CHAPTER 1

Foundations

Terminology

Messages and Encryption

- Plaintext: A message in normal form is Plaintext/Cleartext.
- Ciphertext: A message in encrypted form is Ciphertext.
- Encryption: The process of disguising a message in such a way as to hide its substance is encryption/Encipher (This is ISO term).
- Decryption: The process of turning ciphertext back into plaintext is decryption/Decipher (ISO term).
- Cryptography: The art and science of keeping messages secure is cryptography.
- Cryptographers: The practitioners of cryptography are cryptographers.

Terminology

Messages and Encryption

- Cryptanalysis: The art and science of breaking ciphertext ; that is seeing through the disguise.
- Cryptanalysts: The practitioners of cryptanalysis are cryptanalysts.
- Cryptology: The branch of mathematics encompassing both cryptography and cryptanalysis is cryptology.
- Cryptologists: The practitioners of cryptology are cryptologists.

Terminology

Messages and Encryption

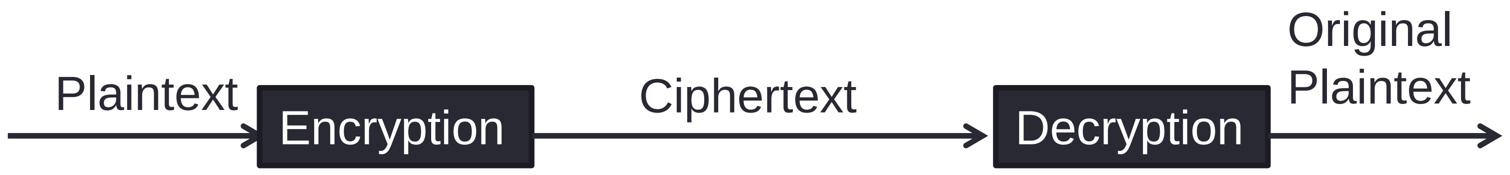
- The plaintext is denoted by M (message) or P (plaintext).
- It can be a stream of bits, text file, bitmap file, a stream of digitized voice, a digital video image etc.
- Ciphertext is denoted by C. The size may be as plaintext or sometimes it can be larger or smaller.
- In mathematical notation:

$E(M)=C$	where E is encryption function.
$D(C)=M$	where D is decryption function.

- In any case $D(E(M))=M$ is true.

Terminology

Authentication, Integrity and Nonrepudiation



Encryption and Decryption

In addition to providing confidentiality, cryptography is often asked to do the following:

1. Authentication: The receiver of a message should ascertain its origin.
2. Integrity: The receiver should be able to verify that it has not been modified in transit.
3. Nonrepudiation: A sender should not be able to falsely deny later that he sent a message.

Terminology

Algorithms and Keys

- A cryptographic algorithm (Cipher) is the mathematical function used for Encryption and Decryption.

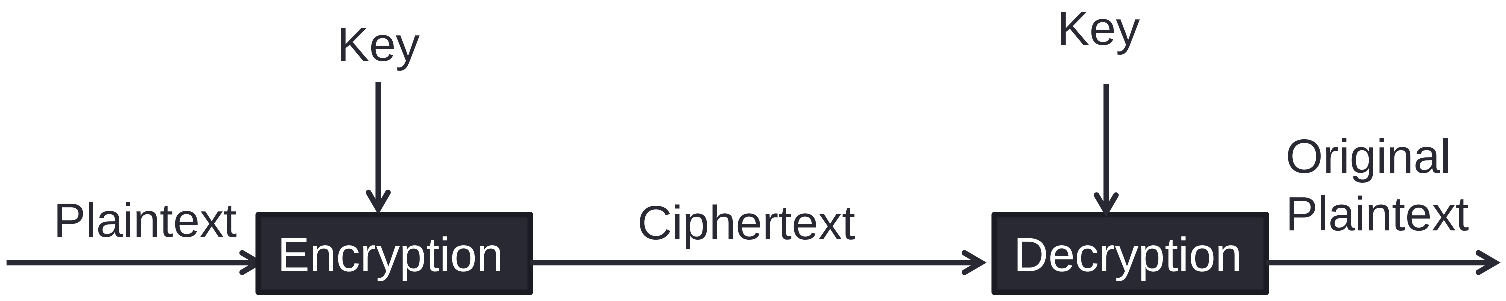
- Restricted Algorithm: If the security of an algorithm is based on keeping the way that algorithm works a secret.
- Drawbacks of Restricted Algorithm:
  1. A large or changing group of users can not use them, because every time a user leaves the group everyone else must switch to a different algorithm.
  2. If someone accidentally reveals the secret, everyone must change their algorithm.
  3. Allow no quality control or standardization.
  4. Every group of users must have their unique algorithm.
  5. Group of users must write their own algorithms and implementations.
  6. If no one in the group is a good cryptographer, then they won’t know if they have a secure algorithm.

Terminology

Algorithms and Keys

- Modern cryptography solves this problem with a key, denoted by k.
- Keyspace: The range of possible values of the key.
- Both encryption and decryption operations use this key.
- The encryption and decryption functions become:

$E_k(M)=C$, $D_k(C)=M$ and $D_k(E_k(M))=M$.



Encryption and Decryption with a Key

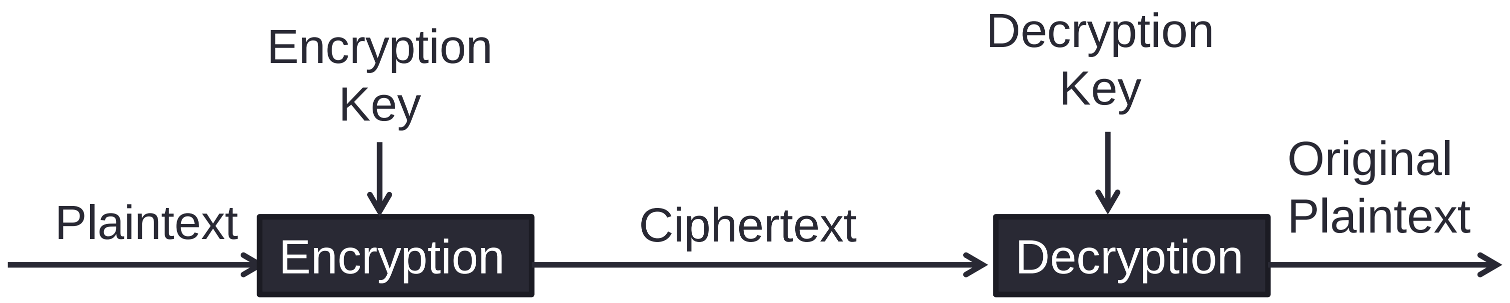
Terminology

Algorithms and Keys

- Some algorithms use a different encryption key and decryption key. In this case:

  $E_{k1}(M)=C$, $D_{k2}(C)=M$ and $D_{k2}(E_{k1}(M))=M$.

- All of the security in these algorithms is based in the key (keys); none is based in the details of the algorithm.



Encryption and Decryption with a Key

Symmetric Algorithm

- Two General Types of key based algorithm:
  - Symmetric Algorithm.
  - Public-key Algorithm.
- In Symmetric key Algorithm encryption key can be calculated from the decryption key and vice versa.

- These algorithms also known as secret key/ single key/ one key or conventional algorithm.

- The sender and the receiver must agree on a key before they can start communication.
- Encryption and Decryption with a symmetric algorithm are denoted by: $E_k(M)=C$, $D_k(C)=M$.

Symmetric Algorithm

- Symmetric algorithms can be divided into two categories.
    - Stream algorithms or Stream Cipher
    - Block algorithms or Block Cipher.
- Stream Cipher operates on the plaintext a single bit (sometimes bytes) at a time.
- Block Cipher operates on the plaintext in group of bits (called blocks) at a time.
- A typical block size is 64 bits.

Public-key Algorithm

- Public-key algorithms also known as Asymmetric algorithm.
- The key used for encryption is different from the key used for decryption.

- These are called public-key algorithm because the encryption key can be made public.

- A complete stranger can use the encryption key to encrypt a message, but only a specific person with the corresponding decryption key can decrypt the message.

- The encryption key is often called the public key.
- The decryption key is called the private key or secret key.

Cryptanalysis

- The whole point of Cryptography is to keep the plaintext (or the key, or both) secret from eavesdroppers.
- Eavesdroppers also known as Adversaries, Attackers, Interceptors, Interlopers, Intruders, Opponents or Enemy.

- Attack: An attempted cryptanalysis is called an attack.
- There are four general types of cryptanalytic attacks.

- In each of these attacks however it is assumes that the cryptanalyst has complete knowledge of the encryption algorithm used.

Cryptanalysis

1. Ciphertext-only attack: In this case the cryptanalyst has several ciphertext message all of which have been encrypted using the same key. The job is to deduce the key (or keys) used to encrypt or an algorithm to decrypt all new messages encrypted with the same key.

   Given: $C_1=E_k(P_1)$, $C_2=E_k(P_2)$, … $C_i=E_k(P_i)$

   Deduce: Either $P_{1,}$ $P_{2,}$ … $P_{i;}$ K, or an algorithm to infer $P_{i+1,}$ from $C_{i+1}=E_k(P_{i+1})$

2. Known-plaintext attack: In this case the cryptanalyst has several ciphertext and their corresponding plaintext. The job is to deduce the key (keys) used to encrypt or an algorithm to decrypt any new message encrypted with the same key.

   Given: $P_{1,}$ $C_1=E_k(P_1)$, $P_{2,}$ $C_2=E_k(P_2)$, … $P_{i,}$ $C_i=E_k(P_i)$

   Deduce: Either K, or an algorithm to infer $P_{i+1,}$ from $C_{i+1}=E_k(P_{i+1})$

Cryptanalysis

3. Chosen-plaintext attack: The cryptanalyst has ciphertext and the associated plaintext. He also chooses the plaintext that gets encrypted. The job is to deduce the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages.

   Given: $P_{1,}$ $C_1=E_k(P_1)$, $P_{2,}$ $C_2=E_k(P_2)$, … $P_{i,}$ $C_i=E_k(P_i)$ where the cryptanalyst gets to choose P_1, $P_{2, \ldots}$ $P_{i,}$

   Deduce: Either K, or an algorithm to infer $P_{i+1,}$ from $C_{i+1}=E_k(P_{i+1})$

4. Adaptive-chosen-plaintext attack: This is a special case of a chosen plaintext attack. In this case the cryptanalyst can choose the plaintext that is encrypted at the same time he can also modify his choice based on the previous result of encryption.

Security of an Algorithm

- Different algorithm have different degree of security.
- If the cost required to break an algorithm is greater than the value of the encrypted data, then you are probably safe.
- If the time required to break an algorithm is greater than the time the encrypted data must remain secret, then you are probably safe.
- If the amount of data encrypted with a single key is less than the amount of data necessary to break the algorithm, then you are probably safe.
- In fact one-time pad is unbreakable given infinite resources.
- All other cryptosystems are breakable in a ciphertext-only attack, simply by trying every possible key one by one, this is called Brute-force attack.

Other Cryptanalytic Attack

There are at least three other types of cryptanalytic attacks.

5. Chosen-ciphertext attack: The cryptanalyst can choose different cipher text to be decrypted and has access to the decrypted plaintext. The job is to deduce the key.

   Given: $C_{1,}$ $P_1=D_k(C_1)$, $C_{2,}$ $P_2=D_k(C_2)$, …$C_{i,}$ $P_i=D_k(C_i)$

   Deduce: K

   These are applicable to public-key algorithms. Chosen-plaintext and Chosen-ciphertext attack are Chosen-text attack.

6. Chosen-key attack: It means that the cryptanalyst has some knowledge about the relationship between different keys.

7. Rubber-hose cryptanalysis: The cryptanalyst threatens, blackmails, or tortures someone until they give him the key. It is referred to as purchase-key attack.

Substitution Cipher and Transposition Cipher

- In Substitution Cipher, each character in the plaintext is substituted for another character in the Ciphertext. The receiver do inverse operation.
- Four types of Substitution Ciphers are as follows;

1. Simple Substitution or Monoalphabetic Cipher: Each character of the plaintext is replaced with a corresponding character of Ciphertext.

2. Homophonic Substitution Cipher: Like simple substitution cipher, except a single character of plaintext can map to one of several characters of ciphertext. For example “A” could correspond to either 5, 13, 27 or 57, “B” could correspond to either 7, 20, 29 or 51, and so on.

Substitution Cipher

3. Polygram Substitution Cipher: Blocks of characters are encrypted in groups. For example, “ABA” could correspond to “XYZ”, “CDD” could correspond to “SLA” and so on.

4. Polyalphabetic Substitution Cipher: is made up of multiple simple substitution ciphers. For example, there might be five different substitution ciphers used; the particular one used changes with the position of each character of the plaintext.

The famous Caesar Cipher: each character is replaced by the character three to the right modulo 26. For example, “A” is replaced by “D”, “B” is replaced by “E”, …”X” is replaced by “A”, “Y” is replaced by “B” is a simple substitution cipher.

Transposition Cipher

In Transposition Cipher the plaintext remains the same, but the order of characters is shuffle around.

- In a simple columnar transposition cipher, the plaintext is written horizontally onto a piece of graph paper of fixed width and the ciphertext is read vertically.
- For decryption, the opposite procedure is applied with same width (i.e. vertically then horizontally).
- The German ADFGVX cipher: Used during world war I, is a transposition cipher combined with a simple substitution cipher.
- It was very complex algorithm for its day but was broken by Georges Painvin, a French Cryptanalyst.

Transposition Cipher Example

Plaintext: DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

D	E	P	A	R	T	M	E	N	T
O	F	C	O	M	P	U	T	E	R
S	C	I	E	N	C	E	A	N	D
E	N	G	I	N	E	E	R	I	N
G									

Ciphertext: DOSEGEFCNPCIGAOEIRMNNTPCEMUEEETARNENITRDN

Double Transposition Cipher

In double Transposition Cipher, the ciphertext (from the plaintext) acts as a plaintext and the corresponding ciphertext is the double transposition ciphertext.

Plaintext: DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERING

Ciphertext: DOSEGEFCNPCIGAOEIRMNNTPCEMUEEETARNENITRDN

D	O	S	E	G	E	F	C	N	P
C	I	G	A	O	E	I	R	M	N
N	T	P	C	E	M	U	E	E	E
T	A	R	N	E	N	I	T	R	D
N									

Double Ciphertext: DCNTNOITASGPREACNGOEEEEMNFIUICRETNMERPNED

ONE-TIME PADS

- One-time pad was invented by Major Joseph Mauborgne and AT&T's Gilbert Vernam in 1917.
- It is a perfect encryption scheme.
- A one-time pad is nothing more than a large nonrepeating set of truly random key letters, written on sheets of paper, and glued together in a pad.
- The sender uses each key letter on the pad to encrypt exactly one plaintext character.
- Encryption is the modulo 26 of the plaintext character and the one-time pad key character.
- The sender encrypts the message and then destroys the used pages of the pad or used section of the tape.

ONE-TIME PADS

- The receiver has an identical pad and uses each key on the pad in turn, to decrypt each letter of the ciphertext.
- The receiver destroys the same pad pages or tape section after decrypting the message.
- Example: Message: ONETIMEPAD

and the key sequence from the pad is

TBFRGFARFM

Then the ciphertext is IPKLPSFHGO

Because O+T modulo 26=I

N+B modulo 26=P

E+F modulo 26=K etc.

Computer Algorithms

- Many Cryptographic Algorithms. These are three of the most common.

1. DES (Data Encryption Standard): is US and international standard symmetric algorithm (same key is used for encryption and decryption).

2. RSA (Rivest, Shamir and Adleman): is the most popular public-key algorithm, used for both encryption and digital signature.

3. DSA (Digital Signature Algorithm): is another public-key algorithm. It can not be used for encryption, but only for digital signatures.

CHAPTER 2

Introduction to Protocols

Introduction to Protocols

- Protocol: A protocol is a series of steps, involving two or more parties, designed to accomplish a task.
- A cryptographic protocol is a protocol that uses cryptography.
- Every steps must be executed in turn, and no step can be taken before the previous step is finished.

- Protocols have some others characteristics as well:

1. Everyone involved in the protocols must know the protocol and all of the steps to follow in advance.
2. Everyone must agree to follow it.
3. The protocol must be unambiguous, each step must be well defined and there must be no chance of a misunderstanding.
4. The protocol must be complete, there must be a specified action for every possible situation.

Communications Using Symmetric Cryptography

Let Alice is sender and Bob is receiver. Then

1. Alice and Bob agree on a cryptosystem.
2. They also agree on a key.
3. Alice encrypts her plaintext to ciphertext using the encryption algorithm and the key.
4. Alice sends the ciphertext message to Bob.
5. Bob decrypts the ciphertext message with the same algorithm and key and reads it.

Communications Using Symmetric Cryptography Continue…

- However Eve, sitting between them can get the ciphertext. If the algorithm used is strong enough then it is difficult for Eve to get plaintext.
- If Eve has the ability to capture both the algorithm and the key then it is decrypted easily.
- In a good cryptosystem, Alice and Bob could perform step 1 in public and step 2 in secret.
- The key must remain secret before, during and after the protocol- as long as the message must remain secret.

Communications Using Symmetric Cryptography Continue…

Mallory an active attacker could do the following things.

1. He could break the communication path so that Alice and Bob could not talk.

2. If Mallory knew the key, he could also intercept Alice’s message and substitute his own.

3. If Mallory didn’t know the key, he could only create a replacement message that would decrypt to gibberish.

Communications Using Symmetric Cryptography Continue…

In Summary the Symmetric Cryptosystem have the following problems:

1. Key must be distributed in secret. For encryption system that span the world, this can be a daunting task.
2. If the key is compromised(Stolen, guessed, etc.), then Eve can decrypt all messages. He could also produce false message to fool the other party.
3. If a separate key is used for each pair of users, the total no of key increases as the no of users increases. For n users it requires n(n-1)/2 keys. For 10 users it is 45 keys; for 100 it is 4950.

One Way Functions

- The notion of one-way function is central to public key cryptography.
- One-way functions are relatively easy to compute but significantly difficult to reverse. i.e. given x, it is easy to calculate f(x), but given f(x), it is hard to compute x. Example: Breaking a plate.
- Trapdoor one-way function: is a special type of one way function, one with a secret trapdoor. i.e. there is some secret information y, such that given f(x) and y it is easy to compute x.

One-Way Hash Functions

- A hash function is a function, mathematical or otherwise, that takes a variable length input string(pre-image) and converts it to a fixed length output string(hash value).

  A simple hash function would be a function that takes pre-image and returns a byte consisting of the XOR of all the input bytes.

- A one-way hash function is a hash function that works in one direction: It is easy to compute a hash value from pre-image, but it is hard to generate a pre-image that hashes to a particular value.

  It is also known as compression function, contraction function, message digest, fingerprint, cryptographic checksum, message integrity check(MIC) and manipulation detection code(MDC).

Communications using public key cryptography

The steps to send message from Alice to Bob are as follows.

1. Alice and Bob agree on a public-key cryptosystem.
2. Bob sends Alice his public key.
3. Alice encrypts her message using Bob’s public key and sends it to Bob.
4. Bob decrypts Alice’s message using his private key.

The public keys of all the users are published in a database somewhere. Another simple protocol can be as follows:

1. Alice gets Bob’s public key from the database.
2. Alice encrypts her message using Bob’s public key and sends it to Bob.
3. Bob then decrypts Alice’s message using his private key.

Hybrid Cryptosystems

In real world, public-key algorithms are not used to encrypt messages; they are used to encrypt keys. The reasons are as follows:

1. Public key are slow. Symmetric key algorithms are generally at least 1000 times faster than public-key algorithms.

2. Public key cryptosystems are vulnerable to chosen-plaintext attacks. If C=E(P), when P is a plaintext out of a set of n possible plaintexts, then a cryptanalyst only has to encrypt all n possible plaintexts and compare the results with C.

Hybrid Cryptosystems Continue…

In most cases public-key cryptography is used to secure and distribute session keys; those session keys are used with symmetric algorithms to secure message traffic. This is sometimes called a hybrid cryptosystem.

1. Bob sends Alice his public key.

2. Alice generates a random session key k, encrypts it using Bob’s public key, and sends it to Bob. $E_B(K)$.

3. Bob decrypts Alice’s message using his private key to recover the session key. $D_B(E_B(K))=K$

4. Both of them encrypt their communications using the same session key.

Digital Signature

Like handwritten signature, digital signature is not just a graphical image file of a written signature. Because:

1. It is easy to cut and paste a valid signature from one document to another document.

2. Computer files are easy to modify after they are signed, without leaving any evidence of modification.

So digital signatures are more than just a digital version of handwritten signature.

Signing Documents with Symmetric Cryptosystems and an Arbitrator.

Alice wants to sign a message and send to Bob. Trent is a powerful, trusted arbitrator. He shares a secret key K_A with Alice and another key K_B with Bob. These key have been established long before the protocol begins and can be reused multiple times for multiple signings.

1. Alice encrypts her message to Bob with K_A and sends it to Trent.
2. Trent decrypts the message with K_A.
3. Trent takes the decrypted message and a statement that he has received this message from Alice, and encrypts the whole bundle with K_B.
4. Trent sends the encrypted bundle to Bob.
5. Bob decrypts the bundle with K_B. He can now read both the message and Trent's certification that Alice sent it.

Signing Documents with Public-key Cryptography

The basic protocol is simple:

1. Alice encrypts the document with her private key, thereby signing the document.
2. Alice sends the signed document to Bob.
3. Bob decrypts the document with Alice's public key, thereby verifying the signature.

The protocol is far better than the previous one(Symmetric key). Because:

1. Trent is not needed to either sign or to verify signatures.
2. The parties do not even need Trent to resolve disputes: If Bob can not perform step (3), then knows the signature is not valid.

Signing Documents and Timestamps

- Actually Bob can cheat Alice in certain circumstances.
- He can reuse the document and signature together.
- This is no problem if Alice signed a contract but it can be very exciting if Alice signed a digital check.

It can be explained as follows:

- Alice sends Bob a signed digital check for $1000.
- Bob takes the money from the Bank.
- Off course Bob can saves a copy of the digital check.

- The following week he can again withdraw money from the bank with the saved check.
- In this way Bob can reuse the signed check as long as there are enough balance in the account.

Signing Doc and Timestamps Continue…

- To protect the misuse of the signed digital check, digital signatures often include timestamps.
- The date and time of the signature are attached to the message and signed along with the rest of the message.
- The Bank stores this timestamp in a database.
- Now, when Bob tries to cash Alice’s check second time, the bank checks the timestamp against its database.
- Since the bank already cashed a check from Alice with the same timestamp, the bank calls the police

Digital Signatures with Encryption

By combining digital signatures with public key cryptography, we develop a protocol that combines the security of encryption with the authenticity of digital signatures.

Think of a letter from your mother: The signature provides proof of authorship and the envelope provides privacy.

1. Alice sign the message with her private key. $S_A(M)$
2. Alice encrypts the signed message with bob's public key and sends it to Bob. $E_B(S_A(M))$
3. Bob decrypts the message with his private key. $D_B(E_B(S_A(M)))= S_A(M)$
4. Bob verifies with Alice's public key and recovers the message. $V_A(S_A(M) =M$

CHAPTER 7

Key Length

Symmetric Key Length

- The security of a symmetric cryptosystem is a function of two things:
  1. The strength of the algorithm. (it is more important) and
  2. The length of the key.
- If the strength of the algorithm is perfect, then there is no better way to break the cryptosystem other than trying every possible key in a brute-force attack.

- Brute-force attack is a known plaintext attack (i.e. ciphertext and the corresponding plaintext is required).
- The cryptanalyst does not need much plaintext to launch this attack.

Symmetric Key Length Continue…

- Calculating the complexity of a brute-force attack is easy.
- For key length 8 bits, there are 2^8=256 possible keys.
- For key length 56 bits, there are 2^{56} possible keys (then for a supercomputer with a speed of million keys/s need 2285 years).

- For key length 64 bits, the supercomputer needs 585,000 years to find the correct key among 2^{64} possible keys.
- For 128 bits, it takes 10^{25} years.
- For 1024 bits, a million million-attempts-per-second computers working in parallel will need 10^{597} years.

Time and Cost estimates of Brute-force attack

- Brute-force attack is a known plaintext attack.
- The speed depends on the following two parameter:
  1. The no. of keys to be tested. And
  2. The speed of each test (less important).

- According to Moore’s law: Computing power doubles approximately every 18 months. This means cost go down a factor of 10 every five years; what cost $1 million to build in 2010 will cost a mere $1 lac in the year 2015.

- A table is shown on the next slide about average time estimates for a Hardware brute-force attacks in 1995.

Time and Cost estimates of Brute-force attack Continue…

Average Time Estimates for a Hardware Brute-force Attacks in1995

Length of Key in Bits

Cost	40	56	64	80	112	128
$100K	2s	35h	1year	70,000y	10^{14}year	10^{19}year
$1M	0.2s	3.5h	37day	7,000y	10^{13}year	10^{18}year
$10M	0.02s	21m	4day	700y	10^{12}year	10^{17}year
$100M	2ms	2m	9h	70y	10^{11}year	10^{16}year
$1G	0.2ms	13s	1h	7y	10^{10}year	10^{15}year
$10G	0.02ms	1s	5.4m	245day	10^{9}year	10^{14}year
$100G	2μs	0.1s	32s	24day	10^{8}year	10^{13}year
$1T	0.2μs	0.01s	3s	2.4day	10^{7}year	10^{12}year
$10T	0.02μs	1ms	0.3s	6	10^{6}year	10^{11}year

Software Crackers

- Without special-purpose hardware and massively parallel machines, brute-force attacks are significantly harder.
- A software attack is about a thousand times slower than a hardware attack.

Public-key key Length

- Public key algorithms are based on the difficulty of factoring large numbers that are the product of two large primes.
- Breaking these algorithms does not involve trying every possible key; rather they involves trying to factor the large numbers.

- If the number is too small, you have no security.
- In 1977 Ron Rivest said that factoring a 125 digit number would take 40 quadrillion years.
- In 1994 a 129 digit number was factored.

Public-key key Length Continue...

- Table below shows factoring records over the past dozen years.

Factoring Using the Quadratic Sieve

Year	# of decimal digits factored	How many times harder to factor a 512 bit number
1983	71	>20 million
1985	80	>2 million
1988	90	2,50,000
1989	100	30,000
1993	120	500
1994	129	100

Birthday Attacks Against One-way Hash Functions

- There are two brute-force attacks against a one way hash functions.
  1. Given the hash of message, H(M), an adversaries would like to be able to create another document, M', such that H(M)=H(M').
  2. An adversaries would like to find two random messages, M, and M', such that H(M)=H(M').
- The second version of attack is much easier.

- The birthday paradox is a standard statistics problem. How many people must be in a room for the chance to be greater than even that one of them shares your birthday? The answer is 253, now how many people must be there be for the chance to be greater than even that at least two of them will share the same birthday? The answer is 23. With only 23 people in a room, there are still 253 different pairs of people in the room.

Birthday Attacks Against One-way Hash Functions Continue…

- Finding someone with a specific birthday is analogous to the first attack; finding two people with the same random birthday is analogous to the second attack.
- The second attack is known as birthday attack.
- Consider a one-way hash function that produce an m-bit output. Finding a message that hashes to a given hash value would require hashing 2^m random messages. Finding two messages that hash to the same value would only require hashing $2^{m/2}$ random messages.
- A machine with million messages/s would take 600,000 years to find a second message that matched a given 64-bit hash.
- The same machine could find a pair of messages that hashed to the same value in about an hour.

CHAPTER 8

Key Management

Generating Keys

- The security of an algorithm rest in the key
- If you are using a cryptographically weak process to generate keys, then your whole system is weak.
- In this case the cryptanalyst just cryptanalyze your key generation algorithm rather than your encryption algorithm.

Reduced keyspaces

- DES has a 56-bit key; thus 2^{56} (10^{16}) possible keys.
- However, Norton Discreet for MS-DOS(version 8.0 and earlier) only allows ASCII key, forcing the high order bit of each byte to be zero. The program also converts lowercase letters to uppercase resulting only 2^{40} keys.
- This poor key generation procedures have made its DES ten thousand times easier to break than proper implementation.
- Table 1 shows the number of possible keys with various constraints on the input strings.
- Table 2 shows the time required for an exhaustive search through all of those keys, given a million attempts per second.

Reduced keyspaces Continue…

Table 1

Number of Possible keys of various Keyspaces

	4-Bytes	5-Bytes	6-Bytes	7-Bytes	8-Bytes
Lowercase letters [26]	460,000	$1.2*10^{7}$	$3.1*10^{8}$	$8.0*10^{9}$	$2.1*10^{11}$
Lowercase letters and digits [36]	1,700,000	$6.0*10^{7}$	$2.2*10^{9}$	$7.8*10^{10}$	$2.8*10^{12}$
Alphanumeric characters [62]	$1.5*10^{7}$	$9.2*10^{8}$	$5.7*10^{10}$	$3.5*10^{12}$	$2.2*10^{14}$
Printable characters [95]	$8.1*10^{7}$	$7.7*10^{9}$	$7.4*10^{11}$	$7.0*10^{13}$	$6.6*10^{15}$
ASCII characters [128]	$2.7*10^{8}$	$3.4*10^{10}$	$4.4*10^{12}$	$5.6*10^{14}$	$7.2*10^{16}$
8-bit ASCII characters [256]	$4.3*10^{9}$	$1.1*10^{12}$	$2.8*10^{14}$	$7.2*10^{16}$	$1.8*10^{19}$

Reduced keyspaces Continue…

Table 2

Exhaustive Search of various keyspaces (1 million attempts/s machine)

	4-Bytes	5-Bytes	6-Bytes	7-Bytes	8-Bytes
Lowercase letters [26]	0.5s	12s	5m	2.2h	2.4d
Lowercase letters and digits [36]	1.7s	1m	36m	22h	33d
Alphanumeric characters [62]	15s	15m	16h	41d	6.9y
Printable characters [95]	1.4m	2.1h	8.5d	2.2y	210y
ASCII characters [128]	4.5m	9.5h	51d	18y	2300y
8-bit ASCII characters [256]	1.2h	13d	8.9y	2300y	580,000y

Poor key choice

- When people choose their own keys, they generally choose poor ones.
- They like to choose “Abdullah” rather than “@67dh4&”.
- Because “Abdullah” is easy to remember than “@67dh4&”.

- A smart brute-force attack does not try all possible keys in numerical order; it tries the obvious keys first.
- This is called a dictionary attack, because the attacker uses a dictionary of common keys.

- Daniel Klein was able to crack 40 percent of the passwords on the average computer using this system.

Random keys

- Good keys are random-bit strings generated by some automatic process.
- If the key is 64-bits long, every possible 64-bit key must be equally likely.
- Generate the key bits from either a reliably random source or a cryptographically secure pseudo-random-bit generator.

Pass Phrases

- A better solution is to use an entire phrase instead of a word, and to convert that phrase into a key.
- These phrases are called pass phrases.
- A technique called key crunching converts the easy-to-remember phrases into random keys.
- Use a one-way hash function to transform an arbitrary-length text string into pseudo-random-bit string.
- For example, the easy to remember text string:

  Rajshahi university is the 2nd largest university of Bangladesh, students are over 26000.

  Might crunch into this 64-bit key: 23ht 5t6r yu76 980y

Transferring Keys

- The X9.17 standard specifies two types of keys:
    1. Key encryption keys: which encrypt other keys for distribution.
    2. Data keys: which encrypt message keys.
- This two-tiered key concept is used a lot in the key distribution.

- Another solution to the distribution problem splits the key into several different parts and sends each of those parts over a different channel.
- One part could be sent over the telephone, one by mail, one by overnight delivery service, one by carrier pigeon, and so on.

Transferring Keys Continue…

- Alice sends Bob the key-encryption key securely, either by a face to face meeting or the splitting technique just discussed.

- Once Alice and Bob both have the key encryption key, Alice can send Bob daily data keys over the same communications channel.
- Alice encrypts each data key with the key encryption key.

- Since the amount of traffic being encrypted with the key encryption key is low, it does not have to be changed as often.

Key Distribution in Large Network

- Key encryption keys works well in small networks, but can quickly get cumbersome if the networks become large.

- Since every pair of users must exchanges keys, the total no. of key exchanges required in an n-person network is n(n-1)/2.
- Example: for 6 person required 15 key exchanges. For 1000 person required nearly 500,000 key exchanges.

- In these cases, creating a central server makes the operation much more efficient.

Verifying Keys

- When Bob receives a key, how does he know it came from Alice and not from someone pretending to be Alice?
- If Alice gives it to him when they are face-to-face, its easy.
- If Alice sends it via a trusted courier, then Bob has to trust the courier.

- If the key is encrypted with the key encryption key, then Bob has to trust the fact that only Alice has the key.
- If Alice uses a digital signature protocol to sign the key, Bob has to trust the public key database when he verifies that signature.

- Bob could also verifies Alice’s key over the telephone.
- If it’s a public key, he can safely recite it in public.
- If it’s a private key, he can use a one way hash function to verify the key.

Updating Keys

- If you want to change key daily, an easier way is to generate a new key from the old key; this is sometimes called key updating.
- All it takes is a one-way function.
- Key updating works, but remember that the new key is only as secure as the old was.
- If Eve managed to get her hands on the old key, she can perform the updating function herself.

Storing Keys

- Users can either directly enter the 64-bit key or enter the key as a longer character string. The system then generates a 64-bit key from the character string using a key crunching technique.

- Another solution is to store the key in a magnetic stripe card, plastic key with embedded ROM chip, or smart card.
- User can then enter the key by inserting the physical token into a special reader in his encryption box or attached to the computer terminal.

- Hard to remember key can be stored in encrypted form, using something similar to a key-encryption key. For example, an RSA private key could be encrypted with a DES key and stored on disk. To recover the RSA key, the user has to type in the DES key to a decryption program.

Lifetime of Keys

- No encryption key should be used for an indefinite period. There are several reasons for this.

- The longer a key is used, the greater the chance that it will be compromised.
- The longer a key is used, the greater the loss if the key is compromised.
- The longer a key is used. The greater the temptation for someone to spend the effort necessary to break it.
- It is generally more easier to do cryptanalysis with more ciphertext encrypted with the same key.

Destroying Keys

- Old key must be destroyed.
- If the key is written on paper, the paper should be shredded or burned.
- If the key is in a hardware EEPROM, the key should be overwritten multiple times.
- If the key is in a hardware EPROM or PROM, the chip should be smashed into tiny bits and scattered to the four winds.

- If the key is stored on a computer disk, the actual bits of the storage should be overwritten multiple times or the disk should be shredded.

Public Key Management

- Alice can get Bob’s public key several ways.
  1. She can get it from Bob.
  2. She can get it from centralized database.
  3. She can get from her own private database.

CHAPTER 11

Mathematical Background

Number Theory

Prime Numbers:

- A prime number is an integer greater than 1 whose only factors are 1 and itself.
- Cryptography, especially public key cryptography, uses large primes (512 bits and even larger).

Greatest Common Divisor (gcd):

- Two numbers are relatively prime when they share no factors in common other than 1.
- In other words, if the greatest common divisor of a and n is equal to 1. This is written as:

$$\gcd(a, n)=1$$

The number 15 and 38 are relatively prime, 15 and 55 are not.

Greatest Common Divisor

- Knuth describes the algorithm and some modern modifications in C.

```
Int gcd (int x, int y)
{
    int g;
    If (x<0) x= -x;
    If (y<0) y= -y;
    If (x+y==0) Error;
    g=y;
    While (x>0)
    {
            g=x;
            x=y%x;
            y=g;
    }
    return(g);
}
```

Inverses Modulo a Number

- The multiplicative inverse of 3 is 1/3, because 3*1/3=1.
- In modulo world, the problem is more complicated.

$$4*x \equiv 1 \pmod{7}.$$

This equation is equivalent to finding an x and k such that

$4x=7k+1.$ where both x and k are integers.

The general problem is finding an x such that

$$1= (a*x) \bmod n.$$

This is written as

$$a^{-1} \equiv x \pmod{n}.$$

- The inverse modulo problem is a lot more difficult to solve.
- Sometimes it has solutions, sometimes not. For example, the inverse of 5, modulo 14, is 3. On the other hand, 2 has no inverse modulo 14.

Prime Number Generation

- Public key algorithms need prime numbers.
- Let us answers some obvious questions:

1. If everyone need a different prime numbers, won't we run out?

   Answer: No. In fact, there are approximately 10^{151} primes 512 bits in length or less. For numbers near n, the probability that a random number is prime is approximately one in (ln n). So the total number of primes less than n is n/(ln n).

2. What if two people accidentally pick the same prime numbers?

   Answer: It won't happen. With over 10^{151} prime numbers to choose from, the odds of that happening are significantly less than odds of your computer spontaneously combusting at the exact moment you win the lottery.

3. If someone creates a database of all primes, won't he be able to use that database to break public key?

   Yes, but he can't do it. If you store 1GB of information on a drive weighing 1 gram, then a list of just the 512 bit primes would weigh so much that it would exceed the chandrasekhar limit and collapse into a black hole.

Prime Number Generation

- The wrong way to find primes is to generate random numbers and then try to factor them.
- The right way is to generate random numbers and test if they are prime.

Lehmann: A simple test was developed independently by lehmann. Here it test if p is prime.

1. Choose a random number a less than less than p.
2. Calculate $a^{(p-1)/2}$ mod p.
3. If $a^{(p-1)/2} \neq 1$ or -1 (mod p), then p is definitely not prime.
4. If $a^{(p-1)/2} \equiv 1$ or -1 (mod p), then the likelihood that p is not prime is no more than 50 percent.

Repeat this test t times. If the calculation equals 1 or -1, but does not always equal 1, then p is probably prime with an error rate of 1 in 2^t.

Robin-Miller

- This algorithm is very easy and used by everyone.
- Choose a random number p to test. Calculate b, where b is the number of times 2 divides p-1 (i.e. 2^b is the largest power of 2 that divides p-1). Then calculate m such that $p=1+2^b*m$.

1. Choose a random number , a, such that a is less than p.
2. Set j=0 and set $z=a^m$ mod p.
3. If z=1, or z=p-1, then p passes the test and may be prime.
4. If j>0 and z=1, then p is not prime.
5. Set j=j+1. If j<b and j ≠ p-1, set $z=z^2$ mod p and go back to step[4]. If z=p-1, then p passes the test and may be prime.
6. If j=b and z ≠ p-1, then p is not prime.

Practical Considerations

- In real world implementations, prime generation goes quickly.
  1. Generate a random n-bit number, p.
  2. Set the high order and low order bit to 1. (The high order bit ensures that the prime is of the required length and the low order bit ensures that it is odd).
  3. Check to make sure p is not divisible by any small primes: 3, 5, 7, 11, and so on. Many implementations test p for divisibility by all primes less than 256. The most efficient is to test for divisibility all primes less than 2000.
  4. Perform the Rabin –Miller test for some random a. If p passes, generate another random a and go through the test again. Choose a small value of a to make the calculations go quicker. If p fails one of the tests, generate another p and try again.

CHAPTER 12

Data Encryption Standard (DES)

Background

- The Data Encryption Standard (DES), also known as Data Encryption Algorithm (DEA) by ANSI and the DEA-1 by ISO, has been a worldwide standard for a twenty years.

Development of the Standard

- In the May 15, 1973 Federal Register, the NBS issued a public request for proposals for a standard cryptographic algorithm. They specified a series of design criteria:
    1. The algorithm must provide a high level of security.
    2. The algorithm must be completely specified and easy to understand.
    3. The security of the algorithm must reside in the key; the security should not depend on the secrecy of the algorithm.
    4. The algorithm must be available to all users.
    5. The algorithm must be adaptable for use in diverse applications.
    6. The algorithm must be economically implementable in electronic devices.
    7. The algorithm must be efficient to use.
    8. The algorithm must be able to be validated.
    9. The algorithm must be exportable.

Description of DES

- DES is a block cipher.

- It encrypts data in 64-bit blocks.
- It is a symmetric algorithm.
- The key length is 56 bits.

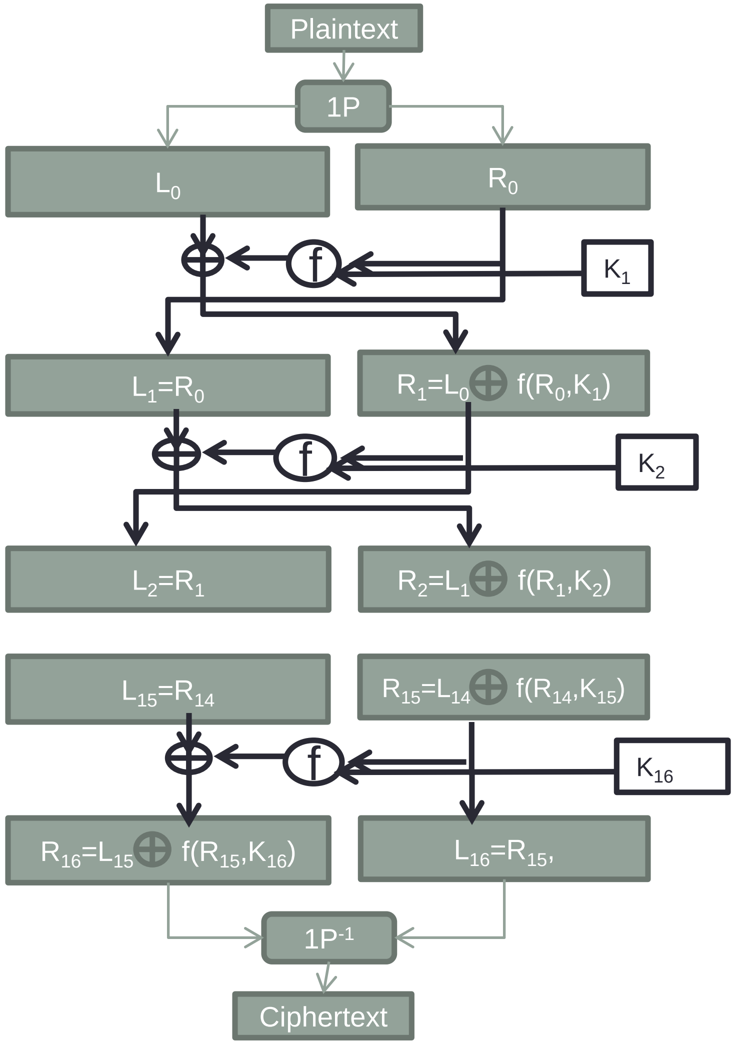
- The key usually expressed as a 64 bit number, but every 8th bit is used for parity checking and is ignored.
- The algorithm is nothing more than a combination of the two basic techniques of encryption: Confusion and Diffusion.

- The fundamental building of DES is a single combination of these techniques (a substitution followed by a permutation) on the test based on the key.

- This is known as a round.
- DES has 16 rounds. (Figure on the next slide).

Figure: DES

- DES:


Plaintext
1P
L_0
R_0
f
K_1
$L_1=R_0$
$R_1=L_0 \oplus f(R_0,K_1)$
f
K_2
$L_2=R_1$
$R_2=L_1 \oplus f(R_1,K_2)$
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14},K_{15})$
f
K_{16}
$R_{16}=L_{15} \oplus f(R_{15},K_{16})$
$L_{16}=R_{15},$
$1P^{-1}$
Ciphertext

Outline of the Algorithm

- After an initial permutation, the block is broken into a right half and a left half, each 32 bits long.
- Then there are 16 rounds of identical operations, called function f, in which the data are combined with the key.

- After the 16 round the right and left halves are joined, and a final permutation (the reverse of initial permutation) are performed.

- In each round(figure on the next slide), the key bits are shifted and then 48 bits are selected from the 56 bits of the key.

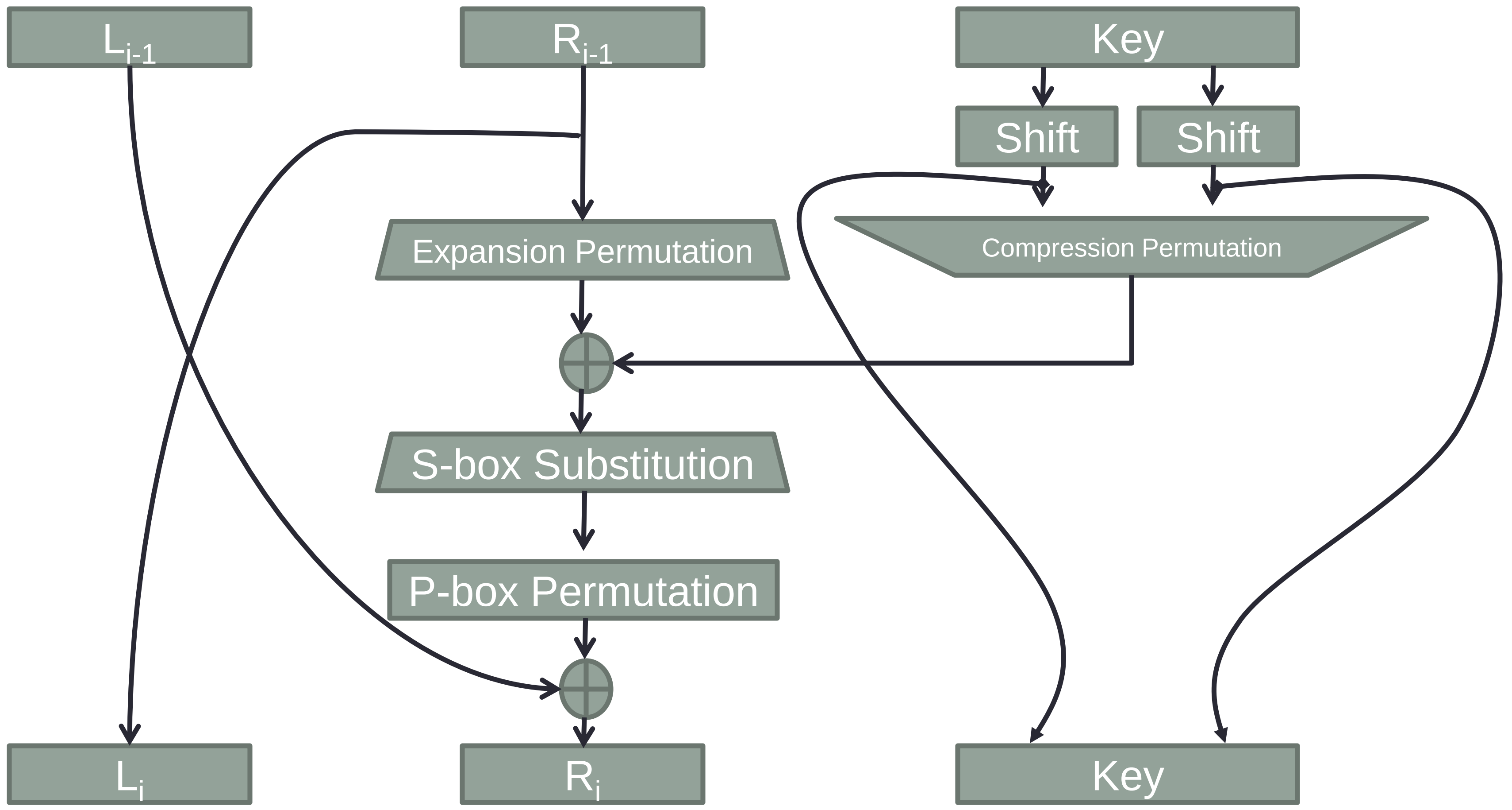
- The right half of the data is expanded 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again.

- These four operations make up the function f.
- The output of function f is then combined with the left half via another XOR.

- The result of these operations becomes the new right half.
- The old right half becomes the new left half.
- These operations are repeated 16 times, making 16 rounds of DES.

Figure: DES one round

- DES one Round:



The Initial Permutation

- This permutation occurs before round 1; it transposes the input block as described in the following table:

58,	50,	42,	34,	26,	18,	10,	2,	60,	52,	44,	36,	28,	20,	12,	4
62,	54,	46,	38,	30,	22,	14,	6,	64,	56,	48,	40,	32,	24,	16,	8
57,	49,	41,	33,	25,	17,	9,	1,	59,	51,	43,	35,	27,	19,	11,	3
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

- This table is read left to right, top to bottom.
- In the table we see, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to position 2, bit 42 to bit position 3 and so forth.

The Key Transformation

- Initially 64 bit DES key is reduced to a 56 bit key by ignoring every eight bit. Table below shows this.

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18
10,	2,	59,	51,	43,	35,	27,	19,	11,	3,	60,	52,	44,	36
63,	55,	47,	39,	31,	23,	15,	7,	62,	54,	46,	38,	30,	22
14,	6,	61,	53,	45,	37,	29,	21,	13,	5,	28,	20,	12,	4

- The bits 8, 16, 24… can be used as a parity check to ensure the key is error free.
- After a 56 bit key is extracted a different 48 bit subkey k_i is generated for each of the 16 rounds of DES.

- For this, 56 bit key is divided into two 28 bit halves. Then, the halves are circularly shifted left by either one or two bits, depending on the round (table on next slide shown).

The Key Transformation Continue…

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number Of bit shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- After being shifted, 48 out of 56 bits are selected.

- Because this operation permutes the order of the bits as well as selects a subset of bits, it is called a compression permutation.

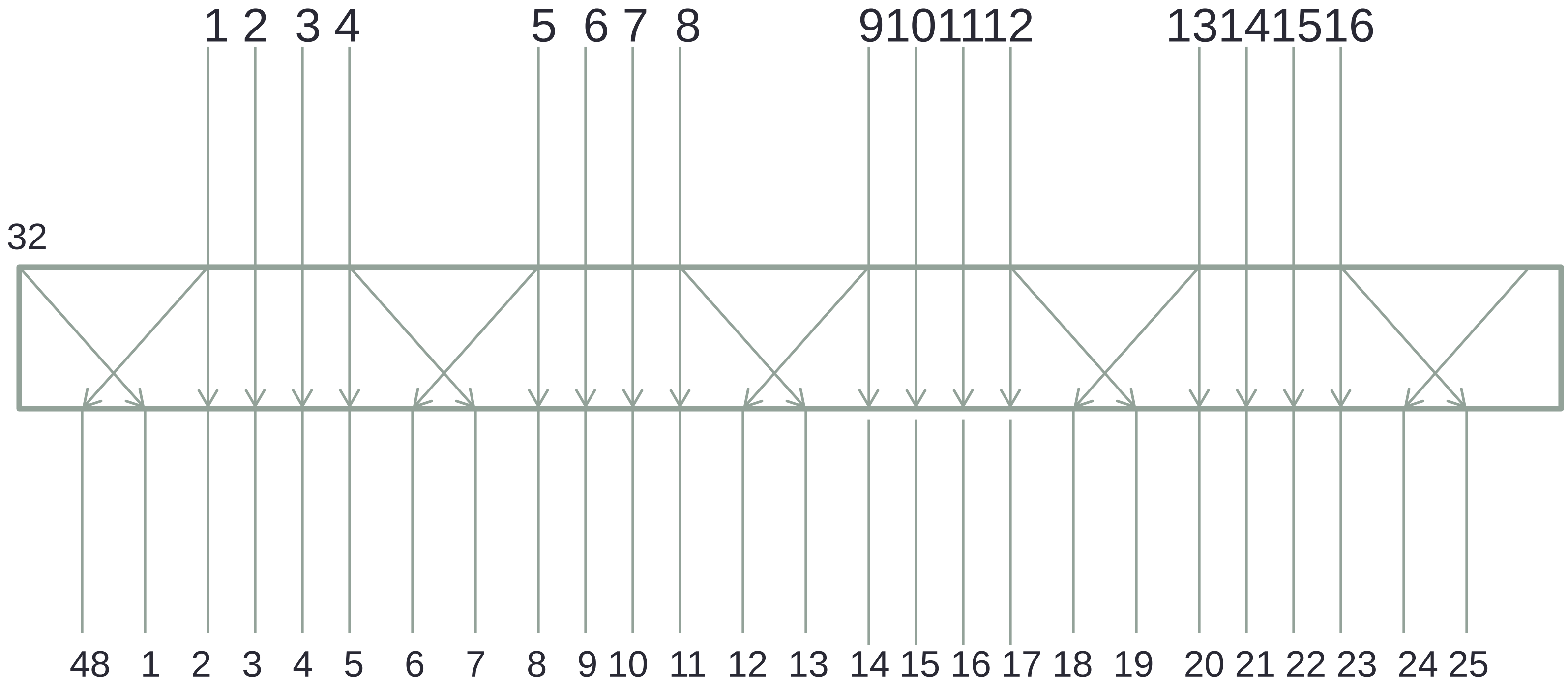
The Expansion Permutation

- The operations expands the right half of the data, R_i, from 32 bits to 48 bits.
- Figure on the next slide show the expansion permutation.
- This is also known as E-box.
- For each 4 bit input block, the first and fourth bits each represent two bit of the output block, while the second and third bits each represent one bit of the output block.
- Table below shows which output positions correspond to which input positions.

Expansion Permutation
32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9
8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25
24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1

Figure: The Expansion Permutation

- The Expansion Permutation:



The S-box Substitution

- The substitutions are performed by eight substitution boxes, or S-boxes.
- Each S-box has a six bit input and a four bit output.
- The first 6 bit block is operated on by S-box 1, the next six bit block is by S-box 2, and so on (Figure on the next slide).
- Each S-box is a table of 4 rows and 16 columns.
- Each entry in the box is a 4 bit number.
- The 6 input bits of the S-box specify under which row and column number to look for the output (See table on the slide after next).
- Bits b1 and b6 are combined to form a 2 bit number, from 0 to 3 which corresponds to a row in the table.
- The middle 4 bits b2 through b5 are combined to form a 4 bit number, from 0 to 15, which corresponds to a column in the table.
- For example: Assume input to sixth S-box (bit 31 through 36 of the XOR function) is 110011, the first and last bits combine to 11, which corresponds to row 3, and the middle four bits 1001 corresponds to column 9 of the sixth S-box so the entry is 14 (row/column start from 0).
- The 1110 is substituted for 110011.

Figure: S-box Substitution

- S-box Substitution:

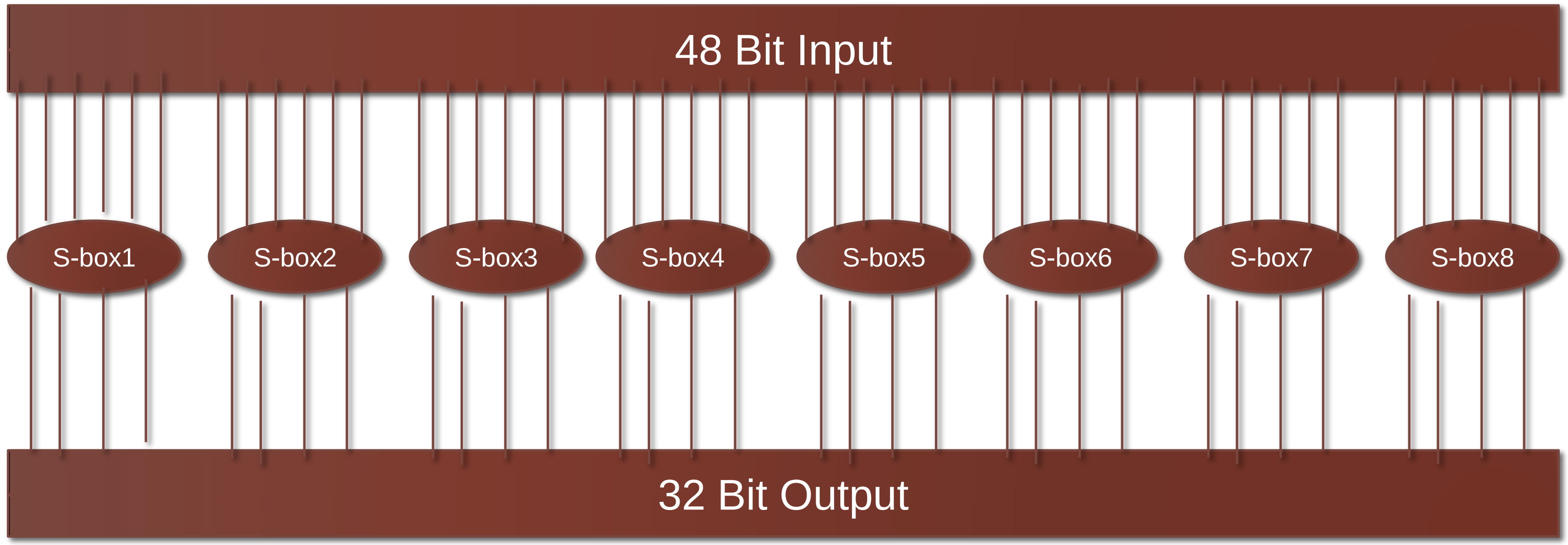


Table: S-Boxes

S-Box 1	14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7
	0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8
	4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0
	15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13

S-Box 2	15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10
	3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5
	0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15
	13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9

S-Box 3	10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8
	13 7 0 9 3 4 6 10 2 8 5 14 12 11 15 1
	13 6 4 9 8 15 3 10 11 1 2 12 5 10 14 7
	1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12

S-Box 4	7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15
	13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9
	10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4
	3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14

S-Box 5	2 12 4 1 7 10 11 6 8 5 3 15 13 0 14 9
	14 11 2 12 4 7 13 1 5 0 15 10 3 9 8 6
	4 2 1 11 10 13 7 8 15 9 12 5 6 3 0 14
	11 8 12 7 1 14 2 13 6 15 0 9 10 4 5 3

S-Box 6	12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11
	10 15 4 2 7 12 9 5 6 1 13 14 0 11 3 8
	9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6
	4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13

S-Box 7	4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1
	13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6
	9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6
	4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13

S-Box 8	13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7
	1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2
	7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8
	2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11

P-Box Permutation

- The 32 bits output of the S-box substitution is permuted according to P-box.
- Table below shows the position to which each bit moves.

- Example: bit 21 moves to bit 4, while bit 4 moves to bit 31.
- The result of P-box permutation is XORed with the left half of the initial 64 bit block.

P-Box Permutation
16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10 2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25

The Final Permutation

- The final permutation is the inverse of the initial permutation as told earlier.
- Table below show this final permutation.
- Also note the left and right halves are not exchanged after the last round of DES; the concatenated block $R_{16}L_{16}$ is used as the input to the final permutation.

Initial Permutation
58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4
62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48, 40, 32, 24, 16, 8
57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3
61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7

Final Permutation
40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63, 31
38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29
36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27
34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25

-

Decryption of DES

- Actually the same algorithm can be used for both encryption and decryption.
- The only difference is that the key must be used in reverse order.

- That is, if the encryption keys for each round are K_1, K_2, K_3,…,K_{16}, then the decryption keys are K_{16}, K_{15}, K_{14},…,K_1.
- The algorithm that generates the key used for each round is circular as well.

- The key shift is a right shift and the number of positions shifted is 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

DES Variants

Multiple DES:

- Some DES implementation use triple-DES (Figure on the next slide).
- Since DES is not a group, then the resultant ciphertext is much harder to break using exhaustive search.

DES with Independent Subkey:

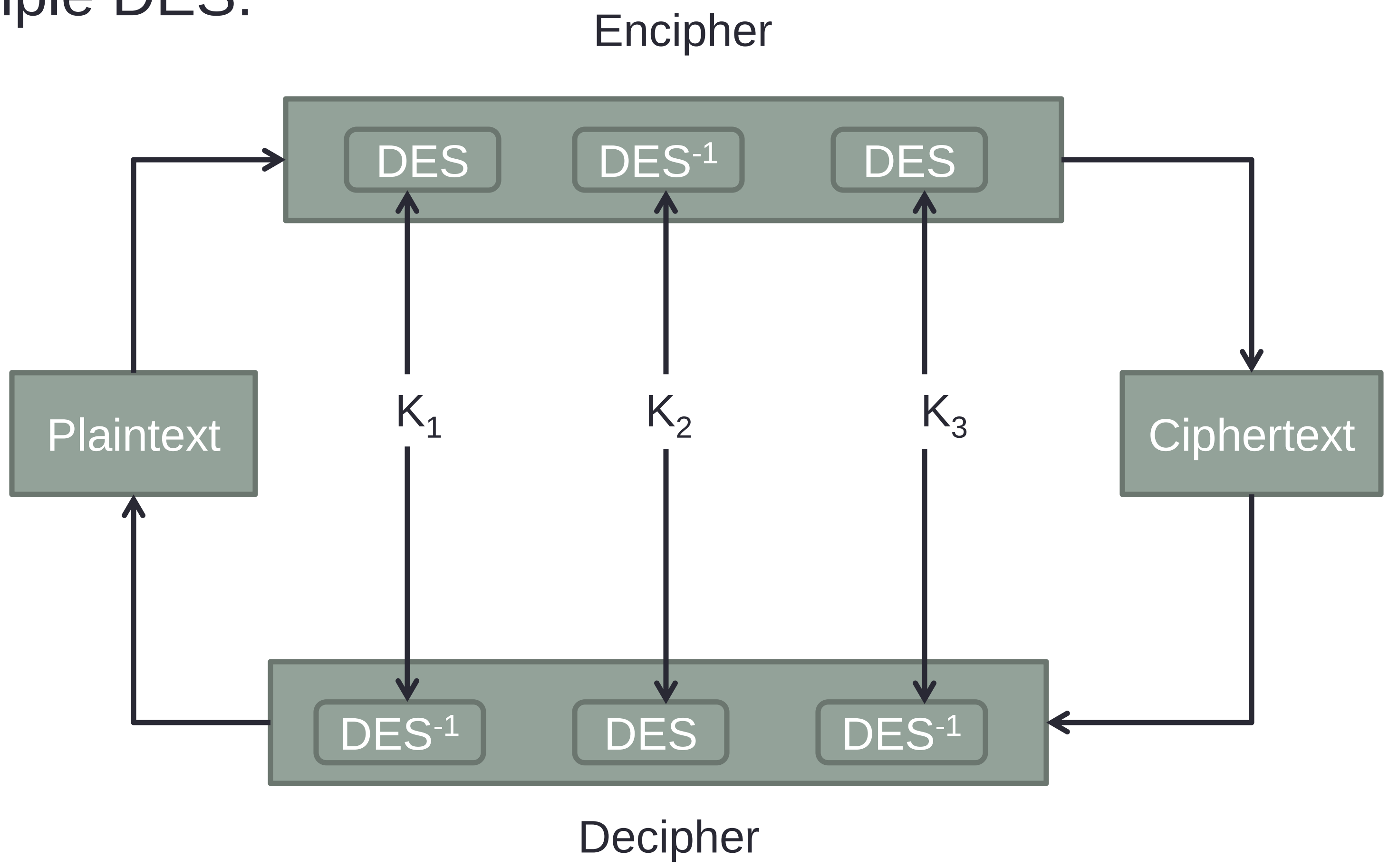
- Another variation is to use a different subkey for each round, instead of generating them from a 56 bit key.

- Since 48 key bits are used in each of 16 rounds, this means that the key length for this variant is 768 bits.

- This variant would drastically increase the difficulty of a Brute-force attack against the algorithm; that attack would have a complexity of 2^{768}.

Figure: Triple DES

- Triple DES:



CHAPTER 14

Still Other Block Cipher

RC5 (Rivest Cipher)

- RC5 is a block cipher with a variety of parameters: block size, key size, and number of rounds.
- It was invented by Ron Rivest and analyzed by RSA laboratories.
- There are three operations: XOR, addition and rotations.
- RC5 is a variable length block, but we will focus on a 64-bit data block.
- Encryption uses 2r+2 key dependent 32-bit words – S_0, S_1, S_2,…, S_{2r+1}; where r is the number of rounds.
- To encrypt, first divide the plaintext block into two 32-bit words: A and B.

RC5 Continue…

- Then encryption is as follows:

$$A = A+S_0$$
$$B = B+S_1$$

For i = 1 to r:

$$A = ((A \oplus B) <<< B)+S_{2i}$$
$$B = ((B \oplus A) <<< A)+S_{2i+1}$$

<<< is left circular shift

Decryption is just as easy. Divide the Ciphertext block into two words, A and B and then

For i = r down to 1:

$$B = ((B-S_{2i+1}) >>> A) \oplus A$$
$$A = ((A-S_{2i}) >>> B) \oplus B$$

$$B = B-S_1$$
$$A = A-S_0$$

RC5 Continue…

- First step of key Expansion:
  - First copy the bytes of the key into an array, L, of c 32-bit words, padding the final word with zero if necessary.

- Second step of Key Expansion:
  - Then initialize an array S, using linear congruential generator mod 2^{32}.

```
$S_0 = P$
For $i = 1$ to $2(r+1)-1$:
$S_i = (S_{i-1}+Q) \bmod 2^{32}$.
```

Where P = 0xb7e15163 and Q = 0x9e3779b9 are constant based on the binary representation of e and phi.

RC5 Continue…

- Third step of Key Expansion: (mix L into S)

```
i=j=0
A=B=0
do 3n times (where n is the maximum of 2(r+1) and c):
$A=S_i=(S_i+A+B)<<<3$
$B=L_j=(L_j+A+B)<<<(A+B)$
$i=(i+1) \bmod 2(r+1)$
$j=(j+1) \bmod c$
```

We just defined RC5 with a 32-bit word size and 64-bit block; the same algorithm can also be used as a 64-bit word size and 128-bit block size. For w=64, P=0xb7e151628aed2a6b and Q=0x9e3779b97f4a7c15.

Rivest designates particular implementations of RC5 as RC5-w/r/b, where w is word size, r is round number and b is length of the key in bytes.

CHAPTER 18

One-Way Hash Functions

Background

- A one-way hash function H(M), operates on an arbitrary-length pre-image message M and returns a fixed-length hash value h.

  h = H(M), where h is of length m.

- Many functions can take an arbitrary length input and return an output of fixed length, but one-way hash function have additional characteristics that make them one-way.

1. Given M, it is easy to compute h.
2. Given h, it is hard to compute M such that H(M)=h.
3. Given h, it is hard to find another message M`, such that H(M) = H(M`)

Length of One-Way Hash Functions

- Hash functions of a 64-bits are just too small to survive a birthday attack. Most practical hash function produce 128-bit hashes.
- This forces anyone attempting the birthday attack to hash 2^{64} random documents to find two that hash to the same value, not enough for lasting security.

- NIST, in its Secure Hash Standard(SHS) uses a 160-bit hash value.
- This makes the birthday attack even harder, required 2^{80} random hashes.
- The following method has been proposed to generate a longer hash value than a given hash function produces.

1. Generate a hash value of a message (using any one-way hash function).
2. Prepend the hash value to the message.
3. Generate the hash value of the concatenation of the message and the hash value.
4. Create a larger hash value consisting of the hash value generated in step (1) concatenated with the hash value generated at step(3).
5. Repeat steps (1) through (3) as many times as you wish, concatenating as you go.

- Although this method has never been proved to be either secure or insecure, various people have some serious reservations about it.

MD4

- MD4 is a one-way hash function designed by Ron Rivest.
- MD stands for Message Digest, the algorithm produces a 128-bit hash or message digest of the input message.

- Design goals of MD4 (outlines by Rivest):

1. Security: It is computationally infeasible to find two messages that hashed to the same value.
2. Direct Security: MD4's security is not based on any assumption, like the difficulty of factoring.

3. Speed: Suitable for high speed software implementation.
4. Simplicity and Compactness: MD4 is as simple as possible without large data structures or a complicated program.
5. Favor Little Endian Architecture: MD4 is optimized for microprocessor architecture specially for Intel Microprocessor.

MD5

- MD5 is an improved version of MD4 and more complex than MD4.
- It is similar in design and also produces a 128-bit hash.

Description of MD5:

- As an initial processing, the algorithm produce a block multiple of 512, of the input message.
- Each 512-bit is then divided into sixteen 32-bit sub-blocks.
- The output of the algorithm is a set of four 32-bit blocks, which concatenate to form a single 128-bit hash value.

- First the message is padded so that its length is just 64 bits short of being a multiple of 512.
- This padding is a single 1-bit added to the end of the message, followed by as many zeroes as are required.

MD5 Continue…

- Then a 64-bit representation of the message's length(before padding were added) is appended to the result.
- This ensures that different messages will not look the same after padding.
- MD5 has 4 rounds of 16 operations each.

- Each operation performs a nonlinear function on three of a, b, c and d. Then it adds the result to the fourth variable, a sub-block of the text and a constant. Then it rotates that result to the right a variable number of bits and adds the result to one of a, b, c, or d. Finally the result replaces one of a, b, c or d.

MD5 Continue…

If M_j represents the jth sub-block of the message (from 0 to 15), and <<<s represents a left circular shift of s bits, the four operations are:
(The FF(), GG(), HH() and II() operations corresponds to 1st, 2nd ,3rd, and 4th rounds respectively)

FF(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + F(b, c, d) + M_j + t_i)<<<s)
GG(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + G(b, c, d) + M_j + t_i)<<<s)
HH(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + H(b, c, d) + M_j + t_i)<<<s)
II(a, b, c, d, M_j, s, t_i) denotes a = b + ((a + I(b, c, d) + M_j + t_i)<<<s)

The four nonlinear functions associated with each operations are as follows:

$$F(X, Y, Z) = (X \wedge Y) \vee (\neg X) \wedge Z)$$
$$G(X, Y, Z) = (X \wedge Z) \vee (Y \wedge (\neg X))$$
$$H(X, Y, Z) = X \oplus Y \oplus Z$$
$$I(X, Y, Z) = Y \oplus (X \vee (\neg Z)$$

MD5 Continue…

- Four 32-bit variables are initialized:

  A = 0x01234567

  B = 0x891bcdef

  C = 0xfedcba98

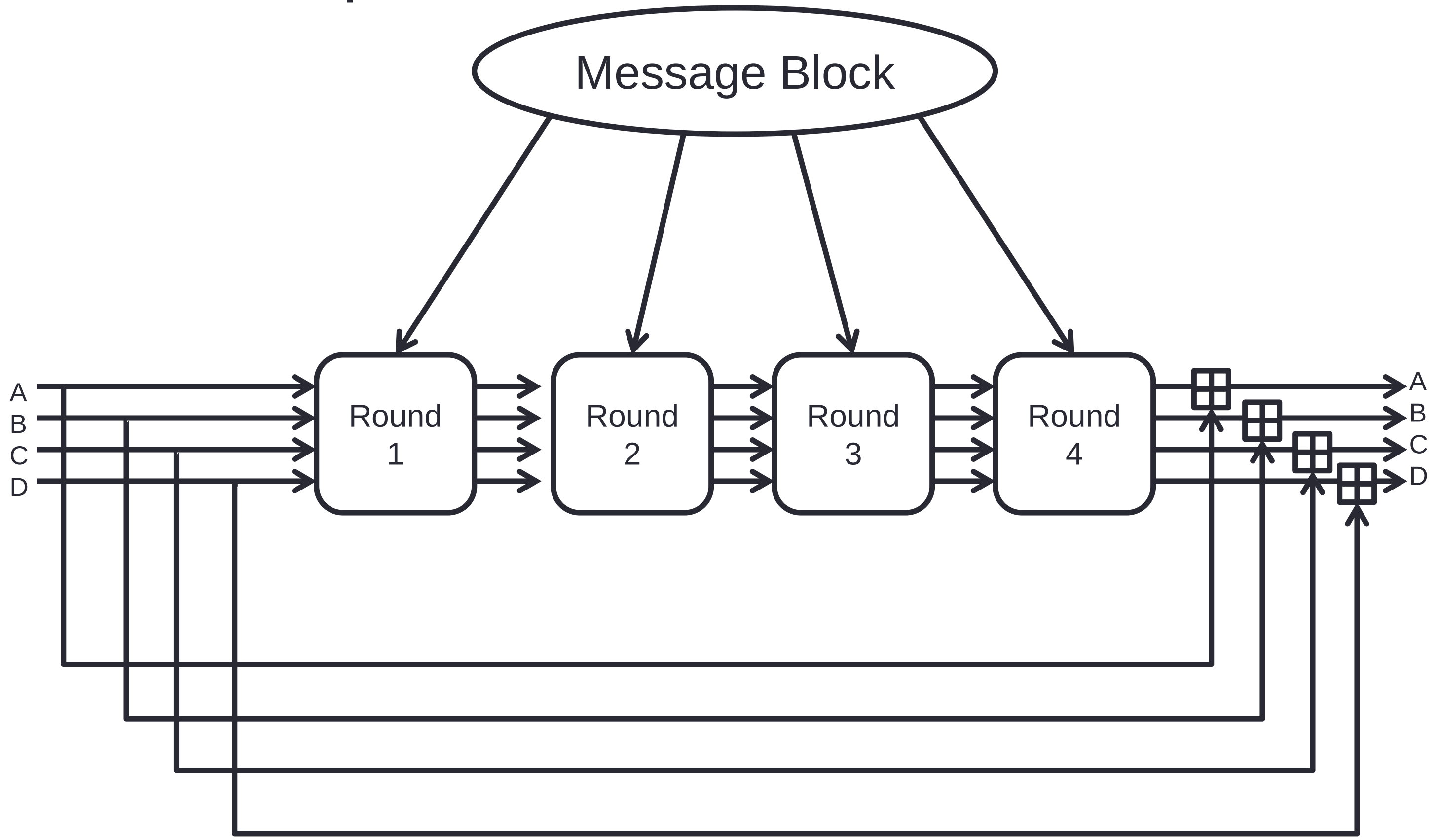
  D = 0x76543210

- These are called chaining variables. Now the main loop begins. The loop continues for as many 512-bit blocks as are in the message.

- The four variables are copied into different variables: a gets A, b gets B, c gets C, and d gets D.

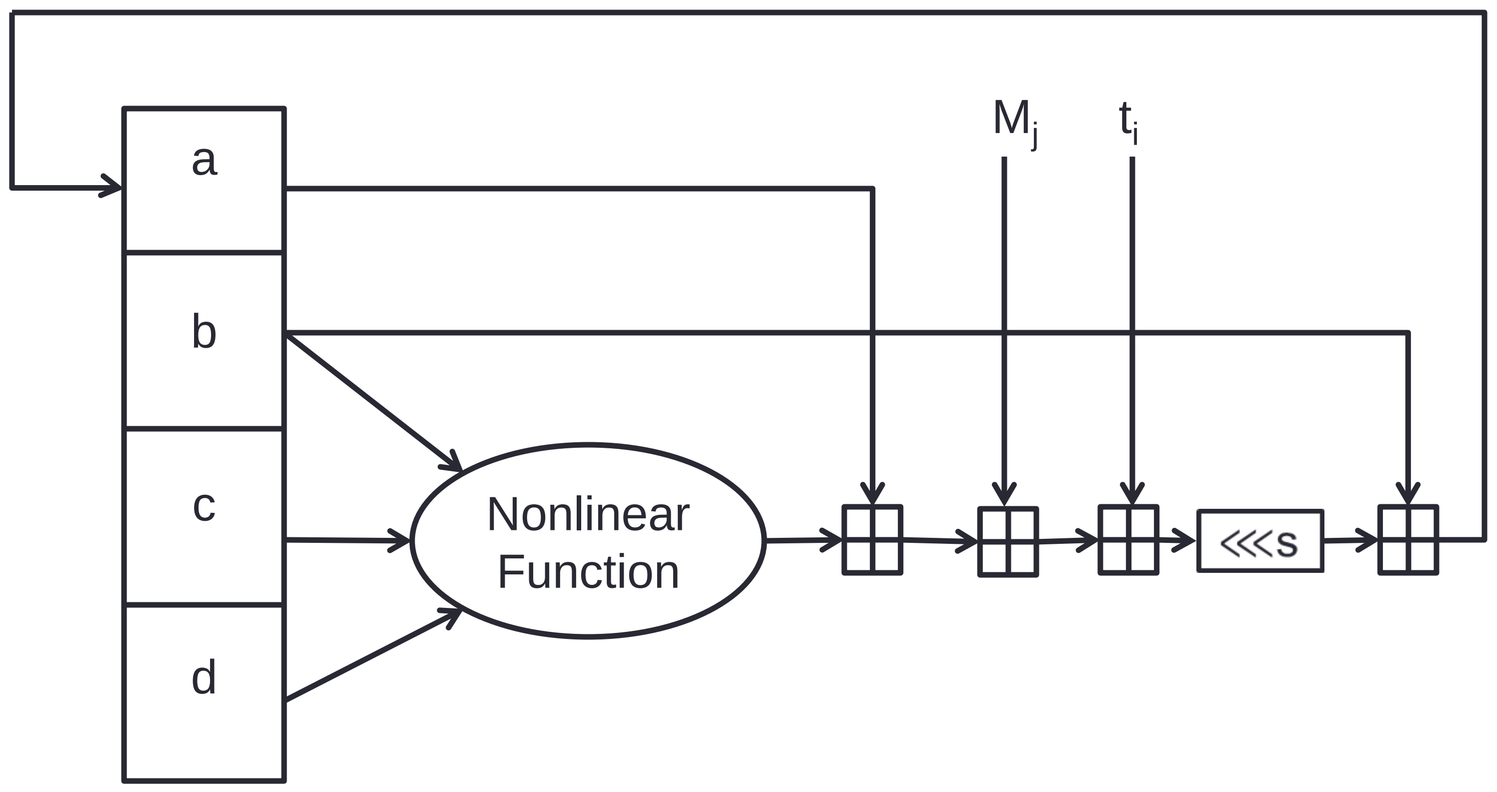
MD5 Continue…

MD5 main loop:



MD5 Continue…

- One MD5 operation:



MD5 Continues…

- The four rounds(64 steps) looks like:

Round 1:

FF(a, b, c, d, M_0, 7, 0xd76aa478)
FF(d, a, b, c, M_1, 12, 0xe8c7b756)
FF(c, d, a, b, M_2, 17, 0x242070db)
FF(b, c, d, a, M_3, 22, 0xclbdceee)
FF(a, b, c, d, M_4, 7, 0xf57c0faf)
FF(d, a, b, c, M_5, 12, 0x4787c62a)
FF(c, d, a, b, M_6, 17, 0xa8304613)
FF(b, c, d, a, M_7, 22, 0xfd469501)
FF(a, b, c, d, M_8, 7, 0x698098d8)
FF(d, a, b, c, M_9, 12, 0x8b44f7af)
FF(c, d, a, b, M_{10}, 17, 0xffff5bb1)
FF(b, c, d, a, M_{11}, 22, 0x895cd7be)
FF(a, b, c, d, M_{12}, 7, 0x6b901122)
FF(d, a, b, c, M_{13}, 12, 0xfd987193)
FF(c, d, a, b, M_{14}, 17, 0xa679438e)
FF(b, c, d, a, M_{15}, 22, 0x49b40821)

MD5 Continue…

Round 2:

GG(a, b, c, d, M_1, 5, 0xf61e2562)
GG(d, a, b, c, M_6, 9, 0xc040b340)
GG(c, d, a, b, M_{11}, 14, 0x265e5a51)
GG(b, c, d, a, M_0, 20, 0xe9b6c7aa)
GG(a, b, c, d, M_5, 5, 0xd62f105d)
GG(d, a, b, c, M_{10}, 9, 0x02441453)
GG(c, d, a, b, M_{15}, 14, 0xd8a1e681)
GG(b, c, d, a, M_4, 20, 0xe7d3fbc8)
GG(a, b, c, d, M_9, 5, 0x21e1cde6)
GG(d, a, b, c, M_{14}, 9, 0xc33707d6)
GG(c, d, a, b, M_3, 14, 0xf4d50d87)
GG(b, c, d, a, M_8, 20, 0x455a14ed)
GG(a, b, c, d, M_{13}, 5, 0xa9e3e905)
GG(d, a, b, c, M_2, 9, 0xfcefa3f8)
GG(c, d, a, b, M_7, 14, 0x676f02d9)
GG(b, c, d, a, M_{12}, 20, 0x8d3a4c8a)

MD5 Continue…

Round 3:

- HH(a, b, c, d, M_5, 4, 0xfffa3942)
- HH(d, a, b, c, M_8, 11, 0x8771f681)
- HH(c, d, a, b, M_{11}, 16, 0x6d9d6122)
- HH(b, c, d, a, M_{14}, 23, 0xfde5380c)
- HH(a, b, c, d, M_1, 4, 0xa4beea44)
- HH(d, a, b, c, M_4, 11, 0x4bdecfa9)
- HH(c, d, a, b, M_7, 16, 0xf6bb4b60)
- HH(b, c, d, a, M_{10}, 23, 0xbebfbc70)
- HH(a, b, c, d, M_{13}, 4, 0x289b7ec6)
- HH(d, a, b, c, M_0, 11, 0xeaa127fa)
- HH(c, d, a, b, M_3, 16, 0xd4ef3085)
- HH(b, c, d, a, M_6, 23, 0x04881d05)
- HH(a, b, c, d, M_9, 4, 0xd9d4d039)
- HH(d, a, b, c, M_{12},11, 0xe6db99e5)
- HH(c, d, a, b, M_{15}, 16, 0x1fa27cf8)
- HH(b, c, d, a, M_2, 23, 0xc4ac5665)

MD5 Continue…

Round 4:

II(a, b, c, d, M_0, 6, 0xf4292244)
II(d, a, b, c, M_7, 10, 0x432aff97)
II(c, d, a, b, M_{14}, 15, 0xab9423a7)
II(b, c, d, a, M_5, 21, 0xfc93a039)
II(a, b, c, d, M_{12}, 6, 0x655b59c3)
II(d, a, b, c, M_3, 10, 0x8f0ccc92)
II(c, d, a, b, M_{10}, 15, 0xffeff47d)
II(b, c, d, a, M_1, 21, 0x85845dd1)
II(a, b, c, d, M_8, 6, 0x6fa87e4f)
II(d, a, b, c, M_{15}, 10, 0xfe2ce6e0)
II(c, d, a, b, M_6, 15, 0xa3014314)
II(b, c, d, a, M_{13}, 21, 0x4e0811a1)
II(a, b, c, d, M_4, 6, 0xf7537e82)
II(d, a, b, c, M_{11},10, 0xbd3af235)
II(c, d, a, b, M_2, 15, 0x2ad7d2bb)
II(b, c, d, a, M_9, 21, 0xeb86d391)

MD5 Continue…

- Those t_i were chosen as follows:
- In step i, t_i is the integer part of 2^{32}*abs(sin(i)), where i is in radians.
- After all of this a, b, c and d are added to A, B, C and D respectively and the algorithm continues with the next block of data.
- The final output is the concatenation of A, B, C and D.

Secure Hash Algorithm(SHA)

- NIST along with the NSA, designed the Secure Hash Algorithm (SHA) for use with the Digital Signature Standard.
- When a message of any length<264 bits is input, the SHA produces a 160-bit output called a message digest.
- The MD is then input to the DSA, which computes the signature for the message.
- Signing the MD rather than the message often improves the efficiency of the process, because the MD is usually smaller than the message.
- The same MD should be obtained by the verifier of the signature when the received version of the message is used as input to SHA.

SHA Continue…

- Description of SHA: SHA produces a 160-bit hash, where as MD5 produces 128-bit hash.
- The message is padded to make it a multiple of 512 bits long. Same as MD5: First append a one, then as many zeros as necessary to make it 64-bits short of a multiple of 512, and finally a 64-bit representation of the length of the message before padding.
- Five 32-bit variables are initialized as follows:

  A = 0x67452301

  B = 0xefcdab89

  C = 0x98badcfe

  D =0x10325476

  E = 0xc3d2e1f0

SHA Continue…

- The main loop of the algorithm then begins.
- It processes the message 512 bits at a time and continues for as many 512-bit blocks as are in the message.

- First the five variables are copied into different variables: a gets A, b gets B, c gets C, d gets D, and e gets E.

- The main loop has four rounds of 20 operations each (MD5 has four rounds of 16 operations each).

- Each operation performs a nonlinear function on three of a, b, c, d and e, and then does shifting and adding similar to MD5.

SHA Continue…

- SHA's set of nonlinear function is:

  $f_t(X, Y, Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$, for t=0 to 19

  $f_t(X, Y, Z) = X \oplus Y \oplus Z$, for t=20 to 39

  $f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$, for t=40 to 59

  $f_t(X, Y, Z) = X \oplus Y \oplus Z$, for t=60 to 79

- Four constants are used in the algorithm:

  K_t = 0x5a827999 for t = 0 to 19

  K_t = 0x6ed9eba1 for t = 20 to 39

  K_t = 0x8f1bbcdc for t = 40 to 59

  K_t = 0xca62c1d6 for t = 60 to 79

  These constants came from: 0x5a827999=$2^{1/2}/4$, 0x6ed9eba1=$3^{1/2}/4$, 0x8f1bbcdc=$5^{1/2}/4$ and 0xca62c1d6=$10^{1/2}/4$; all times 2^{32}.

SHA Continue…

- The message block is transformed from sixteen 32-bit words (M_0 to M_{15}) to eighty 32-bit words (W_0 to W_{79}) using the following algorithm:

  $W_t = M_t$, for t=0 to 15

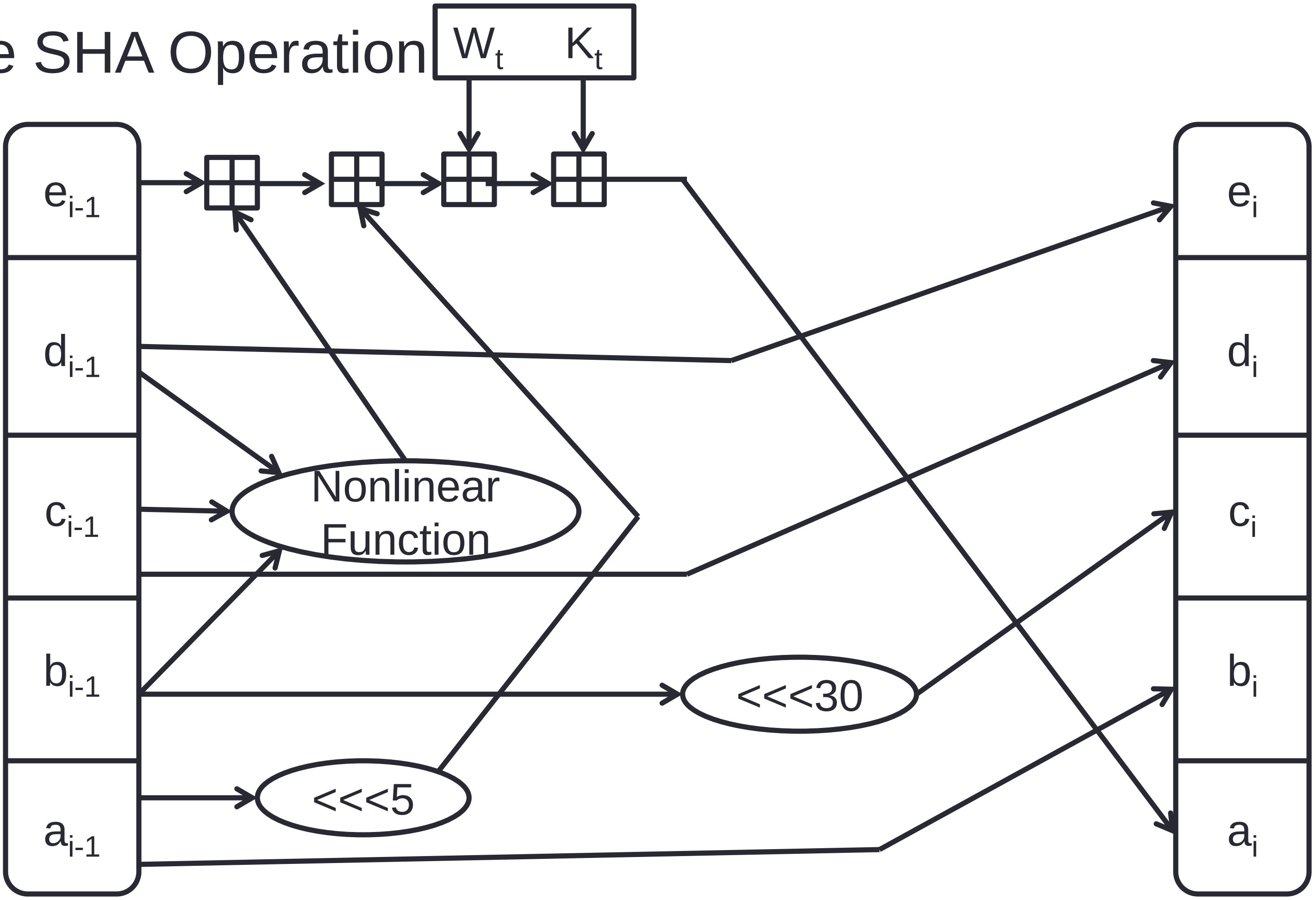
  $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$, for t=16 to 79.

- If t is the operation number (from 0 to 79), W_t represents the t^{th} sub-block of the expanded message, and <<<s represents a left circular shift of s bits, then the main loop look like:

```
For t=0 to 79
    TEMP = (a<<<5) + $f_t(b, c, d)$ + e + $W_t$ + $K_t$
    e = d
    d = c
    c = b<<<30
    b = a
    a = TEMP
```

SHA One Operation

One SHA Operation



SHA Continue…

- After all of this, a, b, c, d and e are added to A, B, C, D and E respectively and the algorithm continues with the next block (next 512 bits if any) of data.

- The final output is the concatenation of A, B, C, D and E.

CHAPTER 19

Public Key Algorithms

Background

- The concept of public-key cryptography was invented by Whitfield Diffie and Martin Hellman, and independently by Ralph Merkle.
- Only three algorithms work well for both encryption and digital signatures: RSA, ElGamal and Rabin.
- They encrypt and decrypt data much slowly than symmetric algorithms; usually that's too slow to support bulk data encryption.
- Hybrid cryptosystems speed things up: A symmetric algorithm with a random session key is used to encrypt the message, and a public key algorithm is used to encrypt the random session key.

Security of a public key Algorithm

- A cryptanalyst has access to the public key, so given $C=E_k(P)$, he can guess the value of M and easily check his guess.

- Public-key algorithms are designed to resist chosen-plaintext attacks; their security is based both on the difficulty of deducing the secret key from the public key and the difficulty of deducing the plaintext from ciphertext.

- In systems where the digital signature operation is the inverse of the encryption operation, this attack is impossible to prevent unless different keys are used for encryption and signature.

RSA

- Of all the public-key algorithms proposed over the years, RSA is by far the easiest to understand and implement.
- It is also most popular, named after three inventors – Ron Rivest, Adi Shamir and Leonard Adleman.
- RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits or higher) prime numbers.
- To generate the two keys, choose two random large prime numbers p and q, compute their product n=pq(for maximum security choose p and q of equal length).

RSA Continue…

- Then randomly choose the encryption key e, such that e and (p-1)(q-1) are relatively prime.

- The decryption key can be calculated (using the extended Euclidean algorithm) as follows:

  $ed \equiv 1 \bmod (p-1)(q-1)$ from that

  $d = e^{-1} \bmod (p-1)(q-1)$

- d and n are also relatively prime.

- The number e and n are the public key; d is the private key. (p and q are no longer needed; they should be discarded, but never revealed).

RSA Continue…

- To encrypt a message m, first divide it into numerical blocks smaller than n (with binary data, choose the largest power of 2 less than n).

- That is, if p and q are 100 digit primes, then n will has just under 200 digits, and each message block m_i, should be just under 200 digits long.

- The encrypted message c will be made up of similarly sized blocks c_i, of about the same length.

- The encryption formula is simply:

$$c_i = m_i^e \bmod n.$$

- The decryption formula is as follows:

$$m_i = c_i^d \bmod n.$$

Since $c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1} = m_i m_i^{k(p-1)(q-1)} = m_i*1 = m_i$; all (mod n).

The message could just as easily have been encrypted with d and decrypted with e; the choice is arbitrary.

RSA Continue…

- Consider p=47 and q=71 then n=pq=3337.

- The encryption key e must have no factors in common with (p-1)(q-1) = 46*70 = 3220.

- Choose e (at random) to be 79. In that case

  $d = 79^{-1} \bmod 3220 = 1019$.

- To encrypt m=6882326879666683 break it into six small block m_i as follows. We choose block size=3 digit because n has 4 digit.

  $m_1 = 688$
  $m_2 = 232$
  $m_3 = 687$
  $m_4 = 966$
  $m_5 = 668$
  $m_6 = 003$

RSA Continue…

- The first block is encrypted as

  $688^{79} \bmod 3337 = 1570 = c_1$

- Performing the same operation on the subsequent blocks generates an encrypted message:

  c = 1570 2756 2091 2276 2423 158

- Decrypting the message requires performing the same exponentiation using the decryption key 1019.

  $1570^{1019} \bmod 3337 = 688 = m_1$.

- The rest of the message can be recovered in this manner.

Speed of RSA

- RSA is about 100 times slower than DES.
- The fastest VLSI hardware implementation for RSA with 512-bit modulus has a throughput of 64 kilobits per second.
- In software, DES is about 100 times faster than RSA

Security of RSA

- The security of RSA depends wholly on the problem of factoring large numbers.

- Off course, it has never been mathematically proven that you need to factor n to calculate m from c and e.

- It is also possible to attack RSA by guessing the value of (p-1)(q-1). The attack is not easier than factoring n.

- Most common algorithms for computing primes p and q are probabilistic; what happen if p and q are composite? The odds are that encryption and decryption won’t work properly.

CHAPTER 20

Public-Key Digital Signature Algorithms

Digital Signature Algorithm(DSA)

- In August 1991, The National Institute of Standards and Technology (NIST) proposed the Digital Signature Algorithm (DSA) for use in their Digital Signature Standard (DSS)

- In all, NIST received 109 comments by the end of the first comment period on February 28, 1992.
- Lets look at the criticisms against DSA one by one.

1. DSA cannot be used for encryption or key distribution.

2. DSA was developed by the NSA, and there may be trapdoor in the algorithm.

DSA

3. DSA is slower than RSA.
4. RSA is a de facto standard.
5. The DSA selection process is not public; sufficient time for analysis has not been provided.
6. DSA may infringe no other patents.
7. The key size is too small.

Description of DSA

- DSA is a variant of the Schnorr and ElGamal Signature Algorithms. The algorithm uses the following parameters:

1. p = a prime number L bits long, where L ranges from 512 to 1024 and is a multiple of 64. (In the original standard, the size of p was fixed at 512 bits).
2. q = a 160-bit prime factor of p-1.
3. $g = h^{(p-1)/q} \bmod p$, where h is any number less than p-1 such that $h^{(p-1)/q} \bmod p$ is greater than 1.
4. x = a number less than q.
5. $y = g^{x} \bmod p$.

Description of DSA Continue…

- The algorithm also makes use of a one way hash function: $H(m)$.
- The parameters p, q and g are public and can be common across the network of users.
- The private key is x; the public key is y.

- To sign a message m:

1. Alice generates a random number k, less than q.
2. Alice generates:
   a. $r = (g^k \bmod p) \bmod q$.
   b. $s = (k^{-1}(H(m)+xr)) \bmod q$.

   The parameters r and s are her signature; she sends these to Bob.

3. Bob verifies the signature by computing.
   a. $w = s^{-1} \bmod q$.
   b. $u_1 = (H(m) * w) \bmod q$.
   c. $u_2 = (rw) \bmod q$.
   d. $v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$.

   If $v = r$, then the signature is verified.

CHAPTER 24

Example Implementation

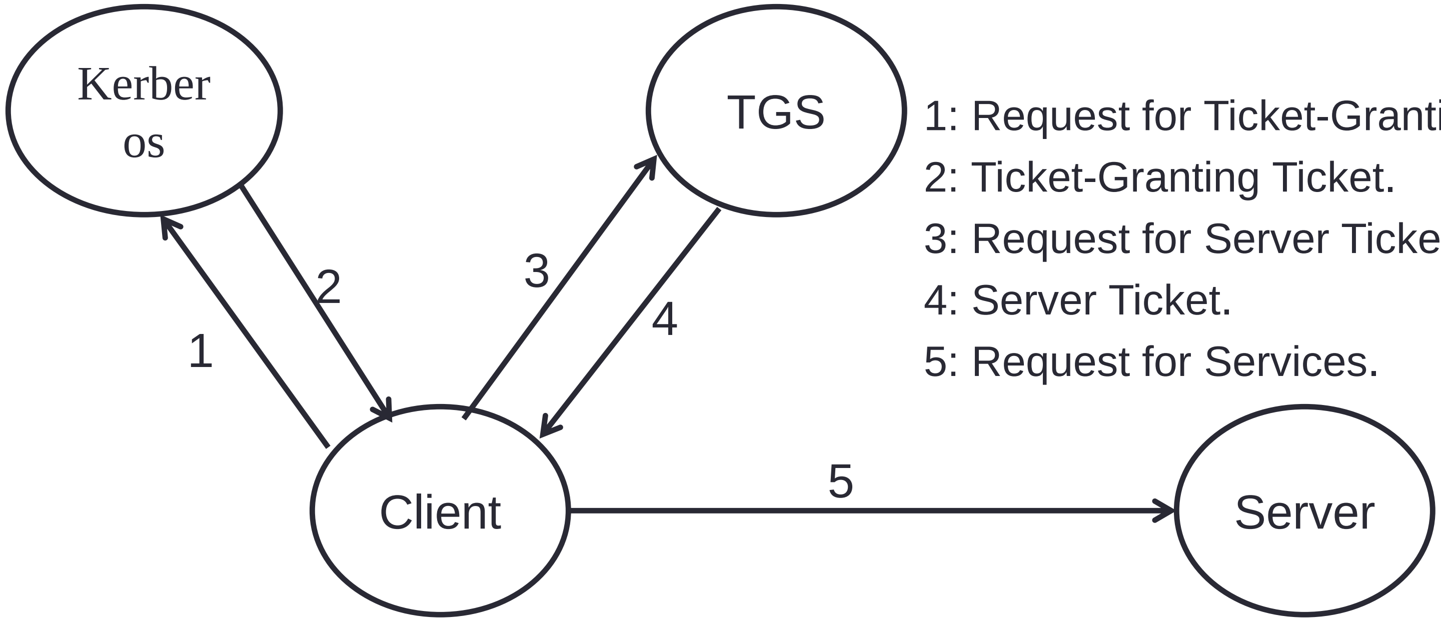
Kerberos

- Kerberos is a trusted third-party authentication protocol designed for TCP/IP network.
- A Kerberos service, sitting on the network, acts as a trusted arbitrator (বিচার/শালিশ).
- Kerberos is based on Symmetric Cryptography.
- It was originally developed at MIT for project Athena.

The Kerberos Model (2014)

- In the Kerberos model, there are entities- clients and servers- sitting on the network.
- Clients can be users, but can also be independent software programs that need to do things: download files, send messages, access databases, access printers, obtain administrative privileges, whatever.
- Kerberos keeps a database of clients and their secret keys. (For a human the secret key is an encrypted password).
- Kerberos also creates session keys which are given to a client and a server (or to two clients) and no one else.
- A session key is used to encrypt messages between two parties, after which it is destroyed.
- Kerberos v.4 provided a nonstandard mode for authentication. This mode is weak: It fails to detect certain changes to the ciphertext.
- Kerberos v.5 uses CBC mode.

Kerberos Authentication steps



1: Request for Ticket-Granting Ticket.
2: Ticket-Granting Ticket.
3: Request for Server Ticket.
4: Server Ticket.
5: Request for Services.

How Kerberos Works (2013)

- The Kerberos protocol is straightforward (as shown on the previous slide).
- A client requests a ticket for a Ticket-Granting Service (TGS) from Kerberos.
- The ticket is sent to the client encrypted with the client's secret key.
- To use a particular server, the client requests a ticket for that server from the TGS.
- Assuming everything is in order, the TGS sends the ticket back to the client.
- The client then presents this ticket to the server along with an authenticator.

Kerberos Table of Abbreviations

c	= client.
s	= server.
a	= client's network address.
v	= beginning and ending validity time for a ticket.
t	= timestamp.
K_x	= x's secret key.
$K_{x, y}$	= session key for x and y.
$\{m\} K_x$	= m encrypted in x's secret key.
$T_{x, y}$	= x's ticket to use y.
$A_{x, y}$	= authenticator from x to y.

Credentials (পরিচয়পত্র) (2013)

- Kerberos uses two types of credentials: tickets and authenticators.

- A ticket is used to pass securely to the server and it is the identitifier of the client for whom the ticket was issued.

- A Kerberos ticket takes this form:

  $$T_{c,\,s} = s, \{c, a, v, K_{c,\,s}\}K_s.$$

  It contains the servers name, client's name and network address, a timestamp and a session key. The information is encrypted with the server's secret key.

- Once the client gets this ticket , she can use it multiple times to access the server – until the ticket expires.

Credentials Continue…

- A Kerberos authenticator takes this form:

  $A_{c,\ s}$ = {c, t, key}$K_{c,\ s}$.

  The client generates it every time she wishes to use a service on the server. It contains the client’s name, a timestamp, and an optional additional session key, all encrypted with the session key shared between the client and the server.

- The authenticator serves two purposes:
  1. It contains some plaintext encrypted with the session key. This proves that it also knows the key.
  2. An eavesdropper who records both the ticket and the authenticator can’t replay them two days later.

Kerberos V.5 Messages

Kerberos V.5 has Five messages:

1. Client to Kerberos:	c, tgs
2. Kerberos to Client:	$\{K_{c, tgs}\} K_{c}, \quad \{T_{c, tgs}\}K_{tgs}.$
3. Client to TGS:	$\{A_{c, s}\} K_{c, tgs},\{T_{c, tgs}\}K_{tgs}.$
4. TGS to Client:	$\{K_{c, s}\} K_{c, tgs},\{T_{c, s}\}K_{s}.$
5. Client to Server:	$\{A_{c, s}\} K_{c, s}, \{T_{c, s}\}K_{s}.$

Getting Initial Ticket

- The client has one piece of information that proves her identity: her password.
- The client sends a message containing her name and the name of her TGS server to the Kerberos authentication server.
- The Kerberos authentication server looks up the client in the database and upon success Kerberos generates a session key to be used between her and the TGS.
- This is called Ticket Granting Ticket(TGT).
- Kerberos encrypt that session key with that client’s secret key.
- Then it creates a TGT for the client to authenticate herself to the TGS, and encrypt that in the TGS’s secret key.
- The Kerberos authentication server sends both of these encrypted messages back to the client.

Getting Initial Ticket Continue…

- The client now decrypts the first message and retrieves the session key.
- The secret key is a one way hash of her password, so a legitimate user will have no trouble doing this.
- If the user were an imposter (ছদ্মবেশী), he would not know the correct password and therefore could not decrypt the response from the Kerberos authentication server.
- The client saves the TGT and session key and erases the password and the one way hash.
- The client can now prove her identity to the TGS for the lifetime of the TGS.

Getting Server Tickets (2013)

- A client has to obtain a separate ticket for each service she wants to use.
- The TGS grants tickets for individual servers.
- When a clients needs a ticket that she does not already have, she sends a request to the TGS.
- Upon receiving the request, the TGS decrypts the TGT with his secret key.
- Then he uses the session key included in the TGT to decrypt the authenticator.
- The TGS responds to a valid request by returning a valid ticket for the client to present to the server.
- The TGS also creates a new session key for the client and the server, encrypted with the session key shared by the client and the TGS.

Requesting a Service

- Now the client is ready to authenticate herself to the server.
- She creates a message very similar to the one sent to the TGS.
- The client creates an authenticator consisting of her name and network address, and a timestamp, encrypted with the session key for her and the server that the TGS generated.
- The server decrypts and checks the ticket and the authenticator and also checks the client's address and the timestamp.
- If everything checks out, the server knows that according to Kerberos, the client is who she says she is.

Kerberos V.4 (2014)

In Kerberos V.4 the five messages looked like:

1. Client to Kerberos:	c, tgs
2. Kerberos to Client:	$\{K_{c,\ tgs}, \{T_{c,\ tgs}\}K_{tgs}\}\ K_c.$
3. Client to TGS:	$\{A_{c,\ s}\}\ K_{c,\ tgs}, \{T_{c,\ tgs}\}K_{tgs}, s.$
4. TGS to Client:	$\{K_{c,\ s}, \{T_{c,\ s}\}K_s\}\ K_{c,\ tgs}.$
5. Client to Server:	$\{A_{c,\ s}\}\ K_{c,\ s}, \{T_{c,\ s}\}K_s.$
	$T_{c,\ s} = \{s, c, a, v, 1, K_{c,\ s}\}K_s$
	$A_{c,\ s} = \{c, a, t\}K_{c,\ s}.$

Message 1, 3 and 5 are identical. The double encryption of the ticket in steps 2 and 4 has been removed in version 5.

Security of Kerberos (2014)

- Kerberos is vulnerable to password-guessing attacks.
- An intruder can collect tickets and they try to decrypt them.
- Remember that the average person does not usually choose good passwords.
- If Mallory collects enough tickets, his chances of recovering a password are good.
- Perhaps the most serious attack involves malicious software

CHAPTER 10

Key Management; Other public-key Cryptosystem

Key Management

- Several techniques have been proposed for the distribution of public keys.
  1. Public Announcement.
  2. Publicly Available Directory.
  3. Public-key authority.
  4. Public-key certificates.

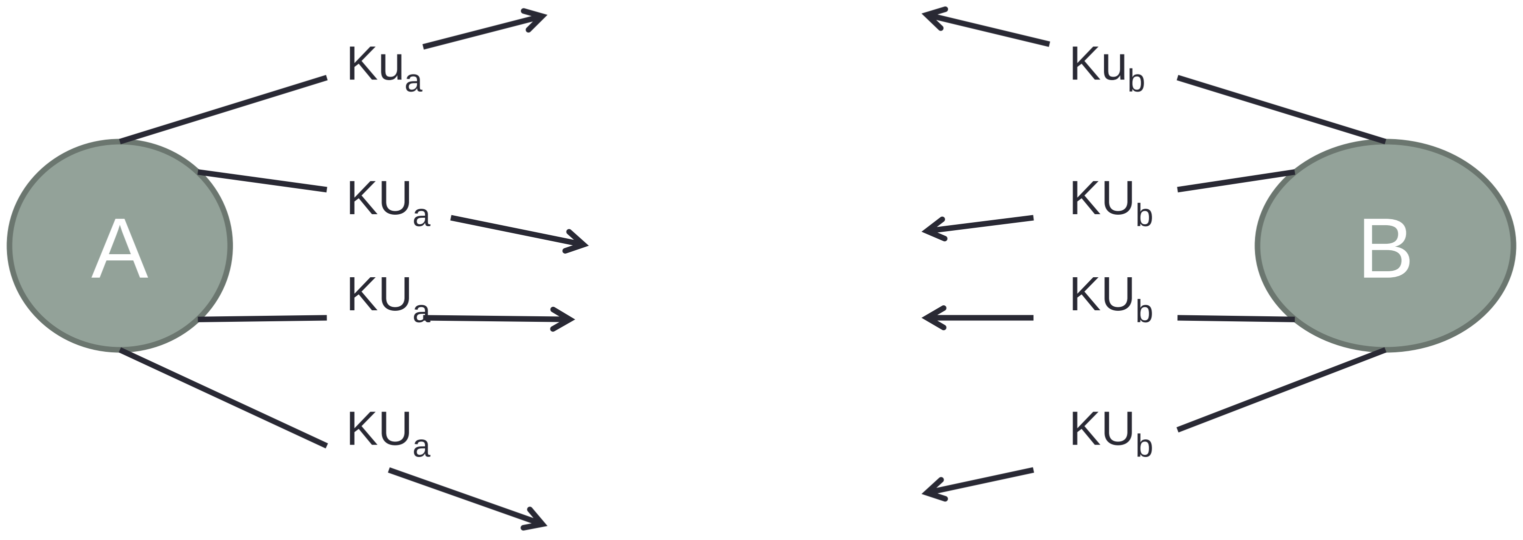
1. Public Announcement of public keys:
   a. Any participant can send his or her public key to any other participants or broadcast the key to the community at large.

   b. Many PGP (pretty good privacy) users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET, newsgroups and Internet mailing list.

   c. It has a major weakness; some user could pretend to be user A and send a public key to another participant or broadcast such a key.

Public Announcement of public keys:

Uncontrolled public key distribution:



Publicly Available Directory

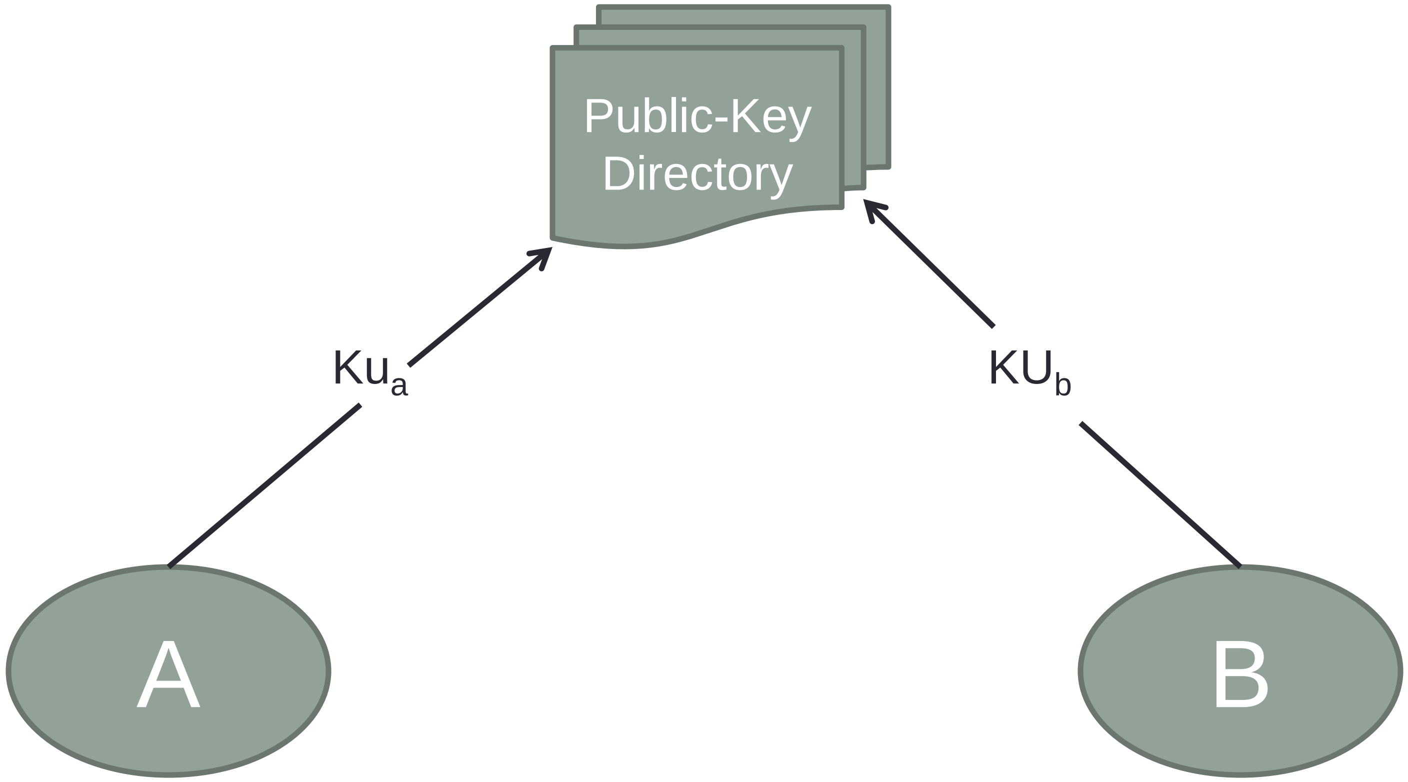
- Public key of each participants are stored on a directory which is maintained by some trusted entity or organization.
- Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Registration would have to be in person or by some form of secure authenticated communication.
3. A participants can update the key if needed (key compromised or after large data transfer).
4. Periodically, the authority publishes (hard copy version) the entire directory or updates.
5. Secure authenticated communication from the authority to the participant is mandatory.

It has still vulnerabilities if an opponent succeeds in obtaining or computing private key of the directory authority.

Publicly Available Directory

Public-Key Publication



Public-key Authority

- Stronger security for public-key distribution can be achieved by providing a tighter control over the distribution of public keys from the directory.

- It is assume that each participant reliably knows a public key for the authority.
  1. A sends a request with a timestamp to the authority to know the current public key of B.

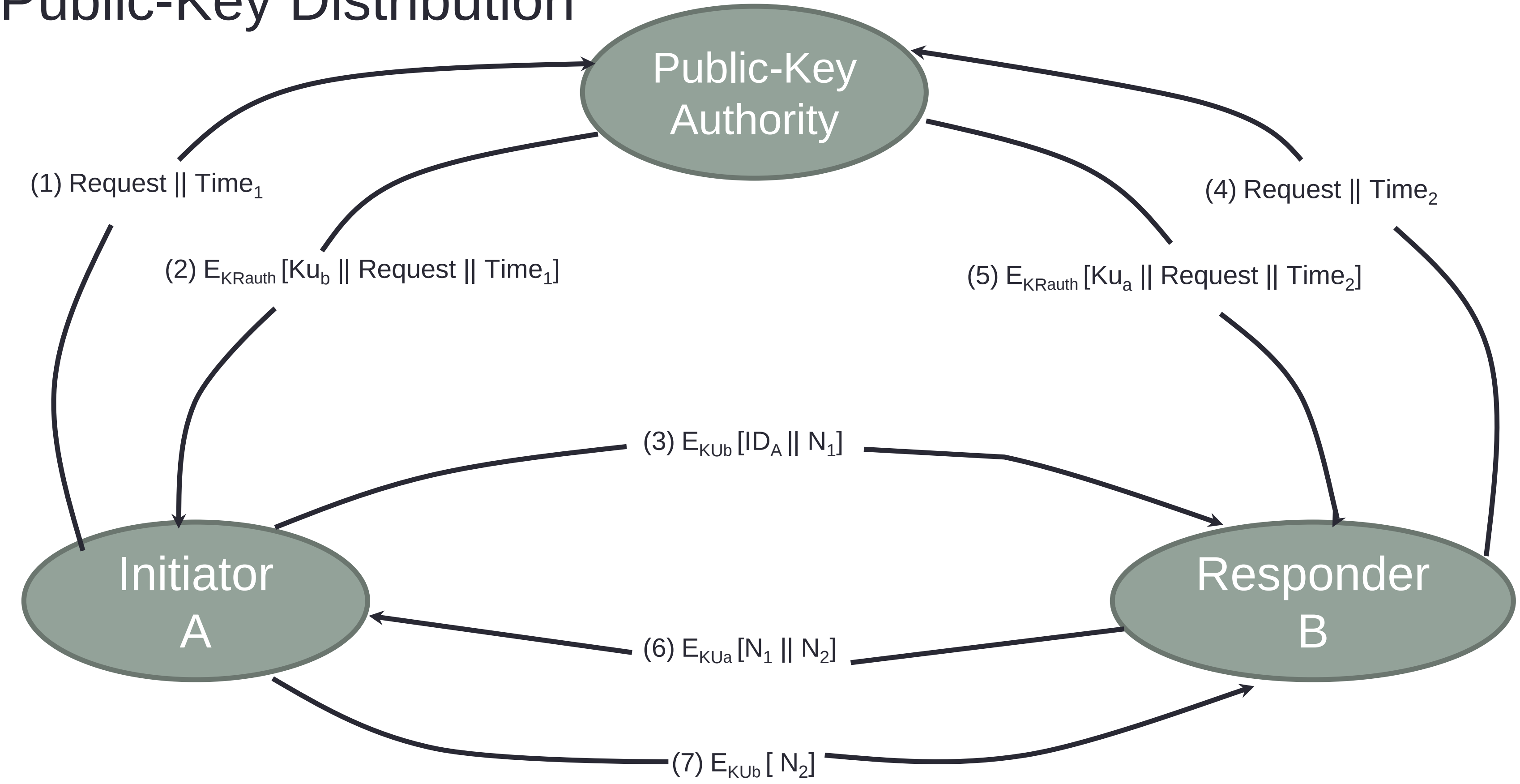
  2. The authority responds by encrypting a message with its own private key KR_{auth}. A can verify the response by decrypting the message with the public key of the authority. The message has:

     a. B's public key, KU_b.
     b. The original request, to enable A that the request was not altered.
     c. The original timestamp to ensure that it is the last request.

  3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1).

Public-key Authority

Public-Key Distribution


Public-Key Authority
Initiator A
Responder B
(1) Request || $Time_1$
(2) E_{KRauth} [Ku_b || Request || $Time_1$]
(3) E_{KUb} [ID_A || N_1]
(4) Request || $Time_2$
(5) E_{KRauth} [Ku_a || Request || $Time_2$]
(6) E_{KUa} [N_1 || N_2]
(7) E_{KUb} [N_2]

Public-key Authority Continue…

4,5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

At this point public key have been securely delivered to A and B, and they may begin their protected exchange. However the two additional steps are desirable:

6. B sends a message to A encrypted with KU_a and containing A's nonce (N_1) as well as a new nonce (N_2).

7. A returns N_2, encrypted using B's public key to assure B that its correspondent is A.

The initial four messages need be used infrequently because both A and B can save the other's public key for future use.

Public-Key Certificate

- Public-Key authority has some drawbacks:
  - The key authority could be bottleneck, since before any data transfer a sender (user) appeal to the authority for receiver(another user) a public key.
- An alternative approach is to use Certificates that can be used by participants to exchange keys without contacting a public key authority.
- Any participants can read a certificate to determine the name and public key of the certificate's owner.
- Any participants can verify that the certificate originated from the certificate authority and is not counterfeit.
- Only the certificate authority can create and update certificate.
- For participant A, the authority provides a certificate of the form.

$$C_A = E_{KRauth}\ [T, ID_A, KU_a]$$

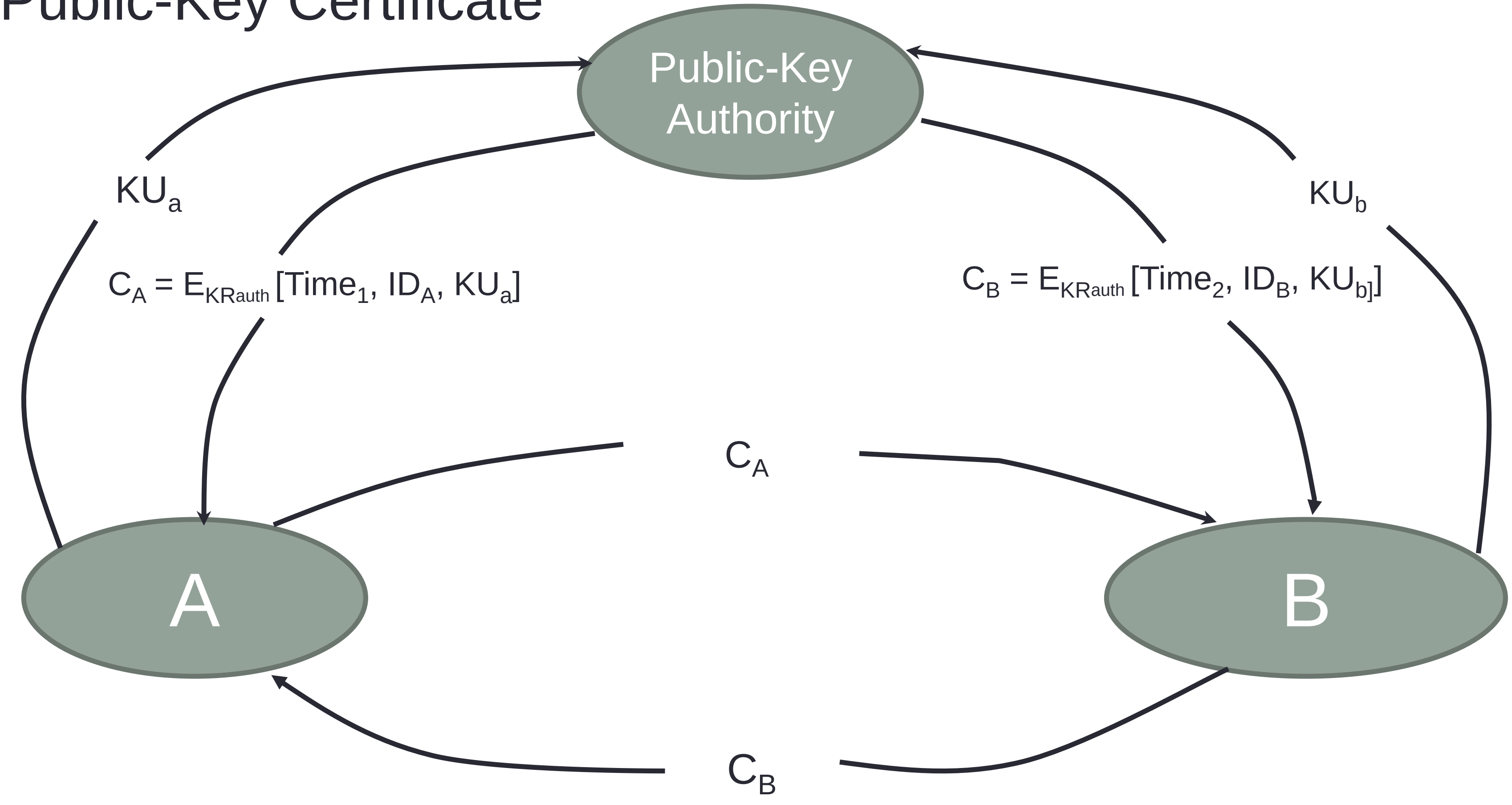
Where KR_{auth} is the private key used by the authority. A may then pass this certificate on to any other participant, who reads and verifies the certificates as follows:

$$D_{KUauth}\ [C_A] = D_{KUauth}\ [E_{KRauth}\ [T, ID_A, KU_a]] = (T, ID_A, KU_a)$$

- The recipient uses the authority's public key KU_{auth} to decrypt the certificate.

Public-Key Certificate

Public-Key Certificate


Public-Key Authority
A
B
KU_a
KU_b
$C_A = E_{KRauth}[Time_1, ID_A, KU_a]$
$C_B = E_{KRauth}[Time_2, ID_B, KU_b]$
C_A
C_B

Diffie-Hellman Key Exchange

- The algorithm is named after Diffie and Hellman.
- The purpose is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.

- First we define a primitive root of a prime number p as one whose powers generate all the integers from 1 to p-1. That is, if a is a primitive root of the prime number p, then the numbers

$$a \bmod p,\ a^2 \bmod p,\ \ldots,\ a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through p-1 in some permutation. Example: prime 13 has a primitive root of 2.

Primitive root Example

$2 \bmod 13 = 2$

$2^2 \bmod 13 = 4$

$2^3 \bmod 13 = 8$

$2^4 \bmod 13 = 3$

$2^5 \bmod 13 = 6$

$2^6 \bmod 13 = 12$

$2^7 \bmod 13 = 11$

$2^8 \bmod 13 = 9$

$2^9 \bmod 13 = 5$

$2^{10} \bmod 13 = 10$

$2^{11} \bmod 13 = 7$

$2^{12} \bmod 13 = 1$

Diffie-Hellman Key Exchange Continue…

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$b \equiv a^i \bmod p$ where $0 \leq i \leq (p-1)$

The exponent i is referred to as the discrete logarithm or index of b for the base a mod p. This value is denoted as $ind_{a,p}(b)$.

- Suppose q is a prime number and α is a primitive root of q.
- User A selects a random integer $X_A<q$ and computes $Y_A = \alpha^{X_A} \bmod q$.

- Similarly B selects a random integer $X_B<q$ and computes $Y_B = \alpha^{X_B} \bmod q$.
- Each side keeps X value as private and Y value as public.

- User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and B computes the key as $K=(Y_A)^{X_B} \bmod q$.

Diffie-Hellman Key Exchange Continue…

- These two calculations produce identical results:

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

$=(\alpha^{X_B} \bmod q)^{X_A} \bmod q$

$=(\alpha^{X_B})^{X_A} \bmod q$ by modular rules.

$=\alpha^{X_B X_A} \bmod q$

$=(\alpha^{X_A})^{X_B} \bmod q$

$=(\alpha^{X_A} \bmod q)^{X_B} \bmod q$

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

The attacker may compute $X_B = \mathrm{ind}_{\alpha,q}(Y_B)$ to get secret key of B.

The security of Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large prime the later task is infeasible.

Diffie-Hellman Key Exchange

- Suppose prime q=353, primitive root α =3, A’s secret key X_A=97 and B’s secret X_B=233.

  A computes $Y_A=3^{97} \bmod 353=40$

  B computes $Y_B=3^{233} \bmod 353=248$

After the exchange of public key, each can compute the common key as follows:

  A computes $K=(Y_B)^{X_A} \bmod 353= 248^{97} \bmod 353=160$

  B computes $K=(Y_A)^{X_B} \bmod 353=40^{233} \bmod 353=160$

CHAPTER 15

Pretty Good Privacy

Pretty Good Privacy

- PGP (Pretty Good Privacy) provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.

- It is the effort of a single person Phil Zimmerman.

- PGP has grown explosively and is now widely used because:
  1. It is available free worldwide in versions that run on a variety of platforms including: Windows, UNIX, Macintosh and many more.

  2. It is based on algorithms that are extremely secure. Specially; the package includes RSA, DSS and Diffie-Hellman for public key encryption; CAST-128, IDEA and 3DES for symmetric encryption and SHA-1 for hash coding.

  3. It has a wide range of applicability from corporations to individuals.

  4. It was not developed by, nor is it controlled by any governmental or standards organization.

  5. PGP is now on an Internet Standards track (RFC 3156).

Notation

- The following symbols are used:

K_S	= Session key used in Symmetric encryption scheme.
KR_a	= Private key of user A, used in public-key encryption scheme.
KU_a	= Public key of user A, used in public-key encryption scheme.
EP	= Public-key encryption.
DP	= Public-key decryption.
EC	= Symmetric encryption.
DC	= Symmetric decryption.
H	= Hash function.
$\|\|$	= Concatenation.
Z	= Compression using ZIP algorithm.
R64	= Conversion to radix 64 ASCII format.

Operational Description

- The actual operation of PGP consists of five services:
    1. Authentication
    2. Confidentiality
    3. Compression
    4. E-mail Compatibility and
    5. Segmentation.

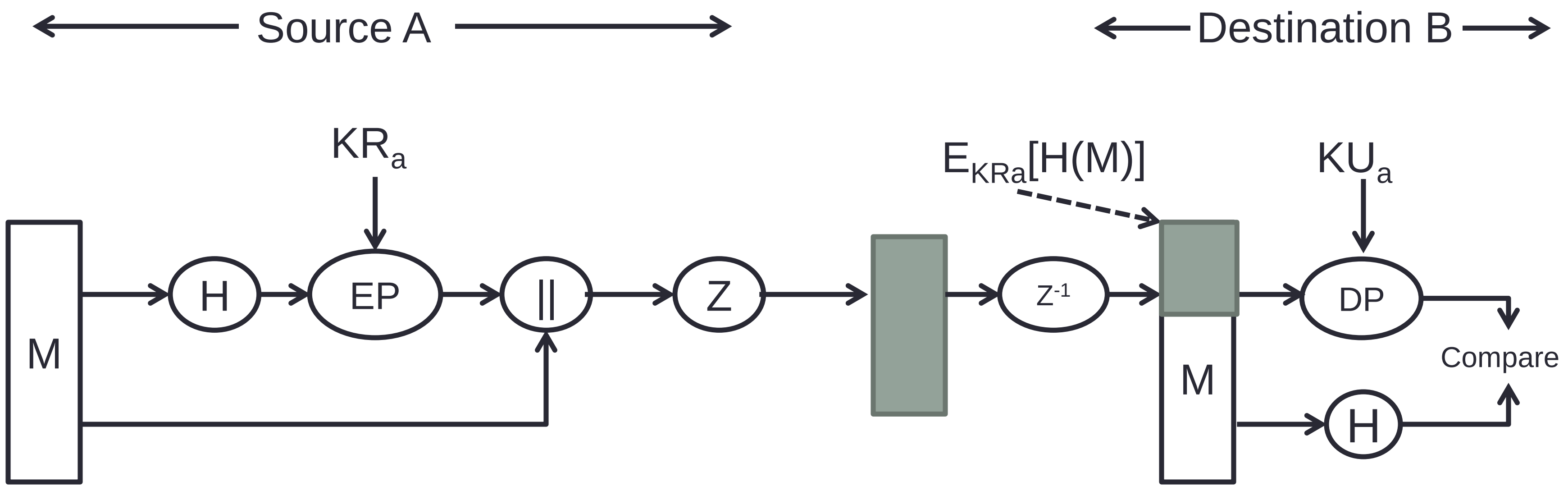
- Authentication:

Figure on the next slide illustrates the digital signature service provided by PGP. The sequence is as follows:

1. The Sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

Authentication

- The combination of SHA-1 and RSA provides an effective digital signature scheme.
- As an alternative, signatures can be generated using DSS/SHA-1.



Confidentiality

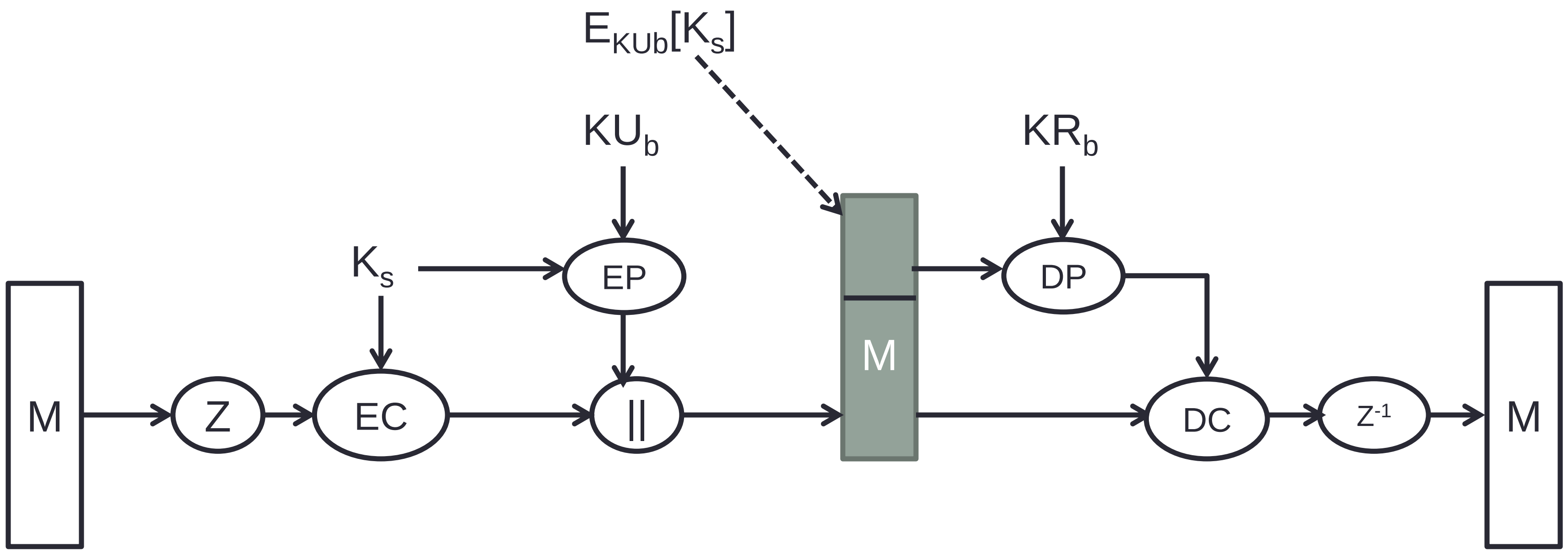
- Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files.
- In both cases the symmetric algorithm CAST-128 may be used.
- Alternatively, IDEA or 3DES may be used.
- In PGP each symmetric key is used only once. That is a new key is generated as a random 128-bit number for each message.
- The session key is bound to the message and transmitted with it.
- To protect the key, it is encrypted with the receiver’s public key.

Confidentiality Continue…

- The sequence can be described as follows:
  1. The sender generates a message and a random 128-bit number to used as a session key for this message only.
  2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
  3. The session key is encrypted with RSA, using the recipient’s public key and is prepended to the message.
  4. The receiver uses RSA with its private key to decrypt and recover the session key.
  5. The session key is used to decrypt the message.

Confidentiality Continue…

- Confidentiality only:

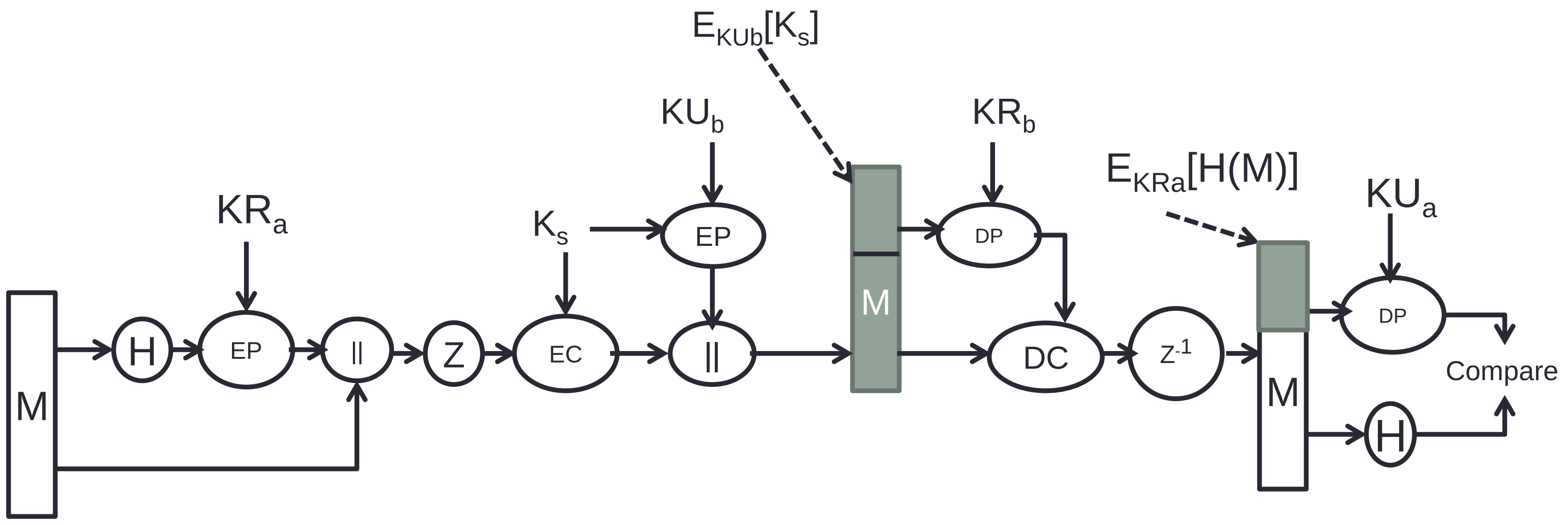


Confidentiality and Authentication

- Both Authentication and Confidentiality can be used for the same message.
- First, a signature is generated for the plaintext message and prepended to the message.
- The plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES) and the session key is encrypted using RSA (or ElGamal).
- In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and then encrypts the session key with the recipient's public key.

Confidentiality and Authentication Continue…

Confidentiality and Authentication:



Compression

- As a default, PGP compresses message after applying the signature but before encryption.
- This has the benefit of saving space both for e-mail transmission and for file storage.
- The signature is generated before compression for two reasons:
  a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.
  b. Even if one were willing to generate dynamically a recompressed message for verification, PGP’s compression algorithm presents a difficulty.
- Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-mail Compatibility

- When PGP is used, at least part of the block to be transmitted is encrypted.
- If only the signature service is used, then the message digest is encrypted.
- If the confidentiality service is used, the message plus signature (if present) are encrypted.

Segmentation and Reassembly

- E-mail facilities often are restricted to a maximum message length.
- Any message longer than the allowed maximum(typically 50,000 octets) must be broken up into smaller segments, each of which is mailed separately.
- PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail.
- The segmentation is done after all of the other processing, including the radix-64 conversion
- Thus the session key components and signature component appear only once, at the beginning of the first segment.

CHAPTER 16

IP Security

IP Security Overview

- The Internet Community has developed application specific security mechanisms in a number of application areas, including electronic mail (S/MIME, PGP), client/server (Kerberos), web access (Secure Socket Layer) and others.

- However users have some security concerns that cut across protocol layers. For example, an enterprise can run a secure, private TCP/IP network by disallowing links to untrusted sites, encrypting packets that leave the premises, and authenticating packets that enters the premises.

- By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but for the many security-ignorant applications.

IP Security Overview Continue…

- The IP-level security encompasses three functional areas:
    1. Authentication
    2. Confidentiality
    3. Key Management

The authentication mechanism assures that a received packet was in fact transmitted by the party identified as the source in the packet header.

The confidentiality facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties.

The key management facility is concerned with the secure exchange of keys.

IP Security Overview Continue…

- According to CERT (Computer Emergency Response Team) report, the most serious types of attacks included IP spoofing.
  - in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP.
  - and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents.

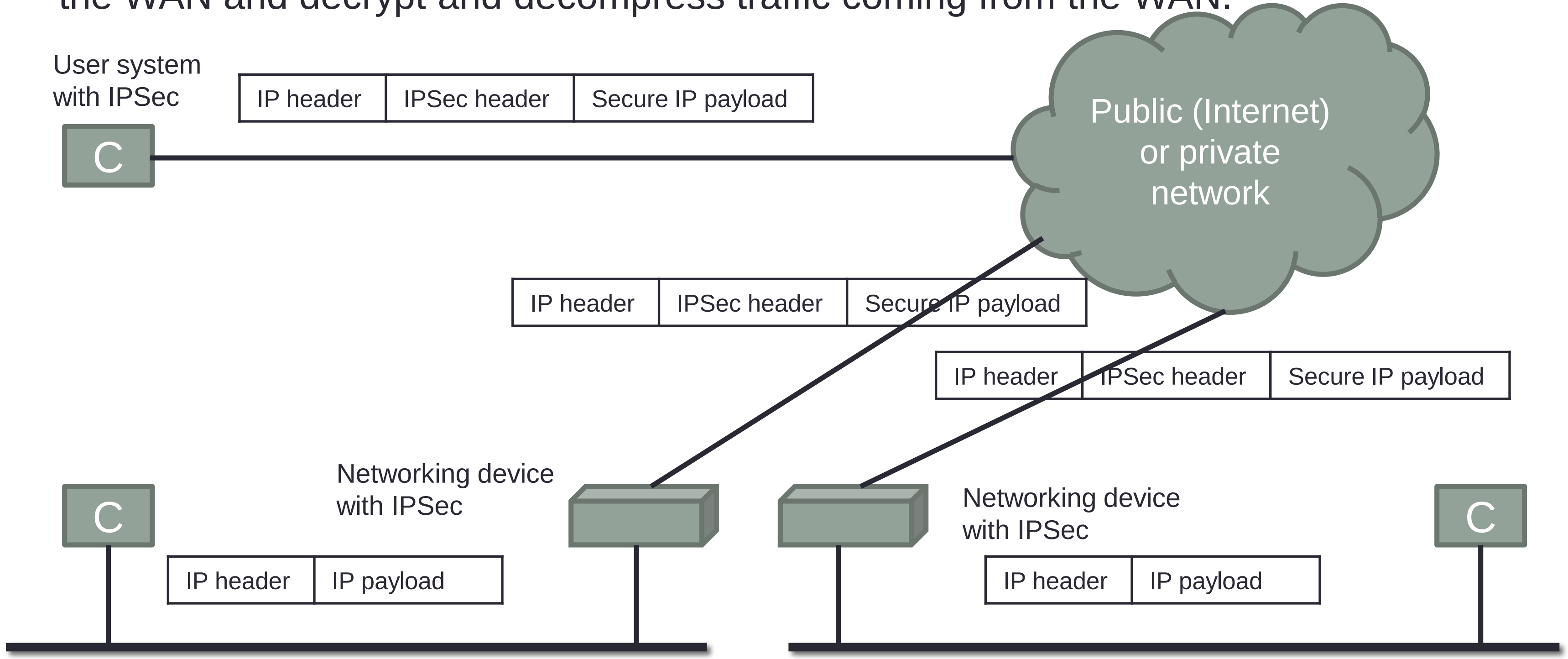
Application of IPSec

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following.

1. Secure branch office connectivity over the Internet:
   A company can build a secure virtual private network over the internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving cost and network management overhead.
2. Secure remote access over the Internet:
   An end user whose system is equipped with IP security protocols can make a local call to an ISP and gain a secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
3. Establishing extranet and intranet connectivity with partners:
   IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
4. Enhancing electronic commerce security:
   Electronic commerce applications have built-in security protocols, however the use of IPSec enhances that security.

Scenario of IPSec usage

- Figure below is a typical scenario of IPSec usage.
- IPSec protocols operate in networking devices, such as a router or a firewall, that connect each LAN to the outside world.
- The IPSec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN.



Benefits of IPSec

- When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter.
- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- IPSec can be transparent to end users.
- IPSec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

Routing Applications

IPSec play a vital role in the routing architecture. IPSec can assures that

1. A router advertisement (a new router advertises its presence) comes from an authorized router.
2. A neighbor advertisement (a router seeks to establish or maintain a neighbor relationship with a router in another routing domain) comes from an authorized router.
3. A redirect message comes from the router to which the initial packet was sent.
4. A routing update is not forged.

Without such security measures, an opponent can disrupt communications or divert some traffic.

CHAPTER 17

Web Security

Web Security Consideration

The world wide web is fundamentally a client/server application running over the Internet and TCP/IP intranets.

1. Unlike traditional publishing environments, the web is vulnerable to attacks on the web servers over the Internet.
2. Corporate reputations can be damaged and money can be lost if the web servers are subverted.
3. The underlying software may hide many potential security flaws.
4. Once the web server is subverted, an attacker may be able to gain access to data and systems not part of the web itself but connected to the server at the local site,
5. Common and untrained users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data. • Trojan horse browser. • Modification of memory. • Modification of message traffic in transit.	• Loss of information. • Compromise of machine • Vulnerability to all other threats.	Cryptographic checksums.
Confidentiality	• Eavesdropping on the net. • Theft of info from server. • Theft of data from client. • Info about network configuration. • Info about which client talks to server.	• Loss of information. • Loss of privacy.	Encryption, Web proxies.
Denial of Service	• Killing of user threads. • Flooding machine with bogus threats. • Filling up disk or memory. • Isolating machine by DNS attacks.	• Disruptive. • Annoying. • Prevent user from getting work done.	Difficult to prevent.
Authentication	• Impersonation of legitimate users. • Data forgery.	• Misrepresentation of user. • Belief that false information is valid.	Cryptographic techniques.

Web Security Threats

- Table on the previous slide shown summary of the types of security threats faced in using the web.
- These can be grouped as passive and active attacks.
- Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a web site that is supposed to be restricted.
- Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a web site.
- Another way to classify web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server.
- Issues of server and browser security is fall into the category of computer system security.
- Issues of traffic security fall into the category of network security.

Web Traffic Security Approaches

A number of approaches to providing web security are possible.

1. One way is to use IP security (figure a).
   a. The advantage is that It is transparent to end users and applications.

   b. Further, IPSec includes a filtering capability so that only selected traffic need to incur the overhead of IPSec processing.

2. Another is to implement security just above TCP (figure b).
   a. The foremost example of this approach is the Secure Socket Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS).

   b. SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications.

   c. Alternatively SSL could be embedded in specific packages.

   d. For example, Netscape and Microsoft explorer browsers come equipped with SSL, and most web servers have implemented the protocol.

Web Traffic Security Continue…

3. Application specific security services are embedded within the particular application (figure c).
   a. The advantage is that the service can be tailored to the specific needs of a given application.

   b. In the context of web security, an important example of this approach is Secure Electronic Transaction (SET).

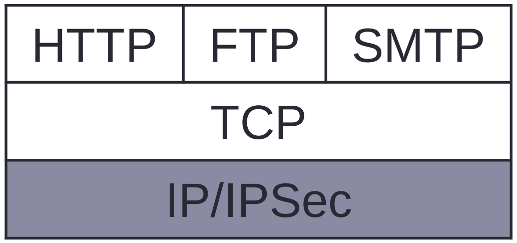


Figure a: Network level

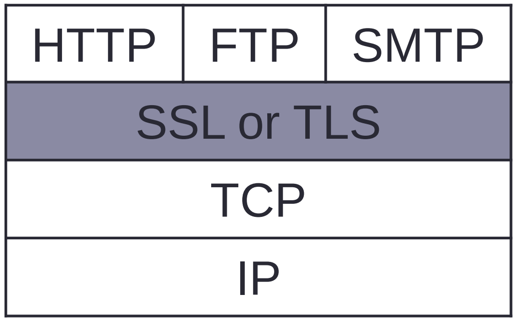


Figure b: Transport layer

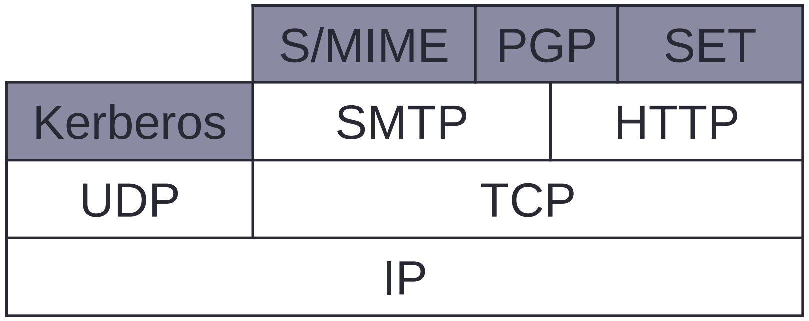


Figure c: Application level

CHAPTER 18

Intruders

Intruders

One of the most publicized threats to security is the intruder, generally referred to as hacker or cracker. Anderson identified three classes of intruders:

1. Masquerader:

   An individual who is not authorized to use the computer and who penetrates a system’s access controls to exploit a legitimate user’s account.

2. Misfeasor:

   A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.

3. Clandestine user:

   An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Intruders Continue…

- The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.
- Intruder attacks range from benign to the serious.
- At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there.
- At the serious intruders attempt to read privileged data, perform unauthorized modifications to data, or disrupt the system.

Intrusion Techniques

- The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system.

- With knowledge of some other user’s password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user.

- A system must maintain a file that associates a password with each authorized user.

- If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords.

Intrusion Techniques Continue…

The password file can be protected in one of two ways:

1. One-way encryption:

   The system stores only an encrypted form of the user’s password. When the user presents a password, the system encrypts that password and compares it with the stored value. The system generally performs a one-way transformation (not reversible) in which the password is used to generate a key for the encryption function.

2. Access control:

   Access to the password file is limited to one or a very few accounts.

If one or both of these measures are in place, some effort is needed for a potential intruder to learn passwords.

Intrusion Techniques Continue…

The following techniques can be used to learn passwords:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

2. Exhaustively try all short password (those of one to three characters)
3. Try words in the system's online dictionary or a list of likely passwords.

4. Collect information about users , such as their full names, the names of their spouse and children, pictures in their office, books are in their office that are related to hobbies.

5. Try users phone numbers, social security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.

7. Use a Trojan horse to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

Intrusion Techniques Continue…

- The first six methods are various ways of guessing a password. If an intruder has to verify the guess by attempting to log in, it is a tedious and easily countered means of attack.

- For example, a system can simply reject any login after three password attempts, thus requiring the intruder to reconnect to the host to try again.

- Under these circumstances, it is not practical to try more than a handful of passwords.

- The 7th method of attack can be particularly difficult to counter. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background it also contained code to copy the password file. Because the game was running under the operator’s high-privilege mode, it was able to gain access to the password file.

- The 8th attack is a matter of physical security. It can be countered with link encryption techniques.

CHAPTER 19

Malicious Software

Viruses and Related Threats

Malicious Programs:

- Slide 3 provides and overall taxonomy of software threats, or malicious programs.

- These threats can be categories into:
  1. Those that need a host program.
  2. Those that are independent.

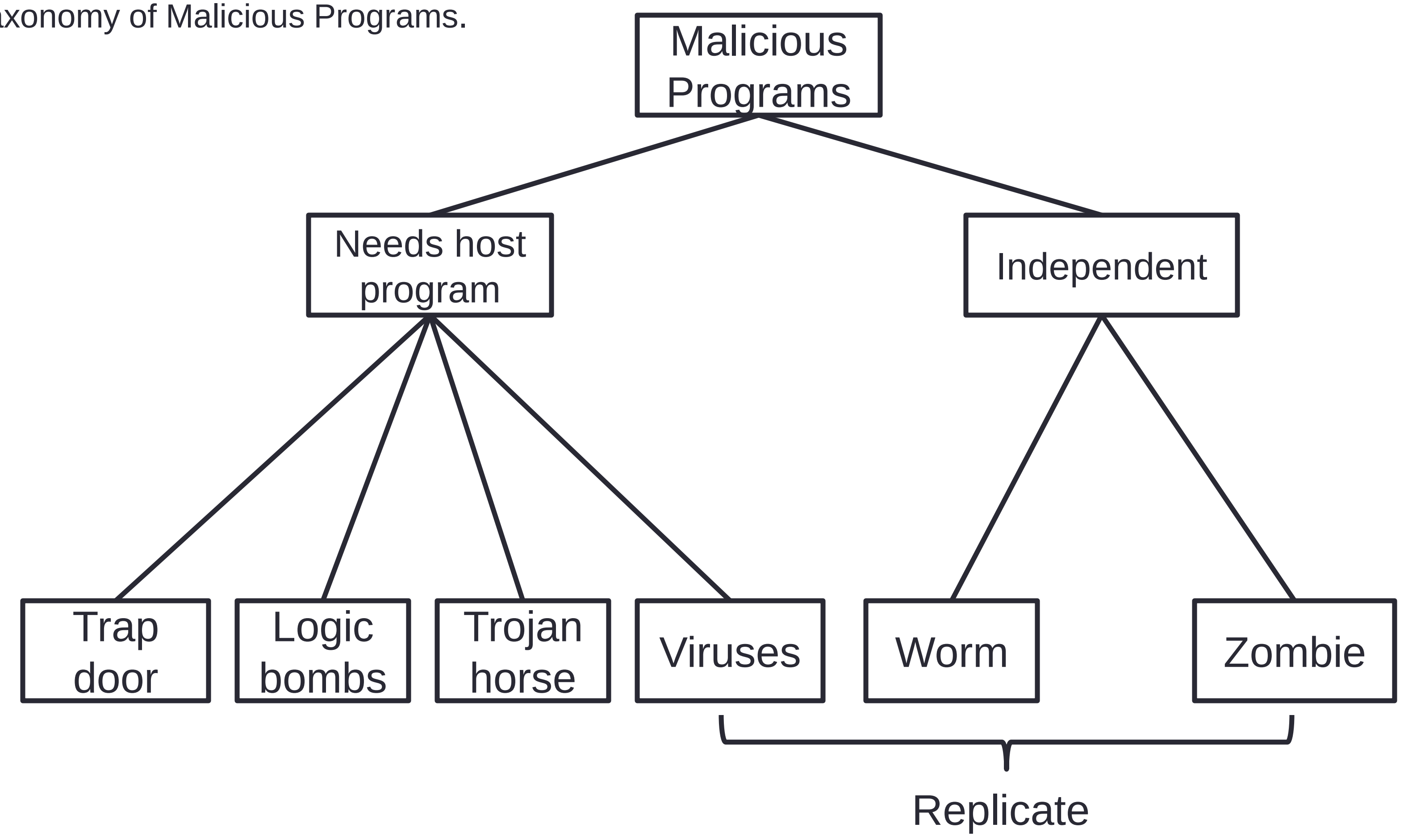
- These threats can also be categories into:
  1. Those that do not replicate.
  2. Those that do replicate.

  - The former are fragments of programs that are to be activated when the host program is invoked to perform a specific function.

  - The latter consist of either a program fragment (virus) or an independent program (worm, bacterium) that, when executed, may produce one or more copies of itself to be activated on the same system or some other system.

Taxonomy of Malicious Programs

Taxonomy of Malicious Programs.



Trap doors

- A trap door is a secret entry point into a program that allows someone to gain access without going through the usual security access procedure.

- The trap door is code that recognizes some special sequences of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.

- Trap doors become threats when they are used by unscrupulous programmers to gain unauthorized access.

Logic Bomb

- The logic bomb is code embedded in some legitimate program that is set to “explode” when certain conditions are met.

- Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application.

- Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

Trojan Horse

- A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
- Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.
- An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program.
- The code creates a trap door in the login program that permits the author to go on to the system using a special password.
- The Trojan horse can never be discovered by reading the source code of the login program.

Zombie

- A Zombie is a program that secretly takes over another Internet-attached computer and then uses that computer to launch attacks that are difficult to trace to the Zombie’s creator.

The Nature of Viruses

- A virus is a program that can infect other programs by modifying them.
- Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself.
- Lodged in a host computer, the typical virus takes temporary control of the computer's disk operating system.
- Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program.
- A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run.

Lifetime/Phases of Virus

During its lifetime, a typical virus goes through the following four phases.

1. Dormant phase:

   The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of a program/file. It is optional phase.

2. Propagation phase:

   The virus places an identical copy of itself into other programs or into certain system areas on the disk.

3. Triggering phase:

   The virus is activated to perform the function for which it was intended. As with dormant phase it can also be caused by a variety of system events.

4. Execution phase:

   The function is performed. The function my be harmless (a message on the screen), or damaging (destruction of programs and data files).

Types of Viruses

There has been a continuous arms race between virus writers and writers of antivirus software since viruses first appeared.

1. Parasitic virus:

   Attaches itself to executable files and replicates when the infected program is executed.

2. Memory-resident virus:

   Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

3. Boot-sector virus:

   Infects a master boot record or boot record and spreads when a system is booted from the disk containing virus.

4. Stealth virus:

   This type of virus hide itself from detection by antivirus software.

5. Polymorphic virus:

   A virus that mutates with every infection, making detection by the “signature: of the virus impossible

Worms

- A worm actively seeks out more machines to infect and each machine that is infected serves as an automated launching pad for attacks on other machines.

- To replicate itself, a network worm uses some sort of network vehicle. Example include:

1. Electronic mail facility:

   A worm mails a copy of itself to another systems.

2. Remote execution capability:

   A worm executes a copy of itself on another system.

3. Remote login capability:

   Logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

CHAPTER 20

Firewalls

Firewall Design Principles

- The firewall is inserted between the premises network and the Internet to establish a controlled link and to establish an outer security wall or perimeter.
- The aim of this perimeter is to protect the premises network from Internet-based attacks and to provide a single choke point where security and audit can be imposed.
- The firewall may be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

Firewall Characteristics

The design goals for a firewall are as follows: **(2014)**

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall.

2. Only authorized traffic (defined by the local security policy) will be allowed to pass.

3. The firewall itself is immune to penetration. This implies that use of a trusted system with secure operating system.

Firewall Characteristics Continue…

Four general techniques to control access and enforce the site's security policy:

1. Service control:

   Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number.

2. Direction control:

   Determines the direction in which particular service requests may be initiated and allowed to flow.

3. User control:

   Controls access to a service according to which user is attempting to access it.

4. Behavior control:

   Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam or it may enable external access to only a portion of the information on a local web server.

Firewall Characteristics Continue…

What one can expect from a firewall? **(2012)**

1. A firewall defines a single choke point that
   - keeps unauthorized users out of the protected network.
   - Prohibits vulnerable services from entering or leaving the network.
   - Provides protection from various kinds of IP spoofing and routing attacks.

2. A firewall provides a location for monitoring security-related events.

3. A firewall acts as a network address translator and perform a network management function that audits or logs Internet usage.

4. A firewall can serve as the platform for IPSec.

Firewall Characteristics Continue…

Firewalls have their limitations, including the following: (2015)

1. The firewall cannot protect against attacks that bypass the firewall.

2. The firewall does not protect against internal threats such as an employee who wittingly cooperates with an external attacker.

3. The firewall cannot protect against the transfer of virus-infected programs or files (it is impossible for the firewall to scan all incoming files, email, and messages for viruses).

Types of Firewalls

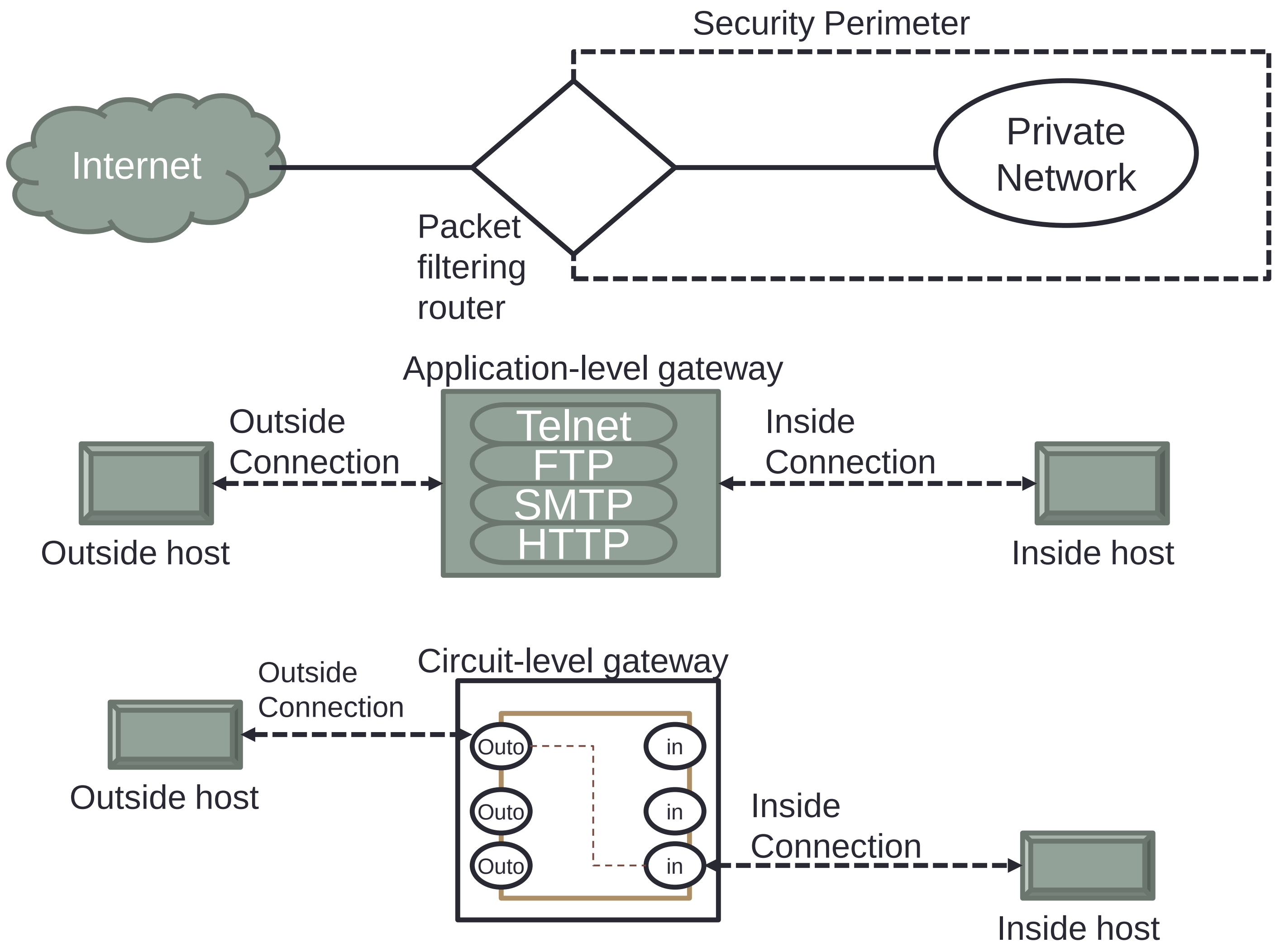
- Figure on the next slide illustrates the three common types of firewalls:
    1. **Packet filters.**
    2. **Application-level gateways. And**
    3. **Circuit-level gateways.**

1. **Packet-Filtering Router: (2010)**

   A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. Filtering rules are based on information contained in a network packet:

- Source IP address: IP (ex 192.168.1.1) of the originated system.
- Destination IP address: IP (ex 192.168.1.2) of the system the IP packet is trying to reach.
- Source and destination transport-level address:
- IP protocol field: Defines the transport protocol.
- Interface: for a router with 3 or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for.

Figure: Firewall Types



Application-Level Gateway (2006,2007)

- Also called a proxy server, acts as a relay of application-level traffic.
- The user contacts the gateway using a TCP/IP application such as Telnet or FTP, and the gateway asks the user for the names of the remote host to be accessed.
- When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.
- If the gateway does implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.
- Application-level gateway tend to be more secure than packet filters.
- The application-level gateway need only scan a few allowable applications.
- It is easy to log and audit all incoming traffic at the application level.
- A prime disadvantage is the additional processing overhead on each connection.

Circuit-Level Gateway

- A circuit-level gateway does not support end-to-end TCP connection.

- Rather the gateway sets up two TCP connections,
  1. one between itself and a TCP user on an inner host and
  2. One between itself and a TCP user on an outside host.

- A typical use of a circuit-level gateways is a situation in which the system administrator trusts the internal users.