KF School of Computing and Information Sciences
Florida International University

CNT 4403
Computing and Network Security

Cryptography – Asymmetric Crypto

Dr. Kemal Akkaya

E-mail: *kakkaya@fiu.edu*

Public-key Systems

Two keys

- *Private key* known only to individual
- *Public key* available to anyone

Idea

- Confidentiality: encrypt using public key, decrypt using private key
- Integrity/authentication: encipher using private key, decipher using public key

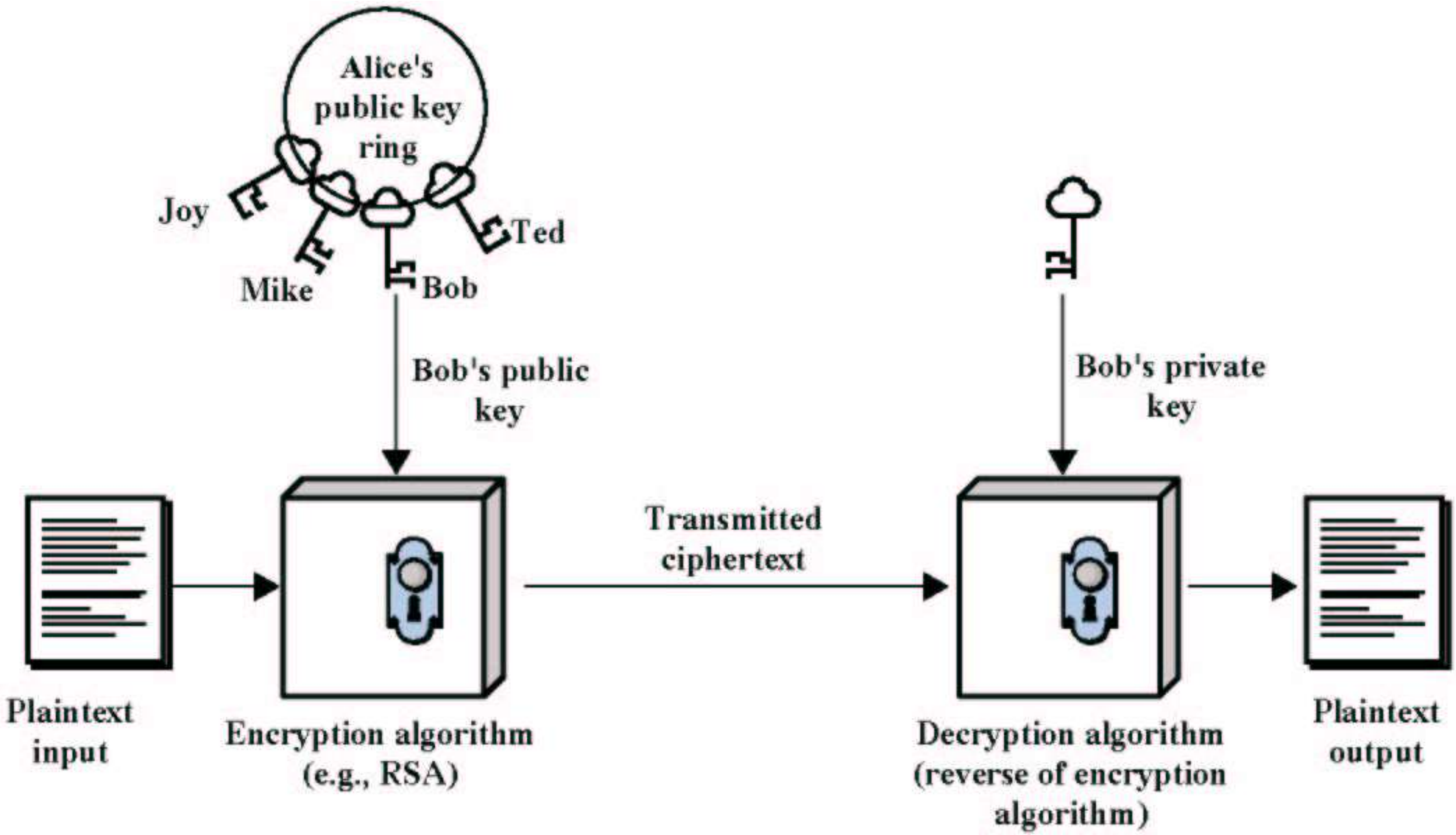
Requirements

- It must be computationally easy to encipher or decipher a message given the appropriate key
- It must be computationally infeasible to derive the private key from the public key
- It must be computationally infeasible to determine the private key from a chosen plaintext attack
  - ✓ Based on Hard problems: Factorization, discrete logarithms, elliptic curves, etc.

Public-Key (Asymmetric) Encryption

- **A *public-key encryption scheme* works as follows:**
  - Each user has a *public key* e and a *private key* d.
  - To send a message M to Bob, Alice obtains Bob's public key e_{Bob} and encrypts M with e_{Bob}:
    $$C = E_{e_{Bob}}(M)$$
  - When Bob receives the ciphertext C, he decrypts it with his private key d_{Bob}:
    $$M = D_{d_{Bob}}(C)$$
  - The decryption algorithm D_d must be the reverse of the encryption algorithm E_e; i.e., $D_d(E_e(M)) = M$ for any message M.
  - It is computationally infeasible to determine the private key d given the knowledge of the public key e and the encryption and decryption algorithms.

Public-Key Encryption



Alice

Bob

RSA

- The ***RSA cryptosystem*** **was invented by Ron Rivest, Adi Shamir and Len Adleman at MIT in 1977. It is, to date, the first unbroken and most widely used public-key cryptosystem.**
- **Rivest, Shamir and Adleman received the Turing Award 2002 for inventing RSA.**


LABORATORY FOR COMPUTER SCIENCE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

A METHOD FOR OBTAINING DIGITAL SIGNATURES AND PUBLIC-KEY CRYPTOSYSTEMS

Ronald Rivest
Adi Shamir
Len Adleman

April 1977


Shamir, Rivest and Adleman in 1978


Shamir, Rivest and Adleman at Turing Award Lecture


Rivest, Shamir and Adleman at RSA 2003


Adleman, Shamir, Rivest at CRYPTO '82

RSA

- **Key generation**
  1. Choose two (large) distinct primes p and q at random.
  2. Compute the *RSA modulus* - $\boldsymbol{n} = pq$.
  3. Compute *Totient* - $\boldsymbol{ø(n)} = (p – 1)(q – 1)$.
  4. Select a random integer $\boldsymbol{e}$ (called the *encryption exponent*), $1 < e < ø(n)$, such that $\gcd(e, ø(n)) = 1$.
  5. Compute the unique integer $\boldsymbol{d}$ (called the *decryption exponent*), $1 < d < ø(n)$, such that $ed \bmod ø(n) = 1$.
  6. The public key is (e, n); the private key is (d, n).
- **Encryption**
  - Given: receiver's public key (e, n), message M ε [0, n-1]
  - Encrypt: $C = M^e \bmod n$
- **Decryption**
  - Given: private key (d, n), ciphertext C
  - Decrypt: $M = C^d \bmod n$

RSA: Example

Key generation

1. Choose $p = 11$ and $q = 47$.
2. Compute the RSA modulus $n = 11 \times 47 = 517$.
3. Compute $\phi(n) = 10 \times 46 = 460$.
4. Choose the encryption exponent $e = 3$. It is clear that $\gcd(e, \phi(n)) = \gcd(3, 460) = 1$.
5. Compute the decryption exponent $d = 307$.
6. The public key is (3, 517); the private key is (307, 517).

Encryption

- Given: public key (3, 517), message $M = 26$
- Encrypt: $C = 26^3 \bmod 517 = 515$

Decryption

- Given: private key (307, 517), ciphertext $C = 515$
- Decrypt: $M = 515^{307} \bmod 517 = 26$

RSA Security

- **The security of RSA depends on the hardness of the *integer factorization problem*: *given an integer x > 1, what is its prime factorization?***
  - Expressing the integer in terms of multiplication of primes
  - Currently there is no known algorithm that can efficiently factor a number whose prime factors are all arbitrarily large.
- **If one can factor the RSA modulus n into p and q, then he can compute $ø(n) = (p – 1)(q – 1)$ and thus he can determine the decryption exponent d from e and $ø(n)$ efficiently using the *Extended Euclidean Algorithm***
  - *Find d, given* $ø(n)$ and e using $ed \bmod ø(n) = 1$
- **The RSA modulus n should be at least 1024 bits long to guard against today's factoring attacks.**

Attacks on RSA

- **Brute force**
  - trying all possible private keys
  - use larger key, but then slower
- **Mathematical attacks (factoring n)**
  - see improving algorithms (QS, GNFS, SNFS)
  - currently 1024-2048-bit keys seem secure
- **Side channel timing attacks (on implementation)**
  - Check the time it takes during decryption of bits 1 and 0 on certain functions.
  - use - constant time, random delays, blinding
- **Chosen ciphertext attacks (on RSA properties)**

Other Public-Key Algorithms

- **Digital Signature Standard (DSS)**
  - FIPS PUB 186 from 1991, revised 1993 & 96
  - cannot be used for encryption
  - Based on El-Gamal
- **Elliptic curve cryptography (ECC)**
  - Equal security for smaller bit size than RSA
  - Seen in standards such as IEEE P1363
  - Based on a mathematical construct known as the elliptic curve (difficult to explain)
    - ✓ Smaller keys is a plus compared to RSA
    - ✓ Decryption is costly and very slow
  - Gives a possible compromise between symmetric key systems and public key systems

Digital Signatures

- Enciphered messages that can be mathematically proven to be authentic

- Created in response to rising need to verify information transferred using electronic systems

- Asymmetric encryption processes are used to create digital signatures
  - Reverse of the asymmetric encryption

- Digital Certificates (will check them later in more details)
  - Electronic document containing the public key and identifying information about the entity that owns and controls key
  - Digital signature attached to certificate's container file to certify that the file is from entity it claims to be from

Digital Signatures

- **A *digital signature scheme* is composed of:**
  - A public key e and a private key d for each user
  - A *signing algorithm* sig_d that, with a message M and the signer's private key d as input, produces the *digital signature* $s = sig_d(M)$ of M
  - A *verification algorithm* ver_e that, with a message M, a digital signature s and the signer's public key e as input, returns true if and only if s is the digital signature of M signed by e's owner; i.e.,

$$ver_e(M,s) = \begin{cases} true & \text{if } s = sig_d(M) \\ false & \text{if } s \neq sig_d(M) \end{cases}$$

- **RSA digital signature scheme**
  - (d, n) = signer's private key, (e, n) = signer's public key
  - Sign: $s = [Hash(M)]^d \bmod n$
  - Verify: return true if and only if $Hash(M) = s^e \bmod n$
- ***Hash-and-sign*: if the message is long, it is common to sign the message digest (H(M)) instead of the message itself to save time**
  - Hash functions will be explained later

Example: Integrity/Authentication

- **Take p = 7, q = 11, so n = 77 and ø (n) = 60**
- **Alice chooses e = 17, making d = 53**
  - *e* is her public key, *d* is her private key
- **Alice wants to send Bob message HELLO (07 04 11 11 14) so Bob knows it is what Alice sent (no changes in transit, and authenticated)**
  - 07^{53} mod 77 = 35 (the signing uses **her private** key---authentication)
  - 04^{53} mod 77 = 09
  - 11^{53} mod 77 = 44
  - 11^{53} mod 77 = 44
  - 14^{53} mod 77 = 49
- **Alice sends 35 09 44 44 49**
- **Notice that anyone can intercept this…can only Bob decode?**

Integrity/Authentication cont'd

- **Bob receives 35 09 44 44 49**
- **Bob uses Alice's public key, *e* = 17, *n* = 77, to decrypt message:**
  - $35^{17} \bmod 77 = 07$
  - $09^{17} \bmod 77 = 04$
  - $44^{17} \bmod 77 = 11$
  - $44^{17} \bmod 77 = 11$
  - $49^{17} \bmod 77 = 14$
- **Bob translates message to letters to read HELLO**
  - Alice sent it as only she knows her private key, so no one else could have enciphered it
  - If (enciphered) message's blocks (letters) altered in transit, would not decrypt properly
  - Coding is not tied to Bob----anyone could intercept and decode.
    - ✓ But whoever does know that the message had to have been encoded with Alice's private key

Security Services Provided by RSA

- **Confidentiality**
  - Only the owner of the private key knows it, so text enciphered with public key cannot be read by anyone except the owner of the private key
- **Authentication**
  - Only the owner of the private key knows it, so text enciphered with private key must have been generated by the owner
- **Integrity**
  - Enciphered letters cannot be changed undetectably without knowing private key
- **Non-Repudiation**
  - Message enciphered with private key came from someone who knew it

Example: BOTH Encryption & Signing

- **Alice wants to send Bob message HELLO both enciphered and authenticated (integrity-checked)**
  - Alice's keys: public (17, 77); private: 53
  - Bob's keys: public: (37, 77); private: 13
- **Alice enciphers HELLO (07 04 11 11 14): authenticate first, code to recipient last**
  - $(07^{53} \bmod 77)^{37} \bmod 77 = 07$
  - $(04^{53} \bmod 77)^{37} \bmod 77 = 37$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(11^{53} \bmod 77)^{37} \bmod 77 = 44$
  - $(14^{53} \bmod 77)^{37} \bmod 77 = 14$
- **Alice sends 07 37 44 44 14**

Symmetric-Key vs. Public-Key

- **In symmetric-key encryption, the sender has to establish a secret key with the receiver prior to encryption.**
  - However, in public-key encryption, the sender just needs to obtain an *authentic* copy of the receiver's public key.
    - In a network of n users, a symmetric-key cryptosystem requires $n(n-1)/2$ secret keys, but a public-key cryptosystem requires only n public-private key pairs.
- **Public-key cryptography (digital signatures) provides <u>non-repudiation</u> while symmetric-key cryptography does not.**
- **Public-key cryptosystems are substantially slower than symmetric-key cryptosystems since the key sizes of public-key cryptosystems are typically much larger.**
  - RSA 10,000 times slower than AES
  - ECC does better but still slower than AES