CS326 – Systems Security

Lecture 8

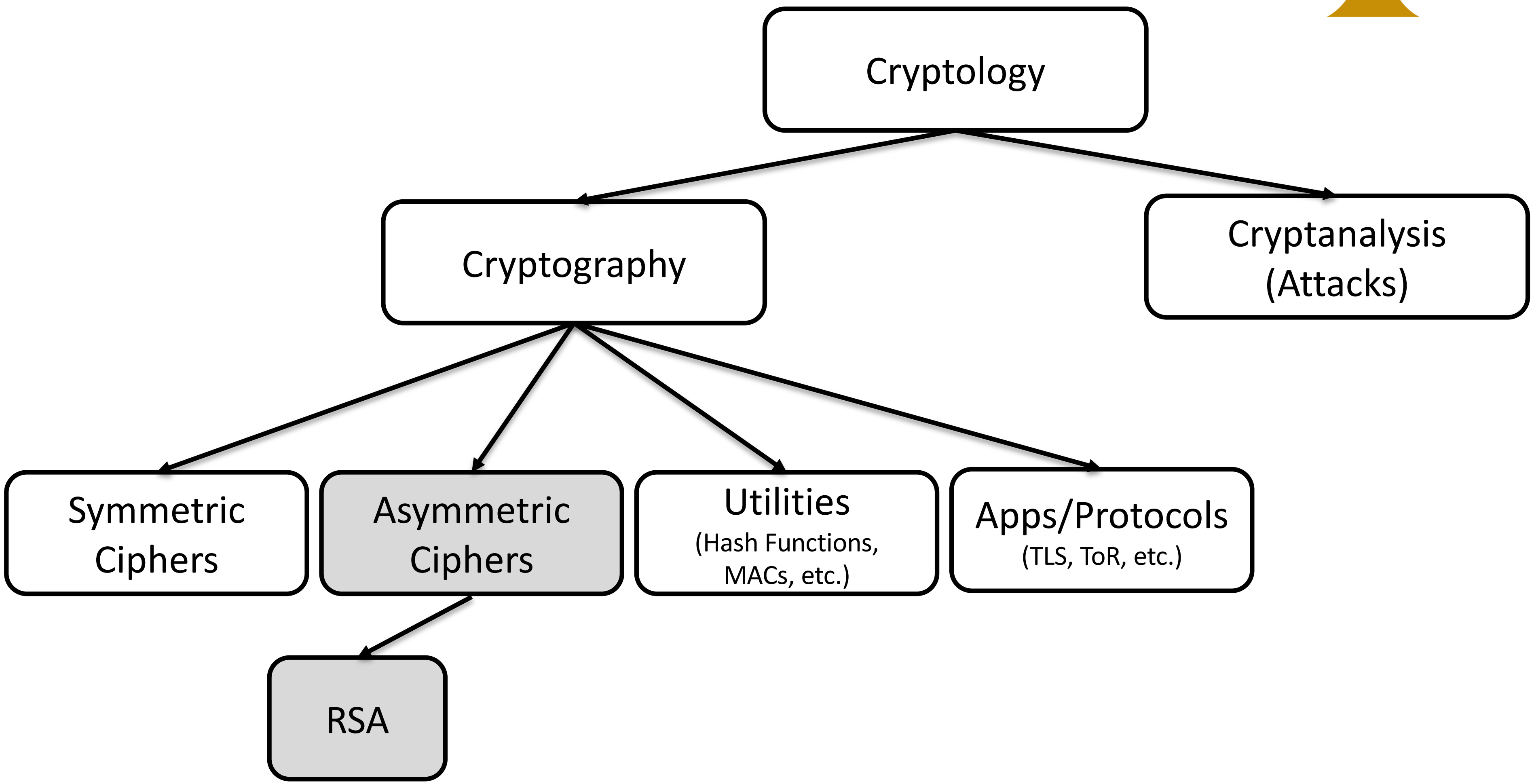
Asymmetric Encryption and RSA

Elias Athanasopoulos
athanasopoulos.elias@ucy.ac.cy

Sections of this Lecture

- RSA
- Public-key Encryption
- Implementation/Security

Cryptography Roadmap



RSA

Recipe 1/3

- Suppose you want to encrypt the message: 2
  - Let's say that A maps to 0, B maps to 1, and C maps to 2; you want to map C to another letter
- Pick two prime numbers

  $p = 2$ and $q = 11$
- Multiply them

  $n = pq = 2 \cdot 11 = 22$

Recipe 2/3

- Calculate $\phi(n)$, or $\phi(22)$

  $\phi(n) = (p-1)(q-1) = (2-1)(11-1) = 10$
- Pick a number that is relative prime to 10, greater than 1 and smaller than 10

  $e = 3$
- Solve the equation using the Extended Euclidean Algorithm: $x \cdot 3 \equiv 1 \pmod{10}$

  Find an integer x that if multiplied with 3 the result is 1 mod 10

  $x = 7$, because $7 \cdot 3 \pmod{10} = 1 \pmod{10}$

  let's call that $d = 7$

Recipe 3/3

- For encryption

  2^3 mod 22 = 8 mod 22 = 8

  (so 2 becomes 4)

- For decryption

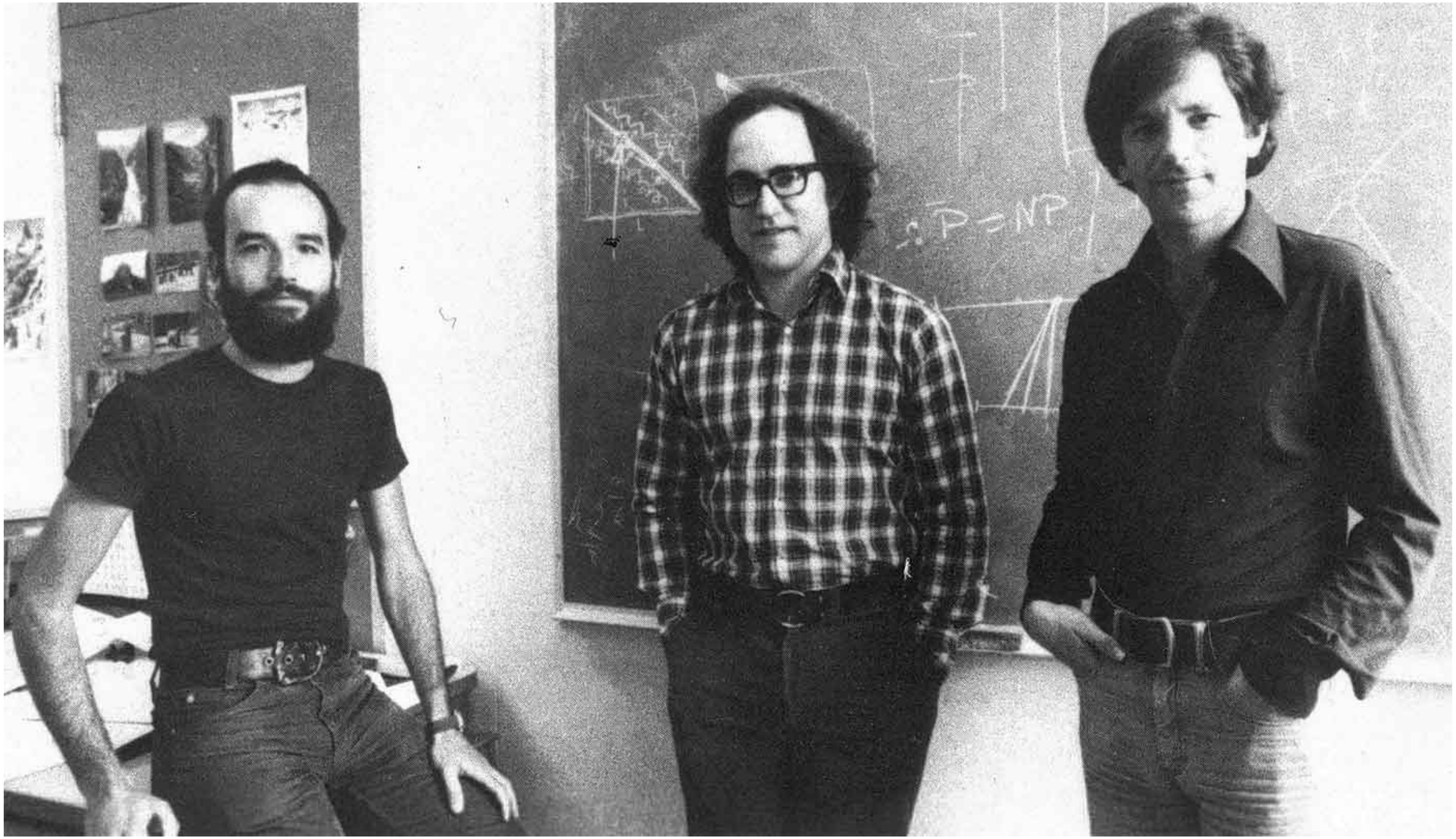
  8^7 mod 22 = 2,097,152 mod 22 = 2

Attention: this is a **toy/educational** example
just to get an intuition of how the algorithm works;
the numbers used in the example are not appropriate.

What did just happen?

- We encrypted 2 to 8
- We decrypted 8 back to 2
- No confusion/diffusion
- Encryption and Decryption is not based on a **single key**

RSA



Properties

- Two keys
  - Public Key (no secrecy)
  - Private Key (if stolen everything is lost)
- Easy algorithm, but **hard** to reverse
  - Computationally hard to infer p and q from n = pq and thus hard to compute ϕ(n) with no access to p and q
  - Computationally hard means solvable in non-polynomial time

RSA Setup / Key generation

1. Select p and q, which are two large prime numbers

   Picking p and q can be complicated, since the numbers should be prime and large

2. Compute $n = pq$
3. Pick e so that: $\gcd(e, \phi(n)) = 1$, and $1 < e < \phi(n)$ → public

   This means that e and $\varphi(n)$ are co-primes

4. Compute d from e, and $\phi(n)$ using the Extended Euclidean Algorithm: → private

   $ed \equiv 1 \bmod \phi(n)$, and $1 < d < \phi(n)$

5. We can compute d because

   (a) we can compute $\phi(n)$ since we know p and q

   (b) $\gcd(e, \phi(n)) = 1$, so that d exists

RSA Encryption/Decryption

- Encryption

  $C \equiv M^e \bmod n$

- Decryption

  $d(M)_{Kpr} \equiv C^d \bmod n$

- Keys
  - Public Key = {e, n}
  - Private Key = {d, n}

  $ed \equiv 1 \bmod \phi(n)$

Proof of Correctness

$C \equiv M^{e} \bmod n$

$d(M)_{Kpr} \equiv C^{d} \bmod n \equiv (M^{e} \bmod n)^{d} \bmod n \equiv (M^{e})^{d} \bmod n$

Therefore, $d(M)_{Kpr} \equiv M^{ed} \pmod{n}$

***Recall**: $ed \equiv 1 \pmod{\varphi(n)}$*

$d(M)_{Kpr} \equiv M^{1+k\phi(n)} \bmod n \equiv (M^{\phi(n)})^{k}\ M \bmod n$

Therefore, $d(M)_{Kpr} \equiv (M^{\phi(n)})^{k}\ M \pmod{n}$

Euler's Theorem: $\alpha^{\varphi(m)} \equiv 1 \pmod{m}$, when $\gcd(a, m) = 1$

Now, we have two cases

- $\gcd(M, n) = 1$
- $\gcd(M, n) \neq 1$

$\gcd(M, n) = 1$

$d(M)_{Kpr} \equiv (1 \bmod n)^k M \bmod n \equiv (1)^k M \bmod n$

Therefore, $d(M)_{Kpr} \equiv M \pmod{n}$

The decrypted message is equivalent to the original message M

$\gcd(M, n) \neq 1$

$\gcd(M, pq) \neq 1$, which means that M has p or q as a factor (i.e., $M = \lambda p$), since n has **only** p and q as factors

Then, we can write $\gcd(M, q) = 1$, since q is a prime, and this means (*Euler Theorem*) that: $1 \equiv (1)^t \equiv (M^{\phi(q)})^t \bmod q$

But, $(M^{\phi(n)})^t = (M^{(p-1)(q-1)})^t = ((M^{\phi(q)})^t)^{(p-1)} = 1^{(p-1)} \equiv 1 \pmod q$

or $(M^{\phi(n)})^t = 1 + uq$

We multiply both sides by M (recall, $M = \lambda p$)

$M(M^{\phi(n)})^t = M + Muq = M + \lambda puq = M + \lambda un$

Therefore, $M(M^{\phi(n)})^t \equiv M \pmod n$

But, $d(M)_{Kpr} \equiv (M^{\phi(n)})^k\ M \pmod n$

The decrypted message is equivalent to the original message M

Another RSA example

1. Select **p** = 17 and **q** = 11
2. Then, **n** = **pq** = 17 · 11 = 187
3. ϕ(n) = (**p**-1)(**q**-1) = 16 · 10 = 160
4. Select **e** relatively prime to ϕ(n) = 160 and less than ϕ(n); **e** = 7
5. Determine **d** using EEA

   de ≡ 1 (mod 160)

   d = 23, because 23 · 7 = 161 = (1 · 160) + 1

p
(big random prime)

q
(big random prime)

n = p · q
computing p and q from n requires super-polynomial time in the number of digits

Compute **ϕ(n), ϕ(n) = (p-1)(q-1)**
only if n can be expressed as **n = p · q**,
where p and q are primes

Select **e** which is relative prime to ϕ(n):
gcd(e, ϕ(n))=1

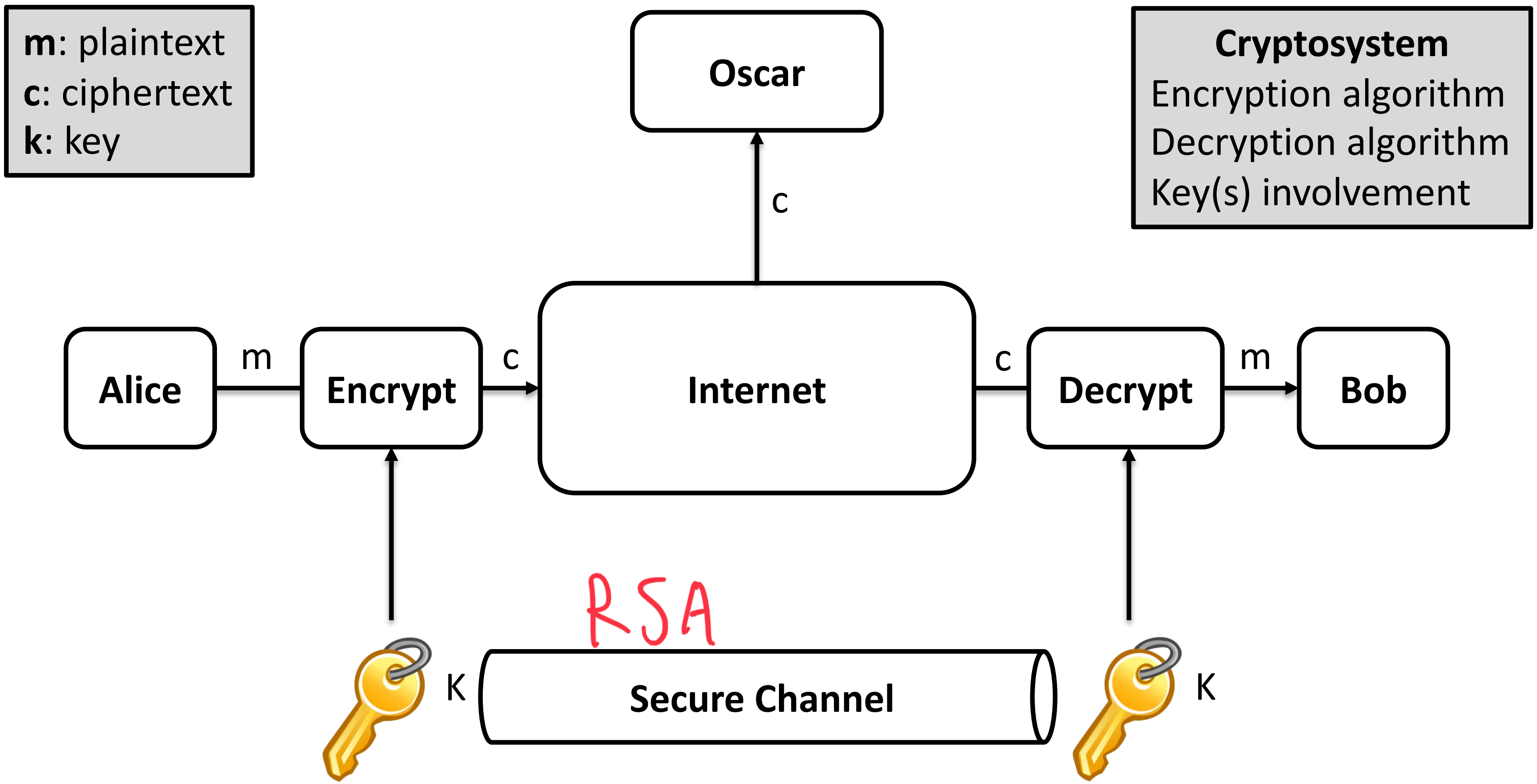
Select **d** using EEA
d · e ≡ 1 mod (p-1)(q-1)

Public Key
{e, n}

Private Key
{d, n}

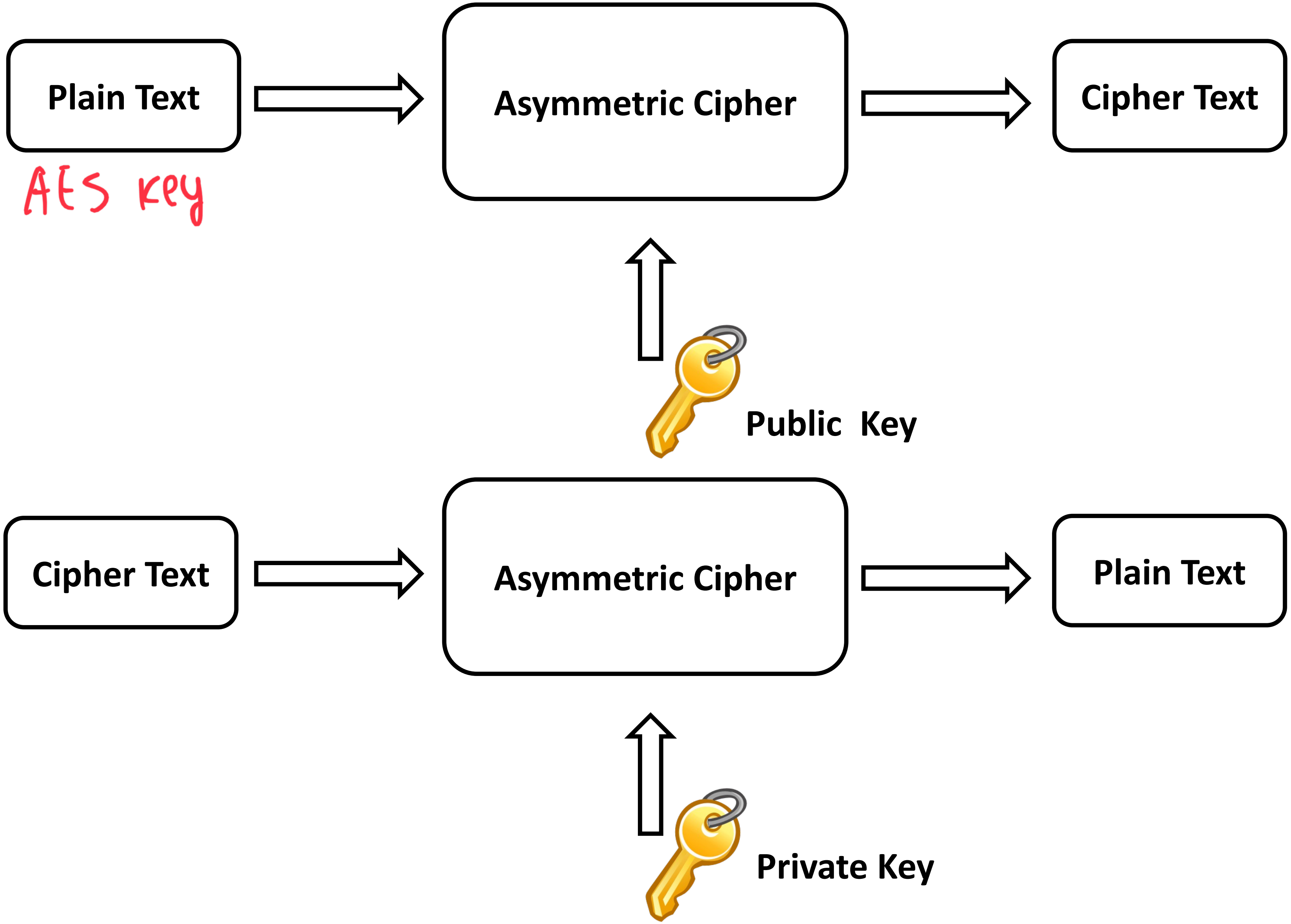
PUBLIC-KEY ENCRYPTION

Recall

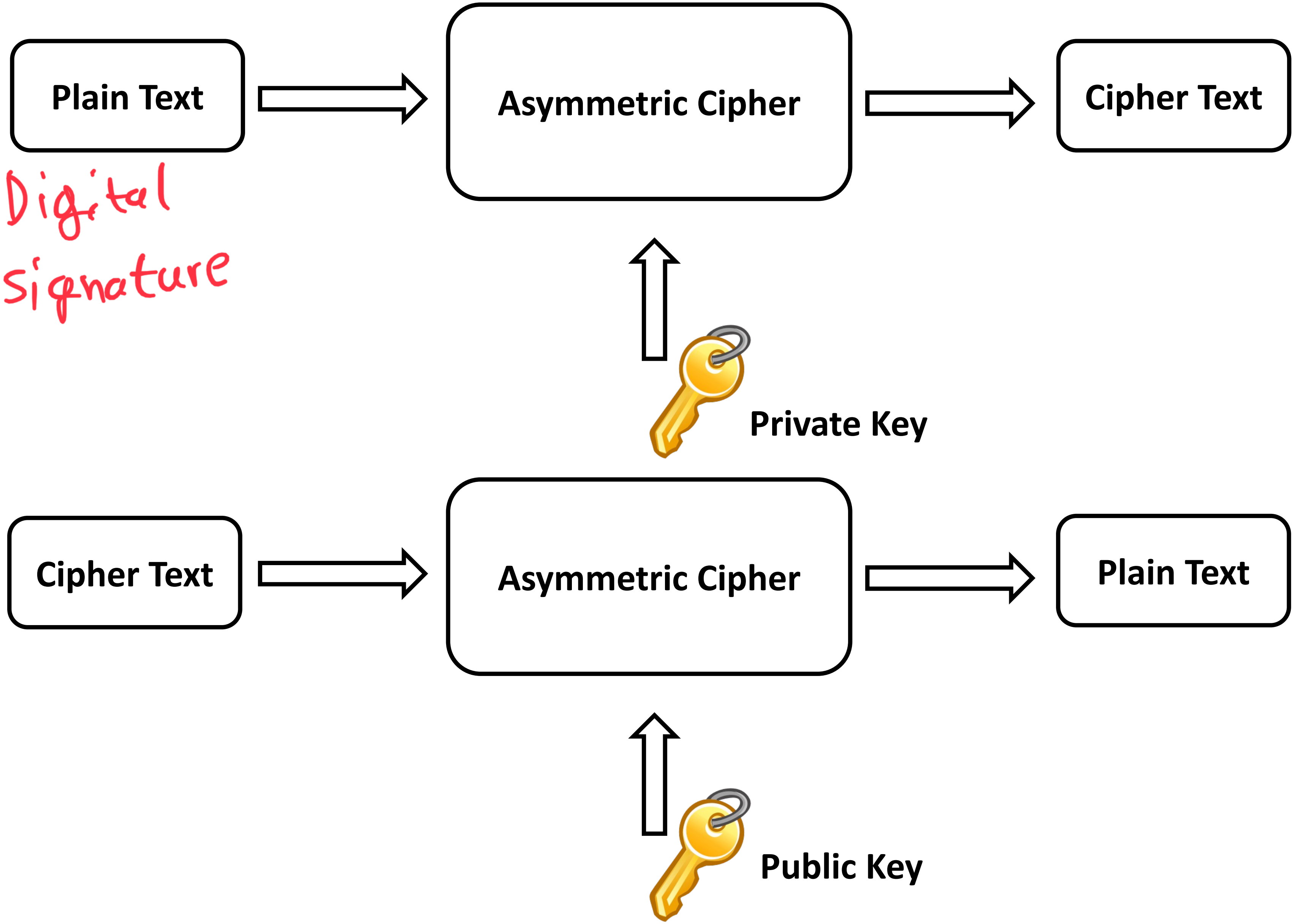


m: plaintext
c: ciphertext
k: key
Oscar
c
Cryptosystem
Encryption algorithm
Decryption algorithm
Key(s) involvement
Alice
m
Encrypt
c
Internet
c
Decrypt
m
Bob
RSA
K
Secure Channel
K

Asymmetric Encryption Mode 1

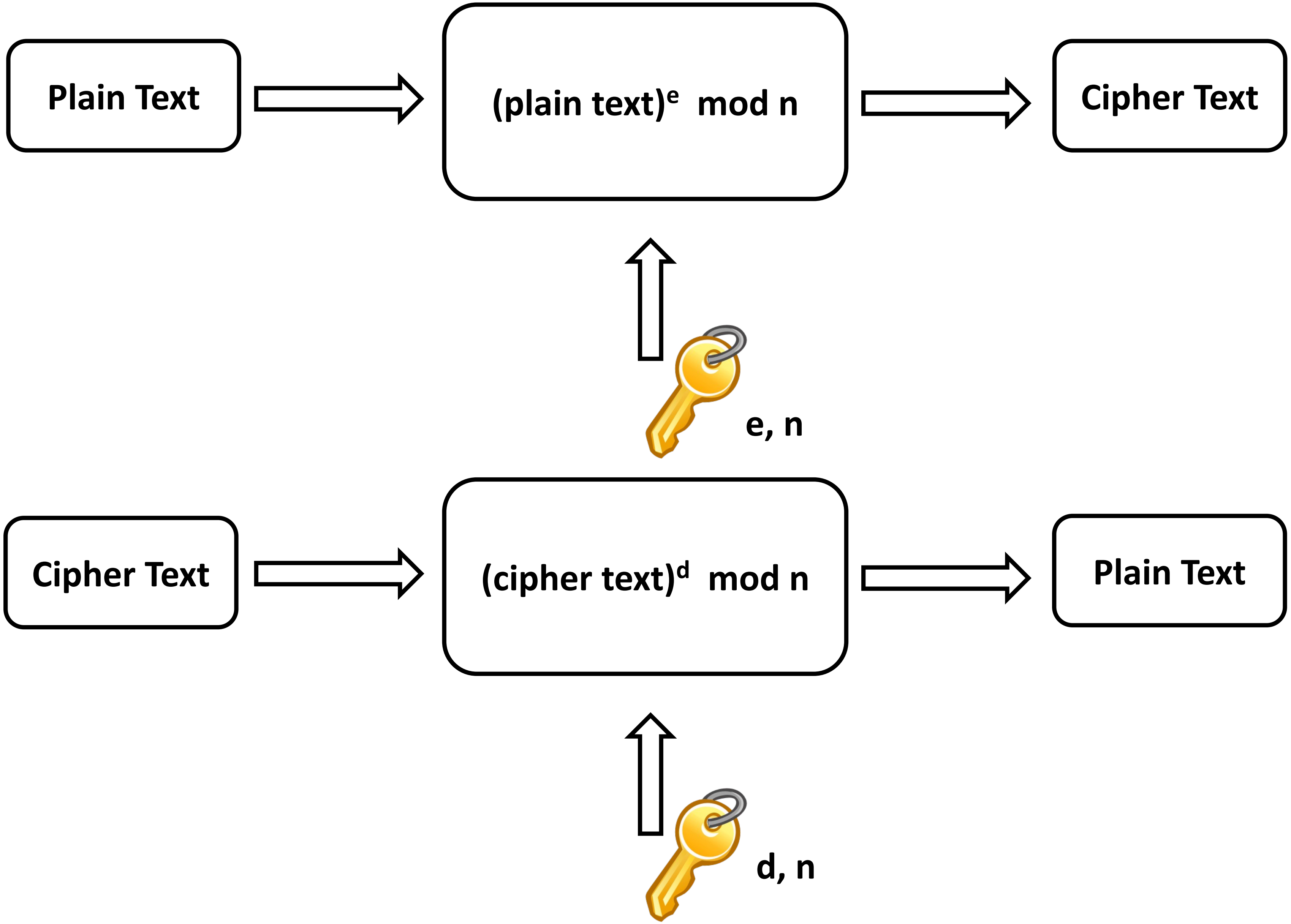


Asymmetric Encryption Mode 2



Digital signature

RSA



Plain Text
(plain text)^e mod n
Cipher Text
e, n
Cipher Text
(cipher text)^d mod n
Plain Text
d, n

Remarks

- All asymmetric cryptosystems are based on expensive mathematical operations
  - RSA uses exponentiation with very large prime numbers
- Not ideal for encrypting long messages
  - RSA will encrypt something which is less than *n* (typically 2-4Kbits)
- Ideal for (at least) two major applications
  - Secret Key distribution
  - Digital Signatures

IMPLEMENTATION/SECURITY

Implementation

- The RSA cryptosystem uses only one arithmetic operation (modular exponentiation) which makes it conceptually a simple asymmetric scheme
- Even though conceptually simple, due to the use of very long numbers, RSA is orders of magnitude slower than symmetric schemes, e.g., DES, AES
- When implementing RSA (esp. on a constrained device such as smartcards or cell phones) close attention has to be paid to the correct choice of arithmetic algorithms
- Choosing a small public exponent *e* does not weaken the security of RSA
  - A small public exponent improves the speed of the RSA encryption significantly

Analytical Attacks

- Mathematical attacks
  - The best known attack is factoring of *n* in order to obtain $\Phi(n)$
  - Can be prevented using a sufficiently large modulus *n*
  - The current factoring record is 664 bits, thus, it is recommended that *n* should have a bit length between 1024 and 3072 bits
- Malleability
  - A ciphertext can be transformed into another ciphertext which decrypts to a valid plaintext – without knowing the private key
  - $C \equiv M^e \pmod{n}$, then the attacker constructs $M^e t^e \bmod n = (Mt)^e \bmod n$
  - Can be prevented by proper padding
- Quantum Computers
  - RSA and all asymmetric cryptosystems can be defeated by algorithms running in Quantum Computers
  - Post-quantum Cryptography studies this field

Implementation Attacks

- Side-channel analysis
  - Exploit physical leakage of RSA implementation (e.g., power consumption, EM emanation, etc.)

Other Asymmetric Ciphers

- RSA
  - Factoring
- ElGamal
  - Computing discrete logarithms
- Elliptic curves
  - Computing the addition of two points in an elliptic curve

Resources

- This lecture was built using material that can be found at
  - Chapter 6 and 7, Understanding Cryptography, http://www.crypto-textbook.com
  - Chapter 8, Handbook of Applied Cryptography, http://cacr.uwaterloo.ca/hac/
  - Chapter 10, Serious Cryptography, https://nostarch.com/seriouscrypto
- Gift
  - https://www.youtube.com/watch?v=lEvXcTYqtKU