COMPUTATIONAL NUMBER THEORY



## Secret key exchange

**Problem:** Obtain a joint secret key via interaction over a public channel:

| **Alice** | | **Bob** |
|---|---|---|
| $x \xleftarrow{\$} \ldots;\ X \leftarrow \ldots$ | | |
| | $\xrightarrow{X}$ | |
| | | $y \xleftarrow{\$} \ldots;\ Y \leftarrow \ldots$ |
| | $\xleftarrow{Y}$ | |
| $K_A \leftarrow F_A(x, Y)$ | | $K_B \leftarrow F_B(y, X)$ |

Desired properties of the protocol:

- $K_A = K_B$, meaning Alice and Bob agree on a key
- Adversary given $X, Y$ can't compute $K_A$



## Secret Key Exchange

Can you build a secret key exchange protocol?



## Secret Key Exchange

Can you build a secret key exchange protocol?

Symmetric cryptography has existed for thousands of years.

But no secret key exchange protocol was found in that time.

Many people thought it was impossible.

## Secret Key Exchange

Can you build a secret key exchange protocol?

Symmetric cryptography has existed for thousands of years.

But no secret key exchange protocol was found in that time.

Many people thought it was impossible.

In 1976, Diffie and Hellman proposed one.

This was the birth of public-key (asymmetric) cryptography.



## DH Key Exchange Video

http://www.youtube.com/watch?v=3QnD2c4Xovk



## DH Secret Key Exchange

The following are assumed to be public: A large prime $p$ and a number $g$ called a generator mod $p$. Let $\mathsf{Z}_{p-1} = \{0, 1, \ldots, p-2\}$.

| **Alice** | | **Bob** |
|---|---|---|
| $x \xleftarrow{\$} \mathsf{Z}_{p-1};\ X \leftarrow g^x \bmod p$ | | |
| | $\xrightarrow{X}$ | |
| | | $y \xleftarrow{\$} \mathsf{Z}_{p-1};\ Y \leftarrow g^y \bmod p$ |
| | $\xleftarrow{Y}$ | |
| $K_A \leftarrow Y^x \bmod p$ | | $K_B \leftarrow X^y \bmod p$ |

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo $p$, so $K_A = K_B$
- Adversary is faced with computing $g^{xy} \bmod p$ given $g^x \bmod p$ and $g^y \bmod p$, which nobody knows how to do efficiently for large $p$.



## DH Secret Key Exchange: Questions

- How do we pick a large prime $p$, and how large is large enough?
- What does it mean for $g$ to be a generator modulo $p$?
- How do we find a generator modulo $p$?
- How can Alice quickly compute $x \mapsto g^x \bmod p$?
- How can Bob quickly compute $y \mapsto g^y \bmod p$?
- Why is it hard to compute $(g^x \bmod p, g^y \bmod p) \mapsto g^{xy} \bmod p$?
- ...

To answer all that and more, we will forget about DH secret key exchange for a while and take a trip into computational number theory ...

## Notation

$\mathsf{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

$\mathsf{N} = \{0, 1, 2, \ldots\}$

$\mathsf{Z}_+ = \{1, 2, 3, \ldots\}$

For $a, N \in \mathsf{Z}$ let $\gcd(a, N)$ be the largest $d \in \mathsf{Z}_+$ such that $d$ divides both $a$ and $N$.

Example: $\gcd(30, 70) = 10$.



## Integers mod $N$

For $N \in \mathsf{Z}_+$, let

- $\mathsf{Z}_N = \{0, 1, \ldots, N-1\}$
- $\mathsf{Z}_N^* = \{a \in \mathsf{Z}_N \ : \ \gcd(a, N) = 1\}$
- $\varphi(N) = |\mathsf{Z}_N^*|$

Example: $N = 12$

- $\mathsf{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $\mathsf{Z}_{12}^* =$



## Integers mod $N$

For $N \in \mathsf{Z}_+$, let

- $\mathsf{Z}_N = \{0, 1, \ldots, N-1\}$
- $\mathsf{Z}_N^* = \{a \in \mathsf{Z}_N \ : \ \gcd(a, N) = 1\}$
- $\varphi(N) = |\mathsf{Z}_N^*|$

Example: $N = 12$

- $\mathsf{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $\mathsf{Z}_{12}^* = \{1, 5, 7, 11\}$
- $\varphi(12) =$



## Integers mod $N$

For $N \in \mathsf{Z}_+$, let

- $\mathsf{Z}_N = \{0, 1, \ldots, N-1\}$
- $\mathsf{Z}_N^* = \{a \in \mathsf{Z}_N \ : \ \gcd(a, N) = 1\}$
- $\varphi(N) = |\mathsf{Z}_N^*|$

Example: $N = 12$

- $\mathsf{Z}_{12} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $\mathsf{Z}_{12}^* = \{1, 5, 7, 11\}$
- $\varphi(12) = 4$

## Division and mod

$\text{INT-DIV}(a, N)$ returns $(q, r)$ such that

- $a = qN + r$
- $0 \leq r < N$

Refer to $q$ as the quotient and $r$ as the remainder. Then

$$a \bmod N = r \in \mathbb{Z}_N$$

is the remainder when $a$ is divided by $N$.

Example: $\text{INT-DIV}(17, 3) = (5, 2)$ and $17 \bmod 3 = 2$.

**Def:** $a \equiv b \pmod{N}$ if $a \bmod N = b \bmod N$.

Example: $17 \equiv 14 \pmod{3}$



## Groups

Let $G$ be a non-empty set, and let $\cdot$ be a binary operation on $G$. This means that for every two points $a, b \in G$, a value $a \cdot b$ is defined.

Example: $G = \mathbb{Z}_{12}^*$ and "$\cdot$" is multiplication modulo 12, meaning

$$a \cdot b = ab \bmod 12$$

**Def:** We say that $G$ is a *group* if it has four properties called closure, associativity, identity and inverse that we present next.

**Fact:** If $N \in \mathbb{Z}_+$ then $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ is a group.



## Groups: Closure

**Closure:** For every $a, b \in G$ we have $a \cdot b$ is also in $G$.

Example: $G = \mathbb{Z}_{12}$ with $a \cdot b = ab$ does not have closure because $7 \cdot 5 = 35 \notin \mathbb{Z}_{12}$.

**Fact:** If $N \in \mathbb{Z}_+$ then $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies closure, meaning

$$\gcd(a, N) = \gcd(b, N) = 1 \text{ implies } \gcd(ab \bmod N, N) = 1$$

**Example:** Let $G = \mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$. Then

$$5 \cdot 7 \bmod 12 = 35 \bmod 12 = 11 \in \mathbb{Z}_{12}^*$$

**Exercise:** Prove the above Fact.



## Groups: Associativity

**Associativity:** For every $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

**Fact:** If $N \in \mathbb{Z}_+$ then $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ satisfies associativity, meaning

$$((ab \bmod N)c) \bmod N = (a(bc \bmod N)) \bmod N$$

Example:

$$\begin{aligned}(5 \cdot 7 \bmod 12) \cdot 11 \bmod 12 &= (35 \bmod 12) \cdot 11 \bmod 12 \\ &= 11 \cdot 11 \bmod 12 = 1 \\ 5 \cdot (7 \cdot 11 \bmod 12) \bmod 12 &= 5 \cdot (77 \bmod 12) \bmod 12 \\ &= 5 \cdot 5 \bmod 12 = 1\end{aligned}$$

**Exercise:** Given an example of a set $G$ and a natural operation $a, b \mapsto a \cdot b$ on $G$ that satisfies closure but *not* associativity.

## Groups: Identity element

**Identity element:** There exists an element $1 \in G$ such that $a \cdot 1 = 1 \cdot a = a$ for all $a \in G$.

**Fact:** If $N \in \mathbb{Z}_+$ and $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ then 1 is the identity element because $a \cdot 1 \bmod N = 1 \cdot a \bmod N = a$ for all $a$.



## Groups: Inverses

**Inverses:** For every $a \in G$ there exists a unique $b \in G$ such that $a \cdot b = b \cdot a = 1$.

This $b$ is called the inverse of $a$ and is denoted $a^{-1}$ if $G$ is understood.

**Fact:** If $N \in \mathbb{Z}_+$ and $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ then $\forall a \in \mathbb{Z}_N^* \quad \exists b \in \mathbb{Z}_N^*$ such that $a \cdot b \bmod N = 1$.

We denote this unique inverse $b$ by $a^{-1} \bmod N$.

Example: $5^{-1} \bmod 12$ is the $b \in \mathbb{Z}_{12}^*$ satisfying $5b \bmod 12 = 1$, so $b =$



## Groups: Inverses

**Inverses:** For every $a \in G$ there exists a unique $b \in G$ such that $a \cdot b = b \cdot a = 1$.

This $b$ is called the inverse of $a$ and is denoted $a^{-1}$ if $G$ is understood.

**Fact:** If $N \in \mathbb{Z}_+$ and $G = \mathbb{Z}_N^*$ with $a \cdot b = ab \bmod N$ then $\forall a \in \mathbb{Z}_N^* \quad \exists b \in \mathbb{Z}_N^*$ such that $a \cdot b \bmod N = 1$.

We denote this unique inverse $b$ by $a^{-1} \bmod N$.

Example: $5^{-1} \bmod 12$ is the $b \in \mathbb{Z}_{12}^*$ satisfying $5b \bmod 12 = 1$, so $b = 5$



## Examples of groups

**Fact:** If $N \geq 1$ is an integer then $\mathbb{Z}_N$ is a group under the operation of addition modulo $N$, namely $a \cdot b = (a + b) \bmod N$.

**Fact:** If $N \geq 2$ is an integer then $\mathbb{Z}_N^*$ is a group under the operation of multiplication modulo $N$, namely $a \cdot b = (ab) \bmod N$.

## Exercises

Let $n \in \mathbb{Z}_+$ and let $G = \{0,1\}^n$. Prove that $G$ is a group under the operation $a \cdot b = a \oplus b$.

Let $n \in \mathbb{Z}_+$ and let $G = \{0,1\}^n$. Prove that $G$ is *not* a group under the operation $a \cdot b = a \wedge b$. (This is bit-wise AND, for example $0110 \wedge 1101 = 0100$.)



## Computational Shortcuts

Fact: Let $a, b, c \in \mathbb{Z}$ and $N \in \mathbb{Z}_+$. Then

$$abc \bmod N = ((ab \bmod N)\, c) \bmod N$$

**Example:** What is $5 \cdot 8 \cdot 10 \cdot 16 \bmod 21$?

Slow way:

- $5 \cdot 8 \cdot 10 \cdot 16 = 40 \cdot 10 \cdot 16 = 400 \cdot 16 = 6400$
- $6400 \bmod 21 = 16$

Faster way, using above Fact:

- $5 \cdot 8 \bmod 21 = 40 \bmod 21 = 19$
- $19 \cdot 10 \bmod 21 = 190 \bmod 21 = 1$
- $1 \cdot 16 \bmod 21 = 16$



## Exponentiation

Let $G$ be a group and $a \in G$. We let $a^0 = 1$ be the identity element and for $n \geq 1$, we let

$$a^n = \underbrace{a \cdot a \cdots a}_{n}.$$

Also we let

$$a^{-n} = \underbrace{a^{-1} \cdot a^{-1} \cdots a^{-1}}_{n}.$$

This ensures that for all $i, j \in \mathbb{Z}$,

- $a^{i+j} = a^i \cdot a^j$
- $a^{ij} = (a^i)^j = (a^j)^i$
- $a^{-i} = (a^i)^{-1} = (a^{-1})^i$

Meaning we can manipulate exponents "as usual".



## Group Orders

The order of a group $G$ is its size $|G|$, meaning the number of elements in it.

Example: The order of $\mathbb{Z}_{21}^*$ is

## Group Orders

The order of a group $G$ is its size $|G|$, meaning the number of elements in it.

Example: The order of $Z_{21}^*$ is 12 because

$$Z_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$$

**Fact:** Let $G$ be a group of order $m$ and $a \in G$. Then, $a^m = 1$.

Examples: Modulo 21 we have

- $5^{12} \equiv (5^3)^4 \equiv 20^4 \equiv (-1)^4 \equiv 1$
- $8^{12} \equiv (8^2)^6 \equiv (1)^6 \equiv 1$



## Simplifying exponentiation

**Fact:** Let $G$ be a group of order $m$ and $a \in G$. Then, $a^m = 1$.

**Corollary:** Let $G$ be a group of order $m$ and $a \in G$. Then for any $i \in Z$,

$$a^i = a^{i \bmod m}.$$

Proof: Let $(q, r) \leftarrow \text{INT-DIV}(i, m)$, so that $i = mq + r$ and $r = i \bmod m$. Then

$$a^i = a^{mq+r} = (a^m)^q \cdot a^r$$

But $a^m = 1$ by Fact.



## Simplifying exponentiation

**Corollary:** Let $G$ be a group of order $m$ and $a \in G$. Then for any $i \in Z$,

$$a^i = a^{i \bmod m}.$$

**Example:** What is $5^{74} \bmod 21$?



## Simplifying exponentiation

**Corollary:** Let $G$ be a group of order $m$ and $a \in G$. Then for any $i \in Z$,

$$a^i = a^{i \bmod m}.$$

**Example:** What is $5^{74} \bmod 21$?

**Solution:** Let $G = Z_{21}^*$ and $a = 5$. Then, $m = 12$, so

$$\begin{aligned} 5^{74} \bmod 21 &= 5^{74 \bmod 12} \bmod 21 \\ &= 5^2 \bmod 21 \\ &= 4. \end{aligned}$$

## Exercise

Evaluate the expressions shown in the first column. Your answer, in the second column, should be a member of the set shown in the third column. In the first case, the inverse refers to the group $Z^*_{101}$. Don't use any electronic tools; these are designed to be done by hand.

| Expression | Value | In |
|---|---|---|
| $34^{-1} \bmod 101$ | | $Z^*_{101}$ |
| $5^{1602} \bmod 17$ | | $Z^*_{17}$ |
| $\lvert Z^*_{24}\rvert$ | | N |



## Measuring Running Time of Algorithms on Numbers

In an algorithms course, the cost of arithmetic is often assumed to be $\mathcal{O}(1)$, because numbers are small. In cryptography numbers are

very, very BIG!

Typical sizes are $2^{512}$, $2^{1024}$, $2^{2048}$.

Numbers are provided to algorithms in binary. The length of $a$, denoted $|a|$, is the number of bits in the binary encoding of $a$.

**Example:** $|7| = 3$ because 7 is 111 in binary.

Running time is measured as a function of the lengths of the inputs.



## Algorithms on numbers

| Algorithm | Input | Output | Time |
|---|---|---|---|
| ADD | $a,\ b$ | $a+b$ | $\mathcal{O}(\lvert a\rvert + \lvert b\rvert)$ |
| MULT | $a,\ b$ | $ab$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert b\rvert)$ |
| INT-DIV | $a,\ N$ | $q,r$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| MOD | $a,\ N$ | $a \bmod N$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| EXT-GCD | $a,\ N$ | $(d, a', N')$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| MOD-INV | $a \in Z^*_N,\ N$ | $a^{-1} \bmod N$ | $\mathcal{O}(\lvert N\rvert^2)$ |
| MOD-EXP | $a \in Z_N,\ n,\ N$ | $a^n \bmod N$ | $\mathcal{O}(\lvert n\rvert \cdot \lvert N\rvert^2)$ |
| $\mathrm{EXP}_G$ | $a \in G,\ n$ | $a^n \in G$ | $\mathcal{O}(\lvert n\rvert)$ $G$-ops |



## Extended gcd

$\mathrm{EXT\text{-}GCD}(a, N)$ returns $(d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

**Example:** $\mathrm{EXT\text{-}GCD}(12, 20) =$

## Extended gcd

$\text{EXT-GCD}(a, N)$ returns $(d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

**Example:** $\text{EXT-GCD}(12, 20) = (4, -3, 2)$ because

$$4 = \gcd(12, 20) = 12 \cdot (-3) + 20 \cdot 2 \ .$$



## Extended gcd Algorithm

$\text{EXT-GCD}(a, N) \mapsto (d, a', N')$ such that

$$d = \gcd(a, N) = a \cdot a' + N \cdot N' \ .$$

**Lemma:** Let $(q, r) = \text{INT-DIV}(a, N)$. Then, $\gcd(a, N) = \gcd(N, r)$

**Alg** $\text{EXT-GCD}(a, N)$ // $(a, N) \neq (0, 0)$
if $N = 0$ then return $(a, 1, 0)$
else
  $(q, r) \leftarrow \text{INT-DIV}(a, N)$; $(d, x, y) \leftarrow \text{EXT-GCD}(N, r)$
  $a' \leftarrow y$; $N' \leftarrow x - qy$ ; return $(d, a', N')$

Running time is $\mathcal{O}(|a| \cdot |N|)$, so the extended gcd can be computed in quadratic time. If $a \geq N > 0$ then $\text{abs}(a') \leq N$ and $\text{abs}(N') \leq a$ where $\text{abs}(\cdot)$ denotes the absolute value. Analysis showing all this is non-trivial (worst case is Fibonacci numbers).



## Modular Inverse

For $a, N$ such that $\gcd(a, N) = 1$, we want to compute $a^{-1} \bmod N$, meaning the unique $a' \in \mathsf{Z}_N^*$ satisfying $aa' \equiv 1 \pmod N$.

But if we let $(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$ then

$$d = 1 = \gcd(a, N) = a \cdot a' + N \cdot N'$$

But $N \cdot N' \equiv 0 \pmod N$ so $aa' \equiv 1 \pmod N$

**Alg** $\text{MOD-INV}(a, N)$
$(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$
return $a' \bmod N$

Modular inverse can be computed in quadratic time.



## Modular Exponentiation

Let $G$ be a group and $a \in G$. For $n \in \mathsf{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return $y$

Question: Is this a good algorithm?

## Modular Exponentiation

Let $G$ be a group and $a \in G$. For $n \in \mathbb{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return $y$

Question: Is this a good algorithm?

Answer: It is correct but VERY SLOW. The number of group operations is $\mathcal{O}(n) = \mathcal{O}(2^{|n|})$ so it is exponential time. For $n \approx 2^{512}$ it is prohibitively expensive.



## Fast exponentiation idea

We can compute

$$a \longrightarrow a^2 \longrightarrow a^4 \longrightarrow a^8 \longrightarrow a^{16} \longrightarrow a^{32}$$

in just 5 steps by repeated squaring. So we can compute $a^n$ in $i$ steps when $n = 2^i$.

But what if $n$ is not a power of 2?



## Square-and-Multiply Exponentiation Example

Suppose the binary length of $n$ is 5, meaning the binary representation of $n$ has the form $b_4 b_3 b_2 b_1 b_0$. Then

$$\begin{aligned} n &= 2^4 b_4 + 2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0 \\ &= 16 b_4 + 8 b_3 + 4 b_2 + 2 b_1 + b_0 \ . \end{aligned}$$

We want to compute $a^n$. Our exponentiation algorithm will proceed to compute the values $y_5, y_4, y_3, y_2, y_1, y_0$ in turn, as follows:

$$\begin{aligned} y_5 &= 1 \\ y_4 &= y_5^2 \cdot a^{b_4} &&= a^{b_4} \\ y_3 &= y_4^2 \cdot a^{b_3} &&= a^{2b_4 + b_3} \\ y_2 &= y_3^2 \cdot a^{b_2} &&= a^{4b_4 + 2b_3 + b_2} \\ y_1 &= y_2^2 \cdot a^{b_1} &&= a^{8b_4 + 4b_3 + 2b_2 + b_1} \\ y_0 &= y_1^2 \cdot a^{b_0} &&= a^{16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0} \ . \end{aligned}$$



## Square-and-Multiply Exponentiation Algorithm

Let $\mathrm{bin}(n) = b_{k-1} \ldots b_0$ be the binary representation of $n$, meaning

$$n = \sum_{i=0}^{k-1} b_i 2^i$$

**Alg** $\mathrm{EXP}_G(a, n)$ // $a \in G$, $n \geq 1$
$b_{k-1} \ldots b_0 \leftarrow \mathrm{bin}(n)$
$y \leftarrow 1$
for $i = k-1$ downto 0 do $y \leftarrow y^2 \cdot a^{b_i}$
return $y$

The running time is $\mathcal{O}(|n|)$ group operations.

MOD-EXP$(a, n, N)$ returns $a^n$ mod $N$ in time $\mathcal{O}(|n| \cdot |N|^2)$, meaning is cubic time.

## Exercise

Consider the following computational problem:

INPUT: $N, a, b, x, y$ where $N \geq 1$ is an integer, $a, b \in \mathbb{Z}_N^*$ and $x, y$ are integers with $0 \leq x, y < N$
OUTPUT: $a^x b^y \bmod N$

Let $k = |N|$.

1. Consider the algorithm that first computes $X = a^x \bmod N$, then computes $Y = b^y \bmod N$, and returns $XY \bmod N$. Explain why this has worst case cost of $4k + 1$ multiplications modulo $N$.
2. Design an alternative, faster algorithm for this problem that uses at most $2k + 1$ multiplications modulo $N$.



## Algorithms on numbers

| Algorithm | Input | Output | Time |
|---|---|---|---|
| ADD | $a,\ b$ | $a + b$ | $\mathcal{O}(\lvert a\rvert + \lvert b\rvert)$ |
| MULT | $a,\ b$ | $ab$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert b\rvert)$ |
| INT-DIV | $a,\ N$ | $q, r$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| MOD | $a,\ N$ | $a \bmod N$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| EXT-GCD | $a,\ N$ | $(d, a', N')$ | $\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$ |
| MOD-INV | $a \in \mathbb{Z}_N^*,\ N$ | $a^{-1} \bmod N$ | $\mathcal{O}(\lvert N\rvert^2)$ |
| MOD-EXP | $a \in \mathbb{Z}_N,\ n,\ N$ | $a^n \bmod N$ | $\mathcal{O}(\lvert n\rvert \cdot \lvert N\rvert^2)$ |
| $\mathrm{EXP}_G$ | $a \in G,\ n$ | $a^n \in G$ | $\mathcal{O}(\lvert n\rvert)$ $G$-ops |



## Subgroups

Definition: Let $G$ be a group and $S \subseteq G$. Then $S$ is called a subgroup of $G$ if $S$ is itself a group under $G$'s operation.

Example: Let $G = \mathbb{Z}_{11}^*$ and $S = \{1, 2, 3\}$. Then $S$ is not a subgroup because

- $2 \cdot 3 \mod 11 = 6 \notin S$, violating Closure.
- $3^{-1} \mod 11 = 4 \notin S$, violating Inverse.

But $\{1, 3, 4, 5, 9\}$ is a subgroup, as you can check.

Fact: $S$ is a subgroup of $G$ iff $S \neq \emptyset$ and $\forall x, y \in S : xy^{-1} \in S$

The *order* of subgroup $S$ is its size $|S|$.

Fact: If $S$ is a subgroup of $G$ then the order of $S$ divides the order of $G$.



## Generators and cyclic groups

Let $G$ be a group of order $m$ and let $g \in G$. We let

$$\langle g \rangle = \{\, g^i \ :\ i \in \mathbb{Z}_m \,\}\ .$$

Fact: $\langle g \rangle$ is a subgroup of $G$.

The size $|\langle g \rangle|$ of the set $\langle g \rangle$ need not equal m. It could be smaller. It is always a divisor of $m$.

The *order* of $g$ is defined to be $|\langle g \rangle|$.

We say that $g \in G$ is a *generator* (or primitive element) of $G$ if $\langle g \rangle = G$, meaning the order of $g$ is $m$.

We say that $G$ is *cyclic* if it has a generator, meaning there exists $g \in G$ such that $g$ is a generator of $G$.

## Generators and cyclic groups: Example

Let $G = \mathsf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^i \mod 11$ | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 | 1 |
| $5^i \mod 11$ | 1 | 5 | 3 | 4 | 9 | 1 | 5 | 3 | 4 | 9 | 1 |

so

$$\begin{aligned} \langle 2 \rangle &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \langle 5 \rangle &= \{1, 3, 4, 5, 9\} \end{aligned}$$

- 2 a generator because $\langle 2 \rangle = \mathsf{Z}_{11}^*$.
- 5 is not a generator because $\langle 5 \rangle \neq \mathsf{Z}_{11}^*$.
- $\mathsf{Z}_{11}^*$ is cyclic because it has a generator.



## Exercise

Let $G$ be the group $\mathsf{Z}_{10}^*$ under the operation of multiplication modulo 10.

1. List the elements of $G$
2. What is the order of $G$?
3. Determine the set $\langle 3 \rangle$
4. Determine the set $\langle 9 \rangle$
5. Is $G$ cyclic? Why or why not?



## Discrete Logarithms

If $G = \langle g \rangle$ is a cyclic group of order $m$ then for every $a \in G$ there is a unique exponent $i \in \mathsf{Z}_m$ such that $g^i = a$. We call $i$ the discrete logarithm of $a$ to base $g$ and denote it by

$$\mathrm{DLog}_{G,g}(a)$$

The discrete log function is the inverse of the exponentiation function:

$$\begin{aligned} \mathrm{DLog}_{G,g}(g^i) &= i \quad \text{for all } i \in \mathsf{Z}_m \\ g^{\mathrm{DLog}_{G,g}(a)} &= a \quad \text{for all } a \in G. \end{aligned}$$



## Discrete Logarithms: Example

Let $G = \mathsf{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathsf{Z}_{10}$ such that $2^i \bmod 11 = a$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $2^i \mod 11$ | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 |

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathrm{DLog}_{G,2}(a)$ | | | | | | | | | | |

## Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $2^i \bmod 11$ | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 |

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathrm{DLog}_{G,2}(a)$ | 0 | 1 | 8 | 2 | 4 | 9 | 7 | 3 | 6 | 5 |



## Exercise

Let $G$ be the group $\mathbb{Z}_{10}^*$ under the operation of multiplication modulo 10.

1. Show that 3 and 7 are generators of $G$
2. What is $\mathrm{DLog}_{G,3}(7)$?
3. What is $\mathrm{DLog}_{G,7}(9)$?



## Finding Cyclic Groups

Fact 1: Let $p$ be a prime. Then $\mathbb{Z}_p^*$ is cyclic.

Fact 2: Let $G$ be any group whose order $m = |G|$ is a prime number. Then $G$ is cyclic.

Note: $|\mathbb{Z}_p^*| = p - 1$ is not prime, so Fact 2 doesn't imply Fact 1.

Fact 3: If $F$ is a finite field then $F \setminus \{0\}$ is a cyclic group under the multiplication operation of $F$.



## Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order $m$ with generator $g \in G$.

Input: $X \in G$
Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want $x$ such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return $x$

Is this a good algorithm?

## Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order $m$ with generator $g \in G$.

Input: $X \in G$
Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want $x$ such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return $x$

Is this a good algorithm? It is

- Correct (always returns the right answer)



## Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order $m$ with generator $g \in G$.

Input: $X \in G$
Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want $x$ such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return $x$

Is this a good algorithm? It is

- Correct (always returns the right answer), but
- SLOW!

Run time is $O(m)$ exponentiations, which for $G = \mathrm{Z}_p^*$ is $O(p)$, which is exponential time and prohibitive for large $p$.



## Computing Discrete Logs: Best known algorithms

| Group | Time to find discrete logarithms |
|---|---|
| $\mathrm{Z}_p^*$ | $e^{1.92(\ln p)^{1/3}(\ln \ln p)^{2/3}}$ |
| $\mathrm{EC}_p$ | $\sqrt{p} = e^{\ln(p)/2}$ |

Here $p$ is a prime and $\mathrm{EC}_p$ represents an elliptic curve group of order $p$.

Note: In the first case the actual running time is $e^{1.92(\ln q)^{1/3}(\ln \ln q)^{2/3}}$ where $q$ is the largest prime factor of $p-1$.

In neither case is a polynomial-time algorithm known.

This (apparent, conjectured) computational intractability of the discrete log problem makes it the basis for cryptographic schemes in which breaking the scheme requires discrete log computation.



## Discrete logarithm computation records

In $\mathrm{Z}_p^*$:

| $\lvert p \rvert$ in bits | When |
|---|---|
| 431 | 2005 |
| 530 | 2007 |
| 596 | 2014 |
| 795 | 2019 |

For elliptic curves, current record seems to be for $|p|$ around 114.

## Why Elliptic curve (EC) groups?

Say we want 80-bits of security, meaning discrete log computation by the best known algorithm should take time $2^{80}$. Then

- If we work in $Z_p^*$ ($p$ a prime) we need to set $|Z_p^*| = p - 1 \approx 2^{1024}$
- But if we work on an elliptic curve group of prime order $p$ then it suffices to set $p \approx 2^{160}$.

This is because

$$e^{1.92(\ln 2^{1024})^{1/3}(\ln \ln 2^{1024})^{2/3}} \approx \sqrt{2^{160}} = 2^{80}$$

But now:

| Group Size | Cost of Exponentiation |
|---|---|
| $2^{160}$ | 1 |
| $2^{1024}$ | 260 |

Exponentiation will be 260 times faster in the smaller group.



## DL Formally

Let $G = \langle g \rangle$ be a cyclic group of order $m$, and $A$ an adversary.

Game $\mathrm{DL}_{G,g}$

**procedure** Initialize
$x \xleftarrow{\$} Z_m; X \leftarrow g^x$
return $X$

**procedure** Finalize($x'$)
return $(x = x')$

The dl-advantage of $A$ is

$$\mathrm{Adv}^{\mathrm{dl}}_{G,g}(A) = \Pr\left[\mathrm{DL}^A_{G,g} \Rightarrow \mathrm{true}\right]$$



## CDH: The Computational Diffie-Hellman Problem

Let $G = \langle g \rangle$ be a cyclic group of order $m$ with generator $g \in G$. The CDH problem is:

Input: $X = g^x \in G$ and $Y = g^y \in G$
Desired Output: $g^{xy} \in G$

This underlies security of the DH Secret Key Exchange Protocol.

**Obvious algorithm:** $x \leftarrow \mathrm{DLog}_{G,g}(X)$; Return $Y^x$.

So if one can compute discrete logarithms then one can solve the CDH problem.

The converse is an open question. Potentially, there is a way to quickly solve CDH that avoids computing discrete logarithms. But no such way is known.



## CDH Formally

Let $G = \langle g \rangle$ be a cyclic group of order $m$, and $A$ an adversary.

Game $\mathrm{CDH}_{G,g}$

**procedure** Initialize
$x, y \xleftarrow{\$} Z_m$
$X \leftarrow g^x; Y \leftarrow g^y$
return $X, Y$

**procedure** Finalize($Z$)
return $(Z = g^{xy})$

The cdh-advantage of $A$ is

$$\mathrm{Adv}^{\mathrm{cdh}}_{G,g}(A) = \Pr\left[\mathrm{CDH}^A_{G,g} \Rightarrow \mathrm{true}\right]$$

## Building cyclic groups

We will need to build (large) groups over which our cryptographic schemes can work, and find generators in these groups.

How do we do this efficiently?



## Building cyclic groups

To find a suitable prime $p$ and generator $g$ of $Z_p^*$:

- Pick numbers $p$ at random until $p$ is a prime of the desired form
- Pick elements $g$ from $Z_p^*$ at random until $g$ is a generator

For this to work we need to know

- How to test if $p$ is prime
- How many numbers in a given range are primes of the desired form
- How to test if $g$ is a generator of $Z_p^*$ when $p$ is prime
- How many elements of $Z_p^*$ are generators



## Finding primes

Desired: An efficient algorithm that given an integer $k$ returns a prime $p \in \{2^{k-1}, \ldots, 2^k - 1\}$ such that $q = (p-1)/2$ is also prime.

**Alg** Findprime($k$)
do
  $p \xleftarrow{\$} \{2^{k-1}, \ldots, 2^k - 1\}$
until ($p$ is prime and $(p-1)/2$ is prime)
return $p$

- How do we test primality?
- How many iterations do we need to succeed?



## Primality Testing

Given: integer $N$
Output: TRUE if $N$ is prime, FALSE otherwise.

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then return false
return true

## Primality Testing

Given: integer $N$
Output: TRUE if $N$ is prime, FALSE otherwise.

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then return false
return true

Correct but SLOW! $O(N)$ running time, exponential. However, we have:

- $O(|N|^3)$ time randomized algorithms
- Even a $O(|N|^8)$ time deterministic algorithm



## Density of primes

Let $\pi(N)$ be the number of primes in the range $1, \ldots, N$. So if $p \xleftarrow{\$} \{1, \ldots, N\}$ then

$$\Pr[p \text{ is a prime}] = \frac{\pi(N)}{N}$$

Fact: $\pi(N) \sim \dfrac{N}{\ln(N)}$

So

$$\Pr[p \text{ is a prime}] \sim \frac{1}{\ln(N)}$$

If $N = 2^{1024}$ this is about $0.001488 \approx 1/1000$.

So the number of iterations taken by our algorithm to find a prime is not too big.



## Finding generators

If $|G|$ is prime then every $g \in G \setminus \{1\}$ is a generator.

If $G = Z_p^*$ where $p$ is a prime

- It may be hard in general to find a generator
- But easy if the prime factorization of $p - 1$ is known



## Finding generators: Randomly pick and check

repeat
  $g \xleftarrow{\$} G \setminus \{1\}$
until $(\text{TEST-GEN}_G(g) = \text{true})$

- How do we design $\text{TEST-GEN}_G$?
- How many iterations does the algorithm take?

## Finding generators: Randomly pick and check

repeat
  $g \xleftarrow{\$} G \setminus \{1\}$
until $(\text{TEST-GEN}_G(g) = \text{true})$

- How do we design $\text{TEST-GEN}_G$?
- How many iterations does the algorithm take?

We say that $p$ is a SG prime if $p - 1 = 2q$ for some prime $q$.

Example: 7 is a SG prime because 7-1 = 2(3) and 3 is a prime.

We will address the above question for SG primes.



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | | | | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | | | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | | | | | | |

## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | | | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | | | | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | | | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | | | | |

## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | 5 | |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | 5 | 1 |

## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | 5 | 1 |
| $4^i$ | 4 | 2 | 1 | 4 | 2 | 1 |
| $5^i$ | 5 | 4 | 6 | 2 | 3 | 1 |
| $6^i$ | 6 | 1 | 6 | 1 | 6 | 1 |



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | 5 | 1 |
| $4^i$ | 4 | 2 | 1 | 4 | 2 | 1 |
| $5^i$ | 5 | 4 | 6 | 2 | 3 | 1 |
| $6^i$ | 6 | 1 | 6 | 1 | 6 | 1 |

The generators are



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $1^i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $2^i$ | 2 | 4 | 1 | 2 | 4 | 1 |
| $3^i$ | 3 | 2 | 6 | 4 | 5 | 1 |
| $4^i$ | 4 | 2 | 1 | 4 | 2 | 1 |
| $5^i$ | 5 | 4 | 6 | 2 | 3 | 1 |
| $6^i$ | 6 | 1 | 6 | 1 | 6 | 1 |

The generators are 3 and 5



## Generators mod 7

Let $G = Z_7^* = \{1, 2, 3, 4, 5, 6\}$

| $i$ | 2 | 3 |
|---|---|---|
| $1^i$ | 1 | 1 |
| $2^i$ | 4 | 1 |
| $3^i$ | 2 | 6 |
| $4^i$ | 2 | 1 |
| $5^i$ | 4 | 6 |
| $6^i$ | 1 | 6 |

We observe that $g$ is a generator if and only if $g^2 \neq 1$ and $g^3 \neq 1$.

## Testing whether a group element is a generator

Fact: Suppose $p$ is a SG prime, meaning $p-1=2q$ for a prime $q$. Then $g \in Z_p^*$ is a generator if and only if $g^2 \bmod p \neq 1$ and $g^q \bmod p \neq 1$.

Example: Let $p=7$ so that $q=3$. Then $g \in Z_7^*$ is a generator if and only if $g^2 \not\equiv 1$ and $g^3 \not\equiv 1$ modulo 7.

More generally, if $p$ is an arbitrary prime, let

$$p-1=\prod_{i=1}^{n} p_i^{\alpha_i}$$

be the prime factorization of the order the group $Z_p^*$. For $1 \leq j \leq n$ let $s_j=(p-1)/p_j$. Then $g \in Z_p^*$ is a generator if and only if $g^{s_j} \bmod p \neq 1$ for all $j \in \{1,\ldots,n\}$.



## How many generators are there?

Fact: Suppose $p$ is a SG prime, meaning $p-1=2q$ for a prime $q$. Then $Z_p^*$ has $q-1$ generators.

Example: Suppose $p=7$ so that $q=3$. Then $Z_7^*$ has $q-1=2$ generators.

More generally, if $p$ is an arbitrary prime, then the number of generators of $Z_p^*$ is $\varphi(p-1)$.



## Finding generators modulo SG primes

Suppose $p$ is a SG prime with $p-1=2q$.

repeat
  $g \xleftarrow{\$} G-\{1\}$
until ($g^2 \bmod p \neq 1$ and $g^q \bmod p \neq 1$)

The probability that a generator is found in a given step is

$$\frac{q-1}{2q-1} \approx \frac{1}{2}$$

so the expected number of iterations of the algorithm is about 2.



## Recall DH Secret Key Exchange

The following are assumed to be public: A large prime $p$ and a generator $g$ of $Z_p^*$.

| **Alice** | | **Bob** |
|---|---|---|
| $x \xleftarrow{\$} Z_{p-1}$; $X \leftarrow g^x \bmod p$ | | |
| | $\xrightarrow{X}$ | |
| | | $y \xleftarrow{\$} Z_{p-1}$; $Y \leftarrow g^y \bmod p$ |
| | $\xleftarrow{Y}$ | |
| $K_A \leftarrow Y^x \bmod p$ | | $K_B \leftarrow X^y \bmod p$ |

- $Y^x=(g^y)^x=g^{xy}=(g^x)^y=X^y$ modulo $p$, so $K_A=K_B$
- Adversary is faced with the CDH problem.

## DH Secret Key Exchange: Questions

- How do we pick a large prime $p$, and how large is large enough?
- What does it mean for $g$ to be a generator modulo $p$?
- How do we find a generator modulo $p$?
- How can Alice quickly compute $x \mapsto g^x \bmod p$?
- How can Bob quickly compute $y \mapsto g^y \bmod p$?
- Why is it hard to compute $(g^x \bmod p, g^y \bmod p) \mapsto g^{xy} \bmod p$?
- ...

**Exercise:** Answer as many of these questions as you can based on the content of this chapter.



## RSA Math

Recall that $\varphi(N) = |\mathsf{Z}_N^*|$.

Claim: Suppose $e, d \in \mathsf{Z}_{\varphi(N)}^*$ satisfy $ed \bmod \varphi(N) = 1$. Then for any $x \in \mathsf{Z}_N^*$ we have

$$(x^e)^d \bmod N = x \ .$$

Proof:

$$\begin{aligned}(x^e)^d \bmod N &= x^{ed \bmod \varphi(N)} \bmod N \\ &= x^1 \bmod N = x\end{aligned}$$



## The RSA function

A modulus $N$ and encryption exponent $e \in \mathsf{Z}_{\varphi(N)}^*$ define the RSA function $f : \mathsf{Z}_N^* \to \mathsf{Z}_N^*$ via:

$$f(x) = x^e \mod N$$

for all $x \in \mathsf{Z}_N^*$.

A value $d \in \mathsf{Z}_{\varphi(N)}^*$ satisfying $ed \bmod \varphi(N) = 1$ is called a decryption exponent.

Claim: The RSA function $f : \mathsf{Z}_N^* \to \mathsf{Z}_N^*$ is a permutation with inverse $f^{-1} : \mathsf{Z}_N^* \to \mathsf{Z}_N^*$ given by

$$f^{-1}(y) = y^d \mod N$$

Proof: For all $x \in \mathsf{Z}_N^*$, the prior claim says that we have

$$f^{-1}(f(x)) = (x^e)^d \bmod N = x \ .$$



## Example

Let $N = 15$. So

$$\begin{aligned}\mathsf{Z}_N^* &= \{1, 2, 4, 7, 8, 11, 13, 14\} \\ \varphi(N) &= 8 \\ \mathsf{Z}_{\varphi(N)}^* &= \{1, 3, 5, 7\}\end{aligned}$$

## Example

Let $N = 15$. So

$$\begin{aligned} Z_N^* &= \{1, 2, 4, 7, 8, 11, 13, 14\} \\ \varphi(N) &= 8 \\ Z_{\varphi(N)}^* &= \{1, 3, 5, 7\} \end{aligned}$$

Let $e = 3$ and $d = 3$. Then

$$ed \equiv 9 \equiv 1 \pmod 8$$

Let

$$\begin{aligned} f(x) &= x^3 \bmod 15 \\ g(y) &= y^3 \bmod 15 \end{aligned}$$

| $x$ | $f(x)$ | $g(f(x))$ |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 8 | 2 |
| 4 | 4 | 4 |
| 7 | 13 | 7 |
| 8 | 2 | 8 |
| 11 | 11 | 11 |
| 13 | 7 | 13 |
| 14 | 14 | 14 |



## Exercise

**1.** List all possible encryption exponents for RSA modulus 35:

**2.** The decryption exponent corresponding to RSA modulus 187 and encryption exponent 107 is



## Trapdoor permutation

RSA is a trapdoor, one-way permutation:

- Easy to invert given trapdoor $d$
- Hard to invert given only $N, e$

The second is true, to best of our current knowledge, for appropriately-chosen parameters $N, e, d$.

The choice of parameters is done by an algorithm called an RSA generator.



## RSA generators

An RSA generator with security parameter $k$ is an algorithm $\mathcal{K}_{rsa}$ that returns $N, p, q, e, d$ satisfying

- $p, q$ are distinct odd primes
- $N = pq$, and is called the (RSA) modulus
- $|N| = k$, meaning $2^{k-1} \le N \le 2^k$
- $e \in Z_{\varphi(N)}^*$ is called the encryption exponent
- $d \in Z_{\varphi(N)}^*$ is called the decryption exponent
- $ed \bmod \varphi(N) = 1$

## A formula for Phi

Fact: Suppose $N = pq$ for distinct primes $p$ and $q$. Then

$$\varphi(N) = (p-1)(q-1) \ .$$

**Example:** Let $N = 15 = 3 \cdot 5$. Then the Fact says that

$$\varphi(15) = (3-1)(5-1) = 8$$

. As a check, $Z_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ indeed has size 8.



## A more general formula for Phi

Fact: Suppose $N \geq 1$ factors as

$$N = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_n^{\alpha_n}$$

where $p_1 < p_2 < \ldots < p_n$ are primes and $\alpha_1, \ldots, \alpha_n \geq 1$ are integers. Then

$$\varphi(N) = p_1^{\alpha_1 - 1}(p_1 - 1) \cdot p_2^{\alpha_2 - 1}(p_2 - 1) \cdot \ldots \cdot p_n^{\alpha_n - 1}(p_n - 1) \ .$$

Note prior Fact is a special case of the above.

**Example:** Let $N = 45 = 3^2 \cdot 5^1$. Then the Fact says that

$$\varphi(45) = 3^1(3-1) \cdot 5^0(5-1) = 24$$



## Recall

Given $\varphi(N)$ and $e \in Z_{\varphi(N)}^*$, we can compute $d \in Z_{\varphi(N)}^*$ satisfying $ed \bmod \varphi(N) = 1$ via

$$d \leftarrow \text{MOD-INV}(e, \varphi(N)).$$

We have algorithms to efficiently test whether a number is prime, and we know that a random number has a pretty good chance of being a prime.

We use these facts to build RSA generators.



## Building RSA generators

Say we wish to have $e = 3$. (We will see that the smaller is $e$, the more efficient is encryption.) The generator $\mathcal{K}_{rsa}^3$ with (even) security parameter $k$ is as follows:

repeat
  $p, q \xleftarrow{\$} \{2^{k/2-1}, \ldots, 2^{k/2} - 1\}$; $N \leftarrow pq$; $M \leftarrow (p-1)(q-1)$
until
  $N \geq 2^{k-1}$ and $p, q$ are prime and $gcd(e, M) = 1$
$d \leftarrow \text{MOD-INV}(e, M)$
return $N, p, q, e, d$

## One-wayness of RSA

The following should be hard:

Given: $N, e, y$ where $y = f(x) = x^e \mod N$

Find: $x$

Formalism picks $x$ at random and generates $N, e$ via an RSA generator.



## One-wayness of RSA, formally

Let $\mathcal{K}_{\text{rsa}}$ be a RSA generator and $I$ an adversary.

Game $\text{OW}_{\mathcal{K}_{\text{rsa}}}$

**procedure** Initialize
$(N, p, q, e, d) \xleftarrow{\$} \mathcal{K}_{\text{rsa}}$
$x \xleftarrow{\$} Z_N^*$; $y \leftarrow x^e \mod N$
return $N, e, y$

**procedure** Finalize($x'$)
return $(x = x')$

The ow-advantage of $I$ is

$$\text{Adv}^{\text{ow}}_{\mathcal{K}_{\text{rsa}}}(I) = \Pr\left[\text{OW}^I_{\mathcal{K}_{\text{rsa}}} \Rightarrow \text{true}\right]$$



## Inverting RSA

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \mod N = y$



## Inverting RSA

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \mod N = y$

EASY

Know $d$

because $x = y^d \mod N$

## Inverting RSA

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \bmod N = y$

$\uparrow$ EASY because $x = y^d \bmod N$

Know $d$

$\uparrow$ EASY because $d = \text{MOD-INV}(e, \varphi(N))$

Know $\varphi(N)$



## Inverting RSA

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \bmod N = y$

$\uparrow$ EASY because $x = y^d \bmod N$

Know $d$

$\uparrow$ EASY because $d = \text{MOD-INV}(e, \varphi(N))$

Know $\varphi(N)$

$\uparrow$ EASY because $\varphi(N) = (p-1)(q-1)$

Know $p, q$



## Inverting RSA

Inverting RSA : given $N, e, y$ find $x$ such that $x^e \bmod N = y$

$\uparrow$ EASY because $x = y^d \bmod N$

Know $d$

$\uparrow$ EASY because $d = \text{MOD-INV}(e, \varphi(N))$

Know $\varphi(N)$

$\uparrow$ EASY because $\varphi(N) = (p-1)(q-1)$

Know $p, q$

$\uparrow$ ?

Know $N$



## Factoring Problem

Given: $N$ where $N = pq$ and $p, q$ are prime

Find: $p, q$

If we can factor we can invert RSA. We do not know whether the converse is true, meaning whether or not one can invert RSA without factoring.

## A factoring algorithm

**Alg** FACTOR($N$) // $N = pq$ where $p, q$ are primes

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then
    $p \leftarrow i$ ; $q \leftarrow N/i$ ; return $p, q$

This algorithm works but takes time

$$\mathcal{O}(\sqrt{N}) = \mathcal{O}(e^{0.5 \ln N})$$

which is prohibitive.



## Factoring algorithms

| Algorithm | Time taken to factor $N$ |
|---|---|
| Naive | $O(e^{0.5 \ln N})$ |
| Quadratic Sieve (QS) | $O(e^{c(\ln N)^{1/2}(\ln \ln N)^{1/2}})$ |
| Number Field Sieve (NFS) | $O(e^{1.92(\ln N)^{1/3}(\ln \ln N)^{2/3}})$ |



## Factoring records

| bit-length of number | When factored | Algorithm used |
|---|---|---|
| 400 | 1993 | QS |
| 428 | 1994 | QS |
| 431 | 1996 | NFS |
| 465 | 1999 | NFS |
| 515 | 1999 | NFS |
| 576 | 2003 | NFS |
| 768 | 2009 | NFS |
| 795 | 2019 | NFS |
| 829 | 2020 | NFS |



## Modulii sizes

We estimate that a 1024-bit RSA modulus provides 80-bits of security, meaning factoring it takes $2^{80}$ time.

Factorization of a 1024-bit modulus seems out of reach at present, but longer modulii, like 2048 bits, are in use and recommended.

## Choices of encryption exponent

Common choices are $e = 3$, $e = 17$ and $e = 65,537$. Why these?

| $e$ | bin($e$) |
|---|---|
| 3 | 11 |
| 17 | 10001 |
| 65,537 | 10000000000000001 |

Recall that the modular exponentiation algorithm computing $x \mapsto x^e \bmod N$ uses $c(b)$ modular multiplications per bit $b \in \{0,1\}$ in the binary expansion bin($e$), where $c(0) = 1$ and $c(1) = 2$. So the fewer the number of 1s in bin($e$), the faster is the operation.



## Low-exponent and other attacks

Further attacks on RSA include

- Coppersmith's attack
- Franklin-Reiter attack
- Håstad attack

These work for small encryption exponents but do not violate OW-security.

If RSA-based public-key encryption and digital signature schemes use RSA appropriately, these attacks do not threaten them, even if the encryption exponent is small.

Accordingly, in designing RSA-based public-key encryption and digital signature schemes, we seek proofs of security based (only) on the OW-security of RSA.



## RSA: what to remember

The RSA function $f(x) = x^e \bmod N$ is a trapdoor one way permutation:

- Easy forward: given $N, e, x$ it is easy to compute $f(x)$
- Easy back with trapdoor: Given $N, d$ and $y = f(x)$ it is easy to compute $x = f^{-1}(y) = y^d \bmod N$
- Hard back without trapdoor: Given $N, e$ and $y = f(x)$ it is hard to compute $x = f^{-1}(y)$



## Bilinear maps

Let $G, G_T$ be cyclic groups, both of order $m$. A function (map) $\mathbf{e}$: $G \times G \to G_T$ is bilinear if, for any generator $g$ of $G$, the following hold:

- Bilinearity: $\mathbf{e}(g^x, g^y) = \mathbf{e}(g, g)^{xy}$ for all $x, y \in \mathrm{Z}_m$
- Non degeneracy: $\mathbf{e}(g, g)$ is a generator of $G_T$.

Most often, $m$ is a prime number.

Bi-linearity implies that for all $A, B \in G$ and all $x, y \in \mathrm{Z}_m$ we have

$$\mathbf{e}(A^x, B^y) = \mathbf{e}(A^y, B^x) \ .$$

Hard problems include DL or CDH in either $G$ or $G_T$.

DDH (the Decision Diffie-Hellman) problem is, given $X = g^x$, $Y = g^y$ and $Z$, check if $Z = g^{xy}$. This problem is easy to solve for $G$ because we can check that $\mathbf{e}(X, Y) = \mathbf{e}(g, Z)$.

A new hard problem is BDH (Bilinear Diffie-Hellman).

## BDH Formally

Let $\mathbf{e}$: $G \times G \to G_T$ be a bilinear map, and $g$ a generator of $G$.

Game $\mathrm{BDH}_{\mathbf{e},g}$

**procedure** Initialize
$x, y, z \xleftarrow{\$} Z_m$
$X \leftarrow g^x$ ; $Y \leftarrow g^y$ ; $Z \leftarrow g^z$
return $X, Y, Z$

**procedure** Finalize($W$)
return $(W = \mathbf{e}(g, g)^{xyz})$

The bdh-advantage of $A$ is

$$\mathrm{Adv}^{\mathrm{bdh}}_{\mathbf{e},g}(A) = \Pr\left[\mathrm{BDH}^{A}_{\mathbf{e},g} \Rightarrow \mathrm{true}\right]$$



## The quantum threat

On a quantum computer, Shor's algorithm can compute discrete logarithms, and factor, in polynomial time.

Efforts to build quantum computers are underway.

Public-key cryptography can be based on computational problems, involving lattices and codes, for which there are currently no known polynomial time quantum algorithms.