Lecture 9 – Diffie-Hellman key exchange II, computational aspects

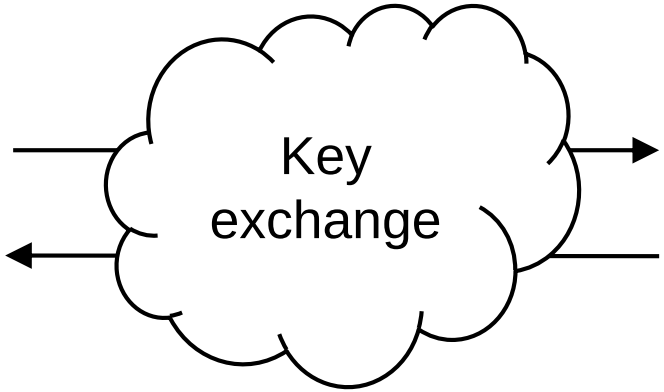
TEK4500

20.10.2021

Håkon Jacobsen

hakon.jacobsen@its.uio.no

Diffie-Hellman



Diffie-Hellman

$G = \langle g \rangle$ ← public

$A = g^a$ →

← $B = g^b$

$a \xleftarrow{\$} \{1, \dots, |G|\}$

$b \xleftarrow{\$} \{1, \dots, |G|\}$

$Z \leftarrow B^a = g^{ab}$

$Z \leftarrow A^b = g^{ab}$

Examples:

$G = (\boldsymbol{Z}_p^*, \cdot)$

$G = (E(\boldsymbol{Z}_p), +)$

Why is $\left(Z_p^*, \cdot\right)$ a group?

- $Z_p^* = \{1, 2, \ \ldots, p-1\}$
- Associativity ✓ $(a \cdot b) \cdot c = a \cdot (b \cdot c) \bmod p$
- Identity ✓ $1 \cdot a = a \cdot 1 = a \bmod p$
- Inverses ?

Definition: A **group** $(G, \circ)$ …

$\mathcal{G}1$: $(a \circ b) \circ c = a \circ (b \circ c)$ (associativity)
$\mathcal{G}2$: $\exists e \in G: e \circ a = a \circ e = a$ (identity)
$\mathcal{G}3$: $\exists a^{-1} \in G: \ a \circ a^{-1} = a^{-1} \circ a = e$ (inverses)

Given $a \in Z_p^*$ can we always find $a^{-1} \in Z_p^*$?

Need to solve: $x \cdot a = 1 \bmod p$

How do we actually find a^{-1}?
Extended Euclidian Algorithm

Claim: $\exists m, n \in Z$ such that $ma + np = 1 \implies ma = 1 - np = 1 \bmod p$

$\implies a^{-1} = m \bmod p \in Z_p^*$

Proof: $I = \{sa + tp \mid s, t \in Z\}$

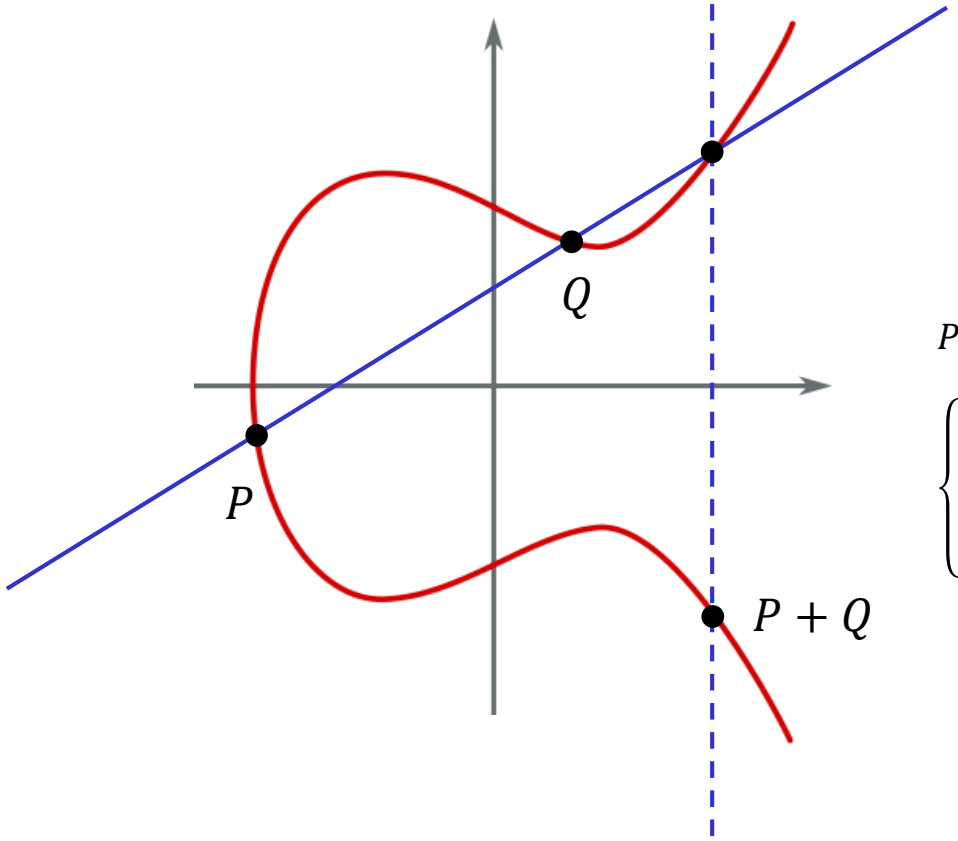
1) I contains a *positive* integer (e.g. a and p)
2) I contains a *smallest* positive integer $d = ma + np$
3) $p = qd + r \qquad 0 \le r < d$
4) $r = p - qd = p - q(ma + np) = -qma + (1 - qn)p \in I$ (since d is the smallest positive integer in I)
5) $r = 0$
6) $p = qd \implies d = 1$ (since p is prime and $d \le a < p$)
7) $1 = ma + np$

NON-CONSTRUCTIVE

Q.E.D

Elliptic curves

Theorem: the points on an elliptic curve, together with $\mathcal{O}$, is an abelian group under "geometric point addition"



$$y^2 = x^3 + ax + b$$

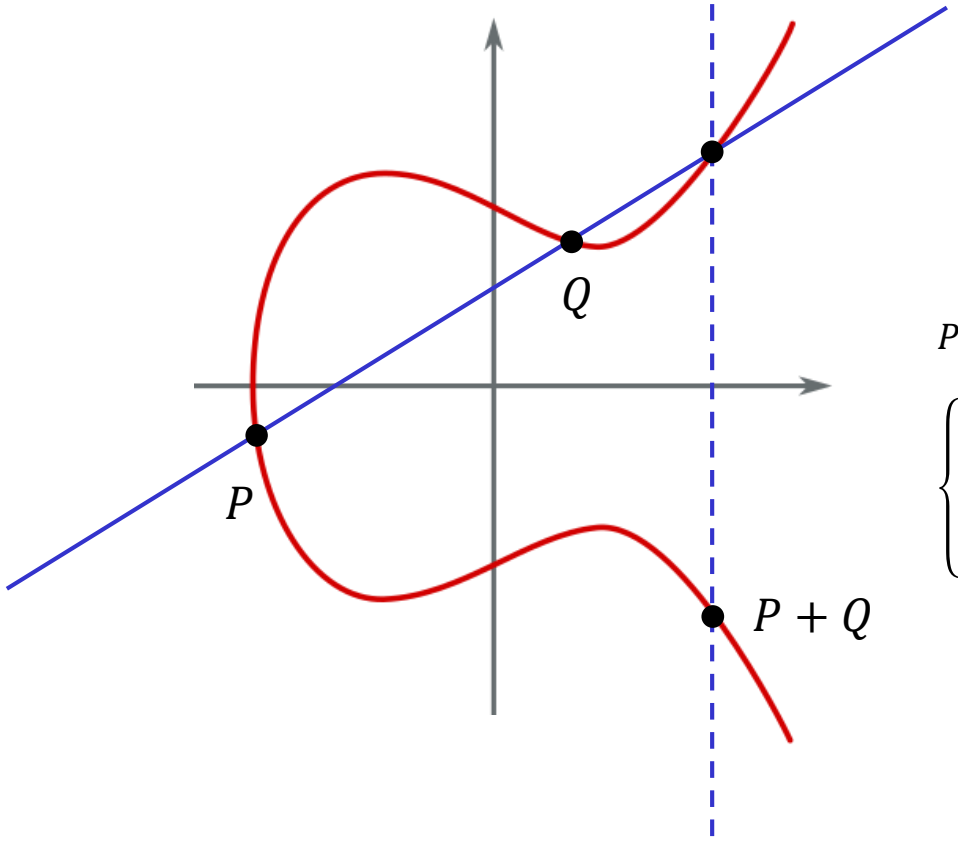
$$a, b, x, y \in \boldsymbol{R}$$

$$P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$\begin{cases} x_3 = \dfrac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b} \\ y_3 = \dfrac{x_1x_2(x_1 + x_2) - x_3\big((x_1 + x_2)^2 - x_1x_2 + a\big) - y_1y_2 - b}{y_1 + y_2} \end{cases}$$

Elliptic curves

Theorem: the points on an elliptic curve, together with $\mathcal{O}$, is an abelian group under "geometric point addition"



$$y^2 = x^3 + ax + b$$

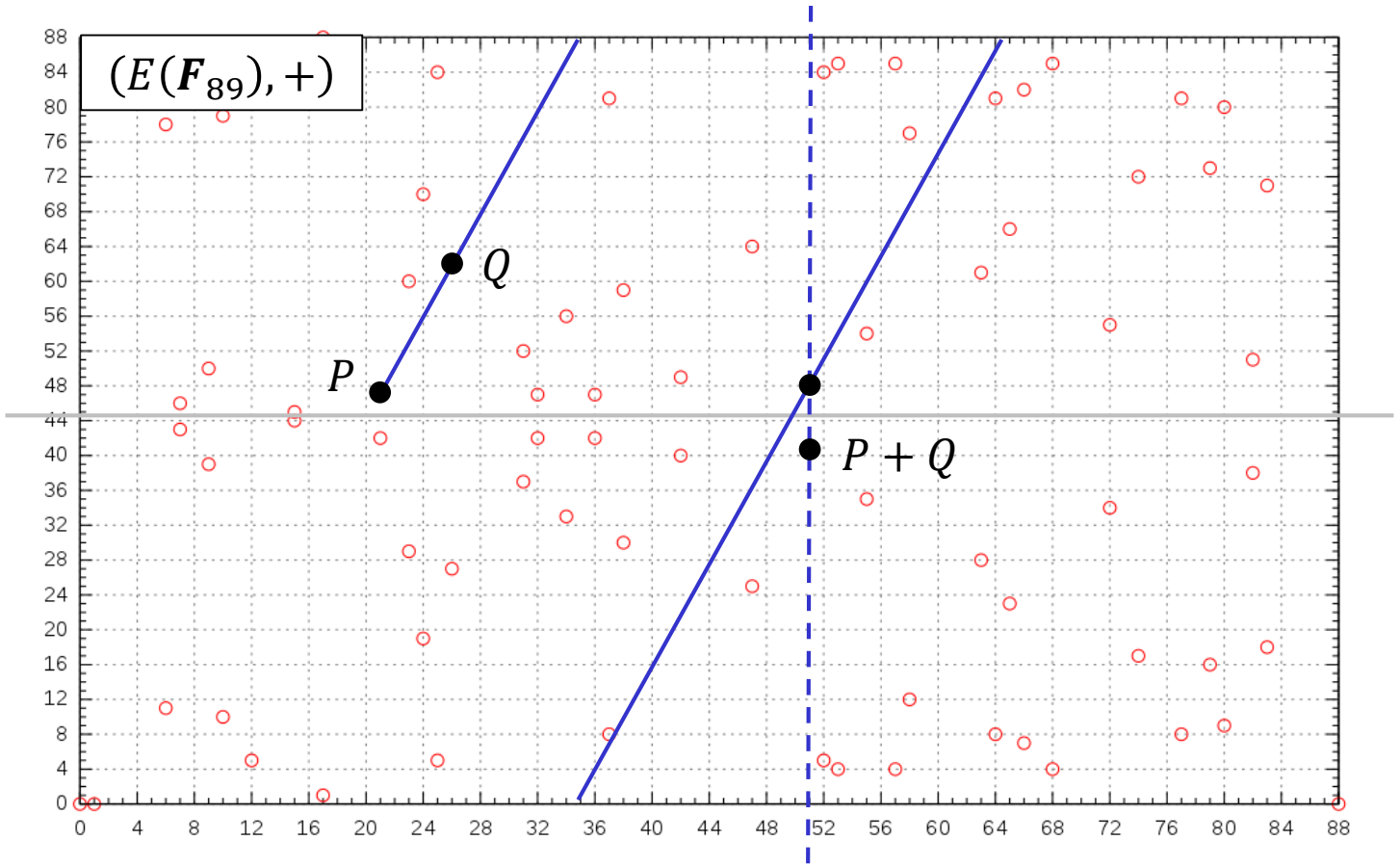
$$a, b, x, y \in \boldsymbol{Z}_{89}$$

$$P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$\begin{cases} x_3 = \dfrac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b} \\ y_3 = \dfrac{x_1x_2(x_1 + x_2) - x_3\big((x_1 + x_2)^2 - x_1x_2 + a\big) - y_1y_2 - b}{y_1 + y_2} \end{cases}$$

Elliptic curves

Theorem: the points on an elliptic curve, together with $\mathcal{O}$, is an abelian group under "geometric point addition"



Notation: $(E(\boldsymbol{F}_p),+)$ = an elliptic curve group defined over $\boldsymbol{F}_p$

$$y^2 = x^3 + ax + b$$

$$a, b, x, y \in \boldsymbol{Z}_{89}$$

$$P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

Still valid!

$$\begin{cases} x_3 = \dfrac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b} \\ y_3 = \dfrac{x_1x_2(x_1 + x_2) - x_3\left((x_1 + x_2)^2 - x_1x_2 + a\right) - y_1y_2 - b}{y_1 + y_2} \end{cases}$$

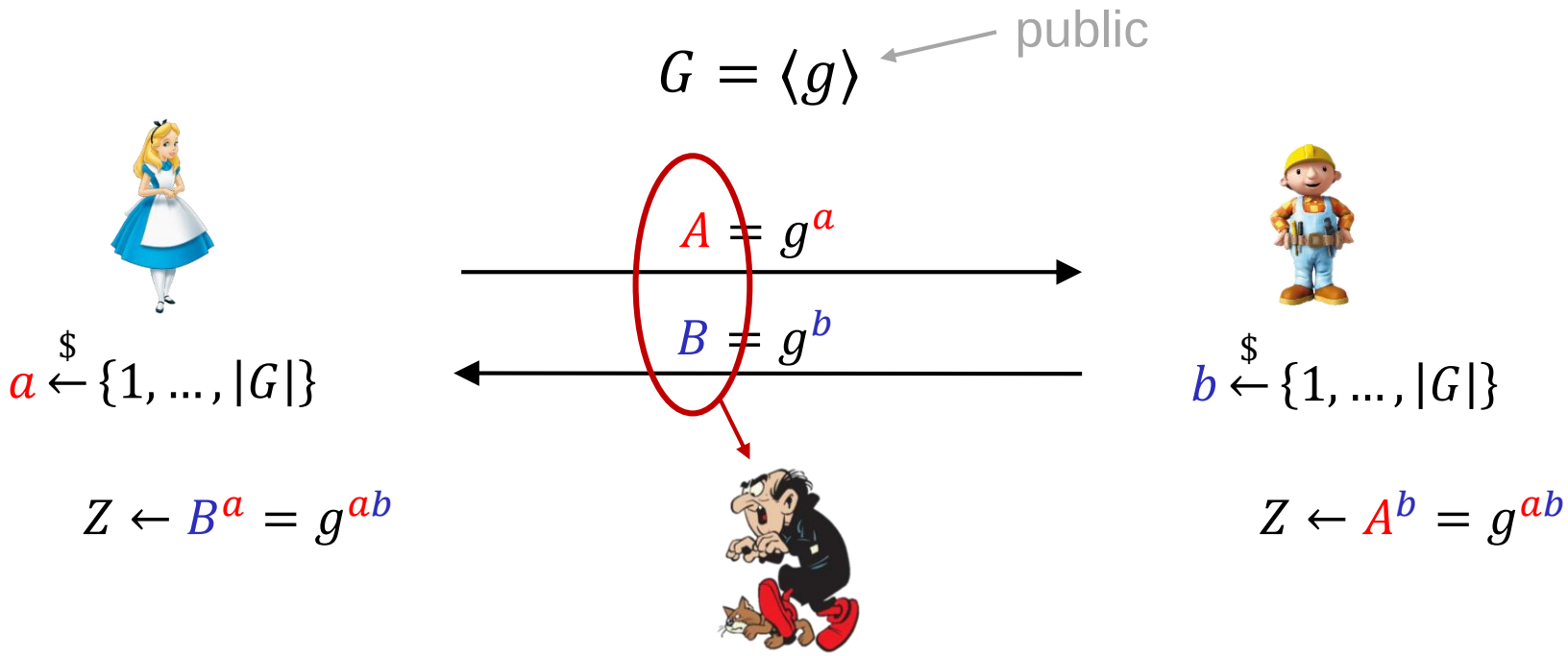
Both addition *and* multiplication needed mod p

$(\boldsymbol{Z}_p,+)$ *and* $(\boldsymbol{Z}_p^*,\cdot)$

$(\boldsymbol{F}_p,+,\cdot)$

Finite field

Diffie-Hellman – security



Security (given G, g, A, B):

- Must be hard to compute $Z \leftarrow g^{ab}$ (DH assumption)
- Must be hard to find a (or b) (DLOG assumption)

- $G = (\boldsymbol{Z}_p, +)$ not secure
- $G = (\boldsymbol{Z}_p^*, \cdot)$ secure for $|\boldsymbol{Z}_p^*|$ **large enough** (conjectured)
- $G = (E(\boldsymbol{F}_p), +)$ secure for $|E(\boldsymbol{F}_p)|$ **large enough** (conjectured)

Diffie-Hellman in $\left(Z_p^*, \cdot\right)$ – what is large enough?

$p =$ 171254583176141379301960419792575778264088323240375085733932929816426671397476217788024387752387285929683446135893799323484756135034769321631669738132186983438164632891441853629126025225404949830905314972329658295365245072698488256583114202993359222957097432675083225259667739503949192575768420387716327420441424710535098501236058838158571626669177751934961573726561955583057270098912760065140004093658772181713883199238963093777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830

$\approx 2^{2048}$

$A = 2$ $\mod p$

$\xleftarrow{\$} \{1 \dots p\}$

$B = 2$ $\mod p$

$\xleftarrow{\$} \{1 \dots p\}$

$Z \leftarrow 2$ $\times$ $\mod p$

Diffie-Hellman in $\left(E\left(Z_p\right), +\right)$ – what is large enough?

$p =$ 115792089210356248762697446949407573530086143415290314195533631308867097853951 $\approx 2^{256}$

$A =$ 89080287395774076886404465276262680764297347818898641719959470631357066012729 $\cdot Q$

89080287395774076886404465276262680764297347818898641719959470631357066012729 $\xleftarrow{\$} \{1 \dots p\}$

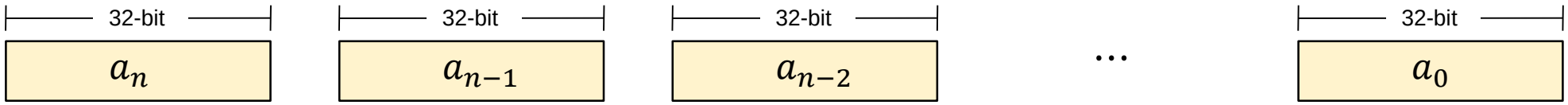
47777709944406917119610162988578419285974852857762532427086507801748071058272 $\xleftarrow{\$} \{1 \dots p\}$

$B =$ 47777709944406917119610162988578419285974852857762532427086507801748071058272 $\cdot Q$

$Z \leftarrow$ 89080287395774076886404465276262680764297347818898641719959470631357066012729 $\times$ 47777709944406917119610162988578419285974852857762532427086507801748071058272 $\cdot Q$

Big-number arithmetic

a = 2048-bit integer, 64-bit CPU



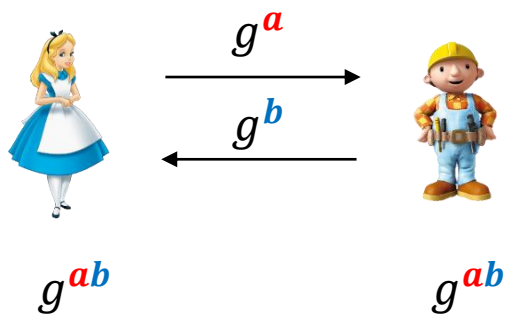
Algorithm	Input	Output	Time
ADD	a, b	$a + b$	$\mathcal{O}(n)$
MULT	a, b	ab	$\mathcal{O}(n^2)$
INT-DIV	a, b	(q, r) s.t. $a = qb + r$ and $0 \leq r < b$	$\mathcal{O}(n^2)$
MOD	a, p	$a \bmod p$	$\mathcal{O}(n^2)$
MOD-INV	a	$a^{-1} \bmod p$	$\mathcal{O}(n^2)$

$\mathcal{O}(n \log n)$*

* $n > 2^{7137398073256634897664758526207831206 41}$

Diffie-Hellman – computations

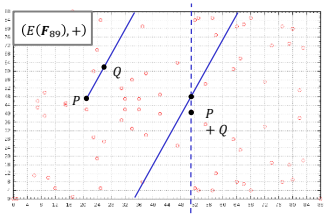
- $(\boldsymbol{Z}_p^*, \cdot)$
  - Find prime p (one-time)
  - Find generator $\langle g \rangle = (\boldsymbol{Z}_p^*, \cdot)$ (one-time)
  - Compute $g^a = gg \cdots g \mod p$
    - Multiply $x \cdot y \mod p$



- $(E(\boldsymbol{F}_p), +)$

Group exponentiation g^a

  - Find prime p (one-time)
  - Generate curve $y^2 = x^3 + ax + b \mod p$ (one-time)
  - Find curve group order $|E(\boldsymbol{F}_p)|$ (one-time)
  - Find generator Q (one-time)
  - Compute $aQ = Q + Q + \cdots + Q$
    - Point addition



Computing in groups – exponentiation

Goal: compute $g^n \in G$

Naïve approach: $g^n = \overbrace{g \circ g \circ g \circ \cdots \circ g}^{n}$

- Running time: $n \approx 2^k \implies 2^k$ group operations!
- Doesn't work for exponent sizes used in cryptography

Computing in groups – exponentiation; square-and-multiply

Goal: compute g^n

$$n = (b_5 b_4 b_3 b_2 b_1 b_0)_2$$
$$= b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$$y_6 \leftarrow e$$
$$y_5 \leftarrow y_6^2 \circ g^{b_5} \quad = g^{b_5}$$
$$y_4 \leftarrow y_5^2 \circ g^{b_4} \quad = g^{2b_5 + b_4}$$
$$y_3 \leftarrow y_4^2 \circ g^{b_3} \quad = g^{2^2 b_5 + 2b_4 + b_3}$$
$$y_2 \leftarrow y_3^2 \circ g^{b_2} \quad = g^{2^3 b_5 + 2^2 b_4 + 2b_3 + b_2}$$
$$y_1 \leftarrow y_2^2 \circ g^{b_1} \quad = g^{2^4 b_5 + 2^3 b_4 + 2^2 b_3 + 2b_2 + b_1}$$
$$y_0 \leftarrow y_1^2 \circ g^{b_0} \quad = \underline{g^{2^5 b_5 + 2^4 b_4 + 2^3 b_3 + 2^2 b_2 + 2b_1 + b_0}}$$

$$g^{43}$$

$$43 = 101011_2$$

$$y_6 \leftarrow e$$
$$y_5 \leftarrow y_6^2 \circ g^1 \quad = g^1$$
$$y_4 \leftarrow y_5^2 \circ g^0 \quad = g^2$$
$$y_3 \leftarrow y_4^2 \circ g^1 \quad = g^{4+1} = g^5$$
$$y_2 \leftarrow y_3^2 \circ g^0 \quad = g^{10}$$
$$y_1 \leftarrow y_2^2 \circ g^1 \quad = g^{20+1} = g^{21}$$
$$y_0 \leftarrow y_1^2 \circ g^1 \quad = g^{42+1} = \underline{g^{43}}$$

Computing in groups – exponentiation; square-and-multiply

Goal: compute g^n

g^{43}

$$n = (b_5 b_4 b_3 b_2 b_1 b_0)_2$$
$$= b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

$43 = 101011_2$

$y_6 \leftarrow e$

$y_5 \leftarrow y_6^2 \circ g^{b_5} \quad = g^{b_5}$

$y_4 \leftarrow y_5^2 \circ g^{b_4} \quad = g^{2b_5 + b_4}$

$y_3 \leftarrow y_4^2 \circ g^{b_3} \quad = g^{2^2 b_5 + 2b_4 + b_3}$

$y_2 \leftarrow y_3^2 \circ g^{b_2} \quad = g^{2^3 b_5 + 2^2 b_4 + 2b_3 + b_2}$

$y_1 \leftarrow y_2^2 \circ g^{b_1} \quad = g^{2^4 b_5 + 2^3 b_4 + 2^2 b_3 + 2b_2 + b_1}$

$y_0 \leftarrow y_1^2 \circ g^{b_0} \quad = g^{2^5 b_5 + 2^4 b_4 + 2^3 b_3 + 2^2 b_2 + 2b_1 + b_0}$

Square–and–Multiply($g \in G, n \in \boldsymbol{Z}$)

1. $n = (b_k \ldots b_1 b_0)_2$
2. $y \leftarrow 1$
3. **for** $i = k$ **downto** 0 **do**
4. $\quad y \leftarrow y^2 \circ g^{b_i}$
5.
6. **return** y

Square-and-multiply

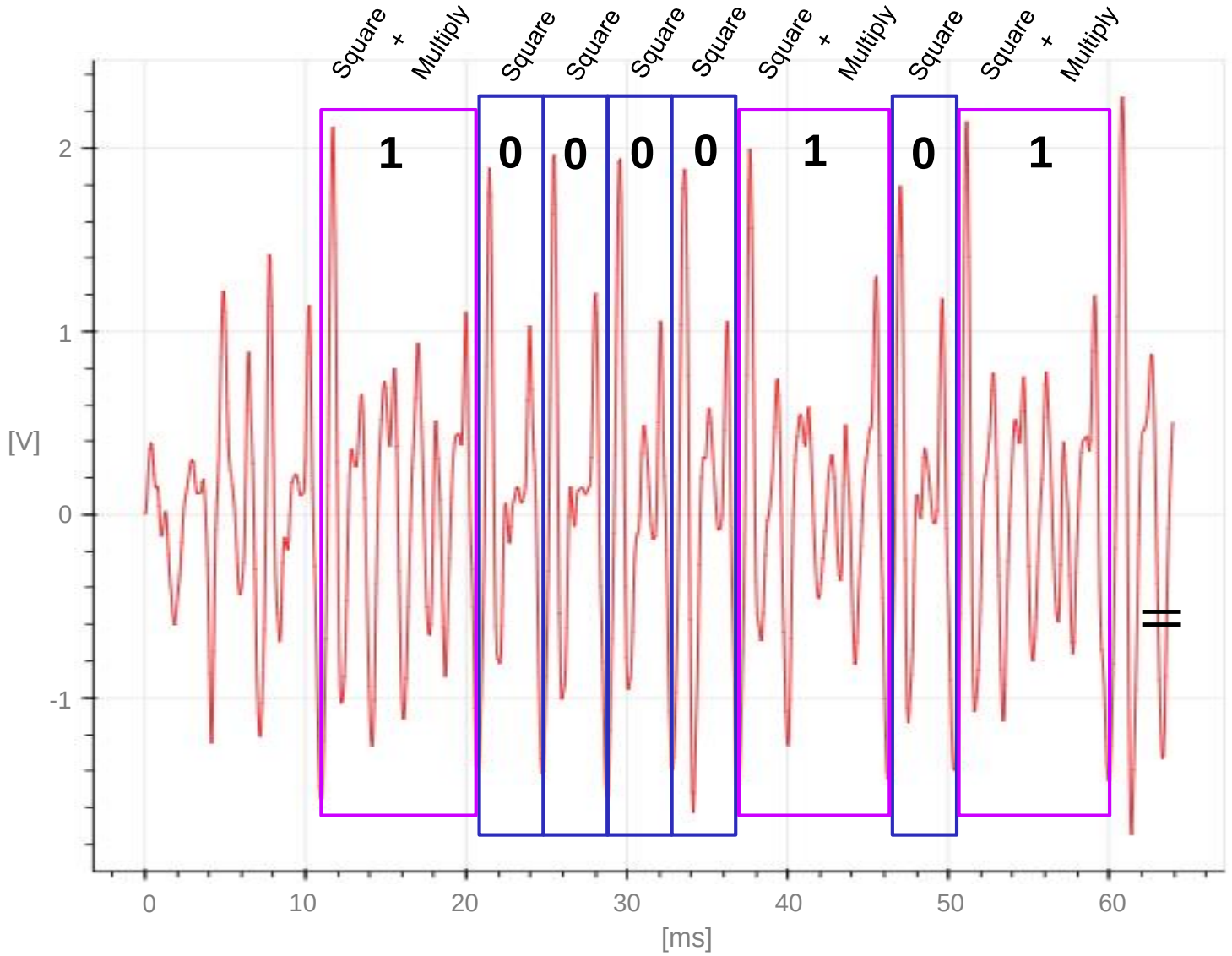
- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Naïve exponentiation: $\mathcal{O}(n) \approx p$
  - Square-and-multiply: $\mathcal{O}(t) = \mathcal{O}(\log_2 n) \approx \log_2 p$

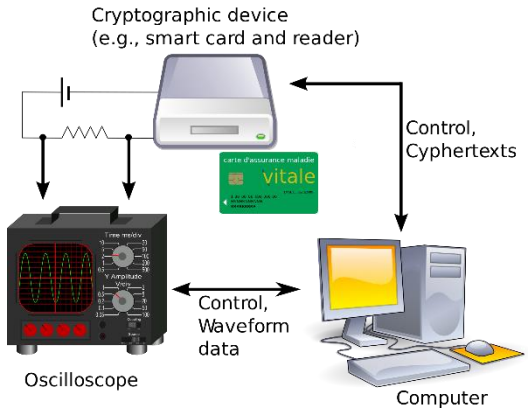
2^{2048} vs. 2048 mults. for $p \approx 2^{2048}$

- $\left(E\left(\boldsymbol{F}_p\right), +\right)$
  - Naïve "exponentiation" (i.e., point multiplication nQ): $\mathcal{O}(n) \approx \left|E\left(\boldsymbol{F}_p\right)\right|$
  - "Square-and-multiply" (i.e., double-and-add): $\mathcal{O}(t) = \mathcal{O}(\log_2 n) \approx \log_2 \left|E\left(\boldsymbol{F}_p\right)\right|$

2^{256} vs. 256 EC adds. for NIST P-256 or Curve25519

Side-channel attacks – power analysis





Square–and–Multiply$(g \in G\,, n \in \boldsymbol{Z})$

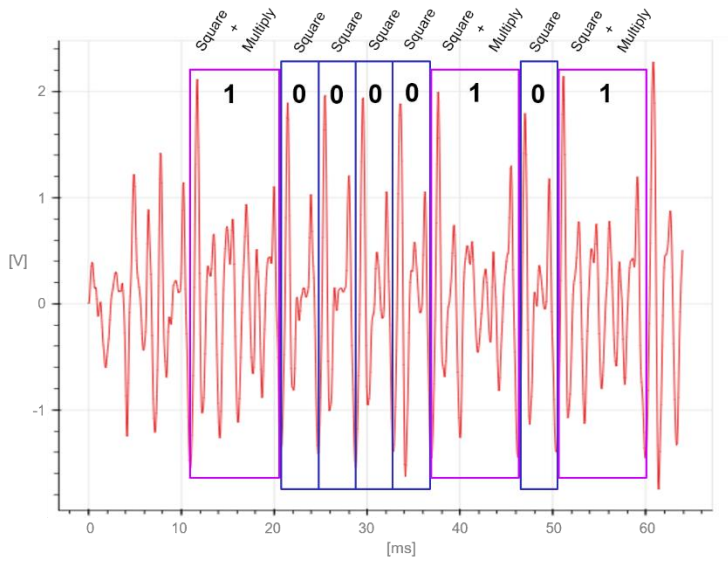
1. $n = (b_k \ldots b_1 b_0)_2$
2. $y \leftarrow e$
3. **for** $i = k$ **downto** 0 **do**
4. $\quad y \leftarrow y^2$
5. $\quad$ **if** $b_i = 1$ **then**
6. $\quad\quad y \leftarrow y \circ g$
7. **return** y

Mitigations

- Power (and time) difference between $b_i = 0$ and $b_i = 1$
- Solution: Square-and-*Always*-Multiply

Square–and–always–Multiply$(g \in G, n \in \boldsymbol{Z})$

1. $n = (b_k \ldots b_1 b_0)_2$
2. $y \leftarrow e$
3. **for** $i = k$ **downto** 0 **do**
4. $\quad y \leftarrow y^2$
5. $\quad z \leftarrow y \circ g$ // always perform multiplication
6. $\quad$ **if** $b_i = 1$ **then**
7. $\quad\quad y \leftarrow z$
8. **return** y



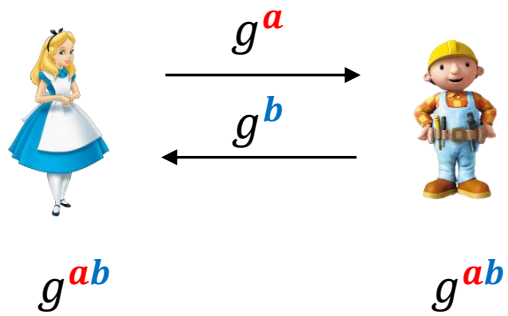
Square–and–Multiply$(g \in G, n \in \boldsymbol{Z})$

1. $n = (b_k \ldots b_1 b_0)_2$
2. $y \leftarrow 1$
3. **for** $i = k$ **downto** 0 **do**
4. $\quad y \leftarrow y^2$
5. $\quad$ **if** $b_i = 1$ **then**
6. $\quad\quad y \leftarrow y \circ g$
7. **return** y

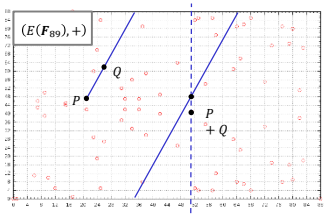
- Wastes computations: $\approx t/2$ of multiplications thrown away
- Better solutions exist

Diffie-Hellman – computations

- $(\boldsymbol{Z}_p^*, \cdot)$
  - Find prime p (one-time)
  - Find generator $\langle g \rangle = (\boldsymbol{Z}_p^*, \cdot)$ (one-time)
  - Compute $g^a = gg \cdots g \mod p$
    - Multiply $x \cdot y \mod p$



- $(E(\boldsymbol{F}_p), +)$
  - Find prime p (one-time)
  - Generate curve $y^2 = x^3 + ax + b \mod p$ (one-time)
  - Find curve group order $|E(\boldsymbol{F}_p)|$ (one-time)
  - Find generator P (one-time)
  - Compute $aP = P + P + \cdots + P$
    - Point addition



Finding prime numbers

- We need *large* prime numbers
  - Both for $(\boldsymbol{Z}_p^*, \cdot)$ and $(E(\boldsymbol{F}_p), +)$
- No efficient "prime-generating" formula is known
- Simple idea: pick random number and check if it's prime
- Efficient? $\Longrightarrow$ What is the success probability?
  - Depends on the ratio primes / non-primes
  - $\pi(n) =$ "#prime numbers $\leq n$"
  - **Prime Number Theorem:** $\frac{\pi(n)}{n} \approx \frac{1}{\ln n}$
  - 2048 bit number: ≈ 700 trials needed

GenPrime(k)

1. **while** true **do**
2. $\quad n \xleftarrow{\$}$ odd k-bit integer
3. $\quad$ **if** IsPrime$(n) =$ PRIME **then**
4. $\quad\quad$ **return** n

$$n \approx 2^{2048} \Longrightarrow \frac{\pi(n)}{n} \approx \frac{1}{\ln 2^{2048}} \approx \frac{1}{1400}$$

n	100	1000	10^6	10^9	10^{12}	10^{15}
$\pi(n)$	25	168	$\approx 78 \cdot 10^3$	$\approx 50 \cdot 10^6$	$\approx 30 \cdot 10^9$	$\approx 29 \cdot 10^{12}$

Finding prime numbers

- We need *large* prime numbers
  - Both for $(\boldsymbol{Z}_p^*, \cdot)$ and $(E(\boldsymbol{F}_p), +)$

- Naïve $\text{IsPrime}(n)$:
  - 3 divides n? Not prime
  - 5 divides n? Not prime
  - 7 divides n? Not prime
  - ⋮
  - $\lfloor \sqrt{n} \rfloor$ divides n? Not prime
  - n is prime!

  - *Very* inefficient: $n \approx 2^k \implies \pi(n) \approx \frac{2^k}{\ln 2^k} \approx \frac{2^k}{k}$

- Want: a simple property which *only* holds for prime numbers

GenPrime(k)

```
1. while true do
2.     n ←$ odd k-bit integer
3.     if IsPrime(n) = PRIME then
4.         return n
```

Fermat's theorem

Lagrange's theorem: if $(G,\circ)$ is a finite group, then for all $g \in G$:

$$g^{|G|} = e$$

$\left(\boldsymbol{Z}_p^*,\cdot\right) = \{1,2,\ldots,p-1\}$ under multiplication modulo p $\qquad$ $\left|\boldsymbol{Z}_p^*\right| = p-1$

Fermat's theorem: if p is prime, then for all $a \not\equiv 0 \pmod p$:

$$a^{p-1} \equiv 1 \pmod p$$

$A \not\Leftarrow B$

p is prime $\Rightarrow$ *all* $a \not\equiv 0 \bmod p$ satisfy Fermat's theorem

p is **not** prime $\Leftarrow$ *some* $a \not\equiv 0 \bmod p$ does **not** satisfy Fermat's theorem

$\bar{A}$ $\qquad$ $\bar{B}$

$$2^{560} = 1 \bmod 561$$
$$3^{560} = 1 \bmod 561$$
$$4^{560} = 1 \bmod 561$$
$$\vdots$$
$$560^{560} = 1 \bmod 561$$

Carmichael number

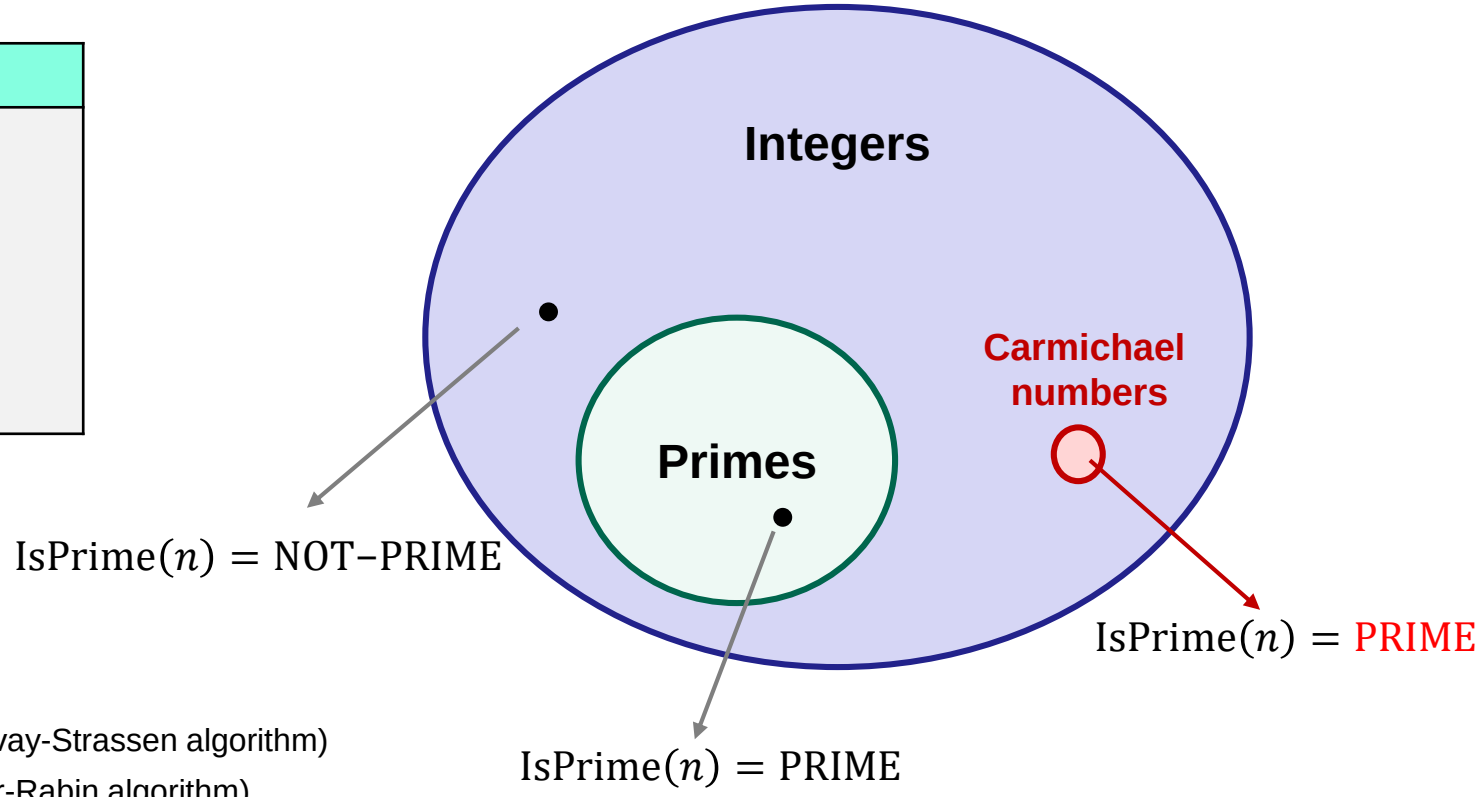
Fermat primality test

Fermat's theorem: if p is prime, then for all $a \neq 0 \pmod p$:

$$a^{p-1} \equiv 1 \pmod p$$

IsPrime(n)

1. **for** $a \in \{2, 3, \ldots, t\}$ **do**
2. **if** $a^{n-1} \neq 1 \mod n$ **then**
3. **return** NOT-PRIME
4. **return** PRIME



Better tests exist:
- Euler-test (Solovay-Strassen algorithm)
- Strong pseudoprime-test (Miller-Rabin algorithm)

Finding primes

IsPrime(n) // Fermat

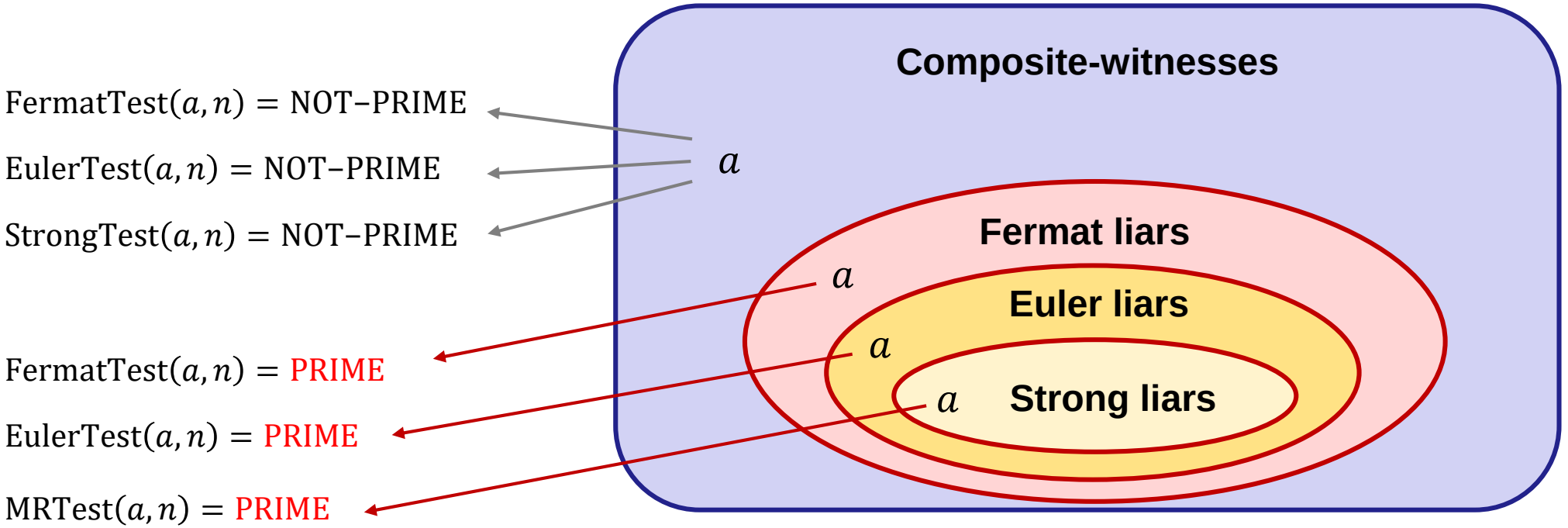
1. **for** $a \in \{2, 3, \dots, t\}$ **do**
2. **if** $\neg$FermatTest(a, n) **then**
3. **return** NOT-PRIME
4. **return** PRIME

IsPrime(n) // Solovay-Strassen

1. **for** $a \in \{2, 3, \dots, t\}$ **do**
2. **if** $\neg$EulerTest(a, n) **then**
3. **return** NOT-PRIME
4. **return** PRIME

IsPrime(n) // Miller-Rabin

1. **for** $a \in \{2, 3, \dots, t\}$ **do**
2. **if** $\neg$StrongTest(a, n) **then**
3. **return** NOT-PRIME
4. **return** PRIME



Finding primes in practice

- Miller-Rabin most used in practice
- Failure probability $\leq \frac{1}{4^t}$
- *Deterministic* primality tests
  - Many exist…but none running in polynomial time (P)
  - Open problem for a long time
  - *Agrawal, Kayal & Saxena '02*: "PRIMES is in P"
  - Original running time $\approx \mathcal{O}(k^{12})$
  - Quickly improved to $\approx \mathcal{O}(k^6)$

IsPrime(n) **// Miller-Rabin**

1. **for** $a \in \{2, 3, \ldots, t\}$ **do**
2. **if** $\neg$StrongTest(a, n) **then**
3. **return** NOT-PRIME
4. **return** PRIME

Finding generators

- Easy in prime-order groups
  - All non-identity elements are generators!

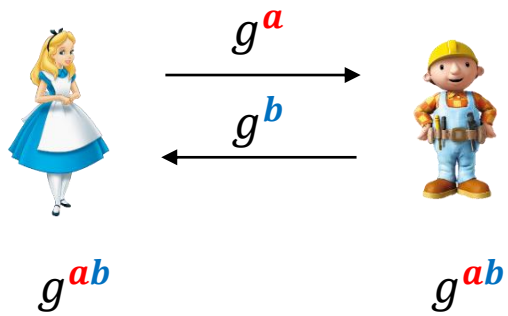
Lagrange's theorem:

if $H < G$ then $|H|$ divides $|G|$

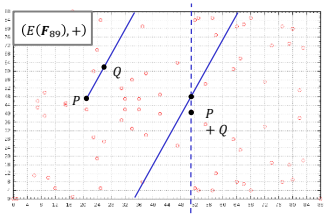
- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Not prime-order!
  - Common in practice: pick element at random; check if generator

Diffie-Hellman – computations

- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Find prime p (one-time)
  - Find generator $\langle g \rangle = \left(\boldsymbol{Z}_p^*, \cdot\right)$ (one-time)
  - Compute $g^a = gg \cdots g \mod p$
    - Multiply $x \cdot y \mod p$



- $\left(E\left(\boldsymbol{F}_p\right), +\right)$
  - Find prime p (one-time)
  - Generate curve $y^2 = x^3 + ax + b \mod p$ (one-time)
  - Find curve group order $\left|E\left(\boldsymbol{F}_p\right)\right|$ (one-time)
  - Find generator P (one-time)
  - Compute $aP = P + P + \cdots + P$
    - Point addition



Tricky! (Schoof's algorithm) $\left|E\left(\boldsymbol{F}_p\right)\right| \neq p$ (typically)

Easy! ($\left|E\left(\boldsymbol{F}_p\right)\right|$ usually prime)

Diffie-Hellman – man-in-the-middle attack

$$G = \langle g \rangle$$

$$A = g^a$$

$$B = g^b$$

$$a \xleftarrow{\$} \{1, \dots, |G|\}$$

$$b \xleftarrow{\$} \{1, \dots, |G|\}$$

$$Z \leftarrow B^a = g^{ab}$$

$$Z \leftarrow A^b = g^{ab}$$

Diffie-Hellman – man-in-the-middle attack

$G = \langle g \rangle$

$A = g^a$

$X = g^x$

$B = g^b$

$a \xleftarrow{\$} \{1, \dots, |G|\}$

$b \xleftarrow{\$} \{1, \dots, |G|\}$

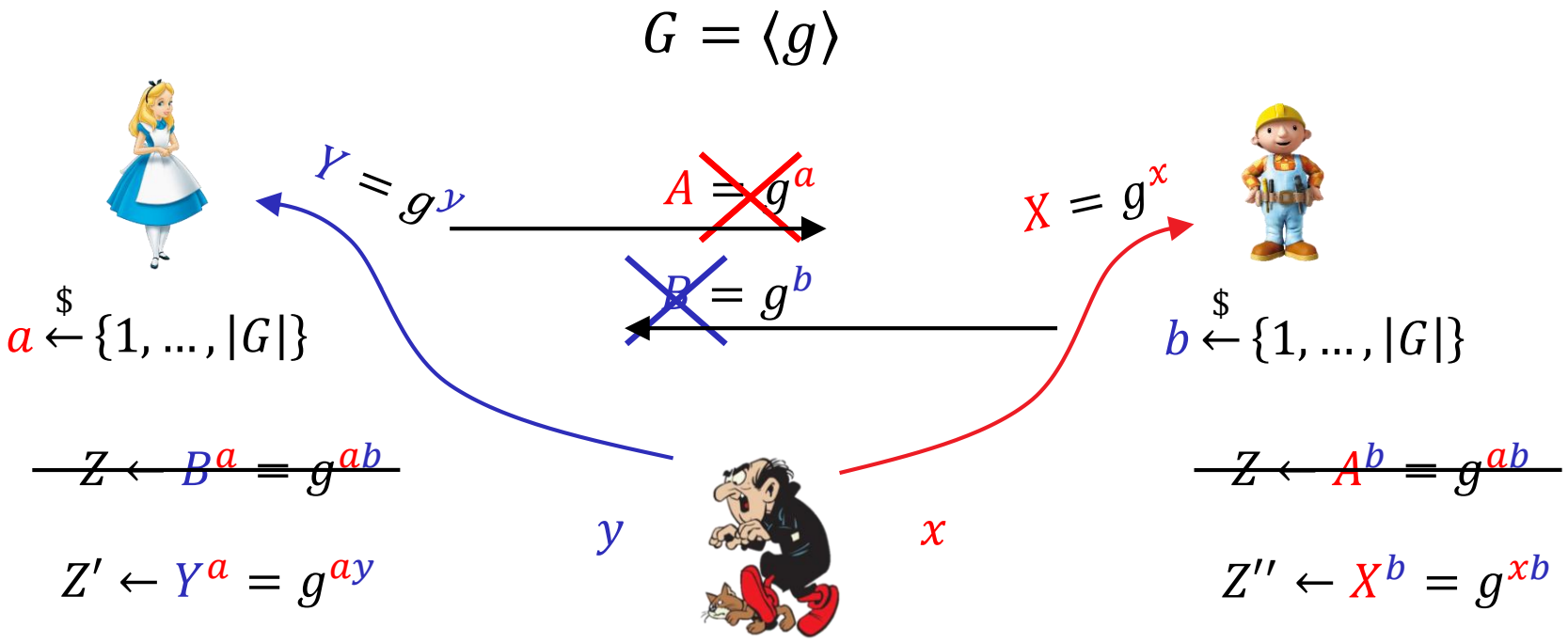
$Z \leftarrow B^a = g^{ab}$

x

~~$Z \leftarrow A^b = g^{ab}$~~

$Z'' \leftarrow X^b = g^{xb}$

Diffie-Hellman – man-in-the-middle attack



Noise-protocol

Long-term key → $A = g^a$

$G = \langle g \rangle, A, B$

$B = g^b$ ← Long-term key

$X = g^x$ →

← $Y = g^y$

$x \xleftarrow{\$} \{1, \dots, |G|\}$

$y \xleftarrow{\$} \{1, \dots, |G|\}$

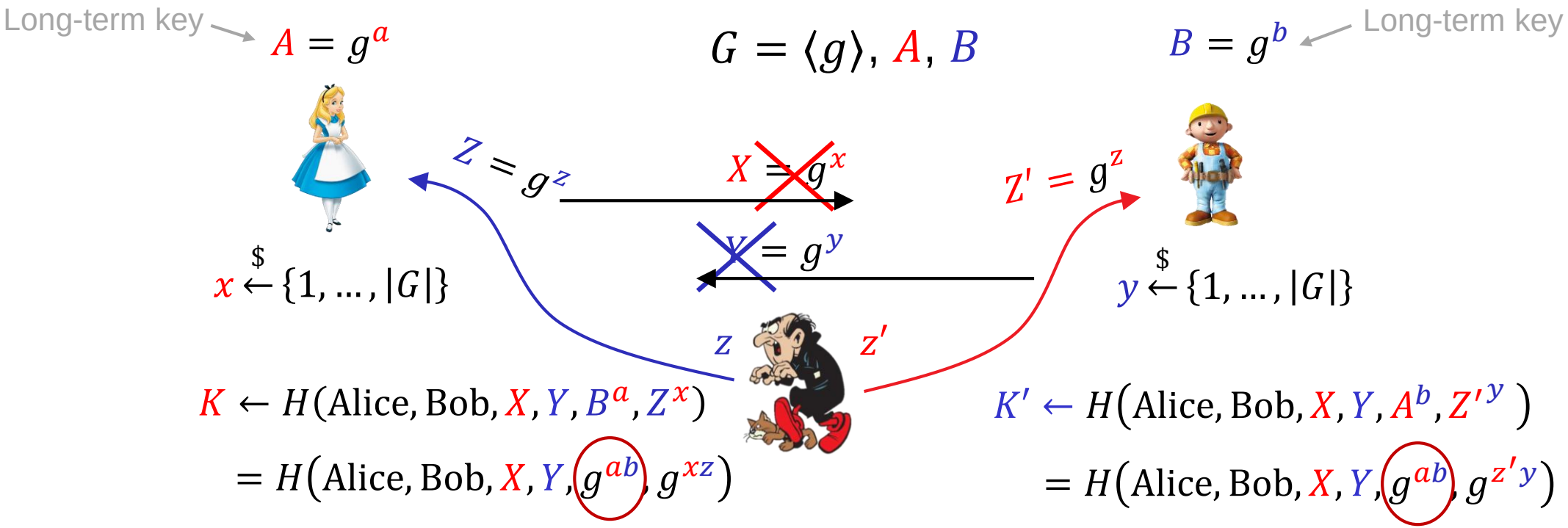
$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, B^a, Y^x)$

$= H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{xy})$

$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, A^b, X^y)$

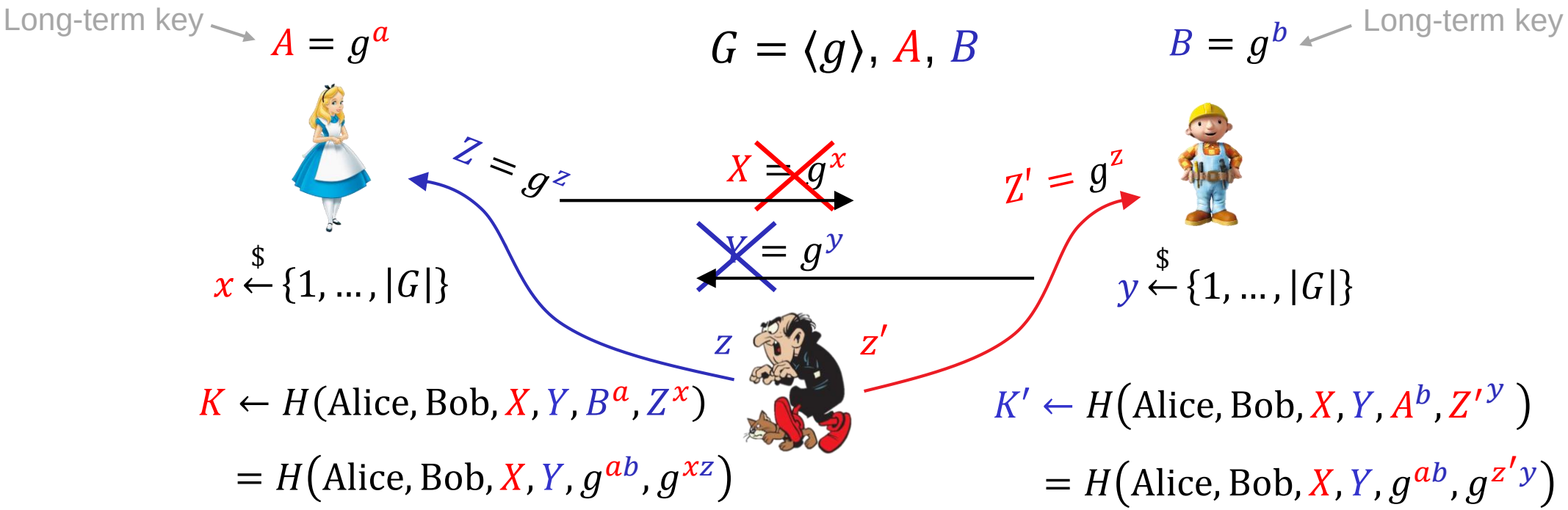
$= H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{xy})$

Noise-protocol



Can't compute K or K'!

Noise-protocol



Many alternatives:

$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ay}, g^{xb})$	static-ephemeral, ephemeral-static
$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ay}, g^{xb}, g^{xy})$	static-ephemeral, ephemeral-static, ephemeral-ephemeral
$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{ay}, g^{xb}, g^{xy})$	static-static, ephemeral-static, ephemeral-static, ephemeral-ephemeral

Noise-protocol

Long-term key → $A = g^a$

$G = \langle g \rangle, A, B$

$B = g^b$ ← Long-term key

$X = g^x$ →

← $Y = g^y$

$x \xleftarrow{\$} \{1, \dots, |G|\}$

$y \xleftarrow{\$} \{1, \dots, |G|\}$

$$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, B^a, Y^x)$$
$$= H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{xy})$$

$$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, A^b, X^y)$$
$$= H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{xy})$$

Many alternatives:

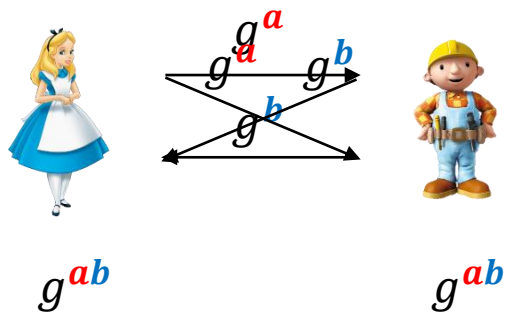
$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ay}, g^{xb})$ static-ephemeral, ephemeral-static

$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ay}, g^{xb}, g^{xy})$ static-ephemeral, ephemeral-static, ephemeral-ephemeral

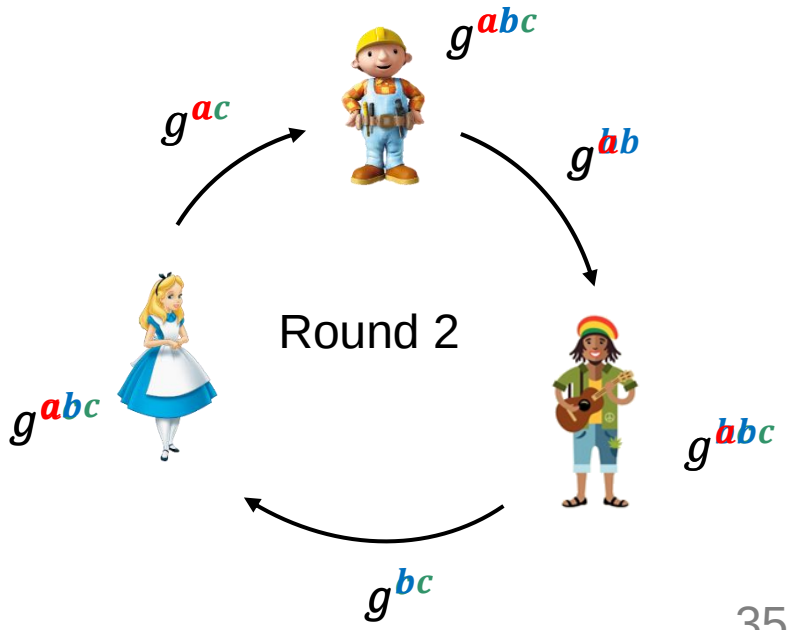
$K \leftarrow H(\text{Alice}, \text{Bob}, X, Y, g^{ab}, g^{ay}, g^{xb}, g^{xy})$ static-static, ephemeral-static, ephemeral-static, ephemeral-ephemeral

N-party Diffie-Hellman

- N-party Diffie-Hellman possible in $N - 1$ rounds



- 1-round N-party Diffie-Hellman:
  - $N = 2$ – normal Diffie-Hellman
  - $N = 3$ – Diffie-Hellman with *bilinear pairings* (**Joux '00**)
  - $N \geq 4$ – open problem
    - Possible with *multilinear maps* (very advanced)
    - …but we don't know any secure multilinear maps ☹



Summary

- Special algorithms needed to deal with the large numbers in asymmetric cryptography
  - Square-and-multiply for group exponentiation
    - Not secure against side-channel attacks
  - Prime finding
    - Common in practice: pick random number; check if prime
    - Primality tests: Fermat's, Miller-Rabin (small one-sided error)

- Plain Diffie-Hellman is *not* secure against active adversaries
  - Man-in-the-middle attack
  - Problem: lack of authentication
  - Common solution: digital signatures (used in TLS, IPsec, SSH)
  - Modern solution: long-term Diffie-Hellman keys mixed with ephemeral Diffie-Hellman keys
    - Noise protocol, Signal, What'sApp, Facebook Messenger Secret Conversations