The Discrete Logarithm Problem

Debdeep Mukhopadhyay
IIT Kharagpur

The DLP Problem

• Consider $\alpha \in G$, having order $n$.
  – $\langle \alpha \rangle = \{\alpha^i : 0 \leq i \leq n-1\}$ is a cyclic sub-group of $G$ having order $n$.

<table>
<thead>
<tr>
<th>Problem 6.1: Discrete Logarithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong></td>
</tr>
<tr>
<td><strong>Question:</strong></td>
</tr>
</tbody>
</table>

We will denote this integer $a$ by $\log_\alpha \beta$; it is called the discrete logarithm of $\beta$. 
Cryptographic Utility of DLP

• For suitable choices of the parameters, finding Discrete Logarithm seems to be difficult.

• However, the inverse operation of exponentiation is efficiently computable by the square and multiply algorithm.
  – Exponentiation is a candidate one-way function.

The ElGamal Cryptosystem

Cryptosystem 6.1: ElGamal Public-key Cryptosystem in \( \mathbb{Z}_p^* \)

Let \( p \) be a prime such that the Discrete Logarithm problem in \( (\mathbb{Z}_p^*, \cdot) \) is infeasible, and let \( \alpha \in \mathbb{Z}_p^* \) be a primitive element. Let \( \mathcal{P} = \mathbb{Z}_p \times \mathbb{Z}_p^* \), and define

\[ X = \{ (p, \alpha, \beta) : \beta \equiv \alpha^x \pmod{p} \}. \]

The values \( p, \alpha \) and \( \beta \) are the public key, and \( \alpha \) is the private key.

For \( K = (p, \alpha, \beta) \), and for a (secret) random number \( k \in \mathbb{Z}_{p-1} \), define

\[ e_K(x, k) = (y_1, y_2), \]

where

\[ y_1 = \alpha^k \pmod{p} \]

and

\[ y_2 = x \beta^k \pmod{p}. \]

For \( y_1, y_2 \in \mathbb{Z}_p^* \), define

\[ d_K(y_1, y_2) = y_2(y_1^*)^{-1} \pmod{p}. \]
Working of the algorithm

- Plaintext \( x \) is masked by multiplying it by \( \beta^k \), yielding \( y_2 \).
- The value \( \alpha^k \) is also transmitted as a part of the ciphertext.
- Bob who has the secret ‘a’ can compute \( \beta^k \) by raising \( \alpha^k \) to ‘a’.
- Then he obtains \( x \) by dividing \( y_2 \) with \( \beta^k \).
- Note that for each plaintext, there are \( p-1 \) possible ciphertexts.

Example

- \( p=2579 \), \( \alpha=2 \) (primitive element of \( \mathbb{Z}_{p}^{*} \))
- \( a=765 \) (secret value)
- \( \beta=2^{765} \mod 2579=949 \).
- Suppose, Alice wishes to send \( x=1299 \) to Bob. She randomly chooses \( k=853 \).
  - \( y_1=2^{853} \mod 2579 = 435 \)
  - \( y_2=1299(949^{853}) \mod 2579=2396 \)
- Alice sends \( y=(435,2396) \)
- Bob computes \( x=2396(435^{765})^{-1} \mod 2579=1299 \).
Algorithms for the DLP Problem

• If $\alpha^i$ was monotonically non decreasing with $i$, we could have done a binary search to find $i$.
  – but the problem with modular exponentiation is that there is no ordering of the powers.
  – Thus one have to do an exhaustive search in the worst case.
    • Thus it can be solved in $O(n)$ time and $O(1)$ space.
    – However pre-computation helps.

Time Memory Trade Off

• Suppose we store all possible values of $\alpha^i \pmod{p}$ as ordered pairs $(i, \alpha^i \mod p)$ and sort the elements wrt the second parameter. Now search for the given challenge by employing binary search.
  • Complexity: Pre-computation $O(n)$, Memory $O(n)$, Time to sort: $O(n\log n)$ [using a good sorting algorithm], Time to search $O(\log n)$
  • Often we neglect the $\log n$ terms in these algorithms, as $n$ is much larger than $\log n$
    – thus Time to search $O(1)$ and Pre-computation or Memory both are $O(n)$
Non-trivial Algorithms

- **Shank’s Algorithm**
- Pollard Rho Discrete Log Algorithm
- Index Calculus Method

**Shanks Algorithm**

\[
\text{Algorithm 6.1: } \text{SHANKS}(G, n, \alpha, \beta) \\
1. \quad m \leftarrow \lceil \sqrt{n} \rceil \\
2. \quad \text{for } j \leftarrow 0 \text{ to } m - 1 \\
\quad \quad \text{do compute } \alpha^{mj} \\
3. \quad \text{Sort the } m \text{ ordered pairs } (j, \alpha^{mj}) \text{ with respect to their second coordinates, obtaining a list } L_1 \\
4. \quad \text{for } i \leftarrow 0 \text{ to } m - 1 \\
\quad \quad \text{do compute } \beta \alpha^{-i} \\
5. \quad \text{Sort the } m \text{ ordered pairs } (i, \beta \alpha^{-i}) \text{ with respect to their second coordinates, obtaining a list } L_2 \\
6. \quad \text{Find a pair } (j, y) \in L_1 \text{ and a pair } (i, y) \in L_2 \text{ (i.e., find two pairs having identical second coordinates)} \\
7. \quad \log_{\alpha} \beta \leftarrow (mj + i) \mod n
Explanation

• \( \alpha^{mj}=y=\beta \alpha^i \Rightarrow \alpha^{mj+i}=\beta. \)
• If \( \beta \in \mathbf{a} \), \( \log_{\alpha}{\beta}=(mj+i) \mod n \), where both \( 0 \leq i, j \leq m-1 \).
  – The search is successful as we can ensure that \( \log_{\alpha}{\beta} \leq m(m-1)+(m-1)=n-1 \), as desired.
  – Complexity: \( O(m) \)

Example

• Compute, \( \log_3{525} \) in \( Z_{809}^* \). Note 809 is prime and 3 is a primitive element of \( Z_{809}^* \).
• Order=\( n=808 \), \( \beta=525 \), \( m=\sqrt{808}=29 \)
• \( \alpha^{29} \mod 809=99 \)
Tables

L1:

\begin{array}{cccc}
(0,1) & (1,99) & (2,93) & (3,398) \\
(5,329) & (6,211) & (7,664) & (8,307) \\
(10,644) & (11,654) & (12,26) & (13,147) \\
(15,727) & (16,781) & (17,464) & (18,632) \\
(20,528) & (21,496) & (22,564) & (23,15) \\
(25,586) & (26,575) & (27,295) & (28,81) \\
\end{array}

\textbf{Match:}

(10,444) and
(19,644)

\log_3{525}=(29\times10 +19) \mod 808=309.

L2:

\begin{array}{cccc}
(0,525) & (1,175) & (2,328) & (3,379) \\
(5,132) & (6,44) & (7,554) & (8,724) \\
(10,440) & (11,696) & (12,708) & (13,256) \\
(15,388) & (16,399) & (17,133) & (18,314) \\
(20,754) & (21,521) & (22,713) & (23,777) \\
(25,356) & (26,658) & (27,489) & (28,163) \\
\end{array}

The Diffie Hellman Problem

\textbf{Problem 6.3: Computational Diffie-Hellman}

\textbf{Instance:} A multiplicative group \((G, \cdot)\), an element \(\alpha \in G\) having order \(n\), and two elements \(\beta, \gamma \in \langle \alpha \rangle\).

\textbf{Question:} Find \(\delta \in \langle \alpha \rangle\) such that \(\log_\alpha \delta = \log_\alpha \beta \times \log_\alpha \gamma \pmod{n}\). (Equivalently, given \(\alpha^b\) and \(\alpha^c\), find \(\alpha^{bc}\).)

\textbf{Problem 6.4: Decision Diffie-Hellman}

\textbf{Instance:} A multiplicative group \((G, \cdot)\), an element \(\alpha \in G\) having order \(n\), and three elements \(\beta, \gamma, \delta \in \langle \alpha \rangle\).

\textbf{Question:} Is it the case that \(\log_\alpha \delta = \log_\alpha \beta \times \log_\alpha \gamma \pmod{n}\)? (Equivalently, given \(\alpha^b\), \(\alpha^c\) and \(\alpha^d\), determine if \(d \equiv bc \pmod{n}\).)

\textbf{\(DDH \ll_p CDH \ll_p DLP\)}

\textbf{\(\text{Thus DDH hardness is the strongest assumption.}\)}
Application: The DH Key Agreement Scheme

- **Public**: g and p
- **Secret**: Alice’s exponent a, Bob’s exponent b

Alice computes \((g^b)^a = g^{ba} = g^{ab} \mod p\)
Bob computes \((g^a)^b = g^{ab} \mod p\)
Could use \(K = g^{ab} \mod p\) as symmetric key

Subject to man-in-the-middle (MiM) attack

Trudy shares secret \(g^{at} \mod p\) with Alice
Trudy shares secret \(g^{bt} \mod p\) with Bob
Alice and Bob don’t know Trudy exists!
Designing Cryptographic Protocols

• The Man in the Middle Attack on the DH key agreement scheme shows that although the primitives are strong, the protocol can be weak.
• Thus, the next question is how to design strong protocols from strong primitives.
• We will not discuss in depth, but have a brief overview as our last topic this semester…

Possible Preventions

• How to prevent MiM attack?
  – Encrypt DH exchange with symmetric key
  – Encrypt DH exchange with public key
  – Sign DH values with private key
  – May be other methods also exist