How to Change the World with Donald Knuth

Abraham Xiao

Masdar Institute of Science and Technology

Information Security Project Presentation

Discrete Logarithm in a Nutshell

The security of many cryptographic techniques depends on the intractability of discrete logarithm problem.

A partial list of these include:

- DiffieHellman key agreement and its derivatives.
- ElGamal encryption.
- ElGamal signature scheme and its variants.

General setting for algorithms in this section are:

- A (multiplicatively written) finite cyclic group G
- n is the order of group G
- α is a generator of group G[1]

[1] For more math background, refer to [Ros12].

Relevant Definitions

Cyclic group and its generator.

Definition

A group is *cyclic* if there is an element $\alpha \in G$ such that for each $b \in G$ there is an integer i with $b = \alpha^i$. Such an element α is called a generator of G.

Discrete logarithm.

Definition

Let G be a finite cyclic group of order n. Let α be a generator of G, and let $\beta \in G$. The *discrete logarithm of β to the base α*, denoted $\log_\alpha \beta$, is the unique integer x, $0 \leq x \leq n-1$, such that $\beta = \alpha^x$[MVO96].

A Discrete Logarithm Example

Example

Let $p = 97$. Then $\mathbb{Z}_{97}^*$ is a cyclic group of order $n = 96$. A generator of $\mathbb{Z}_{97}^*$ is $\alpha = 5$. Since $5^{32} \equiv 35 \mod 97$, $\log_5 35 = 32$ in $\mathbb{Z}_{97}^*$.

The DiffieHellman Problem

The DiffieHellman problem is closely related to the well-studied discrete logarithm problem.

Definition

The *DiffieHellman problem* is the following: given a prime p, a generator α of $\mathbb{Z}_p^*$, and elements $\alpha^a \mod p$ and $\alpha^b \mod p$, find $\alpha^{ab} \mod p$.

Wait! Could we just possibly do

$$\alpha^a \times \alpha^b \rightarrow \alpha^{ab} \tag{1}$$

Well, life is not as easy as it looks like...

$$\alpha^a \times \alpha^b = \alpha^{a+b} \tag{2}$$

Links between Discrete Logarithm and DiffieHellman Problem

Suppose that the discrete logarithm problem in $\mathbb{Z}_p^*$ could be efficiently solved[2]. Then given α, p, $\alpha^a \mod p$ and $\alpha^b \mod p$, one could first find a from α, p and $\alpha^a \mod p$ by what?!

Solving a discrete logarithm problem, and then compute $(\alpha^b)^a = \alpha^{ab} \mod p$.

[2]In math, the assumption is as important as,
if not more important than induction in many situations.

ElGamal public-key encryption

The ElGamal public-key encryption scheme can be viewed as Diffie–Hellman key agreement[3] in key transfer mode.
Its security is based on the intractability of the discrete logarithm problem (Section 1) and the DiffieHellman problem (Section 2).

[3] Yet another fancy nickname for key exchange

Ensure: A public key and its corresponding private key is created for every entity.

Steps to generate key pairs are described as follows:

1. Generate a prime p that is large enough and cannot be predicted, i.e. it should be generated randomly. Find a generator α of the multiplicative group $\mathbb{Z}_p^*$ of integers modulo p.
2. Randomly select an integer a satisfying $1 \leq a \leq p-2$. Then calculate $\alpha^a \mod p$.
3. The public key is returned as (p, α, α^a); The private key is returned as a.

Figure : **Algorithm** Key generation for ElGamal public-key encryption

Ensure: B uses A's public key to encrypt a message m. Then A uses decrypts using his private key.

1. *Encryption*. The steps for B to take are as follows:
    1. Require or obtain A's authentic public key (p, α, α^a).
    2. Express the plaintext message as an integer in the scale $0, 1, \ldots, p-1$.
    3. Randomly select an integer k satisfying $1 \leq k \leq p-2$.
    4. Calculate $\gamma = \alpha^k \mod p$ and $\delta = m \cdot (\alpha^a)^k \mod p$.
    5. Transmit the ciphertext $c = (\gamma, \delta)$ to A.
2. *Decryption*. The decrypt steps for A are described as follows:
    1. Calculate $\gamma^{p-1-a} \mod p$ using A's private key. (note: due to the characteristic of modulus, $\gamma^{p-1-a} = \gamma^{-a} = \alpha^{-ak}$).
    2. Calculate plaintext m by evaluating $(\gamma^{-a} \cdot \delta \mod p)$.

Figure : **Algorithm** ElGamal public-key encryption [Elg85]

ElGamal Encryption with artificially small parameters

Key generation. Entity A selects the prime $p = 2357$ and a generator $\alpha = 2$ of $\mathbb{Z}^*_{2357}$. A chooses the private key $a = 1751$ and computes

$$\alpha^a \mod p = 2^{1751} \mod 2357 = 1185 \tag{3}$$

A's public key is $(p = 2357, \alpha = 2, \alpha^a = 1185)$.
Encryption. To encrypt a message $m = 2035$, B selects a random integer $k = 1520$ and computes

$$\gamma = 2^{1520} \mod 2537 = 1430 \tag{4}$$

and

$$\delta = 2035 \cdot 1185^{1520} \mod 2357 = 697 \tag{5}$$

ElGamal Encryption cont.

B sends $\gamma = 1430$ and $\delta = 697$ to A.
Decryption. To decrypt, A computes

$$\gamma^{p-1-a} = 1430^{605} \mod 2357 = 872 \tag{6}$$

and recovers m by computing

$$m = 872 \cdot 697 \mod 2357 = 2305 \tag{7}$$

A clear disadvantage of ElGamal encryption is that there is a *message expansion* by a factor of 2. That is, the ciphertext is twice as long as the corresponding plaintext.

$$m = 2035 \Rightarrow (\gamma = 1430, \delta = 697) \tag{8}$$

Exhaustive Search

The most obvious algorithm for discrete logarithm problem is to successively compute α^0,α^1,α^2,... until β is obtained.

This method takes $O(n)$ multiplications, where n is the order of α, and is therefore inefficient if n is large (i.e. in cases of cryptographic interest).

Say we use 1024 bits, then the number should be maximum 2^{1024} as large.

2^{64}	**=**	**18,446,744,073,709,551,616**	**2^{80}**	**=**	**1,208,925,819,614,629,174,706,176**
2^{65}	=	36,893,488,147,419,103,232	2^{81}	=	2,417,851,639,229,258,349,412,352
2^{66}	=	73,786,976,294,838,206,464	2^{82}	=	4,835,703,278,458,516,698,824,704
2^{67}	=	147,573,952,589,676,412,928	2^{83}	=	9,671,406,556,917,033,397,649,408
2^{68}	**=**	**295,147,905,179,352,825,856**	**2^{84}**	**=**	**19,342,813,113,834,066,795,298,816**
2^{69}	=	590,295,810,358,705,651,712	2^{85}	=	38,685,626,227,668,133,590,597,632
2^{70}	=	1,180,591,620,717,411,303,424	2^{86}	=	77,371,252,455,336,267,181,195,264
2^{71}	=	2,361,183,241,434,822,606,848	2^{87}	=	154,742,504,910,672,534,362,390,528
2^{72}	**=**	**4,722,366,482,869,645,213,696**	**2^{88}**	**=**	**309,485,009,821,345,068,724,781,056**
2^{73}	=	9,444,732,965,739,290,427,392	2^{89}	=	618,970,019,642,690,137,449,562,112
2^{74}	=	18,889,465,931,478,580,854,784	2^{90}	=	1,237,940,039,285,380,274,899,124,224
2^{75}	=	37,778,931,862,957,161,709,568	2^{91}	=	2,475,880,078,570,760,549,798,248,448
2^{76}	**=**	**75,557,863,725,914,323,419,136**	**2^{92}**	**=**	**4,951,760,157,141,521,099,596,496,896**
2^{77}	=	151,115,727,451,828,646,838,272	2^{93}	=	9,903,520,314,283,042,199,192,993,792
2^{78}	=	302,231,454,903,657,293,676,544	2^{94}	=	19,807,040,628,566,084,398,385,987,584
2^{79}	=	604,462,909,807,314,587,353,088	2^{95}	=	39,614,081,257,132,168,796,771,975,168

Figure : Power of Two

Index Calculus

The index-calculus algorithm is the most powerful method known for computing discrete logarithms. The technique employed does not apply to all groups, but when it does (apply to specific groups), it often gives a subexponential-time algorithm.

The index-calculus algorithm requires the selection of a relatively small subset S of elements of G, called the *factor base*, in such a way that a significant fraction of elements of G can be effectively expressed as *products pf elements* from S.

Index Calculus Toy Example I

Let $p = 229$. The element $\alpha = 6$ is a generator of $\mathbb{Z}^*_{229}$ of order $n = 228$. Consider $\beta = 13$. Then $\log_6 13$ is computed as follows, using index-calculus technique.

1. The factor base is chosen to be the first 5 primes: $S = 2, 3, 5, 7, 11$.
2. The following six relations involving elements of the factor base are obtained (unsuccessful attemps are not shown):

$$6^{100} \mod 229 = 180 = 2^2 \cdot 3^2 \cdot 5$$
$$6^{18} \mod 229 = 176 = 2^4 \cdot 11$$
$$6^{12} \mod 229 = 176 = 3 \cdot 5 \cdot 11$$
$$\cdots$$

These relations yields the following equations involving the logarithms of the elements in the factor base:

$$
\begin{gathered}
100 \equiv 2\log_6 2 + 2\log_6 3 + \log_6 5 \mod 228 \\
18 \equiv 4\log_6 2 + 2\log_6 11 \mod 228 \\
12 \equiv \log_6 3 + \log_6 5 + \log_6 11 \mod 228 \\
\cdots
\end{gathered}
$$

Solving the linear system of six equations in five unknown (the logarithms $x_i = \log_6 p_i$) yields the solutions $\log_6 2 = 21$, $\log_6 3 = 208$, $log_6 5 = 98$, $\log_6 7 = 107$, and $\log_6 11 = 162$. All in modulus.

Suppose that the integer $k = 77$ is selected. Since $\beta \cdot \alpha^k = 13 \cdot 6^{77} \mod 229 = 147 = 3 \cdot 7^2$.

References I

- T. Elgamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Information Theory, IEEE Transactions on **31** (1985), no. 4, 469–472.
- Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot, *Handbook of applied cryptography*, 1st ed., CRC Press, Inc., Boca Raton, FL, USA, 1996.
- Kenneth H. Rosen, *Discrete mathematics and its applications*, 7th ed., McGraw-Hill Higher Education, 2012.