4. Discrete Logarithm Attacks

Many cryptosystems could be broken if we can compute discrete logarithms quickly, that is, if we could solve the equation $a^x = b$ in a large finite field.

Samuel S. Wagstaff, Jr.
Professor of Computer Science at Purdue University

4.1 Introduction

The problem of computing discrete logarithms is fundamental in computational number theory, computational algebra, and of great importance in public-key cryptography.

First of all, let us recall the Discrete Logarithm Problem (DLP):

$$\text{DLP : } \{ n \in \mathbb{Z}_{>1}^+, x, y, k \in \mathbb{Z}^+, y \equiv x^k \pmod n \} \xrightarrow{\text{find}} \{ k \}.$$

As we already know DLP is computationally intractable and the security of several well known cryptosystems, such as the Diffie-Hellman-Merkle key-exchange scheme, the ElGamal systems and the US Government’s digital signature standard/algorithym (DSS/DSA), is based on the intractability of DLP. So, as for the factoring attack for RSA, the discrete logarithm attack is also the most direct attack for all the DLP-based cryptographic systems.

It is interesting to note that the discrete logarithm attack is also the most direct attack for the IFP-based cryptographic systems in general and the RSA cryptographic system in particular. On the one hand, if the adversary Eve has an efficient integer factorization algorithm, she can factor $N = pq$ in polynomial-time, and can compute $d \equiv 1/e \mod ((p - 1)(q - 1))$ and hence recover $C \equiv M^d \pmod N$. On the other hand, if Eve has an efficient discrete logarithm algorithm, she can make her own plaintext $M$ and use the publically known information $(e, N)$ to get her own ciphertext $C$ for $M$, then
she can use her discrete logarithm algorithm to compute \( d \equiv \log_{M^e} M \mod N \), that is,

\[
\{ N, M, M^e \} \xrightarrow{\text{find}} \{ d \}.
\]

This is the same as to say that if one can solve the DLP problem in polynomial-time, one can break the RSA system in polynomial-time:

\[
\text{DLP} \xrightarrow{\mathcal{P}} \text{RSA}.
\]

**Example 4.1.1.** Let the public-key and the cipher-text be as follows (see the RSA original paper [262]):

\[
(e, N, C) = (17, 2773, 2258)
\]

Find the corresponding \( M \) such that \( C \equiv M^e \mod N \). The most obvious way to invert the encryption function \( M \mapsto C \) is to find \( d \) by computing \( d \equiv 1/e \mod \phi(N) \) so that \( M \equiv C^d \mod N \), but to do so we need to factor \( N \). Now we claim that we can find \( d \) in a different way as follows:

1. Choose a piece of any reasonable plain-text \( M \) with \( \gcd(N, M) = 1 \), say \( M = 2113 \) and do the following calculation:

   \[
   C \equiv M^e \equiv 2113^{17} \equiv 340 \mod 2773.
   \]

2. Compute \( d \) by taking the discrete logarithm \( M \) to the base \( C \) modulo \( N \):

   \[
   d \equiv \log_C M \equiv \log_{340} 2113 \equiv 157 \mod 2773.
   \]

3. Now we can decrypt the given cipher-text \( C = 2258 \) by using this \( d \):

   \[
   M \equiv C^d \equiv 2258^{157} \equiv 1225 \mod 2773.
   \]

4. The result can be easily verified to be true, since

   \[
   C \equiv M^e \equiv 1225^{17} \equiv 2258 \mod 2773.
   \]

**Example 4.1.2.** Let the public-key and the cipher-text be as follows

\[
(e, N, C) = (7, 69056069, 19407420)
\]

(see page 173 of [218]), find \( M \) such that \( C \equiv M^e \mod N \). We perform the following computations:

1. Choose a piece of any reasonable plain-text \( M \) with \( \gcd(N, M) = 1 \), say \( M = 59135721 \) and do the following calculation:

   \[
   C \equiv M^e \equiv 59135721^7 \equiv 60711351 \mod 69056069.
   \]