Lecture 10 – Diffie-Hellman key exchange III, computational aspects, Noise

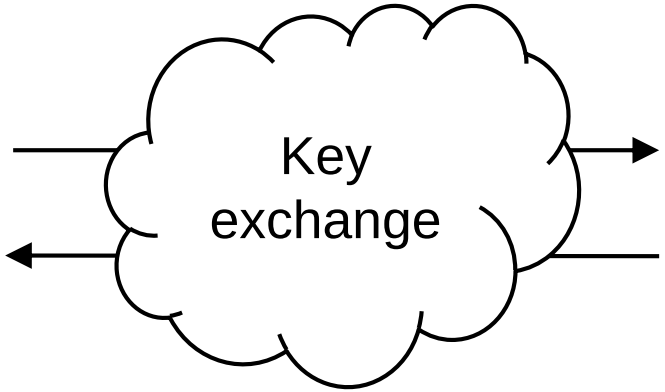
TEK4500

26.10.2022

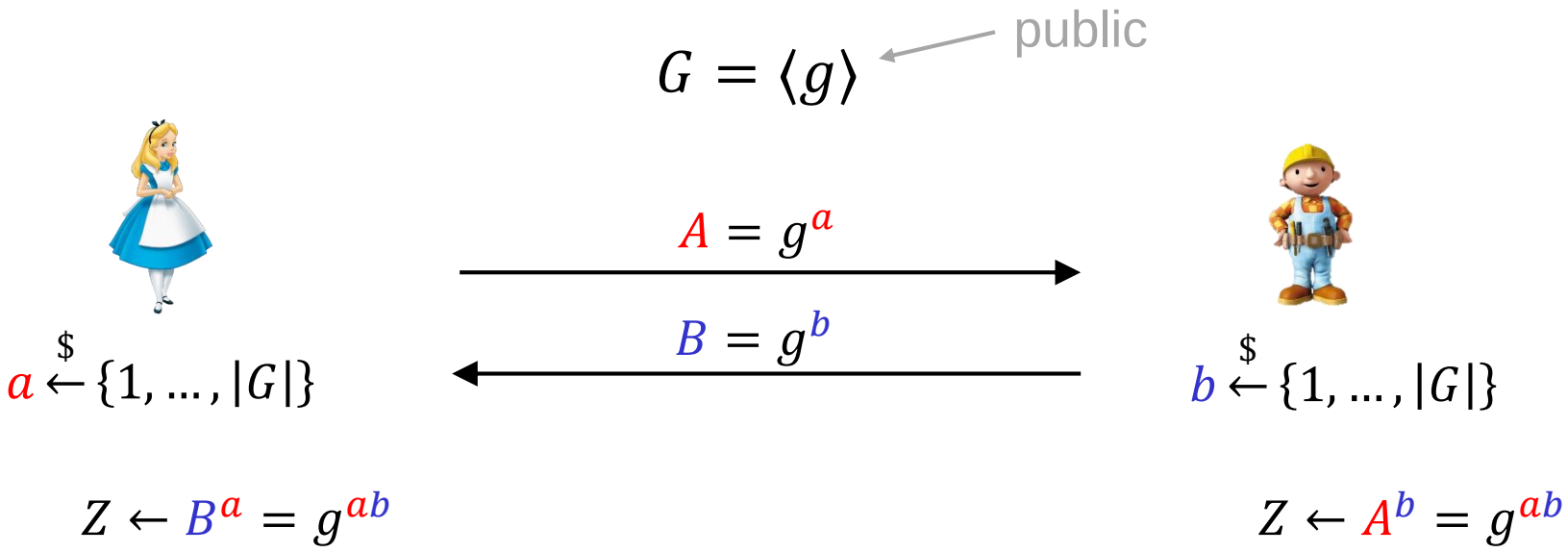
Håkon Jacobsen

hakon.jacobsen@its.uio.no

Diffie-Hellman



Diffie-Hellman – security



Examples

- $G = (\boldsymbol{Z}_p, +)$ 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 $\left(Z_p^*, \cdot\right)$

$p =$ 171254583176141379301960419792575778264088323240375085733932929816426671397476217788024387752387285929683446135893799323484756135034769321631669738132186983438164632891441853629126025225404949830905314972329658295365245072698488256583114202993359222957097432675083225259667739503949192575768420387716327420441424710535098501236058838158571626669177751934961573726561955583057270098912760065140004093658772181713883199238963093777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830 $\approx 2^{2048}$

$A = 2$ 32317006071311007300338913926423828248817941241140239112842009751400741706634354222619689417363569347117901737909704191754605873209195028853758986185622153212175412514901774520270235796078236248884246189477587641105928646099411723245426622522193230540919037680524235519125679715870117001058055877651038861847280257976054903569737258325616760813393617995413364765591603683178967290731783845896806396719009772021941686472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905591 $\mod p$

323170060713110073003389139264238282488179412411402391
128420097514007417066343542226196894173635693471179017
379097041917546058732091950288537589861856221532121754
125149017745202702357960782362488842461894775876411059
286460994117232454266225221932305409190376805242355191
256797158701170010580558776510388618472802579760549035
697325615261670813393617995413364765591603683178967290
731783845896806396719009772021941686472258710314113364
293195361934716365332097170774482279885885653692086452
966360772502689555059283627511211740969729980684105543
595848665832916421362182310789909994486524682624169720
359118525070453610905591 $\leftarrow \{1 \dots p\}$

665680077810100734070476416898417408020670352441378967
997224101900193957211854827569676933273935982997835638
067380762809024311842715496660113976613131478051784487
322446809608196031803691263486170912693625523249742383
264476433978409434004770395167011614217279552198029936
108023398643999746539201834933168220864789880837484536
563067997761270826642223539437541791058416731683176995
365899867720283590420932417377927106884773199080517477
615608675490982255098980545292032908358093913531433702
049429047413071525556307299541445514389721222256380833
743627991333762777307118317040885305789910094293548481
550130956194277068752 $\leftarrow \{1 \dots p\}$

665680077810100734070476416898417408020670352441378967
997224101900193957211854827569676933273935982997835638
067380762809024311842715496660113976613131478051784487
322446809608196031803691263486170912693625523249742383
264476433978409434004770395167011614217279552198029936
108023398643999746539201834933168220864789880837484536
563067997761270826642223539437541791058416731683176995
365899867720283590420932417377927106884773199080517477
615608675490982255098980545292032908358093913531433702
049429047413071525556307299541445514389721222256380833
743627991333762777307118317040885305789910094293548481
550130956194277068752 $B = 2$ $\mod p$

323170060713110073003389139264238282488179412411402391
128420097514007417066343542226196894173635693471179017
379097041917546058732091950288537589861856221532121754
125149017745202702357960782362488842461894775876411059
286460994117232454266225221932305409190376805242355191
256797158701170010580558776510388618472802579760549035
697325615261670813393617995413364765591603683178967290
731783845896806396719009772021941686472258710314113364
293195361934716365332097170774482279885885653692086452
966360772502689555059283627511211740969729980684105547
595848665832916421362182310789909994486524682624169720
359118525070453610905591 $Z \leftarrow 2$

$\times$

665680077810100734070476416898417408020670352441378967
997224101900193957211854827569676933273935982997835638
067380762809024311842715496660113976613131478051784487
322446809608196031803691263486170912693625523249742383
264476433978409434004770395167011614217279552198029936
108023398643999746539201834933168220864789880837484536
563067997761270826642223539437541791058416731683176995
365899867720283590420932417377927106884773199080517477
615608675490982255098980545292032908358093913531433702
049429047413071525556307299541445514389721222256380833
743627991333762777307118317040885305789910094293548481
550130956194277068752 $\mod p$

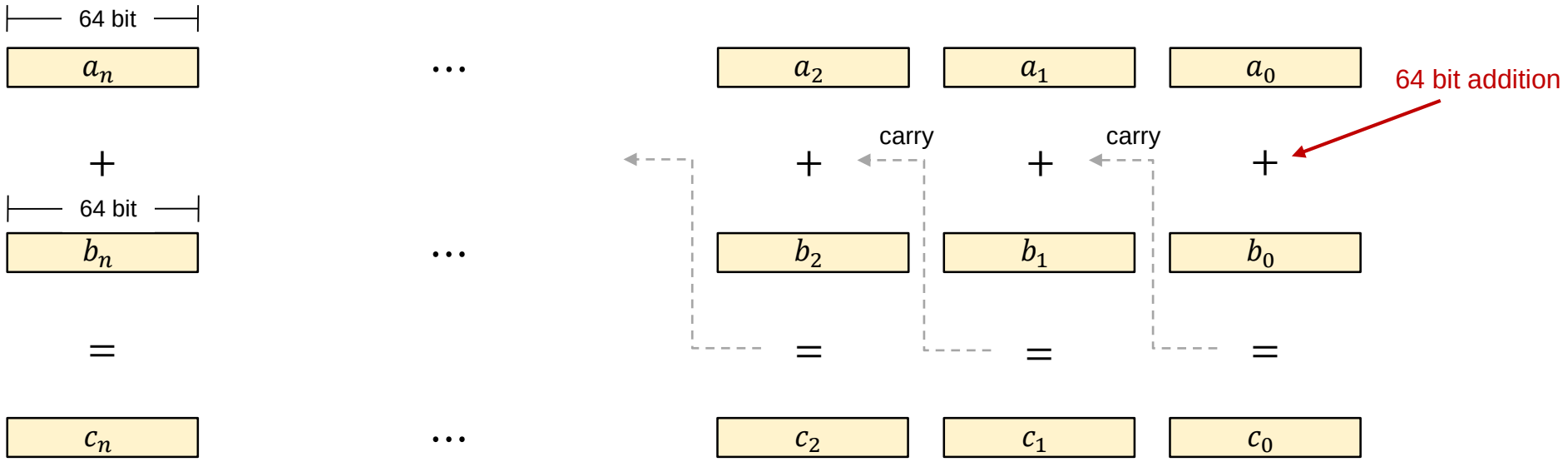
Big-number arithmetic

$a =$ 323170060713110073003389139264238282488179412411402391128420097514007417066343542226196894173635693471179017379097041917546058732091950288537589861856221532121754125149017745202702357960782362488842461894775
876411059286460994117232454266225221932305409190376805242355191256797158701170010580558776510388618472802579760549035697325615261670813393617995413364765591603683178967290731783845896806396719009772021941686
472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905 59

$b =$ 665680077810100734070476416898417408020670352441378967997224101900193957211854827569676933273935982997835638067380762809024311842715496660113976613131478051784487322446809608196031803691263486170912693625523
249742383264476433978409434004770395167011614217279552198029936108023398643999746539201834933168220864789880837484536563067997761270826642223539437541791058416731683176995365899867720283590420932417377927106
884773199080517477615608675490982255098980545292032908358093913531433702049429047413071525556307299541445514389721222256380833743627991333762777307118317040885305789910094293548481550130956194277068752 4

$a + b = ?$

Time complexity: $\mathcal{O}(n)$



Big-number arithmetic

$a =$ 323170060713110073003389139264238282488179412411402391128420097514007417066343542226196894173635693471179017379097041917546058732091950288537589861856221532121754125149017745202702357960782362488842461894775
876411059286460994117232454266225221932305409190376805242355191256797158701170010580558776510388618472802579760549035697325615261670813393617995413364765591603683178967290731783845896806396719009772021941686
472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905 59

$b =$ 665680077810100734070476416898417408020670352441378967997224101900193957211854827569676933273935982997835638067380762809024311842715496660113976613131478051784487322446809608196031803691263486170912693625523
249742383264476433978409434004770395167011614217279552198029936108023398643999746539201834933168220864789880837484536563067997761270826642223539437541791058416731683176995365899867720283590420932417377927106
884773199080517477615608675490982255098980545292032908358093913531433702049429047413071525556307299541445514389721222256380833743627991333762777307118317040885305789910094293548481550130956194277068752 4

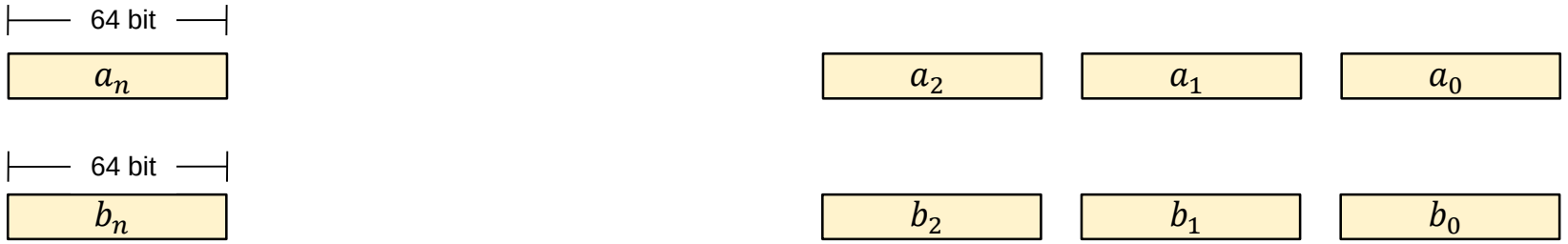
64 bit addition

$a \cdot b = ?$

			⋯	carry	carry		
a_n	a_{n-1}	⋯	a_3	a_2	a_1	a_0	
b_n	b_{n-1}	⋯	b_3	b_2	b_1	b_0	
$a_n b_0$	$a_{n-1} b_0$	⋯	$a_3 b_0$	$a_2 b_0$	$a_1 b_0$	$a_0 b_0$	
+ $a_n b_1$	$a_{n-1} b_1$		⋯	$a_2 b_1$	$a_1 b_1$	$a_0 b_1$	
+ $a_n b_2$	$a_{n-1} b_2$		⋯	$a_2 b_2$	$a_1 b_2$	$a_0 b_2$	
⋮							
+ $a_n b_n$	$a_{n-1} b_n$	⋯	$a_2 b_n$	$a_1 b_n$	$a_0 b_n$		
c_{2n}	c_{2n-1}	⋯		c_3	c_2	c_1	c_0

64 bit multiplication

Big-number arithmetic



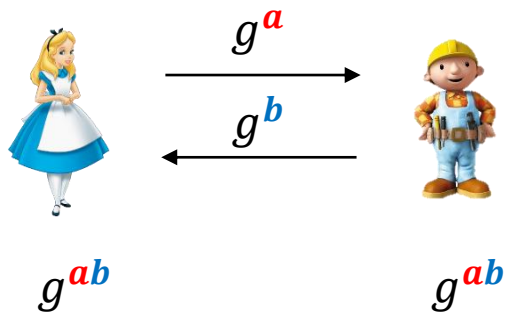
Algorithm	Input	Output	Time
ADD	a, b	$a + b$	$\mathcal{O}(n)$
MULT	a, b	ab	$\mathcal{O}(n^2)$
MOD	a, p	$a \bmod p$	$\mathcal{O}(n^2)$
MOD-INV	a	$a^{-1} \bmod p$	$\mathcal{O}(n^2)$

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

* $n > 2^{713739807325663489766475852620783120641}$

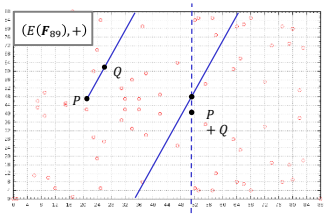
Diffie-Hellman – computations

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



Group exponentiation g^a

- $\left(E\left(\boldsymbol{F}_p\right),+\right)$
  - 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 Q (one-time)
  - Compute $aQ = Q + Q + \cdots + Q$ (every DH exchange)
    - Point addition

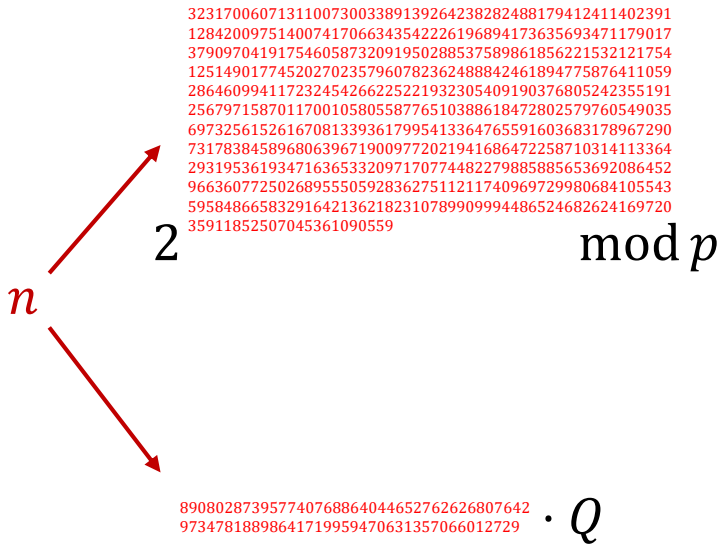


Computing in groups – exponentiation

Goal: compute $g^n \in G$

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

- Running time: n group operations
- $(\boldsymbol{Z}_p^*, \cdot)$: $n \approx 2^{2048}$
- $(E(\boldsymbol{F}_p), +)$: $n \approx 2^{256}$
- Not feasible



Computing in groups – exponentiation; square-and-multiply

Goal: compute $g^n \in G$

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

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

g^{43}

$43 = 101011_2$

$$y \leftarrow e$$
$$y \leftarrow y^2 \circ g^1 \quad = g^1$$
$$y \leftarrow y^2 \circ g^0 \quad = g^2$$
$$y \leftarrow y^2 \circ g^1 \quad = g^{4+1} = g^5$$
$$y \leftarrow y^2 \circ g^0 \quad = g^{10}$$
$$y \leftarrow y^2 \circ g^1 \quad = g^{20+1} = g^{21}$$
$$y \leftarrow y^2 \circ g^1 \quad = g^{42+1} = \underline{g^{43}}$$

Computing in groups – exponentiation; square-and-multiply

Goal: compute $g^n \in G$

g^{43}

$n = (b_5 b_4 b_3 b_2 b_1 b_0)_2$

$= b_5 \cdot 2^5 + b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0$

$43 = 101011_2$

$y \leftarrow e$

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

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

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

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

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

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

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

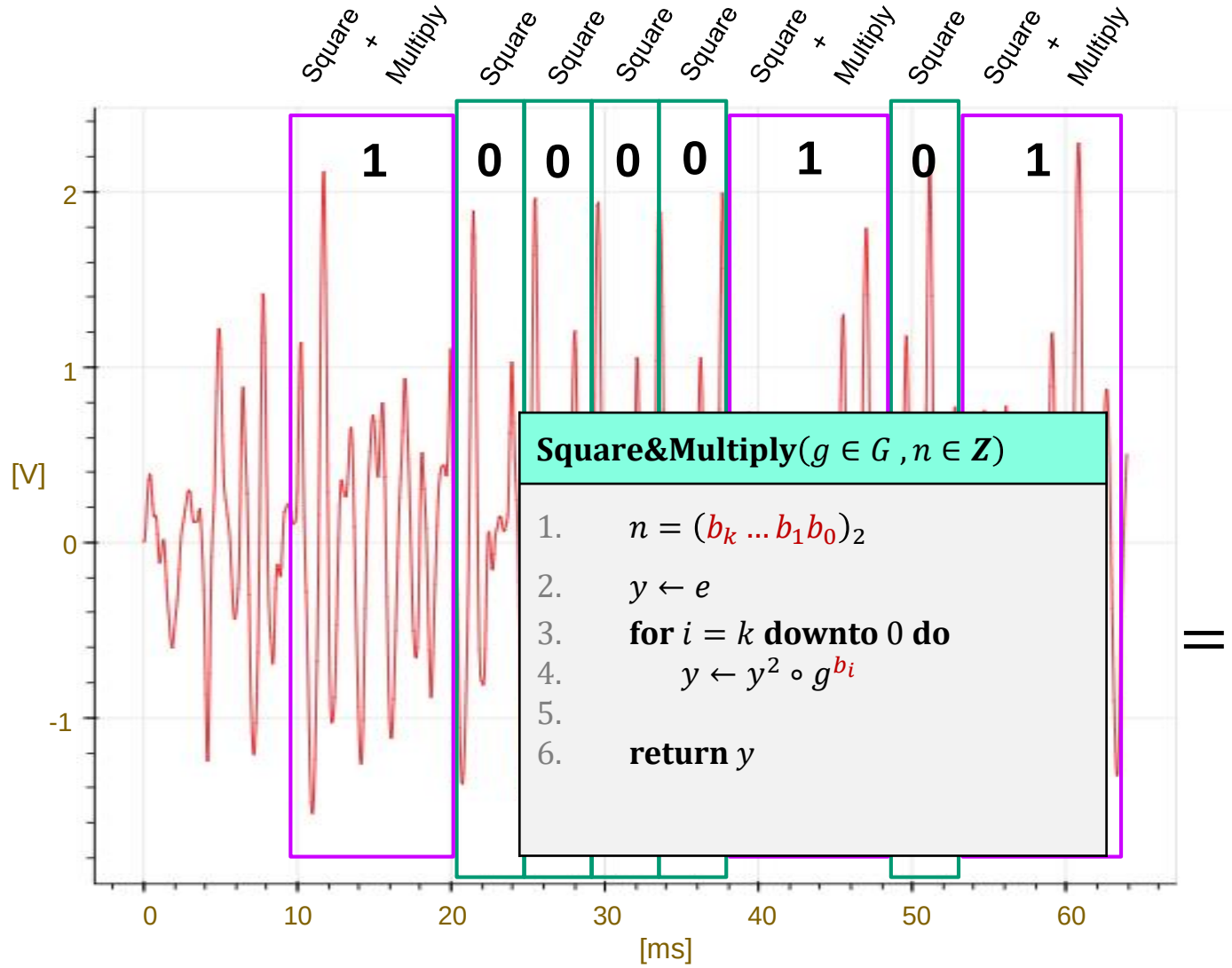
1. $n = (b_k \ldots b_1 b_0)_2$
2. $y \leftarrow e$
3. **for** $i = k$ **downto** 0 **do**
4. $\quad y \leftarrow y^2 \circ g^{b_i}$
5.
6. **return** y

Square-and-multiply

- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Naïve exponentiation: $\mathcal{O}(n) \approx p$ $\approx 2^{2048}$
  - Square-and-multiply: $\mathcal{O}(\log_2 n) \approx \log_2 p$ ≈ 2048

- $\left(E\left(\boldsymbol{F}_p\right), +\right)$
  - Naïve "exponentiation" (i.e., point multiplication nQ): $\mathcal{O}(n) \approx \left|E\left(\boldsymbol{F}_p\right)\right|$ $\approx 2^{256}$
  - "Square-and-multiply" (i.e., double-and-add): $\mathcal{O}(\log_2 n) \approx \log_2 \left|E\left(\boldsymbol{F}_p\right)\right|$ ≈ 256

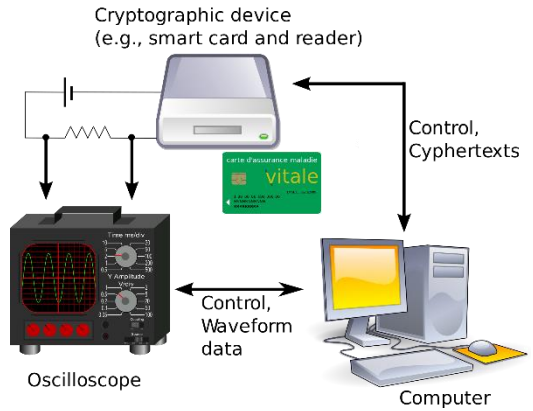
Side-channel attacks – power analysis



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

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

=

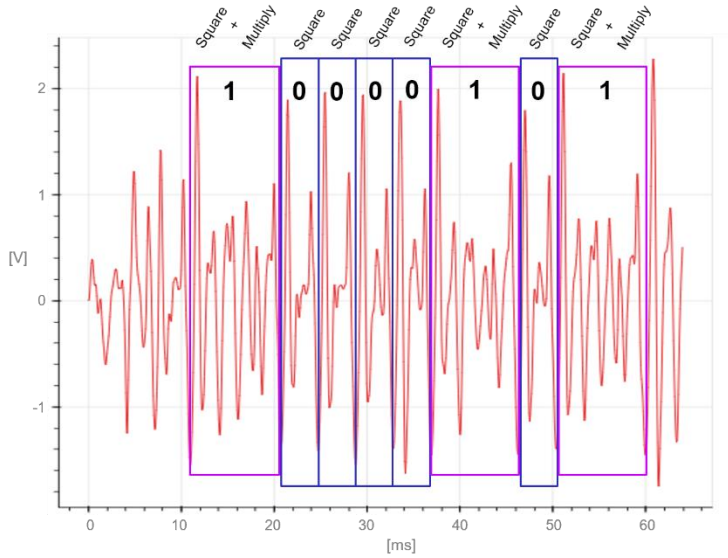


Square&Multiply$(g \in G, n \in \boldsymbol{Z})$

1. $n = (b_k \dots b_1 b_0)_2$
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. $\quad z \leftarrow y \circ g$
6. $\quad$ **if** $b_i = 1$ **then**
7. $\quad\quad y \leftarrow z$
8. **return** y

// always perform multiplication

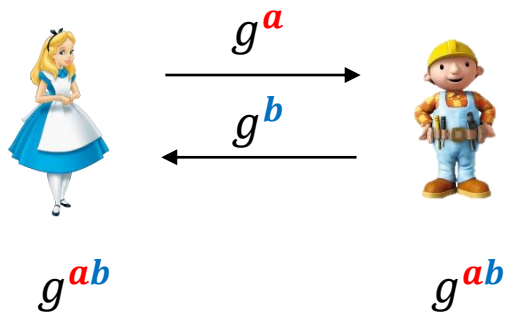
Square&Multiply$(g \in G, n \in \boldsymbol{Z})$

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

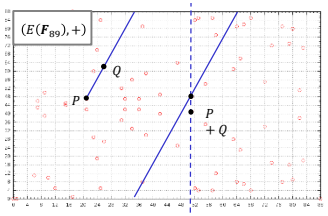
- Wastes computations: $\approx t/2$ of multiplications thrown away
- Better solutions exist

Diffie-Hellman – computations

- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Find prime p (one-time)
  - Find generator $\langle g \rangle = \left(\boldsymbol{Z}_p^*, \cdot\right)$ (one-time)
  - Compute $g^a = gg \cdots g \mod p$ (every DH exchange)
    - 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 Q (one-time)
  - Compute $aQ = Q + Q + \cdots + Q$ (every DH exchange)
    - Point addition



Finding prime numbers

- We need *large* prime numbers for both $(\boldsymbol{Z}_p^*, \cdot)$ 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
  - $\pi(n) \stackrel{\text{def}}{=}$ "#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. $\quad n \stackrel{\$}{\leftarrow}$ 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	$\approx 10^3$	$\approx 10^6$	$\approx 10^9$	$\approx 10^{12}$

Finding prime numbers

- GenPrime efficient?
  - prime-to-non-prime ratio ✓
  - $\text{IsPrime}(n)$?

- 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: $n \approx 2^{2048} \implies \sqrt{\pi(n)} \approx \sqrt{\frac{2^{2048}}{\ln 2^{2048}}} \approx 2^{1000}$

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

GenPrime(k)

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

Fermat's theorem

Theorem: if $(G,\circ)$ is a finite group, then for all $g \in G$:

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

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

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

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

$A \quad \overset{?}{\not\Longrightarrow} \quad B$

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

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

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

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

Carmichael number

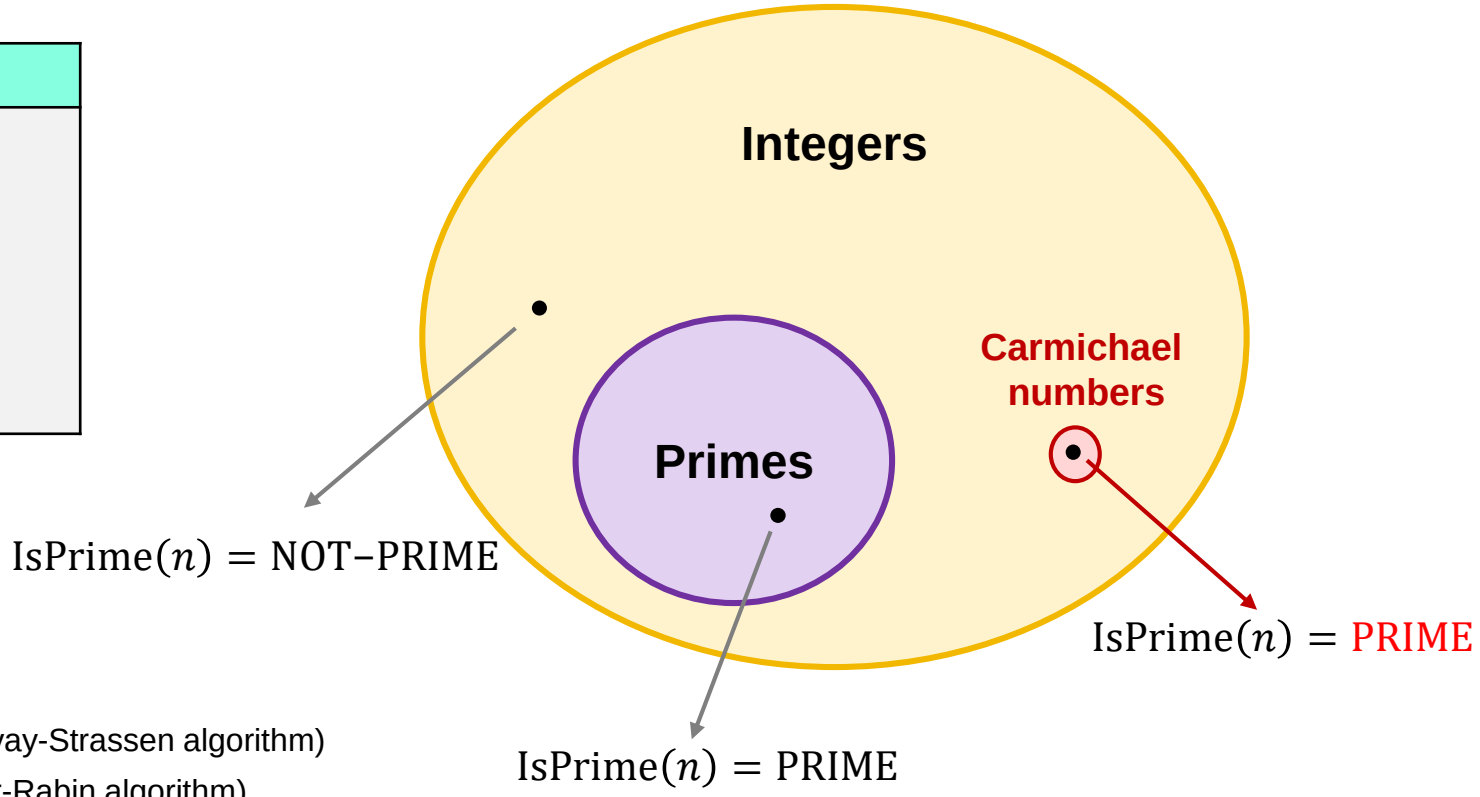
Fermat primality test

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

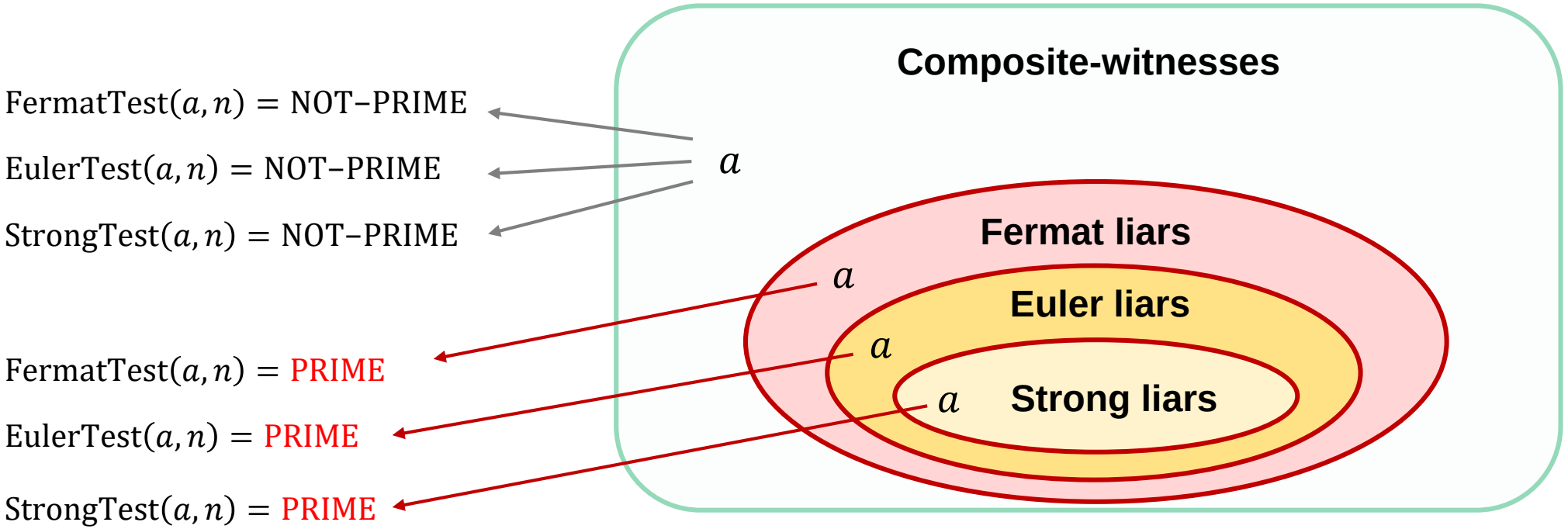
1. **for** $a \in \{2, 3, \dots, t\}$ **do**
2. **if** $\neg$FermatTest(a, n) **then**
3. **return** NOT-PRIME
4. **return** PRIME

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** $a \in \{2, 3, \dots, t\}$ **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!

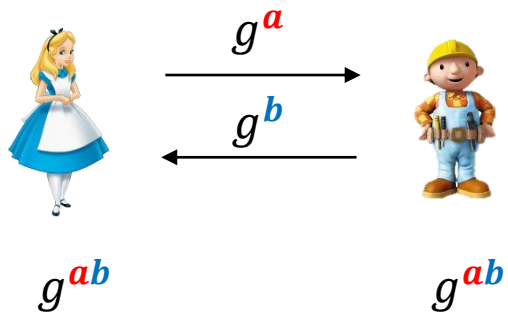
Lagrange's theorem:

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

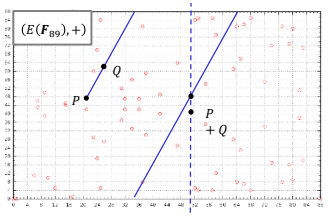
- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Not prime-order!
  - Common in practice: pick element at random; check if generator

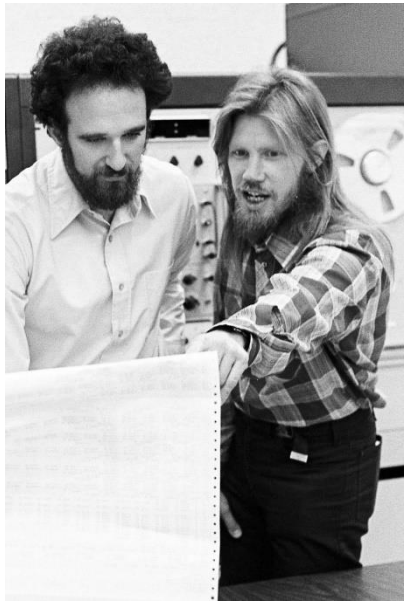
Diffie-Hellman – computations

- $\left(\boldsymbol{Z}_p^*, \cdot\right)$
  - Find prime p (one-time)
  - Find generator $\langle g \rangle = \left(\boldsymbol{Z}_p^*, \cdot\right)$ (one-time)
  - Compute $g^a = gg \cdots g \mod p$ (every DH exchange)
    - 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) Tricky! Standardized choices (NIST P-256, Curve25519)
  - Find curve group order $\left|E\left(\boldsymbol{F}_p\right)\right|$ (one-time) Tricky! (Schoof's algorithm) $\left|E\left(\boldsymbol{F}_p\right)\right| \neq p$ (typically)
  - Find generator Q (one-time) Easy! ($\left|E\left(\boldsymbol{F}_p\right)\right|$ usually prime)
  - Compute $aQ = Q + Q + \cdots + Q$ (every DH exchange)
    - Point addition





Attacking Diffie-Hellman

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

$G = \langle g \rangle$

$A = g^a$

$X = g^x$

$B = g^b$

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

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

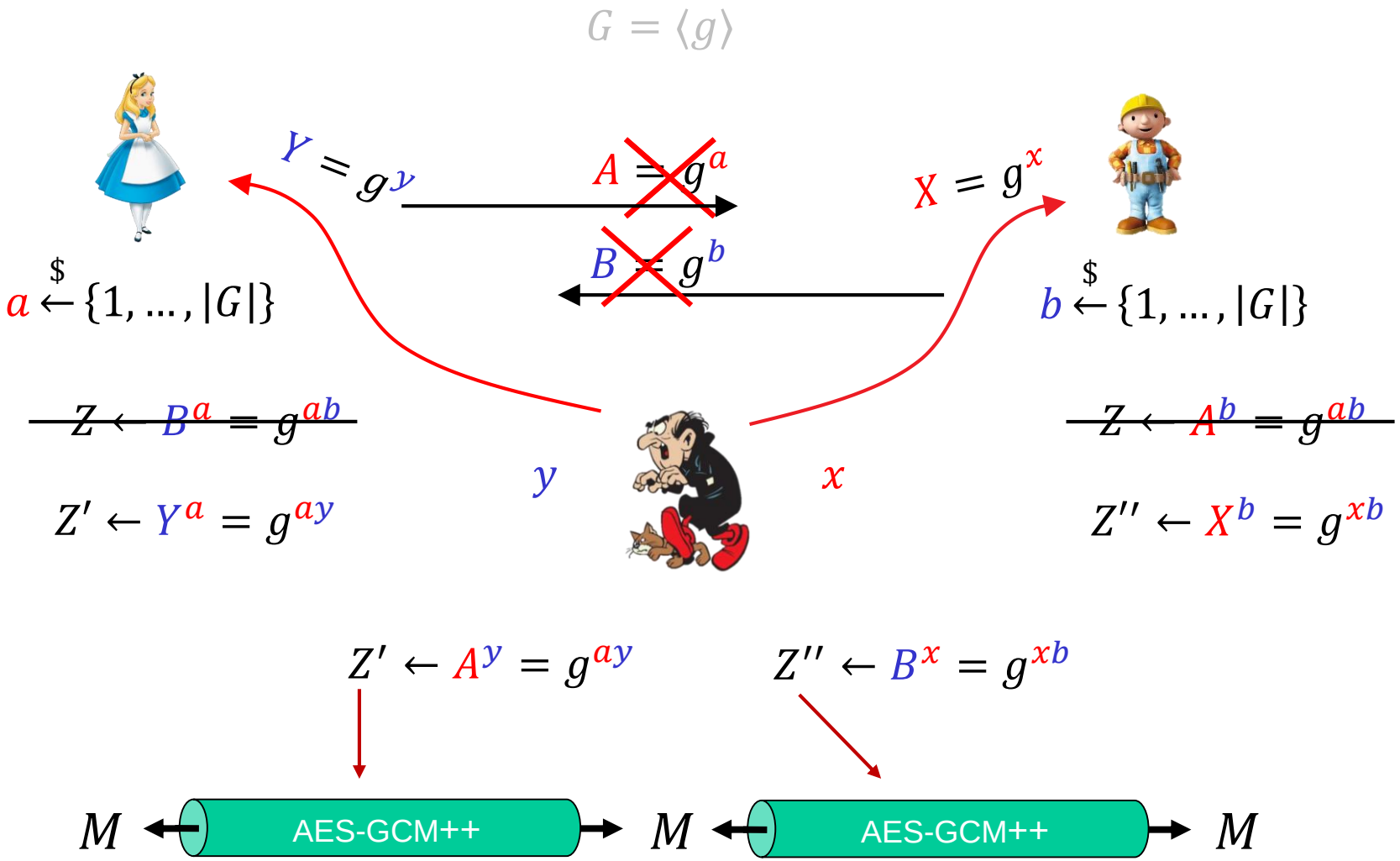
$Z \leftarrow B^a = g^{ab}$

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

$X = g^x$ →

← $Y = g^y$

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

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

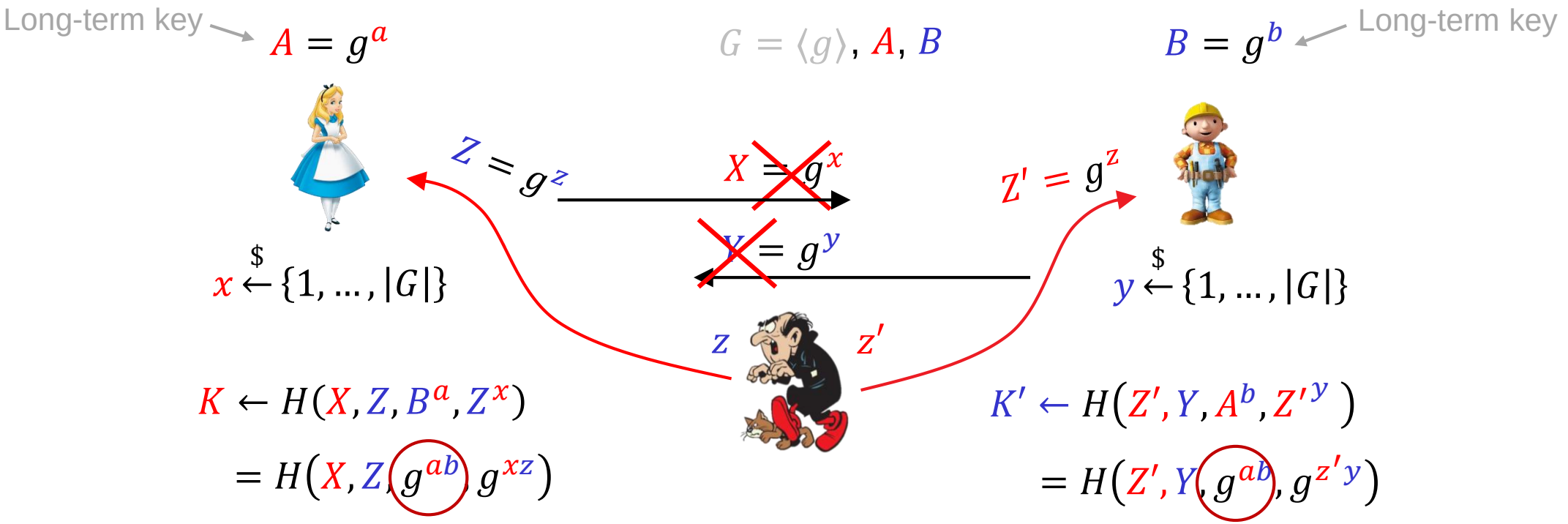
$K \leftarrow H(X, Y, B^a, Y^x)$

$= H(X, Y, g^{ab}, g^{xy})$

$K \leftarrow H(X, Y, A^b, X^y)$

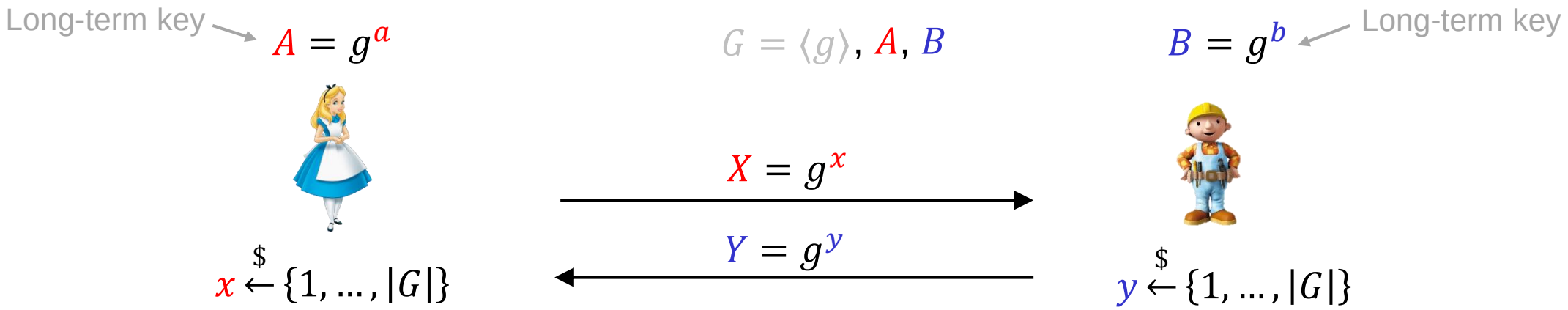
$= H(X, Y, g^{ab}, g^{xy})$

Noise-protocol



Can't compute K or K'!

Noise-protocol

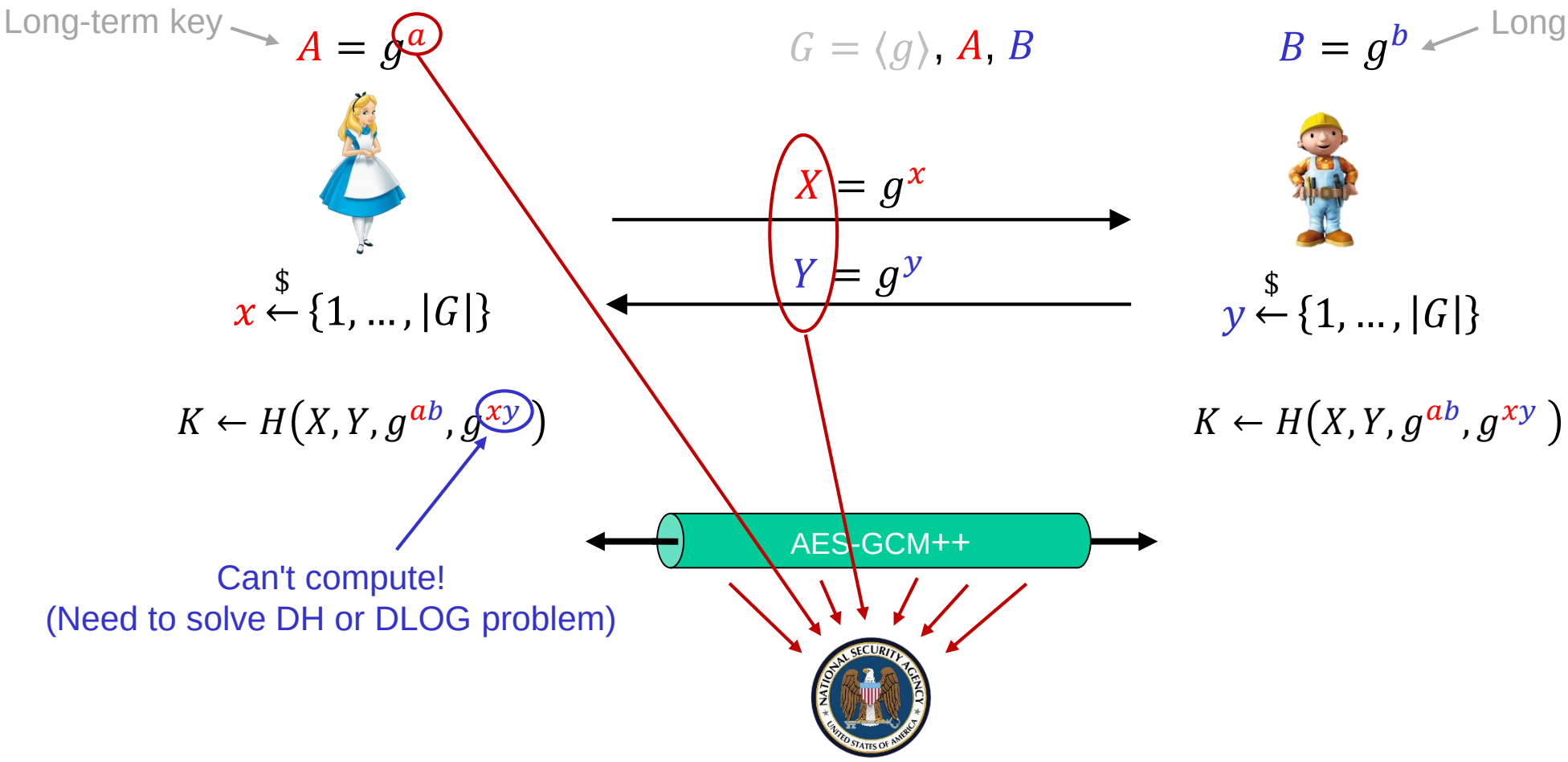


Many alternatives:

Used inside 

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

Forward secrecy

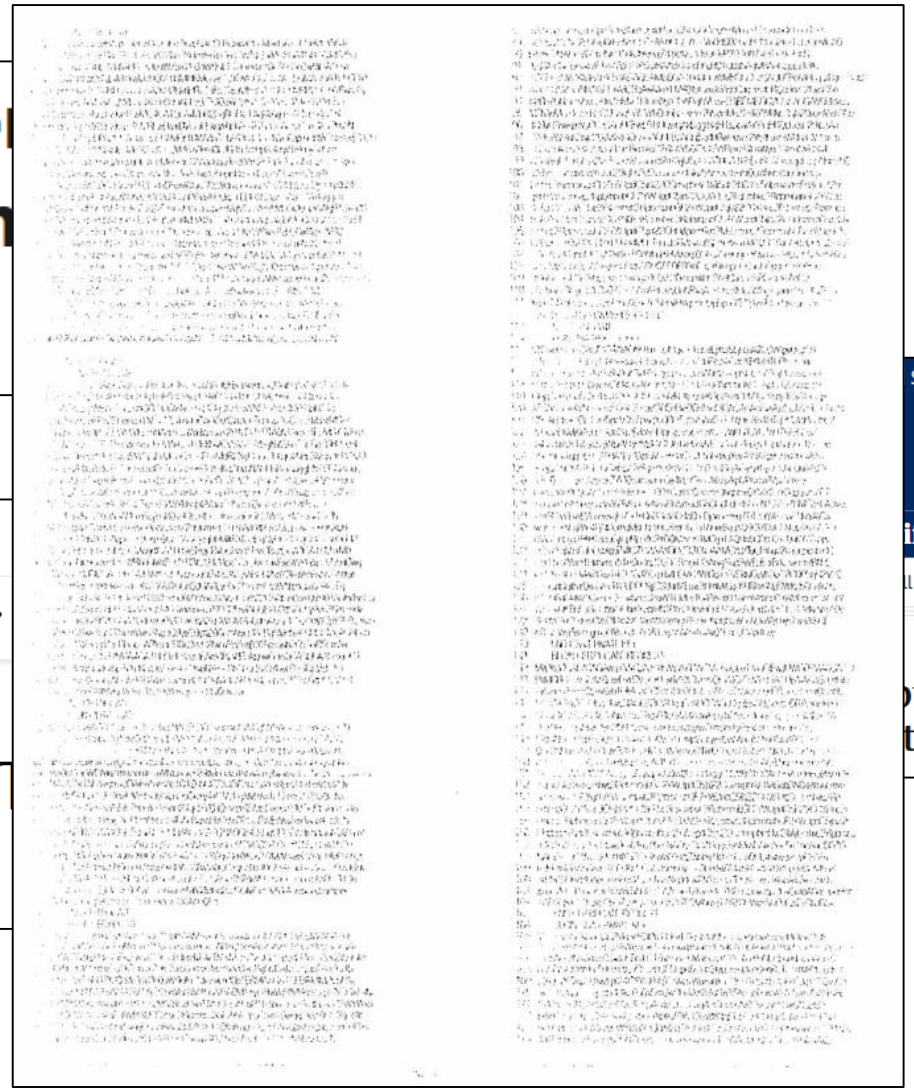


Forward secrecy: attacker can't obtain old traffic keys even if it obtains the parties' long-term keys

Lavabit

Lavabit email se
abruptly, blamin

Graham Cluley • @gcluley
9:44 pm, August 8, 2013



Get 12 weeks for ~~$29.99~~ $6

SCIENCE AND TECH Can Computers Learn Common Sense?

ANNALS OF TECHNOLOGY

HOW LAVABIT MELT

By Michael Phillips and Matt Buchanan
October 7, 2013

Search jobs | Sign in | Search | The Guardian | International e
News website of the year

ifestyle More

l Tech Business Obituaries

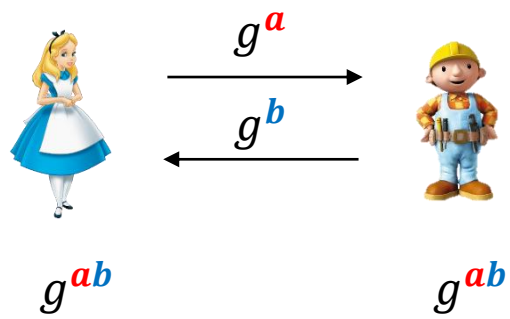
tly shut
terference

Most viewed

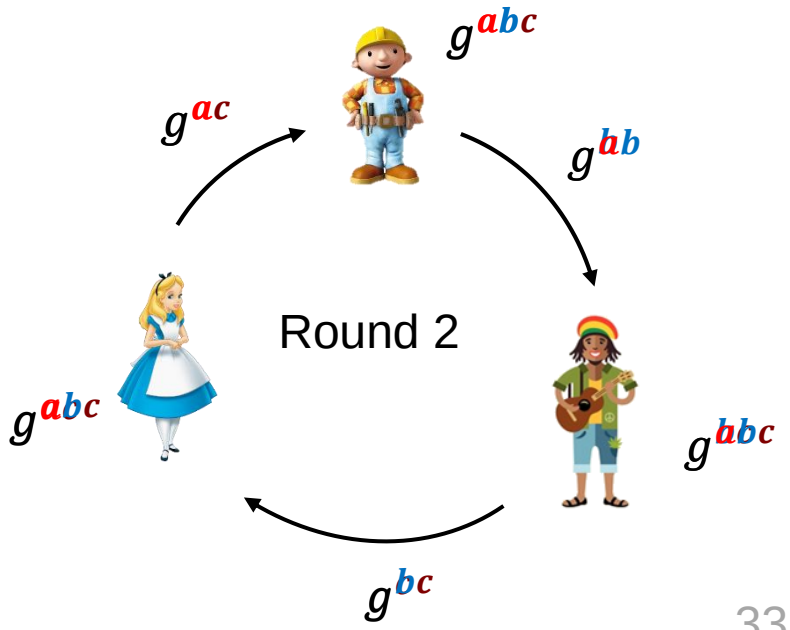
Live Russia-Ukraine wa
live: 'Heaviest of battles
ahead in Kherson, says

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
    - Straightforward implementation 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