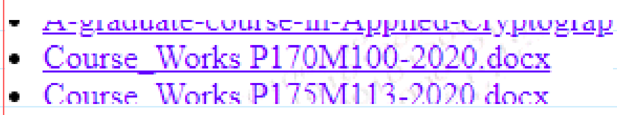
006_KS ElGamal-Sig

Topics of Course Works you will find in
http://crypto.fmf.ktu.lt/xdownload/



- A-graduate-course-in-Applied-Cryptograp
- Course_Works P170M100-2020.docx
- Course_Works P175M113-2020.docx

You must choose suitable topic of Course Work by labeling it in Google drive
https://drive.google.com/file/d/1qjFw1OJnPcwa3CFvg-Of1xt_B9RXXAqq/view?usp=sharing

→ Open v → „Google“ documents

Midterm exam should be from 8 to 16 week: suggest 11-th week, 12-th of November.
It will be arranged during the lecture 17:30-18:00.

https://imimsociety.net/en/14-cryptography
Problems required to solve:
DH-KAP, MiM Attack, RSA signature, RSA encryption
https://imimsociety.net
http://crypto.fmf.ktu.lt/xdownload/

El-Gamal Encryption: example with Octave

Public parameters : (p, g)

***p* = 264043379 Check that *p* is strong prime**
***g*=2**

```
>> genstrongprime(28)
ans = 15412127
>> p=ans
p = 15412127
>> q=(p-1)/2
q = 7706063
>> isprime(q)
ans =  1

>> g=3
g =  3
>> mod_exp(g,2,p)
ans = 9

>> mod_exp(g,q,p)
ans = 1
```

```
>> x=randi(p-1)
x =  3121242
>> a=mod_exp(g,x,p)
a = 13704847
```

$p = 2q+1$

p, q - primes ⇒ p - strong prime

A: key pair ; $x =$ PrK

$a = g^{x} \bmod p =$ PuK

$C = (E, D)$

$m = 123456 < p$

$r \leftarrow$ rand ; $r < p-1$

$E = m \cdot \boxed{a^{r} \bmod p}$

$D = g^{r} \bmod p$

```
    >> m=123456
    m =  123456
    >> r=randi(p-1)
    r =  3716363
    >> e1=mod_exp(a,r,p)
    e1 = 6027330
    >> e=mod(m*e1,p)
E ≡ e = 12560920
    >> d=mod_exp(g,r,p)
D ≡ d = 7241872
```

```
>> g=17
g =  17
>> mod_exp(g,q,p)
ans = 15412126
```

$$B: \xrightarrow{C=(E,D)=(12560920,\ 7241872)} A: m = E\cdot D^{-x} \bmod p$$

Fermat Th.: $D^{p-1} = 1 \bmod p \quad / D^{-x}$

$$D^{p-1}\cdot D^{-x} = D^{-x} \bmod p \Rightarrow \boxed{D^{-x} = D^{p-1-x} \bmod p}$$

```
>> demx=mod_exp(d,p-1-x,p)
demx = 4633989
>> m1=mod(e*demx,p)
m = m1 = 123456
```

El-Gamal E-Signature

Digital Signature Standard – DSS ← NSA

DSA

Elliptic Curve Cryptosystem – ECC → ECDSA

RSA Textbook signature scheme: $PuK=(e, n)$; $PrK=(d)$. $m < n$

$Sig(m, PrK) = m^d \bmod n = s \qquad A \xrightarrow{m,\ s} B$

$Ver(s, PuK) = s^e \bmod n = m^{\overset{=1 \bmod \phi(n)}{de}} \bmod n = m'$

If $m' = m \Rightarrow$ e-signature is valid.

Signature with message recovery.

Deterministic signature algorithm (non-randomised)

The **ElGamal signature scheme** is a digital signature scheme which is based on the difficulty of computing discrete logarithms. It was described by Taher ElGamal in 1984.[1]
The ElGamal signature algorithm is rarely used in practice. A variant developed at NSA and known as the Digital Signature Algorithm is much more widely used. There are several other variants.[2] The ElGamal signature scheme must not be confused with ElGamal

encryption which was also invented by Taher ElGamal.
The ElGamal signature scheme allows a third-party to confirm the authenticity of a message sent over an insecure channel.
From <https://en.wikipedia.org/wiki/ElGamal_signature_scheme>

Discr. exp. funct DEF : having $PP=(p,g)$ and x find $a=g^x \bmod p$

Discr. log. funct. DLF : having $PP=(p,g)$ and a find x.

$\mathrm{dlog}_g a = \mathrm{dlog}_g g^x \bmod p = x \overbrace{\boxed{\mathrm{dlog}_g g \bmod p}}^{=1} = x \Leftarrow$ total breaked

Discrete Logar. Ass. - DSA $\Rightarrow$ computation of $\mathrm{dlog}_g a$ is infeasible!

M - message to be signed: $|M| \sim 1GB = 8 \cdot 2^{30}$ bits
$|p| \sim 2048 = 2^{11}$ bits

$|M| >> |p| \Rightarrow$ signing M is not effective since it is required to split M into the pieces $|M_i| < |p|$.

How to sign large messages?

H-function ; message digest $\leftrightarrow$ santraukos f-ja

$H : \{0,1\}^* \longrightarrow \{0,1\}^{256}$ SHA 3, SHA 256

M - message to be signed ; Public available H-function

$H(M) = h$; $|h| = 256$ bits

1.***System parameters*** (**PP**)

$|p| \sim 2048$ bits

$|h| < |p|$

- Let ***H*** be a collision-resistant hash function.
- Let ***p*** be a large prime such that computing discrete logarithms modulo ***p*** is difficult.
- Let ***g*** < ***p*** be a randomly chosen generator of the multiplicative group of integers modulo *p* **Z_p^*= {1, 2, …, p-1} ={g^i | i=0, 1, 2,…, p-2}**. //Fermat theorem

These System Parameters (**SP**) must be shared between users.
SP = (**p**, **g**)

$p \sim 2^{2048} \approx 10^{700}$; $h < p$. $|M| >> |p|$

$H(M) = h$; $h < p$.

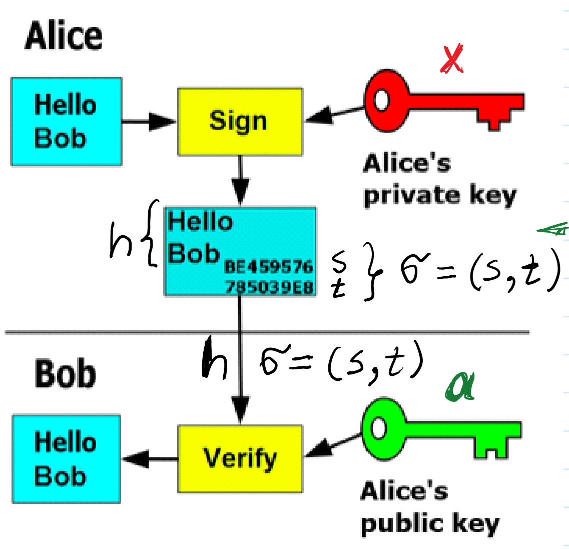
2.***Key generation***

- Randomly choose a private key ***X*** with $\mathbf{1 < x < p-1}$.

- Compute $a = g^x \bmod p$.
- The public key is **PuK = *a***.
- The private key is **PrK = *x***.

These steps are performed once by the signer.

Digital signature



$A: \xrightarrow{PuK = a = g^x \bmod p} B$

$Encode('M') = M_{dec}$

$M_{dec} >> p \,; \; |'Hello\ Bob'| < |p|$

$h = H(M_{dec})$

$h = H('M')$

alternatives

3.***Signature generation***

To sign a message ***M*** the signer performs the following steps.

$\exists!\ k^{-1} \bmod (p-1)$

$k \cdot k^{-1} = 1 \bmod (p-1)$

- Compute **h=H(*M*)**.
- Choose a random ***k*** such that **1 < *k* < *p* — 1** and **gcd(*k*, *p* — 1) = 1**.

k^{-1} mod (p-1) exists if **gcd(*k*, *p* — 1) = 1**, i.e. **k** and **p-1** are relatively prime

k^{-1} can be found using either Extended Euclidean algorithmt or Euler theorem

>> kem1=mulinv(k,p-1)	% k^{-1}mod (p-1) computation

- Compute **$t=g^k \bmod p$**
- Compute **$s=(h-x*t)*k^{-1} \bmod (p-1)$ --> $h=x*t+s*k \bmod (p-1)$**,

Signature **Sigma=(s,t)** $= \sigma$

$h - xt = ks$

$h = xt + ks \bmod (p-1)$

- If **s=0**, start over again.

Then the pair (***s***,***t***) is the digital signature of ***h***.

The signer repeats these steps for every signature.

$A \xrightarrow{M, \; \sigma = (s,t)} B$: computes $H(M) = h$

4.***Verification***

A signature (s,t) on message h is verified as follows.

1. $1<s<p-1$ and $1<t<p-1$.
2. $\underbrace{V1=a^t t^s \bmod p,}$ $\underbrace{V2=g^h \bmod p}$ and $V1=V2$.

The verifier accepts a signature if all conditions are satisfied and rejects it otherwise.

5.**Correctness**

The algorithm is correct in the sense that a signature generated with the signing algorithm will always be accepted by the verifier.

The signature generation implies

$$h=xt+sk \bmod (p-1)$$

Hence Fermat's little theorem implies that all operations in the exponent are computed mod (p-1)

$$\overbrace{g^h}^{V2}=g^{(xt+ks) \bmod (p-1)} \bmod p = g^{xt}g^{ks} = (g^x)^t(g^k)^s = \overbrace{a^t t^s \bmod p}^{V1}$$

Comments: V1 = V2 means that signature is formed with the PrK=x to which corresponds PuK=a and nothing more.

But! To impersonates A: Bob I'm sending you my public key a and please check my singned messages with this key.

```
>> M = 'Hello Bob, I need to meet you'
>> m26 = H26(M)
>> m28 = H28(M)   % |m28| = 28 bits ≡ 7 Bytes
```

Up to that

ElGamal Signature

$Z_p^* = \{1, 2, 3, ..., p-1\}$

ECDSA

EC group ⊞ → ECG

$g \in \mathbb{Z}_p^* = \{g^i \mid i = 0, 1, 2, \dots, p-2\}$

$* \bmod p$

G - generator

$ECG = \{iG \mid i = 1, 2, \dots, |ECG|\}$

$a = g^x \bmod p \Longleftrightarrow A = xG$

x - random secret number

$x < |ECG| \sim 2^{256}$

A, G - Elliptic curve points

Iki čia 2019.10.21

ECDSA po prakt. pvz.

11.65 ***Example***: ElGamal signature generation with artificially small parameters

Key generation.

A selects the prime p = 2357 and a generator g = 2 of Z^*_{2357}. $= \{1, 2, 3, \dots, 2356\}$

A chooses the private key **PrK = x = 1751**

and computes public key **PuK = a = g^x mod p = 2 ^{1757} mod 2357 = 1185**.

System parameters are **SP = (p = 2357, g = 2)**

A's public key is **PuK = (a = 1185)** and private key **PrK = (x = 1751)**.

Signature generation.

For simplicity, let messages will be integers from **Z_P^* = {1, 2, …, p-1}**, **m ≠ 0**.

And for this example only, take **H** to be the identity function, i.e. **H(m) = m**.

Let message **m = 1463**.

A selects a random integer **k = 1529**,

computes **r = g^k mod p = = 2^{1529} mod 2357 = 1490**.

$k^{-1} = 1/k = \frac{1}{1529} = \dots$

To compute **k^{-1} mod (p - 1)**, ***A*** uses Extended Euclidean algorithm:

Let gcd(***k***, ***p*** – **1) = d**, then there exist such **u**, **v** that

$$k \cdot u + (p-1) \cdot v = d = \gcd(k, p-1) = 1 = d$$

```
>>eeuklid(k, p-1)
ans =     gcd(k,p-1) = d          u          v
>>eeuklid(1529, 2357-1)
Ans =              1             245        -159
```

```
>> eeuklid(1529,2357-1)
ans =     1   245  -159              //verification
>> 1529*245+(2357-1)*(-159)
ans =  1
ans =    1   245  -159            //verification
>> 1529*245+(2357-1)*(-159)
ans =  1
```

Then k^{-1} mod (p - 1) = 245. //verification

$k \cdot k^{-1}$ mod (p - 1) = 1529·245 mod (2357-1) = 1

```
>> mod(1529*245,2357-1)
ans =  1
```

Finally, ***A*** computes H(m) = m = 1463

$s=(h-xr)k^{-1}$ mod (p-1) = (1463-1751·1490)·245 mod (2357-1) = 1777

```
>> mod((1463-1751*1490)*245,(2357-1))
ans =  1777
```

A's signature **S** for **m = 1463** is the pair **S = (r = 1490; s = 1777)**.

$$A: \xrightarrow{m,\ S=(r,s)} B$$

Signature verification.

B computes using   Puk = (a = 1185)   SP=(p=2357, g=2)

V1 = $a^r r^s$ mod p = $1185^{1490} \cdot 1490^{1777}$ mod 2357 = 387·557 mod 2357 = 1072.

```
>> mod_exp(1185,1490,2357)
ans =  387
>> mod_exp(1490,1777,2357)
ans =  557
>> mod(387*557,2357)
ans =  1072
```

H(m) = m = 1463 = h

V2 = g^h mod p = 2^{1463} mod 2357 = 1072.

```
>> mod_exp(2,1463,2357)
ans =  1072
```

B accepts the signature since **V1 = V2**.

Iki čia

-- **Iki čia** ---

Homomorphic property

$Sig_{PrK}(h_1) = s_1$ $\quad$ $Sig_{PrK}(h_2) = s_2$

Sig function is homomorphic, if

$$Sig_{PrK}(h_1 \cdot h_2) = Sig_{PrK}(h_1) \cdot Sig_{PrK}(h_2)$$

Textbook RSA signature scheme is homomorphic (isomorphic)
El-Gamal — " — is not — " —

El-Gamal encryption scheme is homomorphic

m- message to be encrypted

$$Enc_{PuK}(m) = C = (E, D) = (E = m\,a^{k}, D = g^{k}) \bmod p$$

$$Dec_{PrK}(C) = E \cdot D^{-x} \bmod p = m\,a^{k} \cdot g^{-kx} \bmod p =$$

$$= m\,\underbrace{(g^{x})}_{a}{}^{k} \cdot g^{-kx} = m \cdot g^{xk} \cdot g^{-kx} = m \cdot g^{xk-kx} = m \cdot g^{0} = m$$

$$Enc_{PuK}(m_1) = C_1 = (E_1 = m_1\,a^{k_1}, D_1 = g^{k_1}) \bmod p$$

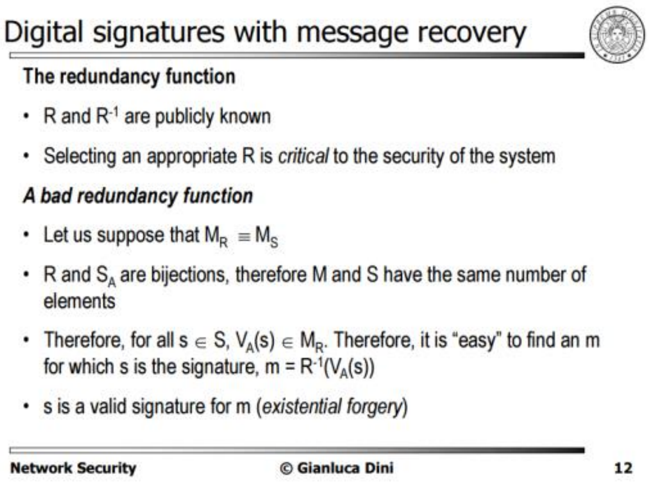
$$Enc_{PuK}(m_2) = C_2 = (E_2 = m_2\,a^{k_2}, D_2 = g^{k_2}) \bmod p$$

$$Enc_{PuK}(m_1 \cdot m_2) = Enc_{PuK}(m_1) \cdot Enc_{PuK}(m_2)$$

11.5.4 The ElGamal signature scheme with message recovery [Menezes]

The ElGamal scheme and its variants (x11.5.2) discussed so far are all randomized digital signature schemes with appendix (i.e., the message is required as input to the verification algorithm). In contrast, the signature mechanismof Algorithm11.81 has the feature that the message can be recovered from the signature itself. Hence, this ElGamal variant provides a randomized digital signature with message recovery.

For this scheme, the signing space is $\mathbf{M_s = Z_p^*}$, **p** a prime, and the signature space is $\mathbf{S = Z_P \times Z_q}$, **q** a prime, where **q** divides **(p — 1)**. Let **R** be a redundancy function from the set of messages **M** to $\mathbf{M_s}$ (see Table 11.1). Key generation for Algorithm 11.81 is the same as DSA key generation (Algorithm 11.54), except that there are no constraints on the sizes of **p** and **q**.



Digital signatures with message recovery

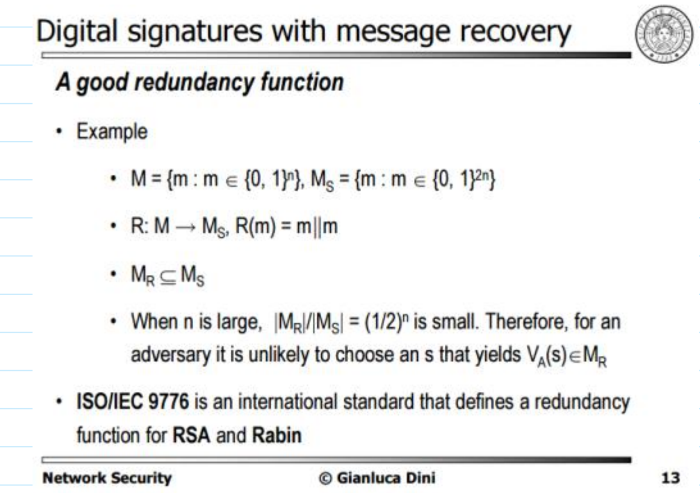
The redundancy function

- R and R^{-1} are publicly known
- Selecting an appropriate R is *critical* to the security of the system

A bad redundancy function

- Let us suppose that $M_R \equiv M_S$
- R and S_A are bijections, therefore M and S have the same number of elements
- Therefore, for all $s \in S$, $V_A(s) \in M_R$. Therefore, it is "easy" to find an m for which s is the signature, $m = R^{-1}(V_A(s))$
- s is a valid signature for m (*existential forgery*)

Network Security © Gianluca Dini 12



Digital signatures with message recovery

A good redundancy function

- Example
  - $M = \{m : m \in \{0, 1\}^n\}$, $M_S = \{m : m \in \{0, 1\}^{2n}\}$
  - $R: M \rightarrow M_S$, $R(m) = m||m$
  - $M_R \subseteq M_S$
  - When n is large, $|M_R|/|M_S| = (1/2)^n$ is small. Therefore, for an adversary it is unlikely to choose an s that yields $V_A(s) \in M_R$
- **ISO/IEC 9776** is an international standard that defines a redundancy function for **RSA** and **Rabin**

Network Security © Gianluca Dini 13

11.81 Algorithm Nyberg-Rueppel signature generation and verification

SUMMARY: entity A signs a messagem2M. Any entity B can verify A's signature and recover the message m from the signature.
1. Signature generation. Entity A should do the following:
(a) Compute em
= R(m).
(b) Select a random secret integer k, 1 ⍰ k ⍰ q−1, and compute r =
−k mod p.
(c) Compute e = emr mod p.
(d) Compute s = ae + k mod q.
(e) A's signature for m is the pair (e; s).
2. Verification. To verify A's signature (e; s) on m, B should do the following:
(a) Obtain A's authentic public key (p; q;
; y).
(b) Verify that 0 < e < p; if not, reject the signature.
(c) Verify that 0 ⍰ s < q; if not, reject the signature.
(d) Compute v =
sy−e mod p and em
= ve mod p.
(e) Verify that em
2MR; if em
62MR then reject the signature.
(f) Recover m = R−1(em
).
Proof that signature verification works. If A created the signature, then v ⍰
sy−e ⍰

s−ae ⍰

k (mod p). Thus ve
k em
−k em
(mod p), as required.

11.82 Example (Nyberg-Rueppel signature generation with artificially small parameters)

Key generation. Entity A selects primes p = 1256993 and q = 3571, where q
divides
(p − 1); here, (p − 1)=q = 352. A then selects a random number g = 42077 2 Z
p and
computes
 = 42077352 mod p = 441238. Since
 6= 1,
 generates the unique cyclic
subgroup of order 3571 in Z
p. Finally, A selects a random integer a = 2774 and computes
y =
a mod p = 1013657. A’s public key is (p = 1256993; q = 3571;
 = 441238; y =
1013657), while A’s private key is a = 2774.
Signature generation. To sign amessagem,Acomputes em
= R(m) = 1147892 (the value
R(m) has been contrived for this example). A then randomly selects k = 1001,
computes
r =
−k mod p = 441238−1001 mod p = 1188935, e = e mr mod p = 138207, and s =
(2774)(138207)+ 1001 mod q = 1088. The signature for m is (e = 138207; s =
1088).
Signature verification. B computes v = 4412381088 1013657−138207 mod
1256993 =
504308, and em
= v 138207 mod 1256993 = 1147892. B verifies that em
2 MR and
recovers m = R−1(em
).