Day 22 – Discrete Log Problem and Applications

**Definition 1.** Fix a prime \( p \) and a primitive root \( \alpha \) modulo \( p \). For a given integer \( \beta \), we want to find \( x \) such that

\[
\alpha^x \equiv \beta \pmod{p}.
\]

In this case, we write \( x = \log_\alpha(\beta) \pmod{p} \).

- We take \( \alpha \) to be a primitive root so that the discrete log problem can have a solution for any RHS value \( \beta \).
- Given a prime \( p \), it is fairly easy to find a primitive root.
- For small \( p \), we can compute discrete logs by exhaustive search.
- In general, computing discrete logs is hard (no known polynomial time algorithm)
- In other words, exponentiation mod \( p \) is believed to be a trapdoor function.
El Gamal Cryptosystem

1. Alice and Bob agree on a prime $p$ and a primitive root $r$ modulo $p$.

2. Alice comes up with her secret component $a$ and compute her public key $\alpha = r^a \mod p$.

3. Bob comes up with his secret component $b$ and compute his public key $\beta = r^b \mod p$.

4. Suppose Bob wants to send the message $X$ to Alice. He then picks a session key $k$, which is a random number in the interval $[2, p - 2]$.

5. Bob then computes $U = r^k \mod p$ and $V = \alpha^k X = r^{ak} X \mod p$. He then sends $(U, V)$ to Alice.

6. To decrypt Bob’s message, Alice computes

$$U^{-a} V = r^{-ka} \alpha^k X = r^{-ak} r^{ak} X = X.$$

<table>
<thead>
<tr>
<th>Alice</th>
<th>Public $\approx$ Eve</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p, r$</td>
<td>prime $p$; primitive root $r$</td>
<td>$p, r$</td>
</tr>
<tr>
<td>private component: $a$</td>
<td>$\alpha, \beta$</td>
<td>private component $b$</td>
</tr>
<tr>
<td>compute $\alpha = r^a \mod p$</td>
<td>$\beta$</td>
<td>compute $\beta = r^b \mod p$</td>
</tr>
<tr>
<td>$U, V$</td>
<td>decrypt: $U \uparrow: V = (X) \mod p$</td>
<td>U, V</td>
</tr>
<tr>
<td>only Alice can decrypt</td>
<td></td>
<td>plaintext: $X$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>session key: $k$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$U = r^k \mod p$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$V = \alpha^k X \mod p$</td>
</tr>
</tbody>
</table>

Enemy knows: $p, r, \alpha, \beta, U, V$
What can the enemy do?

Suppose that Eve intercepts the pair \((U, V)\), but to recover the message, Eve would need to know Alice’s secret component \(a\). Eve then has two choices, either

- solve for \(a\) from \(\alpha = r^a \mod p\) or
- solve for \(k\) from \(U = r^k \mod p\) then compute \(\alpha^{-k} V = \alpha^k a^k X = X\).

An example of El Gamal

\[ p = 11881379 \quad r = 23 \]

Alice publishes \(\alpha = 1308503 = 23^{55} \mod 11881379\)

Bob wants to send the message “LUNCH” to Alice

- converts it to decimals and gets \(5387103 = X\)

Bob chooses \(k = 123\) as the session key and calculates

\[ U = r^k = 23^{123} = 1777907 \mod 11881379 \]

\[ V = \alpha^k X = 1308503^{123} \times 5387103 = 4944577 \mod 11881379 \]

sends Alice \(1777907: 4944577\)

Alice decrypts it and understands that she is invited for lunch.

Eve can also send a message to Alice.

Although she cannot find out what Bob sent to Alice to confuse the issues she sends Alice the pair \(5387871: 7127763\)

Alice computes

\[ U^{-58} = 5387871^{58} = 5387871^{11881379} \mod 11881379 = 3552158 \]

\[ U^{-58} V = 3552158 \times 7127763 = 6866650 \mod 11881379 \]

Alice converts this decimal to text and gets “PARTY”

Alice understands that Eve is inviting her for a party and takes a rain check on Bob’s offer.
\[(11, 20, 0, 2, 0) \quad \text{LUNCH} \quad \downarrow \quad \text{X} \]
\[
666650 = 15 \cdot 20^4 + 12010 = 15 \cdot 20^4 + 17 \cdot 20^2 + 518 \\
= 15 \cdot 20^4 + 0 \cdot 20^3 + 17 \cdot 20^2 + 19 \cdot 20^1 + 24 = 26^5 \\
\downarrow \quad \text{PARTY} \\
\]

One of the major downsides of public key cryptosystems (e.g. RSA, El Gamal) is that they are relatively slow compared to modern ciphers (DES, AES). If you want to encrypt a good bit of data,

1. Use a public key cryptosystem to establish a secret key.

2. Encrypt the actual data using a fast cipher with the established secret key.
Diffie-Hellman Key Exchange

Alice and Bob want to share a secret key for use in a symmetric cipher, but their only means of communication is insecure. Every piece of information that they exchange is observed by their adversary Eve.

1. (public) The first step is for Alice and Bob to agree on a large prime $p$ and a nonzero integer $r$ modulo $p$ with larger order mod $p$. For instance, say $r$ is a primitive root mod $p$. (In practice, it is best if they choose $r$ such that its order is a large prime.)

Alice and Bob make the values of $p$ and $r$ public knowledge; for example, they might post the values on their web sites, so Eve knows them, too.

2. (private): Alice chooses $x \pmod{p}$ and Bob chooses $y \pmod{p}$.

3. (private): Alice computes $A \equiv r^x \pmod{p}$ and Bob computes $B \equiv r^y \pmod{p}$.

4. (public): Alice and Bob exchange the values $A$ and $B$ (Eve can see this).

5. (private) Alice computes $B^x \pmod{p}$ and Bob computes $A^y \pmod{p}$. Note that both these values are equal to $k \equiv r^{xy} \pmod{p}$. This is their shared private key.
A public key exchange system

TABLE OF POWERS OF 17 MOD 37

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |
| 17 | 30 | 29 | 12 | 19 | 27 | 15 | 33 | 6 | 28 | 32 | 26 | 35 | 3 | 14 | 31 | 13 | 36 | 20 | 7 | 8 | 25 | 18 | 10 | 22 | 4 | 31 | 9 | 5 | 11 | 2 | 36 | 23 | 21 | 28 | 1 |

numbers in boxes are the secret exponents

numbers in circles are the public keys

\[ p = 37 \]

\[ \text{primitive root } r = 19 \]

The key common to \( p_3 \) and \( p_6 \) is

\[
\begin{pmatrix}
5 & 17 \\
9 &
\end{pmatrix}
\]
\[ \equiv
\begin{pmatrix}
28 & 17 \\
30 &
\end{pmatrix}
\]
\[ \mod 37 \]

\[ 17 \equiv 9 \mod 37 \]

\( (9, 10) \) are the private component of \( p_3, p_6 \)

What can your enemy do?

Note that if Eve can solve the discrete log problem (i.e., given \( A \) and \( B \) find \( x \) and \( y \)) then she can find \( k \). It is nontrivial, but true, that finding \( k \) given \( A, B, r \) and \( p \) is as hard as solving DLP.
DIGITAL SIGNATURES
SECURITY
If you want security and authentication, (using RSA) we can combine these two ideas.
Alice and Bob agree to use the modulus \( n = p \times q \) which they make public

\[ p, q \text{ and } \varphi(n) \text{ secret} \]

Bob
- \( e_B \) public
- \( d_B \) secret

Alice
- \( e_A \) public
- \( d_A \) secret

- Alice encrypts her actual message using Bob's public key.
- In addition, Alice encrypts a short signature using her private key.
- Alice sends both encrypted messages to Bob.

\[ C_1 = M^{e_B} \mod n \]
\[ C_2 = S^{d_A} \mod n \]

\[ \varphi(n) = (p-1)(q-1) \]

\[(S) \quad \text{Alice} \quad \text{public} \quad \text{Bob} \quad (R) \]

\[ p, q; n = p \cdot q \]

\[ e_A, e_B \]

\[ d_A, d_B \]

\[ \text{private: } d_A \]
\[ \text{public: } e_A \]
\[ \text{private: } d_B \]
\[ \text{public: } e_B \]

Decryption:
\[ M = C_1^{d_B} \mod n \rightarrow \text{RSA} \]
\[ S = C_2^{e_A} \mod n \]

\( \varphi(n) \)

\[ \text{inverse mod } \varphi(n) \]

\[ C_1 = M^{e_B} \mod n \]

\[ C_2 = S^{d_A} \mod n \]

\[ \text{encrypted verifying } q A's \text{ signature.} \]

\[ \text{signature only Alice knows} \]

\[ \text{signature } S \]

\[ \text{plaintext } M \]

\[ 2 \text{ RSA problems} \]
DIGITAL SIGNATURES

In 2000 the US federal government approved legislation that gives electronic signatures the same legal standing as pen and paper signatures.

A bank uses RSA to communicate with its customers.

The bank publishes a modulus $m_b = p_b q_b$ and an encrypting exponent $e_b$ keeping the two primes $p_b q_b$ and the decrypting exponent $d_b$ secret.

Each customer $C_i$ is required to do the same

Let $m_i = p_i q_i < m_b$ be the modulus and $e_i$ the encrypting exponent selected by $C_i$.

$C_i$ is required to make $m_i$ and $e_i$ known to the bank and keep $p_i q_i$ and the decrypting exponent $d_i$ secret.

When the time comes for $C_i$ to send a message $X$ to the bank.

The customer $C_i$ computes

$U = X^{e_i} \pmod{m_b}$ \quad $V = X^{d_i} \pmod{m_b}$ \quad $W = V^{e_b} \pmod{m_b}$

$C_i$ then sends an E-mail to the Bank containing the pair $(U, W) = (X^{e_i}, (X^{d_i})^{e_b})$

The Bank computes

$U^{d_i} = X^{m_b d_i} = X \pmod{m_b}$ \quad $W^{d_b} = V^{m_b d_b} = V \pmod{m_b}$

and then

$V^{e_b} = X^{d_i e_b} = X \pmod{m_b}$

Eve could very well intercept $(U, W)$ and replace it by $(U', W)$

But she would be unable to manufacture $(U', W)$ in such a way that the Bank calculation would yield the same message $X$ from both $U$ and $W$.
**Zero Knowledge Protocol**

Suppose $N = p \cdot q$ is the product of two large primes. Suppose $y = s^2$ is a square modulo $N$ where $\gcd(y, N) = 1$.

Suppose that Alice claims to know the square root $s$ of $y$ but does not want to reveal $s$. Bob wants to verify this. Thus, they agree on the following protocol:

1. Alice chooses a random number $r_1$ and computes $r_2 = sr_1^{-1} \pmod{N}$.
2. Then she computes $x_1 = r_1^2$ and $x_2 = r_2^2 = (sr_1^{-1})^2 \pmod{N}$. She sends $x_1, x_2$ to Bob.
3. Bob computes $x_1 x_2 = (r_1^2)(sr_1^{-1})^2 = (r_1sr_1^{-1})^2 = s^2 = y$.
4. Then Bob chooses at random either $x_1$ or $x_2$ and asks Alice to provide its square root.
5. Repeat the previous step several times until Bob is convinced that Alice has the square root $s$ of $y$. 

---

Bob verifies that may be.

Alice does not know the key.

Only Alice can do, since only her knows $s$. 