# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

Appears to be hard, similar to the Factoring Problem.
So we can use it in cryptosystems, as a one-way function.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

Appears to be hard, similar to the Factoring Problem.
So we can use it in cryptosystems, as a one-way function.

DLP($p$):
Given a large prime $p$, $\alpha, \beta \in \mathbb{Z}_p^*$.
Find $x \in \mathbb{Z}_{p-1}$ such that $\beta = \alpha^x \pmod{p}$.
$x$ is the *discrete logarithm* of $\beta$ w.r.t. $\alpha$.
$x = \log_\alpha \beta \pmod{p}$

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

Appears to be hard, similar to the Factoring Problem.
So we can use it in cryptosystems, as a one-way function.

DLP($p$):
Given a large prime $p$, $\alpha, \beta \in \mathbb{Z}_p^*$.
Find $x \in \mathbb{Z}_{p-1}$ such that $\beta = \alpha^x \pmod{p}$.
$x$ is the *discrete logarithm* of $\beta$ w.r.t. $\alpha$.
$x = \log_\alpha \beta \pmod{p}$

Example: In $\mathbb{Z}_7^*$, $6 = \log_3 1 \pmod{7}$, since $1 \equiv 3^6 \pmod{7}$

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

Appears to be hard, similar to the Factoring Problem.
So we can use it in cryptosystems, as a one-way function.

DLP($p$):
Given a large prime $p$, $\alpha, \beta \in \mathbb{Z}_p^*$.
Find $x \in \mathbb{Z}_{p-1}$ such that $\beta = \alpha^x \pmod{p}$.
$x$ is the *discrete logarithm* of $\beta$ w.r.t. $\alpha$.
$x = \log_\alpha \beta \pmod{p}$

Example: In $\mathbb{Z}_7^*$, $6 = \log_3 1 \pmod{7}$, since $1 \equiv 3^6 \pmod{7}$

There is no discrete log of 3 w.r.t 1 in $\mathbb{Z}_7^*$.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.
   Example: $p - 1$ has order 2 modulo $p$.
   It is easy to find the discrete log of $p - 1$ or 1 w.r.t. $p - 1$.
   Use brute force. How?

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.
   Example: $p - 1$ has order 2 modulo $p$.
   It is easy to find the discrete log of $p - 1$ or 1 w.r.t. $p - 1$.
   Use brute force. How?
2. $p$ must be large.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.
   Example: $p-1$ has order 2 modulo $p$.
   It is easy to find the discrete log of $p-1$ or 1 w.r.t. $p-1$.
   Use brute force. How?
2. $p$ must be large.
   Again, use brute force.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.
   Example: $p-1$ has order 2 modulo $p$.
   It is easy to find the discrete log of $p-1$ or 1 w.r.t. $p-1$.
   Use brute force. How?
2. $p$ must be large.
   Again, use brute force.
3. $p-1$ must have at least 1 large prime factor.

# Discrete Logarithm Problem in $\mathbb{Z}_p^*$

For DLP($p$) to be difficult:

1. The order of $\alpha$ must be large.
   Example: $p-1$ has order 2 modulo $p$.
   It is easy to find the discrete log of $p-1$ or 1 w.r.t. $p-1$.
   Use brute force. How?
2. $p$ must be large.
   Again, use brute force.
3. $p-1$ must have at least 1 large prime factor.
   Use Pohlig-Hellman's algorithm.

# Pohlig-Hellman's Algorithm

Pohlig-Hellman($\alpha, \beta, p$)

Factor $p - 1 = \prod_{i=1}^{k} p_i^{c_i}$
**for** $i = 1$ to $k$ **do**
    Compute $x_i = (\log_\alpha \beta \pmod{p})$ modulo $p_i^{c_i}$
**endfor**
Use Chinese Remainder Theorem to compute $x$ modulo $p - 1$
    s.t. $x \equiv x_i \pmod{p_i^{c_i}}$ for $1 \le i \le k$
**return** $x$

# Pohlig-Hellman's Algorithm

Pohlig-Hellman$(\alpha, \beta, p)$

Factor $p - 1 = \prod_{i=1}^{k} p_i^{c_i}$
**for** $i = 1$ to $k$ **do**
    Compute $x_i = (\log_\alpha \beta \pmod{p})$ modulo $p_i^{c_i}$
**endfor**
Use Chinese Remainder Theorem to compute $x$ modulo $p - 1$
    s.t. $x \equiv x_i \pmod{p_i^{c_i}}$ for $1 \leq i \leq k$
**return** $x$

To compute $x = \log_3 26 \pmod{29}$:
$p - 1 = 4 \cdot 7$

# Pohlig-Hellman's Algorithm

Pohlig-Hellman$(\alpha, \beta, p)$

Factor $p - 1 = \prod_{i=1}^{k} p_i^{c_i}$
**for** $i = 1$ to $k$ **do**
    Compute $x_i = (\log_\alpha \beta \pmod{p})$ modulo $p_i^{c_i}$
**endfor**
Use Chinese Remainder Theorem to compute $x$ modulo $p - 1$
    s.t. $x \equiv x_i \pmod{p_i^{c_i}}$ for $1 \le i \le k$
**return** $x$

To compute $x = \log_3 26 \pmod{29}$:
$p - 1 = 4 \cdot 7$
To compute $x \pmod{7}$: $\alpha' = 3^{28/7} = 23$, $\beta' = 26^{28/7} = 23$
Work in a subgroup of size 7. Get $x \equiv 1 \pmod{7}$.

# Pohlig-Hellman's Algorithm

Pohlig-Hellman$(\alpha, \beta, p)$

Factor $p - 1 = \prod_{i=1}^{k} p_i^{c_i}$
**for** $i$ = 1 to $k$ **do**
    Compute $x_i = (\log_\alpha \beta \pmod{p})$ modulo $p_i^{c_i}$
**endfor**
Use Chinese Remainder Theorem to compute $x$ modulo $p - 1$
    s.t. $x \equiv x_i \pmod{p_i^{c_i}}$ for $1 \leq i \leq k$
**return** $x$

To compute $x = \log_3 26 \pmod{29}$:
$p - 1 = 4 \cdot 7$
To compute $x \pmod{7}$: $\alpha' = 3^{28/7} = 23$, $\beta' = 26^{28/7} = 23$
Work in a subgroup of size 7. Get $x \equiv 1 \pmod{7}$.
More difficult when $c_i > 1$, but $x \equiv 3 \pmod{4}$.

# Pohlig-Hellman's Algorithm

Pohlig-Hellman$(\alpha, \beta, p)$

Factor $p - 1 = \prod_{i=1}^{k} p_i^{c_i}$
**for** $i$ = 1 to $k$ **do**
  Compute $x_i = (\log_\alpha \beta \pmod{p})$ modulo $p_i^{c_i}$
**endfor**
Use Chinese Remainder Theorem to compute $x$ modulo $p - 1$
  s.t. $x \equiv x_i \pmod{p_i^{c_i}}$ for $1 \le i \le k$
**return** $x$

To compute $x = \log_3 26 \pmod{29}$:
$p - 1 = 4 \cdot 7$
To compute $x \pmod{7}$: $\alpha' = 3^{28/7} = 23$, $\beta' = 26^{28/7} = 23$
Work in a subgroup of size 7. Get $x \equiv 1 \pmod{7}$.
More difficult when $c_i > 1$, but $x \equiv 3 \pmod{4}$.
Thus, $x \equiv 15 \pmod{28}$.

# Index Calculus Method

Suppose $\alpha$ is a primitive element modulo $p$.

Index Calculus$(\alpha, \beta, p)$

Choose a factor base $\mathcal{F} = \{p_1, p_2, \ldots, p_s\}$
Find $\log_\alpha p_i$ for all $i$:
Find random $\{x_1, x_2, \ldots, x_t\}$ s.t. $\alpha^{x_j} \pmod{p}$ factors over $\mathcal{F}$:
$\alpha^{x_j} = p_1^{e_{1,j}} p_2^{e_{2,j}} \cdots p_s^{e_{s,j}}$, for integers $e_{i,j}$
$x_j = e_{1,j} \log_\alpha p_1 + e_{2,j} \log_\alpha p_2 + \cdots + e_{s,j} \log_\alpha p_s \pmod{\text{p-1}}$
Solve for the $s$ unknowns $\log_\alpha p_i$ in a linear system of congruences.
Find $x = \log_\alpha \beta$:
**repeat**
Choose random $r \in \mathbb{Z}_{p-1}$
**until** $\beta\alpha^r \pmod{p}$ factors over $\mathcal{F}$
Suppose $\beta\alpha^r \equiv \alpha^{x+r} \equiv p_1^{e_1} p_2^{e_2} \cdots p_s^{e_s} \pmod{p}$
**return** $(-r + e_1 \log_\alpha p_1 + e_2 \log_\alpha p_2 + \cdots + e_s \log_\alpha p_s \pmod{\text{p-1}})$

Expected execution time: $O(e^{c\sqrt{\ln p \ln \ln p}})$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

$2x + y \equiv 7 \pmod{28}$
$x + 2y \equiv 15 \pmod{28}$
$y + z \equiv 19 \pmod{28}$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

$2x + y \equiv 7 \pmod{28}$
$x + 2y \equiv 15 \pmod{28}$
$y + z \equiv 19 \pmod{28}$

Subtracting twice first from second:
$-3x \equiv 1 \pmod{28}$, so $x = 9$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$$
$$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$$
$$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$$

$$2x + y \equiv 7 \pmod{28}$$
$$x + 2y \equiv 15 \pmod{28}$$
$$y + z \equiv 19 \pmod{28}$$

Subtracting twice first from second:
$-3x \equiv 1 \pmod{28}$, so $x = 9$
$y \equiv -11 \equiv 17 \pmod{28}$, so $y = 17$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

$2x + y \equiv 7 \pmod{28}$
$x + 2y \equiv 15 \pmod{28}$
$y + z \equiv 19 \pmod{28}$

Subtracting twice first from second:
$-3x \equiv 1 \pmod{28}$, so $x = 9$
$y \equiv -11 \equiv 17 \pmod{28}$, so $y = 17$
$z = 2$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

$2x + y \equiv 7 \pmod{28}$
$x + 2y \equiv 15 \pmod{28}$
$y + z \equiv 19 \pmod{28}$

Subtracting twice first from second:
$-3x \equiv 1 \pmod{28}$, so $x = 9$
$y \equiv -11 \equiv 17 \pmod{28}$, so $y = 17$
$z = 2$

Try $11^{27} \cdot 23 \equiv 10 \equiv 11^9 \cdot 11^2 \pmod{29}$

# Index Calculus Method Example

To find discrete log of 23 w.r.t. 11 in $\mathbb{Z}_{29}^*$:
Choose factor base $\{2, 3, 5\}$. $11^x = 2$, $11^y = 3$, $11^z = 5 \pmod{29}$.
Choose exponents randomly: 7, 15, 19.

$11^7 \pmod{29} \equiv 12 = 2^2 \cdot 3^1 \equiv 11^{2x} \cdot 11^y$
$11^{15} \pmod{29} \equiv 18 = 2^1 \cdot 3^2 \equiv 11^x \cdot 11^{2y}$
$11^{19} \pmod{29} \equiv 15 = 3^1 \cdot 5^1 \equiv 11^y \cdot 11^z$

$2x + y \equiv 7 \pmod{28}$
$x + 2y \equiv 15 \pmod{28}$
$y + z \equiv 19 \pmod{28}$

Subtracting twice first from second:
$-3x \equiv 1 \pmod{28}$, so $x = 9$
$y \equiv -11 \equiv 17 \pmod{28}$, so $y = 17$
$z = 2$

Try $11^{27} \cdot 23 \equiv 10 \equiv 11^9 \cdot 11^2 \pmod{29}$
Discrete log of 23 is $9 + 2 - 27 \pmod{28} \equiv 12$

# Discrete Logarithm Algorithms

The Index Calculus Method works over $\mathbb{Z}_p^*$ since multiplication over integers holds over $\mathbb{Z}_p^*$ if the result is less than $p$.

# Discrete Logarithm Algorithms

The Index Calculus Method works over $\mathbb{Z}_p^*$ since multiplication over integers holds over $\mathbb{Z}_p^*$ if the result is less than $p$.

This does not necessarily work for other groups.

# Discrete Logarithm Algorithms

The Index Calculus Method works over $\mathbb{Z}_p^*$ since multiplication over integers holds over $\mathbb{Z}_p^*$ if the result is less than $p$.

This does not necessarily work for other groups.

The best of the known general purpose algorithms for discrete logarithms modulo a prime is the General Number Field Sieve. Its expected execution time: $O(e^{c(\ln p(\ln \ln p)^2)^{1/3}})$

# Discrete Logarithm Algorithms

The Index Calculus Method works over $\mathbb{Z}_p^*$ since multiplication over integers holds over $\mathbb{Z}_p^*$ if the result is less than $p$.

This does not necessarily work for other groups.

The best of the known general purpose algorithms for discrete logarithms modulo a prime is the General Number Field Sieve.
Its expected execution time: $O(e^{c(\ln p(\ln \ln p)^2)^{1/3}})$
Better than the $O(e^{c\sqrt{\ln p \ln \ln p}})$ for Index Calculus.
Also does not necessarily work for other groups.

# Discrete Logarithm Algorithms

The Index Calculus Method works over $\mathbb{Z}_p^*$ since multiplication over integers holds over $\mathbb{Z}_p^*$ if the result is less than $p$.

This does not necessarily work for other groups.

The best of the known general purpose algorithms for discrete logarithms modulo a prime is the General Number Field Sieve.
Its expected execution time: $O(e^{c(\ln p(\ln \ln p)^2)^{1/3}})$
Better than the $O(e^{c\sqrt{\ln p \ln \ln p}})$ for Index Calculus.
Also does not necessarily work for other groups.

There is a polynomial time quantum algorithm for discrete logarithms.

# El Gamal Cryptosystem

With RSA, no two users should share the same prime.

With El Gamal, there are

Domain Parameters:

- large primes $p, q$, s.t. $q \mid (p-1)$
- $g \in \mathbb{Z}_p^*$ of order $q$

# El Gamal Cryptosystem

With RSA, no two users should share the same prime.

With El Gamal, there are

Domain Parameters:

- large primes $p, q$, s.t. $q \mid (p-1)$
- $g \in \mathbb{Z}_p^*$ of order $q$
  $g = r^{(p-1)/q} \pmod{p}$ for some $r \in \mathbb{Z}_p^*$

# El Gamal Cryptosystem

With RSA, no two users should share the same prime.

With El Gamal, there are

Domain Parameters:

- large primes $p, q$, s.t. $q \mid (p-1)$
- $g \in \mathbb{Z}_p^*$ of order $q$
  $g = r^{(p-1)/q} \pmod{p}$ for some $r \in \mathbb{Z}_p^*$
  How do you check the order of $g$?

# El Gamal Cryptosystem

With RSA, no two users should share the same prime.

With El Gamal, there are

Domain Parameters:

- large primes $p, q$, s.t. $q \mid (p-1)$
- $g \in \mathbb{Z}_p^*$ of order $q$
  $g = r^{(p-1)/q} \pmod{p}$ for some $r \in \mathbb{Z}_p^*$
  How do you check the order of $g$?
  Check $g \neq 1$.

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

Correctness:

$$\begin{aligned} m' &\equiv c_2 \cdot (c_1^x)^{-1} \\ &\equiv (m \cdot h^k) \cdot ((g^k)^x)^{-1} \\ &\equiv m \cdot (g^x)^k \cdot (g^{xk})^{-1} \\ &\equiv m \pmod{p} \end{aligned}$$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

Security:
Suppose a cryptanalyst can compute discrete logarithms in $\langle g \rangle$.
How can the system be broken?

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

Implementation: Can the system be implemented efficiently?

# Digital Signatures with RSA

Suppose Alice wants to sign a document $m$ such that:

- No one else could forge her signature
- It is easy for others to verify her signature

Note $m$ has arbitrary length.
RSA is used on fixed length messages.
Alice uses a cryptographically secure hash function $h$, such that:

- For any message $m'$, $h(m')$ has a fixed length (512 bits?)
- It is "hard" for anyone to find 2 messages $(m_1, m_2)$ such that $h(m_1) = h(m_2)$.

# Digital Signatures with RSA

Then Alice "decrypts" $h(m)$ with her secret RSA key $(N_A, d_A)$

$$s = (h(m))^{d_A} \pmod{N_A}$$

Bob verifies her signature using her public RSA key $(N_A, e_A)$ and $h$:

$$c = s^{e_A} \pmod{N_A}$$

He accepts if and only if

$$h(m) = c$$

.

This works because $s^{e_A} \pmod{N_A} =$

$$((h(m))^{d_A})^{e_A} \pmod{N_A} = ((h(m))^{e_A})^{d_A} \pmod{N_A} = h(m).$$

# Diffie-Hellman Key Exchange

| Alice – secret $a$ | | Bob – secret $b$ |
|---|---|---|
| Let $u = g^a \pmod{p}$ and<br>$S_u$ = Alice's signature on $u$ | $\xrightarrow{u,\, S_u}$ | |
| | | Verify $S_u$;<br>Let $v = g^b \pmod{p}$ and<br>$S_v$ = Bob's signature on $v$ |
| | $\xleftarrow{v,\, S_v}$ | |
| Verify $S_v$;<br>Let $k = v^a \pmod{p}$ | | |
| | | Let $k = u^b \pmod{p}$ |

# Diffie-Hellman Key Exchange

| Alice – secret $a$ | | Bob – secret $b$ |
|---|---|---|
| Let $u = g^{a} \pmod{p}$ and<br>$S_u$ = Alice's signature on $u$ | $\xrightarrow{u, S_u}$ | |
| | | Verify $S_u$;<br>Let $v = g^{b} \pmod{p}$ and<br>$S_v$ = Bob's signature on $v$ |
| | $\xleftarrow{v, S_v}$ | |
| Verify $S_v$;<br>Let $k = v^{a} \pmod{p}$ | | |
| | | Let $k = u^{b} \pmod{p}$ |

Correctness:

$v^a \equiv (g^b)^a \equiv g^{ab} \pmod{p}$

$u^b \equiv (g^a)^b \equiv g^{ab} \pmod{p}$

# Diffie-Hellman Key Exchange

Secrecy of $k$ depends on difficulty of finding $g^{ab} \pmod{p}$ from $g^a \pmod{p}$ and $g^b \pmod{p}$.

# Diffie-Hellman Key Exchange

Secrecy of $k$ depends on difficulty of finding $g^{ab} \pmod{p}$ from $g^a \pmod{p}$ and $g^b \pmod{p}$.

Easy if you can find discrete logs!

# Diffie-Hellman Key Exchange

Secrecy of $k$ depends on difficulty of finding $g^{ab} \pmod p$ from $g^a \pmod p$ and $g^b \pmod p$.

Easy if you can find discrete logs!

Computational Diffie-Hellman Problem (DHP):
Given an abelian group $G$, $g \in G$ of prime order $q$, $g^u$, $g^v$, $u, v$ unknown, chosen uniformly at random from $\{0, 1, \ldots, q-1\}$, find $g^{uv}$.

# Diffie-Hellman Key Exchange

Secrecy of $k$ depends on difficulty of finding $g^{ab} \pmod{p}$ from $g^a \pmod{p}$ and $g^b \pmod{p}$.

Easy if you can find discrete logs!

Computational Diffie-Hellman Problem (DHP):
Given an abelian group $G$, $g \in G$ of prime order $q$, $g^u$, $g^v$, $u, v$ unknown, chosen uniformly at random from $\{0, 1, \ldots, q-1\}$, find $g^{uv}$.

If you can solve the DHP efficiently in $\langle g \rangle$, Diffie-Hellman Key Exchange is insecure.

# Diffie-Hellman Key Exchange

Secrecy of $k$ depends on difficulty of finding $g^{ab} \pmod{p}$ from $g^a \pmod{p}$ and $g^b \pmod{p}$.

Easy if you can find discrete logs!

Computational Diffie-Hellman Problem (DHP):
Given an abelian group $G$, $g \in G$ of prime order $q$, $g^u$, $g^v$, $u, v$ unknown, chosen uniformly at random from $\{0, 1, \ldots, q-1\}$, find $g^{uv}$.

If you can solve the DHP efficiently in $\langle g \rangle$, Diffie-Hellman Key Exchange is insecure.

If you can break Diffie-Hellman Key Exchange (find $k$) efficiently, you can solve the DHP efficiently.

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod p$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod p, m \cdot h^k \pmod p)$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod p$

Security: If cryptanalyst can efficiently compute discrete logarithms in $\langle g \rangle$, the system be broken efficiently.

# El Gamal Cryptosystem

Suppose a cryptanalyst can efficiently solve the DHP in $\langle g \rangle$.
Eve can compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$, $\beta = g^v \pmod{p}$:

# El Gamal Cryptosystem

Suppose a cryptanalyst can efficiently solve the DHP in $\langle g \rangle$.
Eve can compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$,
$\beta = g^v \pmod{p}$:

Compute $\delta = g^{xk} \pmod{p}$, from
$h = g^x \pmod{p}$ , $c_1 = g^k \pmod{p}$.
Compute $m = c_2 \cdot \delta^{-1} \pmod{p}$.

# El Gamal Cryptosystem

Suppose a cryptanalyst can efficiently solve the DHP in $\langle g \rangle$.
Eve can compute $g^{uv} \pmod p$ from $g$, $\alpha = g^u \pmod p$,
$\beta = g^v \pmod p$:

Compute $\delta = g^{xk} \pmod p$, from
$h = g^x \pmod p$ , $c_1 = g^k \pmod p$.
Compute $m = c_2 \cdot \delta^{-1} \pmod p$.

Then, you can break the El Gamal Cryptosystem.

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

To compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$, $\beta = g^v \pmod{p}$:

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

To compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$,
$\beta = g^v \pmod{p}$:

Use the same $g$.
Let $h = g^x = \alpha$, $c_1 = \beta$, and $c_2 \in_R \langle g \rangle$. (Note $x = u$.)

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

To compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$,
$\beta = g^v \pmod{p}$:

Use the same $g$.
Let $h = g^x = \alpha$, $c_1 = \beta$, and $c_2 \in_R \langle g \rangle$. (Note $x = u$.)

The cryptanalyst computes $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.
Compute $\delta \equiv c_2 \cdot m'^{-1} \equiv c_1^x \equiv \beta^x \equiv (g^v)^x \equiv g^{uv} \pmod{p}$.

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

To compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$,
$\beta = g^v \pmod{p}$:

Use the same $g$.
Let $h = g^x = \alpha$, $c_1 = \beta$, and $c_2 \in_R \langle g \rangle$. (Note $x = u$.)

The cryptanalyst computes $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.
Compute $\delta \equiv c_2 \cdot m'^{-1} \equiv c_1^x \equiv \beta^x \equiv (g^v)^x \equiv g^{uv} \pmod{p}$.

So, you can efficiently solve the DHP in $\langle g \rangle$.

# El Gamal Cryptosystem

Suppose a cryptanalyst can break the El Gamal Cryptosystem.

From $(c_1, c_2), g, h$, he/she can compute $m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.

To compute $g^{uv} \pmod{p}$ from $g$, $\alpha = g^u \pmod{p}$,
$\beta = g^v \pmod{p}$:

Use the same $g$.
Let $h = g^x = \alpha$, $c_1 = \beta$, and $c_2 \in_R \langle g \rangle$. (Note $x = u$.)

The cryptanalyst computes $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$.
Compute $\delta \equiv c_2 \cdot m'^{-1} \equiv c_1^x \equiv \beta^x \equiv (g^v)^x \equiv g^{uv} \pmod{p}$.

So, you can efficiently solve the DHP in $\langle g \rangle$.

Lots more can be said about the security of El Gamal, plus and minus.

# Elliptic Curves

In the El Gamal Cryptosystem and the Diffie-Hellman Key Exchange, we just used that we had an abelian group with a large cyclic subgroup of prime order.

# Elliptic Curves

In the El Gamal Cryptosystem and the Diffie-Hellman Key Exchange, we just used that we had an abelian group with a large cyclic subgroup of prime order.

Suppose we use elliptic curves over $\mathbb{Z}_p$, $p$ large prime:

$$E(p): \; Y^2 \equiv X^3 + aX + b \pmod{p}$$

Point at infinity = identity: $\mathcal{O}$
$\mathcal{O} + P = P + \mathcal{O} = P$

# Elliptic Curves

In the El Gamal Cryptosystem and the Diffie-Hellman Key Exchange, we just used that we had an abelian group with a large cyclic subgroup of prime order.

Suppose we use elliptic curves over $\mathbb{Z}_p$, $p$ large prime:

$$E(p): \; Y^2 \equiv X^3 + aX + b \pmod{p}$$

Point at infinity $=$ identity: $\mathcal{O}$
$\mathcal{O} + P = P + \mathcal{O} = P$

Instead of multiplication, we use addition:
Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$.
$-P = (x_1, -y_1)$, $P + (-P) = \mathcal{O}$

# Elliptic Curves

In the El Gamal Cryptosystem and the Diffie-Hellman Key Exchange, we just used that we had an abelian group with a large cyclic subgroup of prime order.

Suppose we use elliptic curves over $\mathbb{Z}_p$, $p$ large prime:

$$E(p): \; Y^2 \equiv X^3 + aX + b \pmod{p}$$

Point at infinity = identity: $\mathcal{O}$
$\mathcal{O} + P = P + \mathcal{O} = P$

Instead of multiplication, we use addition:
Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$.
$-P = (x_1, -y_1)$, $P + (-P) = \mathcal{O}$

Let $x_3 = \lambda^2 - x_1 - x_2$
$P + Q = (x_3, \; (x_1 - x_3) \cdot \lambda - y_1)$
where if $x_1 \neq x_2$, $\qquad \lambda = \frac{y_2 - y_1}{x_2 - x_1}$,
and if $x_1 = x_2$, $y_1 \neq 0$, $\qquad \lambda = \frac{3x_1^2 + a}{2y_1}$.

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Idea: Use elliptic curves: shorter keys for same security.

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Idea: Use elliptic curves: shorter keys for same security.

Fact: The number of points on an elliptic curve $E(p)$ can be computed in $\tilde{O}((\log p)^4)$ time.

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Idea: Use elliptic curves: shorter keys for same security.

Fact: The number of points on an elliptic curve $E(p)$ can be computed in $\tilde{O}((\log p)^4)$ time.

So we can find a curve and a point with large order.

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Idea: Use elliptic curves: shorter keys for same security.

Fact: The number of points on an elliptic curve $E(p)$ can be computed in $\tilde{O}((\log p)^4)$ time.

So we can find a curve and a point with large order.

How do you do the division? $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Idea: Use elliptic curves: shorter keys for same security.

Fact: The number of points on an elliptic curve $E(p)$ can be computed in $\tilde{O}((\log p)^4)$ time.

So we can find a curve and a point with large order.

How do you do the division? $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

Extended Euclidean Algorithm.

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod p$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod p), m \cdot h^k \pmod p)$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod p$

# El Gamal Cryptosystem

Keys: Easy to create from Domain Parameters

- $PK_A = h = g^x \pmod{p}$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m \in \langle g \rangle$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (c_1, c_2) = (g^k \pmod{p}, m \cdot h^k \pmod{p})$

Decryption of $(c_1, c_2)$:

- $m' = c_2 \cdot (c_1^x)^{-1} \pmod{p}$

Implementation: How do you do the "exponentiation"?

# El Gamal Cryptosystem, with Elliptic Curves

Keys: $E(p)$, $G$ generating a subgroup of large prime order $q$, invertible function $f$ mapping $m$ to $P_m$ on $E(p)$,

- $PK_A = H = x \cdot G$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (C_1, C_2) = (k \cdot G, f(m) + k \cdot H)$

Decryption of $(C_1, C_2)$:

- $P = C_2 - (x \cdot C_1)$
- $m' = f^{-1}(P)$

## El Gamal Cryptosystem, with Elliptic Curves

Keys: $E(p)$, $G$ generating a subgroup of large prime order $q$, invertible function $f$ mapping $m$ to $P_m$ on $E(p)$,

- $PK_A = H = x \cdot G$
- $SK_A = x \in \{0, 1, \ldots, q-1\}$

Encryption of $m$:

- Choose random $k \in \{0, 1, \ldots, q-1\}$
- $E(m, k, PK_A) = (C_1, C_2) = (k \cdot G, f(m) + k \cdot H)$

Decryption of $(C_1, C_2)$:

- $P = C_2 - (x \cdot C_1)$
- $m' = f^{-1}(P)$

Correctness:

$$\begin{aligned} f(m') &\equiv C_2 - (x \cdot C_1) \\ &\equiv (f(m) + k \cdot H) - (x \cdot (k \cdot G)) \\ &\equiv f(m) + k \cdot (x \cdot G) - (xk \cdot G) \\ &\equiv f(m) \end{aligned}$$

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Two values modulo $p$ for each point. Can we save space?

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Two values modulo $p$ for each point. Can we save space?

**Lemma**: Let $p$ be an odd prime, $y_1, y_2 \in \mathbb{Z}_p^*$. If $y_1 = -y_2 \pmod{p}$, then $y_1 \pmod{2} \neq y_2 \pmod{2}$.

# Elliptic Curves over $\mathbb{Z}_p$, $p$ large prime

Two values modulo $p$ for each point. Can we save space?

**Lemma**: Let $p$ be an odd prime, $y_1, y_2 \in \mathbb{Z}_p^*$. If $y_1 = -y_2 \pmod{p}$, then $y_1 \pmod{2} \neq y_2 \pmod{2}$.

$\text{PointCompress}(x, y) = (x, y \pmod{2})$

$\text{PointDecompress}(x, i)$
- $z \leftarrow x^3 + ax + b \pmod{p}$
- $y \leftarrow \sqrt{z} \pmod{p}$
- **if** $y \equiv i \pmod{2}$
  - **then return** $(x, y)$
  - **else return** $(x, p - y)$