Discrete Log based Cryptosystems

Gerardo Pelosi

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
Politecnico di Milano

gerardo.pelosi - at - polimi.it

Overview

Lesson contents

- Discrete Logarithm Problem
- Diffie-Hellmann Protocol
- ElGamal Cryptosystem
- ElGamal Signature scheme
- Digital Signature Standard Algorithm (DSS-DSA)

Discrete Logarithm Problem (1)

Generalized Discrete Logarithm Problem (GDLP)

Let $(G, \cdot)$ be a cyclic group (written multiplicatively) with order $n = |G|$, where $g \in G$ is one of its generators: $G = \langle g \rangle$. The GDLP for the group G is stated as follows:

Given $a \in \langle g \rangle$, find the smallest positive integer $x \in \mathbb{Z}_n$ such that $g^x = a$

Such an integer is the discrete logarithm of a to the base g, and we shall use the notation

$$x = \log_g^{\mathrm{D}} a \;\; \text{or} \;\; x = \mathrm{ind}_g\, a$$

Note that, the familiar formulas for ordinary logarithms remains valid:

- Given $a, b \in G$, then

$$\log_g^{\mathrm{D}}(a \cdot b) = \log_g^{\mathrm{D}}(a) + \log_g^{\mathrm{D}}(b) \bmod n$$

- Given $a \in G$, If g_1 is another generator of G, then:

$$\log_g^{\mathrm{D}}(a) \cdot \log_{g_1}^{\mathrm{D}}(g) = \log_{g_1}^{\mathrm{D}}(a) \bmod n$$

Discrete Logarithm Problem (2)

In cryptography it is crucial to select a cyclic group G where the GDLP is a "computationally hard" problem.
For instance, consider the group $(\mathbb{Z}_{19}, +)$ with $n = |G| = 19$.

- Let's assume $G = \langle g \rangle$, with $g = 2$. (Obs.: every element except 0 is a generator because $n = 19$ is prime!)
- Given a generic element $a \in G$, say $a = 15$, the computation of the discrete logarithm of a to the base g means that we have to find the smallest integer s.t. $x \cdot g = a$ which means that:

$$x \cdot 2 = 15 \bmod 19$$

  The unknown x can be easily found (in polynomial time) through the Euclid's gcd Algorithm:

$$x = \log_g^{\mathrm{D}} a = 2^{-1} \cdot 15 \bmod 19 = 10 \cdot 15 \bmod 19 = 17 \bmod 19$$

Therefore, the cyclic subgroups of $(\mathbb{Z}, +)$ are not good candidates to build a cryptosystem!

Discrete Logarithm Problem (3)

The cyclic groups where there is no polynomial time algorithm to extract the discrete logarithm of an element to any generator are:

- A multiplicative **subgroup** $\langle g \rangle$ of a generic Finite Field $(\mathbb{F}^*_{p^m}, \cdot)$, where $m \geq 1$, p a prime number, $p^m \geq 2^{1024}$, and the order $n{=}|\langle g \rangle|$ of the subgroup is also a prime with $n \geq 2^{160}$
  - if n is a composite integer, the DLP in such a group can be reduced to the DLPs in each of its (smaller) prime subgroups through applying the so-called "Pohlig-Hellman Algorithm" (. . . we'll see the details in the next lecture)
- Additive subgroups $(G, +)$ of elliptic curve points defined over a finite field, with prime order $n \geq 2^{160}$: $G = \mathbb{E}(\mathbb{F}_{p^m})[n]$ (with a proper definition of the group law. . .)

Discrete Logarithm Problem (4)

Currently, the best known algorithms to solve the DLP in the multiplicative subgroups of Finite fields or in the additive subgroups of points of an elliptic curve are adapted from the best methods designed to factor a composite integer.

- The computational complexity of the GNFS adapted to find a discrete log in a prime multiplicative subgroup G included in a finite field is sub-exponential: $\mathcal{O}(L_{|G|}(\alpha, \beta))$, $0 < \alpha < 1$.
- The computational complexity of the GNFS adapted to find a discrete log in a prime subgroup G of a properly chosen elliptic curve is exponential: $\mathcal{O}(L_{|G|}(1, \beta))$.

This will lead to the fact that Elliptic curve cryptosystems (ECC) can employ public/private key pairs shorter than other dicrete log systems without reducing the security margin!

Example (1)

Given $G = (\mathbb{Z}^*_{11}, \cdot)$, with $|G|{=}\varphi(11){=}10$, to formulate a correct GDLP we need:

- to find a generator g of the group G, assuming to know the factorization of the order $n = |G| = \prod_i^s p_i^{e_i}$, $e_i \geq 1$, $s \geq 1$
- to find a proper prime subgroup H, with $p_2 = |H|$

We know the factorization of the order $|G| = 10 = 5 \cdot 2$ thus, we can find a generator of G through the usual basic procedure ($p_1 = 5$, $p_2 = 2$)

Input: $|G| = p_1^{e_1} \cdot \dots \cdot p_s^{e_s}$, $\forall e_i \geq 1$
Output: g, generator of G

```
begin
    while true do
        g <-random- {1, ..., |G| - 1}
        if g^{|G|/p_1} ≠ 1 AND ... AND g^{|G|/p_s} ≠ 1 then
            return g
```

$g^{10/5} \equiv_{11} 2^2 \equiv_{11} 4 \neq 1$, $g^{10/2} \equiv_{11} 2^5 \equiv_{11} 2 \neq 1$ thus, $g = 2$ is a generator.

The prime subgroup $H{=}\langle h \rangle$, is generated by $h = g^{|G|/p_1} {\equiv_{11}} 4$

Example (2)

Given $G = (\mathbb{F}^*_{2^5}, \cdot)$, with generating polynomial $f(x) = x^5 + x^2 + 1 \in \mathbb{F}_2[x]$, find a generator $g \in G$

We note that $n = |G| = 31$ is a prime number, thus every element (except for the neutral element) is a generator.

In particular, representing the field $\mathbb{F}_{2^5}$ as a simple algebraic extension:

$$\mathbb{F}_{2^5} \cong \mathbb{F}_2(\alpha) = \{b_4\alpha^4 + b_3\alpha^3 + b_2\alpha^2 + b_1\alpha + b_0 | b_i \in \mathbb{F}_2, f(\alpha) = 0, \alpha \in \mathbb{F}_{2^5} \backslash \mathbb{F}_2\}$$

for the sake of simplicity, we can pick as generator the element ($\neq 0, 1$) with the least number of coefficients:

$$g = (00010) = 0\,\alpha^4 + 0\,\alpha^3 + 0\,\alpha^2 + 1\,\alpha + 0\,\alpha^0 = \alpha$$

Discrete Log Cryptosystems (1)

Given a cyclic finite group $(G, \cdot)$ with generator $g \in G$ and order $n = |G|$, the common Discrete Log Cryptosystems are classified according to three computational problems:

- The (already defined) Discrete Log Problem (DLP)
  - Given $a \in G$,
    compute $x \in \mathbb{Z}_n$ s.t. $a = g^x$
- The Diffie-Hellman Problem (DHP)
  - Given $a = g^x, b = g^y \in G$, for some unknown $x, y \in \mathbb{Z}_n$,
    compute $c \in G$ s.t. $c = g^{x\,y}$
- The Decisional Diffie-Hellman Problem (DDHP)
  - Given $a = g^x, b = g^y, c = g^z \in G$, for some unknown $x, y, z \in \mathbb{Z}_n$,
    establish whether $z \equiv x\,y \bmod n$, or not

Discrete Log Cryptosystems (2)

DHP: Given $a{=}g^x, b{=}g^y{\in}G$, for some $x, y{\in}\mathbb{Z}_n$, find $c{\in}G$ s.t. $c{=}g^{x\,y}$

Clearly, if we could find x from g^x, we could solve DHP through a single exponentiation ($c = b^x = (g^y)^x$), so

Lemma 1

The DHP problem is no harder than the DLP problem.

Fact

- Solving DLP is a sufficient requirement to solve DHP.
  However, It is not known if this is also a necessary condition for a generic finite cyclic group.
- Only, in some particular groups (a.k.a. Gap-groups), the DLP and DHP has been formally proven to be equivalent.

Discrete Log Cryptosystems (3)

DDHP: Given $a = g^x, b = g^y, c = g^z \in G$, for some $x, y, z \in \mathbb{Z}_n$, establish whether $z \equiv x\,y \bmod n$, or not

Clearly, if we could compute $\gamma =$DHP(a,b), we could solve DDHP through checking whether $\gamma = c$, or not

Lemma 2

The **Decisional DHP** is no harder than the DHP

Fact

- Solving DHP is a sufficient requirement to solve DDHP
  It is NOT clear that this would be necessary, in general
- Only, for trivial groups we can show that DDHP is equivalent to DHP
- There are groups where the DDHP is known to be easy (... if you have a *pairing map* similar to the ones employed in Identity based cryptosystems...), but the DHP is conjectured to be hard!

Discrete Log Cryptosystems (4)

- In general, the facts known for certain for every possible group are:

  DDHP **is no harder than** DHP **is no harder than** DLP

- No results are known in general for any other possible relation among these problems!
- In multiplicative groups of (non trivial) finite fields and in cyclic groups of elliptic curve points DDHP is assumed to be hard.

Diffie-Hellman Key Exchange

Diffie and Hellman (1976): New directions in cryptography.

- This protocol allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel
- The shared secret key can then be used to encrypt subsequent communications using a symmetric key cipher

Diffie-Hellman Key Exchange

Public parameters publicly known are:
a cyclic group $(G = \langle g \rangle, \cdot)$ and its order $n = |G|$.

A

Private ephemeral key:
$k_{\mathrm{priv,A}} \overset{Random}{\leftarrow} \mathbb{Z}_n \backslash \{0, 1\}$

Public ephemeral key:
$k_{\mathrm{pub,A}} \leftarrow g^{k_{\mathrm{priv,A}}}$

Send $k_{\mathrm{pub,A}}$ to B

Compute the shared session key:
$k_{\mathrm{A,B}} = k_{\mathrm{pub,B}}^{k_{\mathrm{priv,A}}} = g^{k_{\mathrm{priv,A}}\, k_{\mathrm{priv,B}}}$

B

Private ephemeral key:
$k_{\mathrm{priv,B}} \overset{Random}{\leftarrow} \mathbb{Z}_n \backslash \{0, 1\}$

Public ephemeral key:
$k_{\mathrm{pub,B}} \leftarrow g^{k_{\mathrm{priv,B}}}$

Send $k_{\mathrm{pub,B}}$ to A

Compute the shared session key:
$k_{\mathrm{A,B}} = k_{\mathrm{pub,A}}^{k_{\mathrm{priv,B}}} = g^{k_{\mathrm{priv,A}}\, k_{\mathrm{priv,B}}}$

Diffie-Hellman Key Exchange

- Security against passive adversaries (that can only eavesdrop msgs over the communication channel) is based on the DHP which is no harder than the DLP
- Active adversaries who can put themselves in the middle of the communication channel, can easily masquerade to one party as its rightful counterpart without being noticed
(Man-in-the-middle attack (MiTM))
    - This problem arises because the DH protocol performs a non-authenticated key exchange
    - the transmission of each DH public ephemeral key should be managed employing a public key cryptosystem so to digitally sign the msgs

Why we should use the Diffie-Hellman protocol instead of exchanging a "shared secret" (or password) via public-key encryption?

Key Exchange

Forward Secrecy

A system is said to have **forward secrecy**, if the compromise of a long-term private key (at some point in the future) does not compromise the security of communications made using that key in the past.

Transmitting a password (or the symmetric-key of a block cipher) via public-key encryption does not have forward secrecy.

- Suppose you bulk encrypt a video stream and then encrypt the session key under the recipient's RSA public-key.
- Then suppose that some time in the future, the recipient's RSA private-key is compromised.
- At that point your video stream is also compromised, assuming the attacker copied all data (i.e., the video stream and the encrypted password) at the time they were transmitted!

Key Exchange

- In addition, the use of public-key encryption for sending either a password or a symmetric-key implies that the recipient trusts the sender to generate such a *shared secret*
- Sometimes the recipient may wish to contribute some (fresh) randomness of his own to the shared secret employed as session key
  - This can only be done if both parties are online at the same moment in time

- The transmission of the symmetric-key from sender to receiver, via PKC, is more suited to the case of a key-exchange where only the sender is online. F.i., in an e-mail application.
- The DH protocol is more suited to the case of a key-exchange between on-line parties and it also provides forward secrecy!

Diffie-Hellman Protocol

Many practical implementations of the DH protocol employ a cyclic subgroup G of $(\mathbb{Z}_p^*, \cdot)$ where the prime integer p is generated in such a way that the order of $p_1 = |G|$ is also a sufficiently large prime.

- Usually, two prime numbers p_1, p_2 are generated in such a way that $p_1 \geq 2^{160}$, and $p = 2\,p_1\,p_2 + 1$ is also prime, with $p \geq 2^{1024}$
- given a generator $\alpha \in \mathbb{Z}_p^*$, a generator of the subgroup G is computed as $g = \alpha^{\frac{p-1}{p_1}}$

Example

$p_1 = 5,\ p_2 = 3,\ p = 2\,p_1\,p_2 + 1 = 31$ is a prime number!
$(Z_p^*, \cdot)$; $|Z_p^*| = p - 1 = 30 = 2\,p_1\,p_2$, it is easy to prove that $\alpha = 3$ is a generator.
Therefore, if we want to find a generator of the subgroup G with order $p_1 = 5$, $g = 3^{30/5} \equiv_{31} 16$.
Hence, assuming a pair of DH ephemeral keys as:
$k_{priv,A}$=a=2 and $k_{pub,A} = g^a = 16^2 \bmod 31 \equiv_{31} 8$
$k_{priv,B}$=b=4 and $k_{pub,B} = g^b = 16^4 \bmod 31 \equiv_{31} 2$
the shared secret key is: $k_{BA} = 2^2 \bmod 31 = k_{AB} = 8^4 \bmod 31 = 4 \bmod 31$

ElGamal Cryptosystem

ElGamal (1985): A public key cryptosystem and a signature scheme based on discrete logarithms

- the assumption is that there is a publicly known cyclic group $(G, \cdot)$ with order $n = |G|$ (possibly prime) and a generator $g \in G$, where the DDHP, DHP, DLP are recognized to be computationally hard.
- it did not achieve the same diffusion of RSA, because:
  - the ctx provided as a result of an ElGamal encryption transformation has twice the size of the original ptx
  - the computational performances are slightly worse than the RSA ones $(\times 2)$ when $G = (\mathbb{F}_{p^m}, \cdot)$

ElGamal Cryptosystem

Given the cyclic group $(G, \cdot)$ with order $n = |G|$ (possibly prime) and a generator $g \in G$

Public key: $k_{\text{pub}} \leftarrow \langle\ n,\ g,\ g^s\ \rangle$ Private key: $k_{\text{priv}} \leftarrow \langle\ s \in \mathbb{Z}_n\ \rangle$

Encryption Transformation

$m \in G$

$l \stackrel{\text{Random}}{\leftarrow} \mathbb{Z}_n^*$

$\gamma \leftarrow (g)^l$

$\delta \leftarrow m \cdot (g^s)^l$

$c \leftarrow Enc_{k_{\text{pub}}}(m) = \langle \gamma, \delta \rangle$

Decryption Transformation

$c = \langle \gamma, \delta \rangle$

$m \leftarrow Dec_{k_{\text{priv}}}(c) = \gamma^{n-s} \cdot \delta$

Correctness verification:

$Dec_{k_{\text{priv}}}(Enc_{k_{\text{pub}}}(m)) = m \quad \forall\, m \in G$

$\gamma^{n-s} \cdot \delta = (g^l)^{-s} \cdot m \cdot (g^s)^l = m \cdot g^{0 \bmod n} = m$

Example

Given $G = \mathbb{F}_{31}^*$, with generator $g = 3$ and $n = |G| = 30$

a party A wishes to send the message $m = 12 \in G$ to B,
knowing that $k_{\mathrm{pub,B}} \leftarrow \langle n,\, g,\, g^s \rangle = \langle 30,\, 3,\, 26 \rangle$

Assuming $k_{\mathrm{priv,B}} \leftarrow \langle s \in \mathbb{Z}_n \rangle = \langle 5 \rangle$,
simulate the ElGamal encryption/decryption procedures

A

$l \overset{\mathrm{Random}}{\longleftarrow} \mathbb{Z}_{30}^*,\ l = 3$

$\gamma \leftarrow (g)^l = 3^3 \bmod 31 \equiv_{31} 27$
$\delta \leftarrow m \cdot (g^s)^l = 12 \cdot (26)^3 \bmod 31 \equiv_{31} 19$
$c \leftarrow Enc_{k_{\mathrm{pub}}}(m) = \langle \gamma, \delta \rangle = \langle 27, 19 \rangle$

B

$$m \leftarrow Dec_{k_{\mathrm{priv}}}(c) = \gamma^{n-s} \cdot \delta = 27^{25} \cdot 19 \bmod 31 \equiv_{31} 27^{(11001)_2} \cdot 19 \equiv_{31}$$
$$\equiv_{31} (((27^2\, 27)^2)^2)^2\, 27\, 19 \equiv_{31} \ldots \equiv_{31} 12$$

Security of the ElGamal Cryptosystem

Lemma

The Diffie–Hellman problem (DHP) problem is equivalent to the ElGamal problem

Lemma

Assuming the Diffie–Hellman problem (DHP) is hard, then ElGamal is secure under a chosen plaintext attack (CPA)

- where security means it is hard for the adversary, given the ciphertext, to recover the whole of the plaintext

Security of the ElGamal Cryptosystem

Lemma

ElGamal is **not secure** against a Adaptively Chosen Ciphertext Attack (CCA2)

Proof.

Suppose the message an eavesdropper wants to break is

$$c = (\gamma, \delta) = (g^l, m(g^s)^l)$$

she creates the related msg:

$$c' = (\gamma, 2\delta)$$

and asks the decryption oracle to decrypt c'. Thus, she obtains m'.
Then she computes $\frac{m'}{2} = \frac{2\delta\gamma^{-s}}{2} = \frac{2mg^{sl}g^{-sl}}{2} = m$ □

In practice a "modified version" of the ElGamal Cryptosystem is actually implemented in any practical scenario where this is scheme is adopted.

Security of the ElGamal Cryptosystem

Fujisaki and Okamoto in 1999 showed how to turn the scheme of a generic PKC into an encryption scheme which is semantically secure against adaptive adversaries in the random oracle model (... it works by showing that the resulting scheme is "plaintext aware")

[ref.: E. Fujisaki and T. Okamoto *How to Enhance the Security of Public-Key Encryption at Minimum Cost*, LNCS 1999, Vol.1560. Springer]

We do not go into the details of the proof, but simply give the transformation

Security of the ElGamal Cryptosystem

To make ElGamal cryptoscheme CCA2 secure we proceed as follows:

- The encryption function $\mathrm{ElGamal\text{-}Enc}(m, l) = \langle\, g^l, m \cdot (g^s)^l \,\rangle$ is altered by setting

$$\begin{aligned} \mathrm{Enc}(m, l) &= \mathrm{ElGamal\text{-}Enc}\,(m||l, H(m||l)) = \\ &= \langle\, g^{H(m||l)}, (m||l) \cdot (g^s)^{H(m||l)} \,\rangle \end{aligned}$$

  where H is a hash function, and $m||l$ is composed in such a way that it belong to the selected algebraic group

- The decryption algorithm is also altered in that we first compute

$$m' = \mathrm{Dec}(c), \text{ and then we check that } c = \mathrm{Enc}(m', H(m'))$$

  If this last equation holds we recover m from $m' = m||l$, otherwise we reject the received communication

- This scheme is only marginally less efficient than raw ElGamal scheme

ElGamal Signature Scheme

Given the cyclic group $(G, \cdot)$ with order $n = |G|$ (preferably prime) a generator $g \in G$, and a hash function $h : \{0,1\}^* \mapsto \mathbb{Z}_n$

Public key: $k_{\text{pub}} \leftarrow \langle\ n,\ g,\ g^s\ \rangle$ Private key: $k_{\text{priv}} \leftarrow \langle\ s \in \mathbb{Z}_n\ \rangle$

Signature Transformation

$m \in G$
$l \overset{\text{Random}}{\longleftarrow} \mathbb{Z}_n^*$
$\gamma \leftarrow (g)^l$
$\delta \leftarrow l^{-1} \cdot \big(h(m) - s \cdot h(\gamma)\big) \bmod n$
$S \leftarrow Sign_{k_{\text{priv}}}(m) = \langle \gamma, \delta \rangle$
Send $\langle m, S \rangle$

Validation Check

Receive $\langle m, S \rangle$
Compute $h(m), h(\gamma)$
Accept the signature only when
$Verify_{\ k_{\text{pub}}}(\langle m, S \rangle) = \text{true} \Leftrightarrow$

$$(g^s)^{h(\gamma)} \cdot \gamma^\delta \overset{?}{=} g^{h(m)}$$

Correctness verification:
$Verify_{\ k_{\text{pub}}}(Sign_{k_{\text{priv}}}(m)) = m, \quad \forall\, m \in G$

$$(g^s)^{h(\gamma)} \cdot \gamma^\delta = g^{s \cdot h(\gamma)} \cdot g^{l \cdot l^{-1} \cdot \left(h(m) - s \cdot h(\gamma)\right) \bmod n} = g^{h(m)}$$

1st Example

Given $G = (\mathbb{F}^*_{2^5}, \cdot)$, $f(x) = x^5 + x^2 + 1 \in \mathbb{F}_2[x]$; $n = |G| = 31$
$\mathbb{F}_{2^5} \cong \mathbb{F}_2(\alpha) = \{b_4\alpha^4 + b_3\alpha^3 + b_2\alpha^2 + b_1\alpha + b_0 | b_i \in \mathbb{F}_2,\ f(\alpha) = 0, \alpha \in \mathbb{F}_{2^5} \setminus \mathbb{F}_2\}$
with generator $g = (00010) = 0\,\alpha^4 + 0\,\alpha^3 + 0\,\alpha^2 + 1\,\alpha + 0\,\alpha^0 = \alpha$

Assume to employ an hash function $(h : \{0,1\}^* \mapsto \mathbb{Z}_n)$ that maps the binary sequence in the corresponding decimal value modulo n, and consider the key pair:

$$k_{\mathrm{priv}} = \langle 19 \bmod 31 \rangle, \quad k_{\mathrm{pub}} = \langle 31, \alpha, \alpha^{19} = \alpha^2 + \alpha = (00110) \rangle$$

Simulate an ElGamal Signature Protocol, knowing that $h(m)$=$16 \in \mathbb{Z}^*_{31}$, and $l \overset{Rand}{\leftarrow} \mathbb{Z}^*_{31},\ l = 24$:

Signature Transformation

$\gamma \leftarrow \alpha^{24} = \ldots = (11110)$
$l^{-1} \bmod n$=$24^{-1} \bmod 31 \equiv_{31} \ldots \equiv_{31} 22$
$h(\gamma) = h(\{(11110)\}) = 30 \bmod 31$
$\delta \leftarrow l^{-1} \cdot (h(m) - s \cdot h(\gamma)) \bmod n \equiv_{31} 26$
$s \leftarrow Sign_{k_{\mathrm{priv}}}(m) = \langle \gamma, \delta \rangle = \langle \gamma, \delta \rangle$
Send $\langle m, s \rangle = \langle m, (11110), 26 \rangle$

Validation Check

$\langle m, s \rangle$=$\langle \{10000\}, \gamma$=$(11110), \delta$=$26 \rangle$
$h(m) = 16, h(\gamma) = 30$
$(\alpha^{19})^{h(\gamma)} \cdot \gamma^{\delta} \overset{?}{=} \alpha^{h(m)}$
$(\alpha^3 + \alpha^2 + \alpha) \cdot (\alpha^4 + \alpha^3 + \alpha^2 + \alpha)^{26} =$
$= (\alpha^3 + \alpha^2 + \alpha) \cdot \alpha^4 = \ldots = (11011)$
$\alpha^{h(m)} = \alpha^{16} = \ldots = (11011)$

2nd Example

Consider the group $G = (\mathbb{F}_{11}^*, \cdot)$ with generator $g = 2$, $n = |G| = 10$, and a message binary string $m = \{1001\}$.

Assume to employ an hash function ($h : \{0,1\}^* \mapsto \mathbb{Z}_n$) that maps the binary sequence in the corresponding decimal value modulo n, and consider the key pair:

$$k_{\text{priv}} = \langle 4 \bmod 10 \rangle, \quad k_{\text{pub}} = \langle n = 10, g = 2, g^4 = 5 \rangle$$

Simulate an ElGamal Signature Protocol, assuming $l \stackrel{Rand}{\longleftarrow} \mathbb{Z}_{11}^*,\ l = 3$.

Recommended Key Lengths (DH & ElGamal schemes)

US NIST recommended key lengths, considering the foreseen technological and theoretical cryptanalysis advancements

Date	Security margin	Symmetric cipher	Group & PubKey size [bit]	Private Key size [bit]
2010	80	2TDEA*	1024	160
2011 – 2030	112	3TDEA	2048	224
$>$ 2030	128	AES-128	3072	256
$>>$ 2030	192	AES-192	7680	384
$>>>$ 2030	256	AES-256	15360	512

- **Security margin**: Minimum computational effort expressed as the log_2 of the number of DES computations
- **Symmetric cipher**: Suggested cipher to achieve the minimum adequate level of security (Note: nTDEA = Triple DES Algorithm with n keys)
    - (*) The assessment of at least 80 bits of security for 2TDES is based on the assumption that an attacker has no more than 2^{40} matched ptx/ctx blocks

Performance DLog schemes (DH & ElGamal schemes)

Overlook

The Diffie-Hellman key exchange has a computational cost greater than the RSA decryption function roughly by a factor $\times 4$ on a commodity machine

Table: OpenSSL-Performances on Intel Core2-duo™ Processor-L9400,1.86Ghz (one core used)

Security Margin	Prime size [bit]	(Sub) Group size [bit]	Diffie-Hellmann [ms]
80	1024	160	1.607
112	2048	224	11.178
128	3072	256	36.063
192	7680	384	542.012
256	15360	512	4,549

Digital Signature Standard Algorithm (DSS-DSA)

The "Digital Signature Algorithm" (DSA) is a US standard for digital signatures

- It was proposed by the NIST for use in their Digital Signature Standard (DSS), and specified in FIPS 186-3 (2009)
- It is a variant of the ElGamal signature scheme

Basic Assumptions

The reference group category is $(\mathbb{Z}_p^*, \cdot)$, p prime s.t. $q \mid (p-1)$, with q also a prime. The employed algebraic structure is then the multiplicative cyclic subgroup $G = \langle g \rangle$ with publicly known generator g and order q

Public Key: $k_{\text{pub}}=(p,\, q,\, g,\, g^s)$

Private Key: $k_{\text{priv}}=s \in \mathbb{Z}_q^*$

Denote with $L = \log_2(p)$ and $N = \log_2(q)$ the bit lengths of the prime numbers p and q, respectively

Digital Signature Standard Algorithm (DSS-DSA)

Parameter Generation

- FIPS 186-3 specifies the ($L = \log_2(p)$, $N = \log_2(q)$) length pairs of (1024,160), (2048,224), (2048,256), and (3072,256)
- Choose a cryptographic hash function H among the SHA-2 functions recommended in FIPS 180-3 (SHA-224, SHA-256, SHA-384, SHA-512), depending on the size selected for the key pair
- p, and $q \mid (p-1)$ primes, $g \in \mathbb{Z}_p^*$ with order q. Usually $g{=}h^{\frac{p-1}{q}}$, with $h{=}2{\in}\mathbb{Z}_p^*$, if g results to be 1, then pick another value h

Digital Signature Standard Algorithm (DSS-DSA)

Public Key: $k_{\mathrm{pub}}=(p,\ q,\ g,\ g^s)$

Private Key: $k_{\mathrm{priv}}=s \in \mathbb{Z}_q^*$

Signature Transformation

- $H(m)\in\{2, \ldots, q-1\},\ l \stackrel{\mathrm{Random}}{\longleftarrow} \mathbb{Z}_q^*$
- $\gamma \leftarrow \big((g)^l \bmod p\big) \bmod q$
- If $\gamma = 0$, repeat with another random l
- $\delta \leftarrow l^{-1} \cdot \big(H(m) + s \cdot \gamma\big) \bmod q$
- If $\delta = 0$, repeat with another random l
- $S \leftarrow Sign_{k_{\mathrm{priv}}}(m) = \langle \gamma, \delta \rangle$

Send $\langle m, S \rangle$

Validation Check

- Receive $\langle m, S \rangle$
- if $\gamma, \delta \notin \{1, \ldots, q-1\}$, reject the signature
- Compute $H(m)$

Accept signature iif
$Verify_{\,k_{\mathrm{pub}}}(\langle m, S \rangle) = \mathrm{true} \Leftrightarrow$

- $u_1 \leftarrow H(m) \cdot \delta^{-1} \bmod q$
- $u_2 \leftarrow \gamma \cdot \delta^{-1} \bmod q$

$(g^{u_1}\,(g^s)^{u_2}) \bmod p \bmod q \stackrel{?}{=} \gamma$