Public Key Cryptography

CSS322: Security and Cryptography

Sirindhorn International Institute of Technology
Thammasat University

Prepared by Steven Gordon on 31 October 2012
CSS322Y12S2L07, Steve/Courses/2012/s2/css322/lectures/rsa.tex, r2531

Contents

Birth of Public-Key Cryptosystems

- Beginning to 1960's: permutations and substitutions (Caesar, rotor machines, DES, ...)
- 1960's: NSA secretly discovered public-key cryptography
- 1970: first known (secret) report on public-key cryptography by CESG, UK
- 1976: Diffie and Hellman public introduction to public-key cryptography
  - Avoid reliance on third-parties for key distribution
  - Allow digital signatures

Principles of Public-Key Cryptosystems

- Symmetric algorithms used same secret key for encryption and decryption
- Asymmetric algorithms in public-key cryptography use one key for encryption and different but related key for decryption
- Characteristics of asymmetric algorithms:
  - Require: Computationally infeasible to determine decryption key given only algorithm and encryption key
  - Optional: Either of two related keys can be used for encryption, with other used for decryption

Public and Private Keys

Public Key

- For secrecy: used in encryption
- For authentication: used in decryption
- Available to anyone

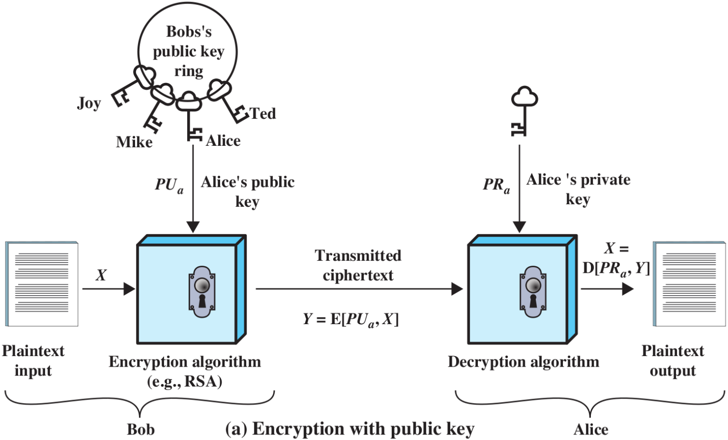
Private Key

- For secrecy: used in decryption
- For authentication: used in decryption
- Secrect, known only by owner

Public-Private Key Pair

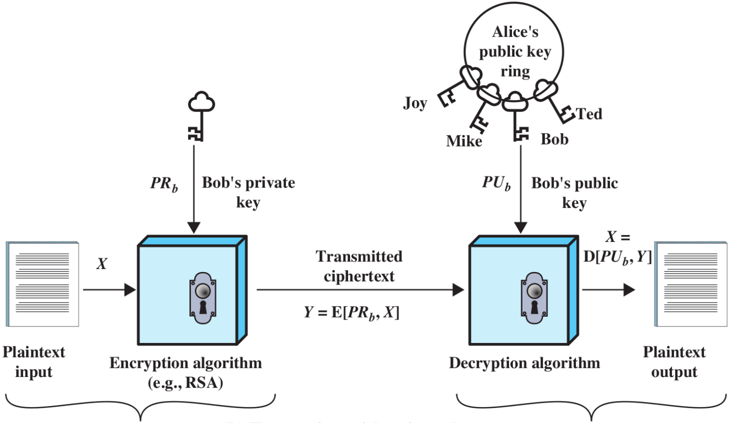
- User A has pair of related keys, public and private: (PU_a, PR_a)

Encryption with Public Key



(a) Encryption with public key

Encryption with Private Key

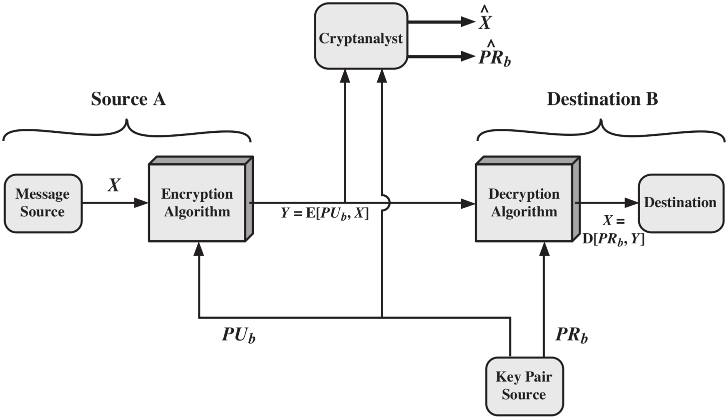


(b) Encryption with private key

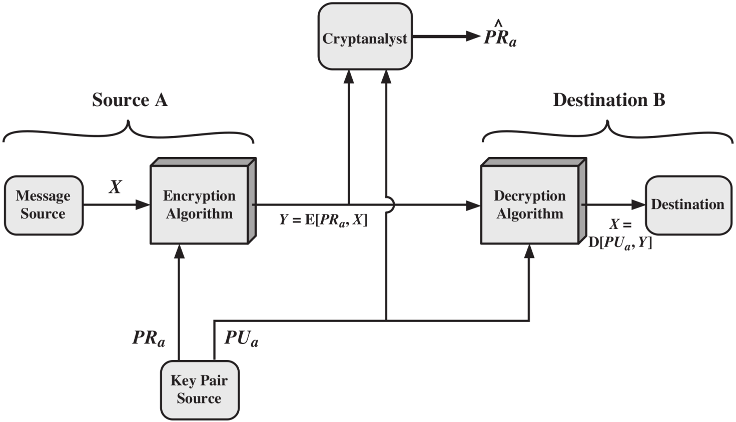
Conventional vs Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

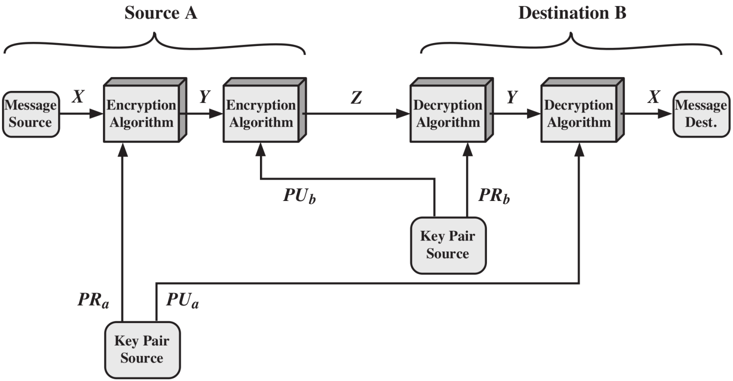
Secrecy in a Public Key Cryptosystem



Authentication in a Public Key Cryptosystem



Secrecy and Authentication in a Public Key Cryptosystem



Applications of Public Key Cryptosystems

- Secrecy, encryption/decryption of messages
- Digital signature, *sign* message with private key
- Key exchange, share secret session keys

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements of Public-Key Cryptography

1. Computationally easy for B to generate pair (PU_b, PR_b)
2. Computationally easy for A, knowing PU_b and message M, to generate ciphertext:

$$C = \mathrm{E}(PU_b, M)$$

3. Computationally easy for B to decrypt ciphertext using PR_b:

$$M = \mathrm{D}(PR_b, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

4. Computationally infeasible for attacker, knowing PU_b and C, to determine PR_b
5. Computationally infeasible for attacher, knowing PU_b and C, to determine M
6. (Optional) Two keys can be applied in either order:

$$M = \mathrm{D}[PU_b, \mathrm{E}(PR_b, M)] = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$$

Requirements of Public-Key Cryptography

6 requirements lead to need for trap-door one-way function

- Every function value has unique inverse
- Calculation of function is easy
- Calculation of inverse is infeasible, unless certain information is known

$Y = f_k(X)$	easy, if k and Y are known
$X = f_k^{-1}(Y)$	easy, if k and Y are known
$X = f_k^{-1}(Y)$	infeasible, if Y is known but k is not

- What is easy? What is infeasible?
  - Computational complexity of algorithm gives an indication
  - Easy if can be solved in polynomial time as function of input

Public-Key Cryptanalysis

Brute Force Attacks

- Use large key to avoid brute force attacks
- Public key algorithms less efficient with larger keys
- Public-key cryptography mainly used for key management and signatures

Compute Private Key from Public Key

- No known feasible methods using standard computing

Probable-Message Attack

- Encrypt all possible M' using PU_b—for the C' that matches C, attacker knows M
- Only feasible of M is short
- Solution for short messages: append random bits to make it longer

Contents

RSA

- Ron Rivest, Adi Shamir and Len Adleman
- Created in 1978; RSA Security sells related products
- Most widely used public-key algorithm
- Block cipher: plaintext and ciphertext are integers

The RSA Algorithm

- Plaintext encrypted in blocks, each block binary value less than n
- In practice, block size i bits where $2^i < n \leq 2^{i+1}$; n is 1024 bits
- Encryption of plaintext M:

$$C = M^e \bmod n$$

- Decryption of ciphertext C:

$$\begin{aligned} M &= C^d \bmod n \\ &= (M^e)^d \bmod n = M^{ed} \bmod n \end{aligned}$$

- Sender A and receiver B know n; Sender A knows e; Receiver B knows d
- $PU_b = \{e, n\}$, $PR_b = \{d, n\}$

Requirements of the RSA Algorithm

1. Possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$
2. Easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$
3. Infeasible to determine d given e and n

- Requirement 1 met if e and d are relatively prime
- Choose primes p and q, and calculate:

$$n = pq$$
$$1 < e < \phi(n)$$
$$ed \equiv 1 \pmod{\phi(n)} \text{ or } d \equiv e^{-1} \pmod{\phi(n)}$$

- n and e are public; p, q and d are private

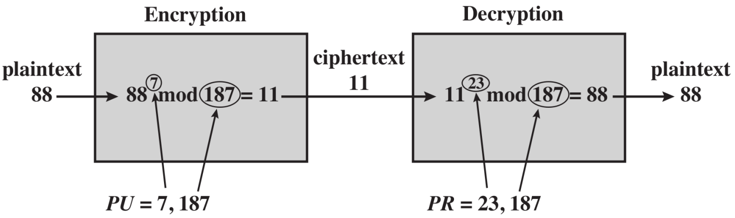
The RSA Algorithm

Key Generation by Alice	
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	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

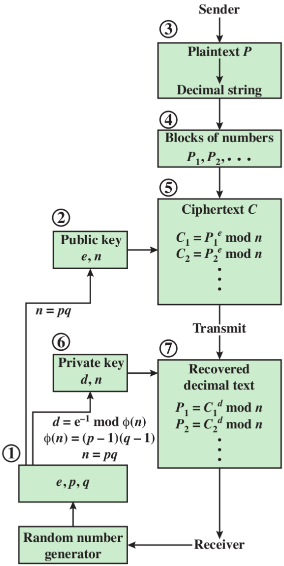
Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Private Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

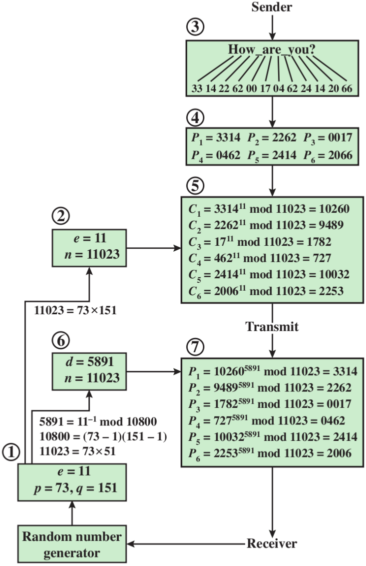
Example of RSA Algorithm



RSA Processing of Multiple Blocks



Example of RSA Processing of Multiple Blocks



Computational Efficiency of RSA

- Encryption and decryption require exponentiation
  - Very large numbers; using properties of modular arithmetic makes it easier:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

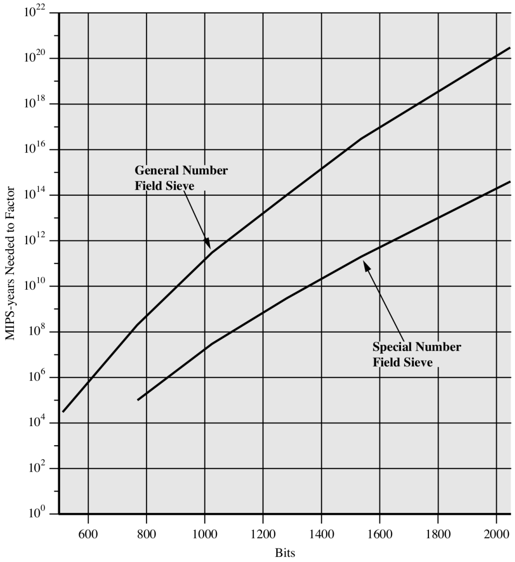
- Choosing e
  - Values such as 3, 17 and 65537 are popular: make exponentiation faster
  - Small e vulnerable to attack: add random padding to each M
- Choosing d
  - Small d vulnerable to attack
  - Decryption using large d made faster using Chinese Remainder Theorem and Fermat's Theorem
- Choosing p and q
  - p and q must be very large primes
  - Choose random odd number and test if its prime (probabilistic test)

Security of RSA

- Brute-Force attack: choose large d (but makes algorithm slower)
- Mathematical attacks:
  1. Factor n into its two prime factors
  2. Determine $\phi(n)$ directly, without determining p or q
  3. Determine d directly, without determining $\phi(n)$

  - Factoring n is considered fastest approach; hence used as measure of RSA security
- Timing attacks: practical, but countermeasures easy to add (e.g. random delay). 2 to 10% performance penalty
- Chosen ciphertext attack: countermeasure is to use padding (Optimal Asymmetric Encryption Padding)

MIPS-Years Needed To Factor



Progress in Factorization

Number of Decimal Digits	Approximate Number of Bits	Date Achieved	MIPS-Years	Algorithm
100	332	April 1991	7	Quadratic sieve
110	365	April 1992	75	Quadratic sieve
120	398	June 1993	830	Quadratic sieve
129	428	April 1994	5000	Quadratic sieve
130	431	April 1996	1000	Generalized number field sieve
140	465	February 1999	2000	Generalized number field sieve
155	512	August 1999	8000	Generalized number field sieve
160	530	April 2003	—	Lattice sieve
174	576	December 2003	—	Lattice sieve
200	663	May 2005	—	Lattice sieve

See `http://www.rsa.com/rsalabs/node.asp?id=2092` for update. RSA-768 has been solved.

Contents

Diffie-Hellman Key Exchange

- Diffie and Hellman proposed public key cryptosystem in 1976
- Algorithm for exchanging secret key (not for secrecy of data)
- Based on discrete logarithms
- Easy to calculate exponentials modulo a prime
- Infeasible to calculate inverse, i.e. discrete logarithm

Diffie-Hellman Key Exchange Algorithm

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

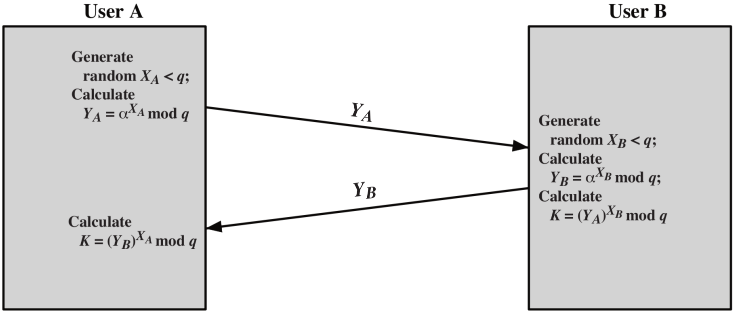
Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange



Security of Diffie-Hellman Key Exchange

- Insecure against man-in-the-middle-attack
- Countermeasure is to use digital signatures and public-key certificates

Contents

Other Public-Key Cryptosystems

ElGamal Cryptosystem

- Similar concepts to Diffie-Hellman
- Used in Digital Signature Standard and secure email

Elliptic Curve Cryptography

- Uses elliptic curve arithmetic (instead of modular arithmetic in RSA)
- Equivalent security to RSA with smaller keys (better performance)
- Used for key exchange and digital signatures