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

\[
\begin{align*}
    & \text{Alice} & \text{Bob} \\
    & x \leftarrow \ldots; X \leftarrow \ldots & y \leftarrow \ldots; Y \leftarrow \ldots \\
    & X \rightarrow & \\
    & \leftarrow Y \\
    & K_A \leftarrow F_A(x, Y) & K_B \leftarrow F_B(y, X)
\end{align*}
\]

Desired properties of the protocol:

- \( K_A = K_B \), meaning Alice and Bob agree on a key
- Adversary given \( X, Y \) can’t compute \( K_A \)
Can you build a secret key exchange protocol?
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.
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
The following are assumed to be public: A large prime $p$ and a number $g$ called a generator mod $p$. Let $Z_{p-1} = \{0, 1, \ldots, p - 2\}$.

Alice

\[
x \leftarrow Z_{p-1}; \quad X \leftarrow g^x \mod p
\]

Bob

\[
y \leftarrow Z_{p-1}; \quad Y \leftarrow g^y \mod p
\]

\[
K_A \leftarrow Y^x \mod p
\]

\[
K_B \leftarrow X^y \mod p
\]

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo $p$, so $K_A = K_B$
- Adversary is faced with computing $g^{xy} \mod p$ given $g^x \mod p$ and $g^y \mod p$, 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 $x \mapsto g^x \mod p$?
- How can Bob quickly compute $y \mapsto g^y \mod p$?
- Why is it hard to compute $(g^x \mod p, g^y \mod p) \mapsto g^{xy} \mod p$?
- ...

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

\[ Z = \{ \ldots, -2, -1, 0, 1, 2, \ldots \} \]

\[ N = \{ 0, 1, 2, \ldots \} \]

\[ Z_+ = \{ 1, 2, 3, \ldots \} \]

For \( a, N \in Z \) let \( \gcd(a, N) \) be the largest \( d \in Z_+ \) such that \( d \) divides both \( a \) and \( N \).

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

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

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

Example: $N = 12$

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

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

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

Example: $N = 12$

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

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

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

Example: $N = 12$

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

\[
a \mod N = r \in \mathbb{Z}_N
\]

is the remainder when \( a \) is divided by \( N \).

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

**Def:** \( a \equiv b \pmod{N} \) if \( a \mod N = b \mod N \).

**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

$$a \cdot b = ab \text{ mod } 12$$

**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 $a \cdot b = ab \text{ mod } N$ is a group.
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 $a \cdot b = ab \mod N$ satisfies closure, meaning

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

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

$$5 \cdot 7 \mod 12 = 35 \mod 12 = 11 \in \mathbb{Z}_{12}^*$$
**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 $a \cdot b = ab \mod N$ satisfies associativity, meaning

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

**Example:**

$$(5 \cdot 7 \mod 12) \cdot 11 \mod 12 = (35 \mod 12) \cdot 11 \mod 12$$

$$= 11 \cdot 11 \mod 12 = 1$$

$$5 \cdot (7 \cdot 11 \mod 12) \mod 12 = 5 \cdot (77 \mod 12) \mod 12$$

$$= 5 \cdot 5 \mod 12 = 1$$
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 $a \cdot b = ab \mod N$ then 1 is the identity element because $a \cdot 1 \mod N = 1 \cdot a \mod N = a$ 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 $a \cdot b = ab \mod N$ then
\[
\forall a \in \mathbb{Z}_N^* \quad \exists b \in \mathbb{Z}_N^* \text{ such that } a \cdot b \mod N = 1.
\]

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

**Example:** $5^{-1} \mod 12$ is the $b \in \mathbb{Z}_{12}^*$ satisfying $5b \mod 12 = 1$, so $b =$
**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 $a \cdot b = ab \mod N$ then

$$\forall a \in \mathbb{Z}_N^* \exists b \in \mathbb{Z}_N^* \text{ such that } a \cdot b \mod N = 1.$$ 

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

**Example:** $5^{-1} \mod 12$ is the $b \in \mathbb{Z}_{12}^*$ satisfying $5b \mod 12 = 1$, so $b = 5$
Fact: If $N \geq 1$ is an integer then $\mathbb{Z}_N$ is a group under the operation of addition modulo $N$, namely $a \cdot b = (a + b) \mod N$.

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

**Fact:** Let \( a, b, c \in \mathbb{Z} \) and \( N \in \mathbb{Z}^+ \). Then

\[
abc \mod N = ((ab \mod N) c) \mod N
\]

**Example:** What is \( 5 \cdot 8 \cdot 10 \cdot 16 \mod 21 \)?

**Slow way:**
- \( 5 \cdot 8 \cdot 10 \cdot 16 = 40 \cdot 10 \cdot 16 = 400 \cdot 16 = 6400 \)
- \( 6400 \mod 21 = 16 \)

**Faster way, using above Fact:**
- \( 5 \cdot 8 \mod 21 = 40 \mod 21 = 19 \)
- \( 19 \cdot 10 \mod 21 = 190 \mod 21 = 1 \)
- \( 1 \cdot 16 \mod 21 = 16 \)
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

$$a^n = a \cdot a \cdots a.$$ 

Also we let

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

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”.
Examples

Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 =$$
Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 = 5 \cdot 5 \cdot 5$$
Examples

Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 = 5 \cdot 5 \cdot 5 \equiv 25 \cdot 5 \equiv 11 \cdot 5 \equiv 55 \equiv 13$$

and

$$5^{-3} =$$
Examples

Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 = 5 \cdot 5 \cdot 5 \equiv 25 \cdot 5 \equiv 11 \cdot 5 \equiv 55 \equiv 13$$

and

$$5^{-3} = 5^{-1} \cdot 5^{-1} \cdot 5^{-1}$$
Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 = 5 \cdot 5 \cdot 5 \equiv 25 \cdot 5 \equiv 11 \cdot 5 \equiv 55 \equiv 13$$

and

$$5^{-3} = 5^{-1} \cdot 5^{-1} \cdot 5^{-1} \equiv 3 \cdot 3 \cdot 3$$
Examples

Let $N = 14$ and $G = \mathbb{Z}_N^*$. Then modulo $N$ we have

$$5^3 = 5 \cdot 5 \cdot 5 \equiv 25 \cdot 5 \equiv 11 \cdot 5 \equiv 55 \equiv 13$$

and

$$5^{-3} = 5^{-1} \cdot 5^{-1} \cdot 5^{-1} \equiv 3 \cdot 3 \cdot 3 \equiv 27 \equiv 13$$
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 $\mathbb{Z}^*_{21}$ is 12 because

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

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

Examples: Modulo 21 we have

- $5^{12} \equiv (5^3)^4 \equiv 20^4 \equiv (-1)^4 \equiv 1$
- $8^{12} \equiv (8^2)^6 \equiv (1)^6 \equiv 1$
**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 $i \in \mathbb{Z}$,

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

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

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

But $a^m = 1$ by Fact.
Corollary: Let $G$ be a group of order $m$ and $a \in G$. Then for any $i \in \mathbb{Z}$,

$$a^i = a^i \mod m.$$ 

Example: What is $5^{74} \mod 21$?
Corollary: Let $G$ be a group of order $m$ and $a \in G$. Then for any $i \in \mathbb{Z}$,

$$a^i = a^i \mod m.$$ 

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

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

$$5^{74} \mod 21 = 5^{74 \mod 12} \mod 21$$

$$= 5^2 \mod 21$$

$$= 4.$$
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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
<th>Output</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>$a, b$</td>
<td>$a + b$</td>
<td>$O(</td>
</tr>
<tr>
<td>MULT</td>
<td>$a, b$</td>
<td>$ab$</td>
<td>$O(</td>
</tr>
<tr>
<td>INT-DIV</td>
<td>$a, N$</td>
<td>$q, r$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD</td>
<td>$a, N$</td>
<td>$a \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>EXT-GCD</td>
<td>$a, N$</td>
<td>$(d, a', N')$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD-INV</td>
<td>$a \in \mathbb{Z}_N^*$, $N$</td>
<td>$a^{-1} \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD-EXP</td>
<td>$a \in \mathbb{Z}_N$, $n$, $N$</td>
<td>$a^n \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>EXP$_G$</td>
<td>$a \in G$, $n$</td>
<td>$a^n \in G$</td>
<td>$O(</td>
</tr>
</tbody>
</table>

Nadia Heninger  
UCSD  
35
Extended gcd

\[ \text{EXT-GCD}(a, N) \text{ returns } (d, a', N') \text{ such that } \]
\[ d = \gcd(a, N) = a \cdot a' + N \cdot N'. \]

**Example:** \( \text{EXT-GCD}(12, 20) = \)
EXT-GCD($a, N$) returns ($d, a', N'$) such that

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

**Example:** EXT-GCD(12, 20) = (4, -3, 2) because

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

\[ \text{EXT-GCD}(a, N) \mapsto (d, a', N') \text{ such that} \]
\[ d = \gcd(a, N) = a \cdot a' + N \cdot N' \]

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

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

Running time is \(O(|a| \cdot |N|)\), so the extended gcd can be computed in \textbf{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 $a^{-1} \mod N$, meaning the unique $a' \in \mathbb{Z}_N^*$ 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 $a' \mod N$

Modular inverse can be computed in \textit{quadratic} time.
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

$$a^n = a \cdot a \cdots a$$

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

$$a^n = a \cdot a \cdots a$$

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 $O(n) = O(2^{|n|})$ so it is exponential time. For $n \approx 2^{512}$ it is prohibitively expensive.
Fast exponentiation idea

We can compute

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

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?
Suppose the binary length of $n$ is 5, meaning the binary representation of $n$ has the form $b_4b_3b_2b_1b_0$. Then

$$n = 2^4b_4 + 2^3b_3 + 2^2b_2 + 2^1b_1 + 2^0b_0$$
$$= 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0.$$ 

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:

$$y_5 = 1$$
$$y_4 = y_5^2 \cdot a^{b_4} = a^{b_4}$$
$$y_3 = y_4^2 \cdot a^{b_3} = a^{2b_4+b_3}$$
$$y_2 = y_3^2 \cdot a^{b_2} = a^{4b_4+2b_3+b_2}$$
$$y_1 = y_2^2 \cdot a^{b_1} = a^{8b_4+4b_3+2b_2+b_1}$$
$$y_0 = y_1^2 \cdot a^{b_0} = a^{16b_4+8b_3+4b_2+2b_1+b_0}.$$
Let \( \text{bin}(n) = b_{k-1} \ldots b_0 \) be the binary representation of \( n \), meaning

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

**Alg** \( \text{EXP}_G(a, n) \)  // \( a \in G, n \geq 1 \)

\[
b_{k-1} \ldots b_0 \leftarrow \text{bin}(n)
\]

\[
y \leftarrow 1
\]

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.

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Input</th>
<th>Output</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>$a, b$</td>
<td>$a + b$</td>
<td>$O(</td>
</tr>
<tr>
<td>MULT</td>
<td>$a, b$</td>
<td>$ab$</td>
<td>$O(</td>
</tr>
<tr>
<td>INT-DIV</td>
<td>$a, N$</td>
<td>$q, r$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD</td>
<td>$a, N$</td>
<td>$a \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>EXT-GCD</td>
<td>$a, N$</td>
<td>$(d, a', N')$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD-INV</td>
<td>$a \in \mathbb{Z}_N^*$, $N$</td>
<td>$a^{-1} \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>MOD-EXP</td>
<td>$a \in \mathbb{Z}_N$, $n$, $N$</td>
<td>$a^n \mod N$</td>
<td>$O(</td>
</tr>
<tr>
<td>EXP$_G$</td>
<td>$a \in G$, $n$</td>
<td>$a^n \in G$</td>
<td>$O(</td>
</tr>
</tbody>
</table>
Generators and cyclic groups

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

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

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^i \mod 11$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>$5^i \mod 11$</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>1</td>
</tr>
</tbody>
</table>

so

\[
\langle 2 \rangle = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}
\]

\[
\langle 5 \rangle = \{1, 3, 4, 5, 9\}
\]

- 2 a generator because $\langle 2 \rangle = \mathbb{Z}_{11}^*$.
- 5 is not a generator because $\langle 5 \rangle \neq \mathbb{Z}_{11}^*$.
- $\mathbb{Z}_{11}^*$ is cyclic because it has a generator.
If $G = \langle g \rangle$ is a cyclic group of order $m$ then for every $a \in G$ there is a unique exponent $i \in \mathbb{Z}_m$ such that $g^i = a$. We call $i$ the discrete logarithm of $a$ to base $g$ and denote it by

$$D\text{Log}_{G,g}(a)$$

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

$$D\text{Log}_{G,g}(g^i) = i \quad \text{for all } i \in \mathbb{Z}_m$$

$$g^{D\text{Log}_{G,g}(a)} = a \quad \text{for all } a \in G.$$
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 $\text{DLog}_G,2(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \mod 11 = a$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^i \mod 11$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$a$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{DLog}_G,2(a)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 $\text{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \mod 11 = a$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^i \mod 11$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$a$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{DLog}_{G,2}(a)$</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
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.
Let $G = \langle g \rangle$ be a cyclic group of order $m$ with generator $g \in G$.

**Input:** $X \in G$

**Desired Output:** $\text{DLog}_{G,g}(X)$

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


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

**Input:** $X \in G$

**Desired Output:** $D\text{Log}_{G,g}(X)$

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:** $\text{DLog}_{G,g}(X)$

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 $G = \mathbb{Z}_p^*$ is $O(p)$, which is exponential time and prohibitive for large $p$. 
Here $p$ is a prime and $\text{EC}_p$ represents an elliptic curve group of order $p$.

In the first case, if the largest factor of $p - 1$ is $q$, there is also a $O(\sqrt{q})$ algorithm to solve discrete log.

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 $\mathbb{Z}_p^*$:

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

For elliptic curves, current record seems to be for $|p|$ around 114.
Elliptic curve groups are commonly used for public-key cryptography now.

The mathematical details are a bit complex.

For now, think of an elliptic curve group as a cyclic group.

This means it has a generator, a group operation (typically written as +), an order, and one can define the analogue of discrete logarithm in this group.

The structure of elliptic curve groups does not seem to permit the same types of subexponential-time discrete logarithm algorithms as $\mathbb{Z}_p^*$. 
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 $\mathbb{Z}_p^*$ ($p$ a prime) we need to set $|\mathbb{Z}_p^*| = p - 1 \approx 2^{1024}$
- 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

$$e^{1.92 \left( \frac{\ln 2^{1024}}{3} \right) \left( \frac{\ln \ln 2^{1024}}{3} \right)} \approx \sqrt{2^{160}} = 2^{80}$$

But now:

<table>
<thead>
<tr>
<th>Group Size</th>
<th>Cost of Exponentiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{160}$</td>
<td>$T \approx 160^3$</td>
</tr>
<tr>
<td>$2^{1024}$</td>
<td>$1024^3 \approx 260T$</td>
</tr>
</tbody>
</table>

Exponentiation will be 260 times faster in the smaller group.
Let $G = \langle g \rangle$ be a cyclic group of order $m$, and $A$ an adversary.

**Game $DL_{G,g}$**

**procedure** Initialize

$x \leftarrow \mathbb{Z}_m; X \leftarrow g^x$

return $X$

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

return $(x = x')$

The dl-advantage of $A$ is

$$Adv_{G,g}^{dl}(A) = \Pr \left[ DL_{G,g}^A \Rightarrow true \right]$$
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:**

$x \leftarrow \text{DLog}_{G,g}(X); \text{ Return } Y^x.$

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

The **cdh-advantage** of \( A \) is

\[
\text{Adv}_{G,g}^{\text{cdh}}(A) = \Pr \left[ \text{CDH}_{G,g}^A \Rightarrow \text{true} \right]
\]
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 $\mathbb{Z}_p^*$:

- Pick numbers $p$ at random until $p$ is a prime of the desired form
- Pick elements $g$ from $\mathbb{Z}_p^*$ 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 $\mathbb{Z}_p^*$ when $p$ is prime
- How many elements of $\mathbb{Z}_p^*$ 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** \text{Findprime}(k)
\begin{algorithm}
\textbf{do}
\textbf{do} \quad p \leftarrow \{2^{k-1}, \ldots, 2^k - 1\}
\textbf{until} \quad (p \text{ is prime and } (p - 1)/2 \text{ is prime})
\textbf{return} \quad p
\end{algorithm}

- 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 $N \mod i = 0$ then return false
return true
Given: integer \( N \)
Output: TRUE if \( N \) is prime, FALSE otherwise.

for \( i = 2, \ldots, \lceil \sqrt{N} \rceil \) do
    if \( N \mod i = 0 \) 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
Let $\pi(N)$ be the number of primes in the range $1, \ldots, N$. So if $p \leftarrow \{1, \ldots, N\}$ then

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

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

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.
Recall DH Secret Key Exchange

The following are assumed to be public: A large prime $p$ and a generator $g$ of $\mathbb{Z}_p^*$.

<table>
<thead>
<tr>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \leftarrow Z_{p-1}; X \leftarrow g^x \mod p$</td>
<td>$y \leftarrow Z_{p-1}; Y \leftarrow g^y \mod p$</td>
</tr>
<tr>
<td>$K_A \leftarrow Y^x \mod p$</td>
<td>$K_B \leftarrow X^y \mod p$</td>
</tr>
</tbody>
</table>

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y \mod p$, 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 $x \mapsto g^x \mod p$?
- How can Bob quickly compute $y \mapsto g^y \mod p$?
- Why is it hard to compute $(g^x \mod p, g^y \mod p) \mapsto g^{xy} \mod p$?
- ... 

The slides have sketched the answers to many of these questions.
Recall that \( \varphi(N) = |Z_N^*| \).

**Claim:** Suppose \( e, d \in Z_{\varphi(N)}^* \) satisfy \( ed \mod \varphi(N) = 1 \). Then for any \( x \in Z_N^* \) we have

\[
(x^e)^d \mod N = x.
\]

**Proof:**

\[
(x^e)^d \mod N = x^{ed \mod \varphi(N)} \mod N = x^1 \mod N = x
\]
The RSA function

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

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

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

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

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

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

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

$$f^{-1}(f(x)) = (x^e)^d \mod N = x.$$
Let $N = 15$. So

$$Z_N^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$$

$$\varphi(N) = 8$$

$$Z_{\varphi(N)}^* = \{1, 3, 5, 7\}$$
Example

Let $N = 15$. So

\[
Z_N^* = \{1, 2, 4, 7, 8, 11, 13, 14\}
\]

\[
\varphi(N) = 8
\]

\[
Z_{\varphi(N)}^* = \{1, 3, 5, 7\}
\]

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

\[
ed \equiv 9 \equiv 1 \pmod{8}
\]

Let

\[
f(x) = x^3 \mod 15
\]

\[
g(y) = y^3 \mod 15
\]
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.
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 $2^{k-1} \leq N \leq 2^k$
- $e \in \mathbb{Z}_{\varphi(N)}^*$ is called the encryption exponent
- $d \in \mathbb{Z}_{\varphi(N)}^*$ is called the decryption exponent
- $ed \mod \varphi(N) = 1$
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, $\mathbb{Z}_{15}^*$ indeed has size 8.
A more general formula for Phi

**Fact:** Suppose \( N \geq 1 \) factors as

\[
N = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_n^{\alpha_n}
\]

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

\[
\varphi(N) = p_1^{\alpha_1 - 1}(p_1 - 1) \cdot p_2^{\alpha_2 - 1}(p_2 - 1) \cdots \cdot p_n^{\alpha_n - 1}(p_n - 1).
\]

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 $e \in \mathbb{Z}_{\varphi(N)}^*$, we can compute $d \in \mathbb{Z}_{\varphi(N)}^*$ satisfying $ed \mod \varphi(N) = 1$ 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.
Say we wish to have $e = 3$. (We will see that the smaller is $e$, the more efficient is encryption.) The generator $\mathcal{K}_{rsa}$ with (even) security parameter $k$ is as follows:

repeat  
  $p, q \leftarrow \{2^{k/2-1}, \ldots, 2^{k/2} - 1\}; \ N \leftarrow pq; \ M \leftarrow (p - 1)(q - 1)$  
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 $\mathcal{K}_{rsa}$ be a RSA generator and $I$ an adversary.

Game $OW_{\mathcal{K}_{rsa}}$

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

```
procedure Finalize(x')
return ( x = x' )
```

The ow-advantage of $I$ is

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

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

Easy because $x = y^d \mod N$

Know $d$
Inverting RSA

: given \( N, e, y \) find \( x \) such that \( x^e \mod N = y \)

\[ \text{EASY because } x = y^d \mod N \]

Know \( d \)

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

Know \( \varphi(N) \)
Inverting RSA

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

EASY  because $x = y^d \mod N$

Know $d$

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

Know $\varphi(N)$

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

Know $p, q$
Inverting RSA

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

- **Easy** because $x = y^d \mod N$
  - Know $d$

- **Easy** because $d = \text{MOD-INV}(e, \phi(N))$
  - Know $\phi(N)$

- **Easy** because $\phi(N) = (p - 1)(q - 1)$
  - Know $p, q$

- ?
  - 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

\textbf{Alg} \textsc{FACTOR}(N) \quad // \quad N = pq \text{ where } p, q \text{ are primes}

\text{for } i = 2, \ldots, \left\lceil \sqrt{N} \right\rceil \text{ do}
  \text{if } N \mod i = 0 \text{ then}
  \quad p \leftarrow i; \; q \leftarrow N/i; \; \text{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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time taken to factor $N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>$O(e^{0.5 \ln N})$</td>
</tr>
<tr>
<td>Quadratic Sieve (QS)</td>
<td>$O(e^{c(\ln N)^{1/2}(\ln \ln N)^{1/2}})$</td>
</tr>
<tr>
<td>Number Field Sieve (NFS)</td>
<td>$O(e^{1.92(\ln N)^{1/3}(\ln \ln N)^{2/3}})$</td>
</tr>
</tbody>
</table>
## Factoring records

<table>
<thead>
<tr>
<th>bit-length of number</th>
<th>When factored</th>
<th>Algorithm used</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>1993</td>
<td>QS</td>
</tr>
<tr>
<td>428</td>
<td>1994</td>
<td>QS</td>
</tr>
<tr>
<td>431</td>
<td>1996</td>
<td>NFS</td>
</tr>
<tr>
<td>465</td>
<td>1999</td>
<td>NFS</td>
</tr>
<tr>
<td>515</td>
<td>1999</td>
<td>NFS</td>
</tr>
<tr>
<td>576</td>
<td>2003</td>
<td>NFS</td>
</tr>
<tr>
<td>768</td>
<td>2009</td>
<td>NFS</td>
</tr>
<tr>
<td>795</td>
<td>2019</td>
<td>NFS</td>
</tr>
<tr>
<td>829</td>
<td>2020</td>
<td>NFS</td>
</tr>
</tbody>
</table>
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 hasn’t been done yet in public, but is within reach of large organizations. Longer moduli, like 2048 bits, have been recommended since around 2010.
Choices of encryption exponent

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

<table>
<thead>
<tr>
<th>$e$</th>
<th>bin($e$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>17</td>
<td>10001</td>
</tr>
<tr>
<td>65,537</td>
<td>1000000000000000000000001</td>
</tr>
</tbody>
</table>

Recall that the modular exponentiation algorithm computing $x \mapsto x^e \mod N$ 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.
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.
http://www.youtube.com/watch?v=wXB-V_Keiu8
The RSA function $f(x) = x^e \mod N$ 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 $x = f^{-1}(y) = y^d \mod N$
- Hard back without trapdoor: Given $N, e$ and $y = f(x)$ it is hard to compute $x = f^{-1}(y)$
On a quantum computer, Shor’s algorithm can compute discrete logarithms and factor in polynomial time.

Efforts to build quantum computers are underway.

Efforts are underway to standardize public-key cryptography based on computational problems like finding short vectors in lattices for which there are currently no known efficient quantum algorithms.