

Lecture 10

Public Key Cryptography: Encryption + Signatures

1

Speeding up RSA decryption

Let: C - RSA ciphertext

$$d_p = d \bmod (p - 1)$$

$$d_q = d \bmod (q - 1)$$

compute:

$$M_p = C^{d_p} \bmod p$$

$$M_q = C^{d_q} \bmod q$$

and solve:

$$M = M_p \bmod p$$

$$M = M_q \bmod q$$

$$M = [M_p q (q^{-1} \bmod p) + M_q p (p^{-1} \bmod q)] \bmod (pq)$$

2

More on RSA

- Modulus n is unique per user \rightarrow can't share n
- What happens if Alice and Bob share the same modulus?
 - Alice has (e', d', n) and Bob - (e'', d'', n)
 - Alice wants to compute d'' (Bob's private key)
 - She knows that: $e' * d' = 1 \pmod{\phi(n)}$
 - So: $e' * d' = k * \phi(n) + 1$ and: $e' * d' - 1 = k * \phi(n)$
 - Alice just needs to compute inverse of e'' mod X
 - where $X = e' * d' - 1 = k * \phi(n)$
 - let's call this inverse d'''
 - and remember that: $d''' * e'' = k' * k * \phi(n) + 1$
 - can we be sure that: $d''' = d''$?
 - Is it possible that e'' has no inverse mod X ?
 - Yes, if $e'' = \phi(n)$ or $\gcd(e'', k) > 1$ but this is very, very UNLIKELY!
 - For all decryption purposes, d''' is EQUIVALENT to d''
 - Suppose Eve encrypted for Bob: $C = (m)^{e''} \pmod{n}$
 - Alice computes:

$$C^{d'''} \pmod{n} = m^{e'' d'''} \pmod{n} = (m)^{k' * k * \phi(n) + 1} \pmod{n} = m$$

3

El Gamal PK cryptosystem (83)

p - large prime
 b - base, primitive element, generator
 x - private exponent
 y - public residue; $y \equiv b^x \pmod{p}$
 $P = Z_p^*$
 $C = Z_p^* \times Z_p^*$
 publics : p, b, y
 secrets : x

Encryption :
 1. generate random $r \in Z_{p-1}$
 2. compute : $k = b^r \pmod{p}$
 3. compute : $c = m y^r \pmod{p} = m b^{xr} \pmod{p}$
 4. ciphertext = $\{k, c\}$

Decryption :
 1. compute $k^x \pmod{p}$
 2. compute $(k^x)^{-1} \pmod{p}$
 3. $m' = (k^x)^{-1} c = b^{-rx} m b^{xr} \pmod{p} = m$

4

El Gamal (example)

$p = 13$
 $b = 2$
 $x = 9$
 $y = 2^9 \text{ mod } 13 = 5$

Encryption :

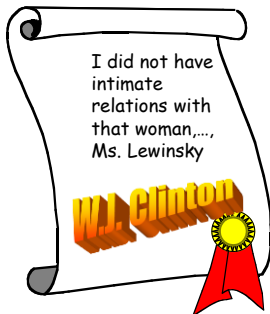
$m = 11$
 $r = 10$
 $k = 2^{10} \text{ mod } 13 = 10$
 $c = 11 * 5^{10} \text{ mod } 13 = 2$
ciphertext = { 10,2 }

Decryption :

$10^9 \text{ mod } 13 = 12$
 $12^{-1} \text{ mod } 13 = 12$
 $2 * 12 = 24 \equiv 11 \text{ mod } 13$

5

Digital Signatures



- Integrity
- Authentication
- Non-repudiation
- Time-stamping
- Causality
- Authorization



6

Digital Signatures

A signature scheme:

Usually message hash

(P,A,K,Sign,Verify)

P - plaintext (msgs)

A - signatures

K - keys

Sign - signing function: $(P * \underline{K}) \rightarrow A$

Verify - verification function: $(P * A * K) \rightarrow \{0,1\}$

7

RSA Signature Scheme

Use the fact that, in RSA, encryption reverses "decryption"

Let $n = pq$ where $p \neq q$ are two (large) primes
 $e \in Z_{\Phi(n)}^*$ and $e = d^{-1} \pmod{\Phi(n)}$ and $ed \equiv 1 \pmod{\Phi(n)}$
 $\Phi(n) = (p-1)(q-1)$
Secrets: p, q, d
Publics: n, e
Signing : $message = m$
Sign(m): $y = m^d \pmod n$
Verification : $signature = y$
Verify(y, m): $(m = y^e) ???$

8

RSA Signature Scheme (contd)

- The good:
 - Verification can be cheap (like RSA encryption)
 - Mechanically same as RSA decryption function
 - Security based on RSA encryption
 - Signing is harder but #verify-s > 1...
 - Deterministic
- The bad:
 - Recall that RSA is malleable: signatures can be "massaged"
 - Phony "random" signatures
 - compute $Y = \text{RSA}(e, X) = X^e \pmod n$
 - X is a signature of Y because $Y^d = X \pmod n$
- The ugly:
 - Signing requires integrity!
 - How to sign multiple blocks?
 - Deterministic - needs additional randomization!

9

El Gamal Signature Scheme

p - large prime
 b - base, generator
 x - private exponent
 y - public residue; $y \equiv b^x \pmod p$
 $P = \mathbb{Z}_p^*$
 $A = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$
 publics : p, b, y
 secrets : x

Signing :
 1. generate random $r \in \mathbb{Z}_{p-1}$
 2. compute : $k = b^r \pmod p$
 3. compute : $c = (m - xk)r^{-1} \pmod{p-1}$
 4. signature = $\{k, c\}$

Verifying :
 $y^k k^c \pmod p = b^m \pmod p$???

notice that :
 $y^k k^c = b^{xb^r} (b^r)^{(m/r - xk/r)} = b^{xb^r + m - xb^r} = b^m$

10

El Gamal PK Cryptosystem

p - large prime
 b - base, primitive element, generator
 x - private exponent
 y - public residue; $y \equiv b^x \pmod{p}$
 $P = Z_p^*$
 $C = Z_p^* \times Z_p^*$
 publics : p, b, y
 secrets : x

Encryption :

1. generate random $r \in Z_{p-1}^*$
2. compute : $k = b^r \pmod{p}$
3. compute : $c = my^r \pmod{p} = mb^{xr} \pmod{p}$
4. ciphertext = $\{k, c\}$

Decryption :

1. compute $k^x \pmod{p}$
2. compute $(k^x)^{-1} \pmod{p}$
3. $m' = (k^x)^{-1} c = b^{-rx} mb^{xr} \pmod{p} = m$

El Gamal Signature Scheme

p - large prime
 b - base, generator
 x - private exponent
 y - public residue; $y \equiv b^x \pmod{p}$
 $P = Z_p^*$
 $A = Z_p^* \times Z_p^*$
 publics : p, b, y
 secrets : x

Signing :

1. generate random $r \in Z_{p-1}^*$
2. compute : $k = b^r \pmod{p}$
3. compute : $c = (m - xk)r^{-1} \pmod{p-1}$
4. signature = $\{k, c\}$

Verifying :

$$y^k k^c \pmod{p} = b^m \pmod{p} \quad ???$$

notice that :

$$y^k k^c = b^{xb^r} (b^r)^{(m/r - xk/r)} = b^{xb^r + m - xb^r} = b^m$$

11

El Gamal Signature Scheme (contd)

The good:

- Signing is cheap(er)
- Designed as a signature function
- Non-deterministic (randomized)

The bad:

- Need GOOD source of random numbers
- Randomizers cannot be revealed (trace)
- Randomizers cannot be reused

12

The Digital Signature Standard (DSS)

- Why DSS?
- RSA issues: patents, malleability, etc.
- A variant of El Gamal
- Originally for $|p|=512$ bits, now up to 1024
- Optimized for signature size (320- vs. 1024-bit)
- Signing - 1 exp, verification - 2 exps
- No attacks thus far

13

DSS (contd)

p - large prime
 b - base, generator
 x - private exponent
 y - public residue; $y \circ b^x \bmod p$
 $P = Z_p^*$, $A = Z_p^* \wedge Z_p^*$
 publics : p, b, y secrets : x

Signing :
 1. generate random $r \in Z_{p-1}^*$
 2. compute : $k = b^r \bmod p$
 3. compute : $c = (m - xk)r^{-1} \bmod p - 1$
 4. signature = $\{k, c\}$

Verifying :
 $y^k k^c \bmod p = b^m \bmod p$???

p - 512 - bit prime
 q - 160 - bit prime, $(p - 1) \% q = 0$
 b - base, $b^q \circ 1 \bmod p$ ($b = d^{(p-1)/q}$)
 x - private exponent
 y - public residue; $y \circ b^x \bmod p$
 $P = Z_p^*$, $A = Z_q \wedge Z_q$
 publics : p, q, b, y secrets : x

Signing :
 1. generate random $r \in Z_{q-1}^*$
 2. compute : $k = (b^r \bmod p) \bmod q$
 3. compute : $c = (m + xk)r^{-1} \bmod q$
 4. signature = $\{k, c\}$

Verifying :
 $(b^{mc^{-1}} k^{kc^{-1}} \bmod p) \bmod q = b^k \bmod p$???

notice that :
 $b^{mc^{-1}} y^{kc^{-1}} = b^{mr/(m+xb^r)} (b^x)^{(b^r r)/(m+xb^r)}$
 $= b^{(mr+xb^r r)/(m+xb^r)} = b^r$

14