Diffie-Hellman key exchange & ElGamal public key cryptosystem

Report

Pedro Trindade Nº 41661

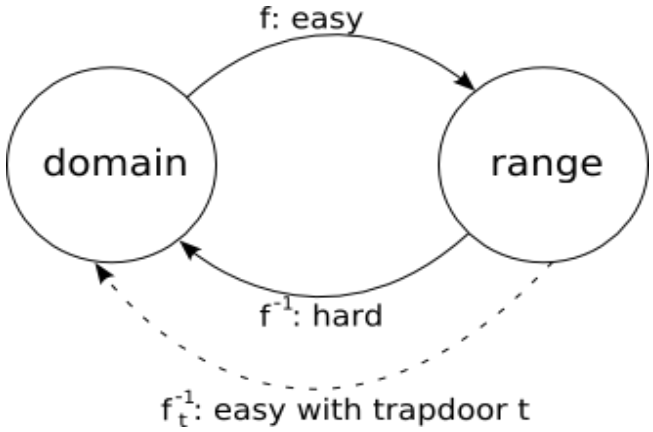
Paulo Martins Nº 41982

Patrick Cunha Nº 41668

The birth of public key cryptography

The concept of public key encryption was originally discovered by James Ellis while working at the GCHQ (*British Government Communications Headquarters*) in **1969**. Later, in **1974**, also while working at the GCHQ, Malcolm Williamson and Clifford Cocks discovered the Diffie-Hellman key exchange algorithm, before its rediscovery and public dissemination by Whitfield Diffie and Martin Hellman in **1976**, when they published their now famous paper "*New Directions in Cryptography*", making several groundbreaking contributions to this new field. Indeed, their paper begins with a call to arms: **"*We stand today on the brink of a revolution in cryptography.*"**

The first important contribution of Diffie and Hellman in their paper was the definition of a *Public Key Cryptosystem* (**PKC**) and its associated components: **one-way functions** and **trapdoor information**.



One-way functions

A **one-way function** is an invertible function that is easy to compute, but whose inverse is difficult to compute, meaning that if any algorithm used to do so in a reasonable amount of time (e.g., less than the age of the universe) will almost certainly fail. The strength of a one-way function is based on the time needed to reverse it: easy to reverse with small prime numbers, hard with big prime numbers.

Trapdoor information

The **trapdoor** is a piece of auxiliary information that allows the inverse of a one-way function to be easily computed, although there is a vast chasm separating the idea of a one-way trapdoor function and the actual construction of such a function.

Public Key Cryptosystem

The key consists of two pieces: a private key ($\mathbf{k_{priv}}$) and a public key ($\mathbf{k_{pub}}$) (computed by applying a key-creation algorithm to $\mathbf{k_{priv}}$). For each public/private key pair there is: an encryption algorithm ($\mathbf{ek_{pub}}$), which is public knowledge and easy to compute; and a corresponding decryption algorithm ($\mathbf{dk_{priv}}$), which must be easily computable by someone who knows the private key $\mathbf{k_{priv}}$, and very difficult to compute for someone who knows only the public key $\mathbf{k_{pub}}$.

The function used to create $\mathbf{k_{pub}}$ from $\mathbf{k_{priv}}$ must be difficult to invert, since $\mathbf{k_{pub}}$ is public knowledge and $\mathbf{k_{priv}}$ allows efficient decryption. The private key $\mathbf{k_{priv}}$ can be considered the ***trapdoor information*** to this ***one-way function*** $\mathbf{ek_{pub}}$, because without the ***trapdoor information***, it is very hard to compute the inverse function to $\mathbf{ek_{pub}}$, but with it it's easy to compute the inverse.

Despite years of research it is still not know whether one-way functions really exist. Diffie and Hellman made several suggestions in their paper for one-way functions, but did not produce an example of a **PKC**, mainly for the lack of finding the right trapdoor information. They did, however, describe a public key method (**Diffie-Hellman key exchange**) based on the assumption that the **discrete logarithm problem** is difficult to solve, by which certain material could be securely shared over an insecure channel.

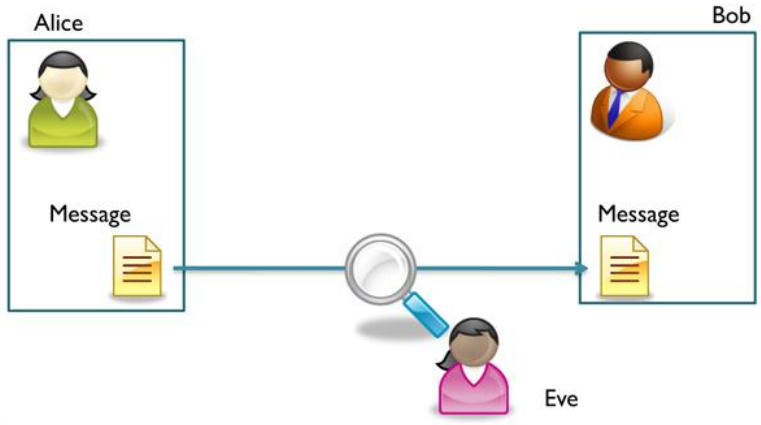
Discrete Logarithm Problem

The construction of the public key in the **Diffie-Hellman key exchange** method is based on the **discrete logarithm problem** in a finite field F_p with a prime number of elements.

Definition. Let g be a primitive root for ***Fp*** and let h be a nonzero element of ***Fp***. *The Discrete Logarithm Problem* (**DLP**) is the problem of finding an exponent x such that: $\boldsymbol{g^x \equiv h \pmod p}$.

Diffie-Hellman key exchange

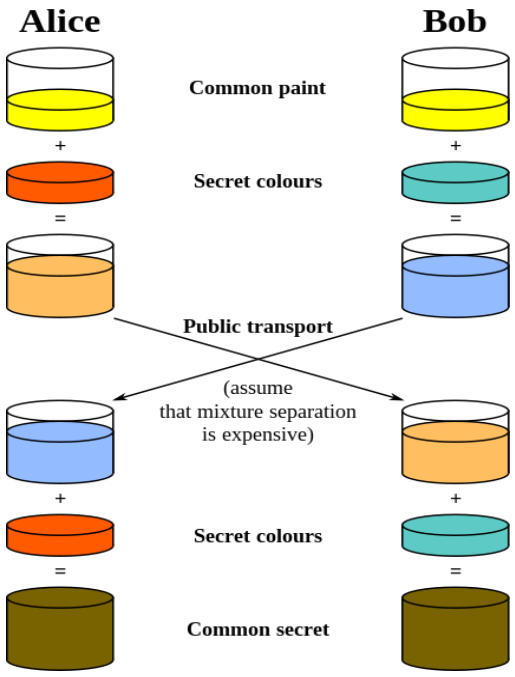
The **Diffie-Hellman key exchange** is a method of securely exchanging cryptographic keys over an insecure communication channel, relying on the difficulty of the **discrete logarithm problem**. It is a non-authenticated key-agreement protocol but provides the basis for a variety of authentication protocols such as the *Station-to-Station* **(STS)** protocol.



Eve is a passive attacker.

How does it work?

First we are going study a simplified example of the **Diffie-Hellman key exchange** method using colors as keys, instead of large prime numbers:



If another party (usually named **Eve** in cryptology publications, **Eve** being a third-party who is considered to be an eavesdropper) had been listening in on the exchange, it would be computationally difficult for that person to determine the common secret color; in fact, when using large numbers rather than colors, this action is impossible for modern supercomputers to do in a reasonable amount of time.

So we should assume that the communication channel between Alice and Bob is always insecure and every piece of information that they exchange is observed by Eve. Then how is it possible for Alice and Bob to share a key without making it available for Eve?

Step 1. The first step is for Alice and Bob to agree on a large prime number ***p*** and a nonzero integer ***g*** (generator) modulo ***p***, making their values public knowledge (Eve sees ***p*** and ***g***). These two values are chosen this way to ensure that the resulting shared key can take on any value from 1 to ***p*** – 1.

Step 2. The next step is for Alice to pick a secret integer ***a*** (Alice's private key) that she does not reveal to anyone, while at the same time Bob picks an integer **b** (Bob's private key) that he keeps secret. Then they use their secret integers to compute:

$$A \equiv g^a \pmod{p}$$

$$B \equiv g^b \pmod{p}$$

Step 3. They next exchange these computed values, Alice sends ***A*** to Bob and Bob sends ***B*** to Alice (Eve sees ***A*** and ***B***).

Step 4. Finally, they use their secret keys again to compute:

$$A' \equiv B^a \pmod{p}$$

$$B' \equiv A^b \pmod{p}$$

These computed values are actually the same since:

$$A' \equiv B^a \equiv (g^b)^a \equiv (g^a)^b \equiv A^b \equiv B' \pmod{p}$$

This common value is their shared key, which they can now use to send messages across the same open communication channel.

Example. Now we are going to illustrate an example of the public key exchange method using actual numbers:

Step 1.

p = 941

g = 627

Eve sees ***941*** and ***627***

Step 2.

a = 347

$A \equiv 390 \equiv 627^{347} \pmod{941}$

$b = 781$

$B \equiv 691 \equiv 627^{781} \pmod{941}$

Step 3.

Alice sends ***390*** to Bob

Bob sends ***691*** to Alice

Eve sees ***390*** and ***691***

Step 4.

Alice computes the shared key $691^{347} \pmod{941}$, which is **470**.

Bob computes the shared key $390^{781} \pmod{941}$, which is **470**.

Eve's dilemma

In the previous example, Eve can reconstitute Alice and Bob's shared key if she can solve either: $627^{a} \equiv 390 \pmod{941}$ or $627^{b} \equiv 691 \pmod{941}$, since then she will know one of their secret keys.

If Eve can solve the **DLP**, then she can compute the secret keys ***a*** and/or ***b*** from the intercepted values of ***A*** and ***B***, making it easy for her to compute the shared key.

Diffie-Hellman Problem

Solving the **DLP** is one method of finding the shared key, but that's not the precise problem that Eve needs to solve. There is another problem called *Diffie-Hellman Problem* (**DHP**), no harder than the **DLP** that Eve could solve to find the shared key.

Definition. Let ***p*** be a prime number and ***g*** an integer. The *Diffie-Hellman Problem* (**DHP**) is the problem of computing the value of $g^{ab} \pmod{p}$ from the known values of $g^{a} \pmod{p}$ and $g^{b} \pmod{p}$.

Security

The key exchange method is secure if the values are chosen properly and with adequate lengths: the recommended size for the modulus is **2048 bits** and the recommended size for the private key is **256 bits**.

The key exchange by itself **does not provide authentication**, but it can be used to achieve **forward secrecy**, being a frequent choice for such protocols, because of its **fast key generation**.

Forward secrecy is a property of secure communication protocols in which compromise of long-term keys does not compromise past session keys. **Forward secrecy** protects past sessions against future compromises of secret keys or passwords. If **forward secrecy** is utilized, encrypted communications and sessions recorded in the past cannot be retrieved and decrypted should long-term secret keys or passwords be compromised in the future, even if the adversary actively interfered.

Implementation details

Prime numbers can be generated. “Safe primes” from already established standards should be used (**RFC 3526**). A safe prime is a prime number of the form $2\boldsymbol{p} + 1$, where $\boldsymbol{p}$ is also a prime.

For the primitive roots if the prime number was generated, find a primitive root of $F_{\boldsymbol{p}}$. Use the generator associated with the prime from already established standards, the most common generator being 2.

The secret keys must be randomly generated and have an appropriate length (the sizes are also defined in standards).

Attacks

The Diffie–Hellman public key exchange method is used to secure a variety of Internet services, however, research published in **October 2015** suggests that the parameters in use for many D-H Internet applications at that time are not strong enough to prevent compromise by very well-funded attackers, such as the security services of large governments.

We will now see the most common attacks to the Diffie-Hellman public key exchange method.

Pohlig-Hellman algorithm

The algorithm was discovered by Roland Silver, but first published by Stephen Pohlig and Martin Hellman. It is a special-purpose algorithm for computing discrete logarithms in a multiplicative group whose order is a smooth integer.

Input: Integers ***p***, ***g***, ***e***.

Output: An Integer ***x***, such that ***e*** ≡ g^x (**mod** ***p***) (if one exists).

General number field sieve

The general number field sieve (**GNFS**) is the most efficient classical algorithm known for factoring integers larger than 10^{100}. If you try factoring with brute force the complexity is $O(2^{\frac{b}{2}})$. Using **GNFS** the complexity is exponential in $b^{\frac{1}{3}} \log b^{\frac{2}{3}}$ in this form:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}} + o(1)\right)(\ln n)^{\frac{1}{3}}(\ln \ln n)^{\frac{2}{3}}\right)$$

Man-in-the-middle

Since there's no authentication mechanism in the Diffie-Hellman key exchange method, Eve can become an active attacker and impersonate Alice, Bob or both. This attack can be prevented with the use of digital signatures.

Logjam

Due to the ever increasing computational power, it becomes feasible to break implementations that use primes with an inadequate size. The Logjam attack allows a man-in-the-middle attacker to downgrade vulnerable TLS connections to 512-bit export-grade cryptography. This allows the attacker to read and modify any data passed over the connection. Are we safe with 1024 bit primes?

Prime Length	Could Be Broken By	Precomputation Time
512 bits	Academics	7 days
768 bits	Academics	~1 month
1024 bits	Nation State Large Organization	~1 year ~$100-300 million

The NSA can use specialized hardware to speed up the process of breaking DH.

Applications

The Diffie-Hellman key exchange method is mainly used to provide **PFS** in **TLS** ephemeral modes, for **SSH** and **IPSec**.

ElGamal public key cryptosystem

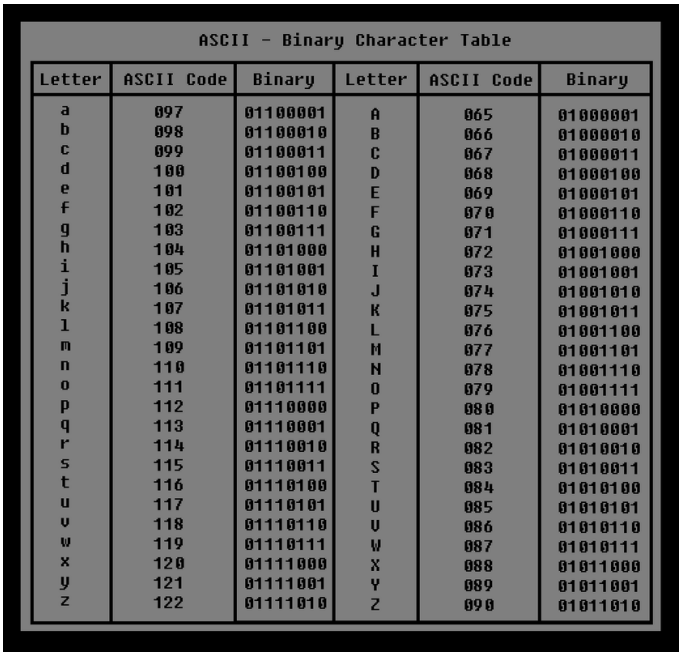
The Diffie-Hellman key exchange algorithm does not achieve the full goal of being a **public key cryptosystem**, since a cryptosystem permits the exchange of specific information, not just a random string of bits.

Although the first public key cryptosystem was the **RSA**, the one most related to the Diffie-Hellman key exchange method is the **ElGamal public key cryptosystem**, proposed by Taher ElGamal in **1985**, which is also based on the **discrete logarithm problem**.

With the use of an **encoding scheme**, a plaintext or ciphertext message may be viewed as a sequence of bits (usually blocks of 8 bits), thus providing a method of exchanging specific information through any communication channel.

Encoding schemes

An encoding scheme is a method of converting one sort of data into another sort of data (e.g. text into numbers). Both the encoding and decoding functions are **public knowledge** and **easy to compute**. The most commonly used encoding scheme is the **ASCII** code.



ASCII - Binary Character Table

Letter	ASCII Code	Binary	Letter	ASCII Code	Binary
a	097	01100001	A	065	01000001
b	098	01100010	B	066	01000010
c	099	01100011	C	067	01000011
d	100	01100100	D	068	01000100
e	101	01100101	E	069	01000101
f	102	01100110	F	070	01000110
g	103	01100111	G	071	01000111
h	104	01101000	H	072	01001000
i	105	01101001	I	073	01001001
j	106	01101010	J	074	01001010
k	107	01101011	K	075	01001011
l	108	01101100	L	076	01001100
m	109	01101101	M	077	01001101
n	110	01101110	N	078	01001110
o	111	01101111	O	079	01001111
p	112	01110000	P	080	01010000
q	113	01110001	Q	081	01010001
r	114	01110010	R	082	01010010
s	115	01110011	S	083	01010011
t	116	01110100	T	084	01010100
u	117	01110101	U	085	01010101
v	118	01110110	V	086	01010110
w	119	01110111	W	087	01010111
x	120	01111000	X	088	01011000
y	121	01111001	Y	089	01011001
z	122	01111010	Z	090	01011010

ElGamal public key cryptosystem

Step 1. Alice begins by selecting a large prime number ***p*** for which the **DLP** is difficult and an element ***g*** modulo ***p*** of large order, making their values public knowledge.

Step 2. Then Alice chooses a secret number ***a*** to act as her private key ($1 \leq a \leq p - 1$) and she computes:

$$A \equiv g^a \pmod{p}$$

This value is then published and Eve sees it.

Step 3. Now suppose that Bob wants to encrypt a message using Alice's public key ***A***, assuming that Bob's message ***m*** is an integer between 2 and ***p***. In order to encrypt ***m***, Bob first randomly chooses another number ***k*** modulo ***p***. Bob uses ***k*** to encrypt one, and only one, message and then he discards it. The number ***k*** is called an ephemeral key, since it exists for the purposes of encrypting a single message. Bob then takes his plaintext message ***m***, his chosen random ephemeral key ***k***, and Alice's public key ***A*** and uses them to compute the two quantities:

$$c_1 \equiv g^k \pmod{p}$$

$$c_2 \equiv m * A^k \pmod{p}$$

Bob's ciphertext (his encryption of ***m***) is the pair of numbers (c_1, c_2), which he then sends to Alice.

Step 4. Finally, to decrypt Bob's ciphertext, Alice computes:

$$s \equiv c_1^a \pmod{p}$$

She also gets $s^{-1} \pmod p$. Then she multiplies c_2 by s^{-1} and the resulting value is the plaintext m, since:

$s^{-1} * c_2 \equiv (c_1^a)^{-1} * c_2 \pmod p$, since $s \equiv c_1^a \pmod p$,

$\equiv (g^{ak})^{-1} * (mA^k) \pmod p$, since $c_1 \equiv g^k$, $c_2 \equiv mA^k \pmod p$,

$\equiv (g^{ak})^{-1} * (m(g^a)^k) \pmod p$, since $A \equiv g^a \pmod p$,

$\equiv m \pmod p$, since the g^{ak} terms cancel out.

And thus, she decrypted Bob's message, recovering the plaintext m.

Example. Now we are going to illustrate an example of the public key cryptosystem using actual numbers:

Step 1.

$p = 467$

$g = 2$

Eve sees 467 and 2

Step 2.

$a = 153$

$224 \equiv 2^{153} \pmod{467}$

Eve sees 224

$A = 224$

Step 3.

$m = 331$

$k = 197$

$c_1 \equiv 87 \equiv 2^{197} \pmod{467}$

$c_2 \equiv 57 \equiv 331 * 224^{197} \pmod{467}$

Bob sends c_1 and c_2 to Alice

Eve sees 87 and 57

Step 4.

Alice computes

$s \equiv 367 \equiv 87^{153} \pmod{467}$

$s^{-1} \equiv 367x \equiv 1 \pmod{467} = 14$

$m \equiv 331 \equiv 57 * 14 \pmod{467}$

Efficiency

In the ElGamal cryptosystem the plaintext message ***m*** is an integer between 2 and ***p*** – 1. The ciphertext integers c_1 and c_2 are in the same range.

ElGamal has a **2-to-1 message expansion** because, in general, it takes **twice** as many bits to write down the ciphertext compared to the plaintext. It also has ***homomorphic properties***, since it allows computations to be carried out on ciphertext, thus generating an encrypted result which, when decrypted, matches the result of operations performed on the plaintext.

Security

As the Diffie-Hellman key exchange algorithm, the ElGamal public key cryptosystem relies on the fact that the **Diffie-Hellman Problem** and the **discrete logarithm problem** are hard to solve. In conclusion, the ElGamal cryptosystem is at least as hard to attack as its underlying problem.

Chosen ciphertext attack

An attack in which Eve has access to an **oracle** that decrypts arbitrary ciphertexts. The ElGamal cryptosystem is proven to be secure against chosen ciphertext attacks if one assumes that the **Diffie-Hellman Problem** is hard.