COMPUTER & NETWORK SECURITY

Lecture 9:

Number Theory for Public Key Crypto

INTRO TO NUMBER THEORY (DH & RSA)

Motivation

A (very) brief introduction to number theory

Understand the security of Diffie-Hellman

Understand asymmetric cryptography (e.g. RSA)

Core concept in number theoretic cryptography:

Find a number theoretic problem that's incredibly difficult to solve if you don't have a key piece of information

Notation

Z	the set of all integers
p, **q**	will be reserved for prime numbers

BACKGROUND: ORIGINS OF PUBLIC KEY CRYPTOGRAPHY

1976:	**Diffie Hellman & Merkle (Stanford) invented concept seperately**
1976:	**RSA invented (Rivest, Shamir, Adelman)**
1973:	**Clifford Cocks wrote a variant of RSA (GCHQ) [kept secret until 1997]**
1967:	**James Ellis (GCHQ) proved public key possible**

Even earlier:

NSA claims they invented public key cryptography earlier

– Jim Frazer (NSA - retired) claims a 1962 NSA memo on command & control of nuclear weapons was the basis for it’s invention.

– Side note: the STU-III secure telephone used PKI in the mid 1970’s, well before certificates were in the civilian world.

Black sector (esp. the NSA) is way ahead of civilian crypto

– Many claim at least 20 years

PUBLIC KEY CRYPTOGRAPHY (ASYMMETRIC CRYPTOGRAPHY)

Each entity has a *public key* **e** and a *private key* **d**.

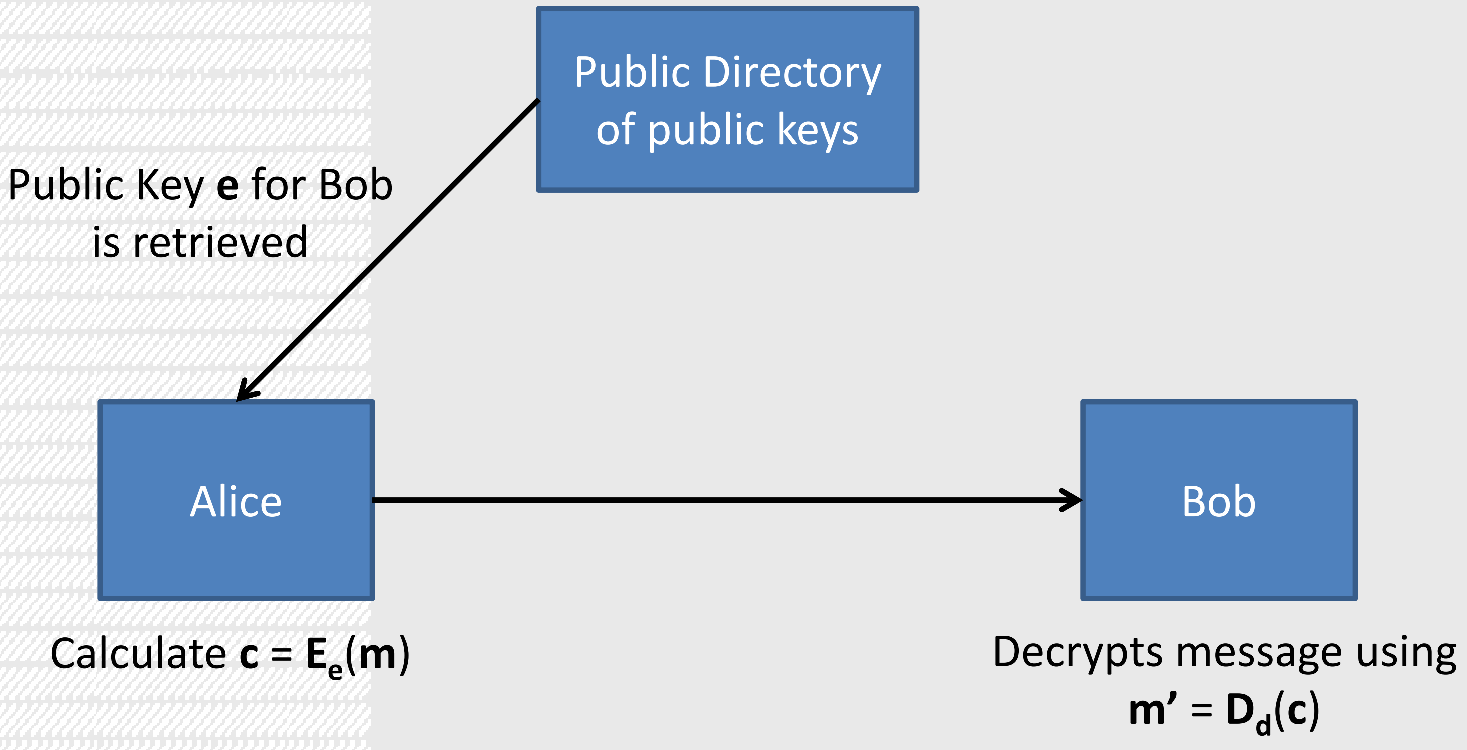
Public key is used for **e**ncryption, private key is used for **d**ecryption.

The public key and algorithm need not be secret, can be published openly.

Public key encryption alone provides confidentiality, not data origin authentication nor integrity (as encryption key and algorithm are known).

Public key decryption can be used to provide authentication guarantees.

PUBLIC KEY EXAMPLE


Public Directory of public keys
Public Key **e** for Bob is retrieved
Alice
Bob
Calculate **c** = $\mathbf{E_e}(\mathbf{m})$
Decrypts message using
$\mathbf{m'} = \mathbf{D_d}(\mathbf{c})$

Note: encryption key **e** and decryption key **d** are two different entities

PROBLEMS WITH PUBLIC KEY CRYPTO

Based on known algorithms, public key cryptography is much slower than private key cryptography

– Public key crypto often used to negotiate keys used for bulk data encryption by symmetric algorithms (commonly known as session keys)

– Public key crypto is also used to encrypt small things (e.g. credit card transactions, PINs, etc.)

Unless the public keys are verified, public key systems are vulnerable to impersonation attacks

– How do we really know the public key we retrieved is actually Alice’s?

Since an attacker knows the encryption algorithm and key, they can perform a chosen plaintext attack

– Anyone can send a message to a receiver; there is no authentication of the message source

RSA

Generate two large random, distinct primes **p** and **q** (**p**, **q** > 1024 bits)

Compute **n** = **pq**

The order of $\mathbf{Z_n}$ = $\mathbf{\Phi}$(**n**) = $\mathbf{\Phi}$(**p**) $\mathbf{\Phi}$(**q**) = (**p**-1)(**q**-1)

Remember: $\mathbf{\Phi}$(**n**) indicates the number of elements in $\mathbf{Z_n}$ that have a multiplicative inverse

We want to select **e** and **d** (the multiplicative inverse of **e**) such that:

$$\mathbf{m}^{ed} \bmod \mathbf{n} \equiv \mathbf{m}$$

The encryption exponent **e** will be part of the public key

The decryption exponent **d** will be the private key

Public Key: (n, e)

Private Key: d

p, q and $\mathbf{\Phi}$(**n**) can be discarded after key generation

RSA: SELECTING E AND D

We want to select **e** and **d** (the multiplicative inverse of **e**) such that:

$$\mathbf{m}^{\mathbf{ed}} \bmod \mathbf{n} \equiv \mathbf{m}$$

We know that we want $\mathbf{m}^{\mathbf{ed}} \bmod \mathbf{n} \equiv \mathbf{m}^{1} \bmod \mathbf{n} \equiv \mathbf{m}$

Euler's Theorem: $\mathbf{x}^{\Phi(\mathbf{n})} = 1 \bmod \mathbf{n}$ (where **x** is any value)

Therefore, we need $\mathbf{ed} = \mathbf{k}\,\mathbf{\Phi}(\mathbf{n}) + 1$

Why? Imagine $\mathbf{ed} = \mathbf{2}\,\mathbf{\Phi}(\mathbf{n}) + 1$, then...

$\mathbf{m}^{\mathbf{2}\,\Phi(\mathbf{n}) + 1} \bmod \mathbf{n} \equiv \mathbf{m}^{\Phi(\mathbf{n})} \times \mathbf{m}^{\Phi(\mathbf{n})} \times \mathbf{m}^{1} \bmod \mathbf{n} \equiv 1 \times 1 \times \mathbf{m} \bmod \mathbf{n} \equiv \mathbf{m}$

RSA: TINY EXAMPLE

Select two primes **p**, **q** = 11, 17

This makes **n** = 11 x 17 = 187

The order of $\mathbf{Z_{187}}$ = **Φ**(**187**) = **Φ**(**11**) **Φ**(**17**) = (**11**-1)(**17**-1) = 10 x 16 = 160

Let's say we select **e** = 7

Select d such that **ed** = **k Φ**(**n**) + 1 = 160 + 1 = 161

Note that 7 x 23 = 161, therefore we'll select **d** = 23

Bob wants to send you a message **m** = 32 using this system...

c = $\mathbf{m}^e$ mod **n** = 32^7 mod 187 = pow(32, 7, 187) = 76

You want to decrypt the message sent by Bob...

m = $\mathbf{c}^d$ mod **n** = 76^{23} mod 187 = pow(76, 23, 187) = 32

THE SECURITY OF RSA: CALCULATING IN Z_N

Let's create $\mathbf{Z_n}$, which is a multiplicative group modulo **n** where **n = pq**

Easy:

- Generate a random number
- Add and multiply
- Compute $\mathbf{g^r \bmod p}$
- Inverting an element
- Solving a linear system

Hard:

- Finding the prime factors of **n**
- Computing the e^{th} root (as hard as factoring **n**)
- Solving polynomial equation of degree **d**

THE CORE SECURITY OF RSA

The RSA problem

Recovering the value **m** from $\mathbf{c} \equiv \mathbf{m}^{\mathbf{e}} \pmod{\mathbf{n}}$

(i.e. Taking the e^th^ root of **m** modulo **n**)

This is hard: the most efficient means so far known is to factor **n** into **p**, **q**

Integer Factorization

Given a composite number **n** turn it into the prime factors $\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_n$

No polynomial time algorithm is yet known

Note: Like DLP and DHP from DH key exchange, we assume the RSA and integer factorization problems are computationally "hard" but have no proofs. There also exist quantum algorithms that would easily break RSA.

RSA OVERVIEW

Encryption

Obtain Alice's public key (**n**, **e**)

Represent the message as an integer in $\mathbf{Z_n}^*$ = {0, 1, 2, 3, ..., n-1}

Compute $\mathbf{c} = \mathbf{m}^e \bmod \mathbf{n}$

Send cyphertext **c** to Alice

Decryption

Alice computes

$$\begin{aligned} \mathbf{c}^d \bmod \mathbf{n} &= (\mathbf{m}^e)^d \bmod \mathbf{n} \\ &= \mathbf{m}^{ed} \bmod \mathbf{n} \\ &= \mathbf{m}^{1+k\Phi(n)} \bmod \mathbf{n} \\ &= \mathbf{m}^1 \times (\mathbf{m}^{\Phi(n)})^k \bmod n \\ &= \mathbf{m} \times (1 \times 1 \times \ldots \times 1) \bmod \mathbf{n} \\ &= \mathbf{m} \end{aligned}$$

EL GAMAL CRYPTOSYSTEM (TURNING DH INTO PUBLIC KEY CRYPTO)

Alice selects **g**, **p** and a *private key* **a**

She then publishes the *public key* $\mathbf{A = g^a \bmod p}$ as well as **g, p**

Bob wants to send Alice a message **m** and so selects **k** and generates:

$$\mathbf{B_1 = g^k} \bmod p \qquad \text{and} \qquad \mathbf{B_2 = A^k\, m} \bmod p = \mathbf{(g^a)^k\, m} \bmod \mathbf{p}$$

$$\mathbf{B_1^{-a}\, B_2} \bmod \mathbf{p} \equiv \mathbf{g}^{-ak}\, \mathbf{A}^k\, \mathbf{m} = \mathbf{g}^{-ak}\, (\mathbf{g^a} \bmod \mathbf{p})^k\, \mathbf{m} \equiv \mathbf{g}^{-ak}\, \mathbf{g^{ak}}\, \mathbf{m} = \mathbf{m} \bmod \mathbf{p}$$

Security

Alice is the only person who knows **a** and can hence compute **inv(a)**

An attacker would need to find **a** or **k** using **A** or $\mathbf{B_1}$ (i.e. discrete log problem)

EXAMPLE OF EL GAMAL CRYPTOSYSTEM (NOT FINISHED)

Alice selects **g**, **p** = 2, 19 and decides on *private key* **a** = 8

She publishes the *public key* **A = 9 (**g^a mod **p = 2**^{8} mod **19** = **9)** as well as **g**, **p**

Bob wants to send Alice a message **m** = 13, secret key **k** = 7 and so generates:

$$\mathbf{B_1} = \mathbf{g}^k \text{ mod } \mathbf{p} = \mathbf{2}^7 \text{ mod } \mathbf{19} = 14$$

$$\mathbf{B_2} = \mathbf{A}^k\, \mathbf{m} \text{ mod } p = (\mathbf{g}^a)^k\, \mathbf{m} \text{ mod } \mathbf{p} = 9$$

$\mathbf{B_1}^{-a}\, \mathbf{B_2} \text{ mod } \mathbf{p} \equiv \mathbf{g}^{-ak}\, \mathbf{A}^k\, \mathbf{m} = \mathbf{g}^{-ak}\, (\mathbf{g}^a \text{ mod } \mathbf{p})^k\, \mathbf{m} \equiv \mathbf{g}^{-ak}\, \mathbf{g}^{ak}\, \mathbf{m} = \mathbf{m} \text{ mod } \mathbf{p}$

Security

Alice is the only person who knows **a** and can hence compute **inv(a)**

An attacker would need to find **a** or **k** using **A** or $\mathbf{B_1}$ (i.e. discrete log problem)

REFERENCES

Handbook of Applied Cryptography

– read §1

– read §3.1

– skim §3.2 - §3.2.3

– read §3.6 - §3.6.2

– skim §3.6.3

– read §8-8.2, 8.4