Whitfield Diffie
and
Martin E. Hellman
(Recipients of the 2015 Turing Award)

CS 553
CRYPTOGRAPHY

Lecture 23
Public Key Cryptography

Instructor
Dr. Dhiman Saha

Image Source: Google

Diffie & Hellman's Pioneering Paper

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of me-

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks

The Problem with Symmetric Encryption

How to exchange the **secret** key?

If Alice and Bob want to exchange messages using a **symmetric** cipher, they must first **mutually agree on a secret** key k. This is fine if they have the opportunity to meet in secret or if they are able to communicate once over a secure channel.

- But what if they do not have this opportunity
- What if every communication between them is monitored by their adversary Eve?
- Is it possible for Alice and Bob to exchange a secret key under these conditions?

The search for efficient (and provable) solutions to this problem, which is called **public key** (or **asymmetric**) cryptography

Other Issues with Symmetric Crypto

- Number of Keys
- No Protection Against Cheating by Alice or Bob
  - How to ensure **non-repudiation**?

History of PKC

- In 1976, Whitfield Diffie and Martin Hellman published their now famous paper entitled **New Directions in Cryptography**
- Ralph Merkle's undergraduate project in a computer science class at Berkeley
- Merkles work **Secure communication over insecure channels** appeared in 1982
- In 1997 British documents which were **declassified**
- Revealed that the researchers James Ellis, Clifford Cocks and Graham Williamson from the UK Government Communications Headquarters (GCHQ) discovered and realized the principle of public-key cryptography a few years earlier, in 1972.

- Key-space $\mathcal{K}$,
- Plaintexts-space $\mathcal{M}$, and
- Ciphertexts-space $\mathcal{C}$
- An element k of the key-space is really **a pair of keys**,

$$k = (k_{pr}, k_{pub})$$

called the **private key** and the **public key**, respectively.

- For each public key k_{pub} there is a corresponding **encryption** function

$$e_{k_{pub}} : \mathcal{M} \to \mathcal{C}$$

- For each private key k_{pr} there is a corresponding **decryption** function

$$d_{k_{pr}} : \mathcal{C} \to \mathcal{M}$$

- These have the property that if the pair (k_{pr}, k_{pub}) is in the key-space $\mathcal{K}$, then

$$d_{k_{pr}}(e_{k_{pub}}(m)) = m \quad \text{for all } m \in \mathcal{M}$$

Basic protocol for public-key encryption

Alice

$y = e_{k_{pub}}(x)$

k_{pub} ←

→ y

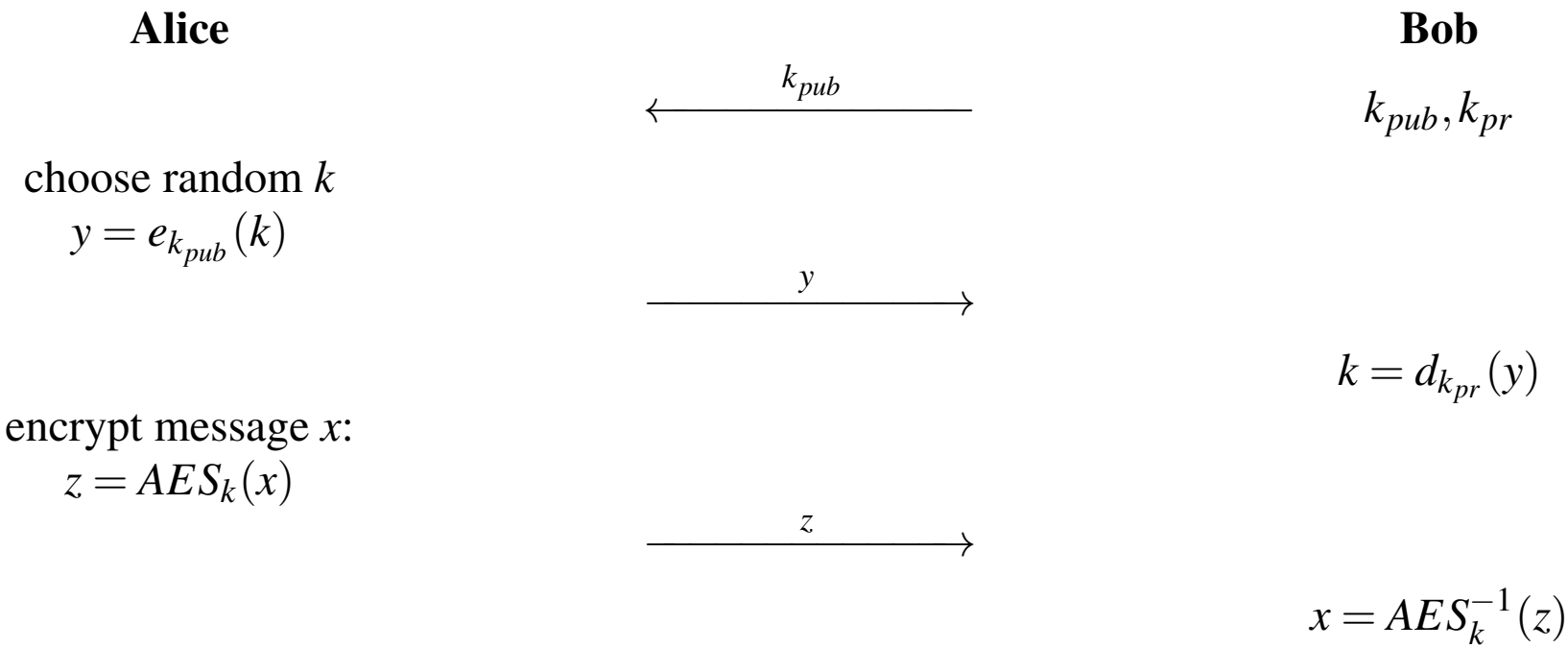
Bob

$(k_{pub}, k_{pr}) = k$

$x = d_{k_{pr}}(y)$

Real-life Analogy

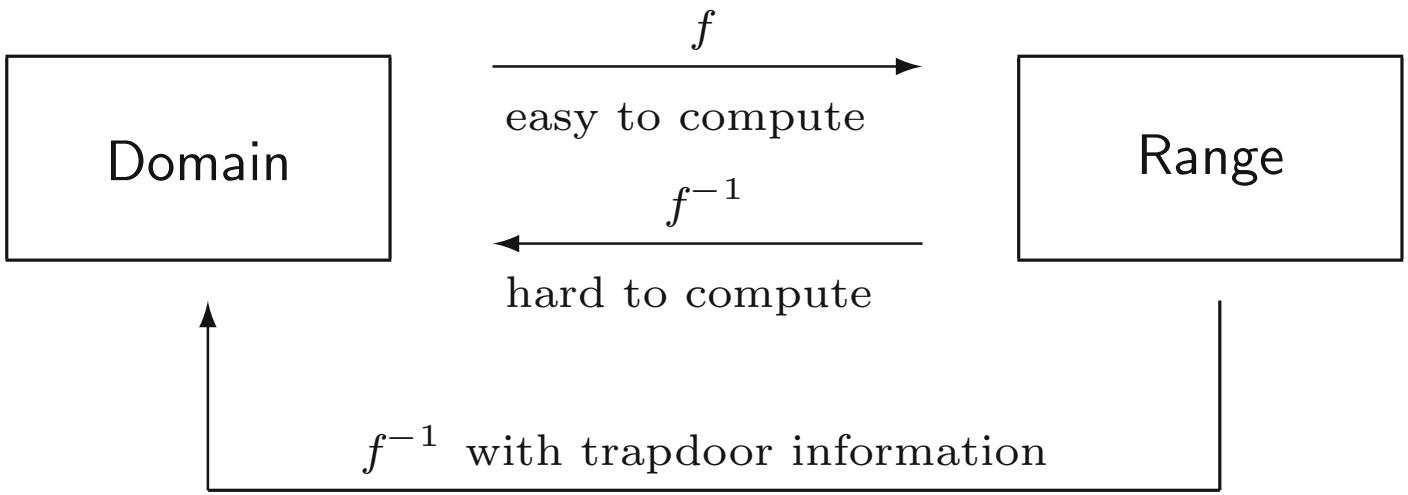
Basic key transport protocol with AES



What is the advantage of this setting?

The question:
How one can build public-key algorithms?

Using One-way Trapdoor Function



- Secure PKCs are built using one-way functions that have a trapdoor.
- The trapdoor is a piece of auxiliary information that allows the inverse to be easily computed.

Two Popular OWF

The first is the integer factorization problem

- Given two large primes, it is easy to compute the product.
- However, it is very difficult to factor the resulting product.

- What is the trapdoor here?

- The other one-way function that is used widely is the discrete logarithm problem.

- Key Establishment
  - DHKE/RSA Transport Protocols
- Non-repudiation
  - Digital Signature Algorithms - RSA DSA ECDSA
- Identification
  - Challenge Response Protocols - Smart Cards.
- Encryption
  - RSA Elgamal

Important Public-Key Algorithms

- Integer-Factorization Schemes
  - RSA
- Discrete Logarithm Schemes
  - DH Key Exchange
  - DSA
  - Elgamal Encryption
- Elliptic Curve (EC) Schemes
  - ECDHKE
  - ECDSA
- Other (not so popular) schemes
  - Multi-variate Crypto
  - Lattice-based Schemes
  - Hyperelliptic Cryptosystems

Key-Length/Security Level Comparison

Algorithm Family	Cryptosystems	Security Level (bit)			
		80	128	192	256
Integer factorization	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Discrete logarithm	DH, DSA, Elgamal	1024 bit	3072 bit	7680 bit	15360 bit
Elliptic curves	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetric-key	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Bit lengths of public-key algorithms for different security levels

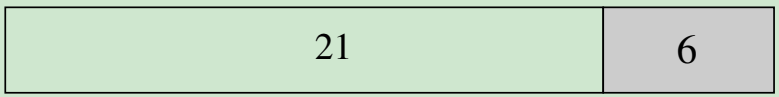
Essential Number Theory

Euclidean GCD

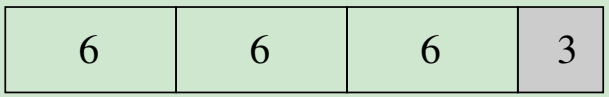
Reduce the problem of finding the gcd of two given numbers to that of the gcd of two **smaller** numbers.

$$gcd(r_0, r_1) = gcd(r_1, r_0 \bmod r_1).$$

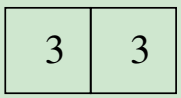
Example



$\gcd(27, 21) = \gcd(1 \cdot 21+6, 21) = \gcd(21, 6)$



$\gcd(21, 6) = \gcd(3 \cdot 6+3, 6) = \gcd(6, 3)$



$\gcd(6, 3) = \gcd(2 \cdot 3+0, 3) = \gcd(3, 0) = 3$

$\gcd(27, 21) = \gcd(21, 6) = \gcd(6, 3) = \gcd(3, 0) = 3$

The Algorithm

Euclidean Algorithm
Input: positive integers r_0 and r_1 with $r_0 > r_1$
Output: $\gcd(r_0, r_1)$
Initialization: $i = 1$
Algorithm:

1 DO
1.1 $i = i + 1$
1.2 $r_i = r_{i-2} \bmod r_{i-1}$
WHILE $r_i \neq 0$
2 RETURN
$\gcd(r_0, r_1) = r_{i-1}$

An extension of the algorithm allows us to compute modular inverses, which is of major importance in public-key cryptography.

- In addition to computing the gcd, EEA computes a linear combination of the form:

$$gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$$

- Also referred to as **Diophantine** equation.

HomeWork

How EEA is used to get modular inverse?

Extended Euclidean Algorithm (EEA)
Input: positive integers r_0 and r_1 with $r_0 > r_1$
Output: $\gcd(r_0, r_1)$, as well as s and t such that $\gcd(r_0, r_1) = s \cdot r_0 + t \cdot r_1$.
Initialization:
$s_0 = 1 \quad t_0 = 0$
$s_1 = 0 \quad t_1 = 1$
$i = 1$
Algorithm:

1 DO
1.1 $i = i + 1$
1.2 $r_i = r_{i-2} \bmod r_{i-1}$
1.3 $q_{i-1} = (r_{i-2} - r_i)/r_{i-1}$
1.4 $s_i = s_{i-2} - q_{i-1} \cdot s_{i-1}$
1.5 $t_i = t_{i-2} - q_{i-1} \cdot t_{i-1}$
WHILE $r_i \neq 0$
2 RETURN
$\gcd(r_0, r_1) = r_{i-1}$
$s = s_{i-1}$
$t = t_{i-1}$

Euler's Phi Function

$\Phi(m)$

The number of integers in $\mathbb{Z}_m$ relatively prime to m

- Computing Eulers phi function in this **naive** way is completely out of reach for the large numbers occurring in public-key cryptography.

$\Phi(m)$

The number of integers in $\mathbb{Z}_m$ relatively prime to m

- Computing Eulers phi function in this **naive** way is completely out of reach for the large numbers occurring in public-key cryptography.

Theorem 6.3.1 *Let m have the following canonical factorization*

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n},$$

where the p_i are distinct prime numbers and e_i are positive integers, then

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1}).$$

Fermat's Little Theorem

Let a be an integer and p be a prime, then:

$$a^p \equiv a (\mathrm{mod}\, p)$$

- Alternative form:

$$a^{p-1} \equiv 1 (\mathrm{mod}\, p)$$

- Useful in crypto
- Can used in computing modular inverses. How?

Euler's theorem

A **generalization** of Fermat's Little Theorem to any integer moduli

- Moduli that are not necessarily primes

Let a and m be integers with $gcd(a, m) = 1$, then:

$$a^{\Phi(m)} \equiv 1 (\mathrm{mod}\, m)$$