CS472 Module 11 Part A - Number Theory and Cryptography Algorithms

Athens State University

April 11, 2016

Outline

Contents

1 Some basics

Number theory

- **Number theory**: branch of mathematics devoted to the study of the behavior of the set of integers, sets that can be derived from the integers, and sets that behave like the integers
- A super-set of what is commonly considered "arithmetic"
- **Computational number theory**: The study of algorithms for performing number theoretic computations

Prime Numbers

- *Prime Number*: a number that has divisors of 1 and self
- *Prime Factorization*: express some number n as a product of prime numbers
  - Note that it is much harder to factor a number compared against multiplying the factors together to generate the number

Relatively Prime Numbers and GCD

- Two numbers are "relatively prime" if they have no common divisors apart from 1
  - Example: 8 and 15 are relatively prime as share no common factor other than 1
- Can determine the GCD of two numbers by comparing their prime factorization and using the least powers

$$300 = 2^1 * 3^1 * 5^2$$
$$18 = 2^1 * 3^2$$
$$GCD(18, 300) = 2^1 * 3^1 * 5^0 = 6$$

2 Four important results from number theory

Fermat's Theorem

$$a^{p-1} = 1(\text{mod } p)$$

where p is prime and $GCD(a, p) = 1$

- useful in both public key and primality testing

Euler's Totient Function $\varnothing(n)$

- Recall: set of residues are set of numbers from 0 to $n - 1$ when doing arithmetic modulo n
- *reduced set of residues*: those residues which are relatively prime to n
  - Example: For $n = 10$, set of residues is $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ and the reduced set of residues is $1, 3, 7, 9$
- *Euler's Totient Function* $\varnothing(n)$: the size of a reduced set of residues

Euler's Totient Function $\varnothing(n)$

- Computing $\varnothing(n)$ requires us to count the number of residues to be excluded
- In general, requires prime factorization of n
- Some quick special cases
  - For a prime number p, $\varnothing(n) = p - 1$
  - For a product of two primes p and q, $\varnothing(pq) = (p - 1)(q - 1)$
  - Example:

$$\varnothing(37) = 36$$
$$\varnothing(21) = (3 - 1) * (7 - 1) = 2 * 6 = 12$$

Euler's Theorem

- A generalization of Fermat's Theorem

$$a^{\varnothing(n)} = 1(mod(n))$$

  for any a,n where $gcd(a, n) = 1$

- Or we can write this as

$$a^{\varnothing(n)+1} = a(mod(n))$$

Primality Testing

- Often need to randomly generate very large prime numbers
- This is computationally painful
- *Sieve using trial division*: divide by all possible prime factors in turn less than the square root of the number
  - Good only for small numbers
- Can use statistical tests on the prime numbers
  - Which all prime numbers meet
  - But so do some pseudo-primes, composite numbers that also have this property
- Some very slow deterministic primality tests exist

Primality Testing: Miller Rabin Algorithm

Algorithm 1: Miller Rabin Algorithm for testing primality

Input: A candidate prime number n
Output: A indication that a number is either composite or a candidate prime
Find numbers $k > 0$ and odd number q such that $(n-1) = 2^k q$;
Select a random integer a, such that $1 < a < (n-1)$;
if $a^q \ mod\ n = 1$ **then**
    return "inconclusive";
for $j \in [0 \dots (k-1)]$ **do**
    if $a^{2^j} \ mod\ n = n-1$ **then**
        return "inconclusive";
return "composite"

Primality Testing: Miller Rabin Algorithm

- Chance this algorithm detects a pseudo-prime is 0.25
- Repeat test with different random number a for a number of tests, so decreases the probability that the number of pseudo-prime
- Then can use one of the deterministic tests

Can take advantage of the prime number theorem

- Theorem states that prime numbers occur every $ln(n)$ numbers on average
  - And can immediately ignore even numbers
- So only need to test $\frac{ln(n)}{2}$ numbers of size n to locate a prime

Chinese Remainder Theorem

- If doing modulus arithmetic, then a product of numbers can be computed by working in each moduli m_i separately
- Much faster than working in the full modulus M

Chinese Remainder Theorem

To compute $A \mod M$)

- compute all $a_i = A \mod m_i$ separately
- Determine constants c_i such that

$$c_i = \left(M_i * M_i^{-1}\right) \mod m_i$$

- and combine everything back together with

$$A = \left(\sum_{i=1}^{k} a_i c_i\right) \mod M$$

3 Public key vs. private key systems

Private Key Cryptography

- Until recently, all cryptographic systems have been based on the elementary systems of substitution and permutation
- Such systems use *ONE* key shared by both sender and receiver
- This puts them into the class of private/secret/single key (symmetric) systems
- If the shared key is disclosed communications are compromised
- As this is a symmetric system, does not protect sender from receiver forging a message and claiming it was sent by the sender

Public-Key Cryptography

- Viewed as most significant advancement in the history of cryptography
- *Asymmetric*: uses two keys - a public and a private key
- Uses a clever application of number theory
- Complements rather than replaces private key cryptography

Motivation and history

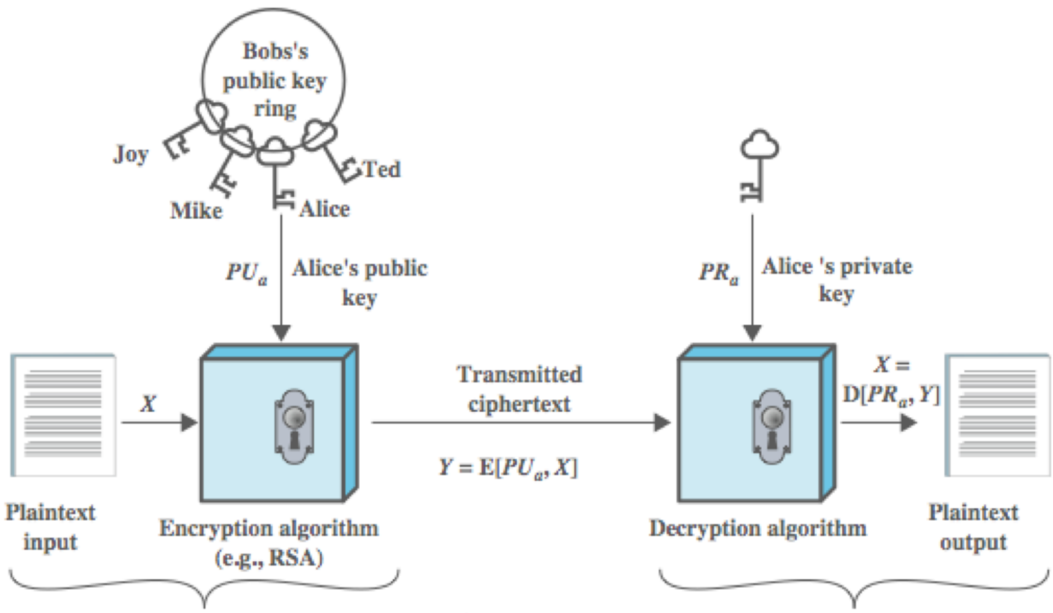
- Addresses two critical issues
  - *key distribution*: how to have secure communications in general without having to trust someone or something with your key
  - *digital signatures*: how to verify a message comes intact from the claimed sender
- Public invention reported in a 1976 paper by Diffie and Hellman
  - Was known much earlier in the classified community

4 Public-key cryptography

General overview of the process

- Two keys
  - *public key*: known to all and used to encrypt messages and verify signatures
  - *private key*: related to the public key but known only by the recipient, used to decrypt messages and sign (create) signatures
- Designed so that infeasible to determine private key from public key
- Is asymmetric because
  - those who encrypt messages of verify signatures cannot decrypt messages or create signatures

Process overview



(a) Encryption with public key

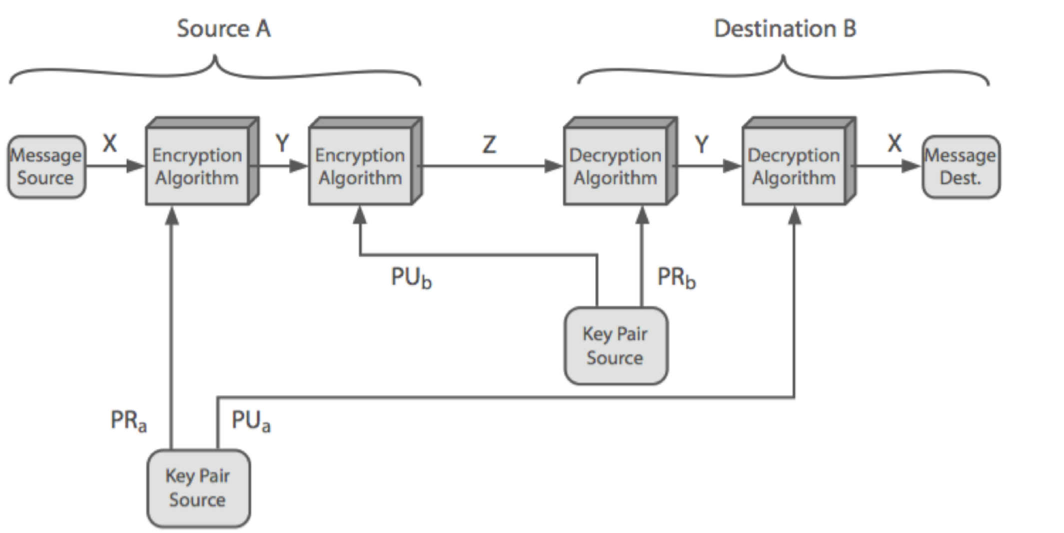
Process overview

- Plaintext: readable message/data that is input to the algorithm
- Encryption algorithm: Transform plaintext to ciphertext
- Public/Private keys: key pair selected so that one if one used for encryption, then the other is used for decryption
- Ciphertext: scrambled message generated as output
- Decryption algorithm: accepts ciphertext and matching key and outputs the plaintext

Symmetric vs. Public-Key comparison

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 and Authentication



Public-Key Applications

- 3 categories

- encryption/decryption (provide secrecy):
    - sender encrypts message with recipient's public key
- digital signature (provide authentication):
    - sender "signs" a message with its private key, either to the whole message or a small block that is a function of a message
- key exchange (of session keys):
    - Two sides cooperate to exchange a session key

Public-Key Requirements

- A key-pair must satisfy the following requirements
    - It must be computationally infeasible to find a decryption key knowing only algorithm and encryption key
    - It must be computationally easy to encrypt or decrypt messages when the relevant en/decryption key is known
    - Either of the two related keys can be used for encryption with the other used for decryption (for known algorithms)
- Hard to meet these requirements, only a few algorithms have been proposed in the last 40 years since the concept was proposed

5 Public key vs. private key systems

Private Key Cryptography

- Until recently, all cryptographic systems have been based on the elementary systems of substitution and permutation
- Such systems use *ONE* key shared by both sender and receiver
- This puts them into the class of private/secret/single key (symmetric) systems
- If the shared key is disclosed communications are compromised
- As this is a symmetric system, does not protect sender from receiver forging a message and claiming it was sent by the sender

Public-Key Cryptography

- Viewed as most significant advancement in the history of cryptography
- *Asymmetric*: uses two keys - a public and a private key
- Uses a clever application of number theory
- Complements rather than replaces private key cryptography

Motivation and history

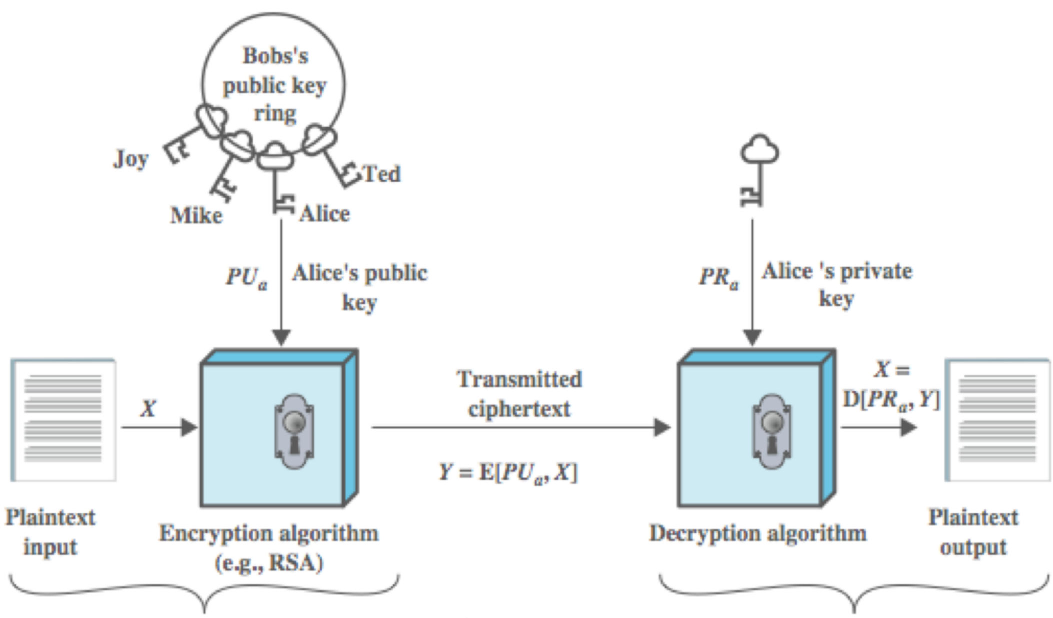
- Addresses two critical issues
  - *key distribution*: how to have secure communications in general without having to trust someone or something with your key
  - *digital signatures*: how to verify a message comes intact from the claimed sender
- Public invention reported in a 1976 paper by Diffie and Hellman
  - Was known much earlier in the classified community

6 Public-key cryptography

General overview of the process

- Two keys
  - *public key*: known to all and used to encrypt messages and verify signatures
  - *private key*: related to the public key but known only by the recipient, used to decrypt messages and sign (create) signatures
- Designed so that infeasible to determine private key from public key
- Is asymmetric because
  - those who encrypt messages of verify signatures cannot decrypt messages or create signatures

Process overview

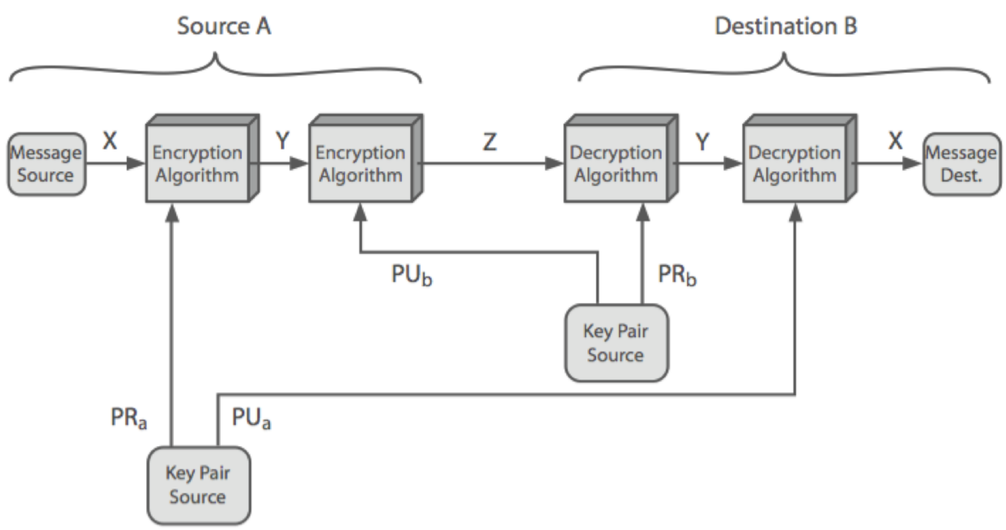


(a) Encryption with public key

Process overview

- Plaintext: readable message/data that is input to the algorithm
- Encryption algorithm: Transform plaintext to ciphertext
- Public/Private keys: key pair selected so that one if one used for encryption, then the other is used for decryption
- Ciphertext: scrambled message generated as output
- Decryption algorithm: accepts ciphertext and matching key and outputs the plaintext

Secrecy and Authentication



Public-Key Applications

- 3 categories
  - encryption/decryption (provide secrecy):
    * sender encrypts message with recipient's public key
  - digital signature (provide authentication):
    * sender "signs" a message with its private key, either to the whole message or a small block that is a function of a message
  - key exchange (of session keys):
    * Two sides cooperate to exchange a session key

Public-Key Requirements

- A key-pair must satisfy the following requirements
  - It must be computationally infeasible to find a decryption key knowing only algorithm and encryption key

- It must be computationally easy to encrypt or decrypt messages when the relevant en/decryption key is known
- Either of the two related keys can be used for encryption with the other used for decryption (for known algorithms)

- Hard to meet these requirements, only a few algorithms have been proposed in the last 40 years since the concept was proposed

7 What is the RSA cipher?

What is the RSA cipher

- Developed in 1977 by Rivest, Shamir, and Adleman
- Based on
  - Exponentiation in a Galois field over intgers modulo a prime number
  - Uses large numbers (typically 1024 bits)
  - Security due to cost of factoring large numbers

8 Working the RSA cipher

RSA en/decryption

- Plaintext is encrypted in blocks
  - Each block must have a binary value less than n
- Encryption and decryption are just a single exponentiation mod(n)
  - Sender encrypts by obtaining public key of recipient $PU = (e, n)$
    * computes $C = M^e$, mod n, where $0 \leq M < n$
  - Recipient decrypts by using their private key $PR = (d, n)$
    * computes $M = C^d$, mod n
- The "magic trick" is the choice of the modulus and exponents

RSA Key Setup

- Select a public/private key pair by
  - randomly selecting two large primes p and q
  - computing their system modulus $n = p.q$
    * note that $\varnothing(n) = (p-1) * (q-1)$
  - Selecting at random the encryption key e
    * $1 < e < \varnothing(n)$
    * $gcd(e, \varnothing(n)) = 1$
  - Solve the equation
$$e.d = 1 mod \varnothing(n) \text{ and } 0 \leq d \leq n$$
  - Publish their public encryption key $PU = (e, n)$
  - Keep their secret decryption key $PR = (d, n)$

RSA Key Setup: Example

1. Select primes: $p = 17$ & $q = 11$
2. Calculate: $n = pq = 17 * 11 = 187$
3. Calculate: ∅(n)=(p-1)*(q-1)=16*10=160
4. Select: $e : gcd(e, 160)$; choose $e = 7$
5. Determine : $d : d * e = 1 mod 160$ and $d < 160$. Value is $d = 23$ as $23 * 7 = 161 = 10 * 160 + 1$
6. Publish public key: $PU = \{7, 187\}$
7. Keep secret key $PR = 23, 187$

RSA En/Decryption: Example

- Given message $M = 88$. Note: $88 < 187$
- Encryption: $C = 88^7 mod 187 = 11$
- Decryption: $M = 11^{23} mod 187 = 88$

Why RSA works

- Recall Euler's Theorem
- In RSA:

$$\begin{gathered} n = p.q \\ \varnothing(n) = (p-1) * (q-1) \end{gathered} \tag{1}$$

- Carefully chose e and d to be inverses mod $\varnothing(n)$
- Thus, $e.d = 1 + k.\varnothing(n)$ for k
- And from Euler's Theorem, by correctly choosing e and d, we get a situation where raising a number to those powers in succession results in the original number!

Making RSA encryption efficient

- Important aspects of exponentiation to power e
- Want to get e small
  - Often use 65537 (2^{16}-1)
  - Also see either 3 or 17
  - Binary representation of these numbers have only two 1 bits
- But if e too small, then RSA becomes vulnerable
  - Use Chinese Remainder Theorem and 3 messages with different moduluii
- Key selection fixes e s.t. relatively prime to $\varnothing(n)$
  - Must ensure that $GCD(e, \varnothing(n)) = 1$

Making RSA decryption efficient

- Want large values of d to avoid brute-force attacks
- There are ways to apply the Chinese Remainder Theorem to compute mod p and q separately and then combine to get desired answer
- Note only owner of private key who knows values of p and q can use this technique

Security of RSA

- Brute force key search - infeasible given size of numbers
- Mathematical attacks - based on difficulty of computing $\varnothing(n)$ by factoring mod n
- Timing attacks - on running of decryption
- Chosen ciphertext attacks

Security of RSA: Factoring

- Factoring algorithms are computationally difficult
- Slow improvements in finding better algorithms
- Currently assume that 1024-2048 bit RSA is secure

Security of RSA: Timing attacks

- Developed in the mid-1990s
- Exploits timing variations in operations
  - example: multiplying by small vs. large number
- Infer operand size based on time taken
- Countermeasures
  - Use constant exponentiation time algorithms
  - Add random delays
  - Blind values used in calculations

Security of RSA: Chosen Ciphertext Attack (CCA)

- *CCA*: attack in which adversary chooses a number of ciphertexts and then is given the corresponding plaintexts, decrypted with the target's private key
- Use mathematical properties of the RSA to selects blocks of data that when processed with target's private key yield information needed for cryptoanalysis
- Can counter with random pad of plaintext. More sophisticated padding techniques exist

9 Key Points

Key Points

- Number Theory
  -
  - Prime numbers
  - Fermat's Theorem, Euler's Theorem, and Euler's Totient Function
  - Primality Testing
  - Chinese Remainder Theorem
- Public-Key Cryptography
  - What is it?
  - Applications
  - How does the RSA algorithm operate?