

Discrete Logarithm: ElGamal, Algorithms, Practice

Ilaria Battiston

Technical University Munich

i.battiston@tum.de

June 23, 2020

- 1 The Discrete Logarithm
 - Functioning and examples
 - Computational aspects and real life applications
- 2 The Diffie-Hellman key exchange
 - Protocol
 - Security
- 3 The ElGamal Cryotosystem
 - Protocol and proof
 - Computation and security
- 4 Algorithms to find Discrete Logarithm
 - Pollard Rho
 - Index Calculus
- 5 Comparison with integer factorization

The Discrete Logarithm

The discrete logarithm is a public key cryptosystem proposed in the 1970s taking advantage of the infeasibility of inverse logarithms within finite cyclic groups.

Discrete Logarithm

If G is a finite group, b is an element of G , and y is an element of G which is a power of b , then the discrete logarithm of y to the base b is any integer x such that $b^x = y$.

Infeasibility of the discrete logarithm

Having a finite cyclic group \mathbb{Z}_p^* of order $p - 1$, taking a primitive element $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$, the discrete logarithm problem aims to determine the integer $1 \leq x \leq p - 1$ such that:

$$\alpha^x \equiv \beta \pmod{p} \quad \rightarrow \quad x = \log_{\alpha} \beta \pmod{p}$$

Raising a number b to a power x in a large finite field is an one-way function, i.e. it is far more difficult to apply the inverse operation and finding $\log_b x$.

Given an element $y = b^x$, finding y is therefore an open problem.

Example

Considering a discrete logarithm in the group \mathbb{Z}_{47}^* , in which $\alpha = 5$ is a primitive element, the aim is finding the positive integer x such that $5^x \equiv 41 \pmod{47}$. A brute-force attack reveals that the result is $x = 15$.

Using prime cardinality is important in order to avoid potential attacks.

Influencing computation:

- Finite fields in which discrete logarithm is computed
- Length of involved operands
- Repeated exponentiation

Performance can be a problem creating serious bottleneck in devices with small CPUs.

Bit lengths	Security level
1024 bit	80 bit
3072 bit	128 bit
7680 bit	192 bit
15360 bit	256 bit

Table: Different security levels for discrete logarithm [Paar]

- Standardized by the Internet Key Exchange
- LAN security
- Authentication and confidentiality protection
- Diffie-Hellman key exchange

Which one of these is NOT an advantage of public key cryptography?

- A. Secure information through insecure channel
- B. Computational speed of encryption and decryption
- C. Robust authentication

The Diffie-Hellman key exchange

Diffie-Hellman is a schema used to exchange private keys through public key cryptography, based on **commutativity of exponentiation** within prime fields.

- Computationally faster operations since public key encryption is only performed once
- Secure identity establishment
- Third parties cannot retrieve original values
- Can be used by more than two parts

Diffie–Hellman Key Exchange

Alice

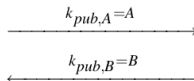
choose $a = k_{pr,A} \in \{2, \dots, p-2\}$

compute $A = k_{pub,A} \equiv \alpha^a \bmod p$

Bob

choose $b = k_{pr,B} \in \{2, \dots, p-2\}$

compute $B = k_{pub,B} \equiv \alpha^b \bmod p$



$$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$$

$$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$$

Figure: Key exchange between Alice and Bob [Paar]

$$(\alpha^a)^b = (\alpha^b)^a$$

Passive attacks

- Man-in-the-middle

Active attacks

- Discrete Log Computation

The discrete logarithm computation is the only available method to crack Diffie-Hellman, however this is infeasible among groups with large prime factors dividing their cardinality.

DHKE, thanks to its many advantages, is widely used in practice:

- SSH, providing security between client-server architectures
- TLS, encrypting communication between computer networks
- IpSec, establishing authentication among virtual private networks

Commonly used groups

Internet traffic mostly relies on commonly-known groups, involving primes with less than 1024 bits, to reduce computational complexity and increase interoperability.

Logjam attack

Logjam exploits usage of these groups to compute 512-bit keys discrete logarithms among commonly-used groups. After the initial phase of precomputation, keys could be cracked within minutes.

The ElGamal Cryptosystem

The ElGamal cryptosystem is an encryption scheme that can also be applied in general cyclic groups.

Its key aspect is the **randomized encryption**: Diffie-Hellman requires interaction of both parties to calculate a common private key.

This can constitute in a problem in case they are not able to interact, due to delays in transmission or unavailability of receiver.

The random exponent is therefore introduced to replace the private exponent of the receiving entity, so that it does not have to take part in the exchange.

The ElGamal Protocol

Elgamal Encryption Protocol

Alice

choose $i \in \{2, \dots, p-2\}$
compute ephemeral key
 $k_E \equiv \alpha^i \mod p$
compute masking key
 $k_M \equiv \beta^i \mod p$
encrypt message $x \in \mathbb{Z}_p^*$
 $y \equiv x \cdot k_M \mod p$

$k_{pub} = (p, \alpha, \beta)$



(k_E, y)



Bob

choose large prime p
choose primitive element $\alpha \in \mathbb{Z}_p^*$
or in a subgroup of \mathbb{Z}_p^*
choose $k_{pr} = d \in \{2, \dots, p-2\}$
compute $k_{pub} = \beta = \alpha^d \mod p$

compute masking key
 $k_M \equiv k_E^d \mod p$
decrypt $x \equiv y \cdot k_M^{-1} \mod p$

Figure: ElGamal key exchange between Alice and Bob [Paar]

$$y = x \cdot \alpha^{id} \quad k_M = \alpha^{di} \quad k_E = \alpha^i$$

$$\begin{aligned}
 d_{k_{pr}}(k_E, y) &\equiv y \cdot (k_M)^{-1} \pmod{p} \\
 &\equiv [x \cdot k_M] \cdot (k_E^d)^{-1} \pmod{p} \\
 &\equiv (x \cdot \alpha^{id}) \cdot (\alpha^{id})^{-1} \pmod{p} \\
 &\equiv [x \cdot (\alpha^d)^i]^{-1} \pmod{p} \\
 &\equiv x \cdot \alpha^{d \cdot i - d \cdot i} \pmod{p} \\
 &\equiv x \pmod{p}
 \end{aligned} \tag{1}$$

The ciphertext is twice as long as the message: therefore, the message expansion factor of ElGamal is two.

ElGamal is a **probabilistic encryption scheme**: encrypting two identical messages x_1, x_2 using the same public key results with *extremely high likelihood* in two different ciphertexts $y_1 \neq y_2$.

The random exponent i , however, **should not be reused**: the two masking keys would be the same, along with the ephemeral keys.

Two identical cyphertexts (y_1, k_E) are therefore sent through the channel. If an attacker can guess the first message, the second one can be computed as well using the same masking key.

ElGamal encryption is not widely used in practice, since one of the best known practices to break it exploits its malleability: the ciphertext (k_E, y) can be replaced with (k_E, sy) for some integer s .

The receiver therefore then computes:

$$\begin{aligned} d_{k_{pr}}(k_E, sy) &\equiv sy \cdot (k_M)^{-1} \pmod{p} \\ &\equiv s \cdot (x \cdot k_M) \cdot k_M^{-1} \pmod{p} \\ &\equiv sx \pmod{p} \end{aligned} \tag{2}$$

The decrypted text is also a multiple of s , and although an attacker would not be able to decrypt the message, he is able to manipulate it and compromise communication.

The ElGamal encryption system is used in GNU Privacy Guard System and recent versions of PGP, however the exponent has to be large enough or those systems will be subject to vulnerabilities.

Despite being easy to manipulate, the cryptography is secure. The only active ways an attacker can break the ElGamal scheme are:

- 1 Finding d , i.e. computing $d = \log_{\alpha} \beta \mod p$, which is computationally infeasible
- 2 Trying to guess the random exponent $i = \log_{\alpha} k \mod p$, which still involves solving the discrete logarithm problem

Which of these characteristics correspond to Diffie-Hellman, which ones to ElGamal and which ones to both?

- ① Used in TLS
- ② Based on discrete logarithm
- ③ Randomized exponentiation
- ④ Mainly used for key exchange
- ⑤ Probabilistic algorithm
- ⑥ Can be used by more than two parts

Theoretical bounds

Attacks against discrete logarithm involve finding the integer x for a given α and β in G such that:

$$\beta = \alpha \circ \alpha \circ \dots \circ \alpha = \alpha^x$$

A generic algorithm that solves *with high probability* the problem must perform at least $\Omega(p^{1/2})$ group operations, where p is the largest prime dividing the cardinality of the group.

In 2019 by a group of researchers who announced the computation of the discrete logarithm of a number composed by 795 bit. This was obtained using a 2.1GHz CPU and took approximately 4000 core-years.

Based on the **birthday paradox**: the basic idea consists in pseudo-randomly generate group elements of the form $\alpha^i \cdot \beta^j$, keeping track of values i and j , to then continue until obtaining a collision:

$$\alpha^{i_1} \cdot \beta^{j_1} = \alpha^{i_2} \cdot \beta^{j_2}$$

Substituting $\beta = \alpha^x$ and comparing the exponents on both sides of the equation, the collision leads to the following formula:

$$i_1 + xj_1 \equiv i_2 + xj_2 \pmod{|G|} \quad \rightarrow \quad x \equiv \frac{i_2 - i_1}{j_1 - j_2} \pmod{|G|}$$

Probabilistic algorithm taking advantage of high parallelism, one of the most efficient methods to find discrete logarithms within polynomial groups.

Here, the term **index** is a commonly used word to define the discrete logarithm: $x = \text{ind}(a) \bmod q$ for some base b , for $b^x \equiv a \bmod m$.

Index-calculus depends on the property that a significant fraction of elements of G can be efficiently expressed as **products of elements** of a small subset of G , assuming $|G| = q = p^n$ is a fairly large power of a small prime p .

Index-calculus: precomputation

An irreducible element $f \in \mathbb{Z}_q^*$ is identified, and a subset $B \subset \mathbb{Z}_q^*$ is chosen.

Then, a random integer $t \in [1, q-2]$ is chosen and then $b^t \in \mathbb{Z}_q^*$ is computed through repeated squaring. The algorithm finds all elements $c \in \mathbb{Z}_q^*$ such that:

$$c = b^t \pmod{f}$$

Now, the index-calculus algorithm attempts to write c as:

$$c = c_0 \prod_{a \in B} a^m$$

m is the highest power of a which divides c .

Index-calculus: precomputation

Now, supposing some $c \equiv b^t \pmod{f}$ has been found, and c has the desired type of factorization, taking the discrete logarithm of both sides allows to obtain:

$$\text{ind}(c) - \text{ind}(c_0) \equiv \sum_{a \in B} m \cdot \text{ind}(a) \pmod{q-1}$$

Repeating the process, eventually a large number of independent congruences will be found, written as:

$$t - \text{ind}(c_0) \equiv \sum_{a \in B} m \cdot \text{ind}(a) \pmod{q-1}$$

Index-calculus: computation

A random integer t is once again chosen, and $y = xb^t$ is computed, i.e. the unique element $y \in \mathbb{Z}_q^*$ satisfying $y \equiv xb^t \pmod{f}$.

As in the first stage, y has to be factored until an integer is obtained such that:

$$y = y_0 \prod_{a \in B} a^m$$

As soon as this happens, the algorithm can terminate:

$$\text{ind}(x) = \text{ind}(y) - t \quad \wedge \quad \text{ind}(y) = \text{ind}(y_0) + \sum_{a \in B} m \cdot \text{ind}(a)$$

This algorithm allows to obtain a runtime which is not exponential in the bit length of the order, but **subexponential**.

Despite the existence of efficient algorithms running on advanced hardware, the discrete logarithm is still considered among the NP-hard problems.

Index-calculus allows to compute discrete logarithms within groups with security level smaller than 80 bits.

1024-bit keys, however, are considered secure and unbreakable by current resources, although computational times of algorithms can be shortened by improving preprocessing and applying other mathematical properties to operations.

Comparison with integer factorization

- ① Both concern cases of the small subgroup problem for finite commutative groups
- ② Both are NP-hard problems, despite the existence of efficient algorithms on quantum computers
- ③ Both have been extensively used to construct strong encryption systems still valid today
- ④ Algorithms are similar due to modular arithmetic









Conclusion and open questions

- Are there better algorithms yet to be discovered?
- Is quantum computing an efficient way to reduce computational time?
- Would attacks on elliptic curves also influence discrete logarithm?

Having these open questions allow space for further research and development, while still ensuring computation infeasibility.

Thank you for your attention!

References

-  N. Koblitz - A Course in Number Theory and Cryptography
-  D. Knuth - The Art of Computer Programming, Vol. II
-  C. Paar, J. Pelzl - Understanding Cryptography
-  P. W. Shor - Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer
-  P. Horster, H. Knoblock - Discrete Logarithm Based Protocols
-  V. Shoup - Lower Bounds for Discrete Logarithms and Related Problems
-  T. Kleinjung, C. Diem, A. Lenstra, C. Priplata, C. Stahlke - Computation of a 768-bit prime field discrete logarithm
-  D. Adrian et al. - Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice
-  A. Meier - The ElGamal Cryptosystem