Lecture 7

Public Key Cryptography I: Encryption + Signatures

1

1

Public Key Cryptography

- aka “asymmetric cryptography”
- Invented in 1974-1978 timeframe
  - Merkle + Diffie-Hellman,
  - First public key encryption schemes: ElGamal Rivest-Shamir-Adleman (RSA)
- Two keys: private (SK), public (PK)
  - Encryption: with public key;
  - Decryption: with private key
  - Digital Signatures:
    - Signing by private key / verification by public key
    - Origin authentication
    - Integrity: similar to symmetric MAC but better
    - Non-repudiation: **can’t do this with symmetric key cryptography**
- Much **slower** than conventional cryptography
    - Often used together with conventional cryptography, e.g., to encrypt session keys

2

2

Public Key Cryptography

Bob's public key PK_B

Bob's private key SK_B





3

3

Public Key Cryptography

Bob's public key

PK_B

Bob's private key SK_B



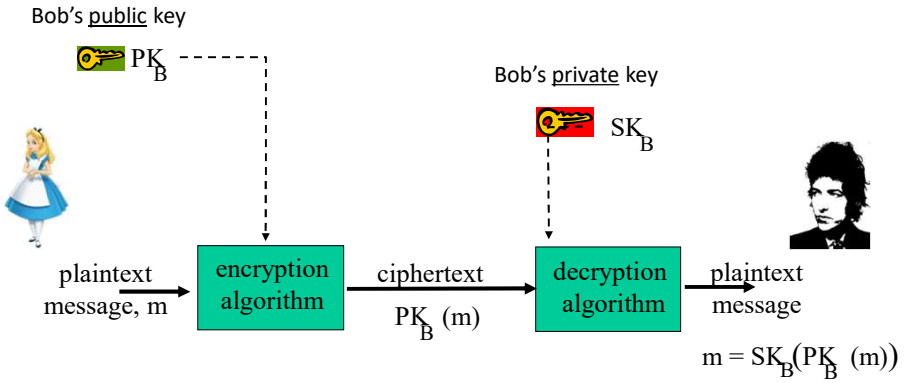


plaintext message, m → encryption algorithm → ciphertext $PK_B(m)$

4

4

Public Key Cryptography



5

5

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: $y_a = a^v \bmod p$

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

$=$

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

$= a^{vw} \bmod p$

6

6

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

7

7

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:

$y_a = a^v \bmod p$

$FIND:\ v$

8

8

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 NP
- DL Assumption: DL problem is NP
- DDH Assumption: solving DDH problem is NP

9

Recap: Complexity Reminder/Re-cap

- **P:** problems that can be solved in polynomial time, i.e., problems that can be solved/decided "efficiently"
- **NP:** broad set of problems that includes P;
  - answers can be verified "efficiently" (in polynomial time);
  - solutions cannot always be efficiently found (as far as we know).
- **NP-complete:** believed-to-be-hard decision problems in NP; they appear to have no efficient solution; answers are efficiently verifiable, solution to one is never much harder than a solution to another
- **NP-hard:** hardest; some of them may not be so
  TM. Many computational version of NP-complet
- **Examples:**
  - Factoring, discrete log are in NP, not known if NP-complete or P
  - Primality testing was "recently" (2002) shown to be in P
  - Knapsack is NP-complete

Relied upon by some of modern public key encryption schemes!

For more info, see: **https://www.nist.gov/dads//**

10

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 …

11

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

(assume Eve is active!)

Choose random u

K_{ae} K_{eb}

Choose random v



$y_a = a^v \bmod p$



$y_e = a^u \bmod p$

Secure communication using K_{ae}

Compute $K_{ae} = a^{vu} \bmod p$

Secure communication using K_{eb}

$y_b = a^w \bmod p$

y_e

Choose random w, Compute $K_{eb} = a^{uw} \bmod p$



12

RSA (1976-8)

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

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

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

$Secrets: p,q,d$

$Publics: n,e$

$Encryption: message = m < n$

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

$Decryption: ciphertext = y$

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

13

Why does it all work?

$x \in Z^*_n$

$x^{ed} =$ ← recall ed = 1 mode Φ(n)

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

But, recall that:

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

14

14

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}=

6.836761427754420001963955599558e+114 mod 221 = 5

15

15

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!

16

16

Recap: Complexity Reminder/Re-cap

- **P:** problems that can be solved in polynomial time, i.e., problems that can be solved/decided "efficiently"
- **NP:** broad set of problems that includes P;
  - answers can be verified "efficiently" (in polynomial time);
  - solutions cannot always be efficiently found (as far as we know).
- **NP-complete:** believed-to-be-hard decision problems in NP; they appear to have no efficient solution; answers are efficiently verifiable, solution to one is never much harder than a solution to another
- **NP-hard:** hardest; some of them may not be so
  TM. Many computational version of NP-complet

Relied upon by today's encryption schemes!

- **Examples:**
  - **Factoring**, discrete log are in NP, not known if NP-complete or P
  - Primality testing was "recently" (2002) shown to be in P
  - Knapsack is NP-complete

For more info, see: **https://www.nist.gov/dads//**

17

17

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 mod n, and c^d mod 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 … $O(b^2*2L)$

18

18

Square-and-Multiply

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

High-level idea:

To compute m^4

- No need to compute m → m^2 → m^3 → m^4
- Instead compute m → m^2 → m^4

19

19

Speeding up RSA Encryption/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:

$M = M_p \bmod p$

$M = M_q \bmod q$

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

20

20

More on RSA

- **Modulus n is unique per user →**
  - **2 or more parties cannot share the same 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), 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 = $(m)^{e"}$ mod n**
  - **Alice computes:**

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

21

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

22

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

23

23

Summary: Public Key Cryptography

- Diffie-Hellman (DH) key pre-distribution protocol
  - **Not for encryption by itself** → it's for establishing shared secret K_{ab}
    - With a shared secret key, can then use symmetric crypto like AES
  - Security measured by $|p|$, i.e., # of bits in p, usually 2048/3072/…
  - Interactive key exchange is vulnerable to **MitM attack**
- RSA encryption
  - Public exponent e is usually **short** (common ones: 3, 5, 17, 257 or 65537)
  - Private exponent d is usually **very long** → may it hard to easily decrypt a message by brute-forcing the key
  - Security measured by $|n|$, where $n = pq$, e.g., RSA-2048 means $|n| = 2048$
  - Performance improvement: **square-and-multiply** to reduce exponentiation cost, **CRT** to accelerate encryption/decryption
  - **Lack of randomness** → vulnerable to replay attack
- El Gamal encryption
  - Based on DH, but for encryption
  - Added randomness to improve the security against replay attacks

Remember AES key is 128/192/256 bits

24

24

Digital Signatures



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

25

25

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 it with symmetric key cryptography**
- Much **slower** than conventional cryptography
    - Often used together with conventional cryptography, e.g., to encrypt session keys

26

26

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}***

27