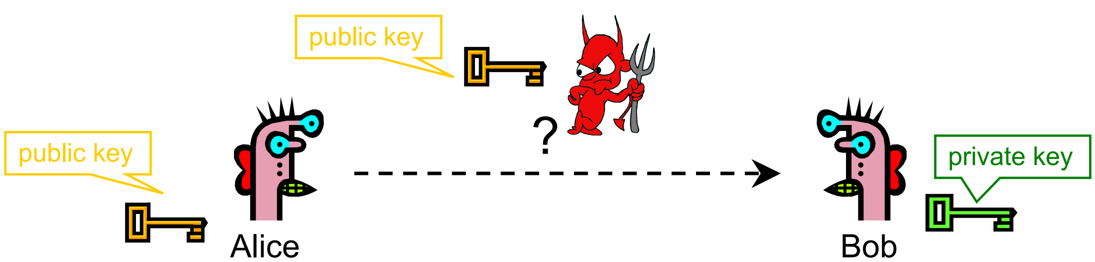
Public Key Cryptography

EJ Jung

Basic Public Key Cryptography



Given: Everybody knows Bob's public key
- How is this achieved in practice?

Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a secret message to Bob

2. Bob wants to authenticate himself

Requirements for Public-Key Crypto

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
  - Computationally infeasible to determine private key PK given only public key PK
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C=E_{PK}(M)$
- Decryption: given ciphertext $C=E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to compute M from C without SK
  - Decrypt(SK,Encrypt(PK,M))=M

Requirements for Public-Key Cryptography

1. Computationally easy for a party B to generate a pair (public key KU_b, private key KR_b)
2. Easy for sender to generate ciphertext:

$$C = E_{KUb}(M)$$

3. Easy for the receiver to decrypt ciphertect using private key:

$$M = D_{KRb}(C) = D_{KRb}[E_{KUb}(M)]$$

Requirements for Public-Key Cryptography

4. Computationally infeasible to determine private key (KR_b) knowing public key (KU_b)
5. Computationally infeasible to recover message M, knowing KU_b and ciphertext C
6. Either of the two keys can be used for encryption, with the other used for decryption:

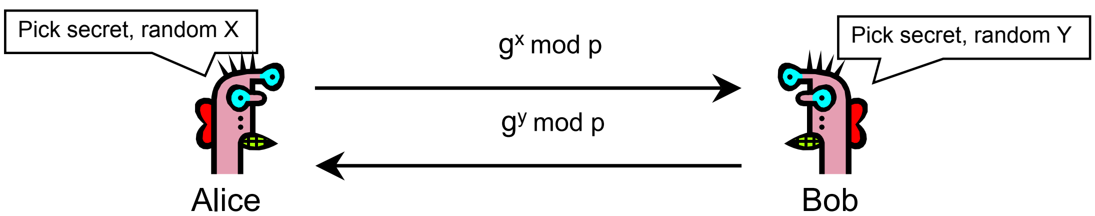
$$M = D_{KRb}[E_{KUb}(M)] = D_{KUb}[E_{KRb}(M)]$$

Public-Key Cryptographic Algorithms

- RSA and Diffie-Hellman
- **RSA** - Ron Rives, Adi Shamir and Len Adleman at MIT, in 1977.
  - RSA is a block cipher
  - The most widely implemented
- **Diffie-Hellman**
  - Echange a secret key securely
  - Compute discrete logarithms

Diffie-Hellman Protocol (1976)

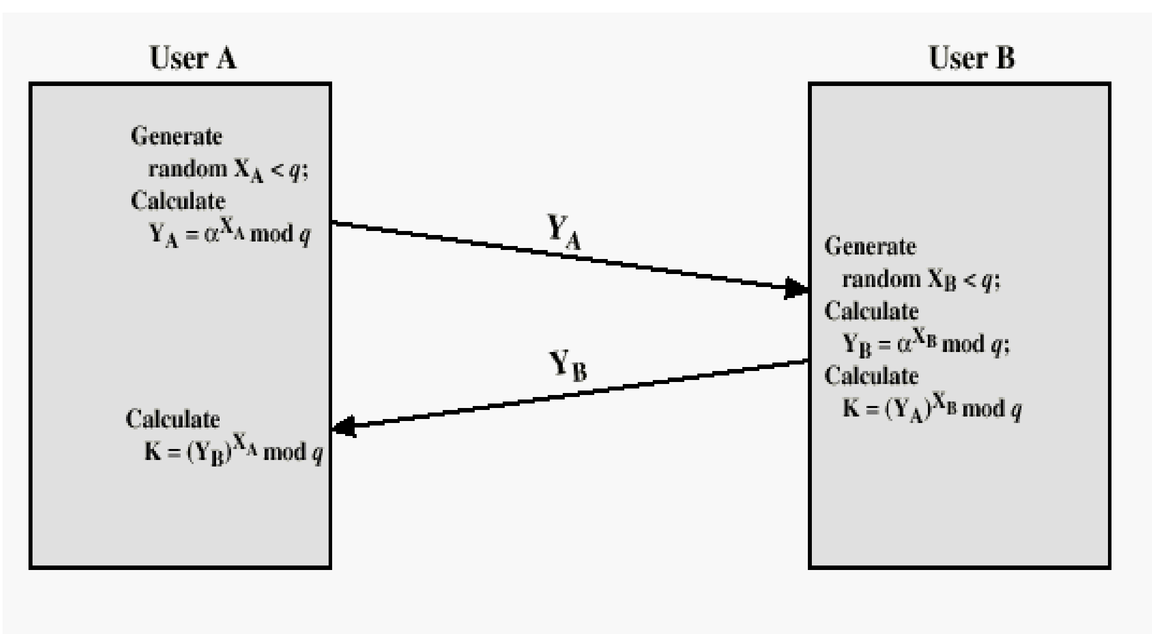
- Alice and Bob never met and share no secrets
- Public info: p and g
  - p is a large prime number, g is a generator of Z_p*
    - $Z_p^* = \{1, 2 \ldots p-1\}$; $\forall a \in Z_p^* \ \exists i$ such that $a = g^i \bmod p$
    - Modular arithmetic: numbers "wrap around" after they reach p



Compute $k=(g^y)^x=g^{xy} \bmod p$

Compute $k=(g^x)^y=g^{xy} \bmod p$

Diffie-Hellman Key Echange



Why Is Diffie-Hellman Secure?

- Discrete Logarithm (DL) problem: given g^x mod p, it's hard to extract x
  - There is no known efficient algorithm for doing this
  - This is not enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem: given g^x and g^y, it's hard to compute g^{xy} mod p
  - … unless you know x or y, in which case it's easy
- Decisional Diffie-Hellman (DDH) problem: given g^x and g^y, it's hard to tell the difference between g^{xy} mod p and g^r mod p where r is random

Properties of Diffie-Hellman

- Assuming DDH problem is hard, Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  - Eavesdropper can't tell the difference between established key and a random value
  - Can use new key for symmetric cryptography
    - Approx. 1000 times faster than modular exponentiation

Limitations of Diffie-Hellman

- Diffie-Hellman protocol alone does not provide authentication
- Why?
  - authentication means associating a certain identity
  - needs to know whose public key this is

Rivest, Shamir and Adleman (1977)

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

RSA en/decryption

Calculate d	$de \bmod \phi(n) = 1$
Public key	$KU = \{e, n\}$
Private key	$KR = \{d, n\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod{n}$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \pmod{n}$

Example of RSA Algorithm

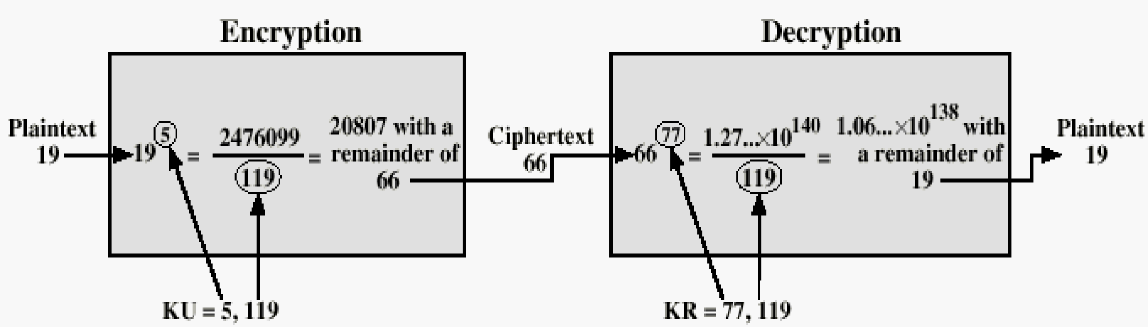


Figure 3.9 Example of RSA Algorithm

Why Is RSA Secure?

- RSA problem: given $n=pq$, e such that $\gcd(e,(p-1)(q-1))=1$ and c, find m such that $m^e=c \bmod n$
  - i.e., recover m from ciphertext c and public key (n,e) by taking e^{th} root of c
  - There is no known efficient algorithm for doing this
- Factoring problem: given positive integer n, find primes $p_1, \ldots, p_k$ such that $n=p_1^{e_1}p_2^{e_2}\ldots p_k^{e_k}$
- If factoring is easy, then RSA problem is easy, but there is no known reduction from factoring to RSA
  - It may be possible to break RSA without factoring n

Other Public-Key Cryptographic Algorithms

- Digital Signature Standard (DSS)
  - Makes use of the SHA-1
  - Not for encryption or key echange
- Elliptic-Curve Cryptography (ECC)
  - Good for smaller bit size
  - Low confidence level, compared with RSA
  - Very complex

Applications of Public-Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric crypto, must know secret key to encrypt
  - Only someone who knows private key can decrypt
  - Key management is simpler (maybe)
    - Secret is stored only at one site: good for open environments
- Digital signatures for authentication
  - Can "sign" a message with your private key
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

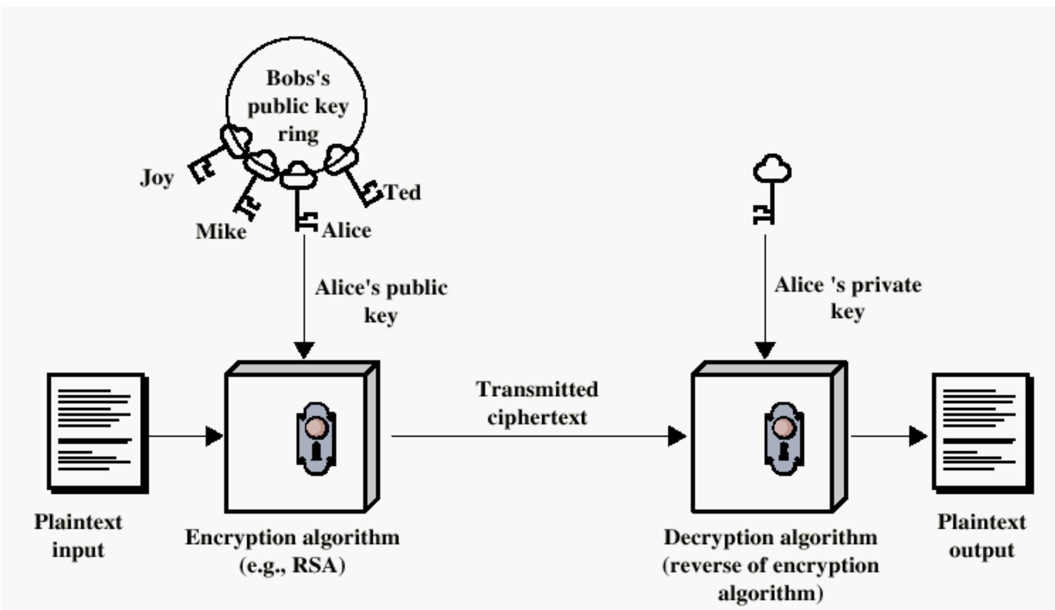
Advantages of Public-Key Crypto

- Confidentiality without shared secrets
  - Very useful in open environments
  - No "chicken-and-egg" key establishment problem
    - With symmetric crypto, two parties must share a secret before they can exchange secret messages
- Authentication without shared secrets
  - Use digital signatures to prove the origin of messages
- Reduce protection of information to protection of authenticity of public keys
  - No need to keep public keys secret, but must be sure that Alice's public key is really her true public key

Disadvantages of Public-Key Crypto

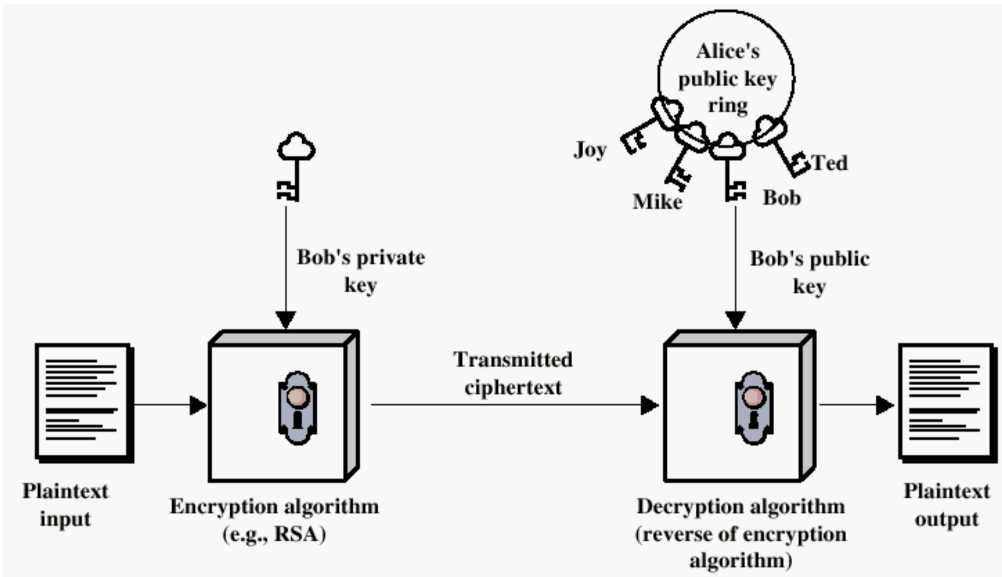
- Calculations are 2-3 orders of magnitude slower
  - Modular exponentiation is an expensive computation
  - Typical usage: use public-key cryptography to establish a shared secret, then switch to symmetric crypto
    - We'll see this in IPSec and SSL
- Keys are longer
  - 1024 bits (RSA) rather than 128 bits (AES)
- Relies on unproven number-theoretic assumptions
  - What if factoring is easy?
    - Factoring is believed to be neither P, nor NP-complete

Encryption using Public-Key system



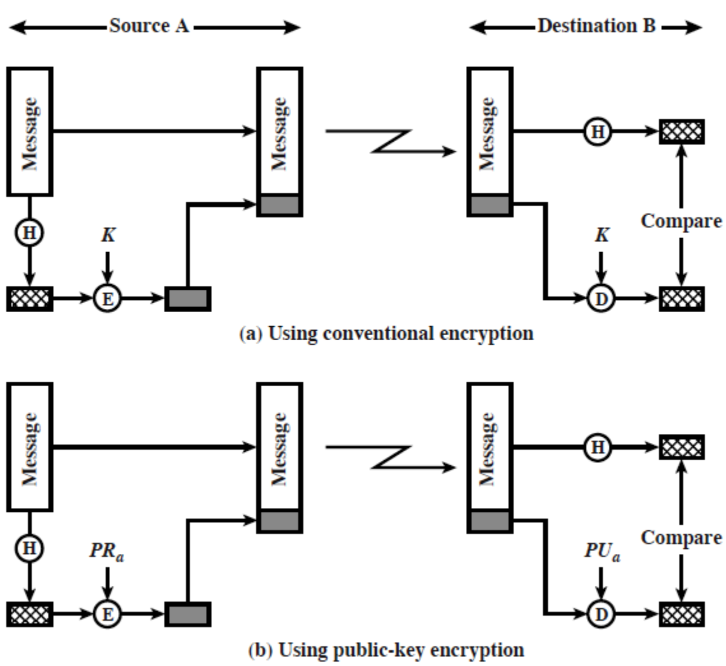
usfCS UNIVERSITY of SAN FRANCISCO department of computer science

Authentication using Public-Key System

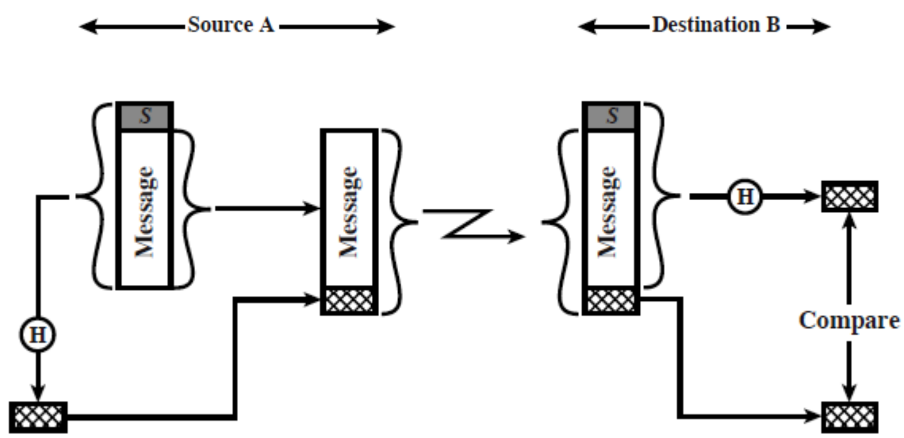


usfCS UNIVERSITY of SAN FRANCISCO department of computer science

MAC in encryptions



MAC with secret value



(c) Using secret value

Figure 3.2 Message Authentication Using a One-Way Hash Function

Key Management
Public-Key Certificate Use