Cryptography

Greek: kryptos + graphein → hidden writing

Cryptography

- Privacy and security needed while communicating over insecure media (internet)
- In the past, cryptography is heavily used for military applications to keep sensitive information secret from enemies (e.g. Caesar cipher)
- Nowadays, with the technologic progress as our dependency on electronic systems has increased we need more sophisticated techniques.
- Cryptography provides most of the methods and techniques for a secure communication

Terminology

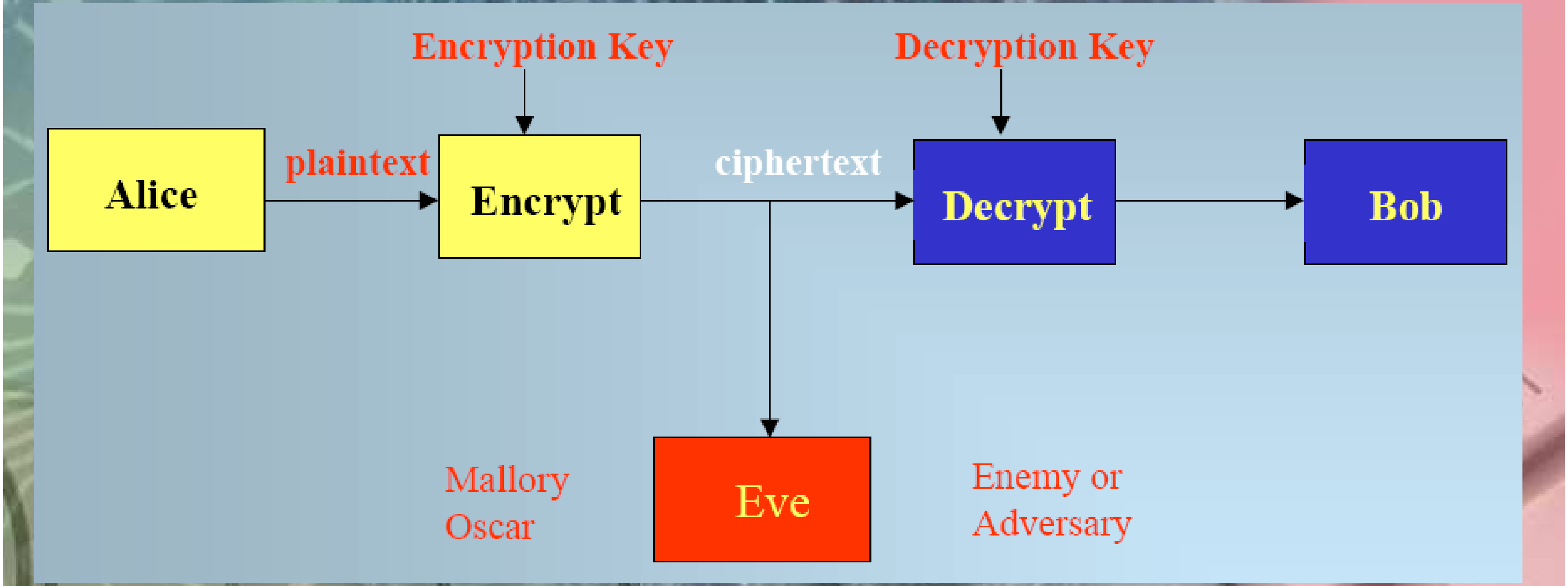
- **Cryptology**
  - All-inclusive term used for the study of secure communication over non-secure channels and related problems.
- **Cryptography**
  - The process of designing systems to realize secure communications over non-secure channels
- **Cryptoanalysis**
  - The discipline of breaking the cryptographic systems
- **Coding Theory**
  - Deals with representing the information using codes. It covers: compression, secrecy, and error-correction. Recently, it is predominantly associated with error-correcting codes which ensures the correct transmissions over noisy-channels.

Cryptography

- depends on:
  - mathematics & usage of digital systems
- Inter-disciplinary study of three fields:
  - Mathematics
  - Computer Science
  - Electrical Engineering

  } Computer Engineer
- The importance of crypto-analysis
  - Without having a complete understanding of crypto-analysis (or crypto-analytic techniques) it is impossible to design *good* (secure, unbreakable) cryptographic systems
- Other Disciplines
  - It makes use of other disciplines such as error-correcting codes, compression

Secure Communication



Encryption Key

Decryption Key

Alice

plaintext

Encrypt

ciphertext

Decrypt

Bob

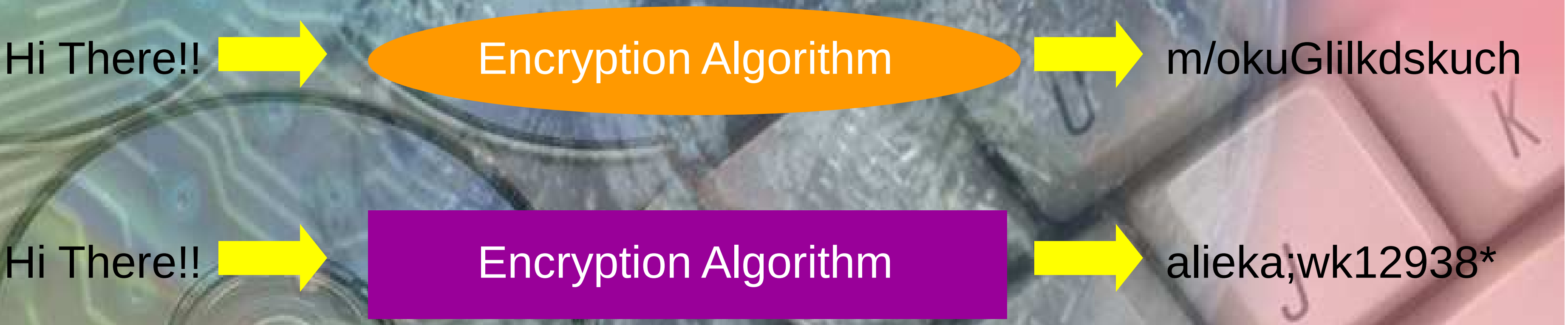
Mallory
Oscar

Eve

Enemy or
Adversary

Encryption

- Convert normal, readable data into obscured, unreadable data

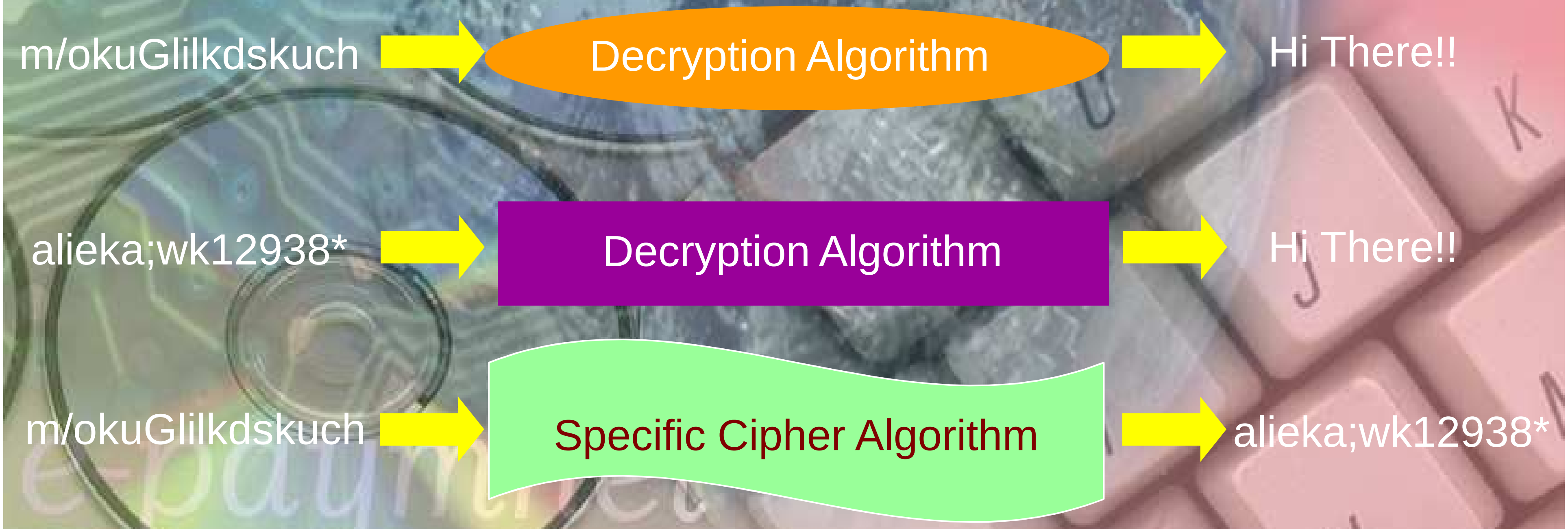


Hi There!! → Encryption Algorithm → m/okuGlilkdskuch

Hi There!! → Encryption Algorithm → alieka;wk12938*

Decryption

- Convert obscured, unreadable data into normal, readable data



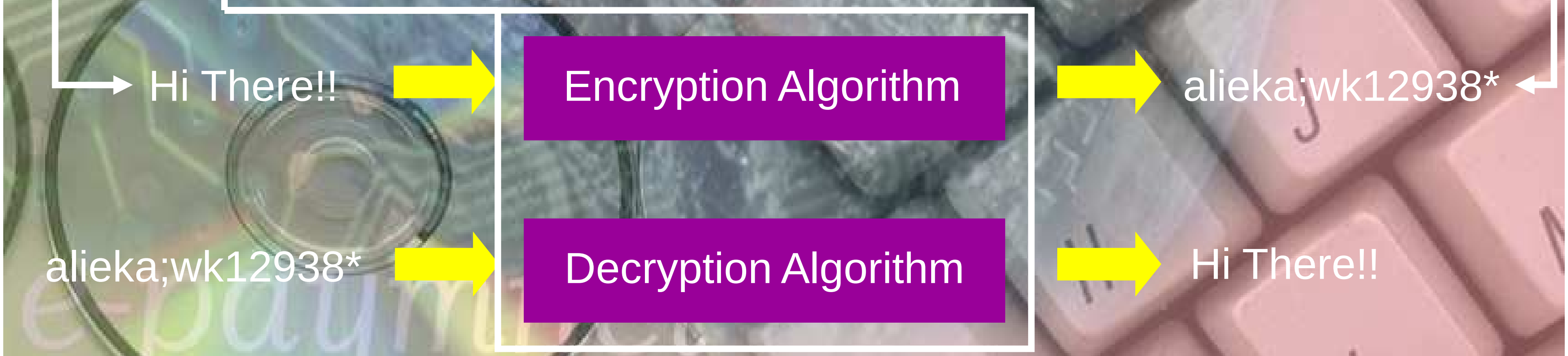
m/okuGlilkdskuch → Decryption Algorithm → Hi There!!

alieka;wk12938* → Decryption Algorithm → Hi There!!

m/okuGlilkdskuch → Specific Cipher Algorithm → alieka;wk12938*

Terminology

- plaintext - clear readable text
- ciphertext - unreadable text
- cipher - algorithm(s) for encryption and decryption

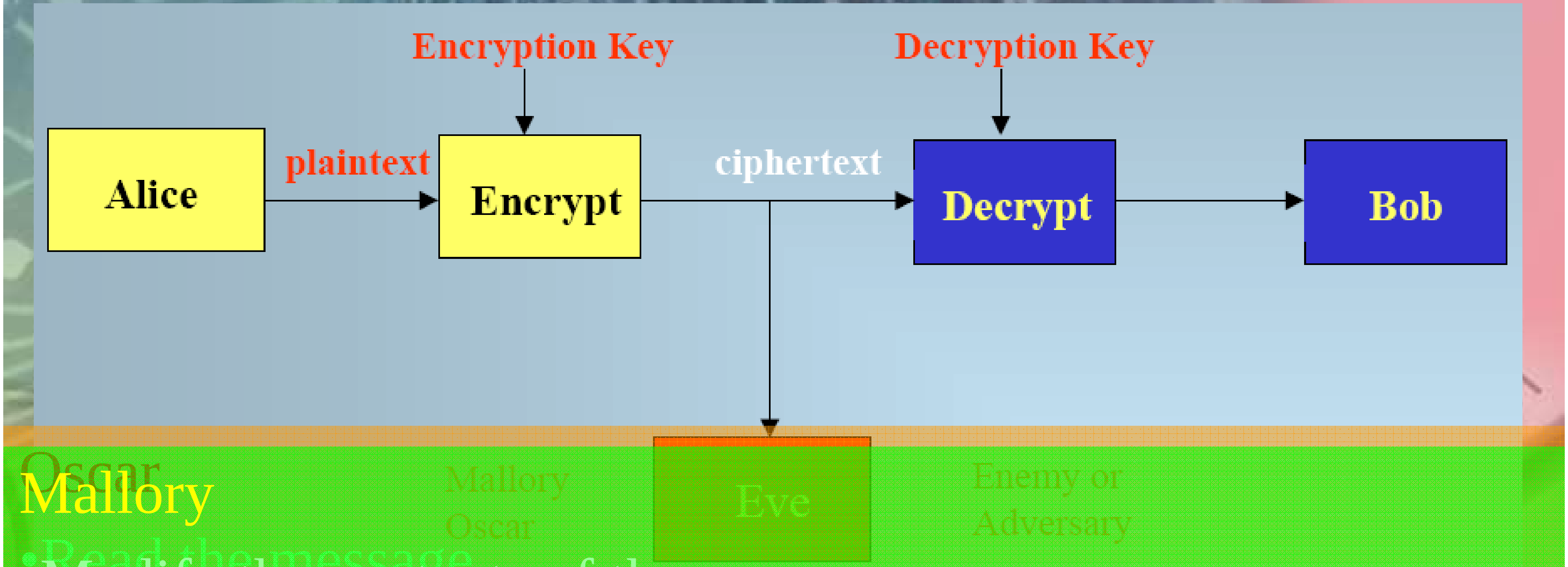


Hi There!! → Encryption Algorithm → alieka;wk12938*

alieka;wk12938* → Decryption Algorithm → Hi There!!

Secure Communication

Eve's Goals



Oscar

- Read the message
- Figure out Alice's key
  - Read all messages encrypted with this key

Mallory

- Modify the contents of the message
  - Bob will think Alice sent the altered message
- Impersonate Alice
  - communicate with Bob who thinks he is communicating with Alice

Attack Means

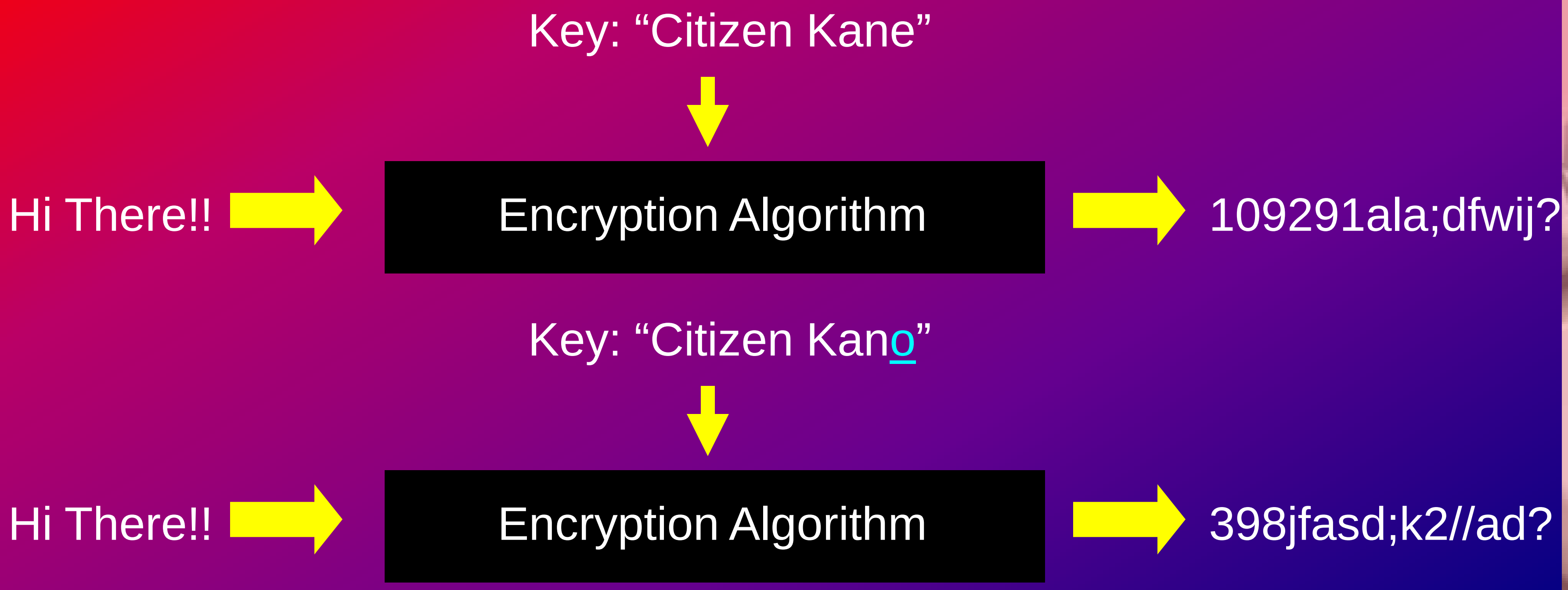
- Ciphertext only
  - Alice has only a copy of ciphertext
- Known Plaintext
  - Eve has a copy of ciphertext and the corresponding plaintext and tries to figure out the key
- Chosen Plaintext
  - Eve can have a ciphertext corresponding to a sample plaintext which she believes is useful to figure the key
- Chosen Ciphertext
  - Eve can have a plaintext corresponding to a sample ciphertext which she believes is useful to figure the key

Terminology

- Security through obscurity
  - Don't publish some details of your algorithm... assuming people won't figure it out
  - Like hiding the key under the doormat
- Once your flaw/algorithm is leaked, you're screwed

Terminology

- Key -- a secret piece of information that controls how the encryption algorithm works
- Different keys produce different encrypted results



Key: “Citizen Kane”

Hi There!! Encryption Algorithm 109291ala;dfwij?

Key: “Citizen Kano”

Hi There!! Encryption Algorithm 398jfasd;k2//ad?

Kerckhkoffs Principle

- Complete knowledge of the Algorithm
  - While assessing the strength of a cryptosystem, one should always assume that the enemy knows the cryptographic algorithm used
- The security of the system, therefore, should be based on
  - the quality (strength) of the algorithm but not its obscurity or darkness
  - the key space (or key length)

Computer Era

- Moore's law and its implications
- Keys breakable by brute force

Problem

- Monoalphabetic
  - Same letter of plaintext always produces same letter of ciphertext
- Even though there are 26!
  - possible substitutions, monoalphabetic solutions are easy to break!

Modern Ciphers

- Bigger and bigger keys
- More and more complicated algorithms
- Based on hardcore applied mathematics... and the difficulty of factoring large numbers

Terminology

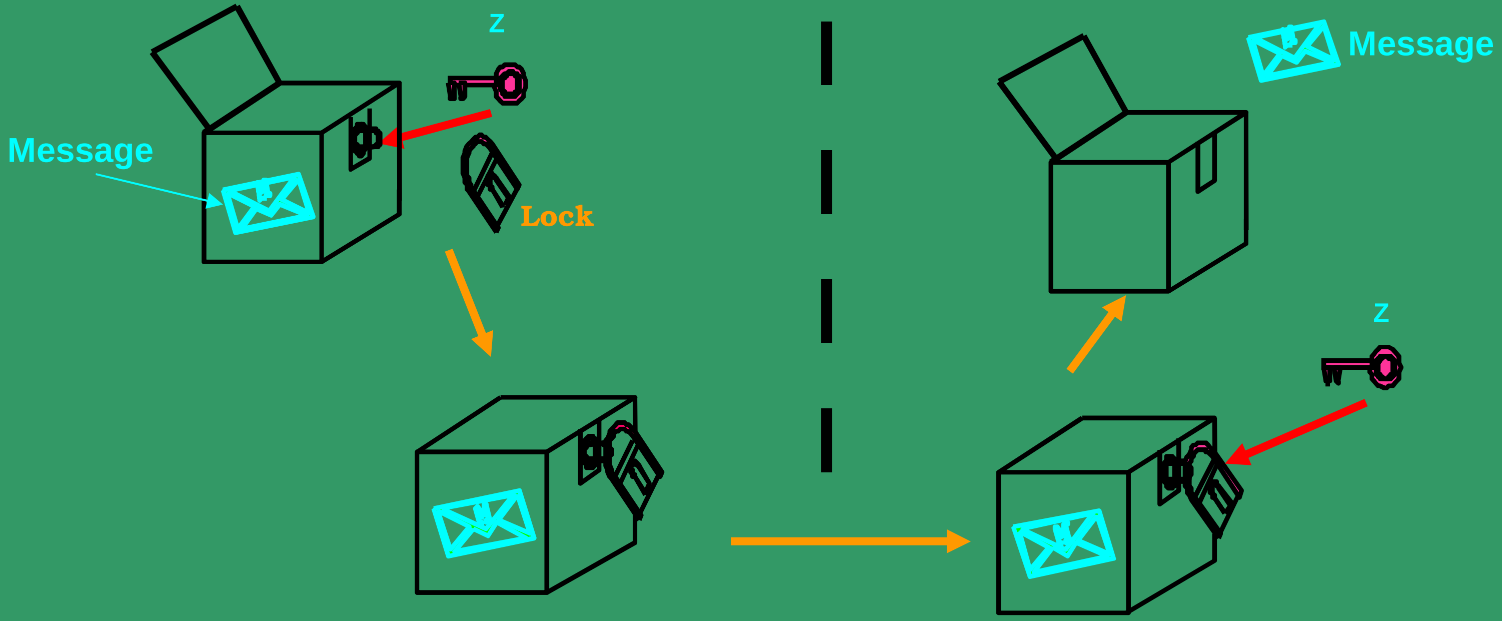
- Symmetric key cryptography
  - Caesar shift, ..., DES, AES
- Asymmetric key cryptography
  - Public/Private key schemes

Symmetric key algorithms

Sender

Receiver

Key = Z ↔ **Same** ↔ **Key = Z**



Symmetric Key Technology

- p = plaintext
- crypt() = encryption/decryption function
- c = cipher text (unreadable)
- k = key (secret; password)

Symmetric Key Technology

- Alice wants to send a private/confidential message to Bob
- Alice computes c=crypt(p,k)
- Sends c to Bob over unsecured wire
- Bob computes p=crypt(c,k)

Symmetric Key Application

- Password login
- Alice sends password to computer to prove identity (authenticity)

Shared Secret Key

- Shared secret is great... but how do we distribute it?

Public Key (Assymetric) Cryptosystems (PKC)

- **Why public key cryptography ?**
- Key Distribution and Management is difficult in symmetric cryptosystems: DES, AES (Rijndael) over large networks.
- Electronic Signatures
- Other cryptographic functions such:
  - Key Exchange
  - Secret Key Derivation
  - Secret Sharing functions

Asymmetric Key Cryptography

- Instead of one key, have two
  - public key
  - private key

- Use one key to encode/encrypt
- Use other key to decode/decrypt

Fundamentals of PKC

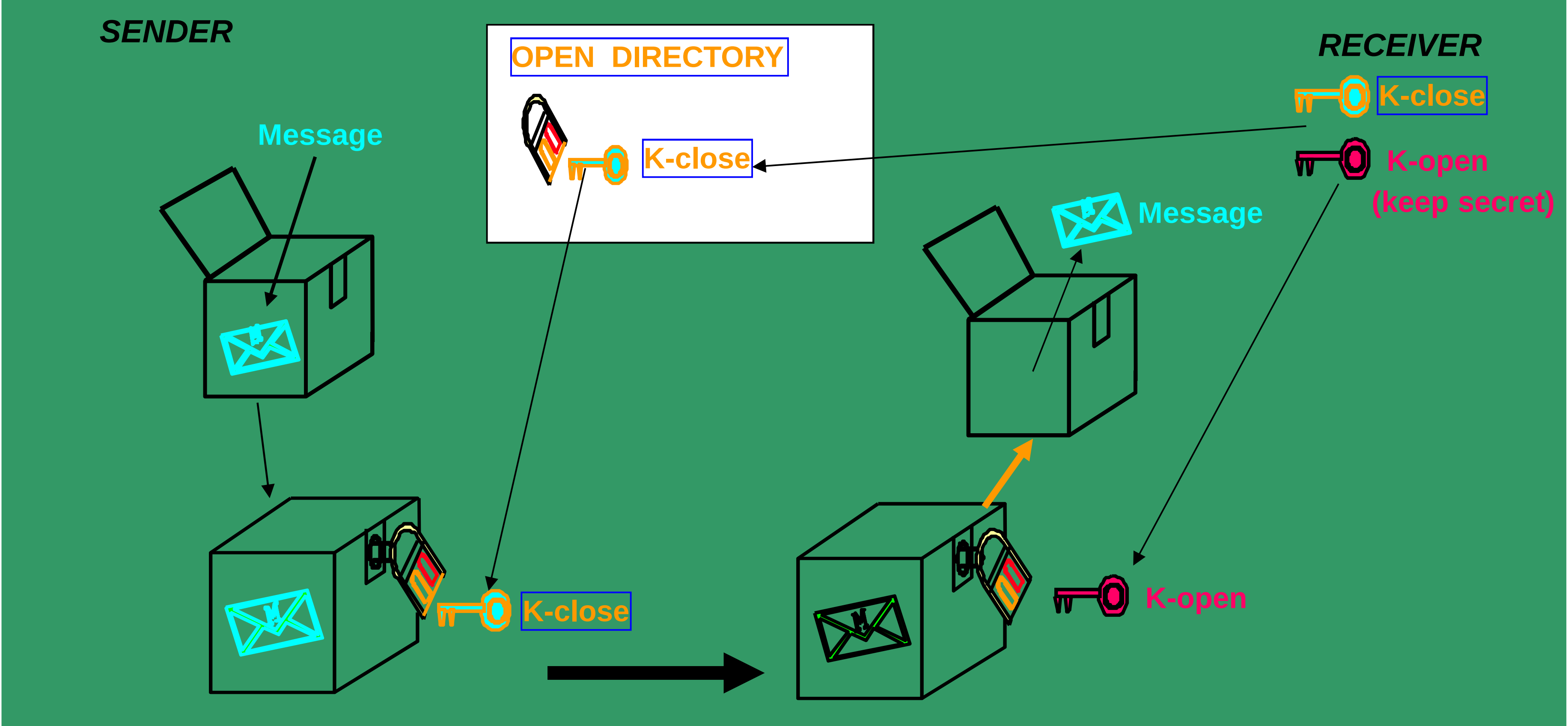
- Each user has a pair of keys which are generated together under a scheme
  - Private Key - known only to the owner
  - Public Key - known to anyone in the systems with assurance
- Encryption
  - Sender encrypts the message by the *Public Key* of the receiver
- Decryption
  - Only the receiver can decrypt the message by his *Private Key*

Asymmetric Key Technology

- Someone can know public key
- Computing private key from public key is very, very difficult (factoring huge number)

Simple Example of PKC

Non-mathematical



Attack: Eve can replace Bob’s padlock with hers on the way

Application: Secrecy

- Bob has Bob.pub, Bob.priv
- Alice has Alice.pub, Alice.priv
- Alice wants to send Bob a secret “Hello" note

Application: Secrecy

- Alice finds Bob.pub from his website
- Alice computes c = crypt(p, Bob.pub)
- Sends c to Bob over unsecured wire
- Bob computes p = crypt(c, Bob.priv)

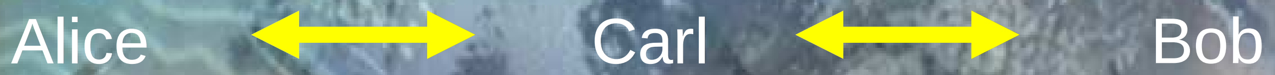
Advantages

- Key distribution not a problem!
- Anyone can send a message to Bob
- Only Bob can decrypt!

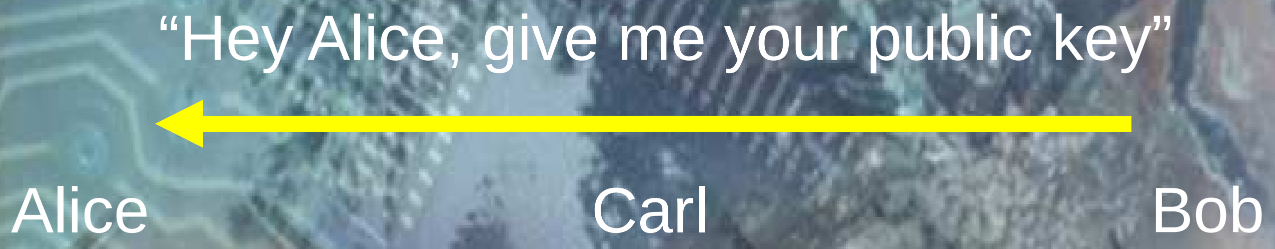
Attack: Spoofer

- One person (hacker) successfully pretends to be as another (normal user)

Man In the Middle Attack (Spoofing)

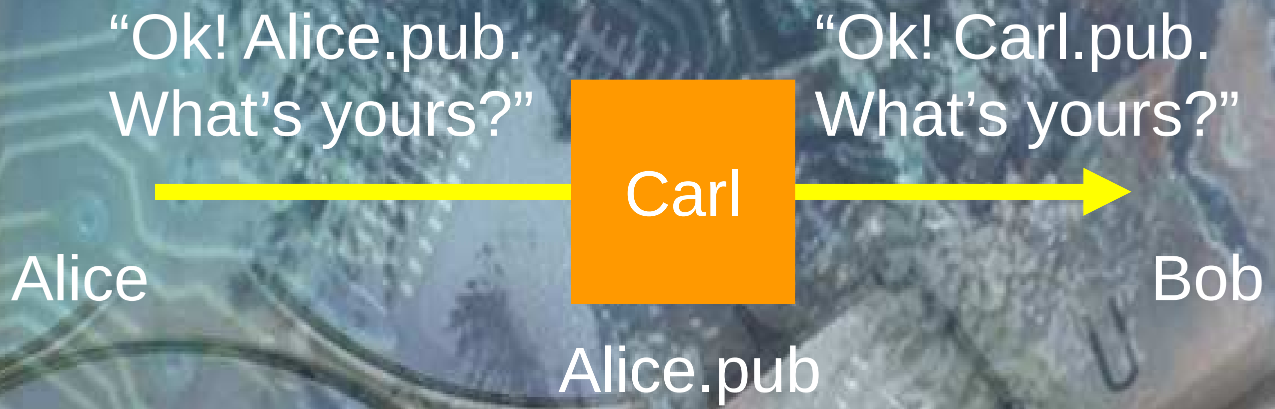


Man In the Middle Attack (Spoofing)



“Hey Alice, give me your public key”

Man In the Middle Attack (Spoofing)



"Ok! Alice.pub. What's yours?"

"Ok! Carl.pub. What's yours?"

Carl

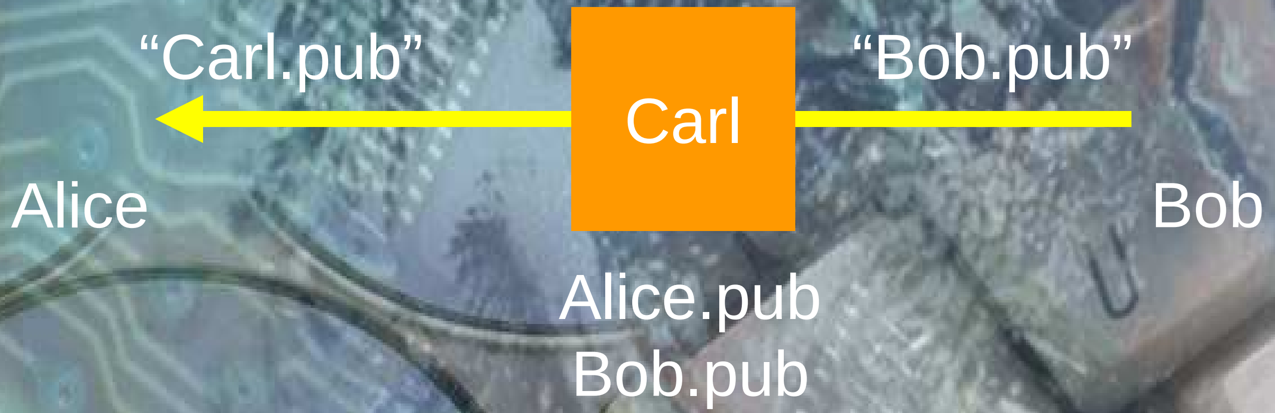
Alice

Bob

Alice.pub

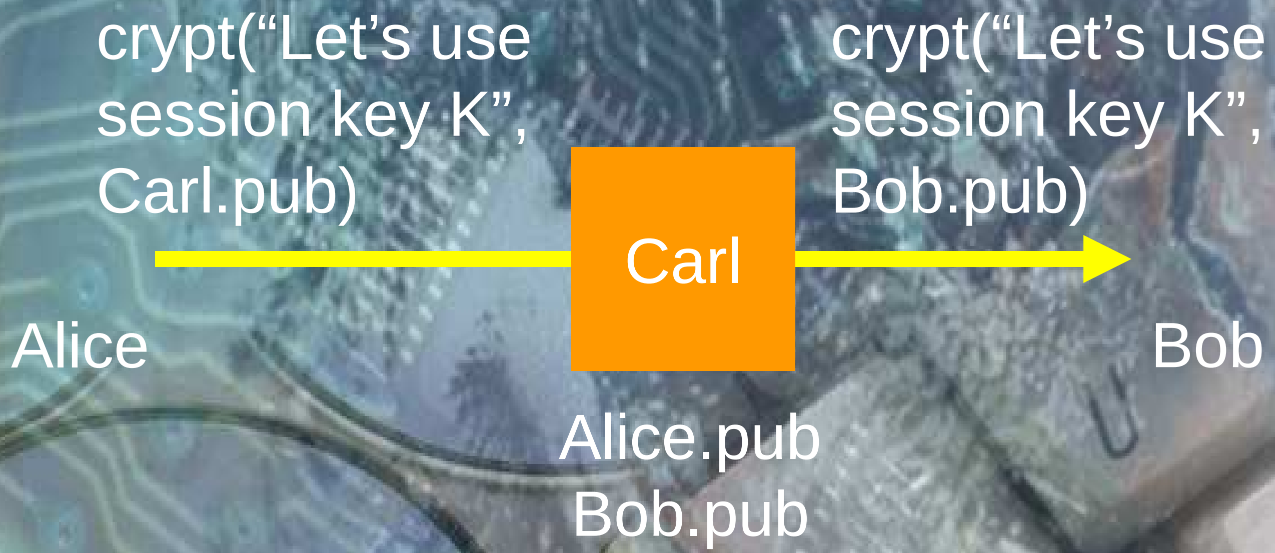
Carl takes Alice.pub and replaces is it by Carl.pub
Bob receives Carl.pub as if it is Alice.pub

Man In the Middle Attack (Spoofing)



Carl then, sends to Alice Carl.pub
Alice receives Carl.pub as if it is Bob.pub

Man In the Middle Attack (Spoofing)



crypt("Let's use session key K", Carl.pub)

crypt("Let's use session key K", Bob.pub)

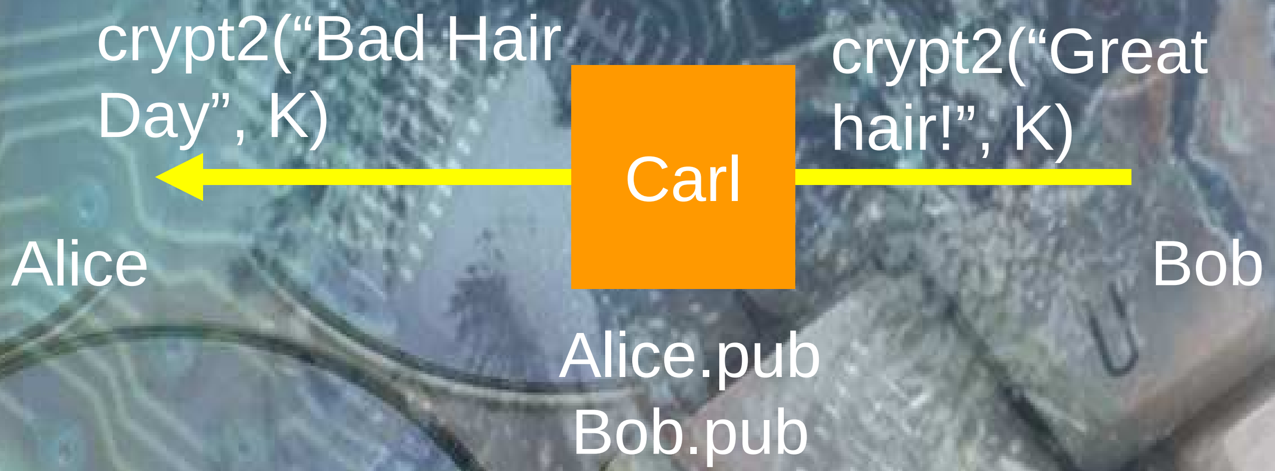
Carl

Alice

Bob

Alice.pub
Bob.pub

Man In the Middle Attack (Spoofing)



Verify Authenticity

- Through digital signatures
- And Certificate Authorities

Certificates

- Certificate Authority (CA): publishes that a particular identity goes with a particular public key
- Alice gets certificate (identity <=> public key), signed by CA
- So if you trust CA, then you can trust the public key

Application: Authenticity

- Alice wants to tell Bob the message is really from her!
- Digital signature
- Alice computes c = crypt(p, Alice.priv)
- Alice sends c over unsecured wire
- Anyone can check that Alice is the sender... by computing p = crypt(c, Alice.pub)

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv

“Hello”

Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

“Hello”
B.pub

A.pub, B.pub, ...

Carl & Eve
Bad People!

Bob
B.priv

Authenticity + Secrecy

Alice
A.priv

“Hello”

B.pub

“This is from A”

A.pub, B.pub, ...

Carl & Eve
Bad People!

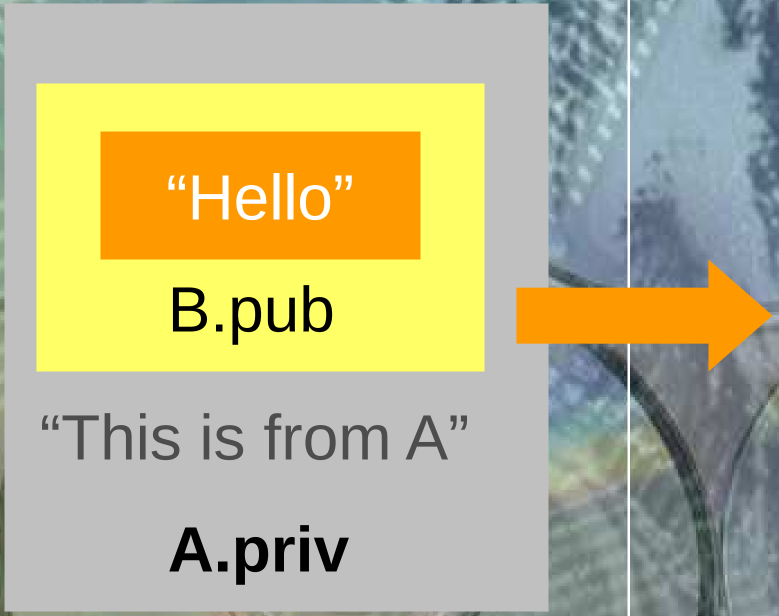
Bob
B.priv

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv



“Hello”

B.pub

“This is from A”

A.priv

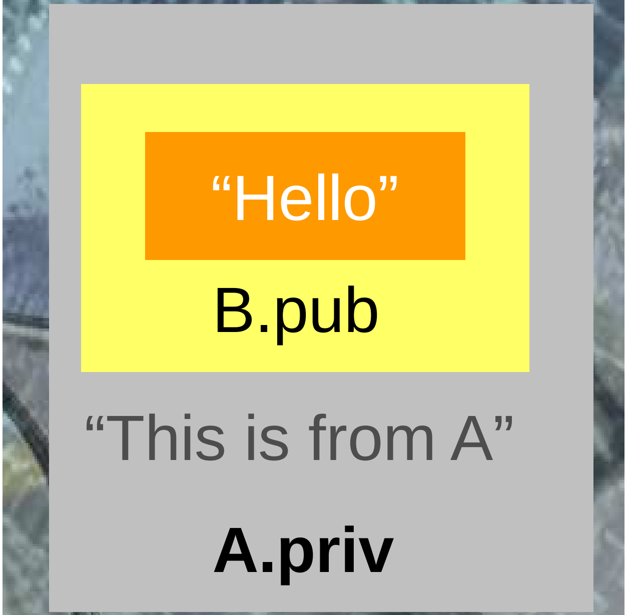
Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv



“Hello”

B.pub

“This is from A”

A.priv

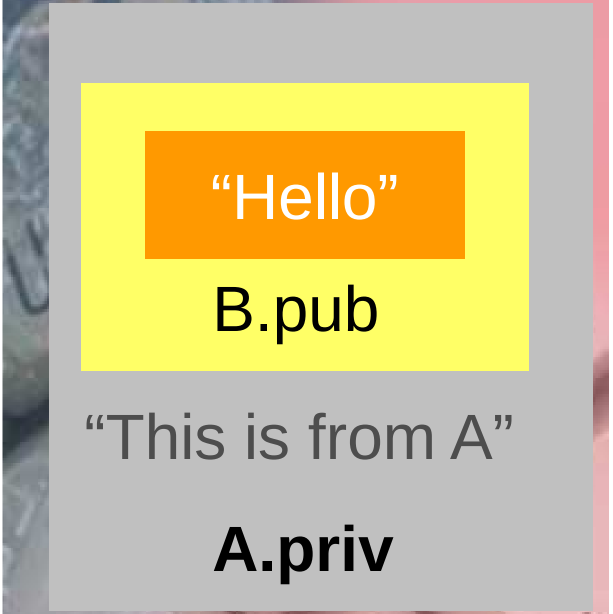
Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv



“Hello”

B.pub

“This is from A”

A.priv

Carl & Eve
Bad People!

Aspects of PKC

- Powerful tools with their own built-in problems
- Computationally demanding operations are needed
- Resource
- Implementation is always a challenge
- Much slower than the symmetric key algorithms.
- PKC should not be used for encrypting large amounts of data
- Example PKCs
  - RSA, Elliptic curve cryptosystems….

Key length

- Kerckhkoffs’s Principle:
  - the strength (security) of cryptosystems based on two important properties:
    - the quality of the algorithm
    - the key length
- The quality of cryptosystems are hard to measure
- Key length must be sufficiently large
  - to prevent adversary to determine the key by trying all possible keys in the key space, brute-force or exhaustive-search attacks
- DES key space utilizes 56-bit keys
  - key space is $2^{56} = 72,057,594,037,927,936 \approx 7.2 \text{ x } 10^{16}$

Key Length & Brute Force

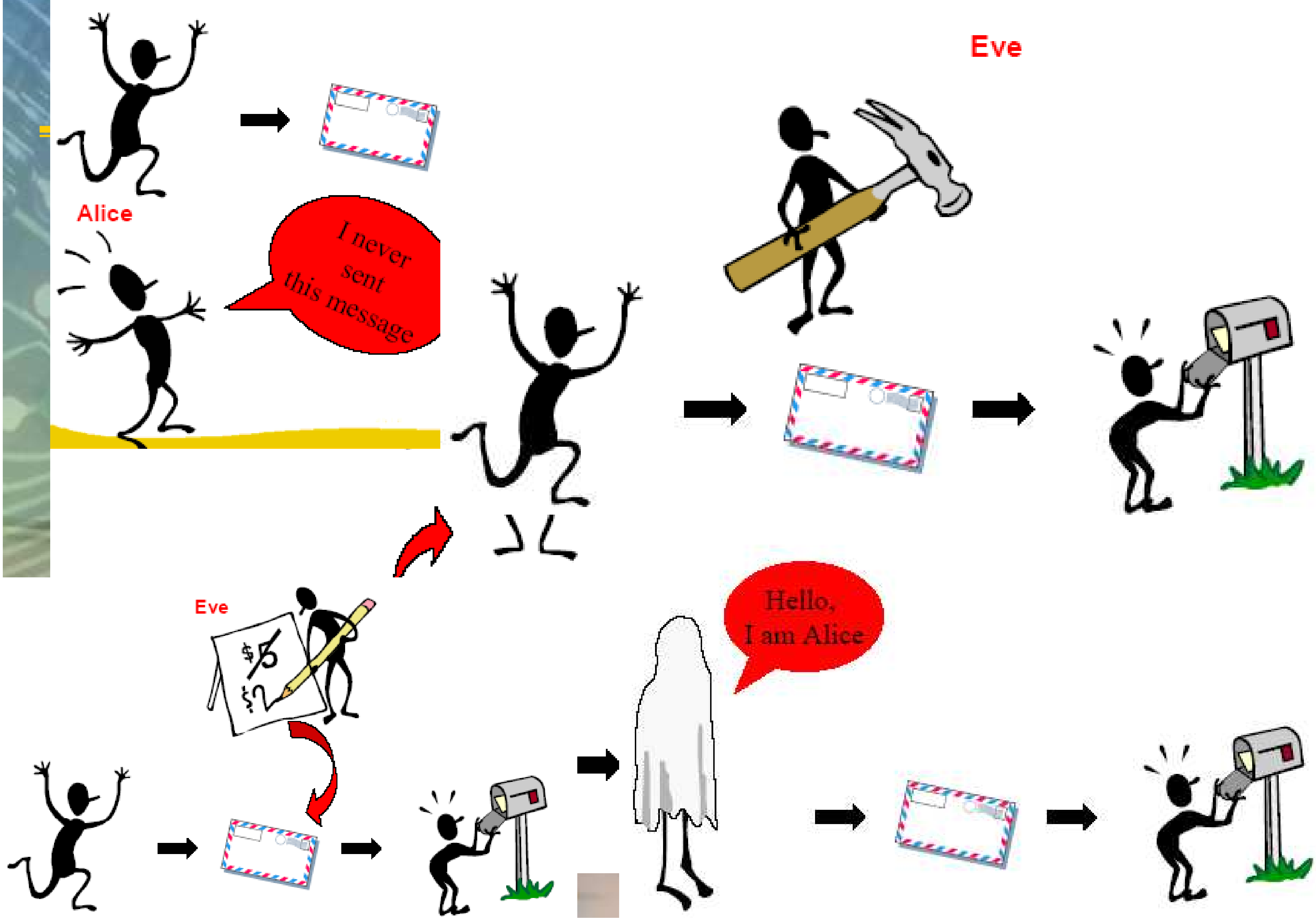
- Assume that there are 10^{30} possible key in key space
- And you can only try 10^{9} key in a second.
- There are around $3x10^{7}$ seconds in year, brute force attack would take more than $3x10^{13}$ years to try out the keys. This time period is longer than the predicted life of the universe.
- Brute force should be the last resort.
- In order to reduce the possible keys to try out one needs to take advantage
  - Weakness in cryptographic algorithm
  - Weakness in implementation of cryptographic algorithm.
- Longer keys do not necessarily improve the security

Unbreakable Cryptosystems ??

- Practical Security
  - Almost all of the practical cryptosystems are theoretically breakable given the time and computational resources
- Theoretically unbreakable system : **One-time-pad**
- One-time pad requires exchanging key that is as long as the plaintext.
- However impractical, it is still being used in certain applications which necessitate very high-level security.
- Security of one-time pad systems relies on the condition that keys are generated using truly random sources

Non-repudiati

Denial of service



Other Cryptographic Applications

- **Digital Signatures**
  - allows electronically sign (personalize) the electronic documents, messages and transactions
- **Identification**
  - is capable of replacing password-based identification methods with more powerful (secure) techniques
- **Key Establishment**
  - To communicate a key to your correspondent (or perhaps actually mutually generate it with him) whom you have never physically met before
- **Secret Sharing**
  - Distribute the parts of a secret to a group of people who can never exploit it individually

Other Cryptographic Applications

- **E-commerce**
  - carry out the secure transaction over an insecure channel like Internet
- **E-cash**
  - The cash can be sent securely through computer networks
  - The cash cannot be copied and reused
  - The spender of the cash can remain anonymous
  - The transaction can be done *offline*
  - The cash transferred to others
  - A piece of cash can be divided into smaller amounts
- **Games**
  - Flipping coins over the phone
- **Electronic Voting**

Hash Functions

- h = hash(input)
- Almost Every bit in input affects output
- Hash function not invertible

Error Checking

- Alice wants to send a LONG message to Bob
- Alice computes h=hash($LONG_MSG);
- Sends data to Bob, includes relatively short h at the end of message
- Bob recomputes hash.
- If match, great! Data's correct!
- If not match, either hash or data was corrupted. Resend.

Digital Signatures

- Bob wants to send $data to Alice, with assurances of his identity (authenticity)
  - h=hash($data)
  - Signature = crypt(h, Bob.priv)
- Sends these to Alice
- Alice confirms Bob's identity by
  - h = crypt(signature, Bob.pub)
  - h = hash($data)
  - Compares!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv

“Hello
……”

Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv

“Hello
……”

hash(" Hello...") →
12fea90897bddc

Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

"Hello
……"

"This is from A"

12fea90897bddc
A.priv

A.pub, B.pub, ...

Carl & Eve
Bad People!

Bob
B.priv

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv

“Hello
……”

“This is from A”

12fea90897bddc
A.priv

Bob.pub

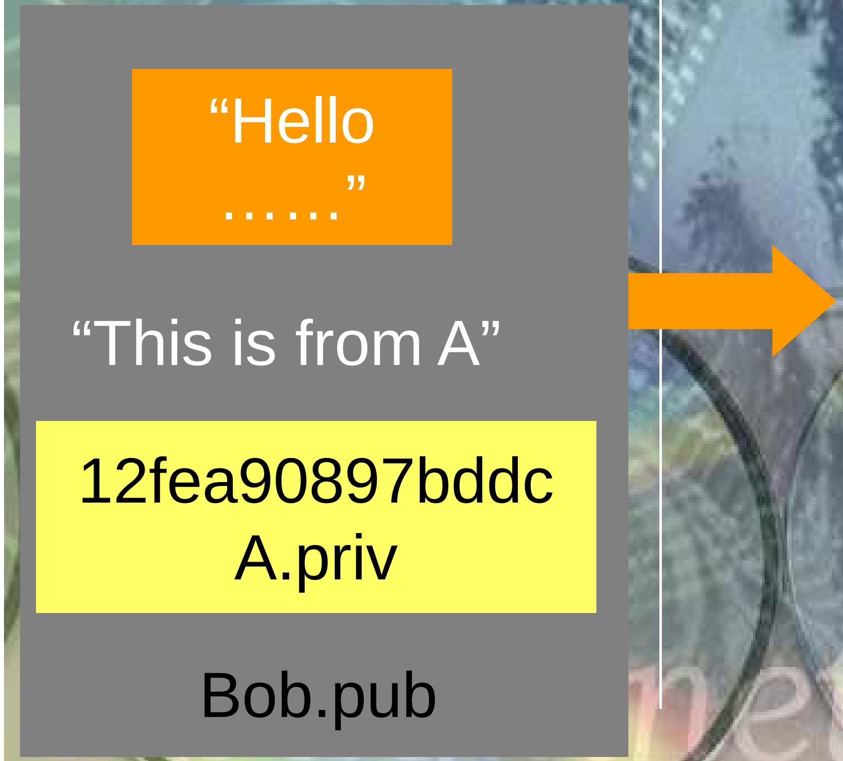
Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv



“Hello
……”

“This is from A”

12fea90897bddc
A.priv

Bob.pub

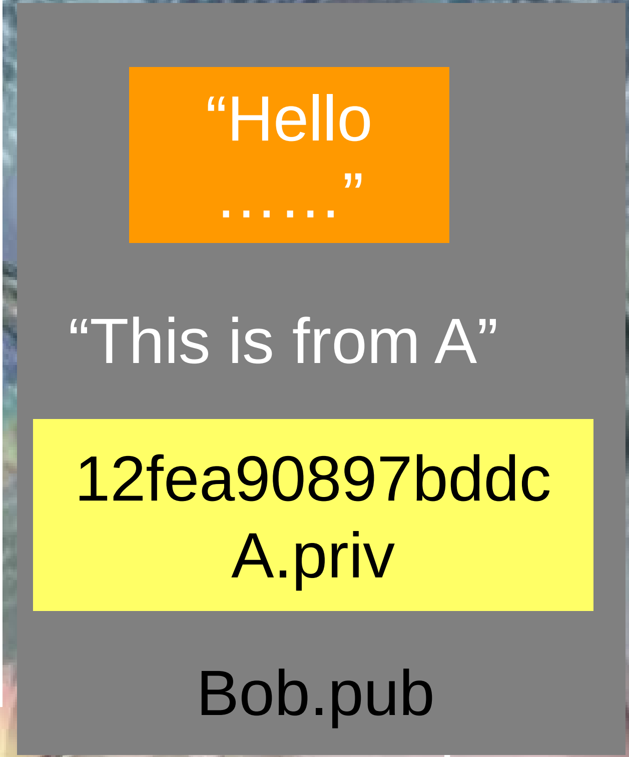
Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv



“Hello
……”

“This is from A”

12fea90897bddc
A.priv

Bob.pub

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Bob
B.priv

"Hello
……"

"This is from A"

12fea90897bddc
A.priv

Bob.pub

Carl & Eve
Bad People!

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Carl & Eve
Bad People!

Bob
B.priv

"Hello
……"

"This is from A"

12fea90897bddc
A.priv

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Carl & Eve
Bad People!

Bob
B.priv

“Hello
……”

“This is from A”

12fea90897bddc
A.priv

Authenticity + Secrecy

Alice
A.priv

A.pub, B.pub, ...

Carl & Eve
Bad People!

Bob
B.priv

“Hello
……”

“This is from A”

12fea90897bddc
==
hash(“Hello ...") → 12fea90897bddc

Secure Sockets Layer (SSL)

- Developed by Netscape
- Uses Asymmetric cryptographic system
- Many Web sites support the protocol to obtain confidential user information, such as credit card numbers.
- SSL - URLs start with https: instead of http:.
- S-HTTP : Another protocol for transmitting data securely over the World Wide Web.
- Whereas SSL creates a secure connection between a client and a server, over which any amount of data can be sent securely,
- S-HTTP is designed to transmit individual messages securely.
- SSL and S-HTTP can be seen as complementary rather than competing technologies.
- Both protocols have been approved by the Internet Engineering Task Force (IETF) as a standard.

Symmetric vs. Asymmetric

- Symmetric faster but relies on shared secret
- Asymmetric slower but “solves” distribution-of-keys problem

Security Risk

- If you write it, they will come... to attack it. :o)
- Be aware of most common attacks...
- Learn the basic tricks to writing safer code.

Terminology

- Vulnerability -- some buggy code that can allow bad guys to compromise your machine, or do other bad guy things

- Exploit -- some code or method to take advantage of the vulnerability

Attack: Social Engineering

- Tricking a naïve person into revealing sensitive data (i.e. his password)
  - Hi this is your bank. We need your PIN to fix your account ASAP!
  - Hi this is Amazon. Your order #2333 didn't go through because your credit card was rejected. Tell us another credit card's info, and your order will be good.

Bottom Line

- People are the weakest link
- Educate people about computer/Internet Security

Attack: Traffic Sniffing

- Looking at packets on the wire, reading off passwords, etc...
- Problem for authentication mechanisms with cleartext passwords

Traffic Sniffing

- (Somehow) compromise a machine. This is the hard part.
- Set ethernet "promiscuous" mode
- Install a root kit
  - hides hacker activity
  - key logger
  - packet sniffer
  - recompiled versions of programs (passwd)

Password Guessing

- one of the most common attacks
- attacker knows a login (from email/web page etc)
- then attempts to guess password for it
  - try default passwords shipped with systems
  - try all short passwords
  - then try by searching dictionaries of common words
  - intelligent searches try passwords associated with the user (variations on names, birthday, phone, common words/interests)
  - before exhaustively searching all possible passwords
- check by login attempt or against stolen password file
- success depends on password chosen by user
- surveys show many users choose poorly

Password Guessing

- How long is YOUR password?
- Ways to break
  - Dictionary attack (words, names, dates)
  - Brute force
- Solutions
  - Freeze/Turn off account if too many incorrect logins?
  - Wait 2 seconds before logging in/displaying error.

Password Capture

- another attack involves **password capture**
  - watching over shoulder as password is entered
  - using a trojan horse program to collect
  - monitoring an insecure network login (eg. telnet, FTP, web, email)
  - extracting recorded info after successful login (web history/cache, last number dialled etc)
- using valid login/password can impersonate user
- users need to be educated to use suitable precautions/countermeasures

Passwords

- What if your website froze accounts if too many incorrect logins?
- Hacker can still attack your sites users!
- By purposefully guessing login/passwords incorrectly, so that your system locks all accounts!

- Denial of Service

Solutions

- Longer passwords
- Other forms of authentication
  - Biometric
  - Physical key/card based

External Executables

- Don't trust other people's code
- If Carl can run code on Alice's computer... then Carl can take it over

Attack: Buffer Overflow

- Bad guy sends a huge, over-sized request to a naïvely implemented program, overflowing the input buffer
- May overwrite data in memory (and/or) program code
- May overwrite the return address on the stack of a program in C, so that the procedure call returns somewhere else

How To Avoid Buffer Overflow

- Write code carefully
- Limit input size; read in small chunks as opposed to reading in whole input
- Use better languages (read: Java)

Attack: Computer Virus

- Attaches itself to a host, another computer program
- Tries to infect other executable files it finds
- When run, it damages resources, files, etc...

Viruses

- A virus is a small program that inserts itself into other executable software.
- Every time that software is opened and used, the virus program will run, making copies of itself to insert into every document and executable file opened.
- This can cause damage to your computer software, including your operating system, by corrupting existing data on all your storage media and overwriting your files.
- As long as a virus program is present in any software you open, it can spread to other computers when you share files and programs with others — over the Internet using e-mail or P2P (peer-to-peer) file-sharing networks, or via infected CDs, DVDs, or floppy disks.
- Viruses persist primarily in stored memory on physical media such as your hard drive.
- New viruses are not as common a threat now as in the past.

Worms

- Self replicating/Spreading computer program.
- Worms are programs that can copy themselves; they exist in RAM (random- access memory).
- They spread by sending themselves via e-mail, instant- message programs, and peer-to-peer (P2P) file-sharing networks to other computers in a network.
- Unlike viruses, worms do not insert themselves into other programs — and they rarely affect the files on your hard drive.
- Worms cripple computers by congesting the flow of information, slowing down the system by using up its resources, or crashing the system altogether — all by making multiple copies of themselves.
- Unpatched computers, — those with- out the software fixes that plug security holes, — are a bonanza for them.
- Worms have shut down large portions of the Internet, causing millions of dollars in damages before they were stopped.
- They can also be carriers ofroot-kits, backdoors, and trojans (which we describe next).

Example

- Morris Worm -- buffer overflow attack on UNIX finger and other programs...
- Robert Tappan Morris, Jr. (CMU student) launched it on Nov 2, 1988 from an MIT computer
- Intended to just spread, but a _bug_ in his code infected computers multiple times, so that computers FROZE after a while
- Infected 6000 UNIX workstations
- CERT created in response to Morris
- Morris now a MIT faculty member

Worms and their Payloads

- Infect computer; send emails to other people… to spread the worm
- Infect computer; install a backdoor program to let bad guy log in… to send mass spam, send more worms, etc

Blaster Worm

- Exploited a buffer overflow in Windows's RPC service
- Programmed to SYN flood windowsupdate.com on August 15 to prevent patches

Attack: Trojan Horse

- Greek allusion
- Innocent looking program, does something malicious

Trojans

- Trojan Horse programs (now mostly referred to as just trojans) are malicious applications masquerading as something helpful or innocuous.
- Veritable “wolves in sheep’s clothing,” they can disguise a destructive program as something more benign, such as an image file.
- A harmless-looking .gif extension, for example, may hide the .exe extension of an executable file.

Dialers

- Two kinds of dialers exist — one good, one bad.
- The good one is installed as part of your operating system; it helps you connect to the Internet via an analog dialup connection.
- The other is malware, used to set up a fraudulent connection (usually to an expensive, long-distance telephone number) or to force downloads — all of which gets charged to your telephone bill — through particular Web sites.
- Malware dialers can be installed by trojans, ActiveX and JavaScript scripts, and from opening attachments in spam e-mails.
- (Users of DSL or Broadband connections are usually not affected by dialers.)

Backdoors

- Backdoors are programs (or modifications to existing programs) that give outside users remote access to your computer without requiring user identification.
- Backdoors attempt to remain hidden or to “hide in plain sight” by appearing to be innocent.
- They can also be special passwords set up on a login system to the same effect.
- Backdoors can be installed through weaknesses in an unpatched or unpro- tected Windows computer, either directly by blackhat hackers or with a trojan, virus, or worm.

Spyware

- is software that tracks user behavior and reports it to a company.
- Currently considered to be one of the greatest threats to Internet and computer security today
- includes a wide range of applications that use stealth and trickery to fool users into installing them.
- Broadly speaking, spyware takes full or partial control of computer operations while denying your rights to privacy and to choose for yourself what runs on your computer — all for the benefit of strangers.

Adware

- is software that displays advertisements.
- is often free.
- shows advertisements of various products.
- Adware programs are often associated with spyware
  - because many adware programs monitor your browsing habits to target you with specific advertisements.

Rootkits

- A rootkit is a program designed to hide not only itself, but another program and all its associated resources (processes, files, folders, Registry keys, ports, and drivers).
- Rootkits can be whitehat (well-intentioned in purpose but still a potential security risk) or blackhat (malicious in nature).
- Malicious rootkits are often used to compromise and maintain remote control over a computer or network for illegitimate, — often criminal — purposes.
- Malicious rootkits do their work by hiding malware that installs a backdoor to allow an attacker to have unlimited and prolonged access to the infected computer.
- There also other types of malware such as exploits, macros, botnets, hijackers and keyloggers.