Lecture 7

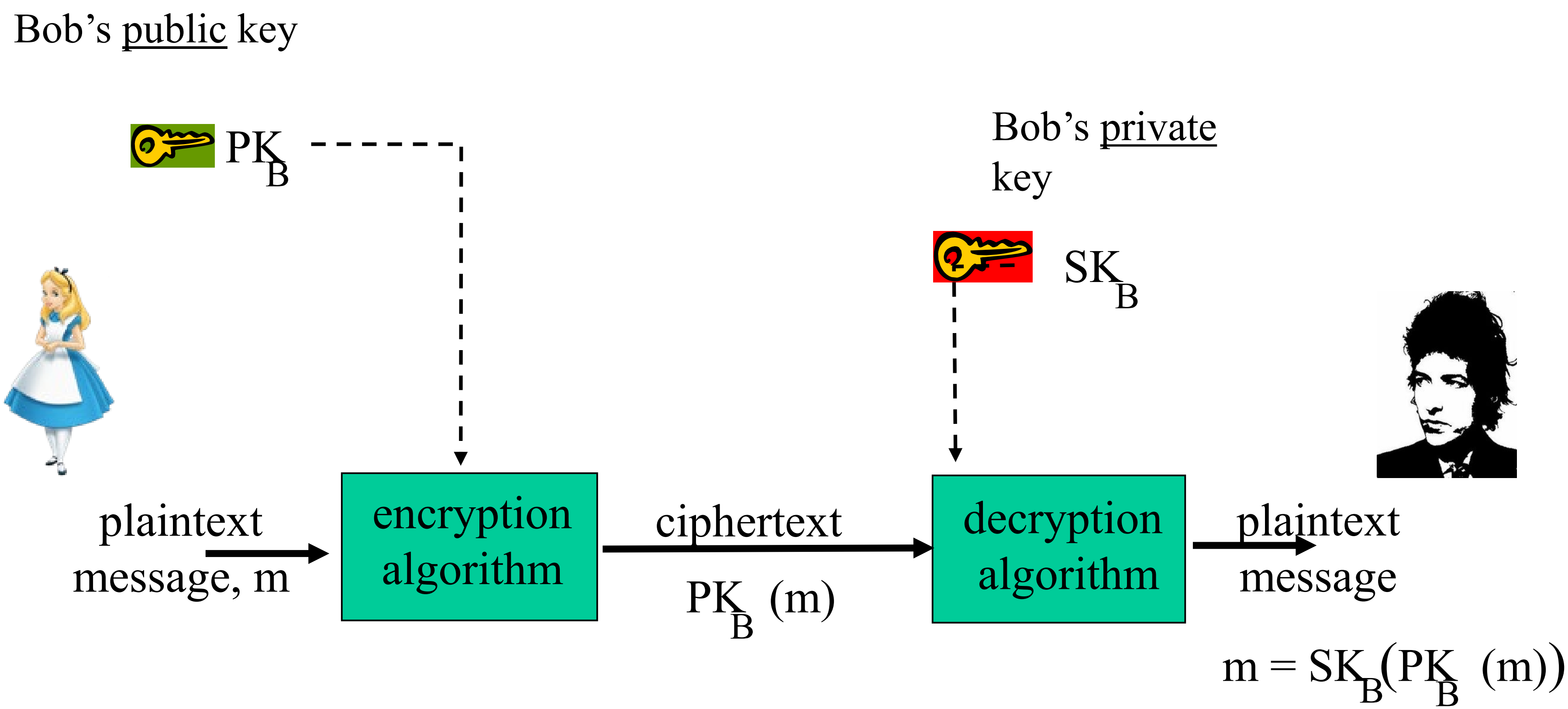
Public Key Cryptography I: Encryption + Signatures

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]

Public Key Cryptography

- Asymmetric cryptography
- Invented in 1974-1978 (Diffie-Hellman and Rivest-Shamir-Adleman)
- Two keys: private (SK), public (PK)
  - Encryption: with public key;
  - Decryption: with private key
  - Digital Signatures: Signing by private key; Verification by public key. i.e., "encrypt" message digest/hash -- $h(m)$ -- with private key
    - Authorship (authentication)
    - Integrity: Similar to MAC
    - Non-repudiation: can't do with symmetric key cryptography
- Much **slower** than conventional cryptography
    - Often used together with conventional cryptography, e.g., to encrypt session keys

Public Key Cryptography



Key Pre-distribution: Diffie-Hellman

"New Directions in Cryptography" 1976

$System-wide\ parameters:$

$p - large\ prime,$

$a - generator\ in\ Z^*_p$

Alice's secret: v, public: $y_a = a^v \bmod p$

Bob's secret: w, public: $y_b = a^w \bmod p$

Alice has: $y_b = a^w \bmod p$

Bob has: $y_a = a^v \bmod p$

$K_{ab} = (y_b)^v \bmod p$

$=$

$K_{ba} = (y_a)^w \bmod p$

Public Key Pre-distribution: Diffie-Hellman

Alice computes

K_{ab}

Bob computes

$K_{ab} = K_{ba}$

Secure communication

with K_{ab}

Eve knows:

p, a, y_a and y_b

Public Key Pre-distribution: Diffie-Hellman

Diffie – Hellman Problem:

p – *large prime*, a – *generator in* Z^{*}_{p}

Given:

$y_a = a^v \bmod p$ *and* $y_b = a^w \bmod p$

FIND: $a^{vw} \bmod p$

Discrete Log Problem:

Given:

$y_a = a^v \bmod p$

FIND: v

Public Key Pre-distribution: Diffie-Hellman

Decision DH Problem:

$p - large\ prime, a - generator$

$Given:$

$y_a = a^v \bmod p,\ \ y_b = a^w \bmod p$

$Distinguish:$

$K_{ab} = a^{vw} \bmod p$

$from\ a\ random\ number!$

- DH Assumption: DH problem is HARD (not P)
- DL Assumption: DL problem is HARD (not P)
- DDH Assumption: solving DDH problem is HARD (not P)

Interactive (Public) Key Exchange: Diffie-Hellman

Choose random v

$$y_a = a^v \bmod p$$

$$y_b = a^w \bmod p$$

Compute

$$K_{ab} = (y_b)^v \bmod p$$

Choose random w, Compute

$$K_{ba} = (y_a)^w \bmod p$$

Secure communication with K_{ab}

Eve is passive …

The Man-in-the-Middle (MitM) Attack

(assume Eve is an active adversary!)

Choose
random v

$$y_a = a^v \bmod p$$

Choose
random w,
Compute

$$K_{ba} = (y_a)^w \bmod p$$

$$y_b = a^w \bmod p$$

Compute

$$K_{ab} = (y_b)^v \bmod p$$

Secure communication
with Kab

RSA (1976-8)

$$Let\ n = pq \quad where \quad p,q - large\ primes$$

$$e,d \in Z^*_{\Phi(n)} \quad and \quad ed \equiv 1 \bmod \Phi(n)$$

$$where: \quad \Phi(n) = (p-1)(q-1) = pq - p - q - 1$$

$$Secrets: p,q,d$$

$$Publics: n,e$$

$$Encryption: message = m < n$$

$$E(m) = y = m^e \bmod n$$

$$Decryption: ciphertext = y$$

$$D(y) = m' = y^d \bmod n$$

Why does it all work?

$$x \in Z_n^*$$

$$x^{ed} = x^{1 \bmod \Phi(n)} \bmod n =$$

$$x^{c*\Phi(n)+1} \bmod n = x$$

But, recall that:

$$g^{\Phi(n)} = 1 \mod n$$

How does it all work?

Example: p=5 q=7 n=35 (p-1)(q-1)=24=3*2^3

pick e=11, d=11

x=2, E(x)=2048 mod 35 =18=y

y=18, D(y)=6.426841007923e+13 mod 35 = 2

Example: p=17 q=13 n=221 (p-1)(q-1)=192=3^4*2

pick e=5, d=77 Can we pick 16? 9? 27? 185?

x=5, E(x)=3125 mod 221 = 31

D(y)=31^{77}=

6.83676142775442000196395599558e+114 mod 221 = 5

Why is it Secure?

Conjecture: breaking RSA is *polynomially equivalent* to factoring **n**

Recall that n is very, very large!

Why: n has unique factors p, q

Given p and q, computing (p-1)(q-1) is easy:

$$ed \equiv 1 \, mod \, \Phi(n)$$

Use extended Euclidian!

Exponentiation Costs

- Integer multiplication -- $O(b^2)$ where b is bit-size of the base
- Modular reduction -- $O(b^2)$
- Thus, modular multiplication -- $O(b^2)$
- Modular exponentiation (as in RSA) -- $m^e \bmod n$
- Naïve method: e-1 modular products -- $O(b^2 * e)$
- BUT what if e is large, (almost) as large as n?

- Let L= |e| (e.g., L=1024 for 1024-bit RSA exponent)
- We can assume b and L are very close, almost the same
- Square-and-multiply method works in $O(b^3)$ time … $O(b^2 * 2L)$

Square-and-Multiply

$$\text{goal: compute } m^e \bmod n$$

```
l = sizeof(n);
temp = 1;
for (i = l - 1; i >= 0; i--)
{   temp *= temp;
    temp %= n;
    if (e[i])
    {   temp *= m;
        temp %= n;
    }
}
```

From left to right in e

- Example 1: e=100
- Example 2: e=10000000
- Example 3: e=11111111

Speeding up RSA Decryption

$Let:$ C - RSA ciphertext

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

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

compute:

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

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

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

and solve:

$M = M_p \bmod p$

$M = M_q \bmod q$

More on RSA

- **Modulus n is unique per user →**
  - **<u>2 or more parties cannot share the same n</u>**
- **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), but does not know phi(n)**
  - **She knows that: e' * d'= 1 mod 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 gcd(e",k)>1 but this is very, very UNLIKELY!**
  - **For all decryption purposes, d'" is EQUIVALENT to d"**
  - **Suppose Eve encrypted for Bob: $C = (m)^{e''} \bmod n$**
  - **Alice computes:**

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

El Gamal PK Cryptosystem (`83)

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

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

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

El Gamal (Example)

$p = 13$

$b = 2$

$x = 9$

$y = 2^9 \bmod 13 = 5$

Encryption:

$m = 11$

$r = 10$

$k = 2^{10} \bmod 13 = 10$

$c = 11 * 5^{10} \bmod 13 = 2$

$\text{ciphertext} = \{10,2\}$

Decryption:

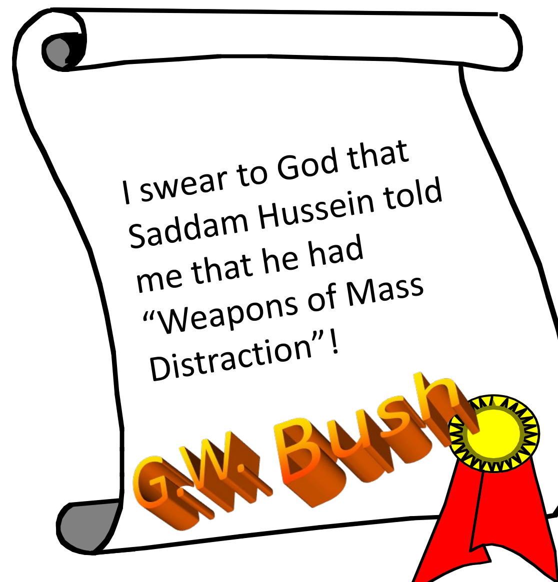
$10^9 \bmod 13 = 12$

$12^{-1} \bmod 13 = 12$

$2 * 12 = 24 \equiv 11 \bmod 13$

Digital Signatures

- Integrity
- Authentication
- Non-Repudiation
- Time-Stamping
- Causality
- Authorization



Digital Signatures

A signature scheme:

Usually message hash

(P,A,K,Sign,Verify)

P **- plaintext (msgs)**

A **- signatures**

K **- keys**

Sign **- signing function: (*P*K)->A***

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

RSA Signature Scheme

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

$$\text{Let } n = pq \text{ where } p \neq q \text{ are two (large) primes}$$

$$e \in Z^*_{\Phi(n)} \text{ and } e = d^{-1} \bmod \Phi(n) \text{ and } ed \equiv 1 \bmod \Phi(n)$$

$$\Phi(n) = (p-1)(q-1)$$

$$Secrets: p, q, d$$

$$Publics: n, e$$

$$\text{Signing}: message = m$$

$$Sign(m): y = m^d \ mod\ n$$

$$\text{Verification}: signature = y$$

$$Verify(y, m): (m = y^e)???$$

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:
  - RSA is malleable: signatures can be “massaged”
    - $m_1^d * m_2^d = (m_1 * m_2)^d$
  - Phony “random” signatures
    - compute $Y=RSA(e,X)=X^e \bmod n$
    - X is a signature of Y because $Y^d=X \bmod n$
- The Ugly:
  - Signing requires integrity!
  - How to sign multiple blocks when m > n?
  - Deterministic – needs additional randomization!

Plaintext	SIG
X^e	X