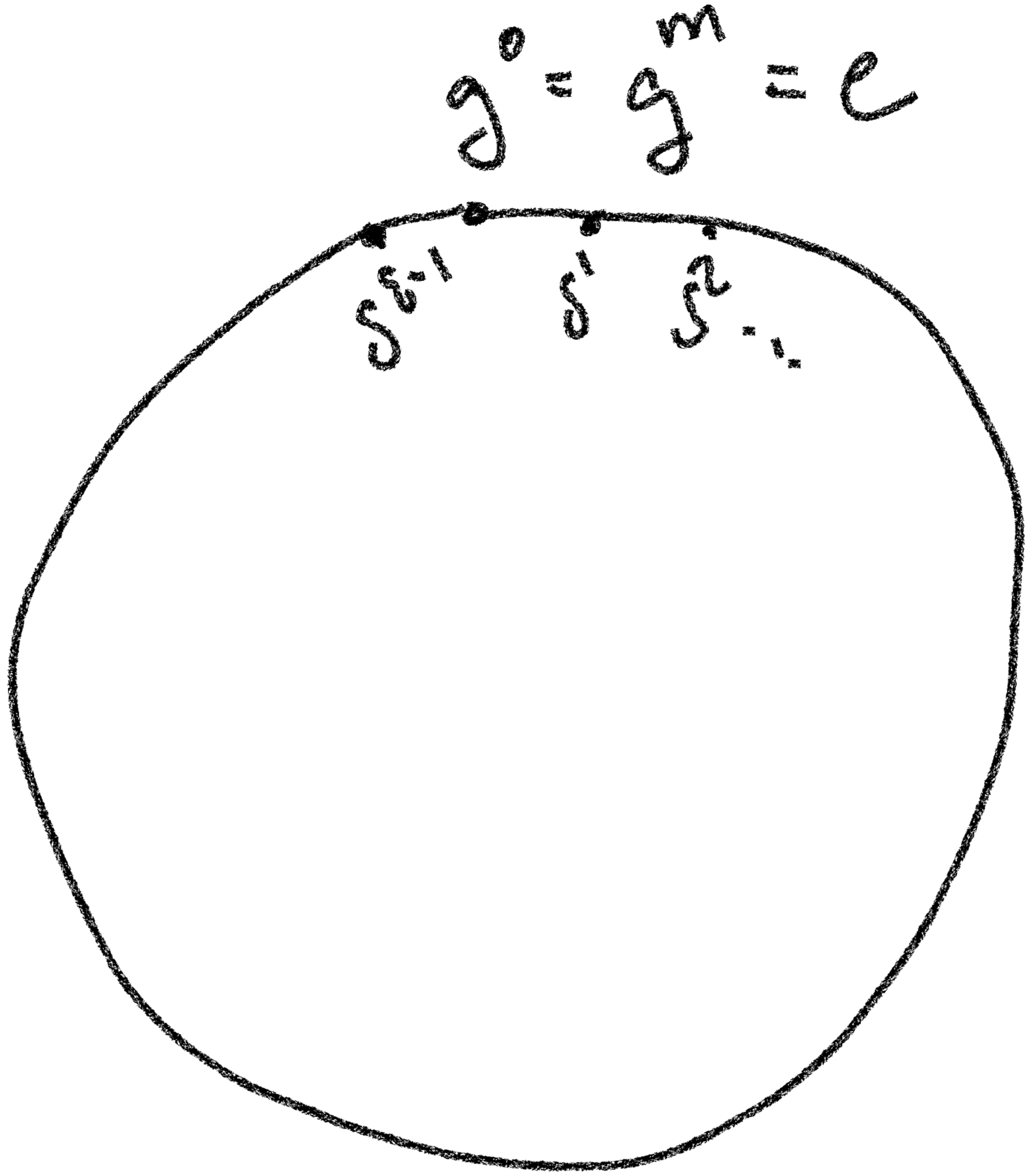
Alg. $A(h)$

For $y = 1, 2, \ldots, m - 1$:

If $g^y = h$: return y

DLog Algorithm 2: Baby-Step Giant Step

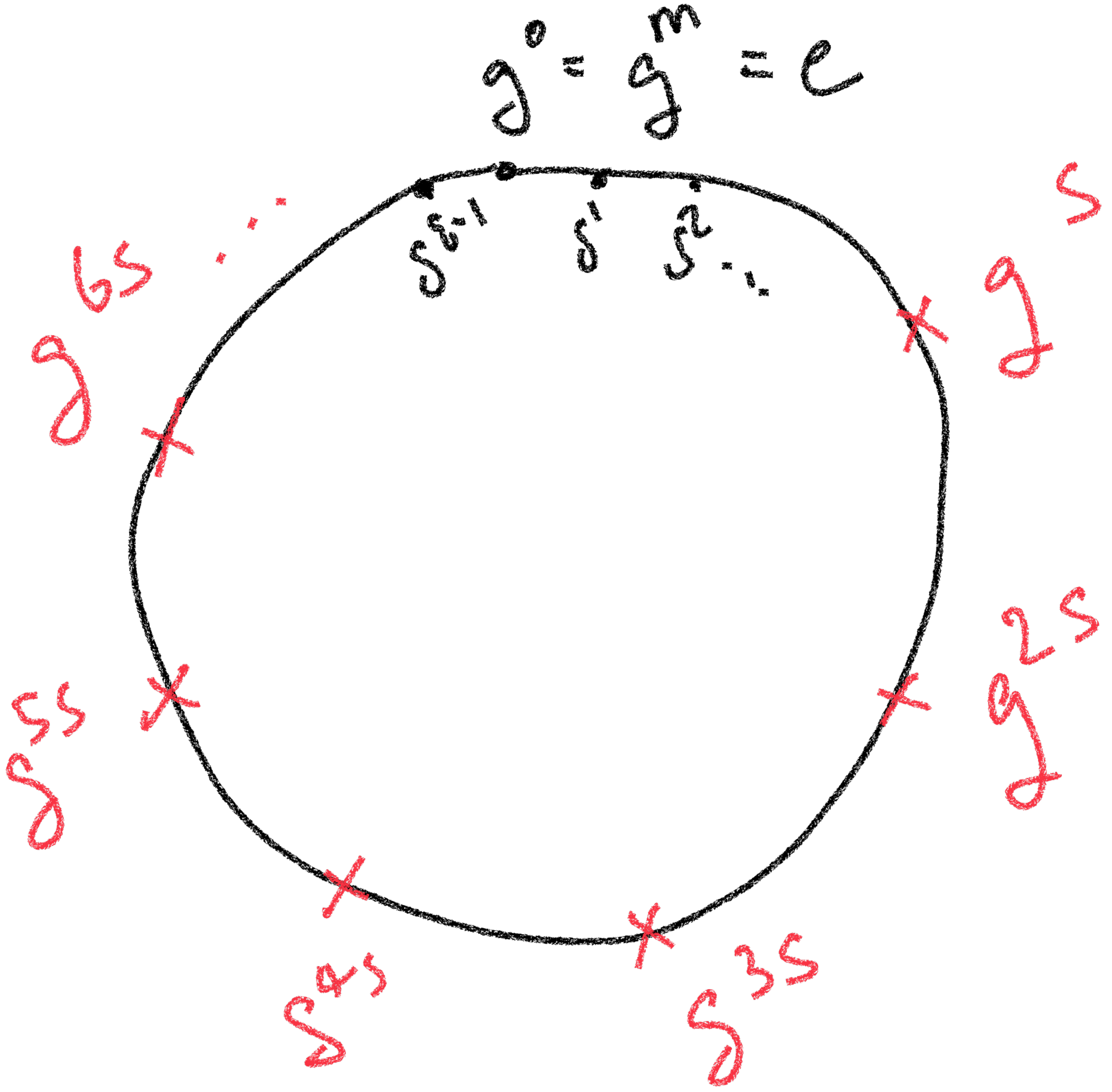
Step 1: "Giant Steps"



DLog Algorithm 2: Baby-Step Giant Step

Step 1: "Giant Steps"

Set $s = \lfloor \sqrt{m} \rfloor$

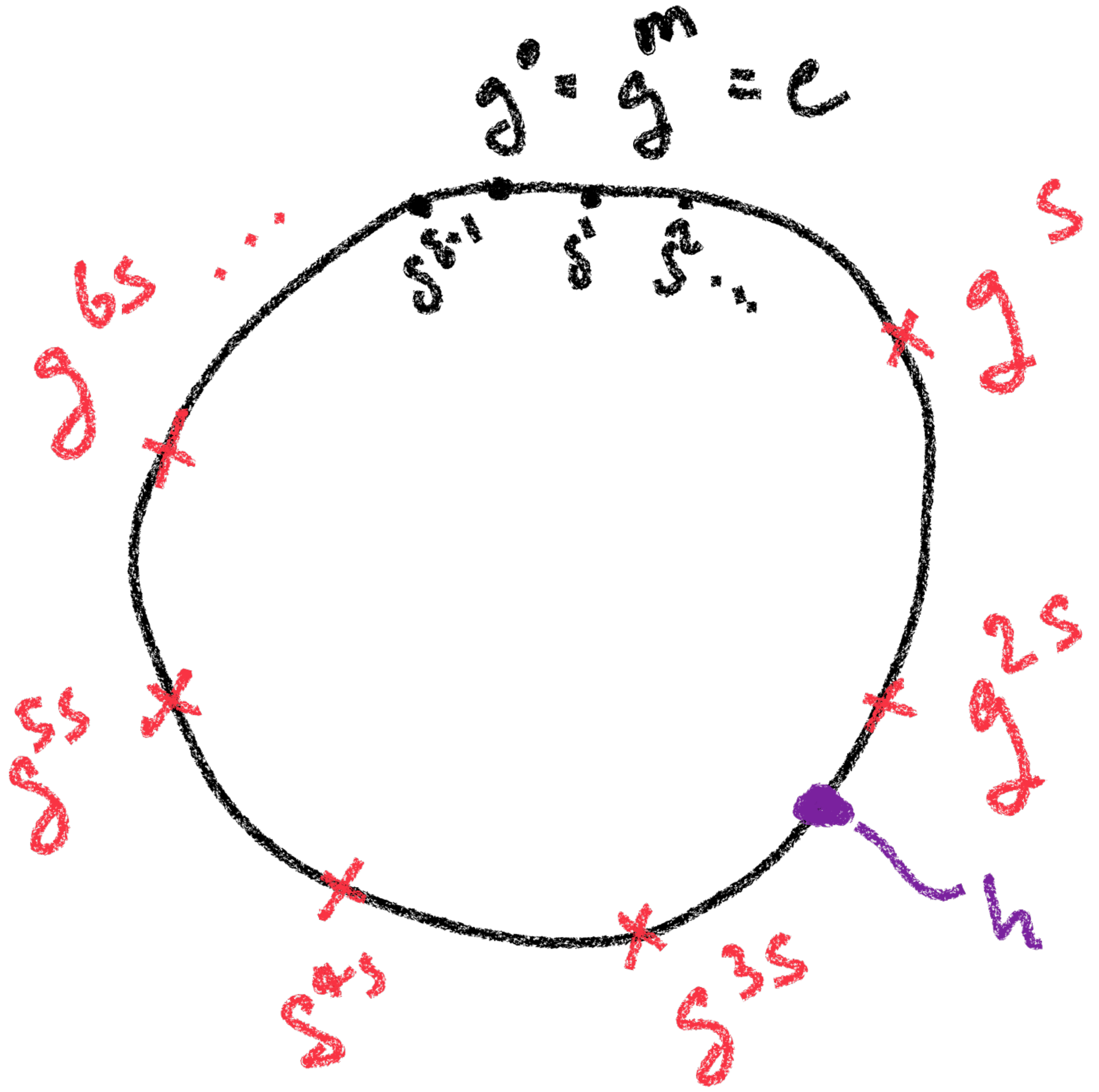


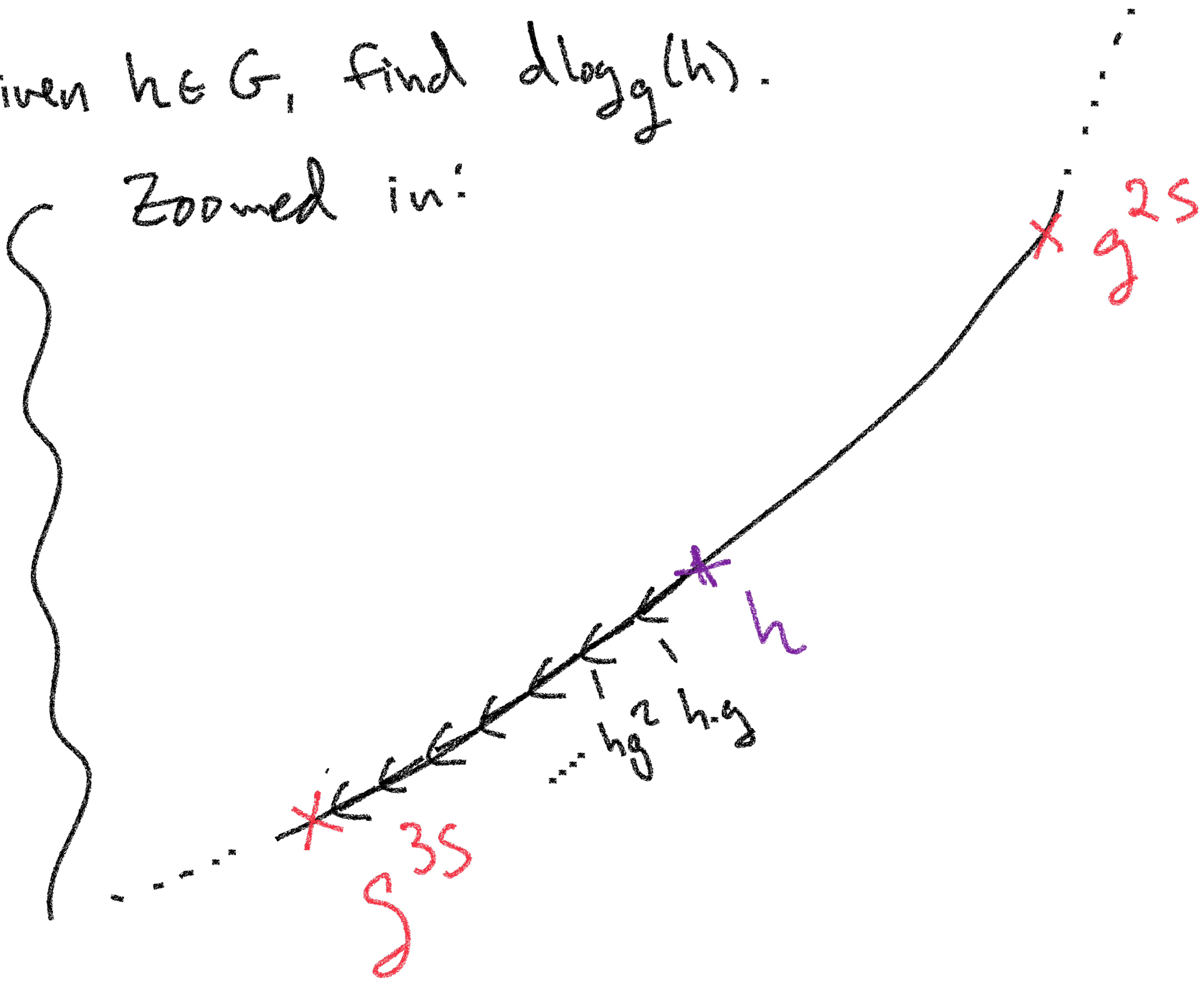
→ Save in hash table

→ About $\sqrt{m}$ steps

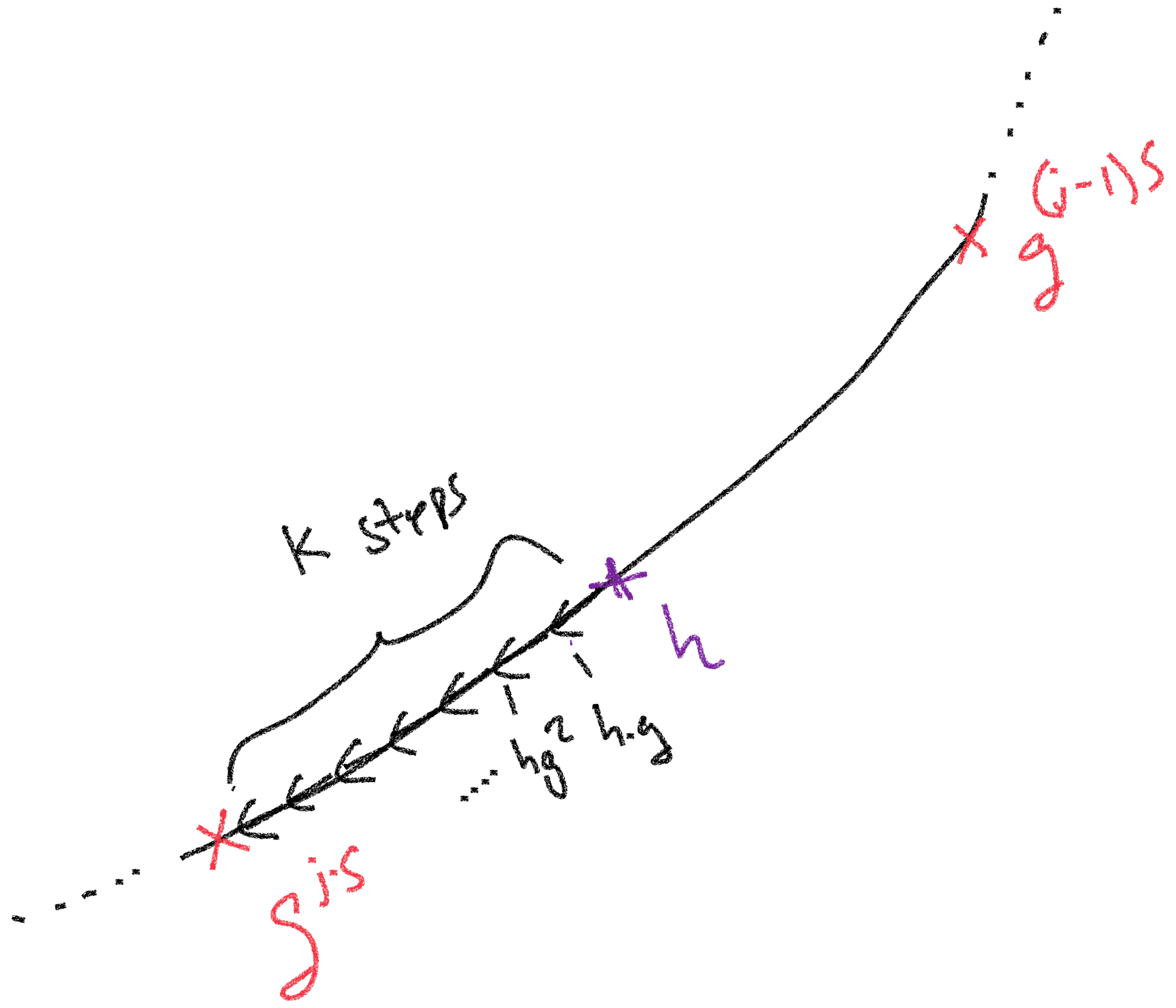
DLog Algorithm 2: Baby-Step Giant Step

Step 2: Baby Steps: Given $h \in G$, find $\mathrm{dlog}_g(h)$.





DLog Algorithm 2: Baby-Step Giant Step



Idea: Compute

$h \cdot g,\ h \cdot g^2,\ hg^3, \ldots$

until a giant step is hit

(it will be in hash table)

→ Then infer $\mathrm{dlog}_g(h)$:

$$h \cdot g^{K} = g^{j \cdot s} \Rightarrow h = g^{j \cdot s - K}$$

Code for Baby-Step-Giant-Step

Alg. $A(h)$

Initialize an empty hash table H

For $j = 1,2,\ldots,\lceil\sqrt{m}\rceil$:

$x \leftarrow g^{j\cdot\lceil\sqrt{m}\rceil}$

$H[x] \leftarrow j$

$y \leftarrow h$

For $k = 1,2,\ldots,\lceil\sqrt{m}\rceil$:

$y \leftarrow y \cdot g$

If $H[y] \neq \perp$: Output $j\lceil\sqrt{m}\rceil - k$

Run Time?

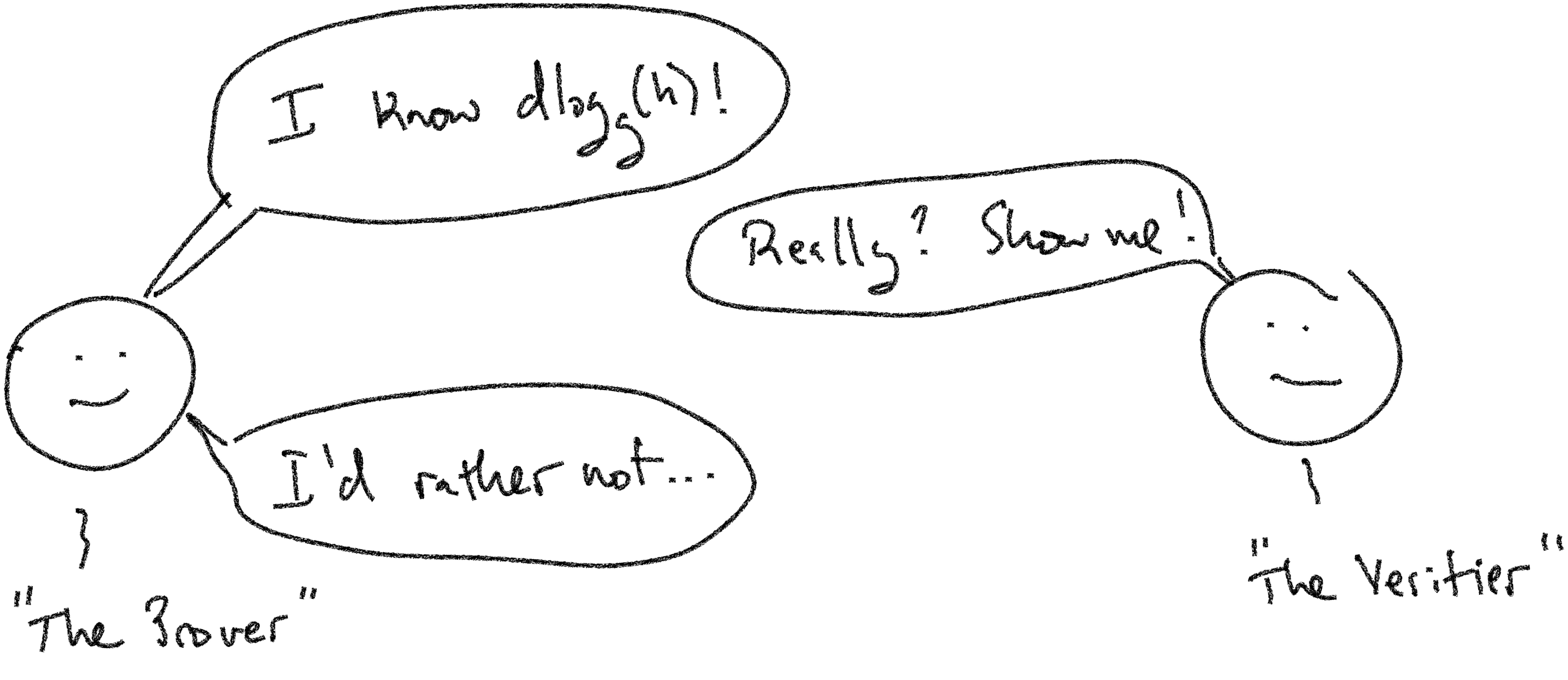
Conclusion: Selecting Diffie-Hellman Parameters

Prime p, integers $g \in \mathbb{Z}_p^*$ and $q > 1$.

- $g \in \mathbb{Z}_p^*$ chosen to generate group of order q
  - q chosen large enough to defeat Baby-Step-Giant-Step (ex: $q \approx 2^{256}$)
- p must be chosen *much larger* to defeat other attacks (similar to factoring)
  - $p \approx 2^{2048}$

Zero Knowledge Proofs: Proving with Explaining

- Group G, g, $g = \langle g \rangle$
- $h \in G$



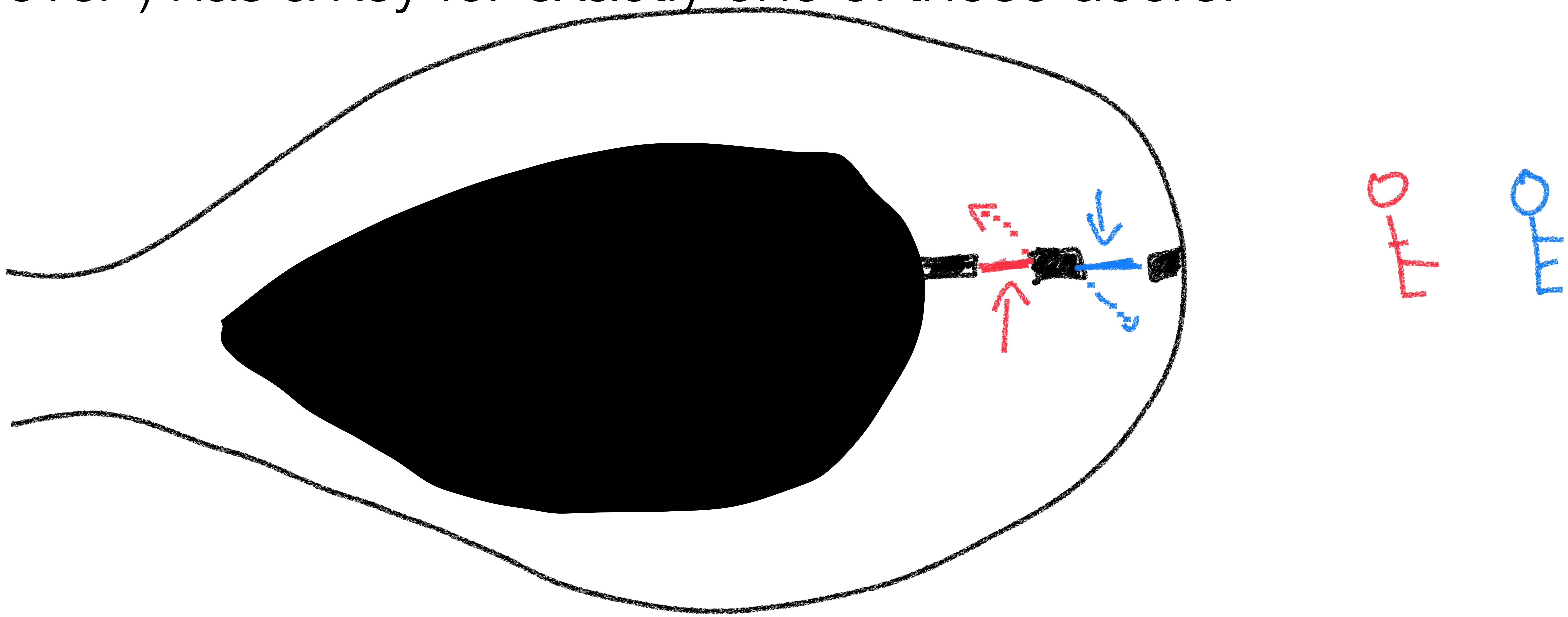
Zero-Knowledge Proofs: Example

<u>Set-up</u>: In the back of a circular cave, there are two locked doors:

Red door opens from bottom.

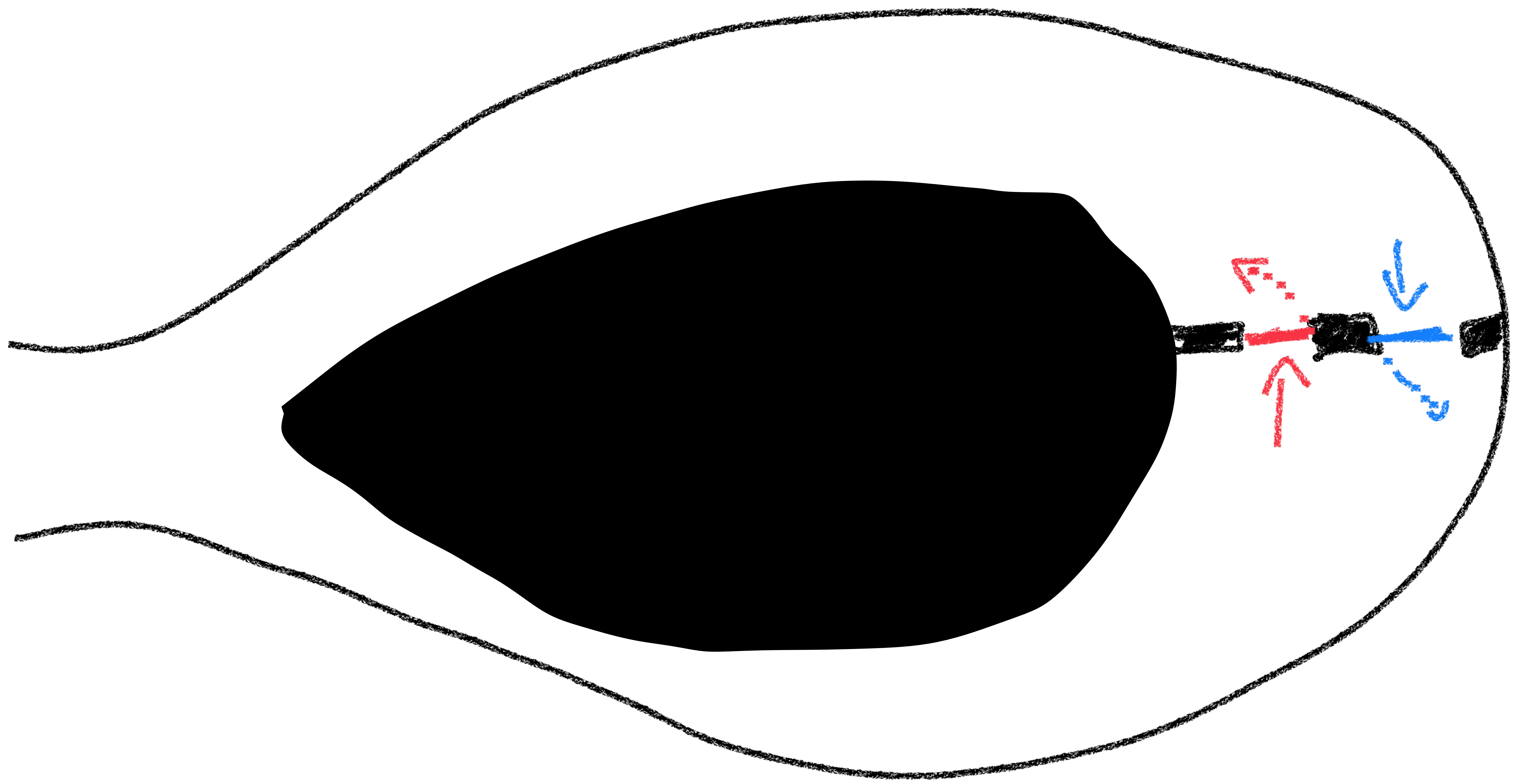
Blue door opens from top.

Person P (the “prover”) has a key for exactly one of those doors.



Zero-Knowledge Proofs: Example

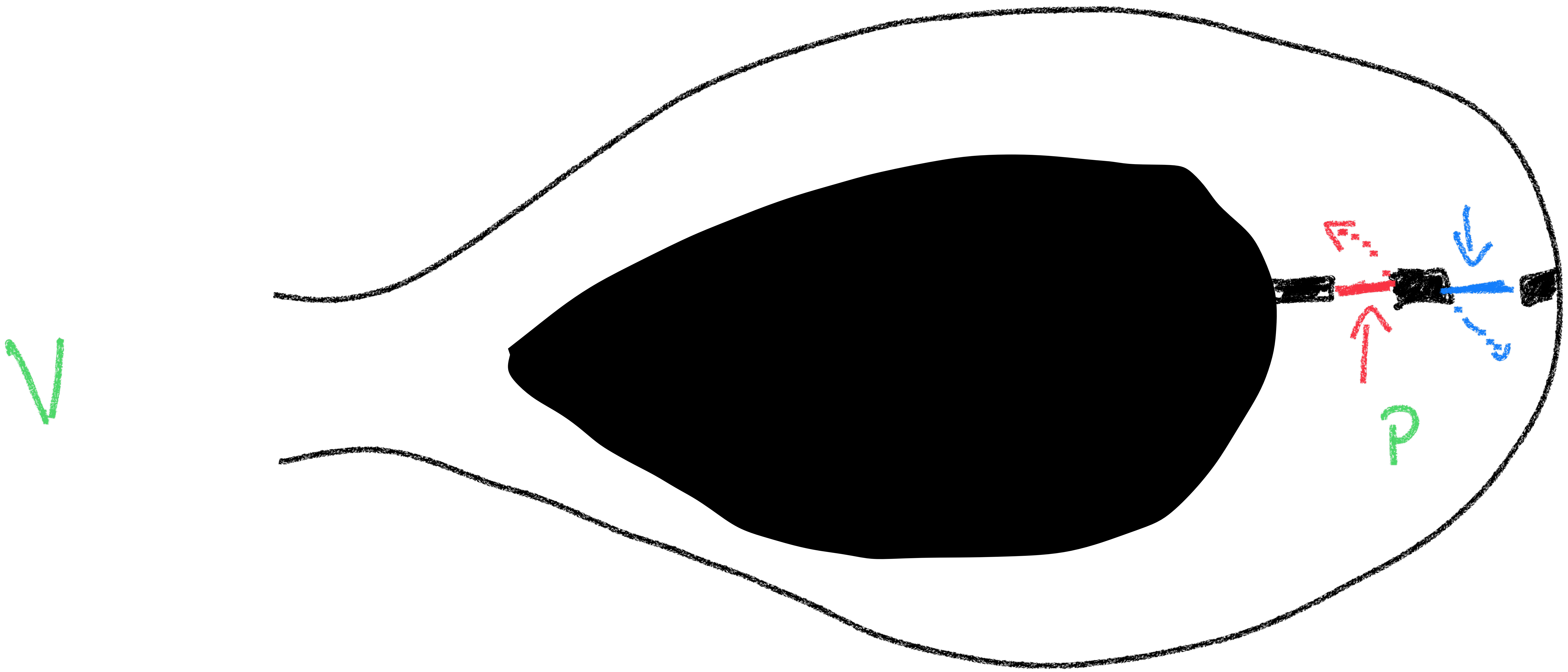
<u>Goal</u>: P wants to convince other person V (the "verifier") that they have a key to one of the doors, but not *which* key.



A Zero-Knowledge Proof Protocol

<u>Protocol</u>:

1. P goes into cave in direction they can open door
2. V flips a coin, and asks P to come out of cave from either top of bottom
3. If P comes out of correct side, V accepts. Otherwise, V rejects.



Proving Knowledge of a Discrete Logarithm

<u>**Common knowledge:**</u> Both P and V know a cyclic group G of order q, a generator $g \in G$ and another element $h \in G$.

<u>**Known to P only:**</u> The discrete logarithm of h with base g; Call it x. (So $h = g^x$).

<u>**Goal**</u>: P wants to convince V that they know x without revealing x.

A ZK Protocol for Discrete Logarithms

<u>Protocol</u>:

1. P picks $r \in \{0,\ldots,q-1\}$, sets $R \leftarrow g^r$, and sends R to V.
2. V flips a coin and sends bit b to P.
3. If $b = 0$: P sends r to V, and V checks that $g^r = R$.
4. If $b = 1$: P sends $x + r \bmod q$ to V, and V checks that $g^{x+r} = h \cdot R$.

A ZK Protocol for Discrete Logarithms

Protocol:

1. P picks $r \in \{0, \ldots, q-1\}$, sets $R \leftarrow g^r$, and sends R to V.
2. V flips a coin and sends bit b to P.
3. If $b = 0$: P sends r to V, and V checks that $g^r = R$.
4. If $b = 1$: P sends $x + r \bmod q$ to V, and V checks that $g^{x+r} = h \cdot R$.

$\underline{P(x)}$ $\qquad\qquad$ $\underline{V(h)}$

$r \in \{0, \ldots, q-1\}$

$R \leftarrow g^r$ $\qquad \xrightarrow{R}$

$\qquad\qquad \xleftarrow{\emptyset}$

$y \leftarrow r$ $\qquad \xrightarrow{y}$ $\qquad g^y \stackrel{?}{=} R$

$\underline{P(x)}$ $\qquad\qquad$ $\underline{V(h)}$

$r \in \{0, \ldots, q-1\}$

$R \leftarrow g^r$ $\qquad \xrightarrow{R}$

$\qquad\qquad \xleftarrow{1}$

$y \leftarrow r + x \bmod q$ $\qquad \xrightarrow{y}$ $\qquad g^y \stackrel{?}{=} h \cdot R$

Security for Verifier: Can P Cheat?

Protocol:

1. P picks $r \in \{0,\ldots,q-1\}$, sets $R \leftarrow g^r$, and sends R to V.
2. V flips a coin and sends bit b to P.
3. If $b = 0$: P sends r to V, and V checks that $g^r = R$.
4. If $b = 1$: P sends $x + r \bmod q$ to V, and V checks that $g^{x+r} = h \cdot R$.

If P can answer for both $b=0$ and $b=1$, ...

Security for Prover: Can V learn about x?

Protocol:

1. P picks $r \in \{0, \ldots, q-1\}$, sets $R \leftarrow g^r$, and sends R to V.
2. V flips a coin and sends bit b to P.
3. If $b = 0$: P sends r to V, and V checks that $g^r = R$.
4. If $b = 1$: P sends $x + r \bmod q$ to V, and V checks that $g^{x+r} = h \cdot R$.

More subtle ...