ElGamal Encryption and Diffie-Hellman Key Exchange

Overview:

- Diffie-Hellman was the first efficient asymmetric system [DiHe_76]
- A little bit weaker than the asymmetric encryption
- ElGamal based encryption is a simple expansion of it [ElGa_85]
- Based on the standard discrete log assumption, but not as secure provable

Diffie-Hellman Key Exchange : The Idea

$$\begin{tikzcd}
& g^{x\,y} & \\
h = g^{x} \arrow[ur, "y"] \arrow[dr, "x"'] & & g^{y} = i \arrow[ul, "x"'] \arrow[dl, "y"] \\
& g &
\end{tikzcd}$$

commutative superscript

$$g^{x\,y} = (g^{x})^{y} = (g^{y})^{x}$$

- ***h*, *i* public, *x*,*y* secret:**
  - keys *h*,*i* known in advance
  - or sent especially for that purpose
- "Hard" to compute x from g^x
- g^{xy} shared secret of the user of x and the user of y

Discrete Logarithm

- Let *p* be a prime number
- *g* be a generator *mod p*;
  i.e *(Z / pZ)** is generated by *g*

- Then for all A ∈ {1, 2, ... , p – 1} there is an *a* with:
  - A º g^{a} *mod p*

- *a* is called the *discrete logarithm* of *A* wrt. radix *p*

Euler‘s Function

- φ(m) computes the number of integers less or equal than m which have no non-trivial common divisor with m:

j(m) = #{*x* | *x* *Î* *{1, .., m}* and *gcd(x, m) = 1*}

Examples: *j(1) = 1, j(2) = 1, j(3) = 2, j(4) = 2, ...j(10) = 4*

obviously: *j(p) = p –1* for all prime numbers *p*

j(m) is the order of (/m)* !!!

Fermat‘s little theorem

- Let *gcd(a, m) = 1* then $a^{j(m)}$ *º 1mod m*

  (Hence $a^{j(m)-1} \cdot a$ *º 1mod m* and $a^{j(m)-1}$ is an inverse!)

- Let p be a prime number then: a^{p-1} *mod p = 1*

  (Useful to check whether *a* is a prime number:
  compute a^{p-1} *mod p* using fast exponentiation)

ElGamal Encryption

Key generation:

- Select a prime number $\boldsymbol{p}$
- Choose a generator $\boldsymbol{g}$ for p
- Choose an exponent $\boldsymbol{a} \in \{0, \ldots, p-2\}$

- Let $A := g^{a} \bmod p$.

Public key: $pk := (p, g, A)$

Secret key: $sk := (p, g, \boldsymbol{a})$

ElGamal Encryption

Encryption (Bob):

For *enc* (*pk*, *m*) with *pk* = (*p*, *g*, *A*) as Alices public key:

- Bob chooses $b \in \{0, \ldots, p-2\}$
- Let $B = g^{b} \bmod p$
- Let $c = A^{b} m \bmod p = g^{ab} m \bmod p$
- Send (B, c) to Alice

Decryption (Alice):

For *dec* (*sk*, *c*) with *sk* = (*p*, *g*, *a*) as Alices secret key:

- Compute $x = p - 1 - a$
- Compute $B^{x} c \bmod p$
- $B^{x} c \bmod p = g^{bx} c \bmod p = g^{b(p-1-a)} A^{b} m \bmod p$
  $= g^{b(p-1)} g^{-ba} g^{ab} m \bmod p = (g^{(p-1)})^{b} m \bmod p = m$

Diffie-Hellman Key Exchange: Security

- Unknown whether Diffie-Hellman assumption is as hard to break as the discrete-log-assumption
- Partial information about m accessable
- Insecure against chosen-ciphertext-attack
- (probabilistic, not a simple generate and test attack)

- Partial information:

  Idea: If $A = g^a$ or $B = g^b$ are in the same **subgroup** then also g^{ab}

  $\Rightarrow$ information about g^{ab}

  $\Rightarrow$ information about $m = c \bullet g^{-ab}$.

El Gamal : Efficency

- El Gamal decryption requires 1 exponentiation
  - Similar to RSA

- El Gamal encryption requires 2 exponentiations:
  - *A^b mod p* and *B = g^b mod p*
  - RSA requires only 1!
  - Length of prime number p are comparable to RSA
  - But Bob may compute *A^b mod p* and *B = g^b mod p ahead:*
    - Then only 1 multiplication is needed
    - More efficient than RSA
    - Storing on a chip card for instance

ElGamal Encryption and Diffie-Hellman Key Exchange

Disadvantages:

- Encryption depends on the keys of the sender
  - not suitable for anonymous communication.
- The group (e.g. p for Z_p) and *g* should be equal.
  Who selects them?
  - A common authority ?
  - One has to confide it.
- Stronger cryptographic assumption is necessary:
  - Hard to calculate *discrete logarithm*, even if one selects *p*, *g* in a special way
  - Still not broken, risky anyway
- Cryptographic protocol, great complexity, old arbitrary digit charts.

Diffie-Hellman Key Exchange

Construction:

- *g is well-known*
- h, i alias pk_1, pk_2: public keys of two participants
- x, y alias sk_1, sk_2 : corresponding secret keys.
- for communicating they use $k_{12} := g^{xy}$ as shared secret keys

Use

- *n:* many participants
- Every participant x publishes one public key g^x
- Now each pair of participants x, y has a secret key g^{xy}

ElGamal Encryption and Diffie-Hellman Key Exchange

g. Chosen-ciphertext-attack

The attacker wants to encrypt:

$\boldsymbol{c = (i, c^*)}$.

- *He sends to the receiver*

  $\boldsymbol{c_1 := (i, c_1^*)}$

  with an arbitrary c_1^*.
- *The recipient decrypts:*

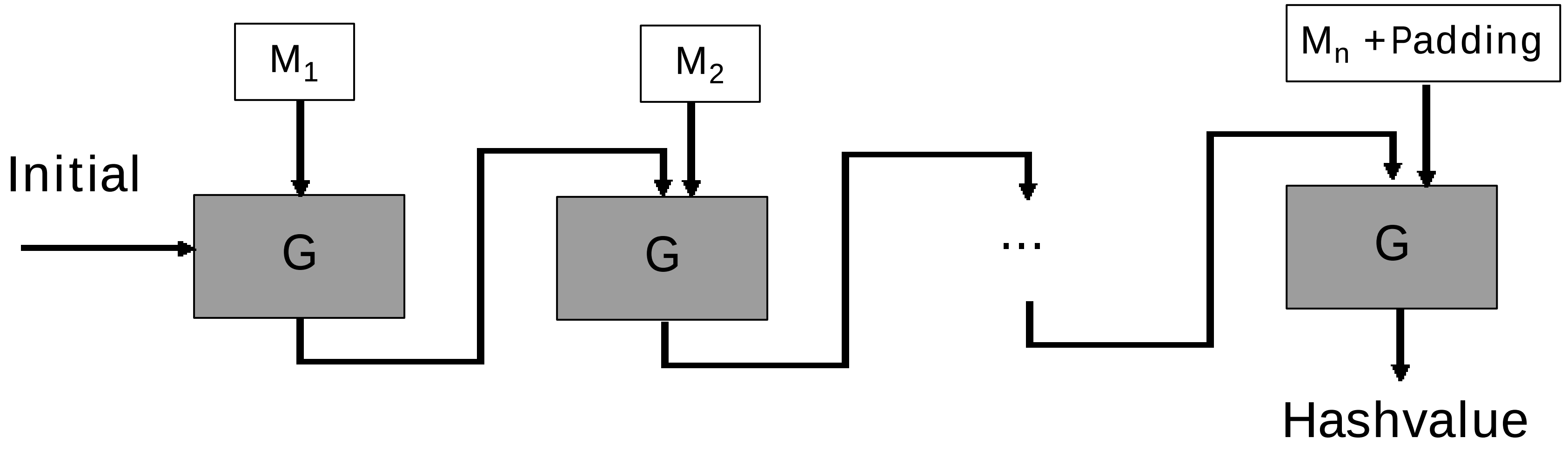
  $m_1 := c_1^* / i^x$.
- The attacker calculates the "real key" $k = g^{xy} = i^x$ as

  $k = c_1^* / m_1$
- He encrypts that way c^* to $m = c^* / k$.

Common Hash Function

Hash - Functions

- $f(0) = \text{Initial};$
- $f(i) = G(f(i-1), M_i)$, with $M = M_1, \ldots, M_n$
- $\text{Hash}(M) = f(n)$



Birthday Paradoxon

- How many persons n are necessary such that the probality is higher than 0.5 that 2 persons have birthday at the same day in the year?

$$n > 1 + \mathrm{sqrt}(1 + (8 \cdot \ln(2)) \cdot 2^{365}) \ / \ 2$$

$$n > 22 \text{ is sufficient !}$$

Basically only 2^{32} tests are necessary to find a hash-value collision of a 64 bit hash value

Secure Hash Algorithm (SHA-1)

- Part of the Secure Hash Standard
- Developed in 1993 by NIST (National Institute of Standards and Technology)
- Produces 160 bit hashvalues.
- Blocksize: 512 bits split into 16 words a 32 bits
- Strong hash function

Some Details on SHA-1

Collection of 80 operations $f_0 \ldots f_{79}$ on 32bit words:

$f_i(x, y, z) =$

- 0x5A827999 + $((x \wedge y) \vee (\neg x \wedge z))$ if $i < 20$
- 0x6ED9EBA + (x *xor* y *xor* z) if $19 < i < 40$
- 0x8f1bbcdc + $((x \wedge y) \vee (x \wedge z) \vee (y \wedge z))$ if $39 < i < 60$
- 0xCA62C1D6 + (x *xor* y *xor* z) if $59 < i < 80$

- Each input block M_i is split into 16 blocks $W_0, \ldots, W_{15}$

SHA-1 Algorithm

```
For t = 16 to 79 do
    $W_t = W_{t-3}$ xor $W_{t-8}$ xor $W_{t-14}$ xor $W_{t-16}$
    Od
A = $h_0$; B = $h_1$; C = $h_2$; D = $h_3$; E = $h_4$;

For t = 0 to 79 do
   temp = $S_5(A) + f_t(B, C, D) + E + W$
   E = D; D = C; C = $S_{30}(B)$; B = A;  A = temp
Od

$h_0 = h_0 + A$; $h_1 = h_1 + B$; $h_2 = h_2 + C$
$h_3 = h_3 + D$; $h_4 = h_4 + E$;
```

Message Digest 5

- MD5 : developed by R. Rivest
- Refinement of MD4
- MD5 yields 128 bits hashvalue
- Operates on 512 bits blocks split into 16 word a 32 bits
- No longer in use

- Problems with MD5 (Dobbertin 1996)
  - General approach to compute collisions
  - Examples use about 10h on a Pentium PC