CISC 468: CRYPTOGRAPHY

LESSON 8: INTRODUCTION TO PUBLIC-KEY CRYPTOGRAPHY

Furkan Alaca

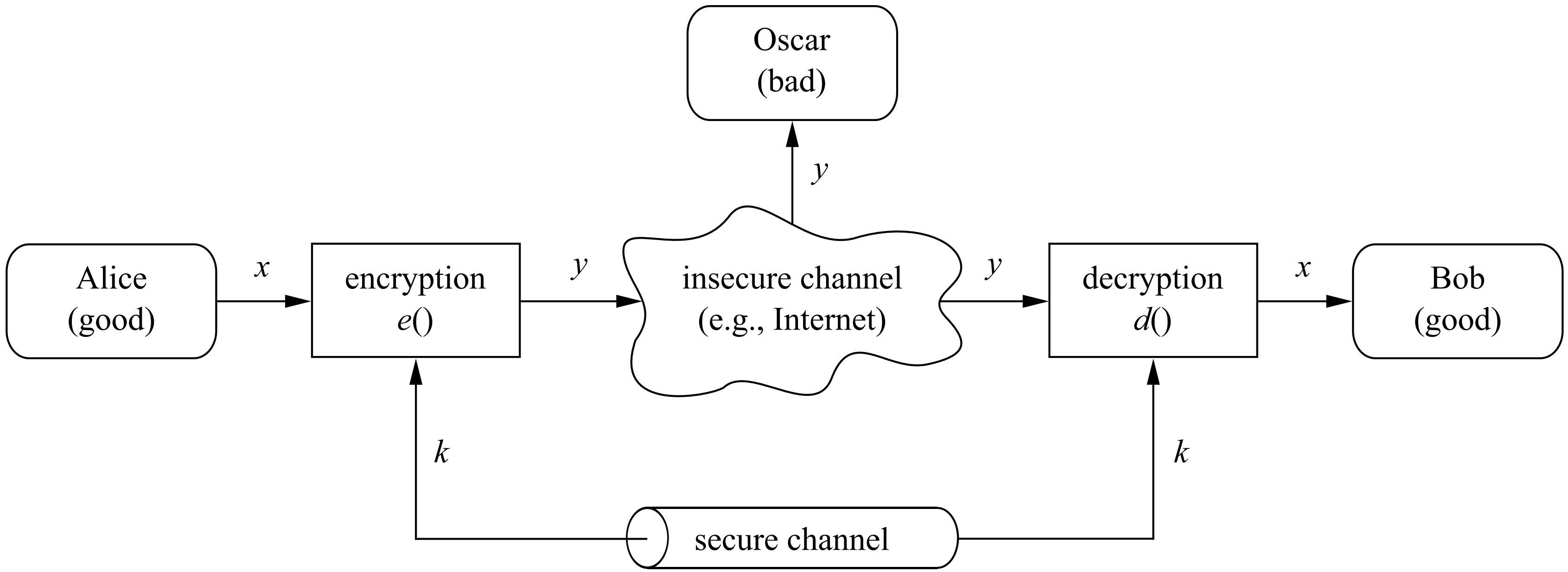
TODAY, WE WILL LEARN ABOUT...

- Public-key cryptography: Basic concepts and use cases
- Review some number theory

READINGS

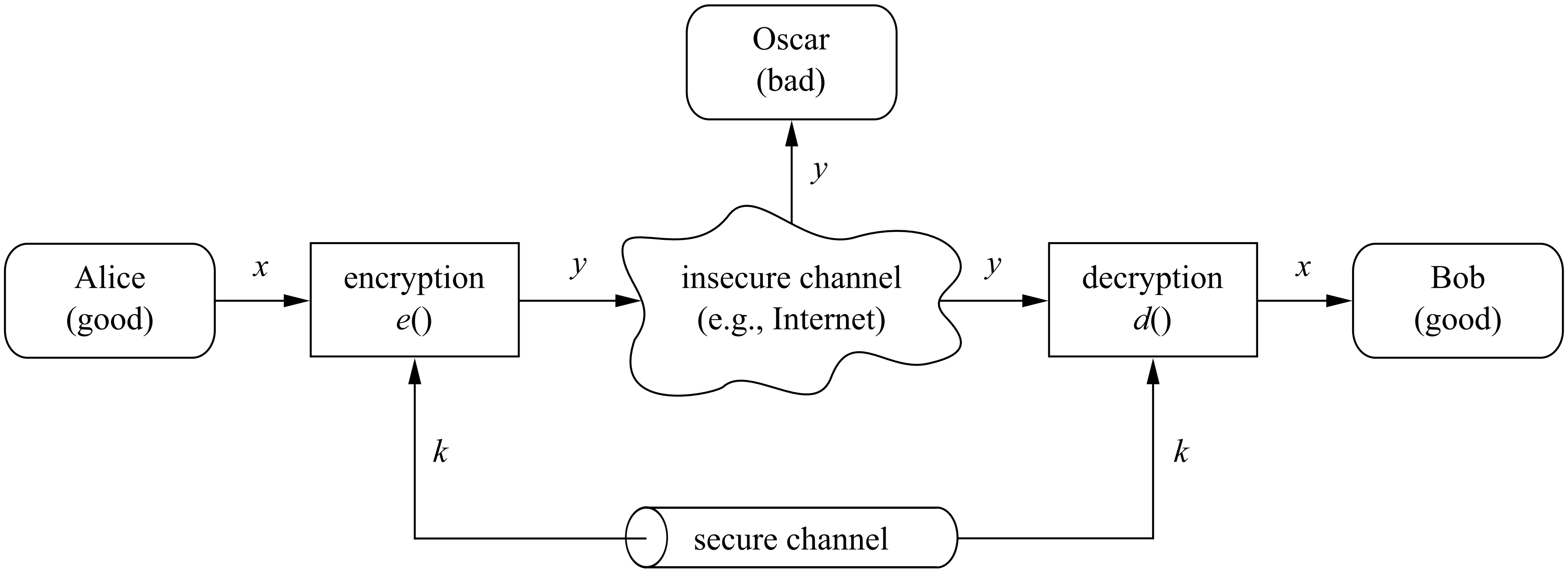
- Section 6.1: Symmetric vs. Asymmetric Cryptography, Paar & Pelzl
- Section 6.2: Practical Aspects of Public-Key Cryptography, Paar & Pelzl
- Section 6.3: Essential Number Theory for Public-Key Algorithms, Paar & Pelzl

SYMMETRIC CRYPTOGRAPHY REVISITED



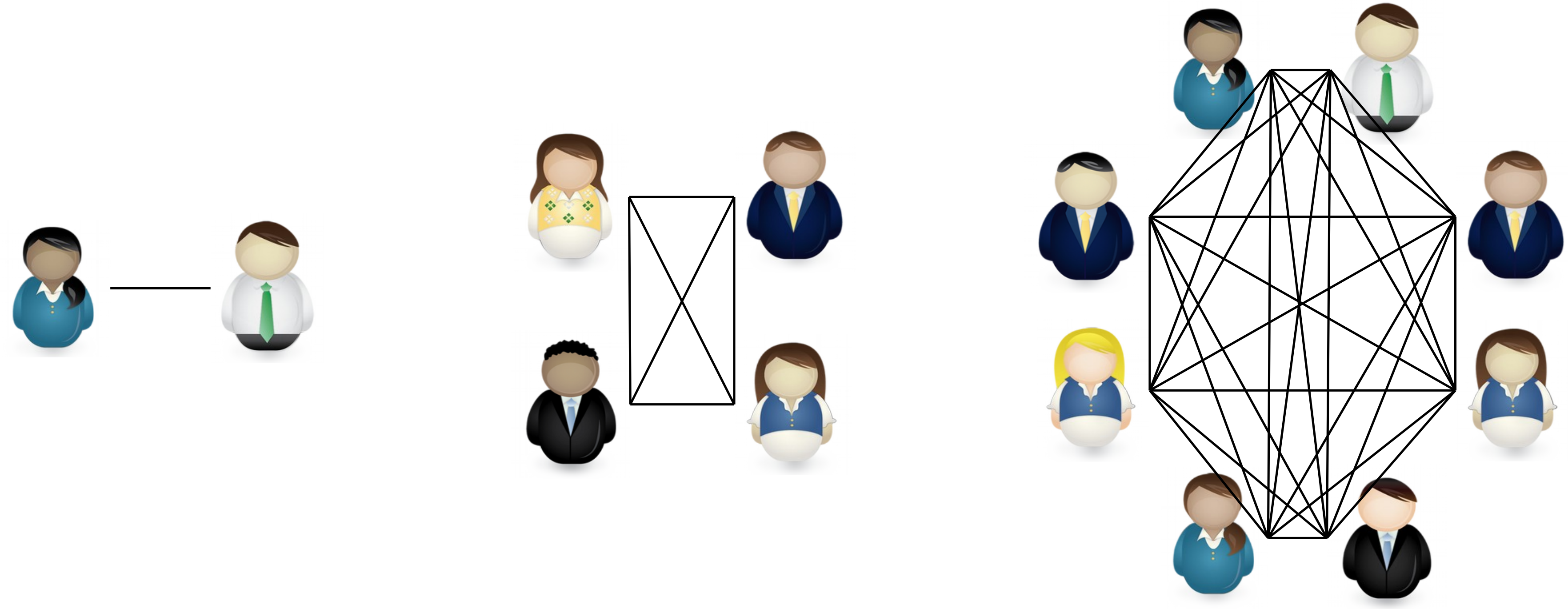
1. The same secret key is used for encryption and decryption.
2. The enryption and decryption functions are very similar (in DES they are essentially identical).

SYMMETRIC CRYPTOGRAPHY: KEY DISTRIBUTION PROBLEM



Key must be established using a secure channel, *out-of-band*

SYMMETRIC CRYPTOGRAPHY: NUMBER OF KEYS



1 key   6 keys   28 keys

Each pair of communicating parties requires a unique key;

$\frac{n(n-1)}{2}$ keys for n users: $\mathcal{O}(n^2)$

SYMMETRIC CRYPTOGRAPHY: NO PROTECTION AGAINST CHEATING

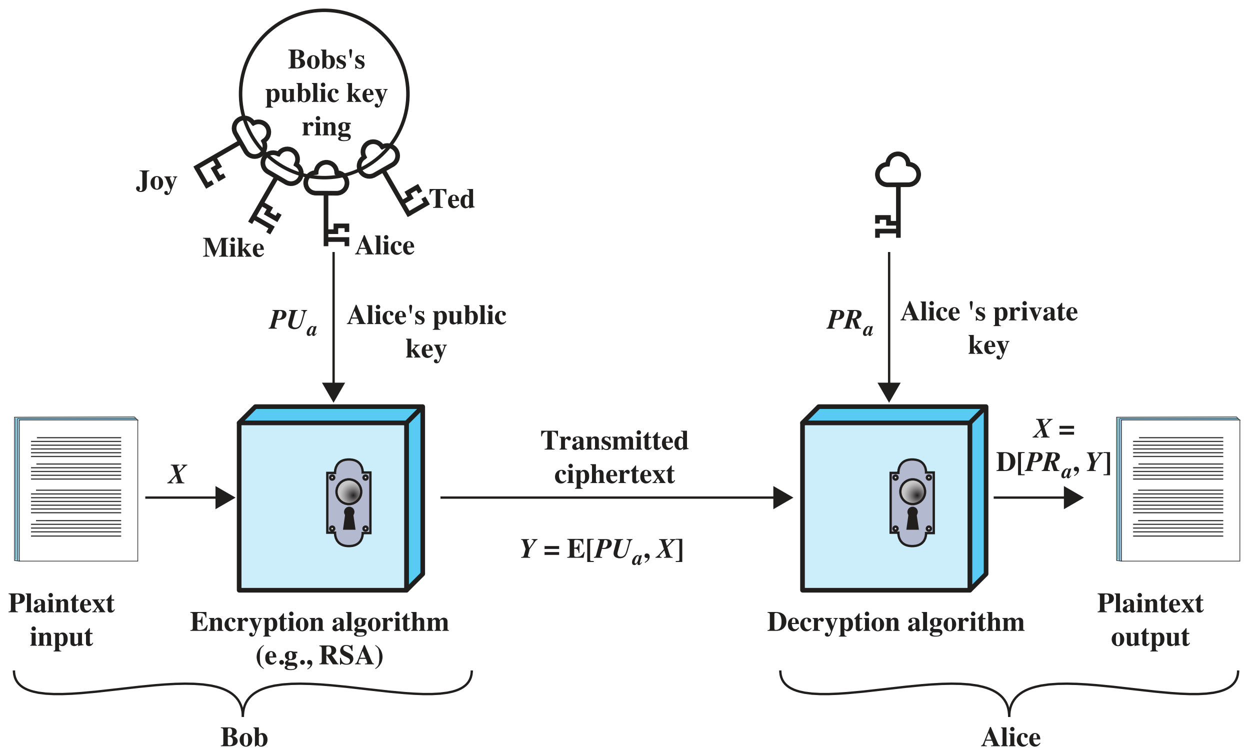
- Sender and receiver share the same key, so, either party is capable of using it to perform any cryptographic operations
  - e.g., encryption, decryption, message integrity code
- Problem: Alice may generate a transaction to purchase goods from Bob, but then later change her mind and claim that Bob falsified the transaction for financial gain
  - Digital signatures, which we will study later, address this issue by providing *nonrepudiation*

ASYMMETRIC CRYPTOGRAPHY: AN ANALOGY

- Imagine that each communicating party has a mailbox
- Anybody has the ability to put a letter into any mailbox
  - To foreshadow an upcoming problem we will face: When putting a message into a mailbox, how do you know who the mailbox belongs to?
- But only the owner of each mailbox has the key to open the mailbox and retrieve its contents
- *Asymmetric cryptography*, also referred to as *public-key cryptography*, can achieve a system that works similarly

PUBLIC-KEY ENCRYPTION AND DECRYPTION

- Each party requires a *private key* and a *public key*
- Sender encrypts plaintext with receiver's public key
- Receiver decrypts ciphertext with own private key



HYBRID ENCRYPTION

- Asymmetric cryptography can be used to securely transport a symmetric key (e.g., for AES) to be used as a *session key*
- This is useful because symmetric cryptography is significantly faster than asymmetric cryptography

Alice		**Bob**
	$\xleftarrow{\quad k_{pub} \quad}$	k_{pub}, k_{pr}
choose random k $y = e_{k_{pub}}(k)$		
	$\xrightarrow{\quad y \quad}$	
		$k = d_{k_{pr}}(y)$
encrypt message x: $z = AES_k(x)$		
	$\xrightarrow{\quad z \quad}$	
		$x = AES_k^{-1}(z)$

ASYMMETRIC CRYPTOGRAPHY: ONE-WAY FUNCTIONS

- Public-key algorithms are built on one-way functions, where
  - $y = f(x)$ is computationally easy to compute, and
  - $x = f^{-1}(y)$ is computationally infeasible to compute
- Computationally "easy" functions can be evaluated in polynomial time
- Computationally "infeasible" functions should not be possible to compute in any "reasonable" time period, e.g., 10,000 years
- Examples: Integer factorization and discrete logarithm problem

PUBLIC-KEY ALGORITHMS: MAIN SECURITY MECHANISMS

- *Key Establishment* over an insecure channel to perform symmetric encryption
- *Nonrepudiation* and *message integrity* provided by digital signature algorithms
- *Identification* of communicating entities such as websites, smart payment cards, and USB authentication tokens
- *Encryption* is typically done on small amounts of data, due to speed limitations of public-key algorithms

KEY LENGTHS AND SECURITY LEVELS

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		128 bit	192 bit	256 bit

- Public-key algorithms require longer keys than symmetric-key algorithms to provide equivalent security
- NIST SP 800-131A Rev. 2 requires the use of keys offering ≥ 112 bits of security strength

RSA: NUMBER THEORY BACKGROUND

- We will review some number theory background (covered in CISC 203), which are important for understanding RSA:
  - Euclidean Algorithm
  - Euler's phi function
  - Fermat's Little Theorem
  - Euler's Theorem

NUMBER THEORY: GREATEST COMMON DIVISORS

- The greatest common divisor of two positive integers r_0 and r_1 is denoted by

$$\gcd(r_0, r_1)$$

and is the largest positive number that divides both r_0 and r_1.

NUMBER THEORY: GREATEST COMMON DIVISORS (CONT'D)

- Example: Let $r_0 = 84$ and $r_1 = 30$. Factoring yields

$$r_0 = 84 = 2 \times 2 \times 3 \times 7$$
$$r_1 = 30 = 2 \times 3 \times 5$$

The gcd is the product of all common prime factors:

$$\gcd(30, 84) = 2 \times 3 = 6.$$

NUMBER THEORY: GREATEST COMMON DIVISORS (CONT'D)

- If $r_0 > r_1$ we have

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

- It follows that

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

- For example,

$$\begin{aligned}\gcd(973, 301) &= \gcd(973 \bmod 301, 301)\\ &= \gcd(70, 301) = \gcd(301, 70).\end{aligned}$$

NUMBER THEORY: EUCLIDEAN ALGORITHM

The Euclidean Algorithm computes $\gcd(a, b)$ very efficiently:

1. Let $c = a \bmod b$.
2. If $c = 0$, then the answer is b.
3. Otherwise (i.e., if $c \neq 0$), the answer is $\gcd(b, c)$.

Note that this algorithm is recursive.

NUMBER THEORY: EUCLIDEAN ALGORITHM EXAMPLE (1)

- Let $r_0 = 973$ and $r_1 = 301$. Their gcd is computed as follows:

$$\begin{aligned} 973 \bmod 301 &= 70 \\ 301 \bmod 70 &= 21 \\ 70 \bmod 21 &= 7 \\ 21 \bmod 7 &= 0 \end{aligned}$$

So, $\gcd(973, 301) = 7$.

NUMBER THEORY: EUCLIDEAN ALGORITHM EXAMPLE (2)

- Let $a = 1205$ and $b = 37$. Their gcd is computed as follows:

$a = q \times b + r$	$\gcd(a, b) = \gcd(b, a \bmod b)$
$1205 = 37 \times 32 + 21$	$\gcd(1205, 37) = \gcd(37, 21)$
$37 = 21 \times 1 + 16$	$\gcd(37, 21) = \gcd(21, 16)$
$21 = 16 \times 1 + 5$	$\gcd(21, 16) = \gcd(16, 5)$
$16 = 5 \times 3 + 1$	$\gcd(16, 5) = \gcd(5, 1)$
$5 = 5 \times 1 + 0$	$\gcd(5, 1) = \gcd(1, 0) = 1.$

- In this example, since $\gcd(a, b) = 1$, we call a and b *relatively prime*

NUMBER THEORY: COMPUTING MODULAR INVERSES

- When doing modular arithmetic in $\mathbb{Z}_n$, we often need to compute the inverse of an element
  - e.g., $5^{-1} = 7$ in $\mathbb{Z}_{19}$ since $5 \otimes 7 = 1$
- Extending the Euclidean Algorithm allows us to compute modular inverses, which is of major importance in public-key cryptography
- First we compute $\gcd(a, b)$ in the form

$$\gcd(a, b) = s \cdot a + t \cdot b$$

where s and t are integer coefficients.

NUMBER THEORY: COMPUTING MODULAR INVERSES EXAMPLE

Using the Euclidean algorithm, we can find that

$$\gcd(1205, 37) = 1 = 37(228) + 1205(-7).$$

Then, taking $\bmod 1205$ of both sides:

$$\begin{aligned} 1 \bmod 1205 &= [37(228) + 1205(-7)] \bmod 1205 \\ &= 37(228) \bmod 1205 + 1205(-7) mod 1205 \\ &= 37(228) \bmod 1205 \end{aligned}$$

So, $37^{-1} = 228$.