Today's topics

- Security
  - Demo from RSA Security (www.rsa.com)
  - *Sildes taken from Tammy Bailey*
  - *Slides taken from Kevin Wayne & Robert Sedgewick at Princeton University*
  - *For further reference "Applied Cryptography" by Bruce Schneier*
- Upcoming
  - Complexity
- Reading
  - Sections 3.5, 4.5 and 11 in *Brookshear.*
  - Chapters 11,13 in *Great Ideas.*

Security

- Computer Security is the prevention of, or protection against:
  - Access to information by unauthorized recipients
  - Intentional but unauthorized destruction or alteration of that information.
- Authentication: verifying the identity of a person or system
  - Username and Password
  - What is an example of a good password?
  - Change your password *often.* A particular implementation of this idea is ONE-TIME PASSWORDS.
  - Physical security of the system is also important.

Cryptography

Cryptography: science of creating secret codes.
Cryptanalysis : science of code breaking
Cryptology: science of secret communication.
Goal: Information Security in presence of malicious adversaries.

- Confidentiality…
- Integrity…
- Authentication…
- Authorization…
- Non-repudiation…

RSA PRESENTATION

Information security

- All measures taken to prevent unauthorized use of electronic data
  - unauthorized use includes disclosure, alteration, substitution, or destruction of the data concerned
- Provision of the following three services
  - Confidentiality
    - concealment of data from unauthorized parties
  - Integrity
    - assurance that data is genuine
  - Availability
    - system still functions efficiently after security provisions are in place
- No single measure can ensure complete security

Why is information security important?

- Governments, commercial businesses, and individuals are all storing information electronically
  - compact, instantaneous transfer, easy access
- Ability to use information more efficiently has resulted in a rapid increase in the value of information
- Information stored electronically faces new and potentially more damaging security threats
  - can potentially be stolen from a remote location
  - much easier to intercept and alter electronic communication than its paper-based predecessors

Building blocks of a secure system

- Confidentiality: concealment from unauthorized parties
  - identification – unique identifiers for all users
  - authentication
    - user: assurance that the parties involved in a real-time transaction are who they say they are
    - data: assurance of message source
  - authorization - allowing users who have been identified and authenticated to use certain resources
- Integrity: assurance the data is has not been modified by unauthorized parties
  - non-repudiation
    - proof of integrity and origin of data which can be verified by any third party at any time

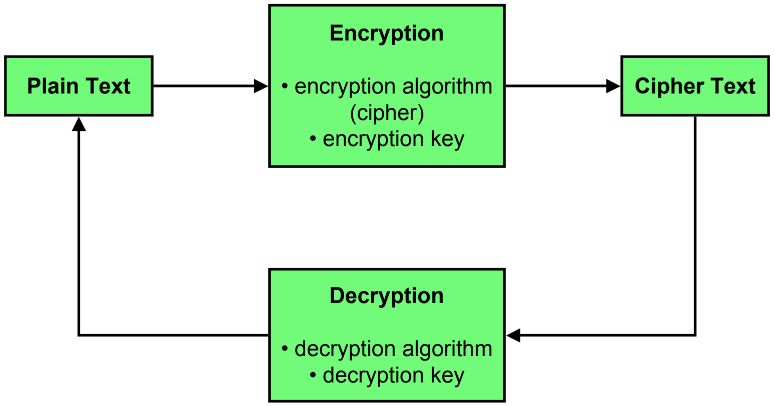
Completing the security process

- Confidentiality + integrity → system security
- However, it is not enough for system to be secure
- System must also be available
  - must allow guaranteed, efficient and continuous use of information
  - security measures should not prohibitively slow down or crash system or make it difficult to use
    - what good is a secure system if you can't use it?
- Cryptographic systems
  - high level of security and flexibility
  - can potentially provide all objectives of information security: confidentiality, integrity, and availability

Encryption

- Goal: information security in presence of malicious adversaries
  - confidentiality
  - integrity
  - authentication
  - authorization
  - non-repudiation
- Encryption can be used to …
  - prevent your kid sister from intercepting, reading, and/or altering your messages and files
  - prevent CIA or FBI from intercepting, reading, and/or altering your messages and files

Process



Terminology

- *Encryption*
  - process of obscuring or scrambling data to render it incomprehensible to unauthorized viewers.
- *Cipher text*
  - encrypted data or "code"
- *Plain text*
  - original, readable data prior to encryption
- *Cipher* or *encryption algorithm*
  - particular method for encrypting or scrambling data
- *Key*
  - data required by the encryption algorithm to process the plain text and convert it to cipher text
- *Decryption*
  - process of converting cipher text back into plain text
  - requires a key and a decryption algorithm

Algorithms & Keys

Restricted Algorithm

- If the security depends on keeping the working of the algorithm secret.
- Can't support a large or changing group of users…Why?
- No quality control.

Modern cryptology solves this with a KEY (*K*).

- Key might be any of a large number of values.
- Range of possible values called a *keyspace.*
- Now security depends on the security of the Key.
- The algorithms for encrypting and decrypting can be mass produced and optimized.

Attacks

- Compromise systems in ways that affect services of information security
  - attack on confidentiality:
    - unauthorized disclosure of information
  - attack on integrity:
    - destruction or corruption of information
  - attack on availability:
    - disruption or denial of services

Prevention, detection, response

- proper planning reduces risk of attack and increases capabilities of detection and response if an attack does occur

Attacks!

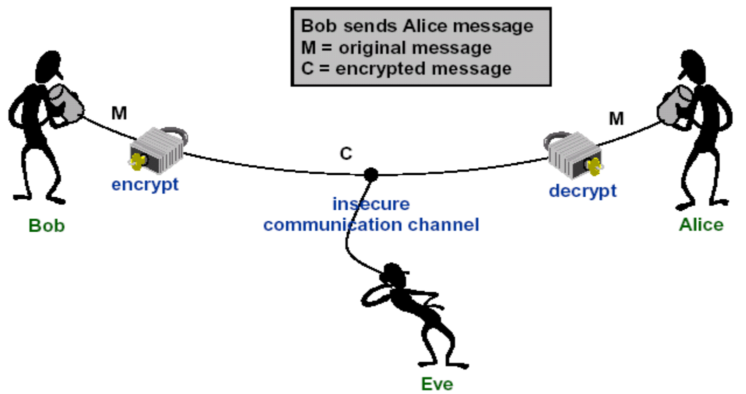
- Ciphertext-only Attack..
- Known-plaintext Attack..
- Chosen-plaintext Attack..
- Chosen-ciphertext Attack..
- Rubber-hose cryptanalysis..

Participants

- Sender & Receiver
  - **people who want to communicate securely or in private**
- Listener **(eavesdropper)**
  - **present on communication channel between sender and receiver**
- The Problem:

> *Suppose that Bob (the sender) wants to send Alice (the receiver) a message but knows that Eve (the eavesdropper) is trying and may very well intercept it. Bob and Alice need to agree on an encryption algorithm and a key. But Eve could intercept this as well.*
>
> *How do they get around this problem?*

Encrypted communication



Substitution Ciphers

- **Each character in the message is replaced by another according to some rule**
- **Order of the encrypted characters is the same as plaintext**
  - Caesar cipher
    - **letters of the alphabet shifted by 3 positions**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- **Shift (additive) ciphers**
  - **letters of the alphabet are shifted by *k* positions**
  - ***k* is called the cipher or** encryption key

Substitution ciphers are easy to break

- Shift ciphers really only have 25 keys
  - same ciphertext results from keys 10, 35, -20, 510, …
  - easy to try all possible keys
- What if we randomly order the alphabet? 26! possibilities

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
L	C	F	R	Q	W	Z	K	M	G	B	X	D	S	Y	N	T	A	U	J	V	O	H	P	E	I

- Still (relatively) easy to break using characteristics of the language to reduce solution space
  - letter and word frequencies
  - context

Additive tables & one time pads

- Lists of random numbers
- Shift first letter of message by first number, shift second letter by second number, etc. until message is completed
- Harder to break because individual letters are not always encrypted to same code letter
- Problem is both sender and receiver must have a copy of the table and/or know where to start in the table
- If the same table is used every time, code can be broken by analyzing enough messages

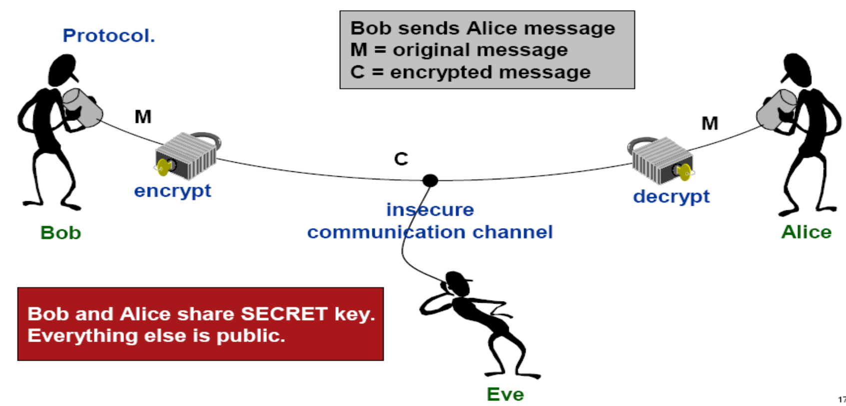
Encryption algorithms

- Symmetric Key
  - perform encryption and decryption with a single key
  - substitution ciphers
- Examples
  - DES/3DES
  - Blowfish
  - IDEA

- Asymmetric Key
  - separate keys used for encryption and decryption
    - public key
    - private key
- Examples
  - RSA
  - DSA

Private Key Encryption

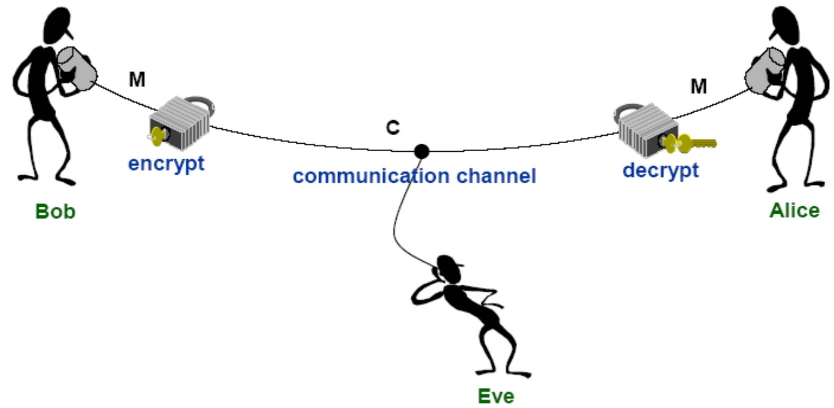
Assume message is encoded as numbers (ASCII, Unicode)



Symmetric key algorithms (private key)

- Perform encryption and decryption with a single key
- Advantages
  - ➢ algorithms are very fast
  - ➢ computationally less intensive
- Security of system determined by protecting the secret key from disclosure
- Applicable only in situations where the distribution of the key can occur in a secure manner
- *If every user is going to communicate with every other user, how many keys are required for a system with 1000 users?*

Public Key Encryption



Two different keys:
Alice's PUBLIC key locks, her PRIVATE key opens.
Everything else is public.

Asymmetric algorithms (public key)

- Two separate keys used for encryption and decryption
  - ➢ public key
    - used for encryption, not secret, available for widespread dissemination
  - ➢ private key
    - used for decryption
    - private to the individual who owns it
- Plain text encrypted with one key can be decrypted with the other key only
  - ➢ similar to a mailbox
- Computationally infeasible to derive the private key from the known public key
- *If every user is going to communicate with every other user, how many keys are required for a system with 1000 users?*

Padlock problem

- Al and Sue are not allowed to directly communicate with each other in any way. Al has a box, a padlock for the box, a key for that padlock, and a diamond. Sue has a different padlock, and a key for that padlock. The only way Al and Sue can communicate, or send things to each other is through Ted, who will steal everything except a locked box, or an empty box. Ted will not try to pry open any locks with any tools, etc. But if a box is unlocked, and not empty, then Ted will steal its contents.

- *Question*: How does Al get the diamond to Sue using Ted?

RSA encryption

- Rivest, Shamir, and Adleman, MIT, 1977
- Most widely-used cryptosystem
- Security relies on the on the difficulty of factoring very large integers into prime factors
  - primes are positive integers that are divisible only by 1 and themselves
  - for example, first 50 prime numbers are …
    2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229

Prime factorization

- A prime factorization is the expression of a positive integer as a product of prime numbers

$$\begin{aligned} 12 &= 3 \times 2 \times 2 \\ 4453 &= 73 \times 61 \\ 10584 &= 7 \times 7 \times 3 \times 3 \times 3 \times 2 \times 2 \times 2 \\ 124937125 &= 2003 \times 499 \times 5 \times 5 \times 5 \end{aligned}$$

- Large primes are easy to multiply
- Factoring large integers is hard

easy

8876044532898802067 = 1500450271 × 5915587277

hard

Encrypting and decrypting

- Alice and Bob would like to communicate with each other in private
- Alice uses RSA algorithm to generate public & private keys
  - Alice makes key (k, n) publicly available to Bob and anyone else wanting to send her private messages
- Bob uses Alice's public key (k, n) to encrypt message M:
  - compute $E(M) = (M^k)\%n$
  - Bob sends encrypted message E(M) to Alice
- Alice receives E(M) and uses private key (d, n) to decrypt it:
  - compute $D(M) = (E(M)^d)\%n$
  - decrypted message D(M) is original message M

RSA algorithm

- Select two large prime numbers p, q
- Compute
  $n = p \times q$
  $v = (p-1) \times (q-1)$
- Select small odd integer k relatively prime to (not a factor of) to v
- Compute d such that
  $(d \times k)\%v = (k \times d)\%v = 1$
- Public key is (k, n)
- Private key is (d, n)
- How large should n be?
  - Number Theory
  - $n / \ln n$ prime numbers between 2 and n.

- example
  p = 11
  q = 29
  n = 319
  v = 280
  k = 3
  d = 187
- public key
  (3, 319)
- private key
  (187, 319)

RSA Attacks

Factoring.

- **Factor n = pq.**
- **Then compute** f.
- **Then compute e.**

Timing attacks.

- **Reconstruct d by sending C and monitoring how long it takes to compute C^d(mod n).**

Other means?

- **Long-standing open research question.**

Digital Signature

Alice sends Bob a response.

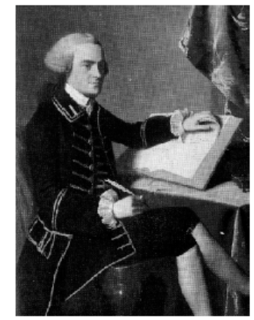
- **Bob wants to be really sure Alice really sent it, and not some imposter.**

Alice wants to send Bob a response S.

- **Alice uses private key d and computes: S'= S^d (mo**
- **Alice sends (S, S').**

Bob receives digital signed response (S, S') .

- **Bob uses Alice's public key e**
  - **Checks if S = $(S')^e$ (mod n).**
- **If yes, then Bob concludes S sent by Alice.**
- **If no, then Bob concludes S or S' corrupted in tran or message is forgery.**



Third party.

- **Bob verifies Alice's signature on digitally signed m (e.g. electronic check).**
- **Bob forwards digitally signed message to bank.**
- **Bank re-verifies Alice's signature.**

Certification authority

- A third party trusted by all users that creates, distributes, revokes, & manages *certificates*
- Certificates bind users to their public keys
- For example, if Alice wants to obtain Bob's public key
  - she retrieves Bob's certificate from a public directory
  - she verifies the CA's signature on the certificate itself
  - if signature verifies correctly, she has assurance from the trusted CA this really is Bob's public key
  - she can use Bob's public key to send confidential information to Bob or to verify Bob's signatures, protected by the assurance of the certificate
- Integrity is provided by the certification authority

Bad Cryptology.

Good introductory explanation & details on Gregory Kesden's site (CMU)
http://www-2.cs.cmu.edu/~dst/DeCSS/Kesden/

Content Scrambling System (CSS).

- **Use to encrypt DVD's.**
- **Each disc has 3 40-bit keys.**
- **Each DVD decoder (software/hardware) has unique 40-bit key.**
- **"Not possible" to play back on computer without disc.**

DeCSS. (Canman and SoupaFrog, 1999).

- **Decryption algorithm written by two Norwegians.**
- **Used "in-circuit emulator" to monitor hardware activity.**

Why CSS is fatally flawed. (Policy and Legal issues..)

Prevention

- Establishment of policy and access control
  - who: identification, authentication, authorization
  - what: granted on "need-to-know" basis
- Implementation of hardware, software, and services
  - users cannot override, unalterable (attackers cannot defeat security mechanisms by changing them)
  - examples of preventative mechanisms
    - passwords - prevent unauthorized system access
    - firewalls - prevent unauthorized network access
    - encryption - prevents breaches of confidentiality
    - physical security devices - prevent theft
- Maintenance

Prevention is not enough!

Prevention systems are never perfect.

No bank ever says: "Our safe is so good, we don't need an alarm system."

No museum ever says: "Our door and window locks are so good, we don't need night watchmen."

Detection and response are how we get security in the real world, and they're the only way we can possibly get security in the cyberspace world.

Bruce Schneier,
Counterpane Internet Security, Inc

Detection

- Determine that either an attack is underway or has occurred and report it
- Real-time monitoring
  - or, as close as possible
  - monitor attacks to provide data about their nature, severity, and results
- Intrusion verification and notification
  - intrusion detection systems (IDS)
  - typical detection systems monitor various aspects of the system, looking for actions or information indicating an attack
    - example: denial of access to a system when user repeatedly enters incorrect password

Outline of implementation

- RSA algorithm for key generation
  - select two prime numbers p, q
  - compute $n = p \times q$
    $v = (p-1) \times (q-1)$
  - select small odd integer k such that
    $\gcd(k, v) = 1$
  - compute d such that
    $(d \times k) \% v = 1$
- RSA algorithm for encryption/decryption
  - encryption: compute $E(M) = (M^k) \% n$
  - decryption: compute $D(M) = (E(M)^d) \% n$

RSA algorithm for key generation

- Input: none
- Computation:
  - select two prime integers p, q
  - compute integers $n = p \times q$
    $v = (p-1) \times (q-1)$
  - select small odd integer k such that $\gcd(k, v) = 1$
  - compute integer d such that $(d \times k) \% v = 1$
- Output: n, k, and d

RSA algorithm for encryption

- Input: integers k, n, M
  - M is integer representation of plaintext message
- Computation:
  - let C be integer representation of ciphertext
    $C = (M^k) \% n$
- Output: integer C
  - ciphertext or encrypted message

RSA algorithm for decryption

- Input: integers d, n, C
  - C is integer representation of ciphertext message
- Computation:
  - let D be integer representation of decrypted ciphertext
    $D = (C^d) \% n$
- Output: integer D
  - decrypted message

This seems hard …

- How to find big primes?
- How to find mod inverse?
- How to compute greatest common divisor?
- How to translate text input to numeric values?
- Most importantly: RSA manipulates big numbers
  - Java integers are of limited size
  - how can we handle this?
- Two key items make the implementation easier
  - understanding the math
  - Java's `BigInteger` class

What is a `BigInteger`?

- Java class to represent and perform operations on integers of arbitrary precision
- Provides analogues to Java's primitive integer operations, e.g.
  - addition and subtraction
  - multiplication and division
- Along with operations for
  - modular arithmetic
  - gcd calculation
  - generation of primes
- http://java.sun.com/j2se/1.5.0/docs/api/

Using `BigInteger`

- If we understand what mathematical computations are involved in the RSA algorithm, we can use Java's `BigInteger` methods to perform them

- To declare a `BigInteger` named B

```
BigInteger B;
```

- Predefined constants

```
BigInteger.ZERO
BigInteger.ONE
```


Randomly generated primes

```
BigInteger probablePrime(int b, Random rng)
```

- Returns random positive `BigInteger` of bit length `b` that is "probably" prime
  - probability that `BigInteger` is not prime < 2^{-100}

- `Random` is Java's class for random number generation
- The following statement

```
Random rng = new Random();
```

  creates a new random number generator named `rng`
- What about *randomized algorithms* in general?

`probablePrime`

- Example: randomly generate two `BigInteger` primes named `p` and `q` of bit length `32` :

```
/* create a random number generator */
Random rng = new Random();

/* declare p and q as type BigInteger */
BigInteger p, q;

/* assign values to p and q as required */
p = BigInteger.probablePrime(32, rng);
q = BigInteger.probablePrime(32, rng
```

Integer operations

- Suppose have declared and assigned values for `p` and `q` and now want to perform integer operations on them
  - use methods `add`, `subtract`, `multiply`, `divide`
  - result of `BigInteger` operations is a `BigInteger`
- Examples:

```
BigInteger w = p.add(q);
BigInteger x = p.subtract(q);
BigInteger y = p.multiply(q);
BigInteger z = p.divide(q);
```


Greatest common divisor

- The greatest common divisor of two numbers x and y is the largest number that divides both x and y
  - this is usually written as gcd(x,y)
- Example: gcd(20,30) = 10
  - 20 is divided by 1,2,4,5,10,20
  - 30 is divided by 1,2,3,5,6,10,15,30
- Example: gcd(13,15) = 1
  - 13 is divided by 1,13
  - 15 is divided by 1,3,5,15
- When the gcd of two numbers is one, these numbers are said to be relatively prime

Euler's Phi Function

- For a positive integer n, $\phi(n)$ is the number of positive integers less than n and relatively prime to n
- Examples:
  - $\phi(3) = 2$ 1,2
  - $\phi(4) = 2$ 1,2,3 (but 2 is not relatively prime to 4)
  - $\phi(5) = 4$ 1,2,3,4
- For any prime number p,

$$\phi(p) = p-1$$

- For any integer n that is the product of two distinct primes p and q,

$$\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$$

Relative primes

- Suppose we have an integer x and want to find an odd integer z such that
  - $1 < z < x$, and
  - z is relatively prime to x
- We know that `x` and `z` are relatively prime if their greatest common divisor is one
  - randomly generate prime values for z until gcd(x,z)=1
  - if x is a product of distinct primes, there is a value of z satisfying this equality

Relative `BigInteger` primes

- Suppose we have declared a `BigInteger x` and assigned it a value
- Declare a `BigInteger z`
- Assign a prime value to `z` using the `probablePrime` method
  - specifying an input bit length smaller than that of `x` gives a value `z<x`
- The expression

  ```
  (x.gcd(z)).equals(BigInteger.ONE)
  ```

  returns true if gcd(x,z)=1 and false otherwise
- While the above expression evaluates to false, assign a new random to `z`

Multiplicative identities and inverses

- The multiplicative identity is the element e such that

  $$e * x = x * e = x$$

  for all elements $x \in X$
- The multiplicative inverse of x is the element x^{-1} such that

  $$x * x^{-1} = x^{-1} * x = 1$$

- The multiplicative inverse of x mod n is the element x^{-1} such that

  $$(x * x^{-1}) \bmod n = (x^{-1} * x) \bmod n = 1$$

  - x and x^{-1} are inverses only in multiplication mod n

`modInverse`

- Suppose we have declared `BigInteger` variables `x`, `y` and assigned values to them
- We want to find a `BigInteger z` such that

  ```
  (x*z)%y =(z*x)%y = 1
  ```

  that is, we want to find the inverse of `x` mod `y` and assign its value to `z`

- This is accomplished by the following statement:

```
BigInteger z = x.modInverse(y);
```