Aspects of Security and Authentication-State-of-the-Art

Prof.(Dr.) J. K. Mandal

Ex-Dean, Faculty of Engineering, Technology & Management

Professor, Department of Computer Science & Engineering

University of Kalyani

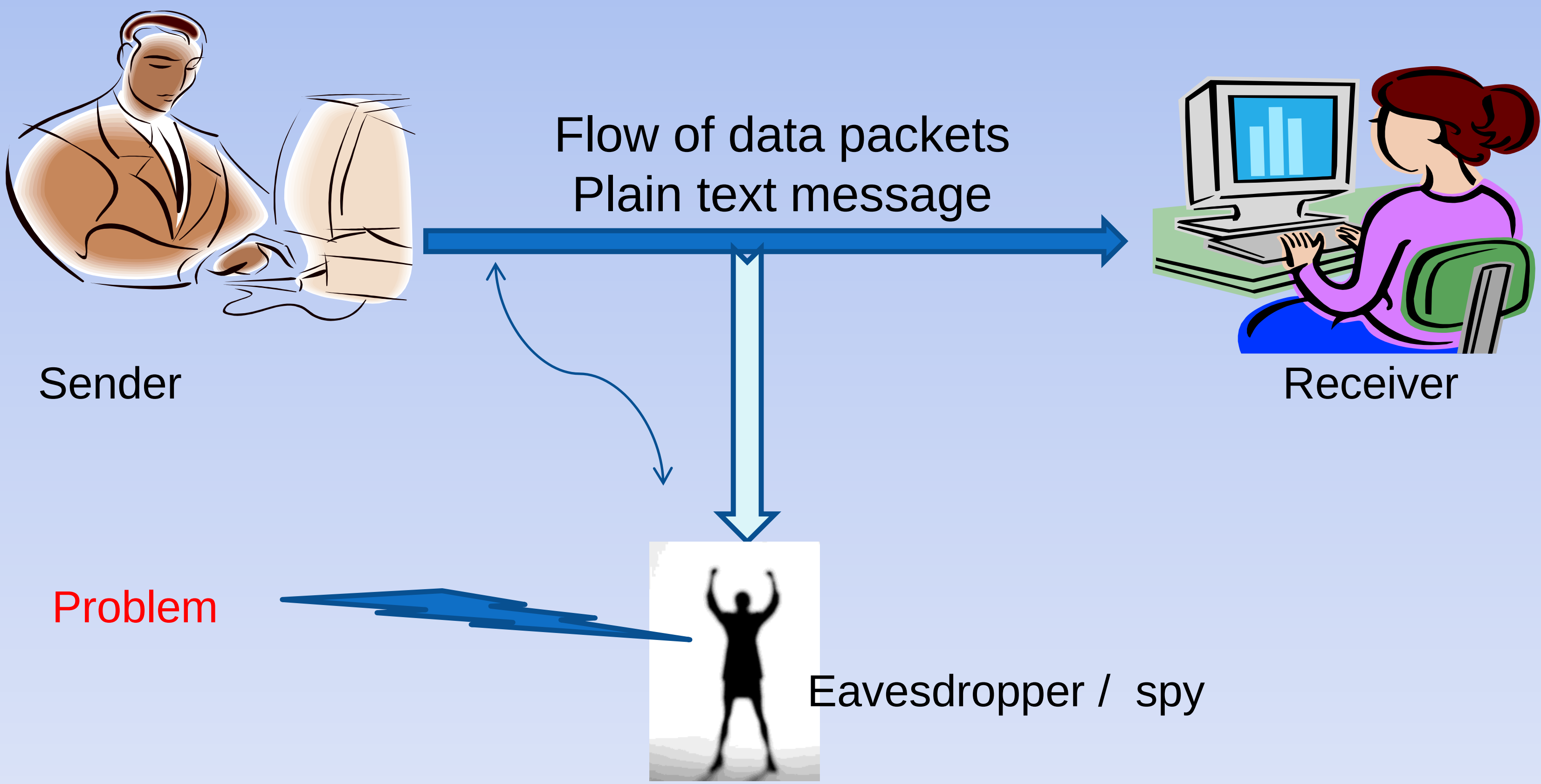
Kalyani, Nadia, West Bengal

E-mail: jkmandal@klyuniv.ac.in, jkm.cse@gmail.com

Web:http://jkmandal.com;Mobile:91 9434352214

COMMUNICATION

Communication Through Network



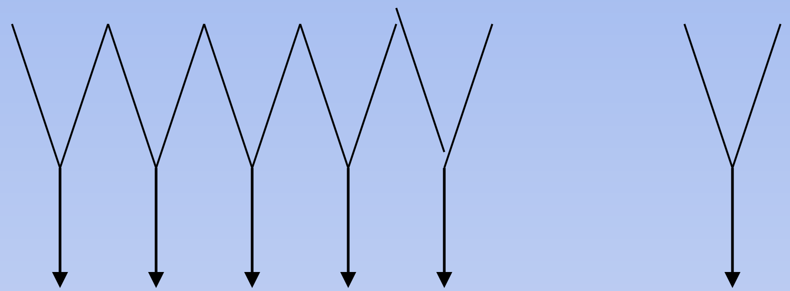
Plain text to Cipher text

- Substitution Techniques
  - Caesar Cipher
  - Mono-alphabetic Cipher
  - Homophonic Substitution Cipher
  - Playfair Cipher...............
- Transposition Techniques
  - Rail Fence Technique
  - Vernam Cipher(One Time Pad)
  - Book Cipher/ Running key cipher.............

Encryption
Decryption
Technique...

TRIANGULARISATION(XNOR)

$S^j = s^j_0 \quad s^j_1 \quad s^j_2 \quad s^j_3 \quad s^j_4 \quad s^j_5 \quad \ldots \quad s^j_{n-(j+2)} s^j_{n-(j+1)}$



$S^{j+1} = s^{j+1}_0 \, s^{j+2}_1 s^{j+3} \, s^{j+4}_3 s^{j+5}_4 \ldots \qquad s^j_{n-(j+2)}$

Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$S^0{}_0\ S^1{}_0\ S^2{}_0\ S^3{}_3\ S^4{}_0 \ldots S^{n-2}{}_0\ S^{n-1}{}_0$	Taking all the MSBs starting from the source block till the last block generated
010	$S^{n-1}{}_0\ S^{n-2}{}_0\ S^{n-3}{}_0\ S^{n-4}{}_0\ S^{n-5}{}_0 \ldots S^1{}_0\ S^0{}_0$	Taking all the MSBs starting from the last block generated till the source block
011	$S^0{}_{n-1}\ S^1{}_{n-2}\ S^2{}_{n-3}\ S^3{}_{n-4}\ S^4{}_{n-5} \ldots S^{n-2}{}_1\ S^{n-1}{}_0$	Taking all the LSBs starting from the source block till the last block generated
100	$S^{n-1}{}_0\ S^{n-2}{}_1\ S^{n-3}{}_2\ S^{n-4}{}_3\ S^{n-5}{}_4 \ldots S^1{}_{n-2}\ S^0{}_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

Different Target Blocks generated using TE for S = 10010101

Source Block S	Target Block Corresponding to Serial No.	Target Block T
10010101	001	10010101
	010	10101001
	011	10111101
	100	10111101

1 0 0

0 1

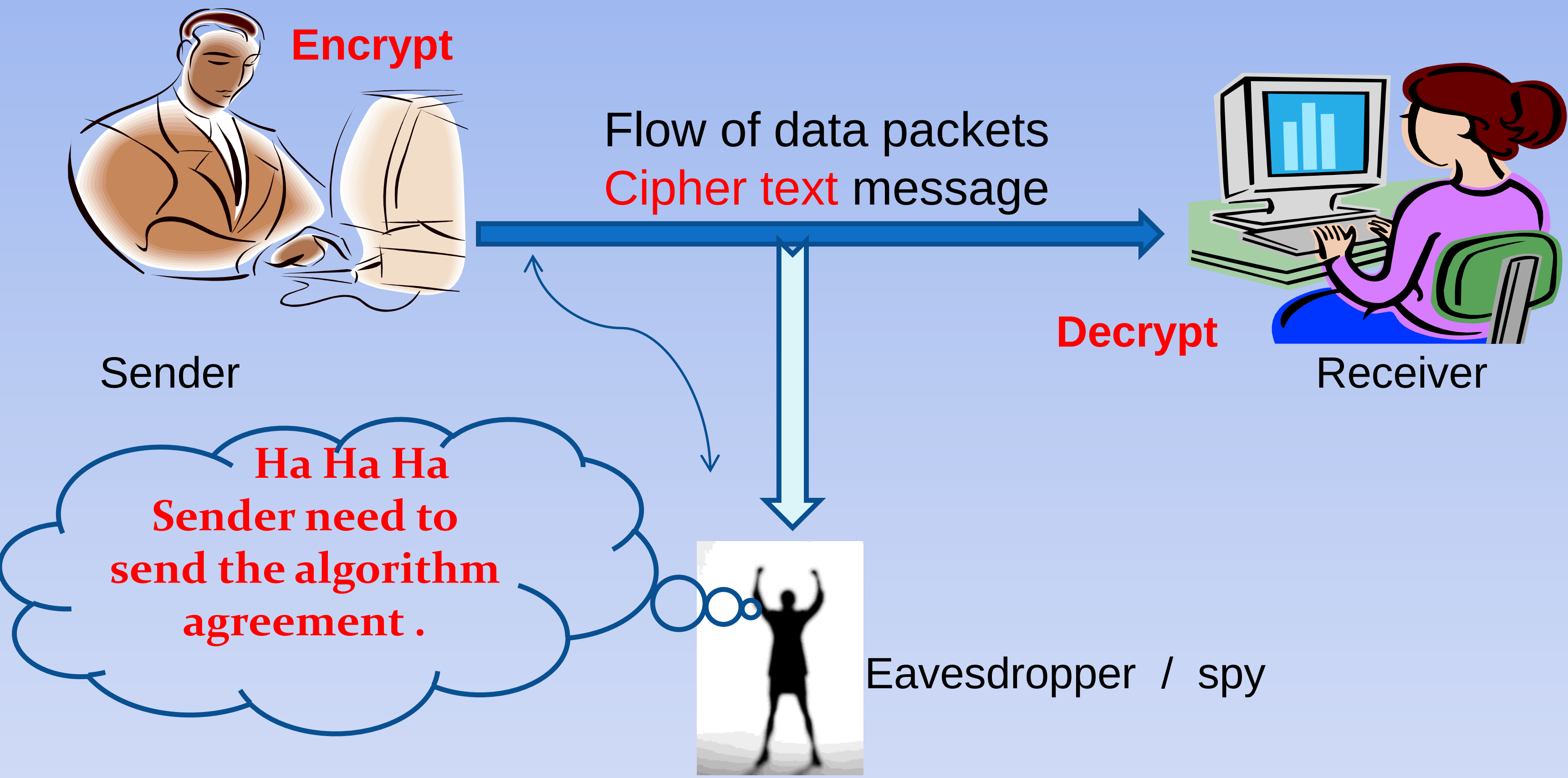
0

0 1 0

0 0

1

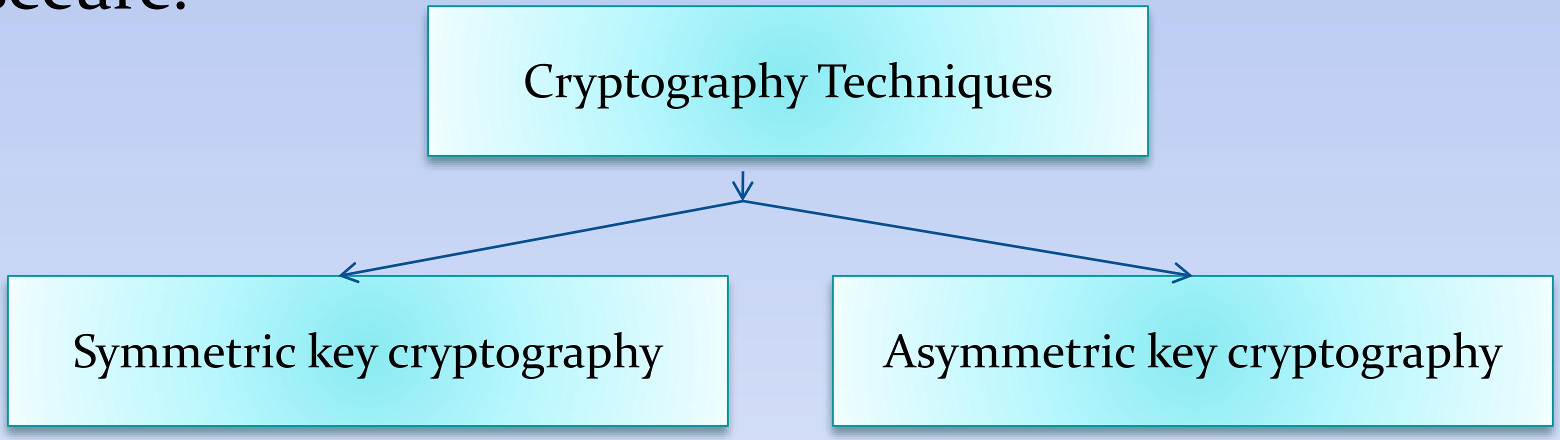
Communication…….



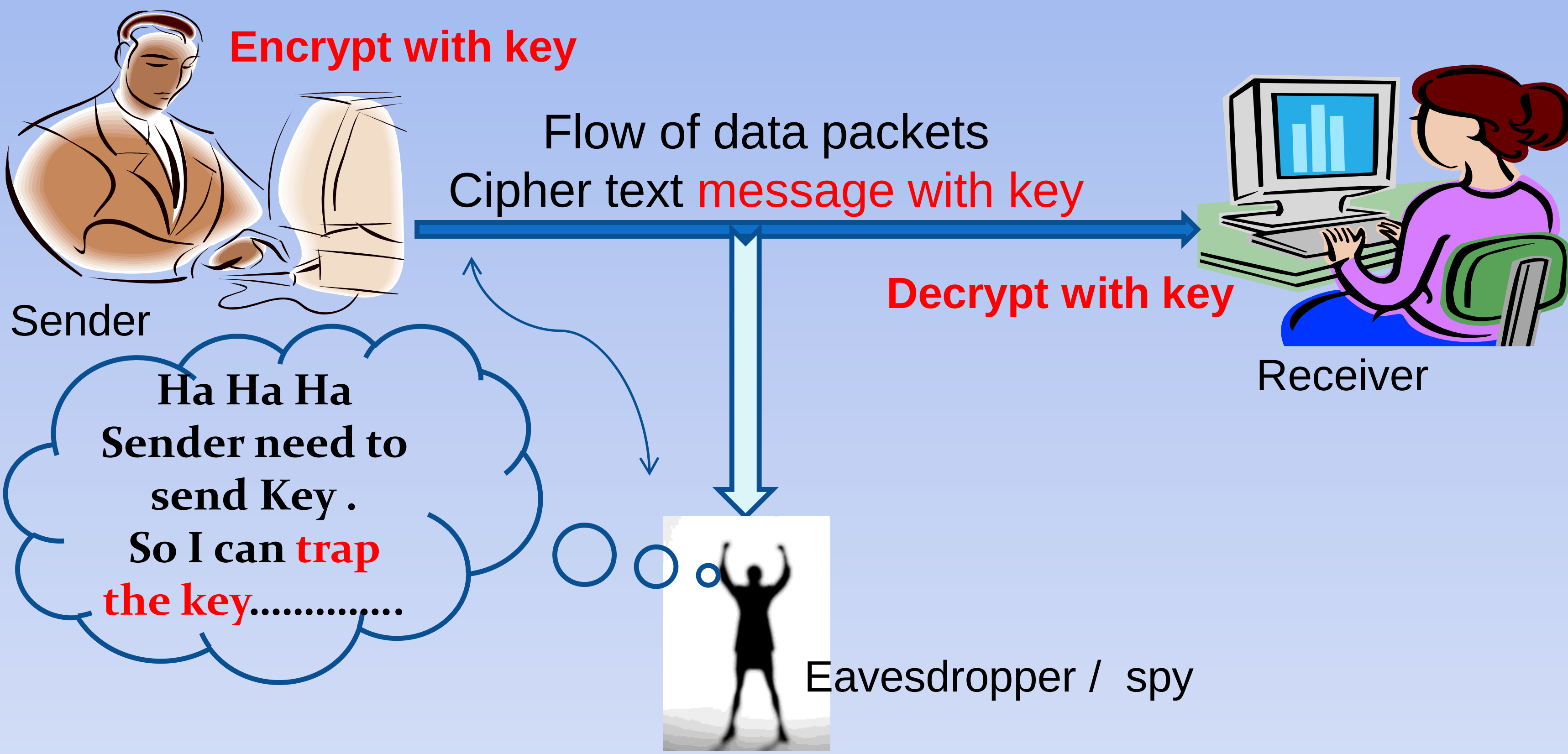
Note:- The decryption algorithm must be the same as the encryption algorithm. Otherwise decryption would not be able to retrieve the original message.

Cryptography

In general , the algorithm used for encryption and decryption process is usually known to everybody. However, it is the key used for encryption and decryption that makes the process of cryptography secure.



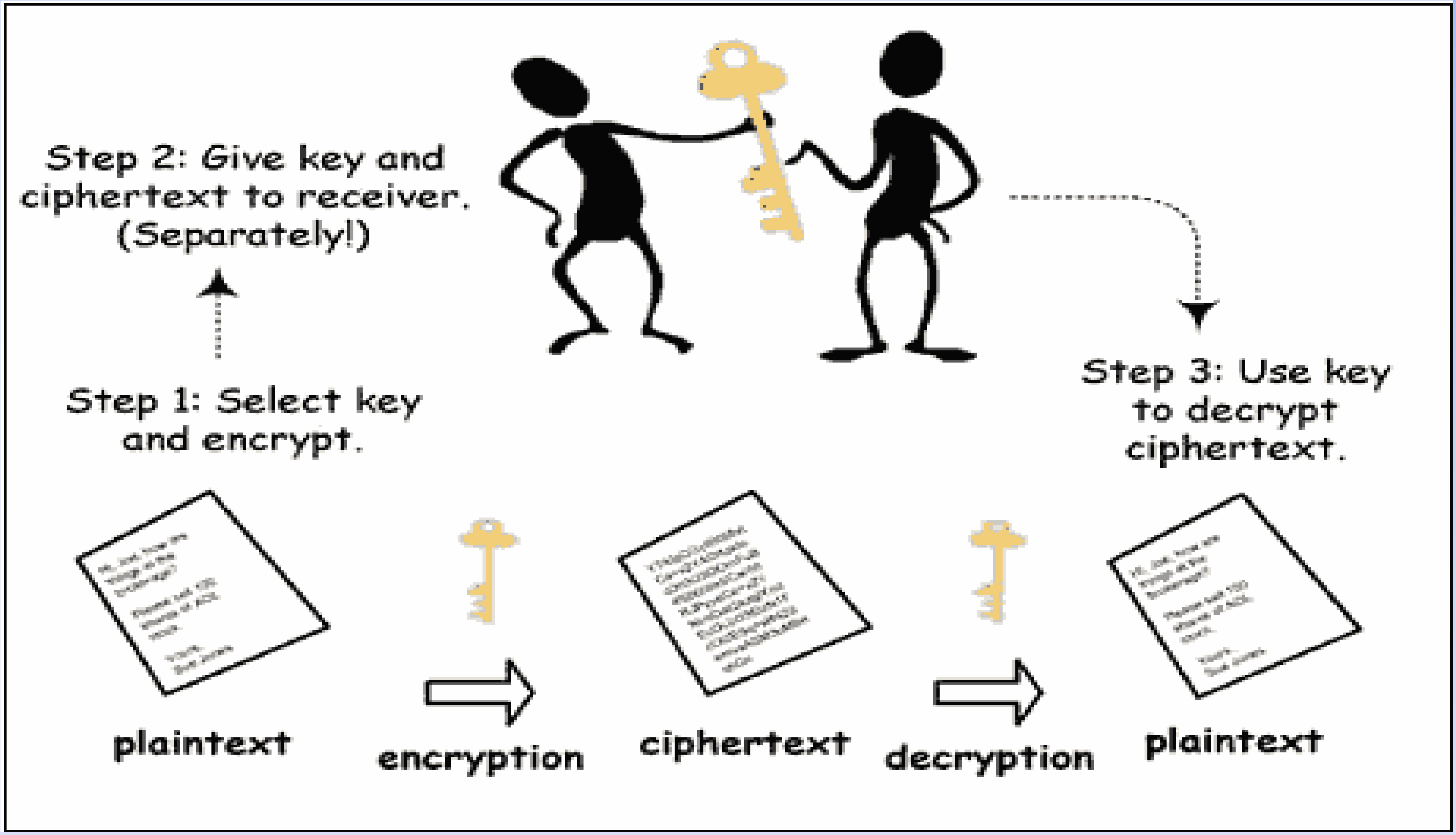
Communication…….
With the concept of key



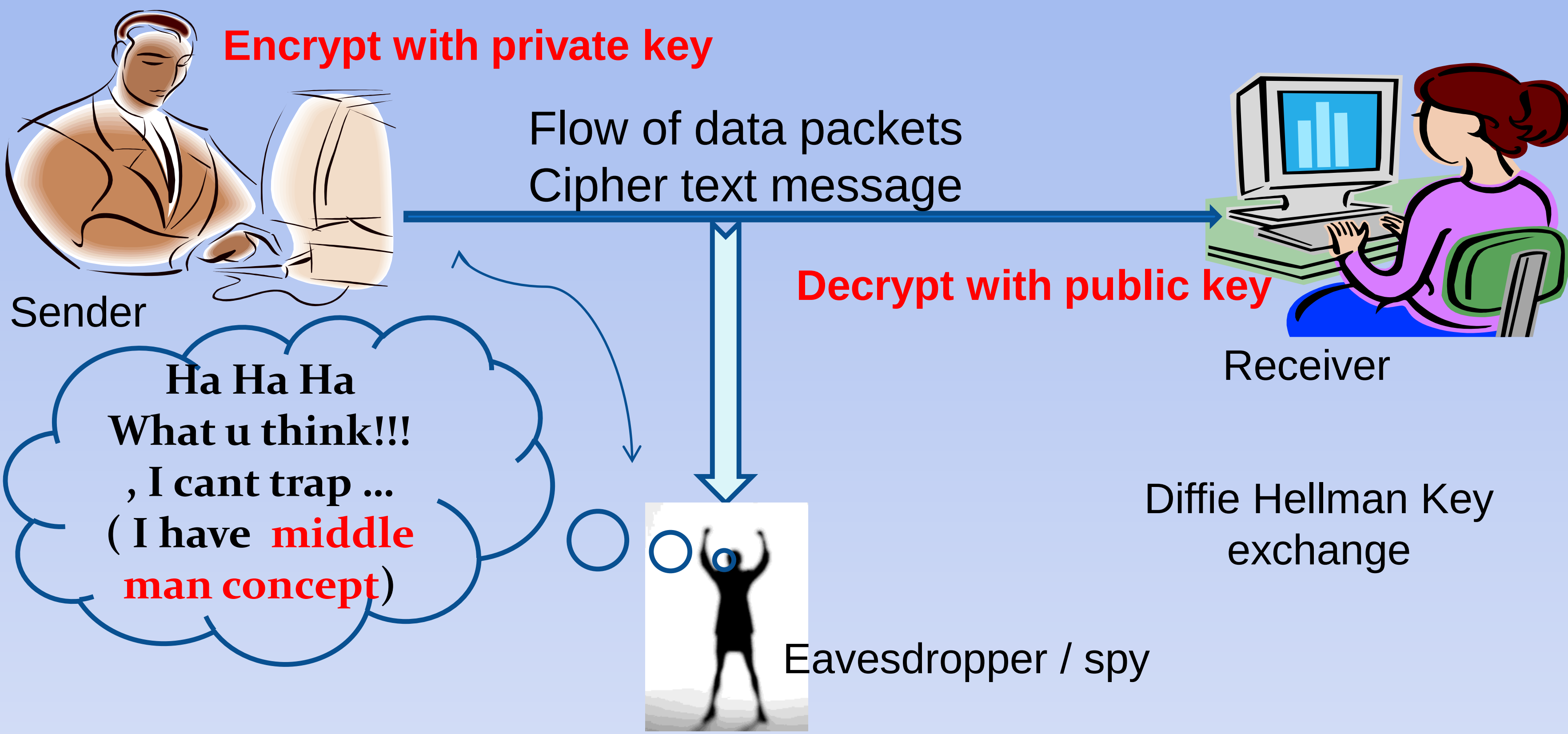
Note:- The sender and the receiver using same key ----------
Symmetric key cryptography

Jkm.cse@gmail.com

Applications of Symmetric Algorithms



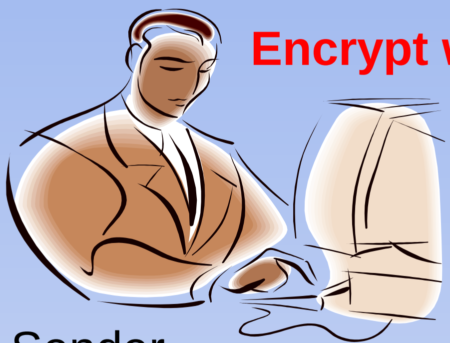
Communication……
With the concept of key



Note:- The sender and the receiver using different key ---------- Asymmetric key cryptography

Jkm.cse@gmail.com

Communication……
With the concept of key



Encrypt with private key

Sender

Flow of data packets
Cipher text message

Decrypt with public key

Receiver

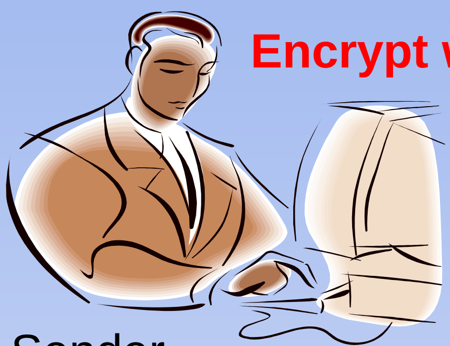
Ha Ha Ha
What u think!!!

The public key of sender is public to all, So any one can decrypt message

Note:- The sender and the receiver using different key ----------
Asymmetric key cryptography

Jkm.cse@gmail.com

Communication…….



Sender

Encrypt with public key of receiver

Flow of data packets
Cipher text message

Decrypt with private key
Of receiver

Receiver

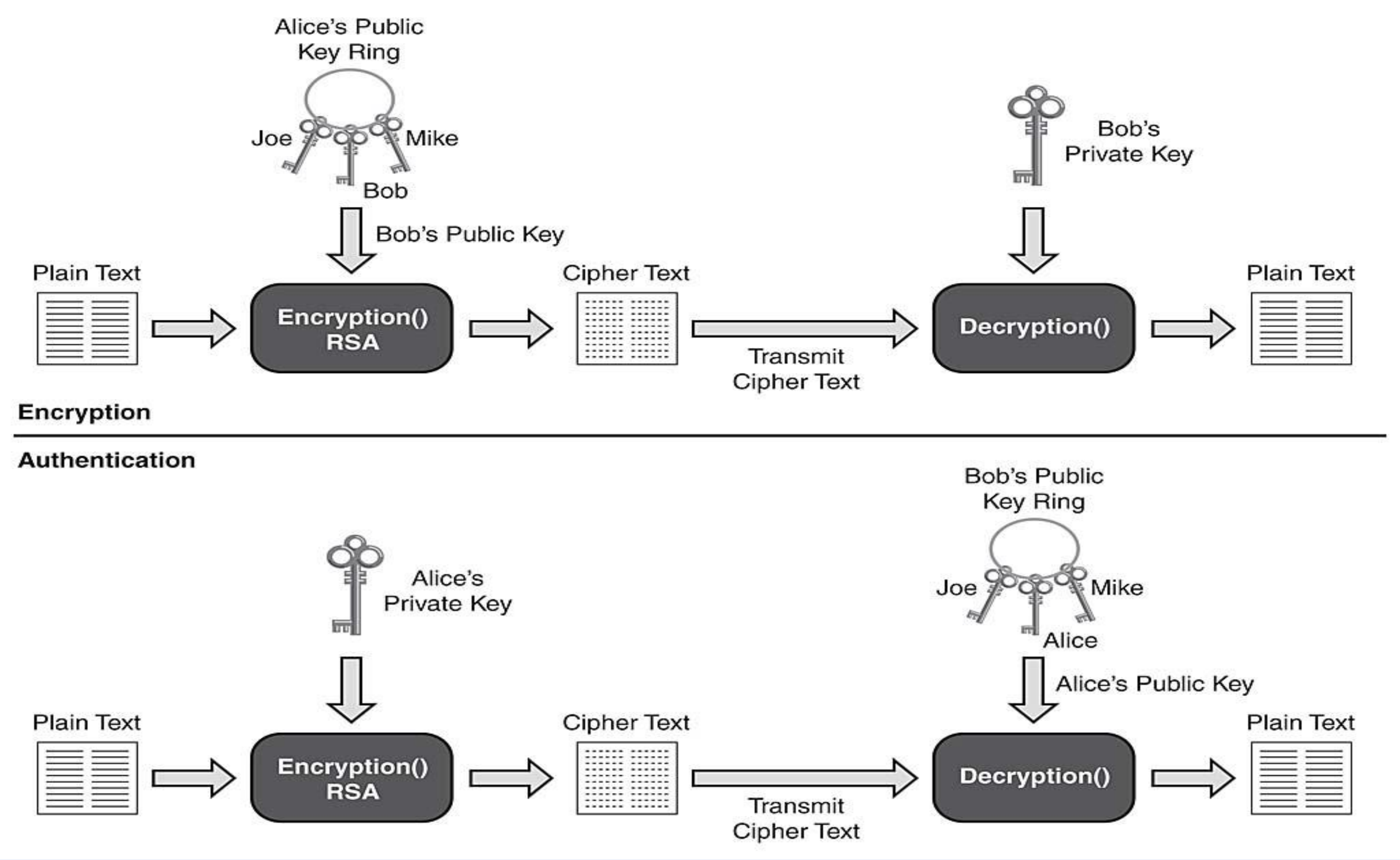
**Here Also I have some roll……
See ,I have The public key of Receiver……**

I can change the message in the mid way and again encrypt. Receiver never able to understand the difference.

Eavesdrop / spy

Jkm.cse@gmail.com

Applications of Asymmetric Algorithms



Eavesdrop / spy

The Main intention of Eavesdrop is to change the information in mid of the way, but the receiver cant able to understand that.

For this

The Concept of **Digital Signature** can be used.

Jkm.cse@gmail.com

Digital Signatures

- A signature is a technique for non-repudiation based on the public key cryptography.
- The creator of a message can attach a code, the signature, which guarantees the source and integrity of the message.

Properties of Signatures

- Similar to handwritten signatures, digital signatures must fulfill the following:
  - ✓Must not be forgeable
  - ✓Recipients must be able to verify them
  - ✓Signers must not be able to repudiate them later
- In addition, digital signatures cannot be constant and must be a function of the entire document it signs.

Types of Signatures

- *Direct digital signature* – involves only the communicating parties
  - ✓ Assumed that receiver knows public key of sender.
  - ✓ Signature may be formed by (1) encrypting entire message with sender's private key or (2) encrypting hash code of message with sender's private key.
  - ✓ Further encryption of entire message + signature with receiver's public key or shared private key ensures confidentiality.

Types of Signatures

- **Problems with direct signatures:**
  - ✓Validity of scheme depends on the security of the sender's private key ⇒ sender may later deny sending a certain message.
  - ✓Private key may actually be stolen from X at time T, so timestamp may not help.

Types of Signatures

- *Arbitrated digital signature* – involves a trusted third party or arbiter
  - ✓Every signed message from sender, X, to receiver, Y, goes to an arbiter, A, first.
  - ✓A subjects message + signature to number of tests to check origin & content.
  - ✓A dates the message and sends it to Y with indication that it has been verified to its satisfaction.

Basic Mechanism of Signature Schemes

- A key generation algorithm to randomly select a public key pair.
- A signature algorithm that takes message + private key as input and generates a signature for the message as output
- A signature verification algorithm that takes signature + public key as input and generates information bit according to whether signature is consistent as output.

Digital Signature Standards

- Kang et al.'s scheme.

- Message recovery and without one-way hash function

Kang et al.'s scheme

- ***Signature generation phase***
- **1. The signer computes s as**

  $s = Y^m \pmod p$ (1)
- **2. The signer selects a random number k in [1, p-1] and computes r as**

  $r = s + m\, g^{-k}$ (2)
- **3. The signer computes t from the following expression.**

  $s + t \equiv x^{-1}(k - r) \bmod (p-1)$ (3)
- **4. The signer sends the signature (r, s, t) of message m to the receiver or verifier.**

p is a large prime no. g is a primitive element in Zp. The signer has private key x, where x < (p-1) and gcd (x, p-1)=1. The public key of the signer is Y, where $Y = g^x \bmod p$. message m Є Zp

Kang et al.'s scheme

- ***Signature verification phase***
- 1. Computes m' as

  $m' \equiv (r-s)\, Y^{s+t} g^{r} \pmod{p}$ (4)

- 2. Checks the authenticity of the signature by verifying (5).

  $s = Y^{m'} \pmod{p}$ (5)

2. Message recovery and without one-way hash function

- *Setup*
- A trusted center chooses an integer n as the product of two primes p and q such that, p=2fp´+1 and q=2fq´+1, where f, p´ and q´ are distinct primes. Then it chooses an integer g of order f both modulo p and q, i.e., g^t (mod n)=1. Then it chooses an integer e which is coprime with both (p-1) and (q-1) and computes d such that ed≡1 mod ϕ(n).
- Finally the trusted center sends d and f to the signer securely and publishes g, n and e as its public data.
- The signer chooses its private key $x \in Z_f$ and Publishes its public key Y, where $Y=g^x$ (mod n)

Message recovery and without one-way hash function

- ***Signature generation phase***
- Computes s as

$$s \equiv Y^d \pmod{n} \qquad (6)$$

- Selects two random numbers k and u both in Z_f and computes r as

$$r = s + m\, g^{(u - k)} \pmod{n} \qquad (7)$$

- The signer computes t from the following expression

$$s + t \equiv x^{-1} (k - r - u) \bmod (n-1) \qquad (8)$$

- The signer then sends the triplet (r, s, t) to the receiver as the signature of the message m.

Message recovery and without one-way hash function

- ***Signature verification phase***
- Checks the authenticity of the signature by computing the following expression.

$$s^e \equiv Y \pmod{n} \quad (9)$$

- It recovers the message m' as

$$m' \equiv (r - s)\, Y^{s+t} g^r \bmod (n - 1) \quad (10)$$

Feature

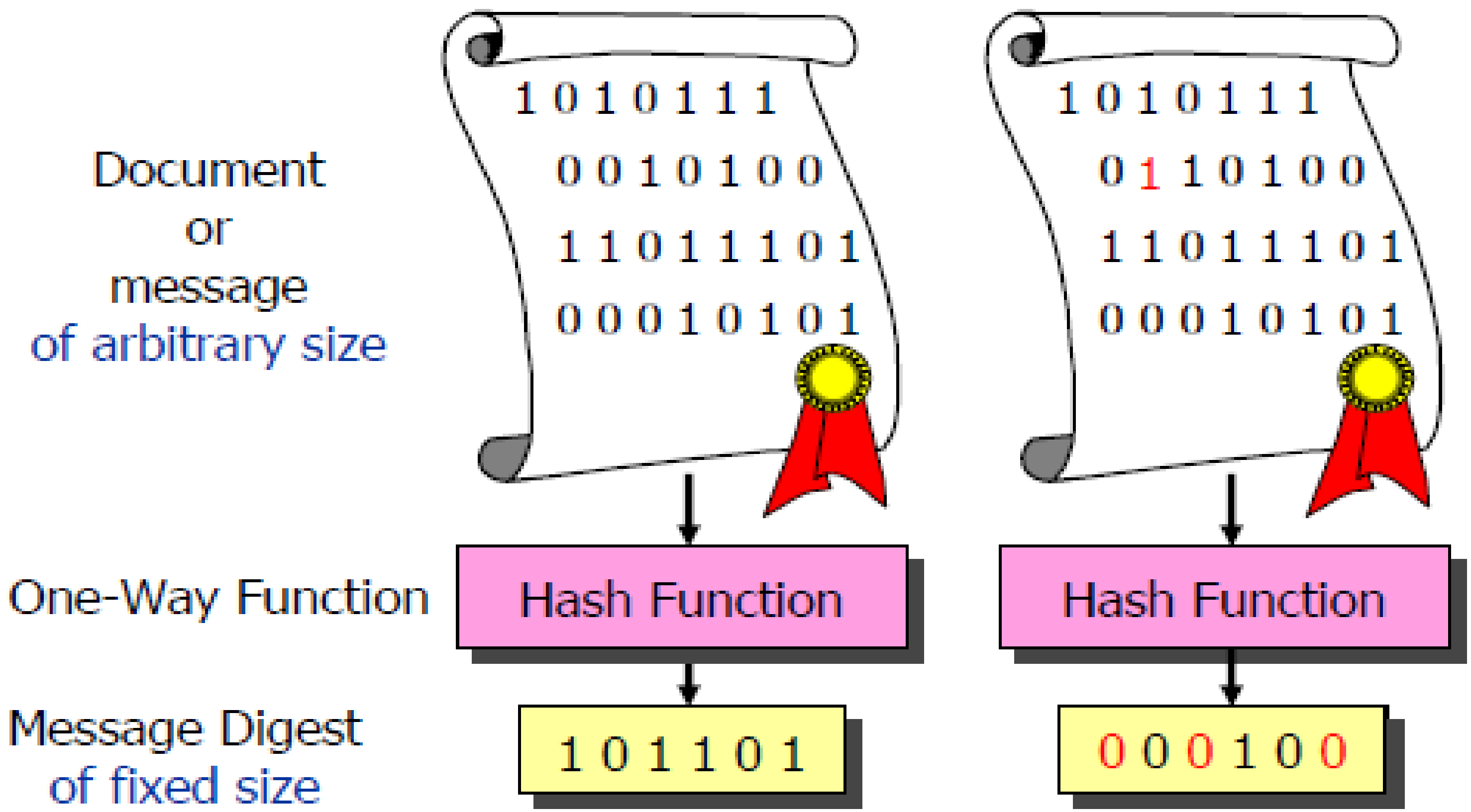
- It prevent following attacks :
  - ✓ *Attacks to recover private key of signer.*
  - ✓ *Attacks for parameter reduction.*
  - ✓ *Forgery Attack.*
- *It is Suitable for long messages.*

Comparison

Features	Kang's scheme	Proposed scheme
Security	Less	More
Message recovery	Supports	Supports
Message redundancy	Supports	Does not
Suitable for long message	No	Yes

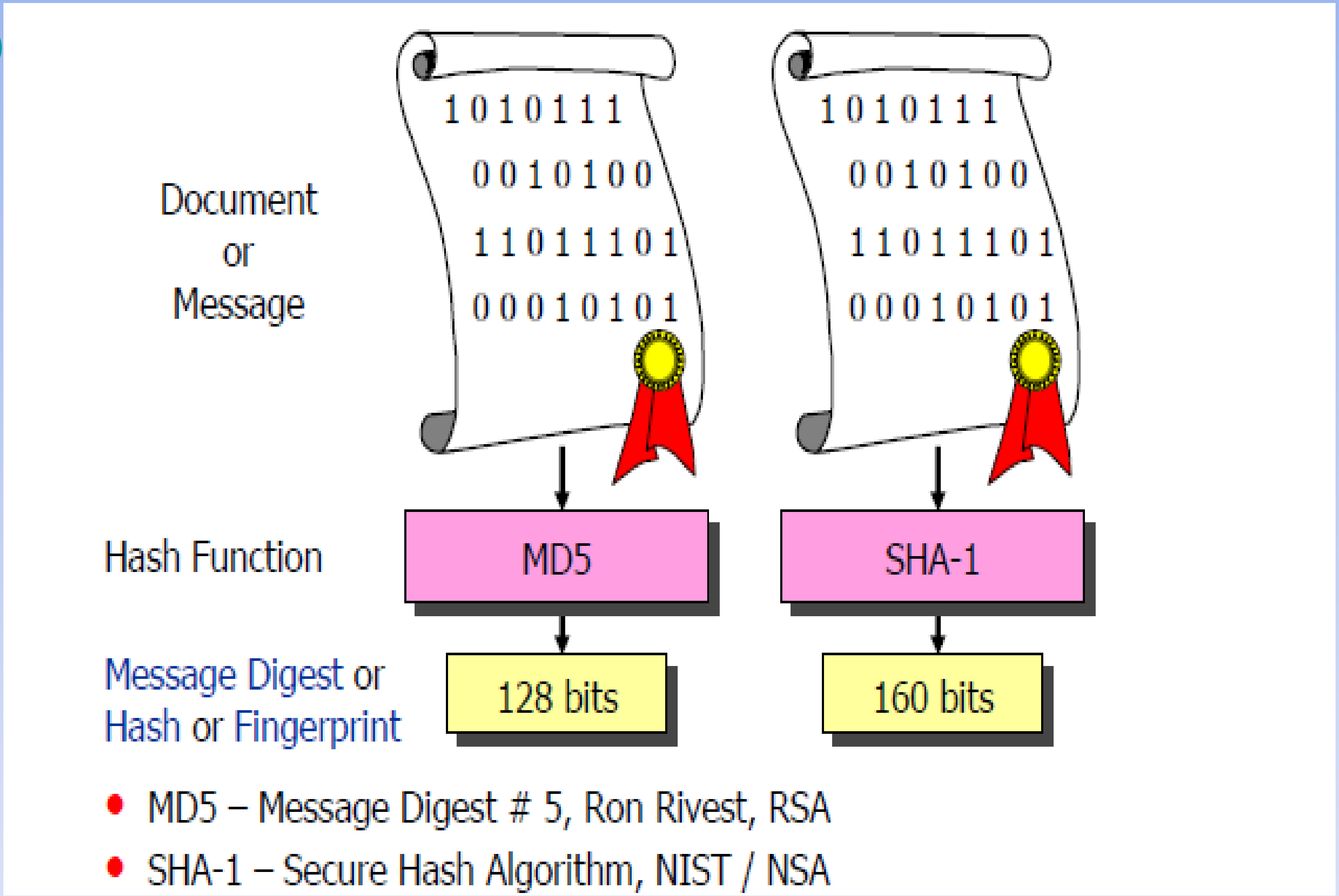
MESSAGE DIGEST

Message Digests based on One–Way Hash Functions

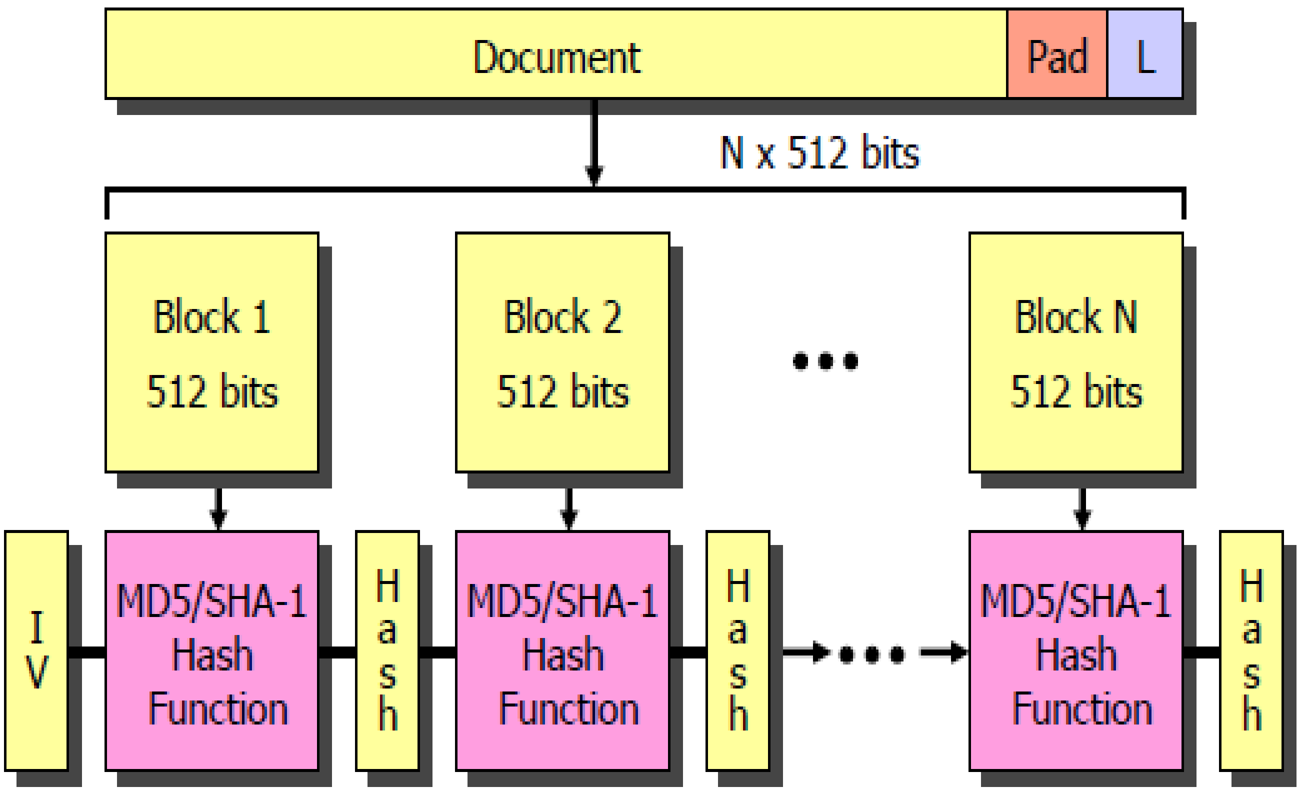


- A single bit change in a document should cause about 50% of the bits in the digest to change their values !

Po


Document
or
Message
1 0 1 0 1 1 1
0 0 1 0 1 0 0
1 1 0 1 1 1 0 1
0 0 0 1 0 1 0 1
1 0 1 0 1 1 1
0 0 1 0 1 0 0
1 1 0 1 1 1 0 1
0 0 0 1 0 1 0 1
Hash Function
MD5
SHA-1
Message Digest or
Hash or Fingerprint
128 bits
160 bits
MD5 – Message Digest # 5, Ron Rivest, RSA
SHA-1 – Secure Hash Algorithm, NIST / NSA

Basic Structure of the MD5 / SHA-1 One–Way Hash Functions



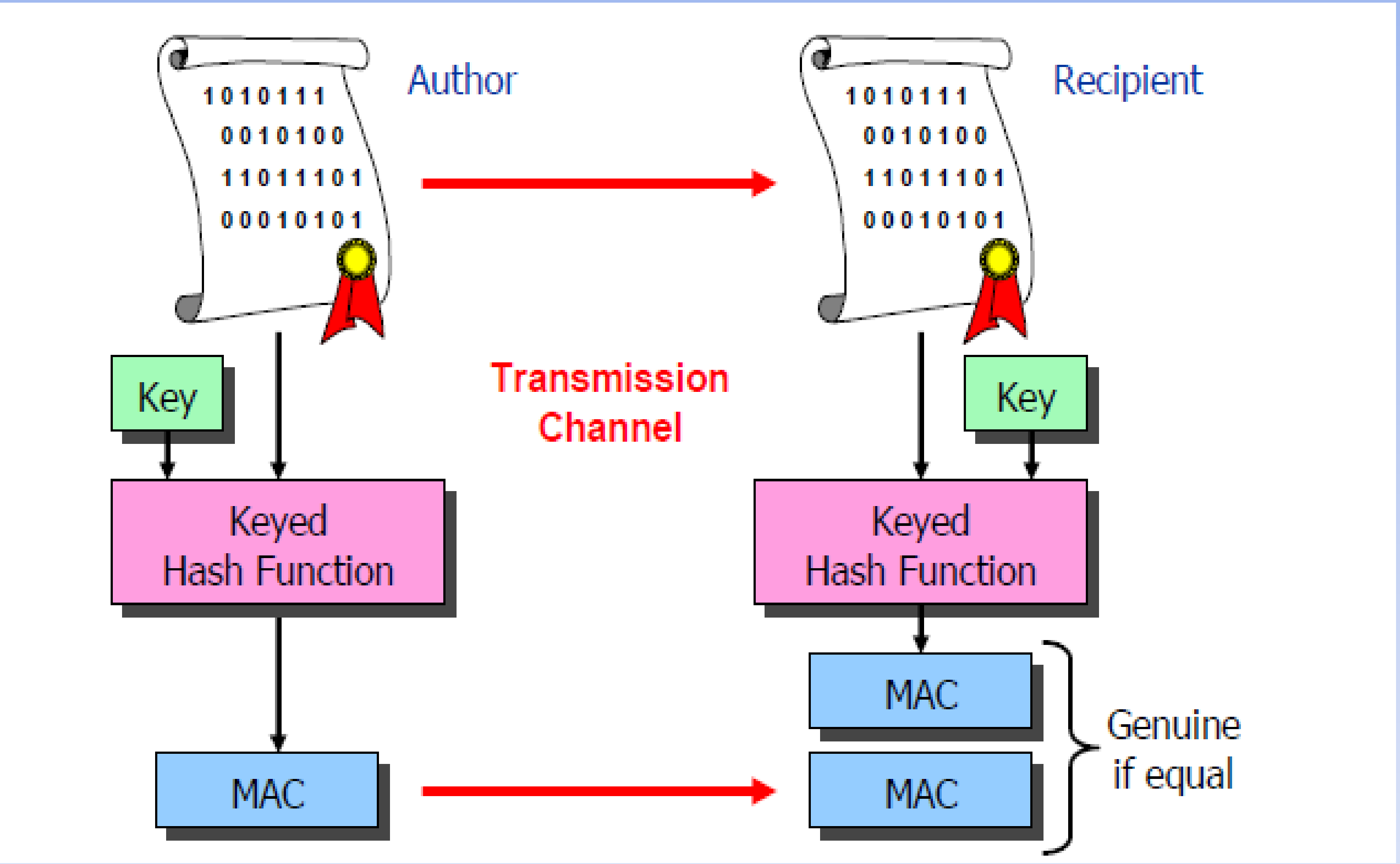
IV 128/160 bit Initialization Vector

Hash 128/160 bit Hash Value

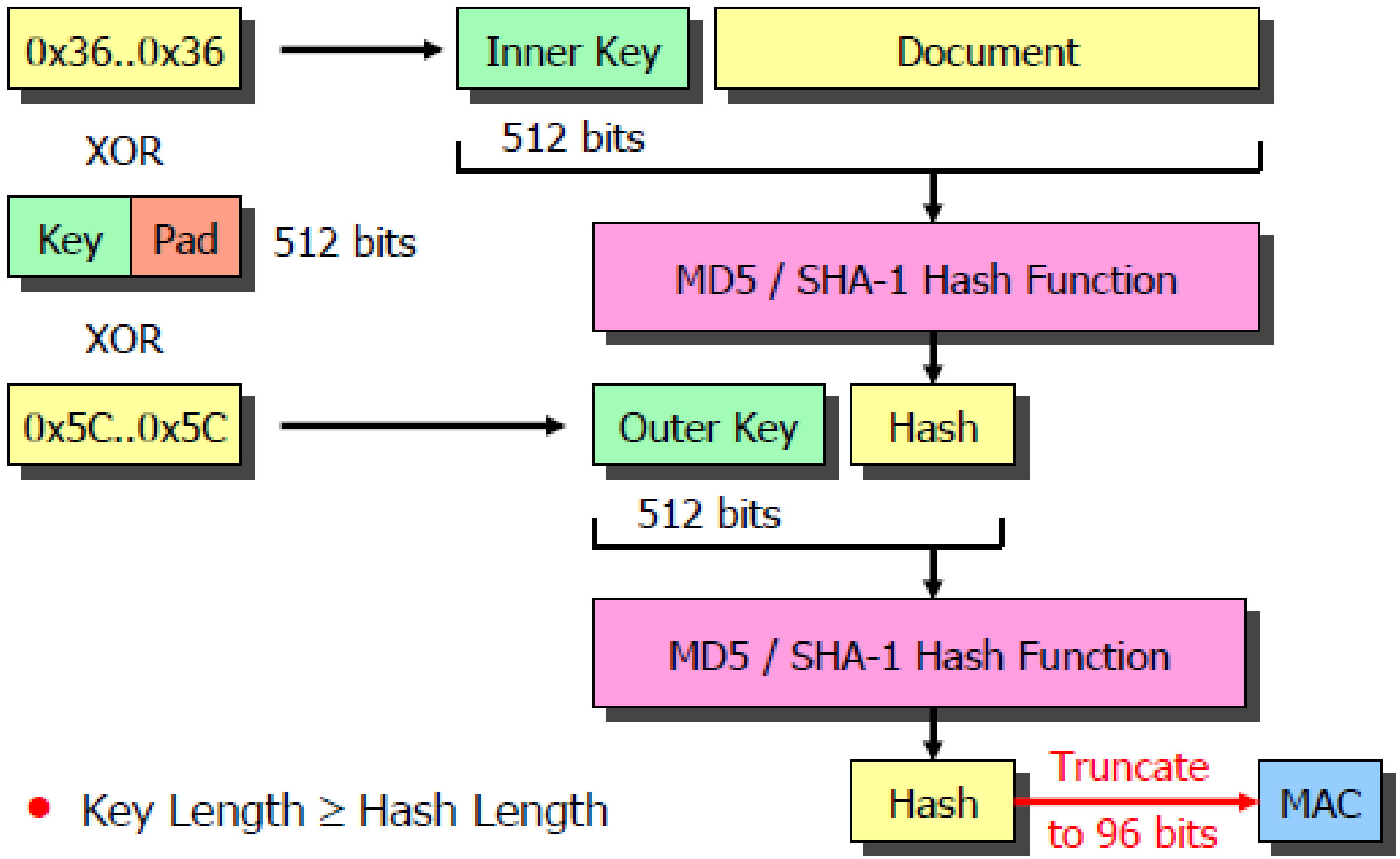
Pad Padding

L 64 bit Document Length

Message Authentication Codes based on Keyed One–Way Hash Function

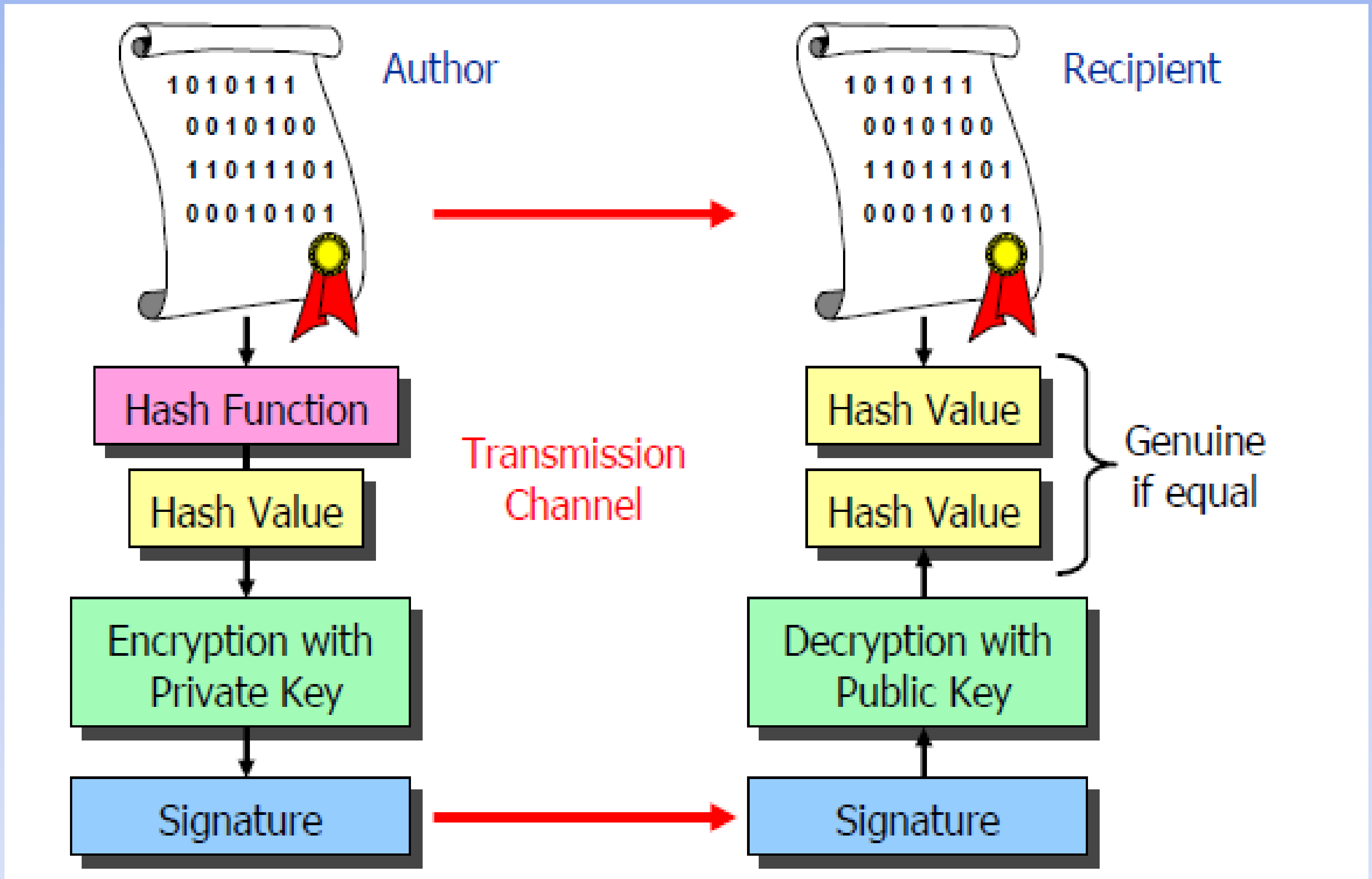


Basic Structure of a Keyed One–Way Hash Function (RFC 2104)

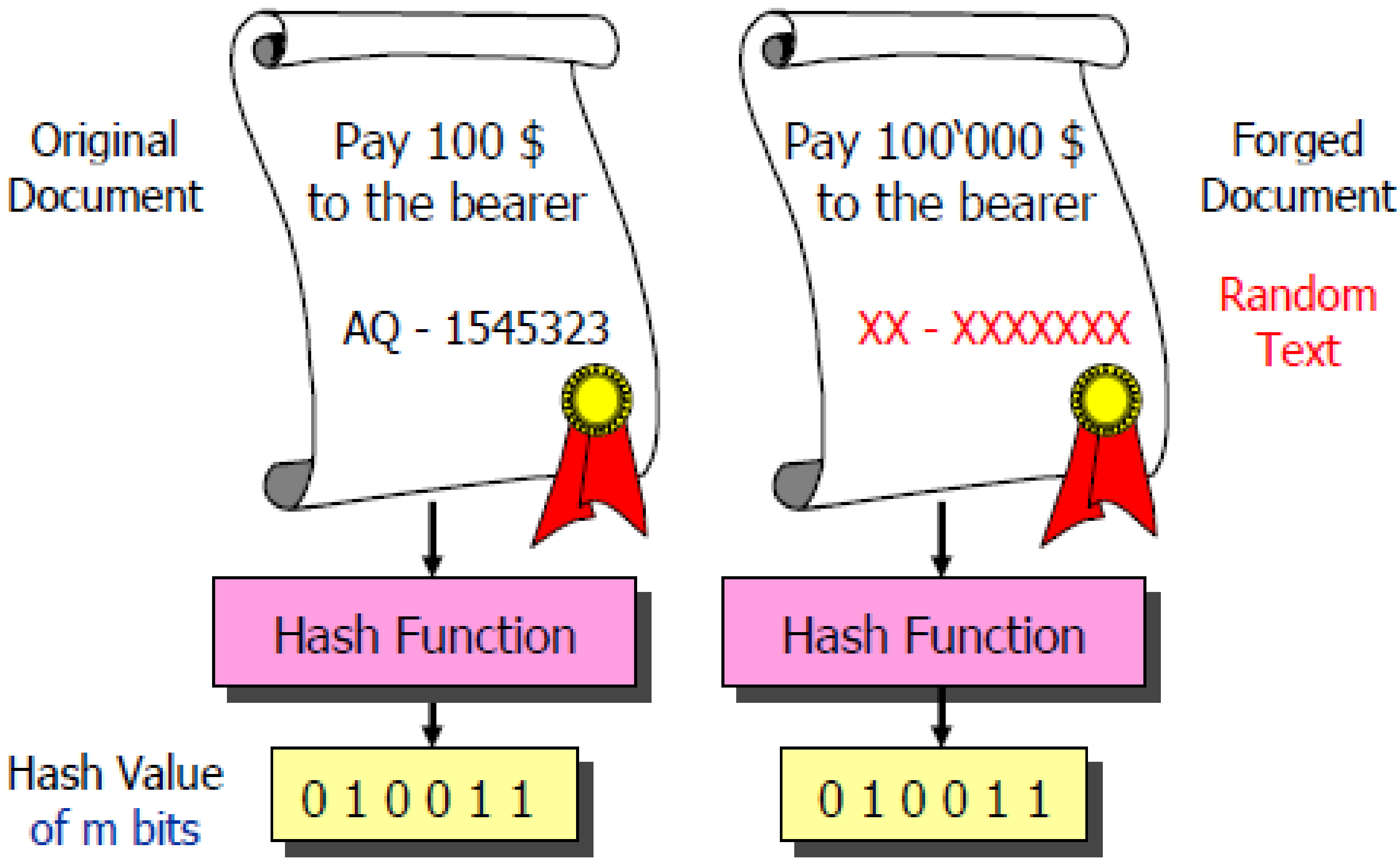


- Key Length ≥ Hash Length

Digital Signatures based on Public Key Cryptosystems



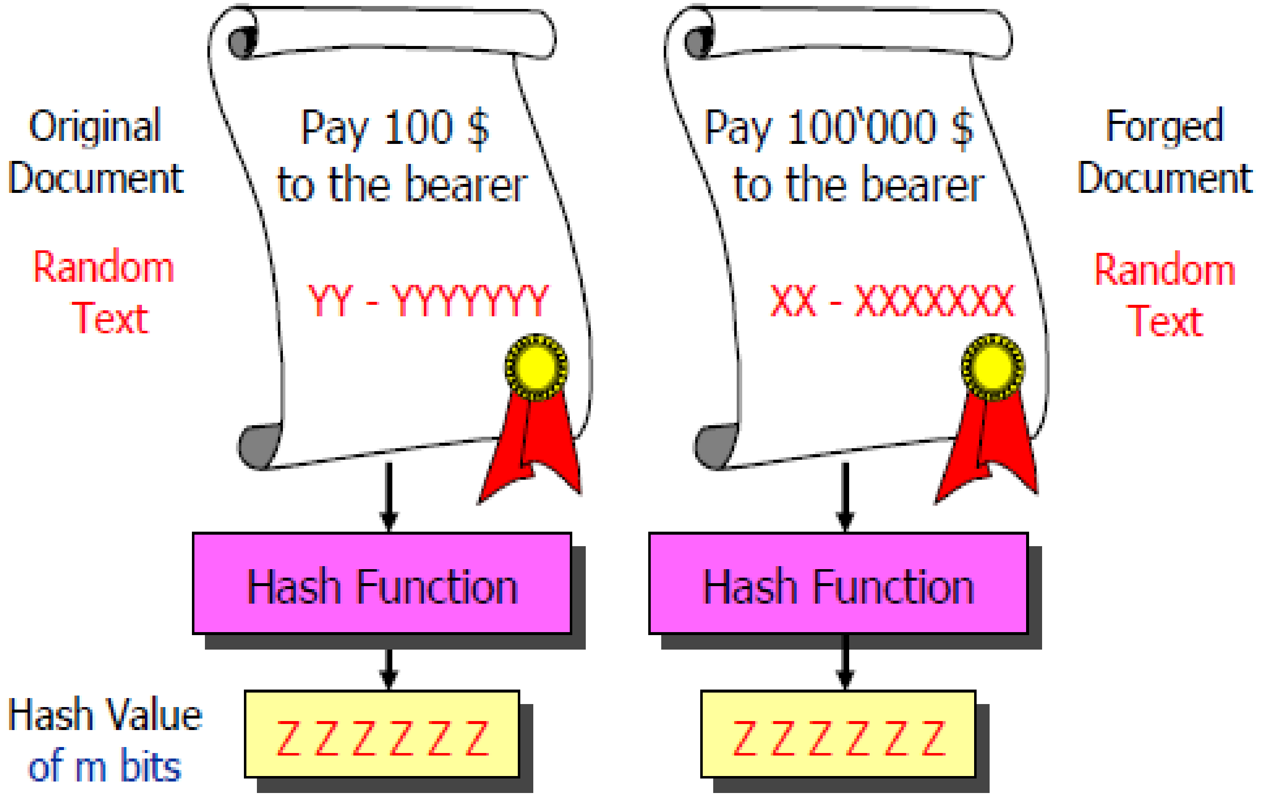
Forging Documents



- On average 2^m trials are required to find a document having the same hash value as a given one !

Birthday Attacks against Hash Functions

Looking for Collisions



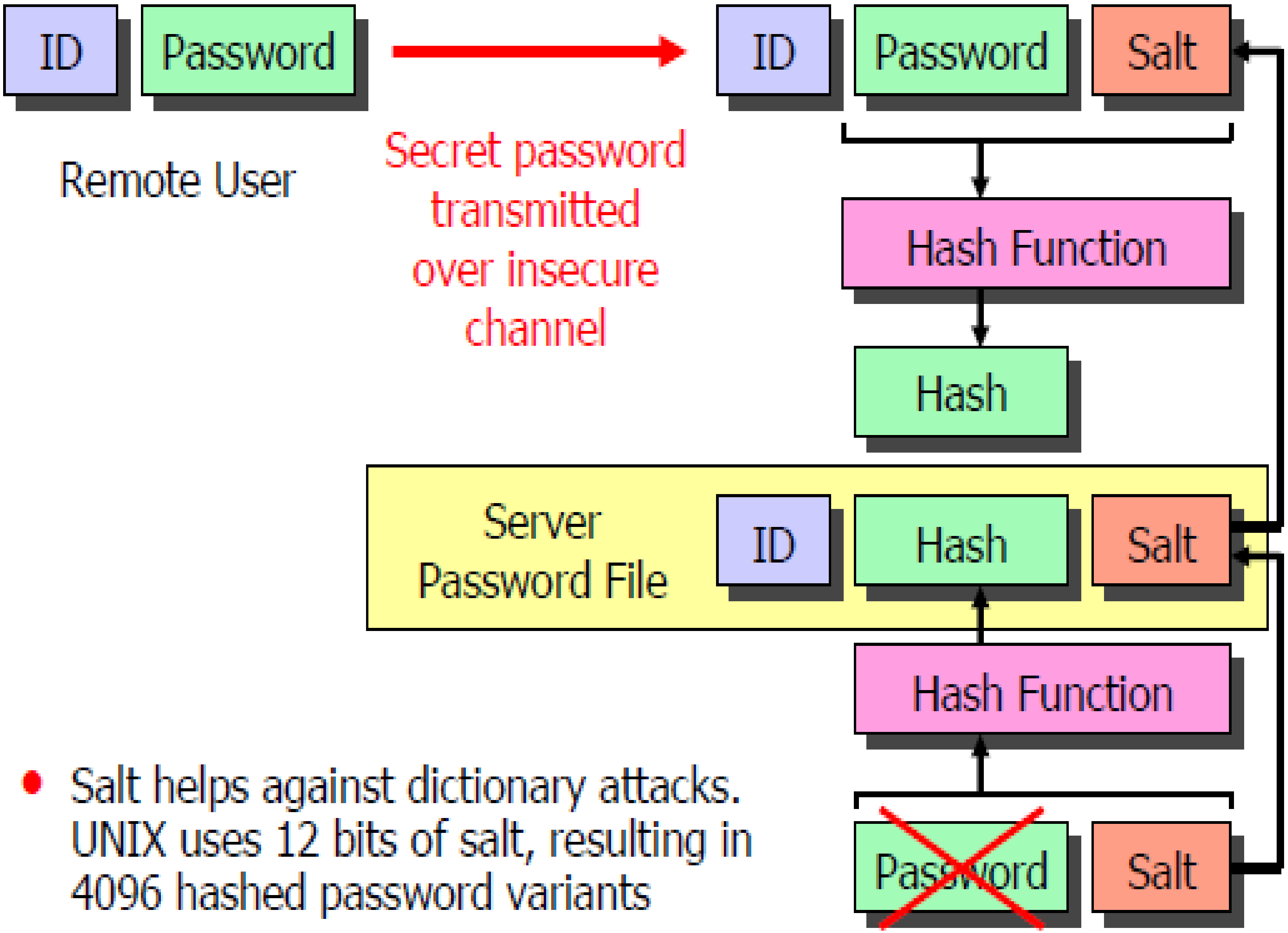
- Less than $2^{m/2}$ trials are required to find two documents having the same hash value $\Rightarrow$ MD5 with 2^{39} and SHA-1 with 2^{63} trials are both insecure !

User Authentication

"On the Internet, nobody knows you're a dog."

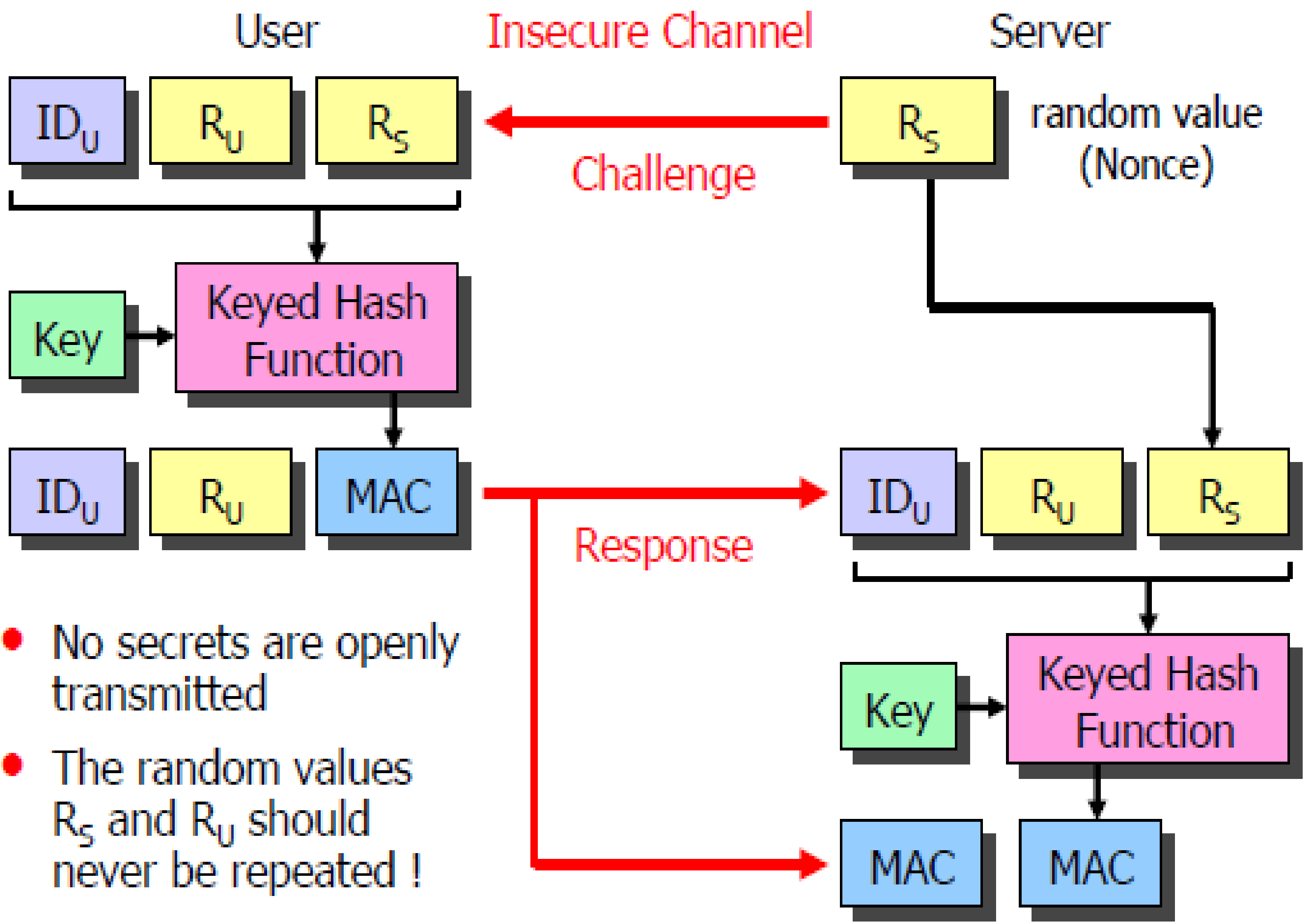
- Username / Password
  - Dictionary Attacks
- One-Time Passwords
  - Token: SecureID, etc.
- Public Key Algorithms
  - Smartcards, Certificates, Public Key Infrastructure
- Biometrical Methods
  - Fingerprint, Iris-Scan, Voice, Face, Hand, etc.

Insecure Authentication based on Passwords



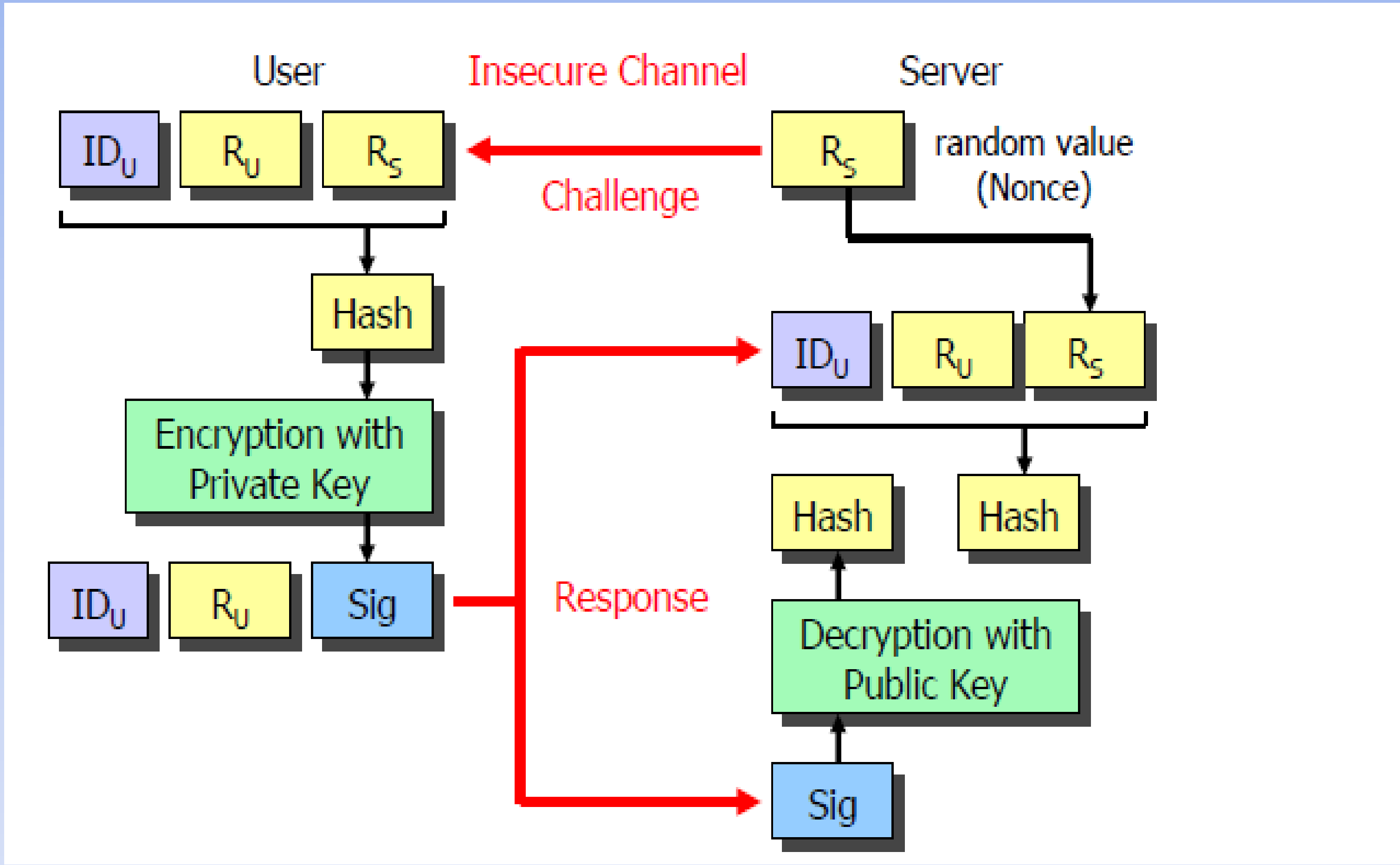
- Salt helps against dictionary attacks. UNIX uses 12 bits of salt, resulting in 4096 hashed password variants

Secure Authentication based on Challenge/Response Protocols

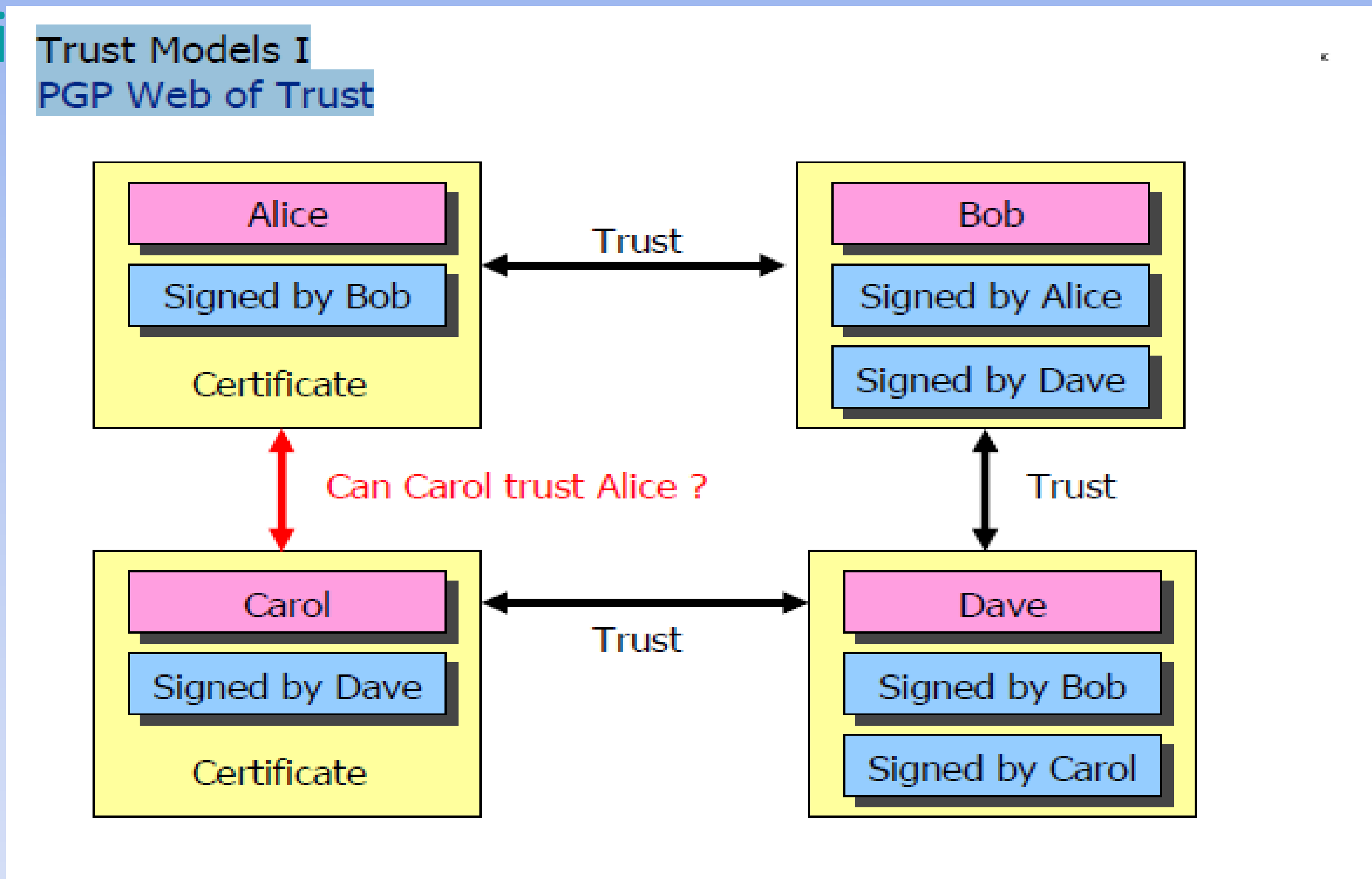


- No secrets are openly transmitted
- The random values R_S and R_U should never be repeated !

Challenge/Response Protocol based on Digital Signatures

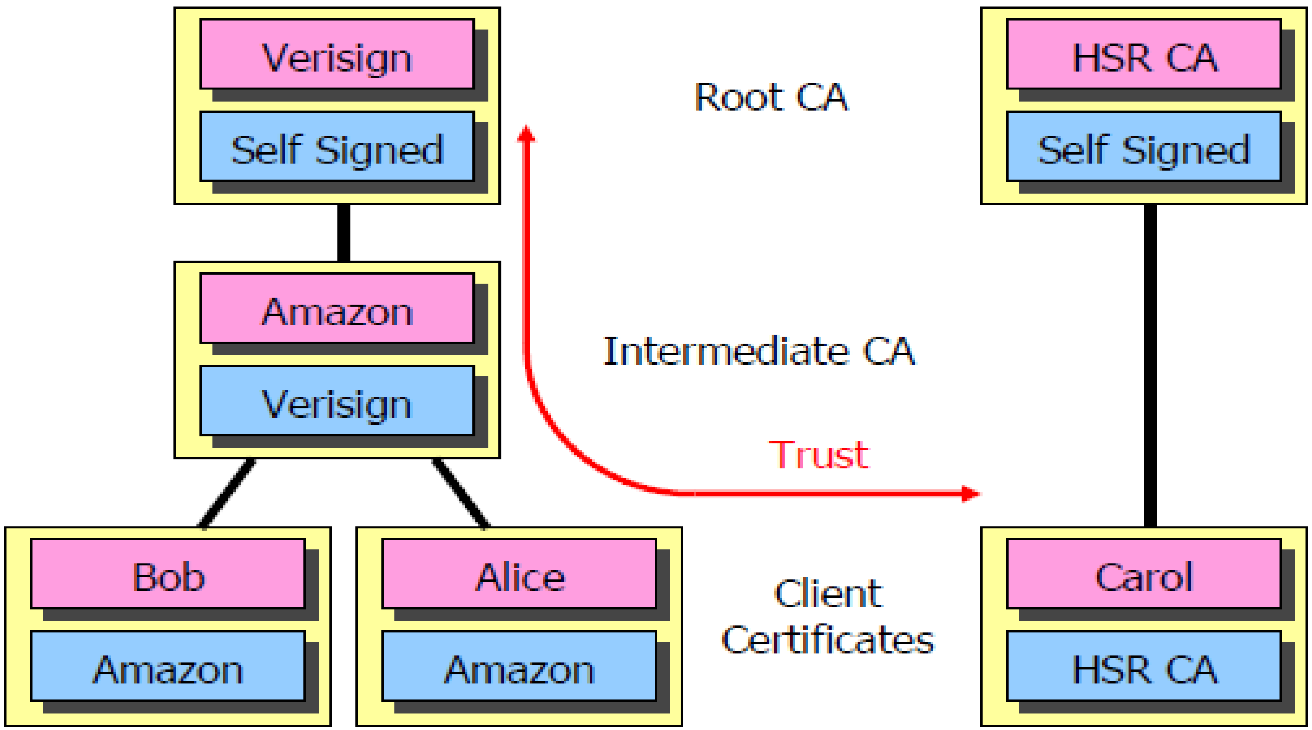


Digi


Trust Models I
PGP Web of Trust
Alice
Signed by Bob
Certificate
Trust
Bob
Signed by Alice
Signed by Dave
Can Carol trust Alice ?
Trust
Carol
Signed by Dave
Certificate
Trust
Dave
Signed by Bob
Signed by Carol

Trust Models II

Trust Hierarchy with Certification Authorities



Authentication and Secret Message Transmission Technique Using Discrete Fourier Transformation.

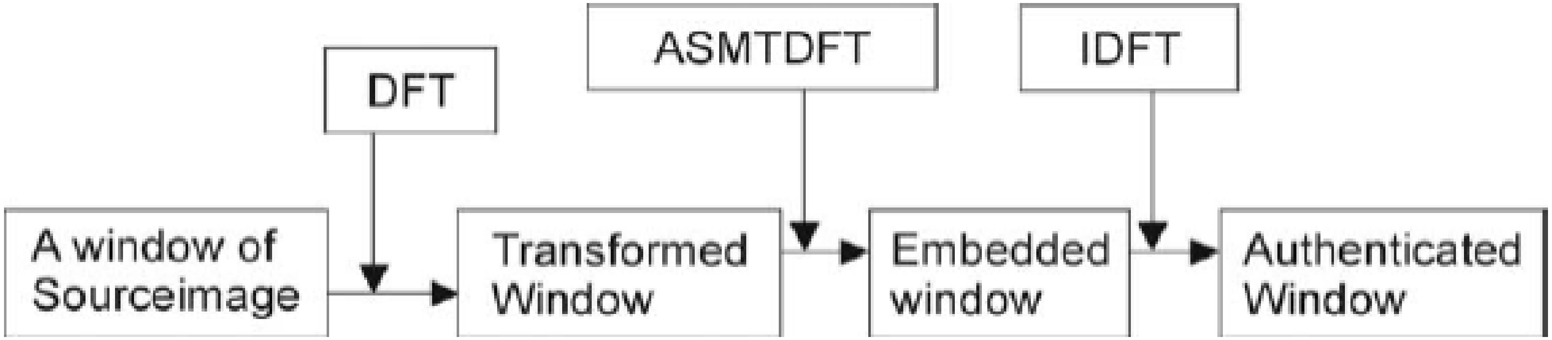


Figure 1. Encoding scheme using ASMTDFT.

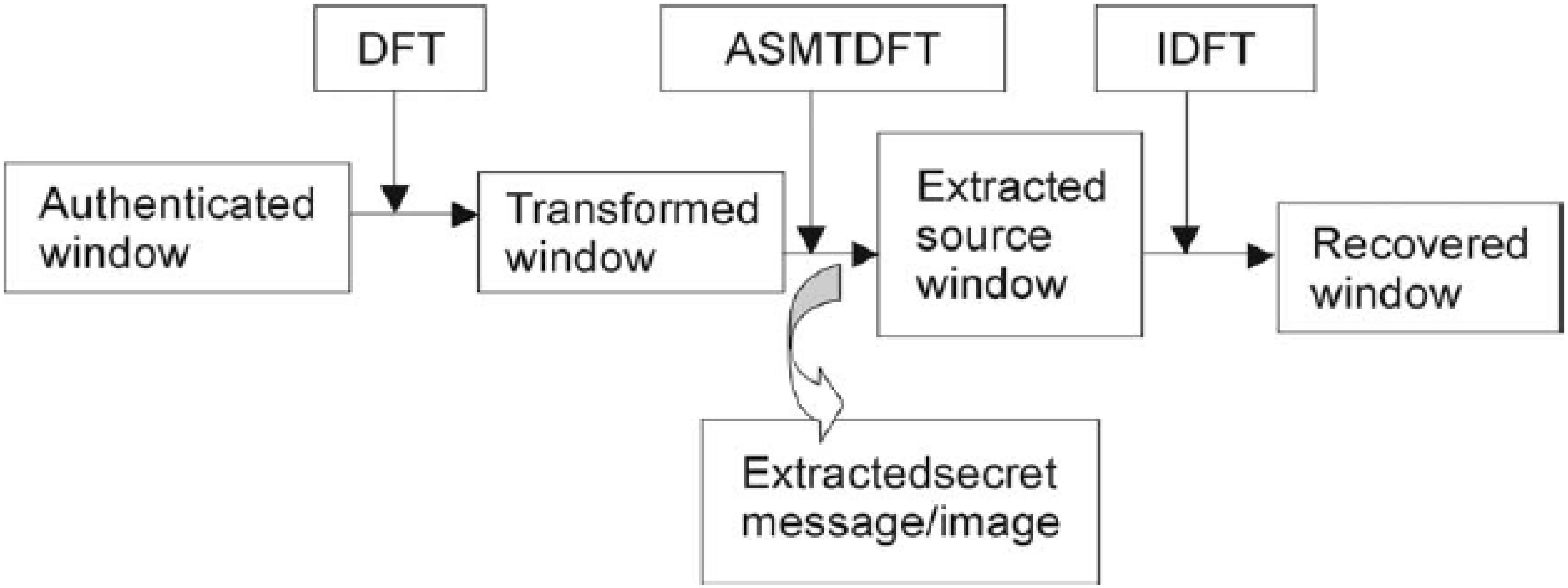


Figure 2. Decoding scheme using ASMTDFT.

Authentication and Secret Message Transmission Technique Using Discrete Fourier Transformation.

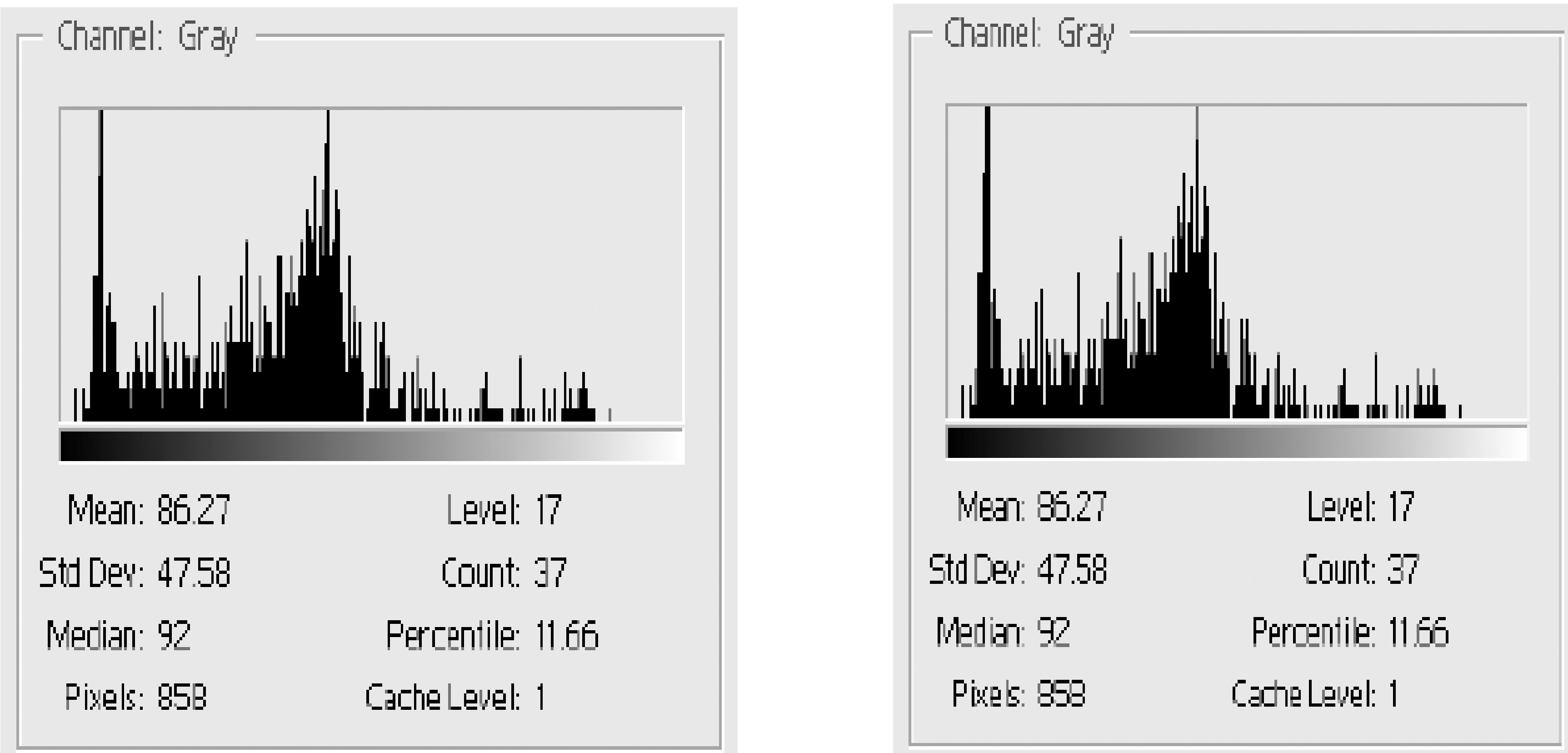
(a). Hill. (b). Lotus. (c). ASMTDFT. (d). S-tools.

Figure 3. Comparison of visual fidelity in embedding 'Lotus' using ASMTDFT and S-Tools.

(a). Rashmancha. (b). Lotus. (c). ASMTDFT. (d). S-tools.

Figure 4. Comparison of visual fidelity in embedding 'Lotus' using ASMTDFT and S-Tools.

Authentication and Secret Message Transmission Technique Using Discrete Fourier Transformation.



(a). Lotus. (b). Extracted Lotus.

. Histogram for authenticating image 'Lotus', extracted image 'Lotus' using ASMTDFT.

Eavesdrop / spy

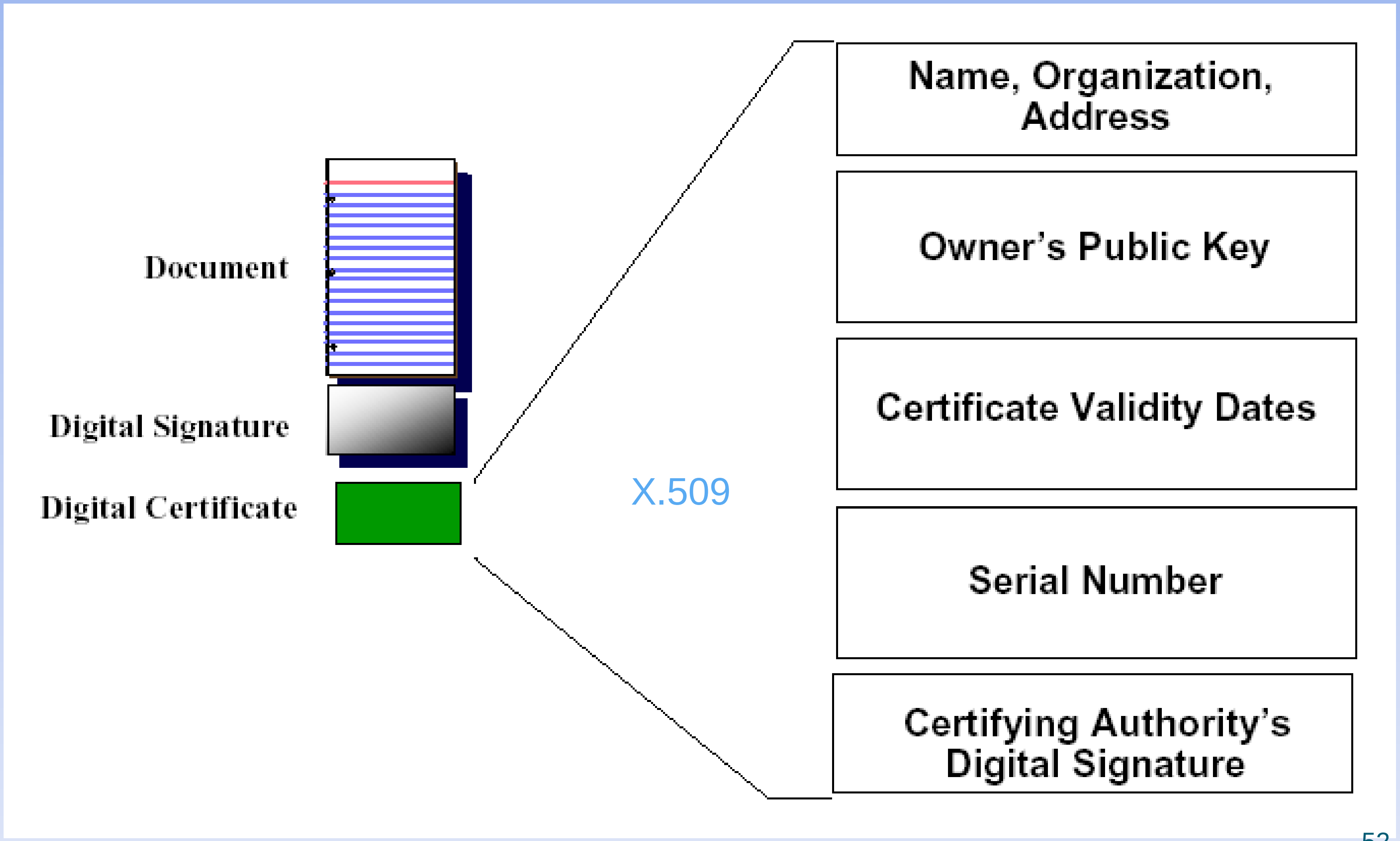
The Main intention of Eavesdrop is to change the information in mid of the way, but the receiver cant able to understand that.

For this

The Concept of **Digital Certificates** can be used.

Jkm.cse@gmail.com

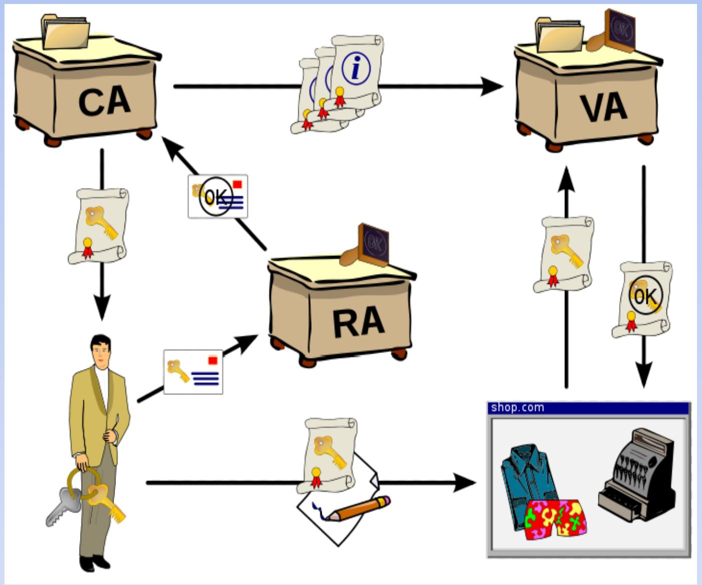
Digital Certificates



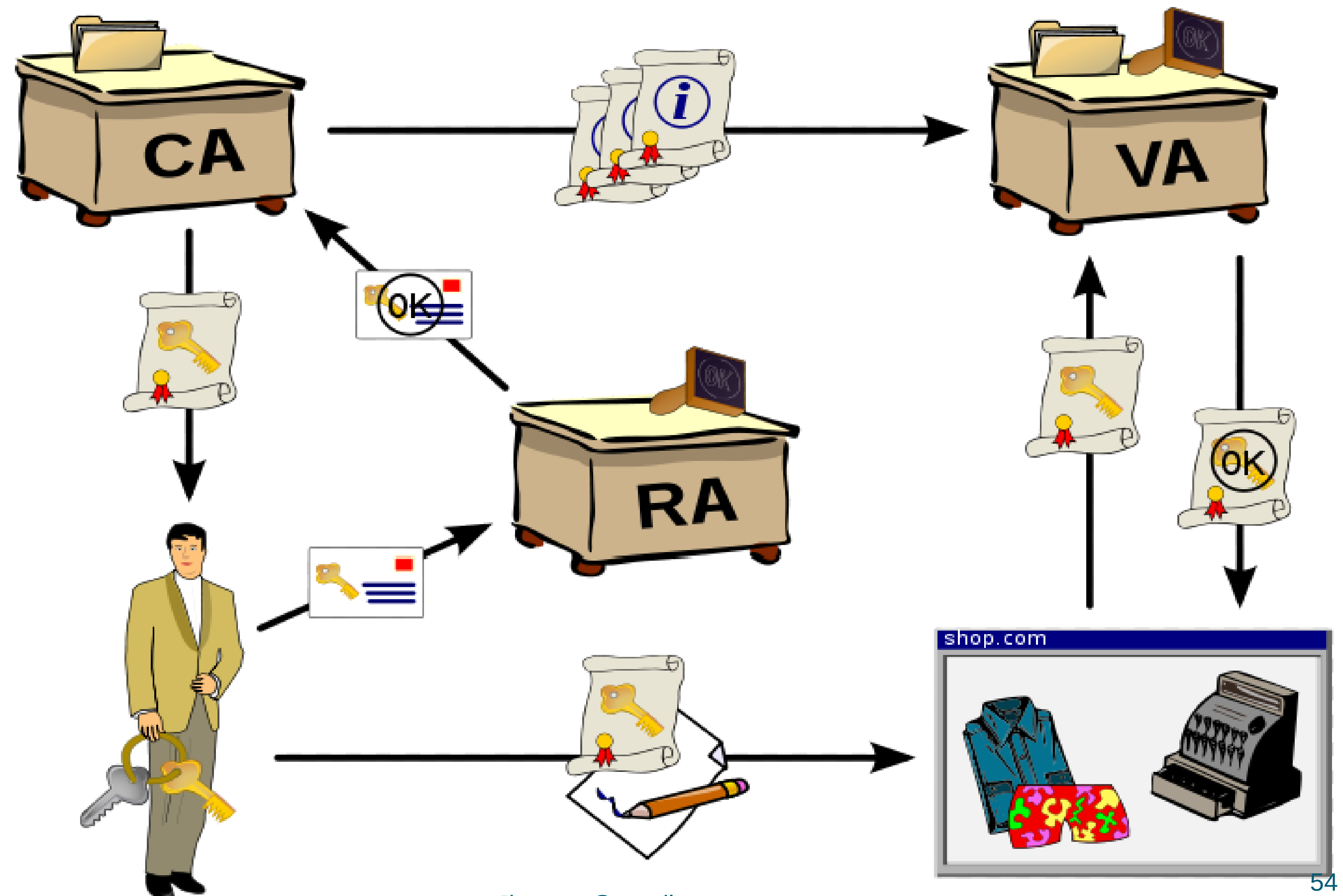
Public Key Infrastructure

The **Public Key Infrastructure (PKI)** is the road ahead for almost all cryptography system.

The **PKI** is a set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates .



Public Key Infrastructure



Jkm.cse@gmail.com

Public Key Infrastructure

- In cryptography, a PKI is an arrangement that binds public keys with respective user identities by means of a certificate authority (CA).

- The PKI role that assures this binding is called the Registration Authority (RA).

- PKIX and PKCS are the two popular standards for digital certificates.

Jkm.cse@gmail.com

Public Key Infrastructure Provides

Privacy and Confidentiality

- Secure Transport
- File Encryption
- Secure e-mail

Authentication

- Network components & end users

Non-repudiation and Data Integrity

- Digital signature
- Trusted time stamp

Jkm.cse@gmail.com

What Organizations Wants?

- Certificates that are accepted nationwide for government, commercial, and financial transactions.
- A trusted CA with strong internal controls over issuance, distribution, and management.
- Policies that are enforceable nationwide.
- Liability protection
- Reasonable pricing

Public Key Infrastructure

As we know, X.509 standard defines the digital certificate structure, format and fields. It also specifies the procedure for distributing the public key. In order to extend such standards and make them universal, the Internet Engineering Task Force (IETF) formed the Public Key Infrastructure X.509(PKIX)

Conceptually, we can compare digital certi... ports or driving licenses. A passport or a driving license helps in establishing our identity. For instance, my passport proves beyond doubt a variety of aspects, the most important ones being:

- My full name
- My nationality
- My date and place of birth
- My photograph and signature

Likewise, my digital certificate would also prove something very critical, as we shall study.

5.2.2 The Concept of Digital Certificates

A digital certificate is simply a small computer file. For example, my digital certificate would actually a computer file with the file name such as atul.cer (where *.cer* signifies the first three characters of word *certificate*. Of course, this is just an example: in actual practice, the file extensions can be different.) Just as my passport signifies the association between me and my other characteristics such as full name, nationality, date and place of birth, photograph and signature, my digital certificate simply signifies the association between my public key and me. This concept of digital certificates is shown in Fig. 5.1. Note that this is merely a conceptual view and does not depict the actual contents of a digital certificate.

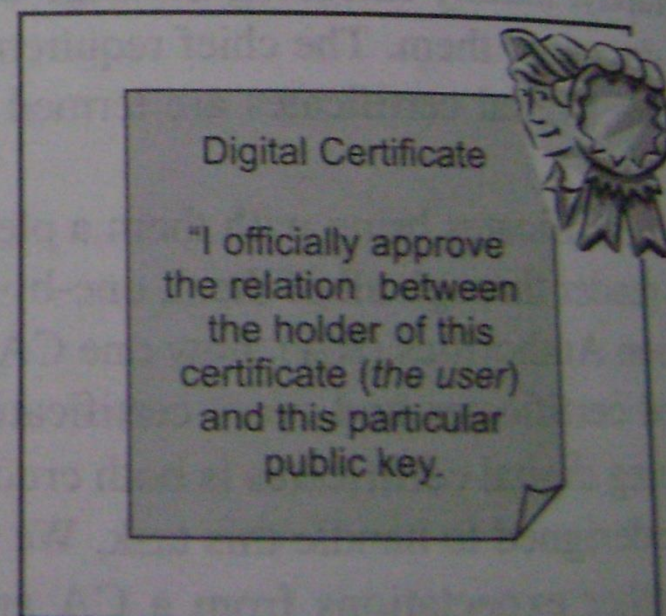


Fig. 5.1 *Conceptual view of a digital cer*

We have not specified who is officially approving the association between a user and the ch as the validity date range for the certificate and who

Passport entry	Corresponding digital certificate entry
Full name	Subject name
Passport number	Serial number
Valid from	Same
Valid to	Same
Issued by	Issuer name
Photograph and signature	Public key

Fig. 5.3 *Similarities between a passport and a digital certificate*

cate also contains other pieces of information, such as the validity date range for the certificate and who has issued it (**issuer name**). Let us try to understand the meanings of these pieces of information by comparing them with the corresponding entries in my passport. This is shown in Fig. 5.3.

As the figure shows, the digital certificate is actually quite similar to a passport. Just as every passport has a unique passport number, every digital certificate has a unique serial number. As we know, no two passports issued by the same issuer (i.e. government) can have the same passport number. Similarly, no two digital certificates issued by the same issuer can have the same serial number. Who can issue these digital certificates? We shall soon answer this question.

Passport entry	Corresponding digital certificate entry
Full name	Subject name
Passport number	Serial number
Valid from	Same
Valid to	Same
Issued by	Issuer name
Photograph and signature	Public key

Fig. 5.3 *Similarities between a passport and a digital certificate*

5.2.3 Certification Authority (CA)

A **Certification Authority** (CA) is a trusted agency that can issue digital certificates. Who can be CA? Obviously, not any Tom, Dick and Harry can be a CA. The authority of acting as a CA has to with someone who everybody trusts. Consequently, the governments in the various countries dec who can and who cannot be a CA. (It is another matter that not everybody trusts the government in first place!) Usually, a CA is a reputed organization, such as a post office, financial institution, softw company, etc. Two of the world's most famous CAs are VeriSign and Entrust. Safescrypt Limite subsidiary of Satyam Infoway Limited, became the first Indian CA in February 2002.

Thus, a CA has the authority to issue digital certificates to individuals and organizations, which to use those certificates in asymmetric key cryptographic applications.

cate also contains other pieces of information, such as the validity date range for the certificate and who has issued it (**issuer name**). Let us try to understand the meanings of these pieces of information by comparing them with the corresponding entries in my passport. This is shown in Fig. 5.3.

As the figure shows, the digital certificate is actually quite similar to a passport. Just as every passport has a unique passport number, every digital certificate has a unique serial number. As we know, no two passports issued by the same issuer (i.e. government) can have the same passport number. Similarly, no two digital certificates issued by the same issuer can have the same serial number. Who can issue these digital certificates? We shall soon answer this question.

Passport entry	Corresponding digital certificate entry
Full name	Subject name
Passport number	Serial number
Valid from	Same
Valid to	Same
Issued by	Issuer name
Photograph and signature	Public key

Fig. 5.3 *Similarities between a passport and a digital certificate*

5.2.3 Certification Authority (CA)

A **Certification Authority (CA)** is a trusted agency that can issue digital certificates. Who can be a CA? Obviously, not any Tom, Dick and Harry can be a CA. The authority of acting as a CA has to be with someone who everybody trusts. Consequently, the governments in the various countries decide who can and who cannot be a CA. (It is another matter that not everybody trusts the government in the first place!) Usually, a CA is a reputed organization, such as a post office, financial institution, software company, etc. Two of the world's most famous CAs are VeriSign and Entrust. Safescrypt Limited, a subsidiary of Satyam Infoway Limited, became the first Indian CA in February 2002.

Thus, a CA has the authority to issue digital certificates to individuals and organizations, which want to use those certificates in asymmetric key cryptographic applications.

rspective. Readers who are not very keen to u... ...ut any lo
ntinuity.

A standard called as **X.509** defines the structure of a digital certificate. The Internat ecommunication Union (ITU) came up with this standard in 1988. At that time, it was a p ther standard called as **X.500**. Since then, X.509 was revised twice (in 1993 and again in 1995 ent version of the standard is Version 3, called as X.509V3. The Internet Engineering Task F) published the RFC2459 for the X.509 standard in 1999. Figure 5.4 shows the structu 9V3 digital certificate.

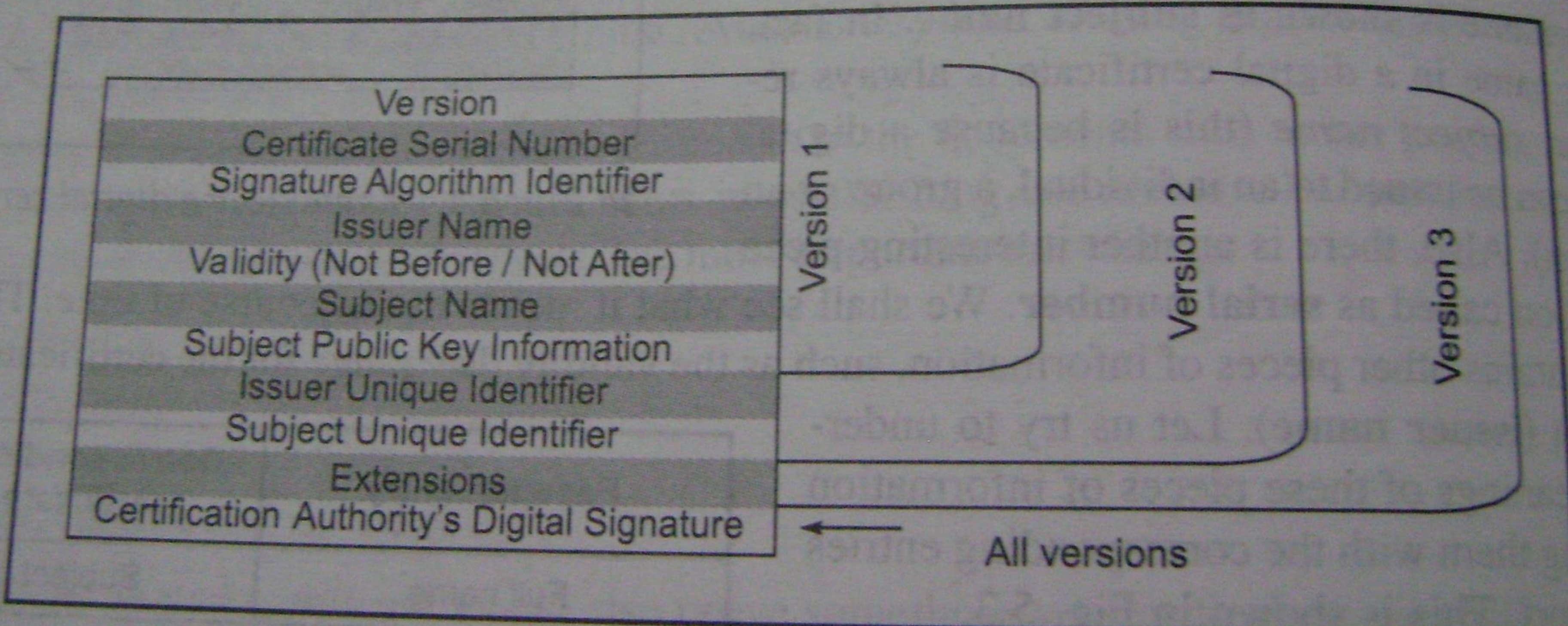


Fig. 5.4 *Contents of a digital certificate*

igure shows the various fields of a digital certificate according to the X.50 r, it also specifies which version of th...

Field	Description
Version	Identifies a particular version of the X.509 protocol, which is used for this digital certificate. Currently, this field can contain 1, 2 or 3.
Certificate Serial Number	Contains a unique integer number, which is generated by the CA.
Signature Algorithm Identifier	Identifies the algorithm used by the CA to sign this certificate. (We shall examine this later).
Issuer Name	Identifies the **Distinguished Name (DN)** of the CA that created and signed this certificate.
Validity (Not Before/Not After)	Contains two date-time values (*Not Before and Not After*), which specify the timeframe within which the certificate should be considered as valid. These values generally specify the date and time up to seconds or milliseconds.
Subject Name	Identifies the *Distinguished Name (DN)* of the end entity (i.e. the user or the organization) to whom this certificate refers. This field must contain an entry unless an alternative name is defined in Version 3 extensions.
Subject Public Key Information	Contains the subject's public key and algorithms related to that key. This field can never be blank.

Fig. 5.5 *(a) Description of the various fields in a X.509 digital certificate – Version 1*

Field	Description
Issuer Unique Identifier	Helps identify a CA uniquely if two or more CAs have used the same *Iss* *Name* over time.
Subject Unique Identifier	Helps identify a subject uniquely if two or more subjects have used the same *Subject Name* over time.

Fig. 5.5 *(b) Description of the various fields in a X.509 digital certificate – Version 2*

Field	Description
Version	Identifies a particular version of the X.509 protocol, which is used for this digital certificate. Currently, this field can contain 1, 2 or 3.
Certificate Serial Number	Contains a unique integer number, which is generated by the CA.
Signature Algorithm Identifier	Identifies the algorithm used by the CA to sign this certificate. (We shall examine this later).
Issuer Name	Identifies the **Distinguished Name (DN)** of the CA that created and signed this certificate.
Validity (Not Before/Not After)	Contains two date-time values (*Not Before* and *Not After*), which specify the timeframe within which the certificate should be considered as valid. These values generally specify the date and time up to seconds or milliseconds.
Subject Name	Identifies the *Distinguished Name (DN)* of the end entity (i.e. the user or the organization) to whom this certificate refers. This field must contain an entry unless an alternative name is defined in Version 3 extensions.
Subject Public Key Information	Contains the subject's public key and algorithms related to that key. This field can never be blank.

Fig. 5.5 *(a) Description of the various fields in a X.509 digital certificate – Version 1*

Field	Description
Issuer Unique Identifier	Helps identify a CA uniquely if two or more CAs have used the same *Issuer Name* over time.
Subject Unique Identifier	Helps identify a subject uniquely if two or more subjects have used the same *Subject Name* over time.

Fig. 5.5 *(b) Description of the various fields in a X.509 digital certificate – Version 2*

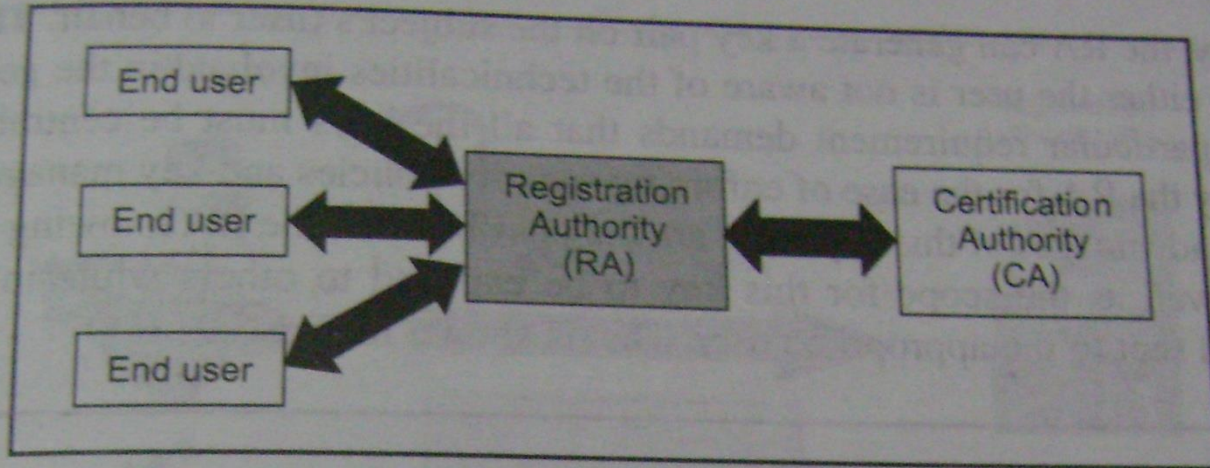


Fig. 5.6 *Registration authority (RA)*

Certificate Creation Steps The creation of a digital certificate consists of several steps. These steps are outlined in Fig. 5.7.

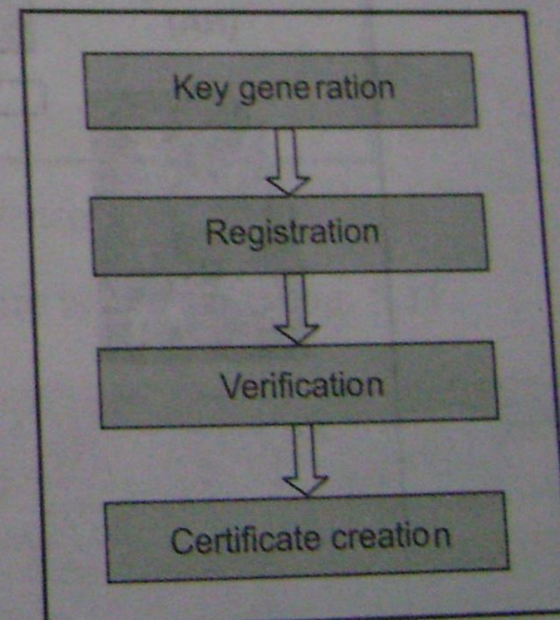


Fig. 5.7 *Digital certificate creation ste*

Let us now examine these steps, required for the creation of a digital certificate.

Step 1: Key generation The action begins with the subject (i.e. the user/organization) who wants to obtain a certificate. There are two different approaches for this purpose:

(a) The subject can create a private key and public key pair using some software. This software is usually a part of the Web browser or Web server. Alternatively, special software programs can be used for this. The subject must keep the pri

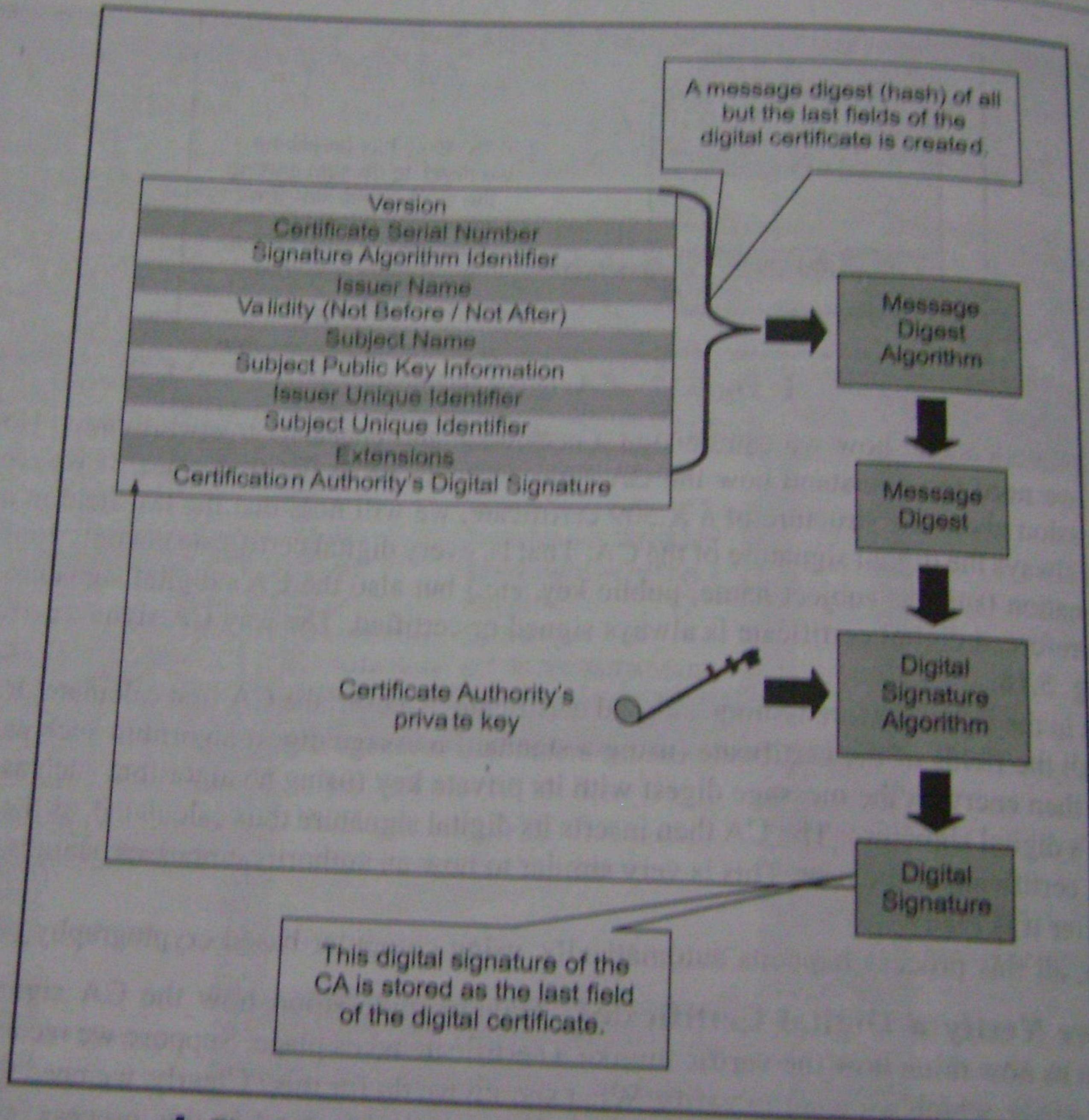


Fig. 5.18 *Creation of the CA signature on a certificate*

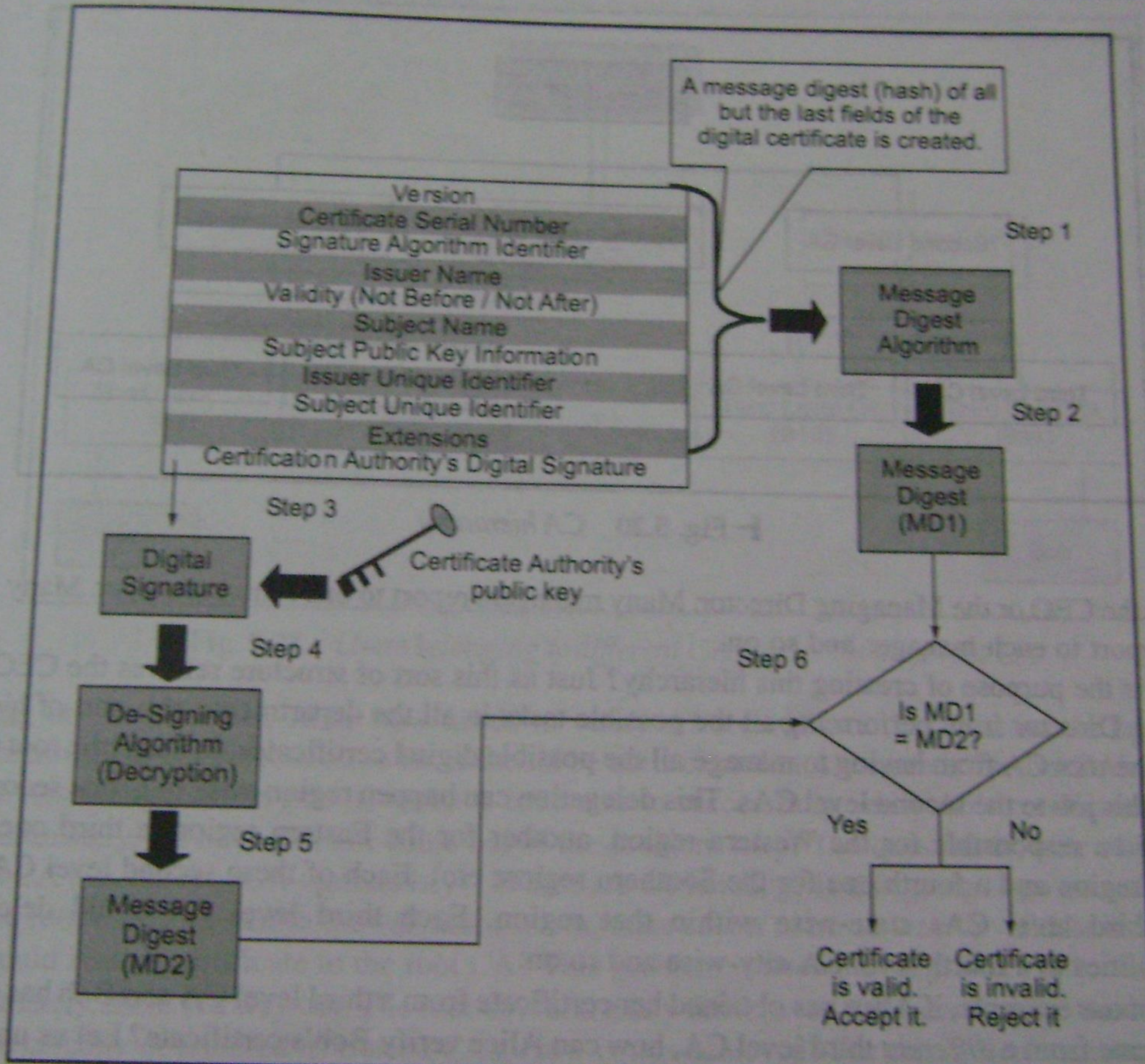
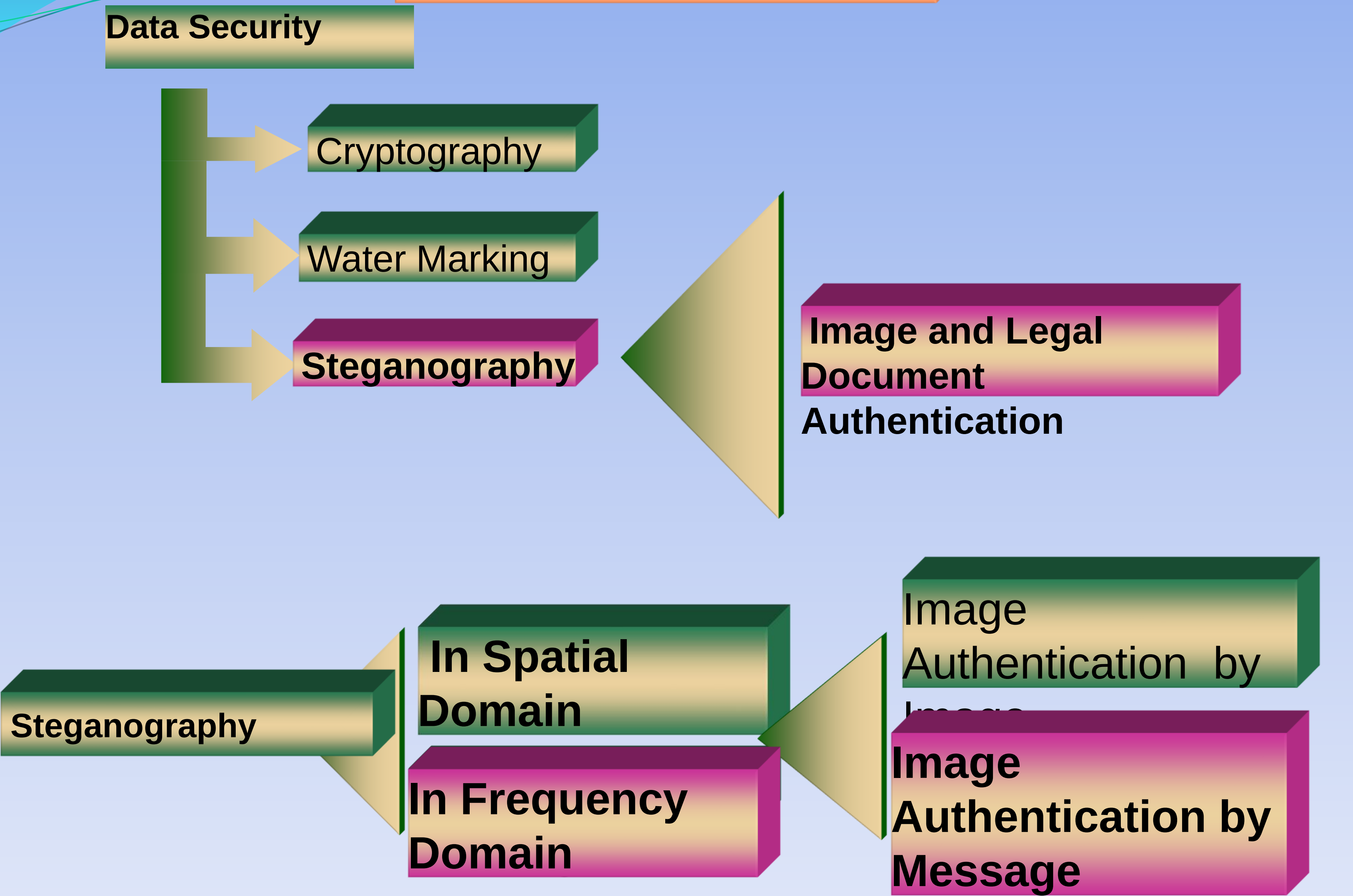


Fig. 5.19 *Verification of the CA signature on a certificate*

PROBLEM DOMAIN



STEGANOGRAPHY

STEGANOGRAPHY

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity (darkness).

SECRET COMMUNICATION

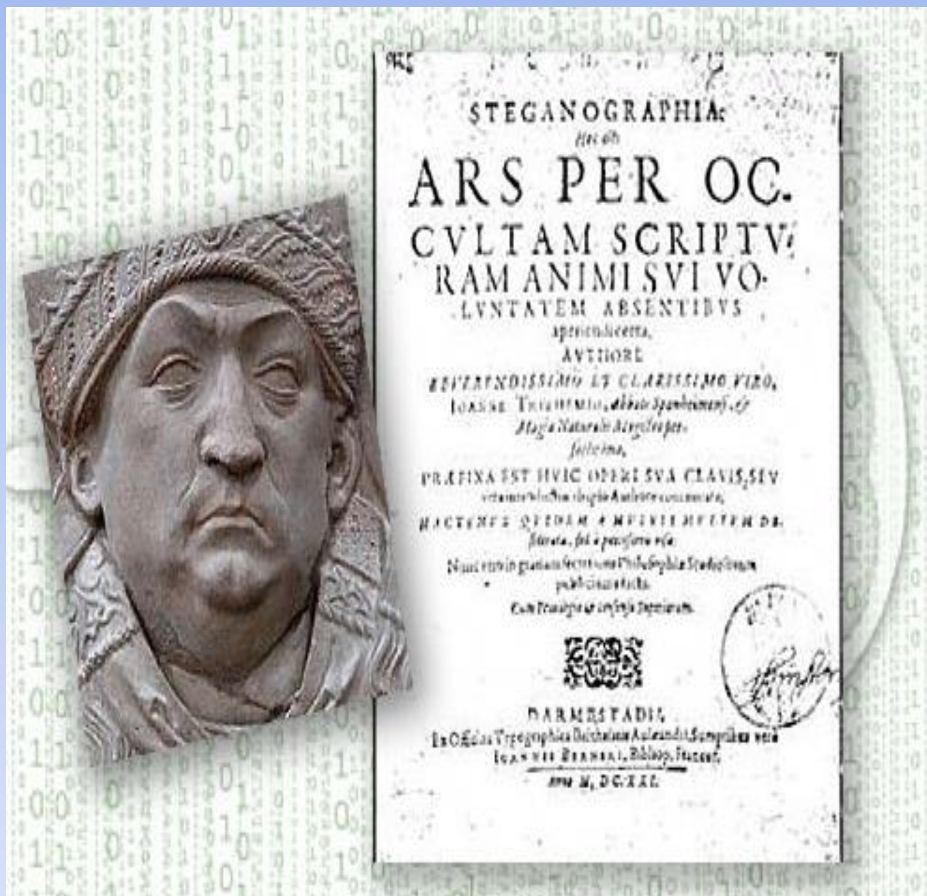
SECRET DATA TRANSFER

IMAGE AUTHENTICATION

1. Greatly reduces the risk of information being leaked in transit. (SECRET COMMUNICATION).
2. Ownership protection through Watermarking. (IMAGE AUTHENTICATION).
3. Semitransparent logos are commonly added to TV programs by broadcasting networks.

SECRET COMMUNICATION

Brief history of how the art and science has evolved.



The word steganography came from a 15th century work called Steganographia by a German abbot named Trithemius. On the face of it, the three books were about magic, but they were also contained an encrypted treatise on cryptography – so Steganographia was itself a case of steganography.

Second Example

An ancient Greek named Histaiaeus was fomenting revolt against the king of Persia and needed to pass along a message secretly. He shaved the head of a slave, tattooed the message on his scalp, then sent him on his way when his hair grew back in. Recipients of the message shaved his head again to read the alert. The Greeks used the same trick shaving and writing on the belly of a rabbit.

Third Example

Sometime in the 5th century B.C., an exiled Greek named Demaratus wrote a warning that the Persians planned to attack Sparta. He wrote the message on the wooden backing for a wax tablet, then hid it by filling in the wood frame with wax so it looked like a tablet containing no writing at all. The wife of the Spartan king divined that there was a message behind the wax, so they scraped it off and got the warning in time to set up a desperate defence at Thermopylae, incidentally giving modern screenwriters the plot for the movie The 300.

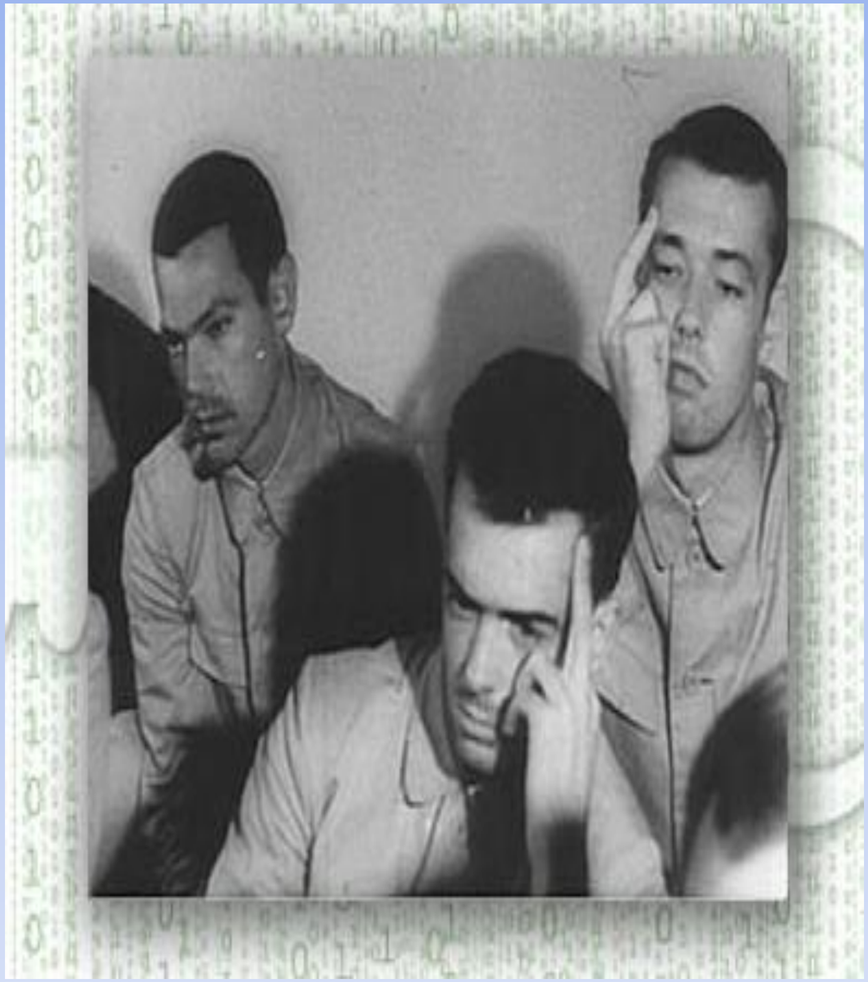
Fourth Example

Encoded messages have been knitted into sweaters and other garments. In this example, the blue dotted lines are Morse Code for, "My girlfriennd knit this." Yes, the sweater has a typo – an extra n in girlfriend - according to the woman who knitted it.

FIFTH EXAMPLE

During World War II, microdots - miniaturized photos that can be hidden in plain sight, then read using magnifiers – were used by spies to carry data out of enemy countries. Here the microdot circled in red piggybacks on a watch face. Blown up, it reveals a message written in German.

Sixth Example



When the USA Pueblo was captured by North Korea in 1968, the crew was forced to pose for propaganda photos to demonstrate they were being well treated. Their finger gestures are a form of steganography that sends a message Americans could decrypt right away, the North Koreans, not so quickly.

Seventh Example

Digital photo steganography uses code fields for unimportant bits as places to hide encoded messages or images. While such manipulation might slightly alter the quality of the original image, it generally goes unnoticed by the naked eye. In these pictures, the image of the cat has been embedded in the image of the branches against the sky.

STEGANOGRAPHY

- ❖TRADITIONAL STEGANOGRAPHY.

- ❖MODERN STEGANOGRAPHY.

STEGANOGRAPHIC PROTOCOLS

- Pure Steganography
- Secret Key Steganography
- Public Key Steganography

Applications Steganography

1. Usage in modern printers

   Steganography is used by some modern printers, including HP and Xerox brand color laser printers. Tiny yellow dots are added to each page. The dots are barely visible and contain encoded printer serial numbers, as well as date and time stamps.

2. Usage in Legal document

   Steganography can be used for digital watermarking, where a message (being simply an identifier) is hidden in an image so that its source can be tracked or verified, copyright protection, Bank draft, cheque and many other.

3. Steganography in audio can be used with mobile phone.

Rumored Usage in Terrorism

Rumors about terrorists using steganography started first in the daily newspaper USA Today on February 5, 2001 in two articles titled "Terrorist instructions hidden online" and "Terror groups hide behind Web encryption". In July of the same year, the information looked even more precise: "Militants wire Web with links to jihad".

DOCUMENT AUTHENTICATION

Technique to Authenticate

Tr

DOCUMENT AUTHENTICATION

We are Indian. We are proud for our country. We always like to look ahead with positive attitude and giving maximum effort to growth our country. We are so much strong in science and Technology.

Nabin Ghoshal

Tran

We are Indian. We are proud for our country. We always like to look ahead with **negative attitude** and giving **minimum effort** to growth our country. We are so **much weak in** science and Technology.

Nabin Ghoshal

DOCUMENT AUTHENTICATION

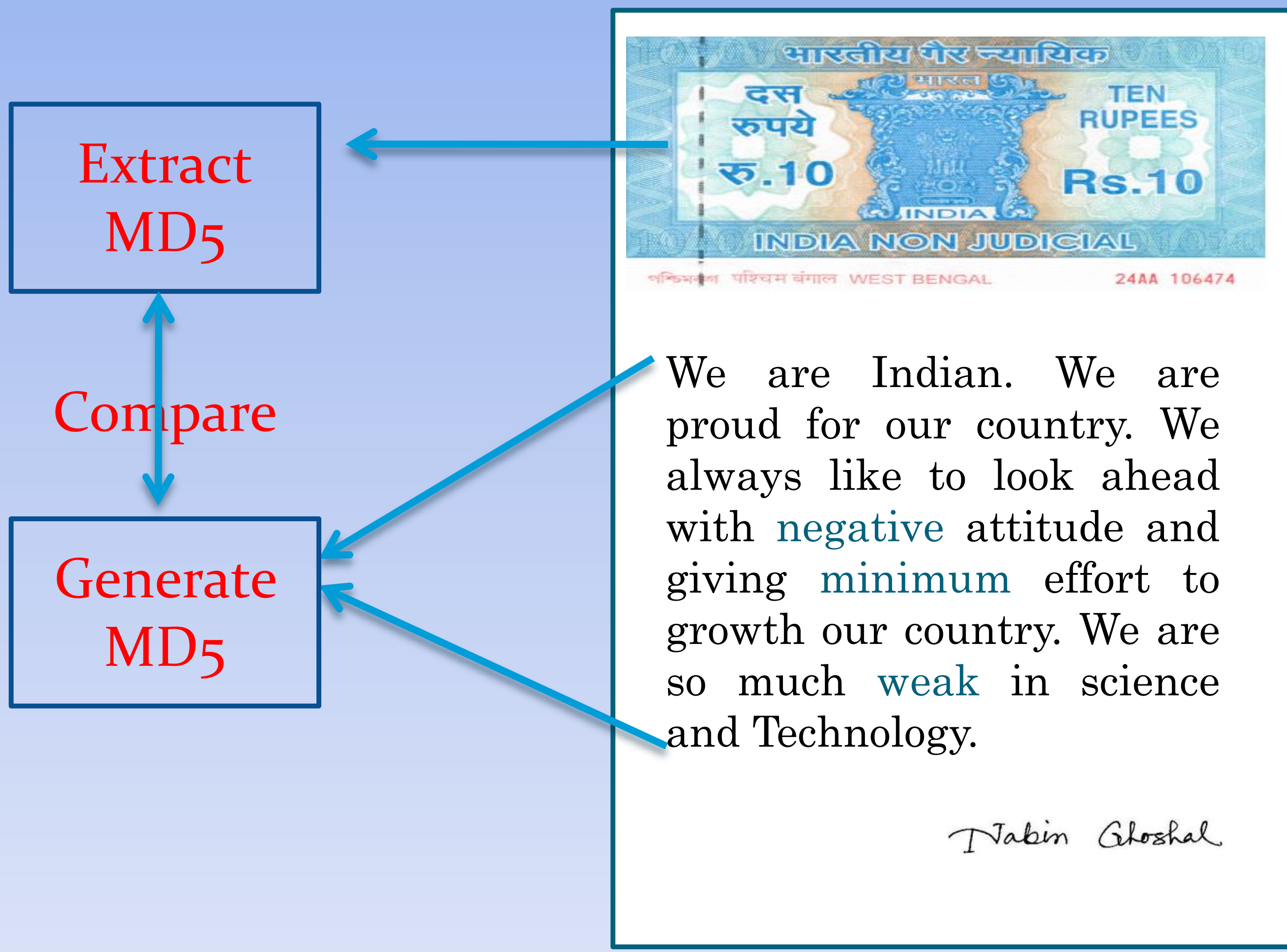


IMAGE AUTHENTICATION

Lena Image

Lena Image

SENDER SIDE OPERATION

IMAGE AUTHENTICATION

Embedded Lena Image

Original Secret Image

COMPARE

Extracted Image

RECEIVER SIDE OPERATION

Objectives of Image Steganography

- Data Hiding
- Secured message Transmission
- Invisible data transmission
- Ownership verification

Embedding/ Authentication

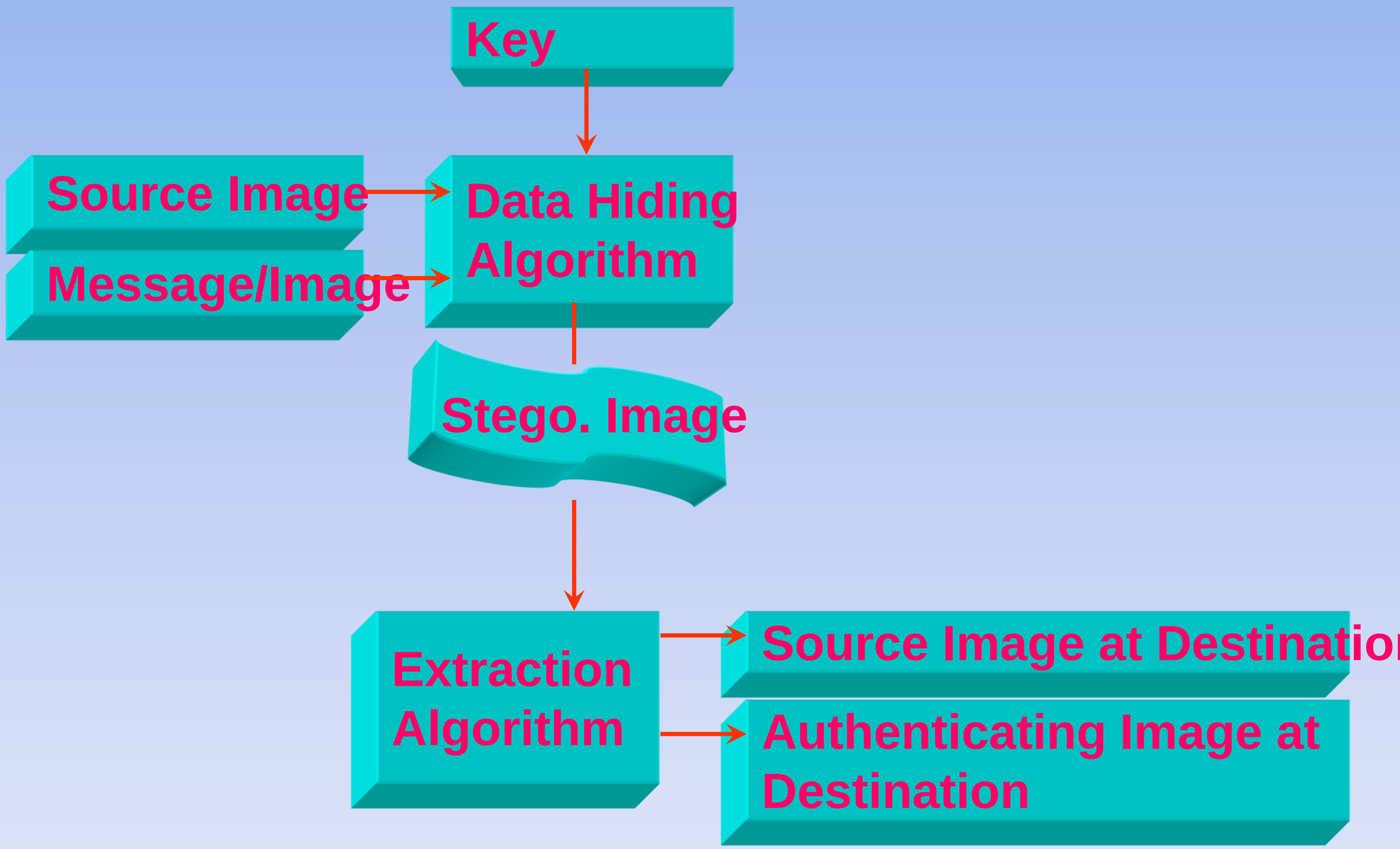
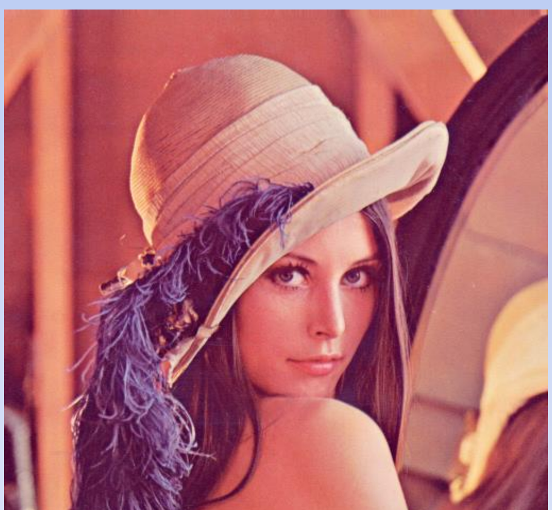


IMAGE STEGANOGRAPHY

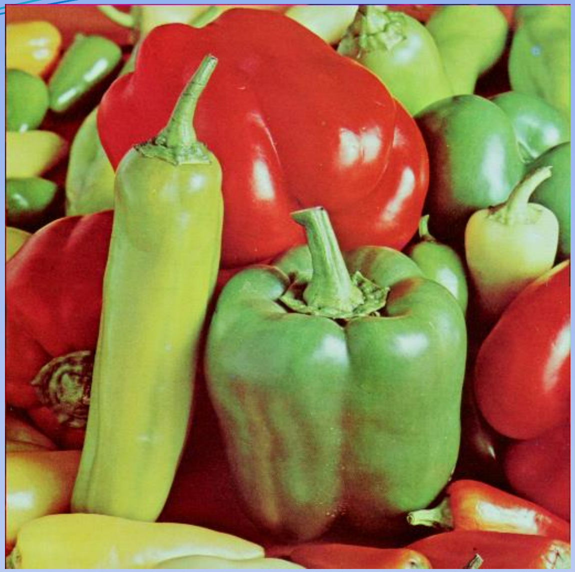
Source Image Lenna

Authenticating Image Earth



Authenticated Image Lenna

IMAGE STEGANOGRAPHY



Source Image Peppers

Authenticating Image

Embedded Image Peppers

TECHNICAL ASPECTS

SPATIAL DOMAIN LSB STEGONAGRAPHY

LSB (Least Significant Bit)

149	13	201
150	15	202
159	16	203

10010101 00001101 11001001
10010110 00001111 11001010
10011111 00010000 11001011

HIDE --- 365

1 0 1 1 0 1 1 0 1

HIDE --- 365

1 0 1 1 0 1 1 0 1

10010101	00001101	11001001
10010110	00001111	11001010
10011111	00010000	11001011

Changed data

1001010**1**	0000110**0**	1100100**1**
1001011**1**	0000111**0**	1100101**1**
1001111**1**	0001000**0**	1100101**1**

Thus, we have successfully hidden 9 bits in 9 bytes but at a cost of only changing 4bit, or roughly 50%, of the LSBs.

FREQUENCY DOMAIN STEGONAGRAPHY

- DISCRETE FOURIER TRANSFORMED
- DISCRETE COSINE TRANSFORMED
- DISCRETE WAVELET TRANSFORMED
- Z-TRANSFORMED

MIXED DOMAIN STEGONAGRAPHY

- **SPATIAL DOMAIN**
- **FREQUENCY DOMAIN**

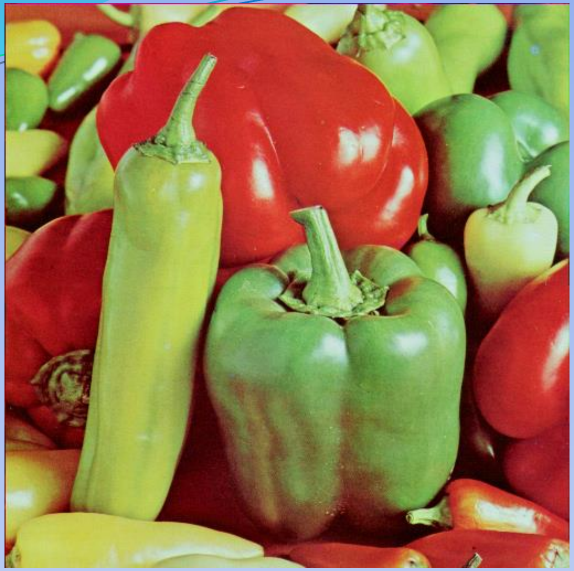
BOTH   DOMAINS ARE USED IN THIS STEGONAGRAPIC PROCESS

TRANSFORMED TECHNIQUED

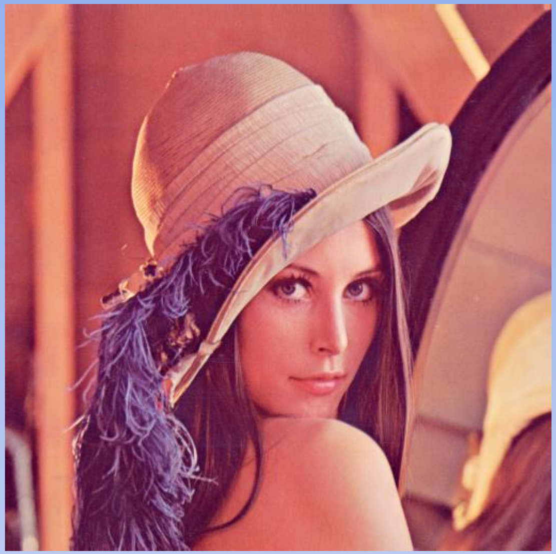
SPECIFICATIONS

- Embedding is done in frequency components
- Source image 512 x 512
- Authenticating image 128 x 128
- Embedding done on Real components

IMAGE STEGANOGRAPHY



Source Image Peppers



Source Image Lenna

DFT and IDFT

$$F(u,v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

where u = 0 to M – 1 and v = 0 to N-1.

$$f(x,y) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v)\, e^{j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

where x = 0 to M – 1 and y = 0 to N-1.

$$F(u, v) = \frac{1}{2}\sum\sum f(x, y)[\cos 2\Pi(ux / 2 + vy / 2) - i \sin 2\Pi(ux / 2 + vy / 2)] = \sum\sum f(x , y)[\cos\Pi (ux + vy) - i \sin\Pi(ux + vy)]$$

where x, y are spatial variables and u, v are frequency variables

Formulation and Motivation of DFTMCIAWC

2 x 2 mask values are {a, b, c, d} from the source image. The DFT values are F(a) = ½ (a + b + c + d) = W (say), F(b) = ½ (a – b + c – d) = X (say), F(c) = ½ (a + b – c – d) = Y (say), and F(d) = ½ (a – b – c + d) = Z (say) for four a, b, c, and d spatial values and W, X, Y and Z are frequency values respectively.

Formulation and Motivation of DFTMCIAWC

Spatial Domain to Frequency Domain (DFT)

$F(a) = ½ (a + b + c + d) = W$

$F(b) = ½ (a – b + c – d) = X$

$F(c) = ½ (a + b – c – d) = Y$

$F(d) = ½ (a – b – c + d) = Z$

DFT to Spatial Domain (IDFT)

$F^{-1}(W) = ½ (W + X + Y + Z)$

$F^{-1}(X) = ½ (W – X + Y – Z)$

$F^{-1}(Y) = ½ (W + X – Y – Z)$

$F^{-1}(Z) = ½ (W – X – Y + Z)$

Problems and Solutions of DFTMCIAWC

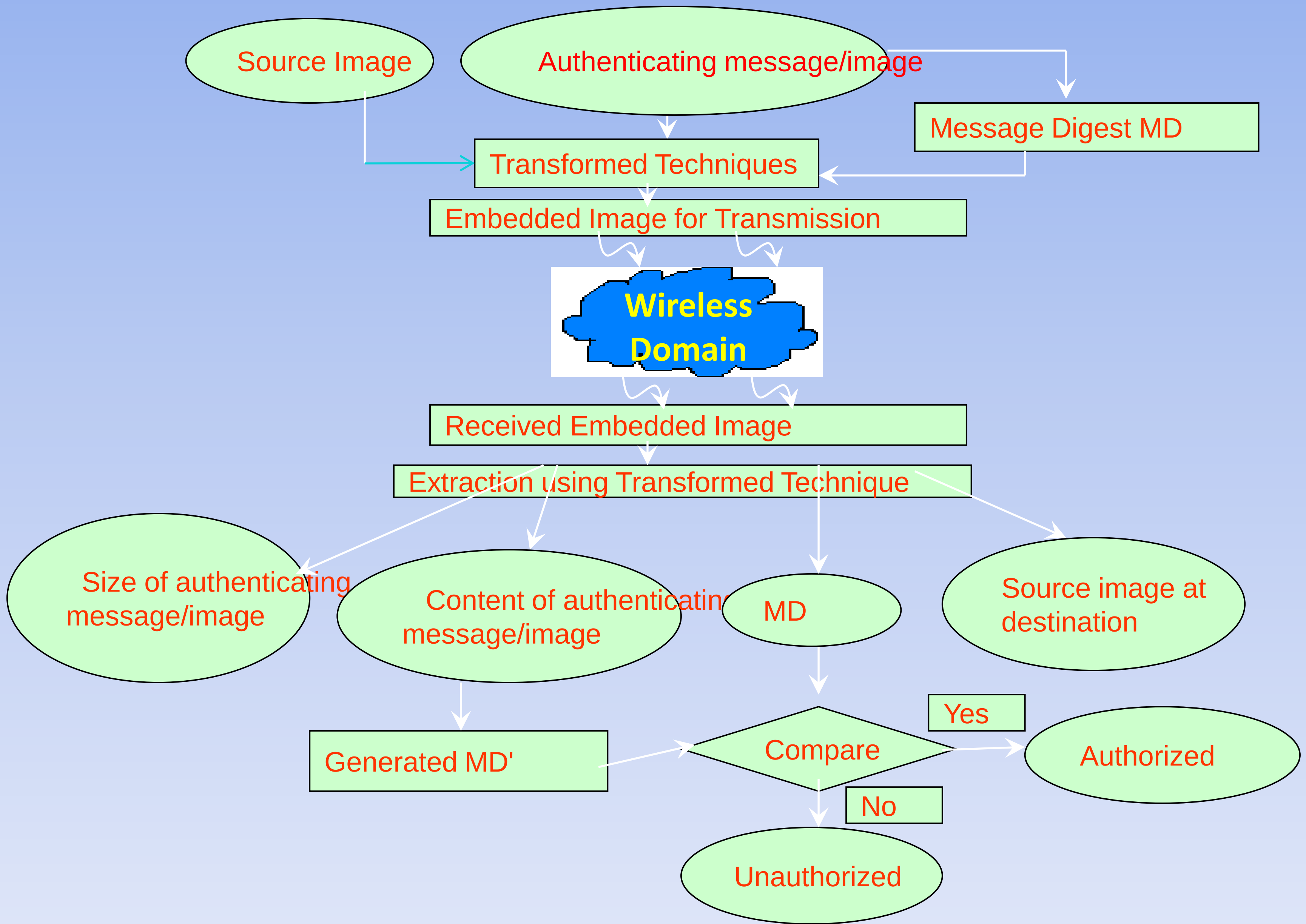
A. The converted value may by negative(-ve).

B. The converted value in spatial domain may be a fractional number.

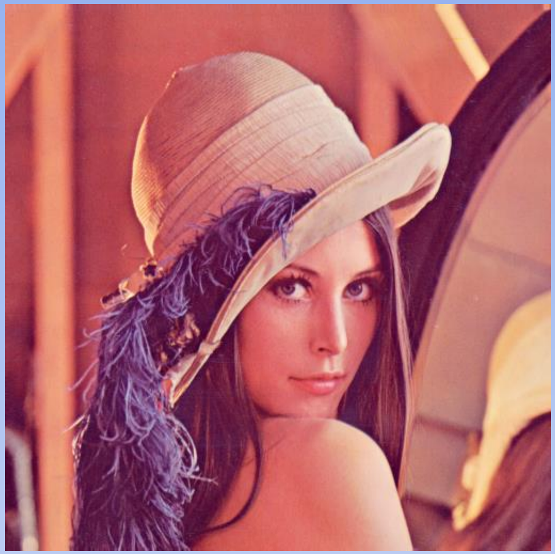
C. The converted value may be greater the maximum value (i.e. 255).

Solutions: Re-adjustment is done on 1^{st} frequency component where embedding is not done.

Flow Diagram of FD Techniques



Visual Interpretation



Source Image Lenna

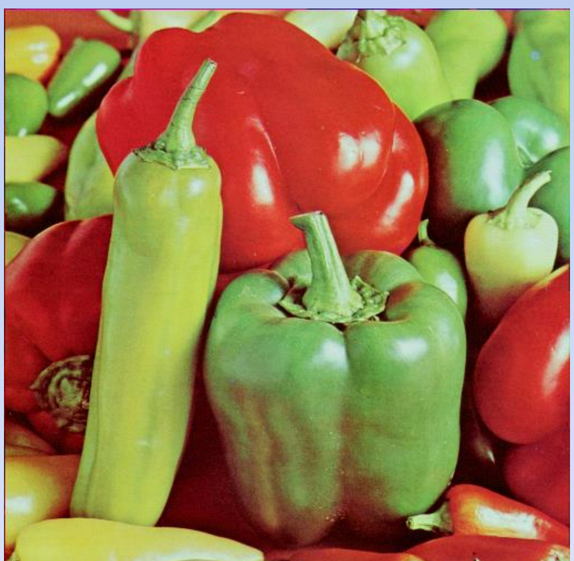
Authenticating Image Earth

Authenticated Image Lenna

Results & Visual Interpretation using DFTMCIAWC

Source Image Peppers

Authenticating Image



Embedded Image using DFTMCIAWC

Example (Insertion rule N % S)

Character	ASCII Code
S	01010011
A	01000001
C	01000011
H	01001000
I	01001000
N	01001110

Secrete Data

15	36	19	45
17	20	55	78
11	10	16	80
4	6	18	91
0	34	15	54
30	15	12	70

Source Image

15	36
17	20

1st mask of source image matrix

DFT →

44.0	-12.0
7.0	-9.0

Real Part

0.0	-0.0
-0.0	-0.0

Imaginary Part

Example

44.0	-13.0
5.0	-10.0

Embedded real part

IDFT

13.0	36.0
18.0	21.0

0.0	-0.0
-0.0	-0.0

Imaginary Part

0.0	-0.0
-0.0	-0.0

Computational Aspect

- Source Colour Image Dimension is m × n bytes
- Authenticating Colour Image size is p × q bytes
  - Source image of size m × n is able to embed 2*((m × n)*3/4) bits of authenticating Data

    Where 8 *(p × q)*3 Bits <= 3 * m × n bytes

- **Total computation for square Authenticating image is 24 * (p * p) = O(p^2)**

WAVELET TRANSFORM

WAVELET TRANSFORM

Wavelet Function ψ (t)
(i.e. Mother wavelet)

Scaling Function φ (t)
(i.e. Father wavelet)

The first DWT was invented by Hungarian mathematician Alfred Haar in the year of 1909

HAAR WAVELETS

$$\varphi(t`) = \varphi(2t) + \varphi(2t - 1)$$

$$\psi(t`) = \varphi(2t) - \varphi(2t - 1)$$

t = 1, 2, 3, 4...

Time

If t =1: 2t → 2; 2t-1 → 1

If t =2: 2t → 4; 2t-1 → 3

If t =3: 2t → 6; 2t-1 → 5

If t =4: 2t → 8; 2t-1 → 7

HAAR WAVELETS

POSITION

1	2	3	4	5	6	7	8	9	10
2	9	6	2	5	1	8	5	4	7

$$\varphi(t`) = \varphi(2t) + \varphi(2t - 1)$$

$$\varphi(t`) = 9 + 2 = 11$$

$$\psi(t`) = \varphi(2t) - \varphi(2t - 1)$$

$$\psi(t`) = 9 - 2 = 7$$

HAAR WAVELETS

$$\varphi(t`) = \varphi(2t) + \varphi(2t - 1)$$

$$\psi(t`) = \varphi(2t) - \varphi(2t - 1)$$

2	9	6	
5	1	8	
4	7	3	
2	9	6	

$\varphi(t)$

11	8
6	13
11	7
11	8

$\psi(t)$

7	-4
-4	-3
3	1
7	-4

Range → 0 to 255

HAAR WAVELETS

NORMALIZATION VALUE

For Haar transformation we have two set of normalization value

+- ½ OR $\sqrt{2}$

2	9	6	2
5	1	8	5
4	7	3	4
2	9	6	2

$\varphi(t)$

11	8
6	13
11	7
11	8

→

5.5	4
3	6.5
5.5	3.5
5.5	4

+- ½

$\psi(t)$

7	-4
-4	-3
3	1
7	-4

→

-3.5	2
2	1.5
-1.5	-0.5
-3.5	2

HAAR WAVELETS

NORMALIZATION VALUE

For Inverse Haar transformation we have two set of de-normalization value

+- 1 OR $\sqrt{2}$

5.5	4
3	6.5
5.5	3.5
5.5	4

$\varphi(t)$

+- 1

-3.5	2
2	1.5
-1.5	-0.5
-3.5	2

$\psi(t)$

2	9	6	
5	1	8	
4	7	3	
2	9	6	

HAAR WAVELETS

Figure 2 : Original Lena Image & vertical transformation.

DISCRETE COSINE TRANSFORM

Forward Transformation

Dimension of Image

x= row order
y=column order

Pixel Intensity Value

$$C(u,v) = \frac{1}{\sqrt{2N}} * \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+2)v\pi}{2N}\right]$$

$$\alpha(i) = \frac{1}{\sqrt{2}} \; if \; i \; is \; 0, else \; 1 \; if \; i > 0$$

Inverse transformation

Dimension of Image

x= row order
y=column order

Frequency Components

$$f(x,y) = \frac{1}{\sqrt{2N}} \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} a(u)a(v)C(u,v)\cos\left[\frac{[2x+1]u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

$$a(i) = \frac{1}{\sqrt{2}} \; if\ i\ is\ 0,\ else\ 1\ if\ i > 0$$

Pictorial Representation

A	**B**
C	**D**

DCT →

P	**Q**
R	**S**

Origi[illegible]el values
f(x, y)

Frequency components after DCT

C(u, v)

Pixel representation with its DCT coefficient after forward transformation

Result of ABCD to PORS

Set No	Original Pixel Value		DCT Components	
1	147	56	211.00000	54.88400
	119	100	-8.135871	35.993118
2	47	90	121.999992	-37.088787
	38	69	14.927637	-5.989834
3	138	93	187.00000	39.894455
	89	54	43.895718	4.950134

Z - TRANSFORM

Generalized Formula of Z

10	25
30	20

Taking ω = 0

X(Z) =10[Cosω0 – jSin ω0] +
25[Cosω1 – jSin ω1] +
30[Cosω2 – jSin ω2] +
20[Cosω3 – jSin ω3]
=85

Taking ω = π/2

X(Z) =10[Cosπ/2*0 – jSin π /2*0] +
25[Cos π /2*1 – jSin π /2*1] +
30[Cos π /2*2– jSin π /2*2] +
20[Cos π /2*3 – jSin π /2*3]
=-20 -5j

10	25
30	20

Taking ω = **π**

X(Z) =10[Cos **π*0** – jSin **π*0**] +
25[Cos **π*1** – jSin **π*1**] +
30[Cos **π*2** – jSin π*2] +
20[Cos **π***3 – jSin **π***3]
=-5

Taking ω = 3π/2

X(Z) =10[Cos3**π**/2*0 – jSin3**π** /2*0] +25[Cos3**π**/2*1 – jSin 3**π** /2*1] +
30[Cos3**π** /2*2– jSin 3**π** /2*2] +
20[Cos3**π** /2*3 – jSin 3 **π** /2*3]
=-20+5j

COVER IMAGE

10	25
30	20

TRANSFORMED COEFFICIENTS

85	--20- 5J
-5	-20+5J

INVERSE TRANSFORM

85	--20- 5J
-5	-20+5J

Taking ω = 0

X(Z) =1/4[85[Cosωn +jSin ωn] - 20[Cosωn + jSin ωn] -5j[Cosωn + jSinωn] -5[Cosωn + jSinωn] - 20 [Cosωn + jSinωn] +5j[Cosωn + jSinωn]]=1/4[85-20-5J-5-20+5 j]=1/4[40]=10

Taking ω = **π/2**

X(Z)=1/4[85[Cos**π/2*0+j**Sin**π/2*0**]-20[Cos**π/2*1+**jSin**π/2*1**]- 5j[Cos**π/2*1+**jSin**π/2*1**]-5[Cos **π/2*2** + jSin **π/2***2] -20[Cos **π/2***3 +jSin **π/2***3]+ 5j[Cos **π/2***3+jSin **π/2***3]

=25

85	--20- 5J
-5	-20+5J

Taking ω = π

X(Z) =1/4[85[Cos π*0 + jSin π*0] -20[Cos π *1 + jSin π*1] -5j[Cos π*1 + jSin π*1]-5 [Cos π *2+ jSin π *2] - 20[Cosπ*3+ jSinπ *3]+5j[Cos π *3 + jSin π *3]]

=30

Taking ω = 3π/2

X(Z) =1/4[85[Cos3**π**/2*0+jSin3**π** /2*0] -20[Cos3**π**/2*1+jSin 3**π** /2*1] -5j[Cos3**π**/2*1+jSin 3**π** /2*1] -5[Cos3**π** /2*2+ jSin 3**π** /2*2] - 20[Cos3**π** /2*3 + jSin 3 **π** /2*3]

=20

10	25
30	20

ORIGINAL MATRIX REGENERATED THROUGH REVERSE TRANSFORM

TRANSFORM MATRIX

85	--20- 5J
-5	-20+5J

Let 85 is the median value of the block

Convert it to binary:

1010101

Embedding

85	--20- 5J
-5	-20+5J

Source Stream

85=1010101

Secrete Information 'S' is

1010011

Embed a bit into Fourth LSB

Embedded Stream:1011101

New Generation(GA Based Tuning)

Source stream:1010101=85

85	--20-5J
-5	-20+5J

One bit from Secrete Information 'S' (1010011) is 1 has been embedded into Fourth LSB

Embedded Stream:1011101

Pixel Value after embedding is:93

Difference:93-85=8

As next bit of embedded position is 1, flip all bits right to embedded bit to zero

Handled Embedded pixel:1011000=88

Original Pixel:85

Differenec:88-85 = 3 which is minimum

COVER IMAGE

10	25
30	20

TRANSFORMED COEFFICIENTS

85	--20- 5J
-5	-20+5J

EMBEDDED COEFFICIENTS

93	--20- 5J
-5	-20+5J

GA BASED ADJUSTMENT

88	--20- 5J
-5	-20+5J

GA BASED ADJUSTMENT

88	--20- 5J
-5	-20+5J

EMBEDDED EINVERSE TRANSFORMED

10	26
30	20

EMBEDDED EINVERSE TRANSFORMED

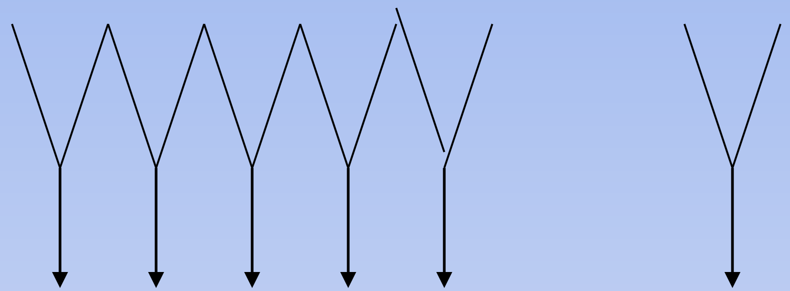
10	26
30	20

GA BASED CROSSOVER

12	25
24	18

TRIANGULARISATION(XNOR)

$S^j = s^j_0 \quad s^j_1 \quad s^j_2 \quad s^j_3 \quad s^j_4 \quad s^j_5 \quad \ldots \quad s^j_{n-(j+2)} s^j_{n-(j+1)}$



$S^{j+1} = s^{j+1}_0 s^{j+2}_1 s^{j+3} s^{j+4}_3 s^{j+5}_4 \ldots \qquad s^j_{n-(j+2)}$

Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$S^{0}_{0}\ S^{1}_{0}\ S^{2}_{0}\ S^{3}_{3}\ S^{4}_{0}\ \ldots\ S^{n-2}_{0}\ S^{n-1}_{0}$	Taking all the MSBs starting from the source block till the last block generated
010	$S^{n-1}_{0}\ S^{n-2}_{0}\ S^{n-3}_{0}\ S^{n-4}_{0}\ S^{n-5}_{0}\ \ldots\ S^{1}_{0}\ S^{0}_{0}$	Taking all the MSBs starting from the last block generated till the source block
011	$S^{0}_{n-1}\ S^{1}_{n-2}\ S^{2}_{n-3}\ S^{3}_{n-4}\ S^{4}_{n-5}\ \ldots\ S^{n-2}_{1}\ S^{n-1}_{0}$	Taking all the LSBs starting from the source block till the last block generated
100	$S^{n-1}_{0}\ S^{n-2}_{1}\ S^{n-3}_{2}\ S^{n-4}_{3}\ S^{n-5}_{4}\ \ldots\ S^{1}_{n-2}\ S^{0}_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

Different Target Blocks generated using TE for S = 10010101

Source Block S	Target Block Corresponding to Serial No.	Target Block T
10010101	001	10010101
	010	10101001
	011	10111101
	100	10111101

1 0 0

0 1

0

0 1 0

0 0

1

GA BASED CROSSOVER

12	25
24	18

GA BASED MUTATION

13	24
26	16

GA BASED CROSSOVER

12	25
24	18

BINARY OF GA BASED CROSSOVER

00001100	00011001
00011000	00010010

BINARY OF GA BASED CROSSOVER

00001100	00011001
00011000	00010010

BINARY OF GA BASED MUTATION

00001101	00011000
00011010	00010000

GA BASED MUTATION

13	24
26	16

EMBEDDED REVERSE TRANSFORMED MATRIX

10	26
30	20

CROSSOVER

12	25
24	18

MUTATION(Final)

13	24
26	16

SOURCE MATRIX

10	26
30	20

Some Open Directions

- Extension to more bits insertion within ea

Byte of pixel information in Color image.

- Extension to chose any dimension of Mas
- Extension to change the direction of acce

of Image Mask (to column major order).

Secure Socket Layer (SSL)

SSL is an Internet Protocol for secure exchange of information between a webbrowser and a web server. Two major functions:

- **Authentication**
- **Confidentiality**

Secure Socket Layer (SSL)

Application Layer

SSL Layer

Transport Layer

Internet Layer

Data link Layer

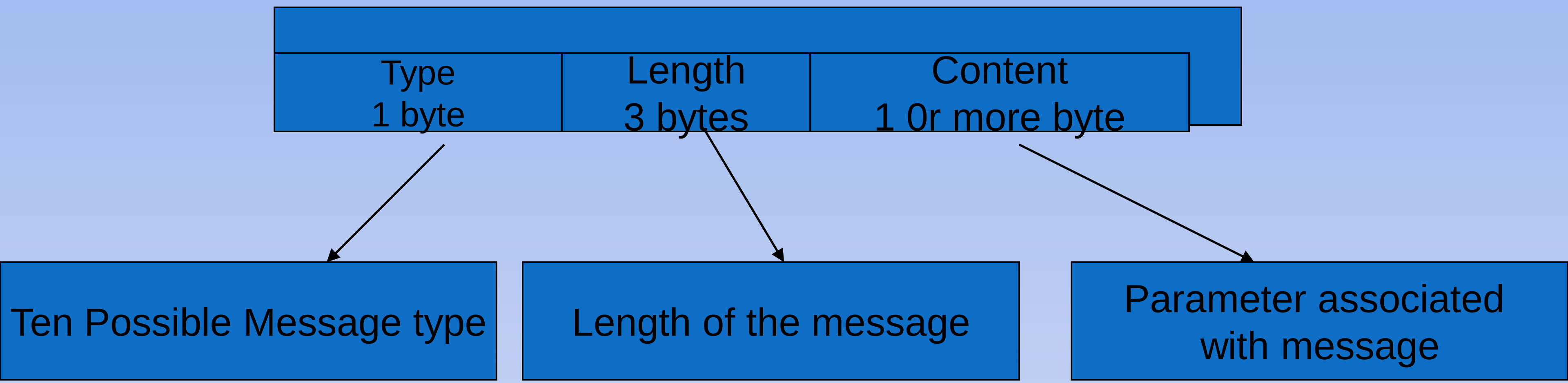
Physical Layer

- **SSL Encrypt Application Layer Data**
- **Other Layer are associated with Header only**

Three Subprotocol of SSL Layers are

- **Handshaking Protocol**
- **Record Protocol**
- **Alert Protocol**

Handshaking SubProtocol Structure



WINDOWS 2000 USER AUTHENTICATION(NTLM)

•User gets screen for Login and enter uer ID and Pass Word

•The User's Computer Compute a Message Digest of the password and destroyed the password

•The client sends the user ID in plain text to the server

•The server send a 16 byte random number challenge to the client

•The client encrypts the random number challenge with message digest of the password

WINDOWS 2000 USER AUTHENTICATION(NTLM)

- **Client send this random challenge as response to the server.**
- **The Server forward user ID, original random challenge and the client's response to a special server called domain controller which keep track of Id, Password and digest**
- **Domain controller computes message digest and compare it with the others.**
- **If matches then user authentication is successful.**

Factors considered for Evaluating Proposed Techniques

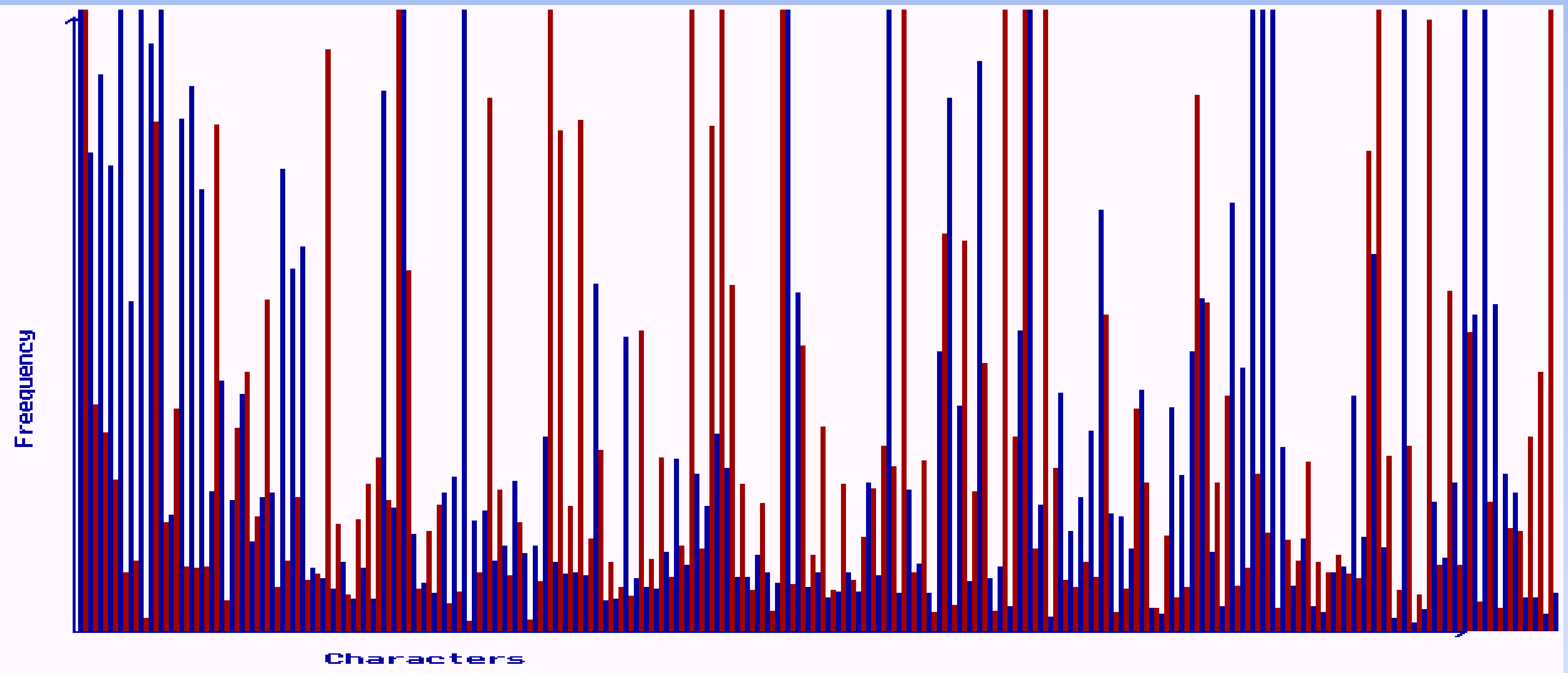
Several factors have been considered to evaluate the proposed techniques. These include the following:

- **Frequency Distribution Test**
- **Chi Square Test**
- **Analysis of the Key Space**
- **Computation of the Encryption/Decryption Time**
- **Comparison of Performance with the RSA System**

Results for *.exe* files in tabular form that shows the time of encryption, time for decryption and the Chi Square values of nine executable files

Source File	Encrypted files	Source Size	Encryption Time	Decryption Time	Chi Square Value
tlib.exe	a1.exe	37220	0.3297	0.2198	9.92
maker.exe	a2.exe	59398	0.6044	0.3846	17.09
unzip.exe	a3.exe	23044	0.2747	0.1648	13.95
rppo.exe	a4.exe	35425	0.3846	0.2747	9.92
prime.exe	a5.exe	37152	0.4945	0.3297	14.86
triangle.exe	a7.exe	36242	0.4396	0.2198	9.92
ping.exe	a8.exe	24576	0.2747	0.1648	17.39
netstat.exe	a9.exe	32768	0.3297	0.2198	17.39
clipbrd.exe	a10.exe	18432	0.2198	0.1648	9.92

A segment of frequency distribution for characters in tlib.exe and its encrypted file

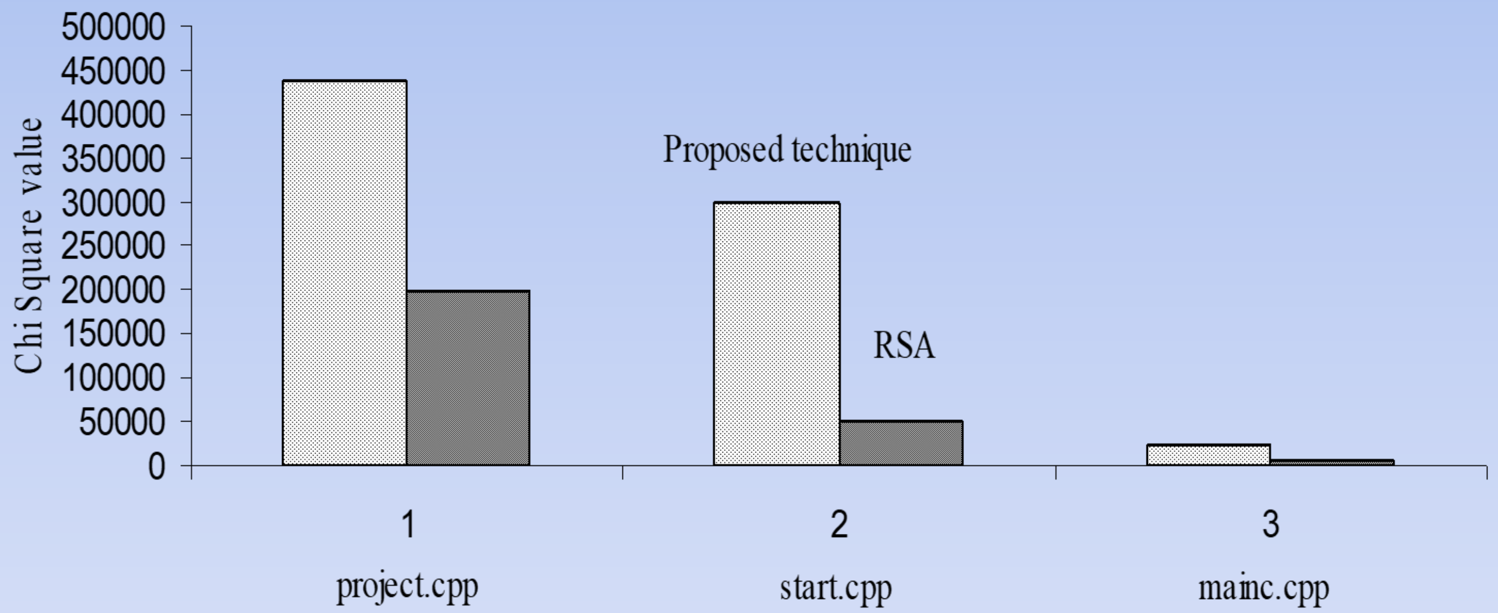


Blue lines indicate the occurrences of characters in the source file and red lines indicate the same in the corresponding encrypted file

Comparative results between RPMS technique and RSA technique for .cpp files for their Chi Square values and corresponding degree of freedom

Source file	Encrypted files using RPMS technique	Encrypted files using RSA technique	Chi Square value for RPMS technique	Chi Square value for RSA technique	Degrees of freedom
bricks.cpp	*a1.cpp*	*cpp1.cpp*	113381	200221	88
project.cpp	*a2.cpp*	*cpp2.cpp*	438133	197728	90
arith.cpp	*a3.cpp*	*cpp3.cpp*	143723	273982	77
start.cpp	*a4.cpp*	*cpp4.cpp*	297753	49242	88
chartcom.cpp	*a5.cpp*	*cpp5.cpp*	48929	105384	84
bitio.cpp	*a6.cpp*	*cpp6.cpp*	9101	52529	70
mainc.cpp	*a7.cpp*	*cpp7.cpp*	22485	4964	83
ttest.cpp	*a8.cpp*	*cpp8.cpp*	1794	3652	69
do.cpp	*a9.cpp*	*cpp9.cpp*	294607	655734	88
cal.cpp	*a10.cpp*	*cpp10.cpp*	143672	216498	77

Files with better result in proposed technique than existing RSA technique in terms of Chi Square values



Proposal of Key Format

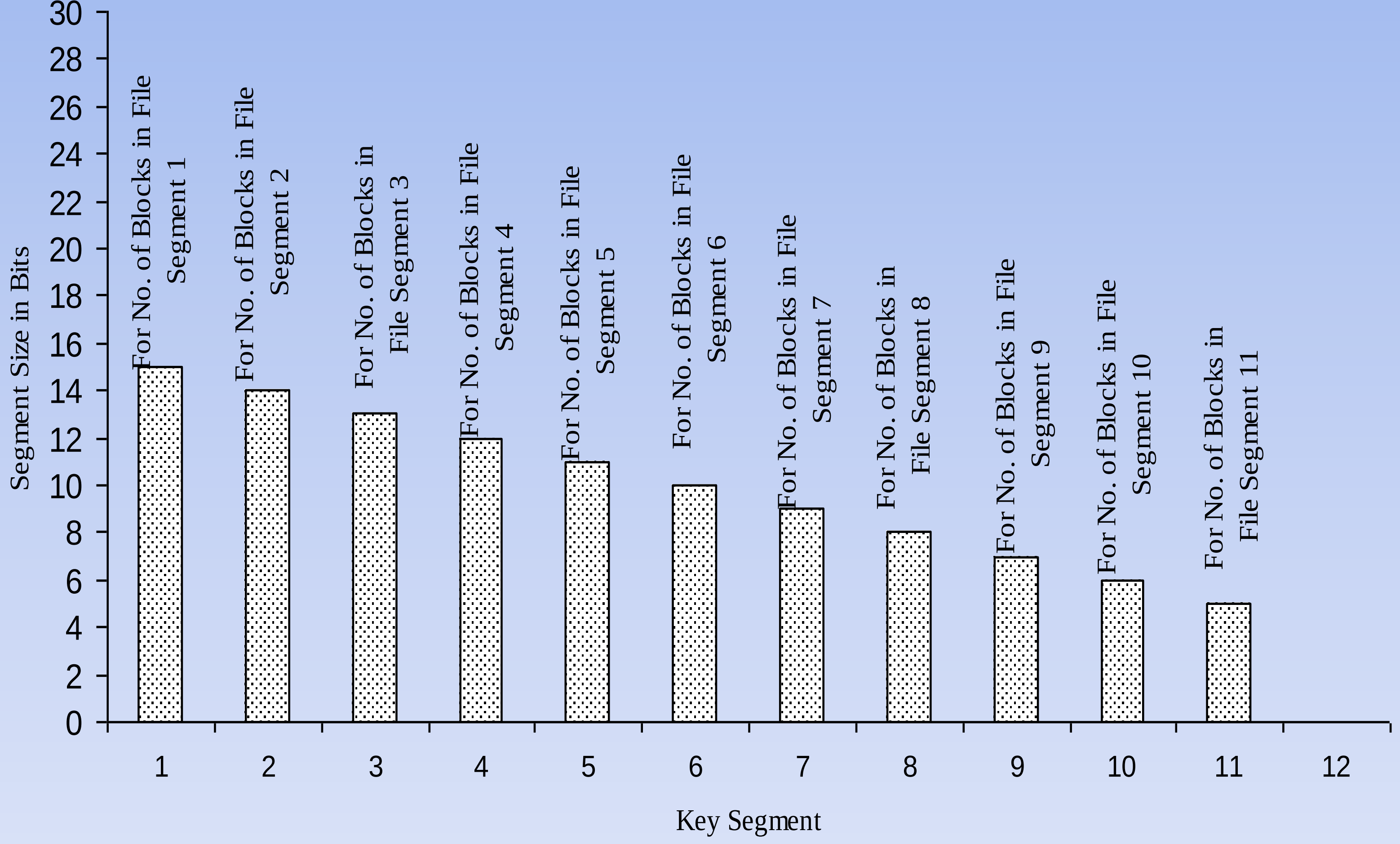
A 110-bit key format consisting of 11 different segments has been proposed For the segment of the rank R, there can exist a maximum of $N = 2^{15-R}$ blocks, each of the unique size of $S = 2^{15-R}$ bits, R starting from 1 and moving till 11.

For different values of R, following segments are generated:

- Segment with R=1 formed with the first maximum 16384 blocks, each of size 16384 bits;
- Segment with R=2 formed with the first maximum 8192 blocks, each of size 8192 bits;
- Segment with R=3 formed with the next maximum 4096 blocks, each of size 4096 bits;
- Segment with R=4 formed with the next maximum 2048 blocks, each of size 2048 bits;
- Segment with R=5 formed with the next maximum 1024 blocks, each of size 1024 bits;
- Segment with R=6 formed with the next maximum 512 blocks, each of size 512 bits;
- Segment with R=7 formed with the next maximum 256 blocks, each of size 256 bits;
- Segment with R=8 formed with the next maximum 128 blocks, each of size 128 bits;
- Segment with R=9 formed with the next maximum 64 blocks, each of size 64 bits;
- Segment with R=10 formed with the next maximum 32 blocks, each of size 32 bits;
- Segment with R=11 formed with the next maximum 16 blocks, each of size 16 bits;

With such a structure, the key space becomes of 110 bits long and a file of the maximum size of around 44.74 MB

110-bit key format with 11 segments for RPMS Technique



Example of Key Generation-110 bit key

0	0	0	0	0	0	0	0	0	1	0	0	1	0	1

37 blocks/37bits

0	0	0	0	0	0	0	1	0	0	0	0	0	1

65 blocks/65bits

0	0	0	0	0	0	1	0	0	0	1	1	1

71 blocks/71bits

0	0	0	0	0	0	0	1	0	1	1	0

22 blocks/22bits

0	0	0	0	0	0	0	1	1	1	1

15 blocks/15bits

0	0	0	1	0	0	0	0	0	0

64 blocks/64bits

0	0	0	0	0	0	0	0	0

0 blocks/0bits

0	0	0	0	0	0	0	0

0	0	0	0	0	0	0

1	1	0	0	0	1

49 blocks/49bits

0	1	0	0	0

8 blocks/8bits

Total 37+65+71+22+15+64+49+8 blocks = 331 blocks

The Size of the file for this Session Key

Total 37+65+71+22+15+64+49+8 blocks = 331 blocks

and

37*37 + 65*65 + 71*71 + 22*22+ 15*15+ 64*64 + 49*49 + 8*8 = 17905 bits + 7 bits =17912 bits

Analysis

- The encryption time and the decryption time vary linearly with the size of the source file.
- There exist not much difference between the encryption time and the decryption time for a file, establishing the fact that the computation complexity of each of the two processes is of not much difference.
- For non-text files, such as *.exe*, *.com*, *.dll*, and *.sys* files there is no relationship between the source file size and the Chi Square value.
- Chi Square values for text files, such as *.cpp* files are very high and vary linearly with the source file size.
- Out of the different categories of files considered here, Chi Square values for .CPP files are the highest.
- The frequency distribution test applied on the source file and the encrypted file shows that the characters are all well distributed.

Chi Square values for this proposed technique and those for the RSA system highly compatible

The first two factors are considered to asses the degree of security of the proposed techniques against the cryptanalytic attack. Through the frequency distribution tests performed on the original as well as the encrypted files, the frequencies of all 256 characters in two files are shown graphically. Through the chi square tests performed on the original and the encrypted files, the non-homogeneity of the two files is tested.

The third factor plays an important role in attempting to tackle the Brute-force attack successfully. The key space of each technique has been attempted to enlarge reasonably to make the techniques computationally secure.

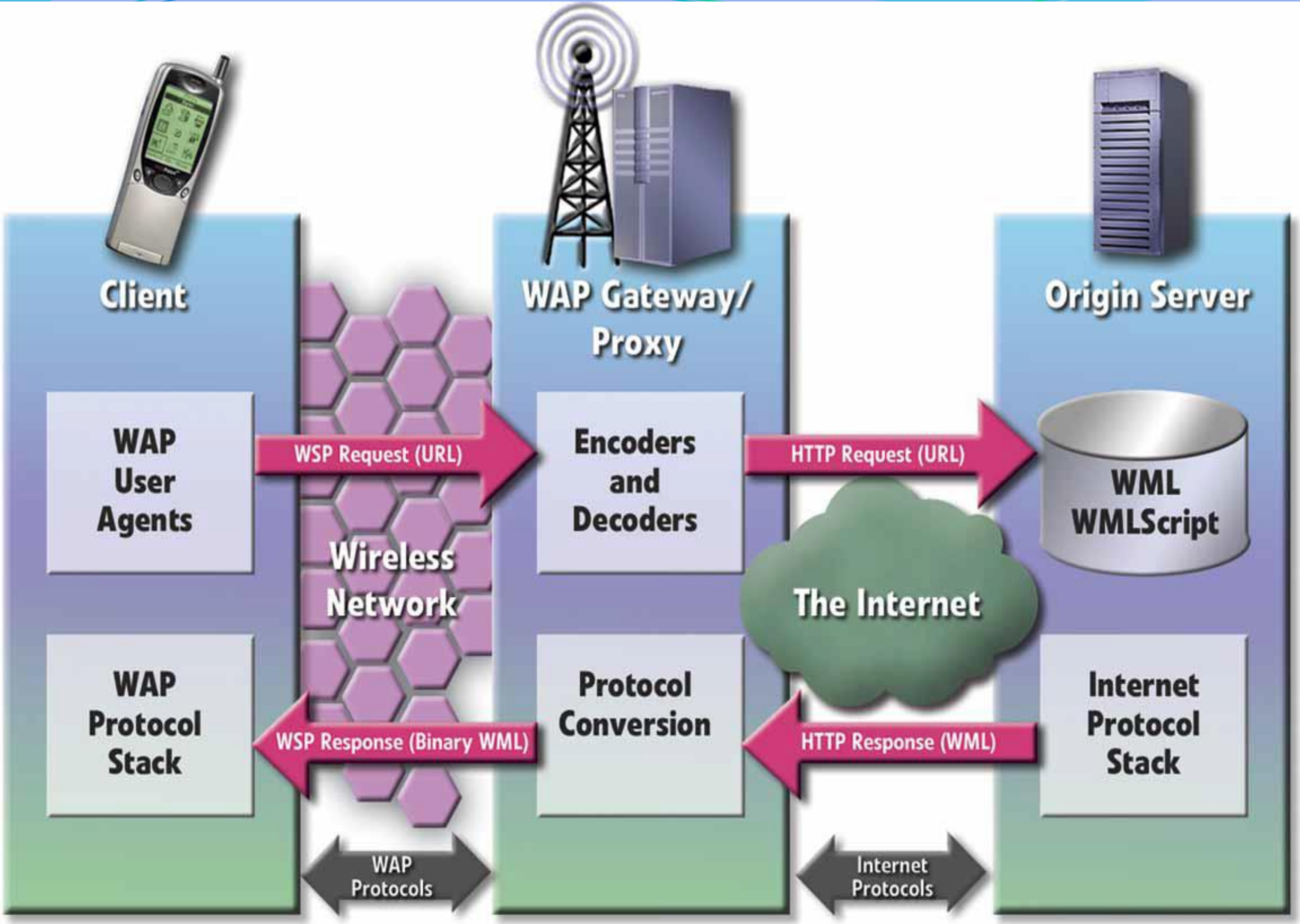
The forth factor plays an important role in assessing the efficiencies of the algorithms from the execution point of view. Here it has been attempted to establish a relationship between the size of the file being encrypted and the encryption/decryption time.

Wireless Application Protocol Security

WAP?

The WAP (Wireless Application Protocol) is a suite of specifications that enable wireless Internet applications; these specifications can be found at (http://www.wapforum.org). WAP provides the framework to enable targeted Web access, mobile e-commerce, corporate intranet access, and other advanced services to digital wireless devices, including mobile phones, PDAs, two-way pagers, and other wireless devices.

Model



Client
WAP User Agents
WAP Protocol Stack
WSP Request (URL)
WSP Response (Binary WML)
Wireless Network
WAP Protocols
WAP Gateway/ Proxy
Encoders and Decoders
Protocol Conversion
HTTP Request (URL)
HTTP Response (WML)
The Internet
Internet Protocols
Origin Server
WML WMLScript
Internet Protocol Stack

WAP Protocol Stack

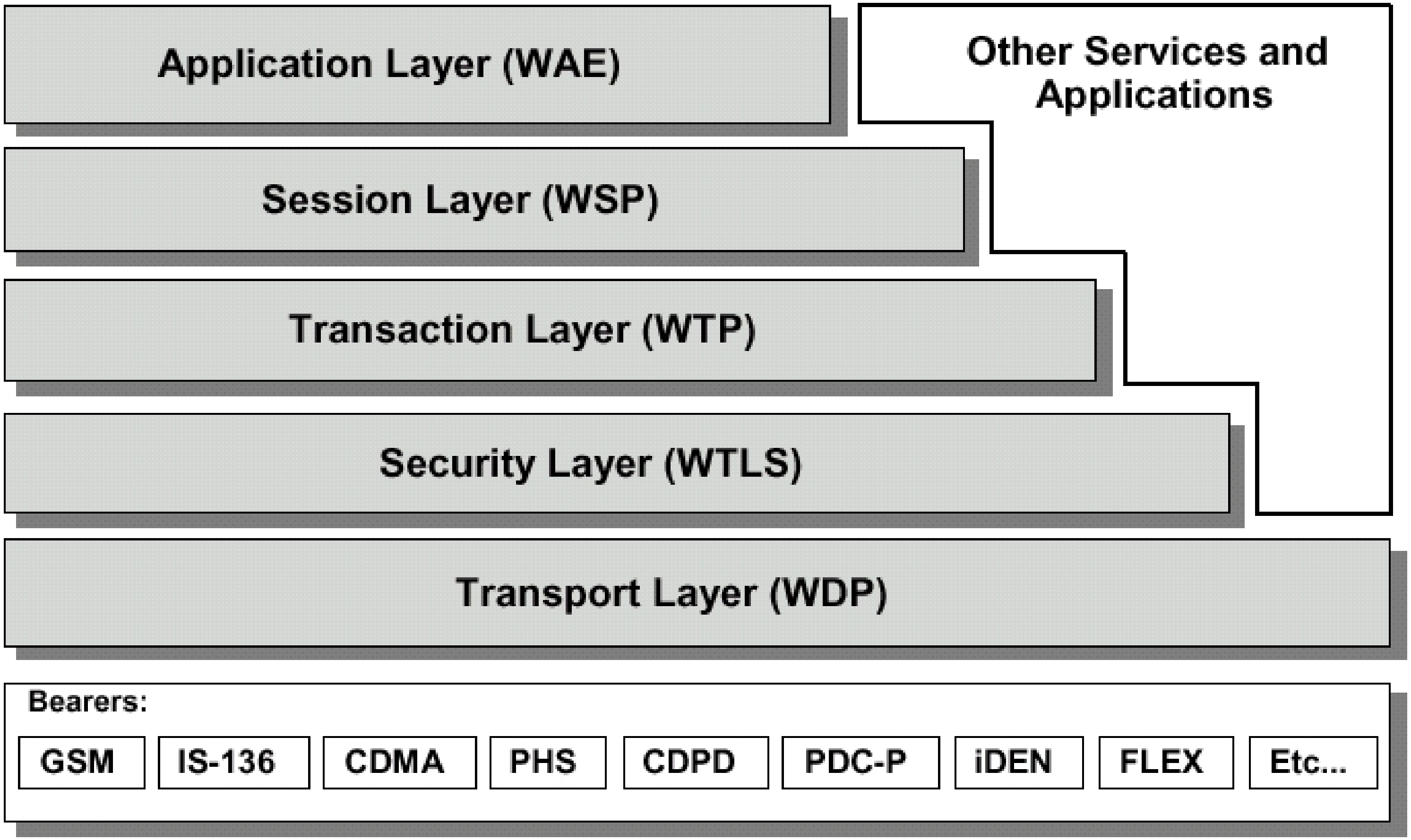


Figure 4. WAP Architecture

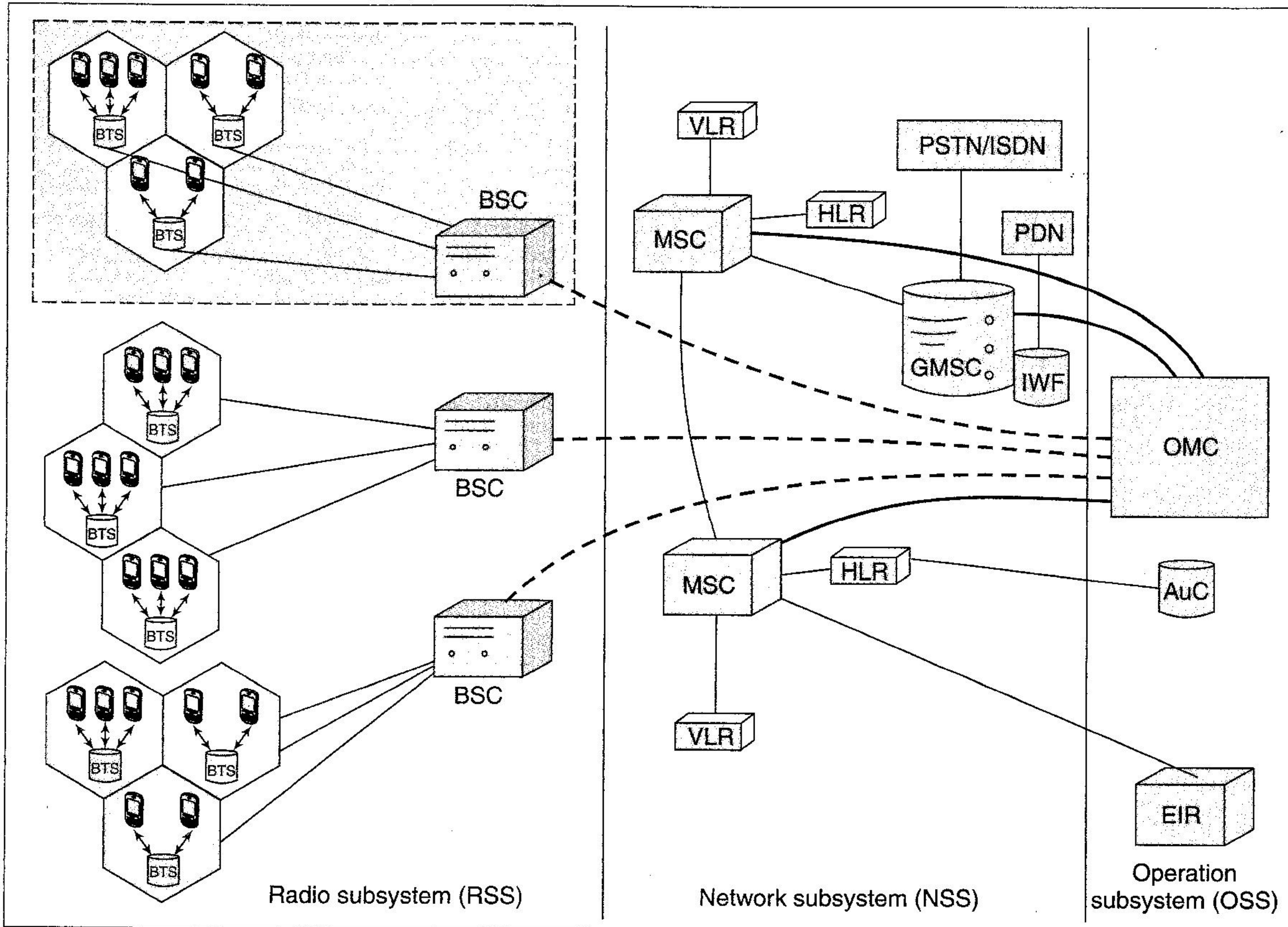
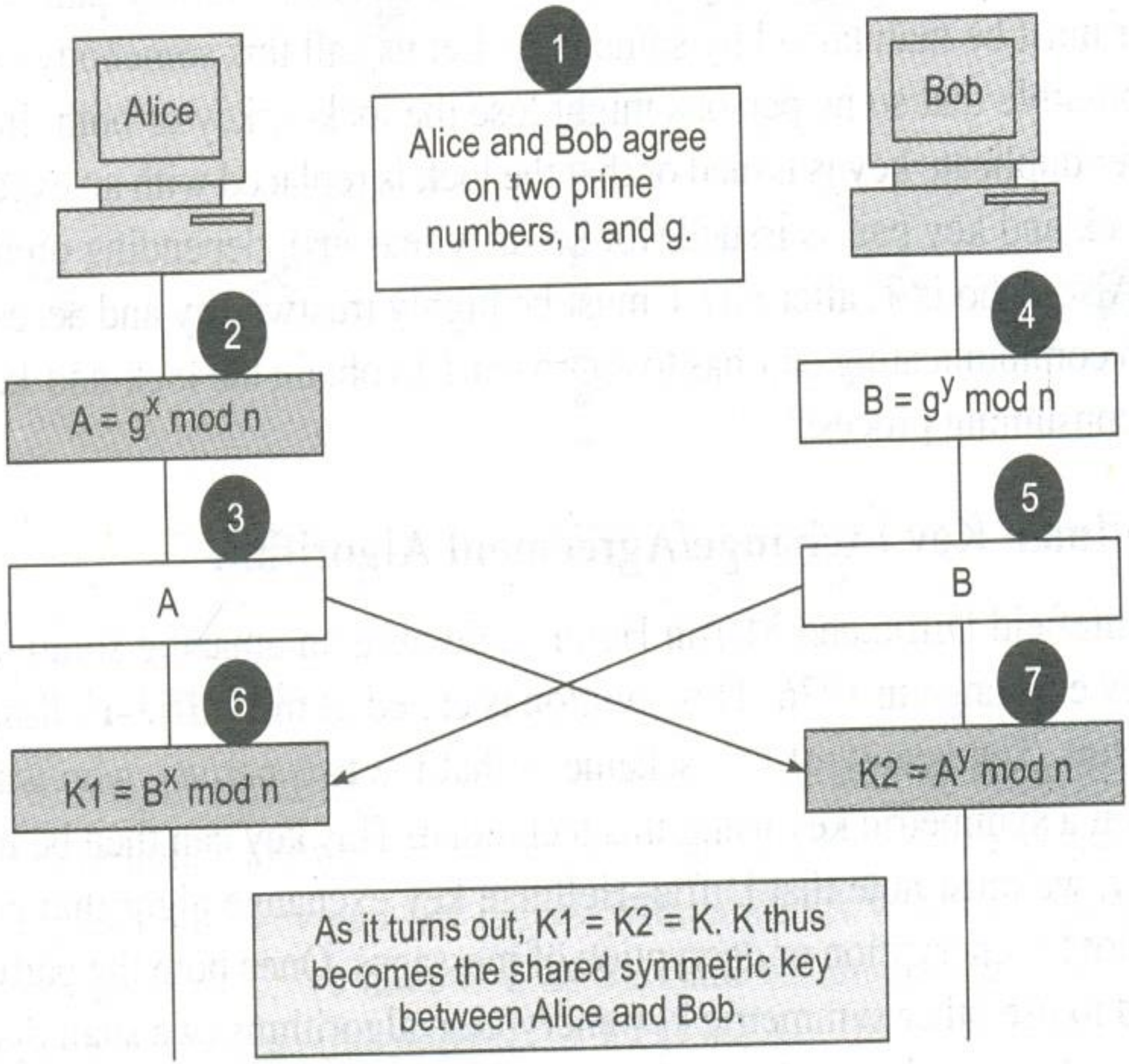


Fig. 3.2 GSM network architecture

Diffie-Hellman Key Exchange/Agreement

Algorithm

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$
3. Alice sends the number A to Bob.
4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$
5. Bob sends the number B to Alice.
6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$
7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$



Alice
1
Alice and Bob agree on two prime numbers, n and g.
Bob
2
$A = g^x \bmod n$
4
$B = g^y \bmod n$
3
A
5
B
6
$K1 = B^x \bmod n$
7
$K2 = A^y \bmod n$
As it turns out, K1 = K2 = K. K thus becomes the shared symmetric key between Alice and Bob.

EXAMPLE

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

   Let $n = 11$, $g = 7$.

2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$

   Let $x = 3$. Then, we have, $A = 7^3 \bmod 11 = 343 \bmod 11 = 2$.

3. Alice sends the number A to Bob.

   Alice sends 2 to Bob.

4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$

   Let $y = 6$. Then, we have, $B = 7^6 \bmod 11 = 117649 \bmod 11 = 4$.

5. Bob sends the number B to Alice.

   Bob sends 4 to Alice.

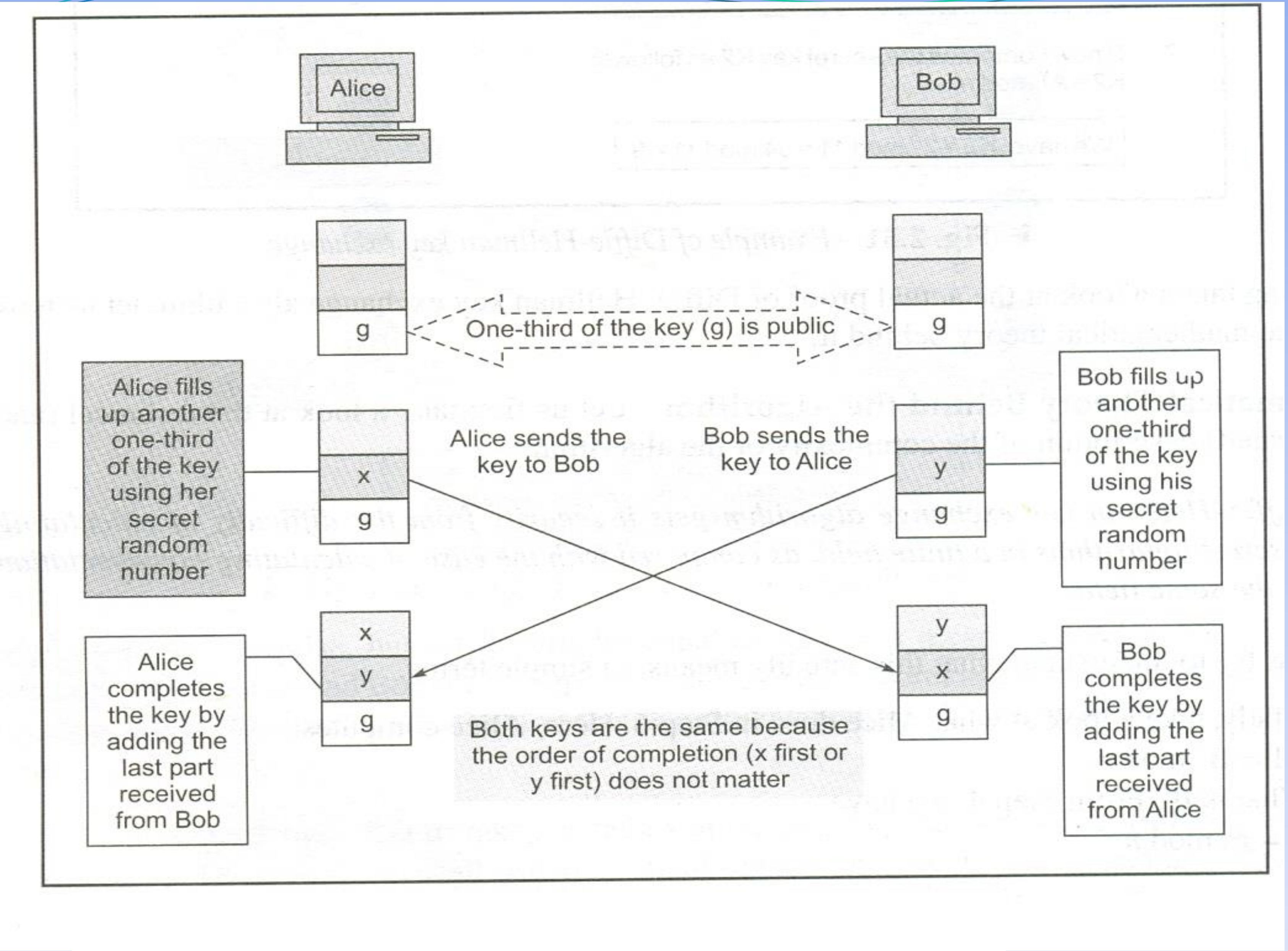
6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$

   We have, $K1 = 4^3 \bmod 11 = 64 \bmod 11 = 9$.

7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

   We have, $K2 = 2^6 \bmod 11 = 64 \bmod 11 = 9$.

EXPLANATION



MAN IN THE MIDDLE

1. Alice wants to communicate with Bob securely and therefore, she first wants to do a Diffie-Hellman key exchange with him. For this purpose, she sends the values of nand g to Bob, as usual. Let $n = 11$ and $g = 7$. (As usual, these values will form the basis of Alice's A and Bob's B, which will be used to calculate the symmetric key Kl = K2 = K.)
2. Alice does not realize that the attacker Tom is listening quietly to the conversation between her and Bob. Tom simply picks up the values of n and g and also forwards them to Bob as they originally were (i.e. $n = 11$ and $g = 7$).

Alice	Tom	Bob
$n = 11, g = 7$	$n = 11, g = 7$	$n = 11, g = 7$

Man-in-the-middle attack - Part I

3. Now, let us assume that Alice, Tom and Bob select random numbers x and y as shown in Fig. 2.54.

Alice	Tom	Bob
$x = 3$	$x = 8, y = 6$	$y = 9$

Man-in-the-middle attack - Part II

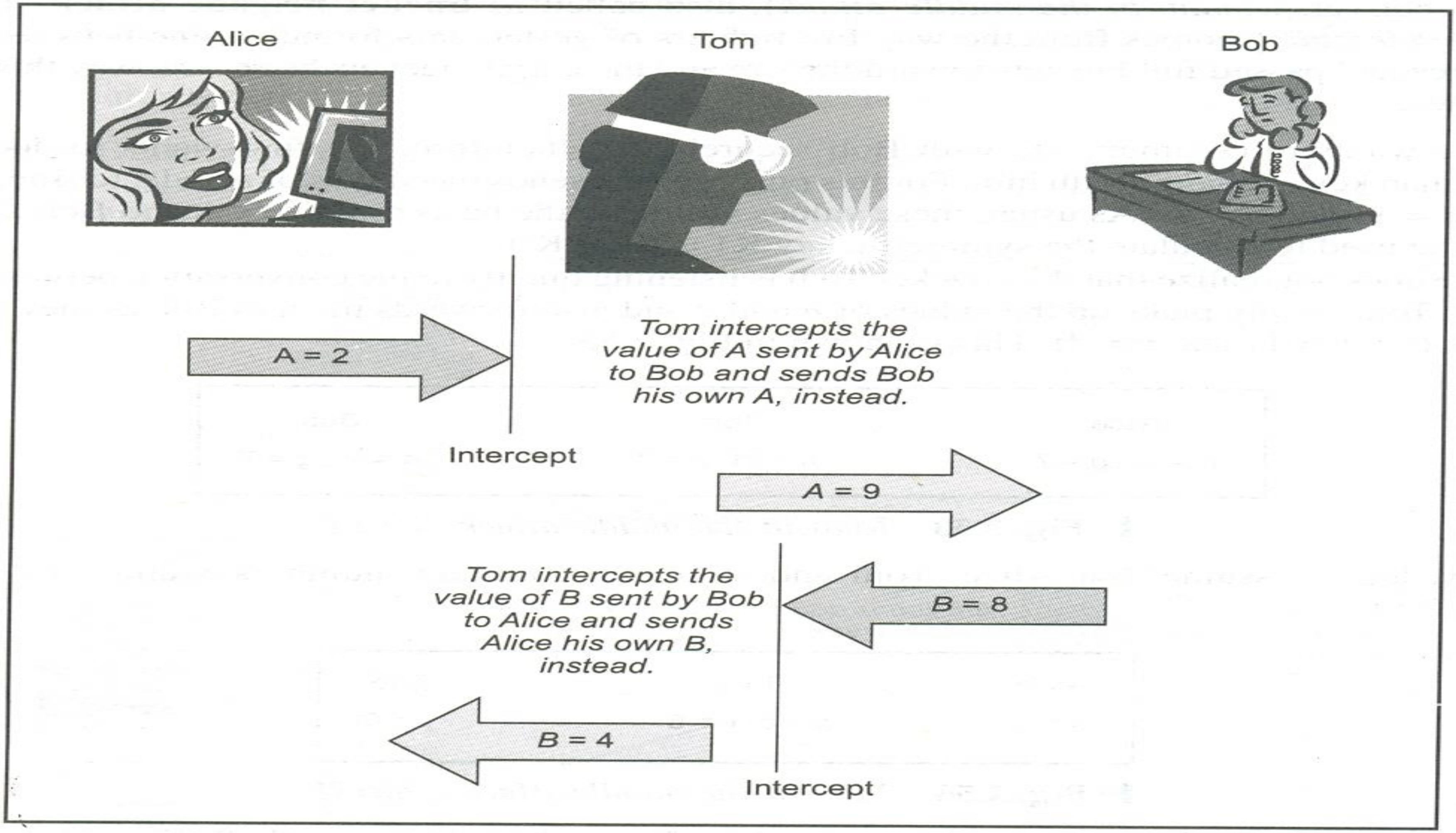
4. One question at this stage could be: why does Tom selects both x and y? We shall answer that shortly. Now, based on these values, all the three persons calculate the values of A and B as shown in Fig. 2.55. Note that Alice and Bob calculate only A and B, respectively. However, Tom calculates both A and B. We shall revisit this shortly.

MAN IN THE MIDDLE

Alice	Tom	Bob
$A = g^x \bmod n$	$A = g^x \bmod n$	$B = g^y \bmod n$
$= 7^3 \bmod 11$	$= 7^8 \bmod 11$	$= 7^9 \bmod 11$
$= 343 \bmod 11$	$= 5764801 \bmod 11$	$= 40353607 \bmod 11$
$= 2$	$= 9$	$= 8$
	$B = g^y \bmod n$	
	$= 7^6 \bmod 11$	
	$= 117649 \bmod 11$	
	$= 4$	

Man-in-the-middle attack - Part III

MAN IN THE MIDDLE



Man-in-the-middle attack - Part IV

MAN IN THE MIDDLE

Alice	Tom	Bob
$A = 2, B = 4^*$	$A = 2, B = 8$	$A = 9^*, B = 8$

(Note: * indicates that these are the values after Tom hijacked and changed them.)

Man-in-the-middle attack - Part V

MAN IN THE MIDDLE

Alice

$K1 = B^x \bmod n$
$= 4^3 \bmod 11$
$= 64 \bmod 11$
$= 9$

Tom

$K1 = B^x \bmod n$
$= 8^8 \bmod 11$
$= 16777216 \bmod 11$
$= 5$

$K2 = A^y \bmod n$
$= 2^6 \bmod 11$
$= 64 \bmod 11$
$= 9$

Bob

$K2 = A^y \bmod n$
$= 9^9 \bmod 11$
$= 387420489 \bmod 11$
$= 5$

Man-in-the-middle attack - Part VI

Diffie Hellman Key Exchange Problem

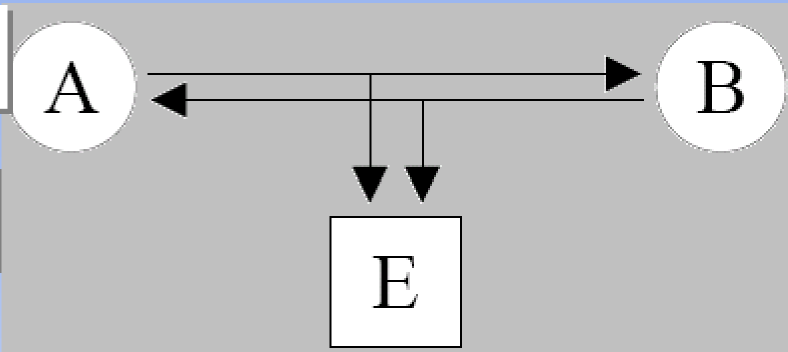
A & B both agree on two large prime no. n=11, g=7

Let secret random no. x=3

A calculates A= g^x mod n = 7^3 mod 11=2

A= 2, B(E version)= 4

K1= B^x mod n = 4^3 mod 11=9



Attacker picks up n & g and forward them to B

E selects two secret random no. x=8, y=6

E calculates
A= g^x mod n = 7^8 mod 11=9
B= g^y mod n = 7^6 mod 11=4

A= 2, B=8

A & B both agree on two large prime no. n=11, g=7

Let secret random no. y=9

B calculates B= g^y mod n= 7^9 mod 11=8

A(E version)= 9, B=8

K2= A^y mod n =9^9 mod 11=5

K1= B^x mod n = 8^8 mod 11=5 (Same as B)

K2= A^y mod n=2^6 mod 11= 9 (Same as A)

- ***proposed Tree Parity Technique using ANN***
- ***Example of Encryption***

History of ANN Cryptography

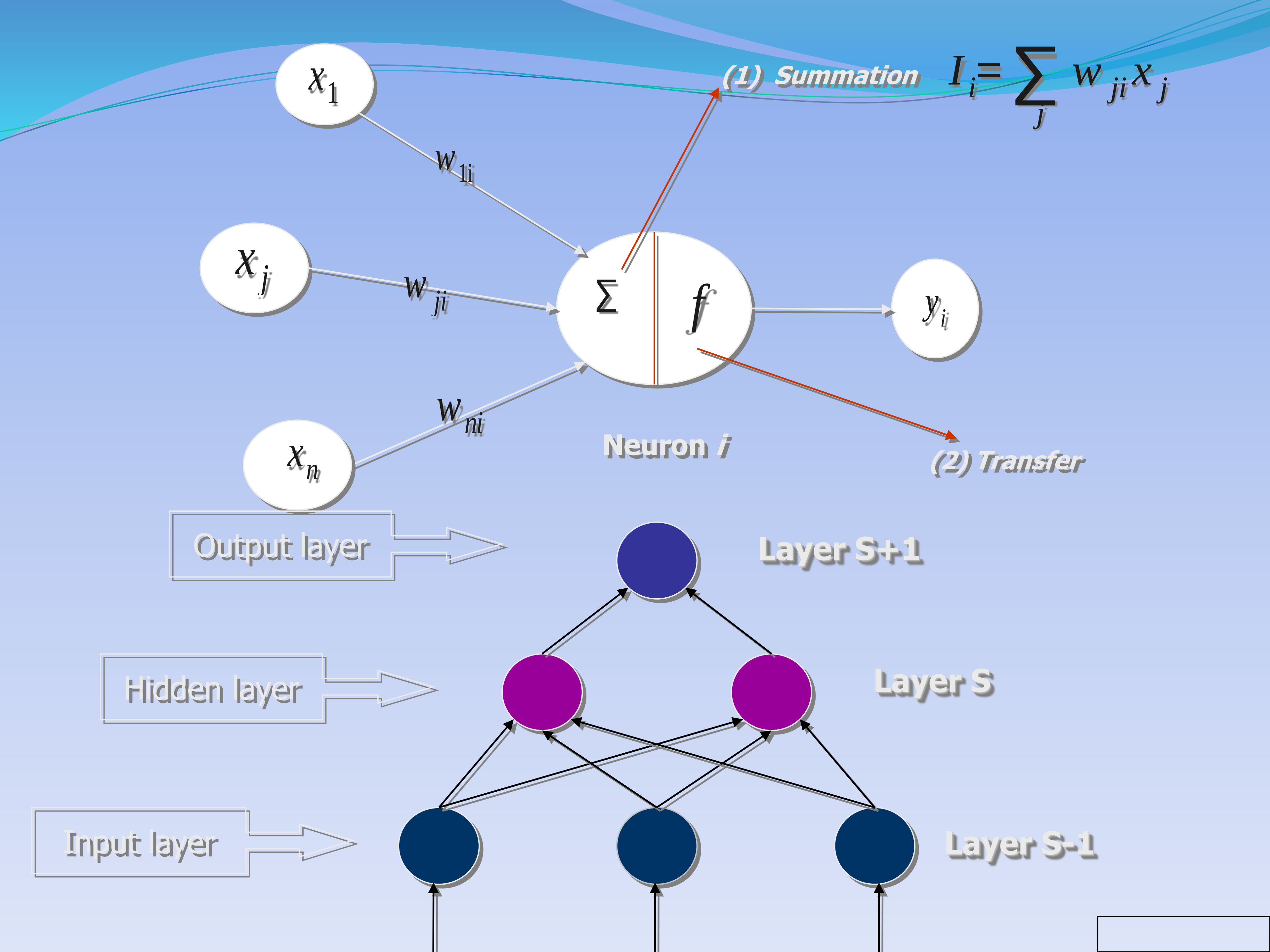
ANN application in cryptology can be categorized in two sub-fields, that is cryptanalysis and key-exchange. Neural cryptanalysis work was conducted by Ramzan [3].

The work on neural key exchange is rather a new research area. The work in this area is performed by a research group from Institute for Theoretical Physics in Wurzburg, Germany and Minerva Center, Bar-Ilan University in Ramat-Gan, Israel [4].

Kanter, I., Kinzel, W. and Kanter in the year 2002 proposed a neural key-exchange protocol that does not employ number theory but is based on a synchronization of neural networks by mutual learning [4].

In the same year Kinzel, W. and Kanter also proposed that the architecture used is a two-layered perceptron, exemplified by a parity machine with *K* hidden units. The secret information of each entity is the initial values for the weights, which are secret. Each network is then trained with the output of its partner. The work was extended to multilayer networks, parity machines [5].

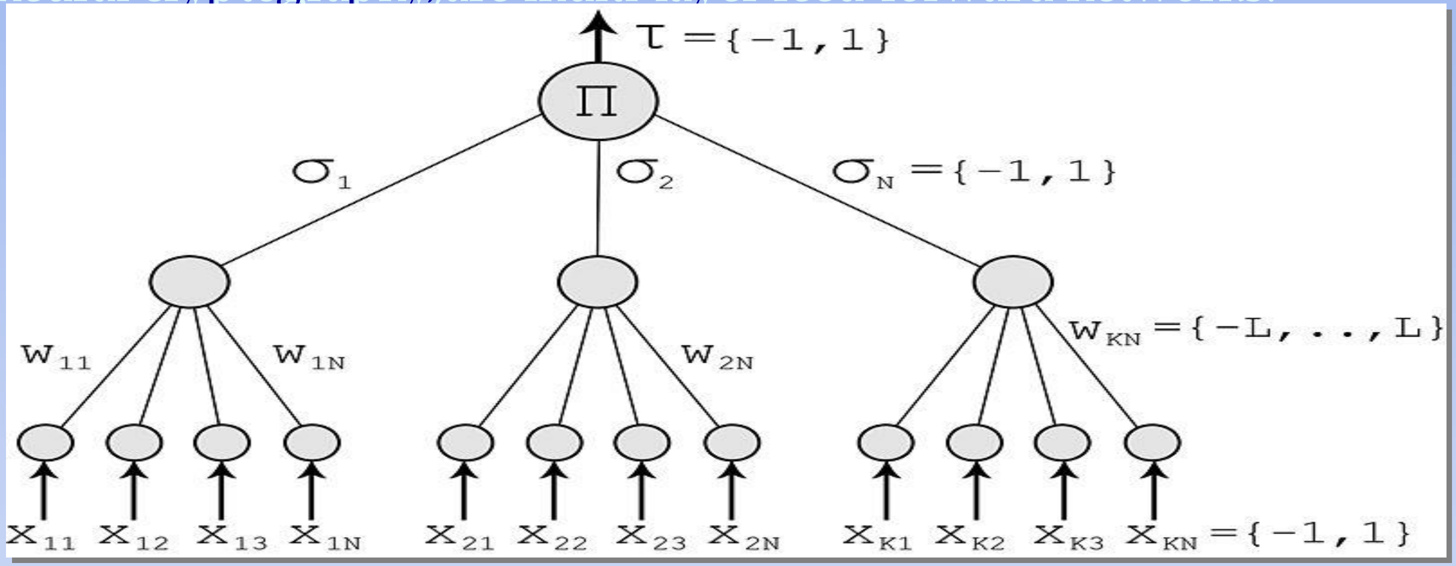
The ANN based Key Generatinion/Exchange Technique


(1) Summation
$I_i = \sum_j w_{ji} x_j$
x_1
x_j
x_n
w_{1i}
w_{ji}
w_{ni}
Σ
f
y_i
Neuron *i*
(2) Transfer
Output layer
Hidden layer
Input layer
Layer S+1
Layer S
Layer S-1

Key Generation using Neural Network

Tree Parity Machines

Tree Parity Machines, which are used by partners and attackers in neural cryptography,are multi-layer feed-forward networks.



K - the number of hidden neurons,

N - the number of input neurons connected to each hidden neuron, total (K*N) input neurons.

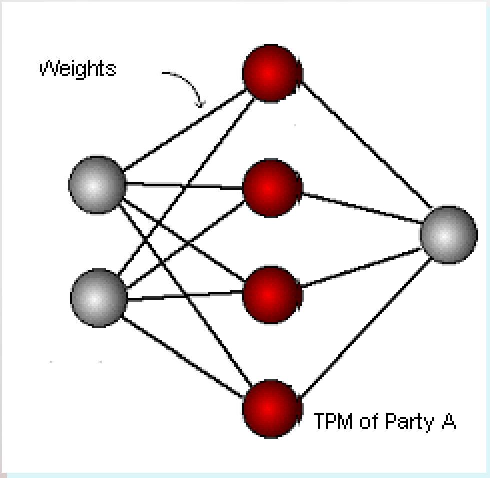
L - the maximum value for weight {-L..+L}

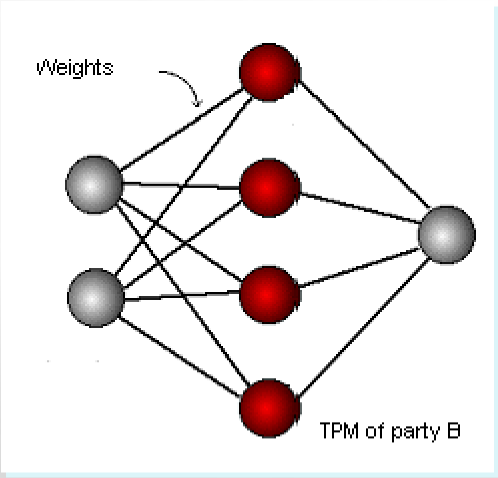
Here K = 3 and N = 4.

Neural Synchronization Scheme

Each party (A and B) uses its own (Same) tree parity machine. Synchronization of the tree parity machines is achieved in these steps

1. Initialize random weight values

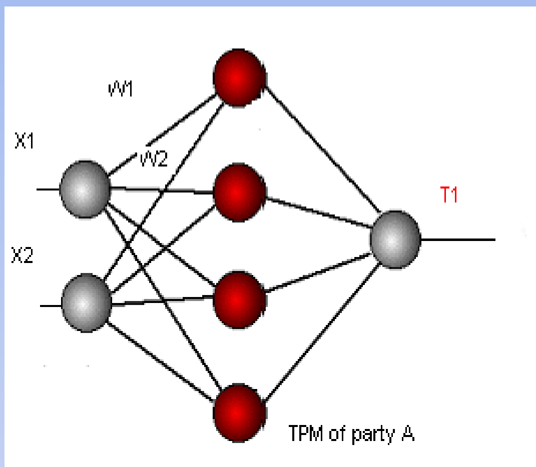




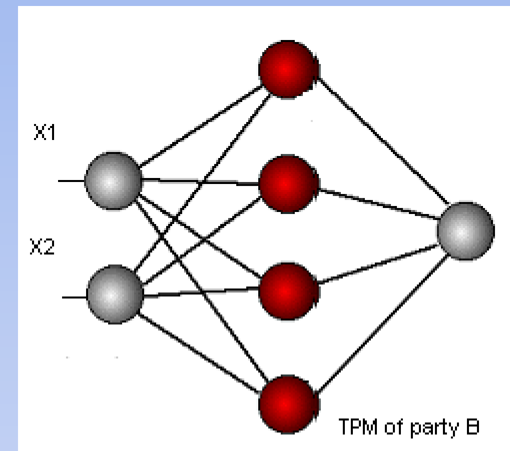
Neural Synchronization Scheme

<u>Execute these steps until the full synchronization is achieved</u>

1. Generate random input vector X



$$x_{ij} \in \{-1, +1\}$$



2. Compute the values of the hidden neurons

$$\sigma_i = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_{ij}\right)$$

Signum is a simple function, which returns -1,0 or 1:

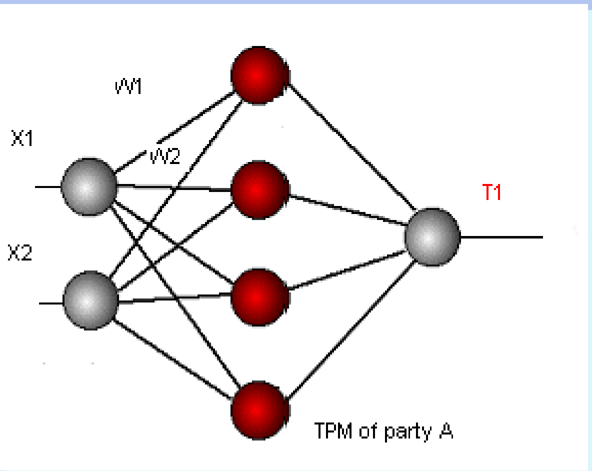
$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Neural Synchronization Scheme

3. Compute the value of the output neuron

$$\tau = \prod_{i=1}^{K} \text{SIGN}\left[\sum_{j=1}^{N} w_{i,j}\, x_{i,j}\right]$$

4. Compare the values of both tree parity machines



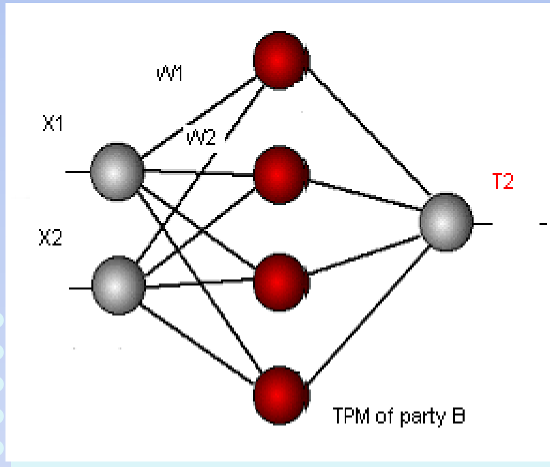
4.1 Outputs are others: go to 2.1

Output(A) ≠ Output(B)

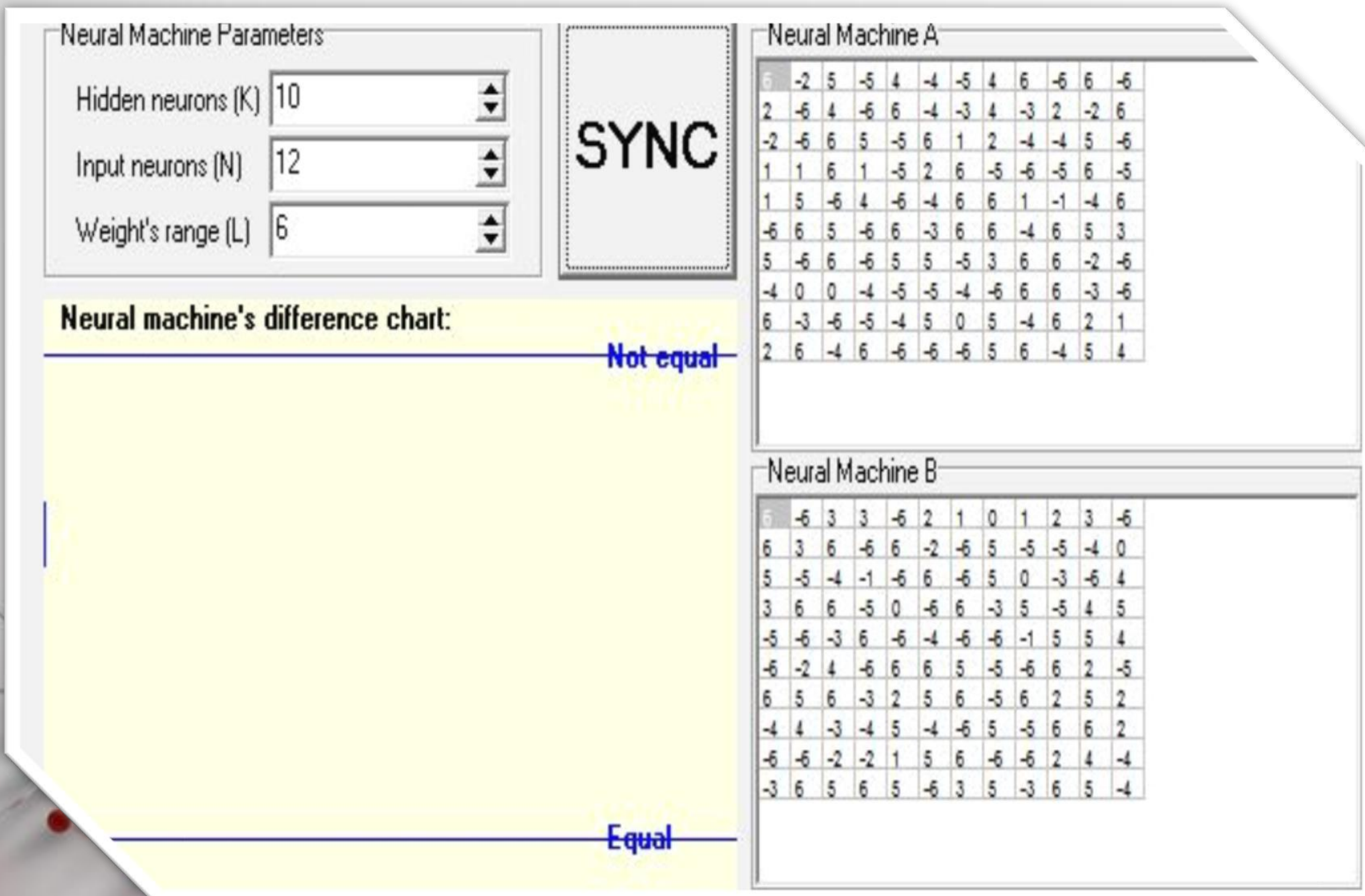
4.2 Outputs are same:

Output(A) = Output(B)

one of the suitable learning rules is applied to the weights



Weight Synchronization process



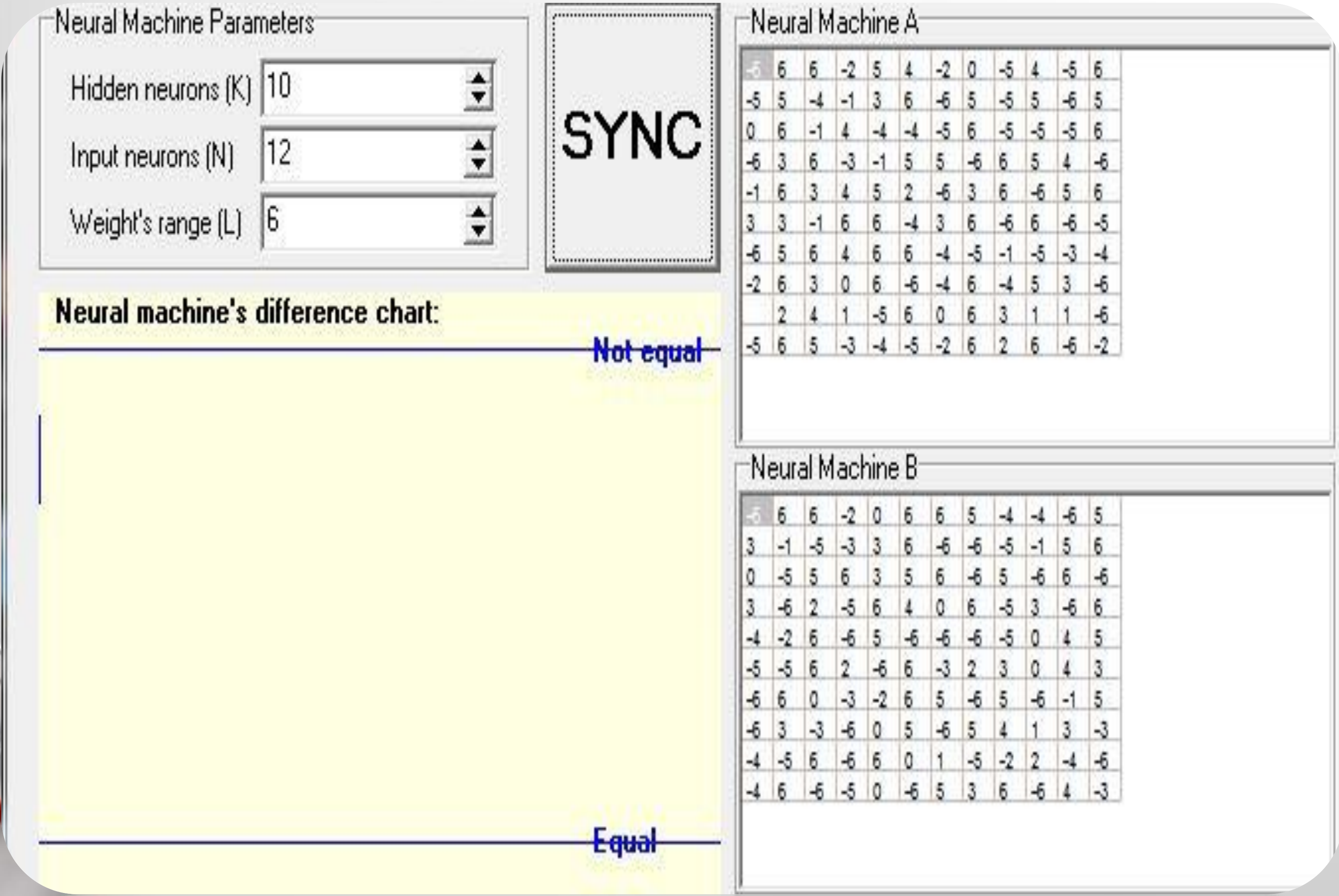
Neural Machine A

6	-2	5	-5	4	-4	-5	4	6	-6	6	-6
2	-6	4	-6	6	-4	-3	4	-3	2	-2	6
-2	-6	6	5	-5	6	1	2	-4	-4	5	-6
1	1	6	1	-5	2	6	-5	-6	-5	6	-5
1	5	-6	4	-6	-4	6	6	1	-1	-4	6
-6	6	5	-6	6	-3	6	6	-4	6	5	3
5	-6	6	-6	5	5	-5	3	6	6	-2	-6
-4	0	0	-4	-5	-5	-4	-6	6	6	-3	-6
6	-3	-6	-5	-4	5	0	5	-4	6	2	1
2	6	-4	6	-6	-6	-6	5	6	-4	5	4

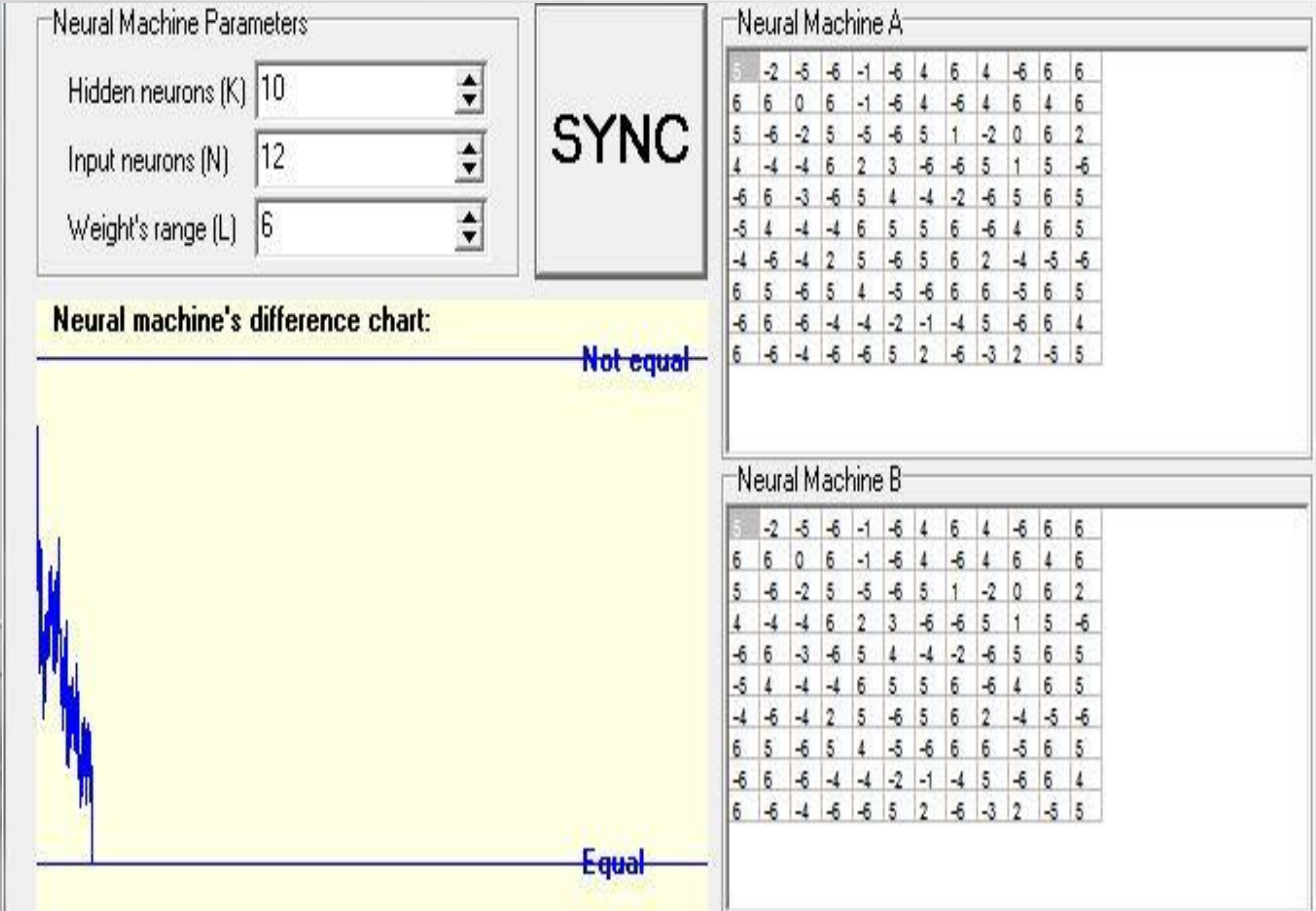
Neural Machine B

6	-6	3	3	-6	2	1	0	1	2	3	-6
6	3	6	-6	6	-2	-6	5	-5	-5	-4	0
5	-5	-4	-1	-6	6	-6	5	0	-3	-6	4
3	6	6	-5	0	-6	6	-3	5	-5	4	5
-5	-6	-3	6	-6	-4	-6	-6	-1	5	5	4
-6	-2	4	-6	6	6	5	-5	-6	6	2	-5
6	5	6	-3	2	5	6	-5	6	2	5	2
-4	4	-3	-4	5	-4	-6	5	-5	6	6	2
-6	-6	-2	-2	1	5	6	-6	-6	2	4	-4
-3	6	5	6	5	-6	3	5	-3	6	5	-4

Weight Synchronization process



Synchronized Weight Vectors



How do we update the weights?

We update the weights only if the final output values of the neural machines are equal.

One of the following learning rules can be used for the synchronization:

- Hebbian learning rule:

$$w_i^+ = w_i + \sigma_i x_i \Theta(\sigma_i \tau)\Theta(\tau^A \tau^B)$$

- Anti-Hebbian learning rule:

$$w_i^+ = w_i - \sigma_i x_i \Theta(\sigma_i \tau)\Theta(\tau^A \tau^B)$$

- Random walk:

$$w_i^+ = w_i + x_i \Theta(\sigma_i \tau)\Theta(\tau^A \tau^B)$$

Advantage of Neural Synchronisation

Each partener uses a seperate, but identical pseudo random no. generator . As these devices are initialized with a secret seed state shared by A& B. They produce exactly the same sequence of input bits.

Attacker does not know this secret seed state.

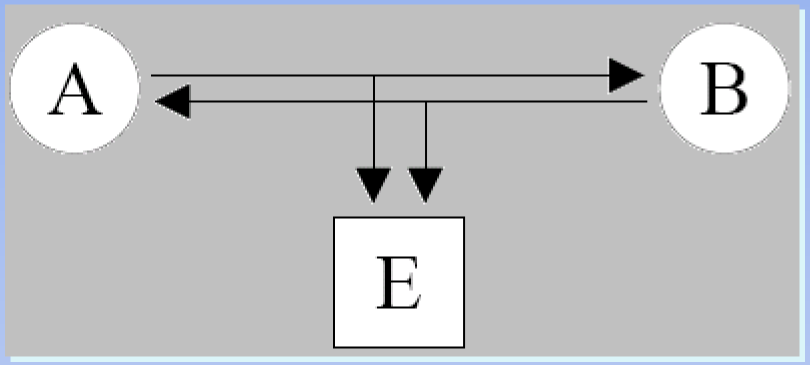
By increasing synaptic depth average synchronize time will be increased polynomial time. But success probability of attacker will be drop exponentially

Synchonization by mutual learning is much faster than learning by adopting to example generated by other network.

Unidirectional learning & bidirectional synchronization. As E can't influence A&B at the time they stop transmit due to synchrnization.

Only 1 weight get changed where, σ_i = T. So, difficult to find weight for attacker to know the actual weight without knowing internal representation it has to guess..

Learning with own tree parity machine

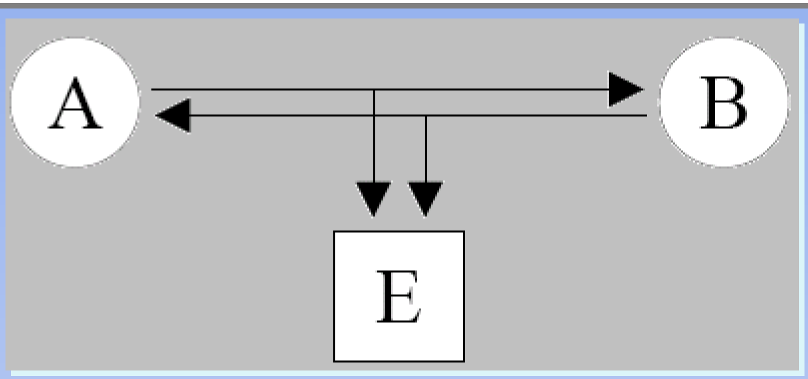


In each step there are 3 situations possible:

1. **Output(A) ≠ Output(B):** None of the parties updates its weights.
2. **Output(A) = Output(B) = Output(E):** All the three parties update weights in their tree parity machines.
3. **Output(A) = Output(B) ≠ Output(E):** Parties A and B update their tree parity machines, but the attacker can not do that. Because of this situation his learning is slower than the synchronization of parties A and B.

Cryptanalysis

Attacks and security of this protocol



Key exchange between two partners with a passive attacker listening to the communication.

In every attack it is considered, that the attacker E can eavesdrop messages between the parties A and B, but does not have an opportunity to change them.

Brute force

To provide a brute force attack, an attacker has to test all possible keys (all possible values of weights Wij). By K hidden neurons, K*N input neurons and boundary of weights L, this gives $(2L+1)^{KN}$ possibilities. For example, the configuration K = 3,L = 3 and N = 100 gives us $3*10^{253}$ key possibilities, making the attack difficult.

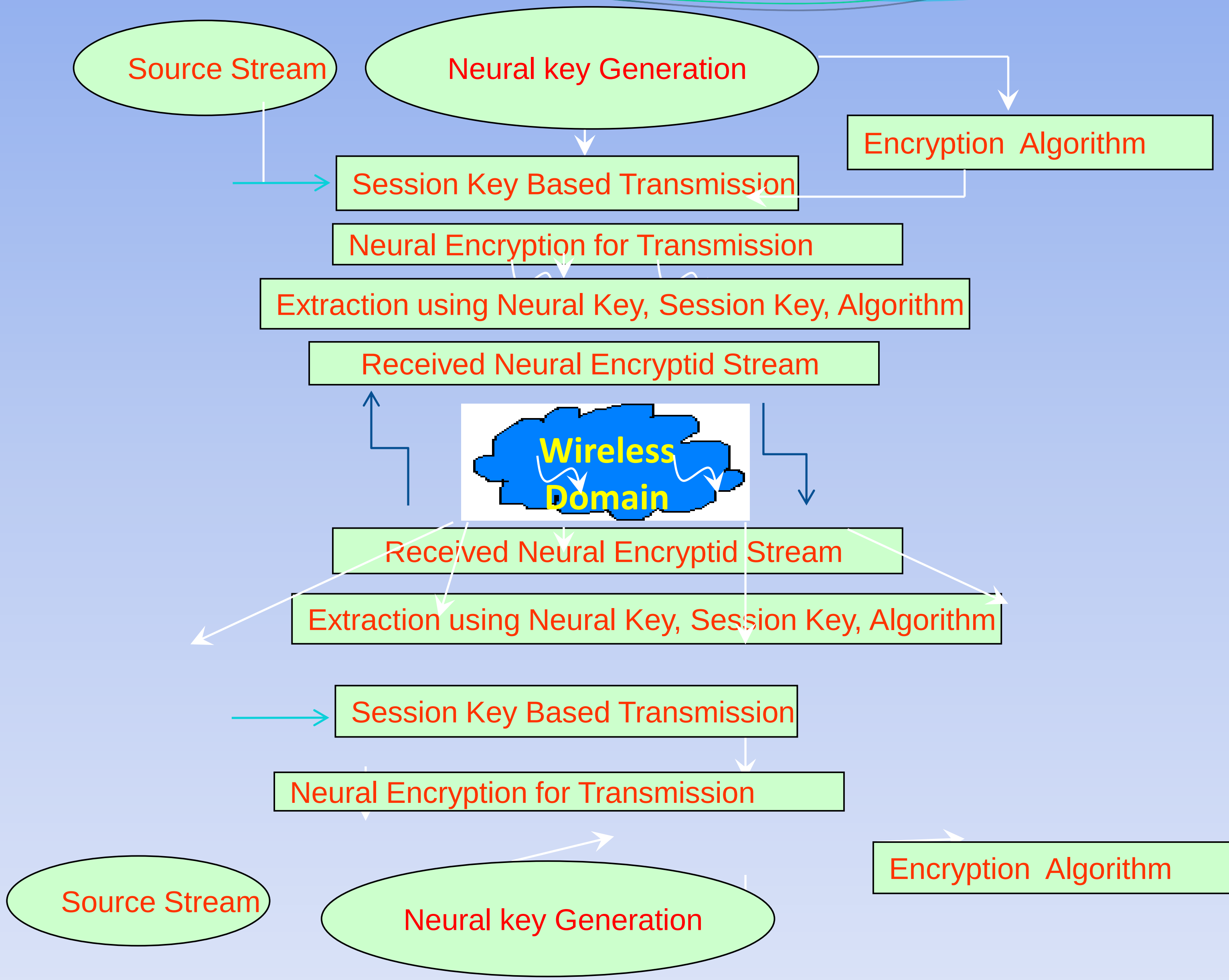
The synchronization of two parties is faster than learning of an attacker.

It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability.

Other attacks

For conventional cryptographic systems, we can improve the security of the protocol by increasing of the key length. In the case of neural cryptography, we improve it by increasing of the synaptic depth L of the neural networks. Changing this parameter increases the cost of a successful attack exponentially, while the effort for the users grows polynomially. Therefore, breaking the security of neural key exchange belongs to the complexity class NP.

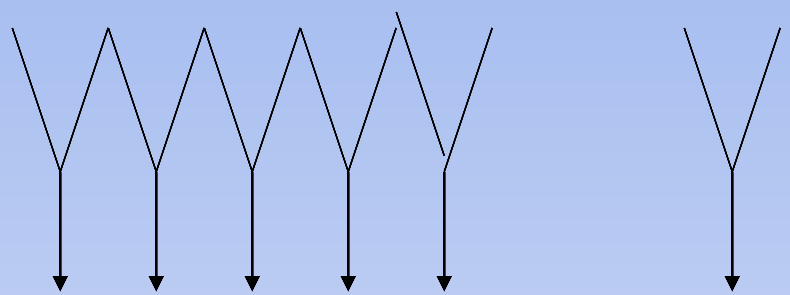
APPLICATION GENERATION



Key Exchange

TRIANGULARISATION(XNOR)

$$S^j = s^j_0 \quad s^j_1 \quad s^j_2 \quad s^j_3 \quad s^j_4 \quad s^j_5 \quad \ldots \quad s^j_{n-(j+2)} s^j_{n-(j+1)}$$



$$S^{j+1} = s^{j+1}_0 s^{j+2}_1 s^{j+3} s^{j+4}_3 s^{j+5}_4 \cdots \qquad s^j_{n-(j+2)}$$

Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$S^0{}_0\ S^1{}_0\ S^2{}_0\ S^3{}_3\ S^4{}_0 \ldots S^{n-2}{}_0\ S^{n-1}{}_0$	Taking all the MSBs starting from the source block till the last block generated
010	$S^{n-1}{}_0\ S^{n-2}{}_0\ S^{n-3}{}_0\ S^{n-4}{}_0\ S^{n-5}{}_0 \ldots S^1{}_0\ S^0{}_0$	Taking all the MSBs starting from the last block generated till the source block
011	$S^0{}_{n-1}\ S^1{}_{n-2}\ S^2{}_{n-3}\ S^3{}_{n-4}\ S^4{}_{n-5} \ldots S^{n-2}{}_1\ S^{n-1}{}_0$	Taking all the LSBs starting from the source block till the last block generated
100	$S^{n-1}{}_0\ S^{n-2}{}_1\ S^{n-3}{}_2\ S^{n-4}{}_3\ S^{n-5}{}_4 \ldots S^1{}_{n-2}\ S^0{}_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

Different Target Blocks generated using TE for S = 10010101

Source Block S	Target Block Corresponding to Serial No.	Target Block T
10010101	001	10010101
	010	10101001
	011	10111101
	100	10111101

ENCODING

1 0 0

0 1

0

DECODING

0 1 0

0 0

1

TRIANGULAR BASED ENCODING

	1	0	1	1	Neural Key
Θ					
	1	0	0	1	Source Stream
	1	1	0	1	Intermediate Stream

1	1	0	1	
1	0	0		
0	1			Triangular Encoding
0				

0	1	0	1	Encoded Stream

TRIANGULAR BASEDDECODING

0	1	0	1	Received Encoded Stream
0	0	0		
1	1			Triangular Decoding
1				

	1	1	0	1	Decoded Stream
Θ					
	1	0	1	1	Neural Key
	1	0	0	1	Decoded Source Stream

Proposal of Key Format

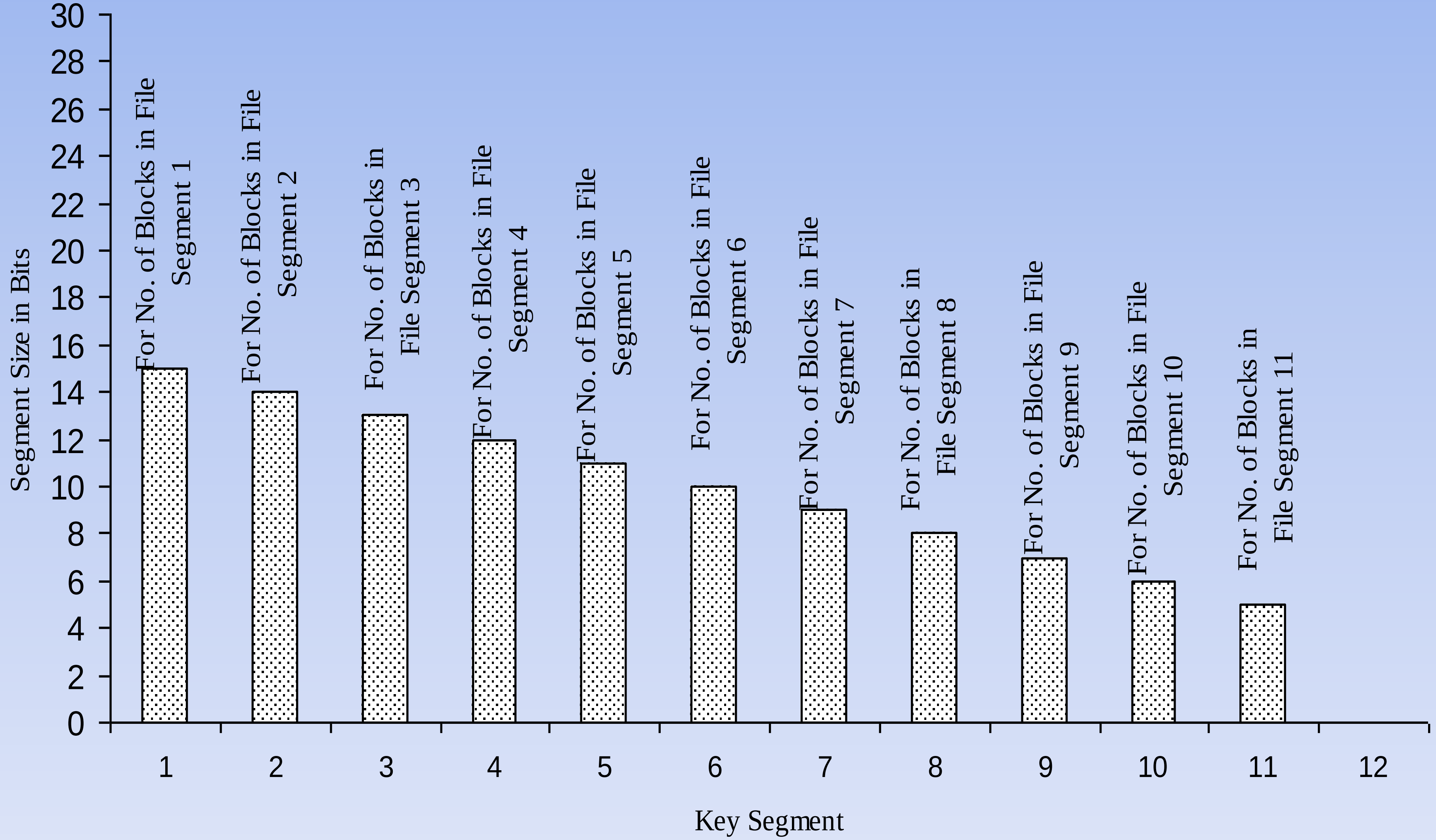
A 110-bit key format consisting of 11 different segments has been proposed For the segment of the rank R, there can exist a maximum of $N = 2^{15-R}$ blocks, each of the unique size of $S = 2^{15-R}$ bits, R starting from 1 and moving till 11.

For different values of R, following segments are generated:

- **Segment with R=1 formed with the first maximum 16384 blocks, each of size 16384 bits;**
- **Segment with R=2 formed with the first maximum 8192 blocks, each of size 8192 bits;**
- **Eegment with R=3 formed with the next maximum 4096 blocks, each of size 4096 bits;**
- **Eegment with R=4 formed with the next maximum 2048 blocks, each of size 2048 bits;**
- **Segment with R=5 formed with the next maximum 1024 blocks, each of size 1024 bits;**
- **Segment with R=6 formed with the next maximum 512 blocks, each of size 512 bits;**
- **Segment with R=7 formed with the next maximum 256 blocks, each of size 256 bits;**
- **Segment with R=8 formed with the next maximum 128 blocks, each of size 128 bits;**
- **Segment with R=9 formed with the next maximum 64 blocks, each of size 64 bits;**
- **Segment with R=10 formed with the next maximum 32 blocks, each of size 32 bits;**
- **Segment with R=11 formed with the next maximum 16 blocks, each of size 16 bits;**

With such a structure, the key space becomes of 110 bits long and a file of the maximum size of around 44.74 MB

110-bit key format with 11 segments for RPMS Technique



Example of Key Generation-110 bit key

0	0	0	0	0	0	0	0	0	1	0	0	1	0	1

37 blocks/37bits

0	0	0	0	0	0	0	1	0	0	0	0	0	1

65 blocks/65bits

0	0	0	0	0	0	1	0	0	0	1	1	1

71 blocks/71bits

0	0	0	0	0	0	0	1	0	1	1	0

22 blocks/22bits

0	0	0	0	0	0	0	1	1	1	1

15 blocks/15bits

0	0	0	1	0	0	0	0	0	0

64 blocks/64bits

0	0	0	0	0	0	0	0	0

0 blocks/0bits

0	0	0	0	0	0	0	0

0	0	0	0	0	0	0

1	1	0	0	0	1

49 blocks/49bits

0	1	0	0	0

8 blocks/8bits

Total 37+65+71+22+15+64+49+8 blocks = 331 blocks

The Size of the file for this Session Key

Total 37+65+71+22+15+64+49+8 blocks = 331 blocks

and

37*37 + 65*65 + 71*71 + 22*22+ 15*15+

64*64 + 49*49 + 8*8 = 17905 bits + 7 bits =17912 bits

=2239 bytes

Modulo 2 Encryption Technique

The ANN Encoding Technique

Source Stream

↓

Intermediate Modulo 2 Encoded Stream

↓

Neural Encoded txt

The ANNRPMS Technique

Source Stream at Source Node

↓

Modulo 2 Encoded Stream

↓

Neural Encoded Stream

↓

Send the encrypted stream through Wireless Nodes

↓

Neural Decoded Stream

↓

Modulo 2 Decoded Stream

↓

Source Stream at Dest. Node

Example of Encryption

1. Consider a plain text

"Local Area Network"

Character	Byte
L	01001100
o	01101111
c	01100011
a	01100001
l	01101100
<Blank>	00100000

Character	Byte
A	01000001
r	01110010
e	01100101
a	01100001
<Blank>	00100000

Character	Byte
N	01001110
e	01100101
t	01110100
w	01110111
o	01101111
r	01110010
k	01101011
<Blank>	00100000

1. Character-to-Byte Conversion for the Text "Local Area Network"

Example of Encryption

Putting together these bytes in the original sequence, we get the source stream of bits as the following:

S=01001100/01101111/01100011/01100001/01101100/00100000/01000001/01110010/01100101/01100001/00100000/01001110/01100101/01110100/01110111/01101111/01110010/01101011

2. Now, we decompose S into a set of 5 blocks, each of the first four being of size 32 bits and the last one being of 16 bits.

S_1=01001100011011110110001101100001

S_2=01101100001000000100000101110010

S_3=01100101011000010010000001001110

S_4=01100101011101000111011101101111

S_5=0111001001101011

Example of Encryption

3. For the block S_1, corresponding to which the decimal value is $(1282368353)_{10}$, the process of encryption is shown below:

1282368353 → **Corresponding Decimal Value**

641184177^{1} → **Position of 1282368353 in the Series of Natural Odd Numbers (1 for Odd)**

320592089^{1} → **Position of 641184177 in the Series of Natural Odd Numbers (1 for Odd)**

160296045^{1} → **Position of 320592089 in the Series of Natural Odd Numbers (1 for Odd)**

80148023^{1} → **Position of 80148023 in the Series of Natural Odd Numbers (1 for Odd)**

40074012^{1} → **Position of 80148023 in the Series of Natural Odd Numbers (1 for Odd)**

20037006^{0} → **Position of 40074012 in the Series of Natural Even Numbers (0 for Even)**

10018503^{0} → **Position of 20037006 in the Series of Natural Even Numbers (0 for Even)**

Example of Encryption

5009252^1 → Position of 10018503 in the Series of Natural Odd Numbers (1 for Odd)

2504626^0 → Position of 5009252 in the Series of Natural Even Numbers (0 for Even)

1252313^0 → Position of 2504626 in the Series of Natural Even Numbers (0 for Even)

626157^1 → Position of 1252313 in the Series of Natural Odd Numbers (1 for Odd)

313079^1 → Position of 626157 in the Series of Natural Odd Numbers (1 for Odd)

156540^1 → Position of 313079 in the Series of Natural Odd Numbers (1 for Odd)

78720^0 → Position of 156540 in the Series of Natural Even Numbers (0 for Even)

39135^0 → Position of 78720 in the Series of Natural Even Numbers (0 for Even)

19568^1 → Position of 39135 in the Series of Natural Odd Numbers (1 for Odd)

9784^0 → Position of 19568 in the Series of Natural Even Numbers (0 for Even)

4892^0 → Position of 9784 in the Series of Natural Even Numbers (0 for Even)

2446^0 → Position of 4892 in the Series of Natural Even Numbers (0 for Even)

Example of Encryption

1223^0 → Position of 2446 in the Series of Natural Even Numbers (0 for Even)

612^1 → Position of 1223 in the Series of Natural Odd Numbers (1 for Odd)

306^0 → Position of 612 in the Series of Natural Even Numbers (0 for Even)

153^0 → Position of 306 in the Series of Natural Even Numbers (0 for Even)

77^1 → Position of 153 in the Series of Natural Odd Numbers (1 for Odd)

39^1 →Position of 77 in the Series of Natural Odd Numbers (1 for Odd)

20^1 →Position of 39 in the Series of Natural Odd Numbers (1 for Odd)

10^0 → Position of 20 in the Series of Natural Even Numbers (0 for Even)

5^0 → Position of 10 in the Series of Natural Even Numbers (0 for Even)

3^1 → Position of 5 in the Series of Natural Odd Numbers (1 for Odd)

2^1 → Position of 3 in the Series of Natural Odd Numbers (1 for Odd)

1^0 → Position of 2 in the Series of Natural Even Numbers (0 for Even)

1^1 → Position of 1 in the Series of Natural Odd Numbers (1 for Odd)

Example of Encryption

4. From this we generate the target block T_1 corresponding to S_1 as:

T_1=11111001001110010000100111001101

Applying the similar process, we generate target blocks T_2, T_3, T_4 and T_5 as follows corresponding to source blocks S_2, S_3, S_4 and S_5 respectively.

T_2=01110001011111011111101111001001

T_3=01001101111110110111100101011001

T_4=10001001000100011101000101011001

T_5=1110100110110001

Key Exchange

The synchronized weight vector from the previous phase in the form of blocks of bits with different size like 8/ 16/32/ 64/ 128/ 256. The rules to be followed for generating a cycle are as follows:

		1st half Weight Vector Block (MSB)			(LSB)	2nd half Weight Vector Block (MSB)			(LSB)
Sender's steps	S=	0	1	0	1	0	0	1	1
	K=	*1*			*0*	*1*			*0*
	I1=	1	1	0	1	1	0	1	1
	K=		*0*	*1*			*0*	*1*	
Sender's / Receiver's steps	I2=	1	1	1	1	1	0	0	1
Receiver's steps	*K=*	*1*			*0*	*1*			*0*
	I3=	0	1	1	1	0	0	0	1
	K=		*0*	*1*			*0*	*1*	
	I4=	0	1	0	1	0	0	1	1

Final Step of Encryption

For different size of weight sub vector different intermediate blocks may be considered as the corresponding encrypted blocks. For example, the policy may be something like that out of three weight sub vector blocks B_1, B_2, B_3 in a key block of bits, the 4^{th}, the 7^{th} and the 5^{th} intermediate blocks respectively are being considered as the final key blocks. In such a case, the key of the scheme will become much more complex, which in turn will ensure better security.

Final Neural Key Block=Intermediate Weight Vector Block of cycle

+

Position information of Intermediate Weight Vector Block of cycle

Now perform cascading xoring of Modulo2 encrypted block with the Neural Secret Key, final encrypted cipher text is generated. This stream of bits, in the form of a stream of characters, is transmitted as the encrypted message.

Results

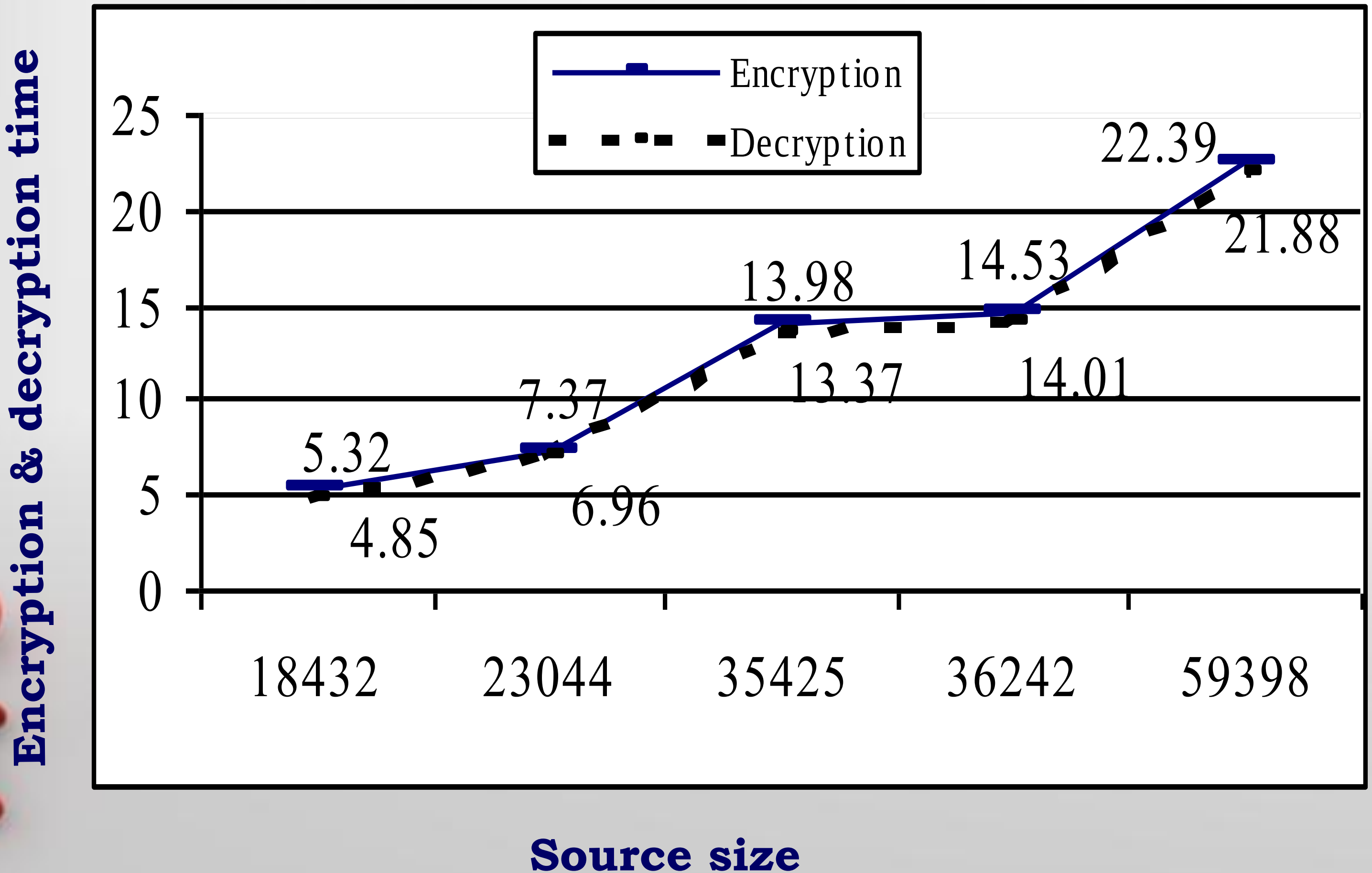
The results have been presented on the basis of the following factors:

- Computation of the encryption time, the decryption time, and the Pearsonian Chi Square value between the source and the encrypted files
- Performing the frequency distribution test
- Comparison with the RSA technique

Encryption/decryption time Vs. File size

Encryption Time (s)			Decryption Time (s)		
Source Size (bytes)	**Proposed ANNRPMS**	**RPSP**	**Encrypted Size (bytes)**	**Proposed ANNRPMS**	**RPSP**
18432	**5. 32**	7.85	18432	**4.85**	7.81
23044	**7. 37**	10.32	23040	**6.96**	9.92
35425	**13. 98**	15.21	35425	**13. 37**	14.93
36242	**14. 53**	15.34	36242	**14. 01**	15.24
59398	**22. 39**	25.49	59398	**21. 88**	24.95

Source size Vs. encryption time & decryption time

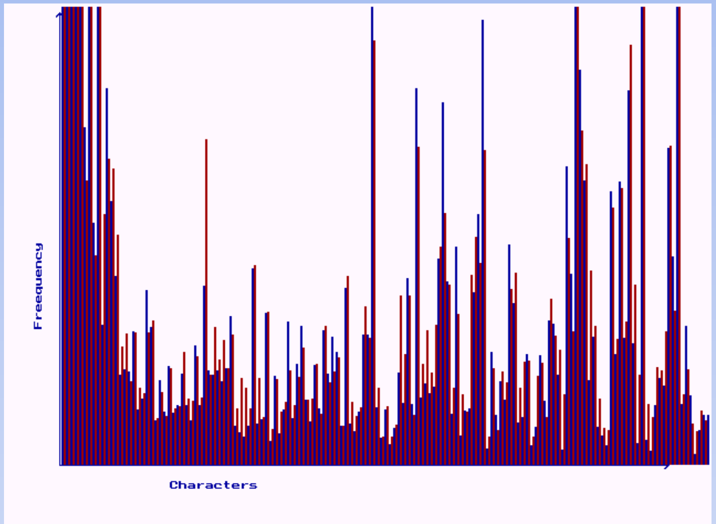


Source size Vs.Chi-square value

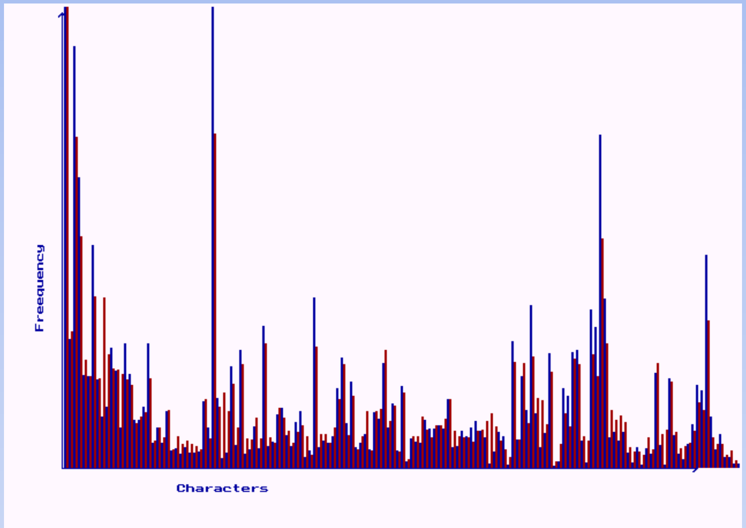
Stream Size (bytes)	Chi-Square value (TDES)	Chi-Square value (Proposed ANNRPMS)	Chi-Square value (RBCM CPCC)	Chi-Square value (RSA)
1500	1228.5803	2465.0645	2464.0324	5623.14
2500	2948.2285	5643.4673	5642.5835	22638.99
3000	3679.0432	6757.1533	6714.6741	12800.355
3250	4228.2119	6996.6177	6994.6189	15097.77
3500	4242.9165	10572.6982	10570.4671	15284.728

Results for Frequency Distribution Test

Frequency Distribution Chart for Source file and Encrypted file



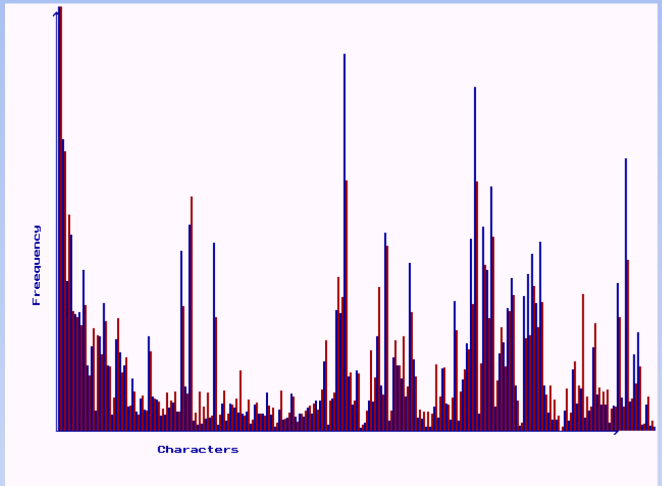
Segment of Frequency Distribution Chart for ANNRBLC.EXE and Encrypted A1.EXE



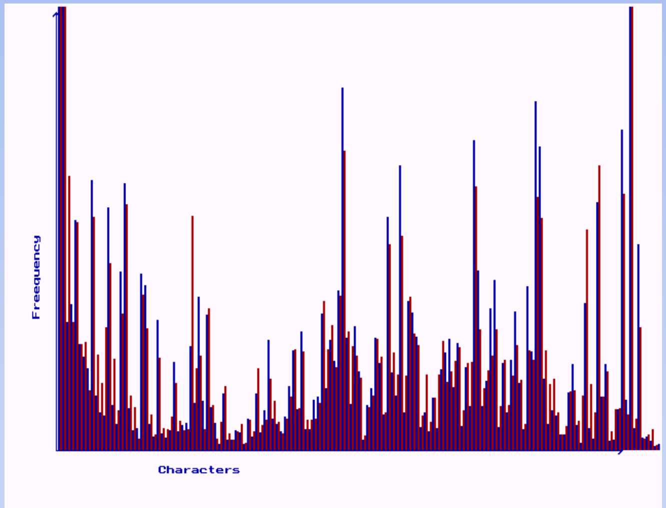
Segment of Frequency Distribution Chart for DOSKEY.COM and Encrypted A3.COM

Blue lines indicate the occurrences of characters in the source file and red lines indicate the same in the corresponding encrypted file

Frequency Distribution Chart for Source file and Encrypted file



Segment of Frequency Distribution Chart for NDDEAPI.DLL and Encrypted A2.DLL



Segment of Frequency Distribution Chart for USBD.SYS and Encrypted A2.SYS

Blue lines indicate the occurrences of characters in the source file and red lines indicate the same in the corresponding encrypted file

Cryptanalysis

The synchronization of two parties is faster than learning of an attacker.

It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability.

Other attacks

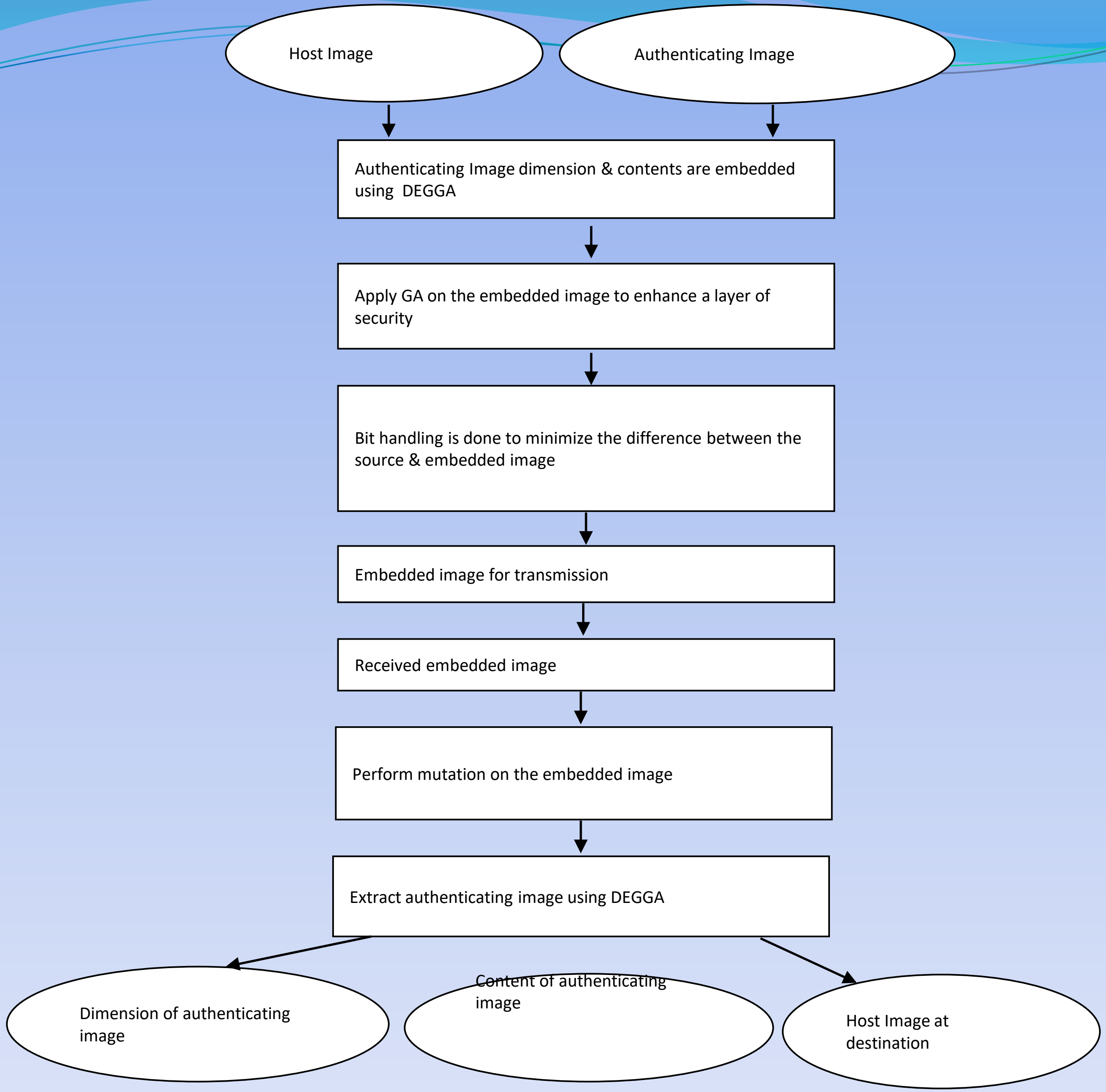
For conventional cryptographic systems, we can improve the security of the protocol by increasing of the key length. In the case of neural cryptography, we improve it by increasing of the synaptic depth L of the neural networks. Changing this parameter increases the cost of a successful attack exponentially, while the effort for the users grows polynomially. Therefore, breaking the security of neural key exchange belongs to the complexity class NP.

Conclusions

- So ANNRPMS technique enhances the security features of the algorithm by increasing of the synaptic depth L of the neural networks.
- In this case, the two partners A and B do not have to exchange a common secret key over a public channel but use their indistinguishable weights as a secret key needed for encryption or decryption. So likelihood of attack of ANNRPMS much lesser.
- The time overhead may increase marginally due to incorporation of neural network based computation and session key.
- But it is shown that all of these initial states move towards the same final weight vector, which may sometimes lead to minimize the strength of the secret key.
- The proposed technique may be used in online mobile communication system through which adaptive transmission may be possible.

GA Based Steganography

A_{00}, A_{01}	A_{02}, A_{03}	A_{04}, A_{05}
A_{10}, A_{11}	A_{12}, A_{13}	A_{14}, A_{15}
A_{20}, A_{21}	A_{22}, A_{23}	A_{24}, A_{25}
A_{30}, A_{31}	A_{32}, A_{33}	A_{34}, A_{35}


Host Image
Authenticating Image
Authenticating Image dimension & contents are embedded using DEGGA
Apply GA on the embedded image to enhance a layer of security
Bit handling is done to minimize the difference between the source & embedded image
Embedded image for transmission
Received embedded image
Perform mutation on the embedded image
Extract authenticating image using DEGGA
Dimension of authenticating image
Content of authenticating image
Host Image at destination

Algo.

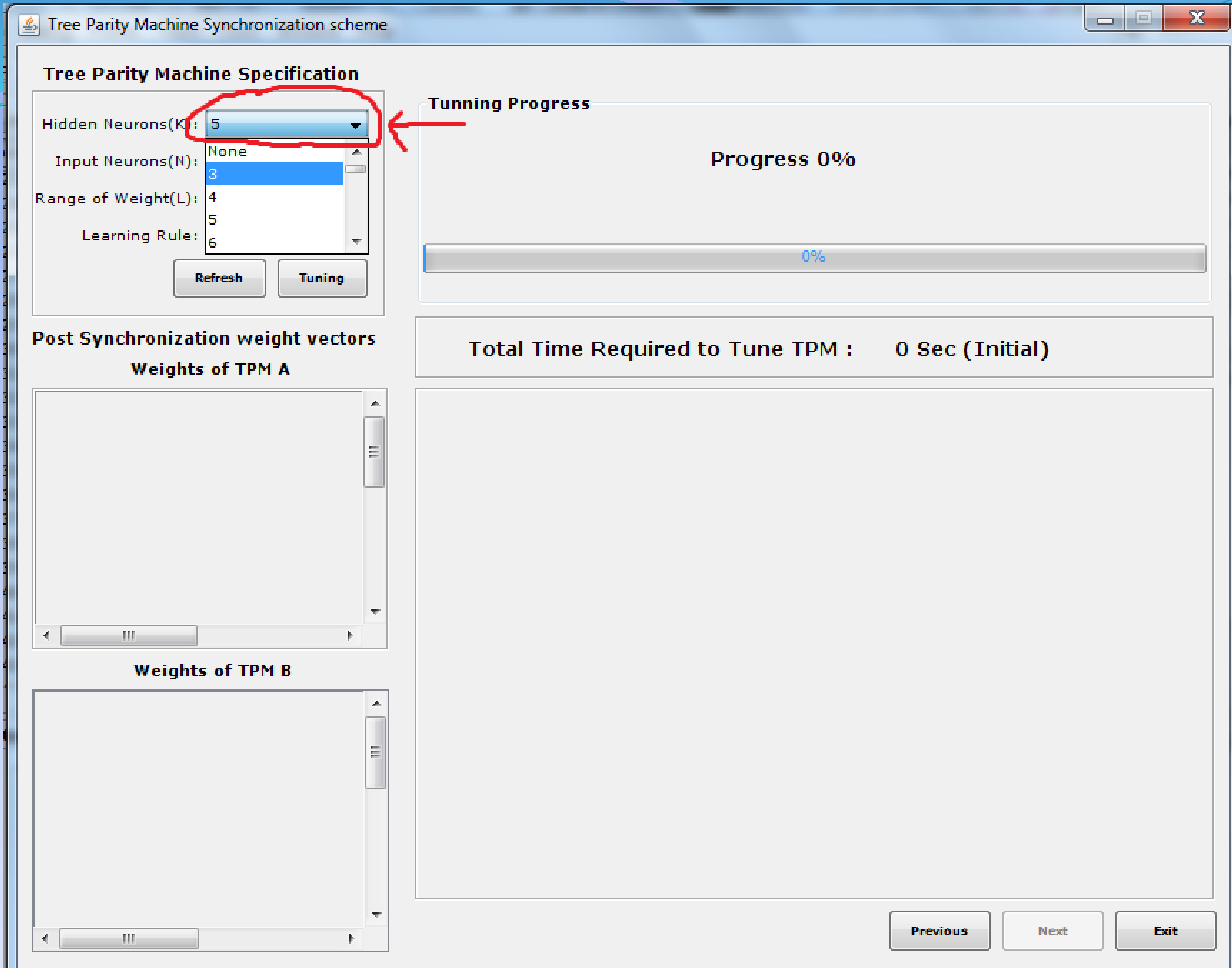
- Step 1: **Obtain the size of the authenticating image m x n.**
- **Step 2: For each authenticating message/image, Read source image block of size 3x3 in row major order. Extract authenticating message/image bit one by one. Replace the authenticating message/image bit in the rightmost 4 bits within the block, four bits in each byte.**
- **Step 3: Read one character/ pixel of the**
- **authenticating message/ image at a time.**
- **Step 4: Repeat step 2 and 3 for the whole**
- **authenticating message/ image size, content.**
- **Step 5: Perform mutation operation for the whole**
- **embedded image. For mutation rightmost 3**
- **bits from each bytes is taken. A consecutive**
- **bitwise XOR is performed on it for the 3**
- **steps. It will form a triangular form and first**
- **bit from each step is taken.**
- **Step 6: A bit handling method is performed on the**
- **embedded image. If the difference between**
- **the host and embedded image is ± 16**
- **then 16 will be added to the embedded**
- **image to keep intact the visibility of the**
- **embedded image.**

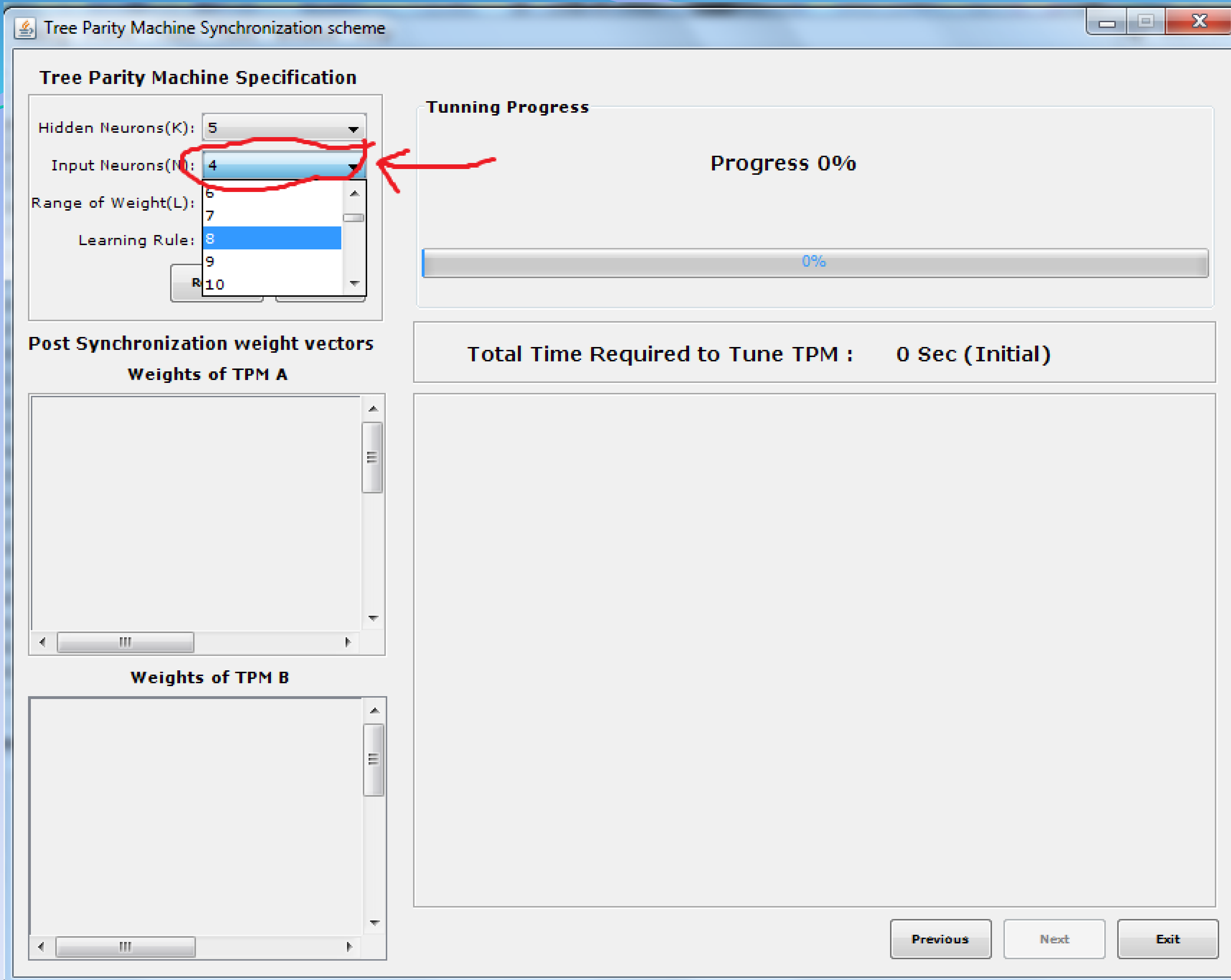
Future Scope

The proposed technique can be used to enhance security in mobile ad hoc network system through which adaptive transmission may be possible and which will be the future scope of the work. Security has become a primary concern in order to provide protected communication between mobile nodes in a hostile environment.

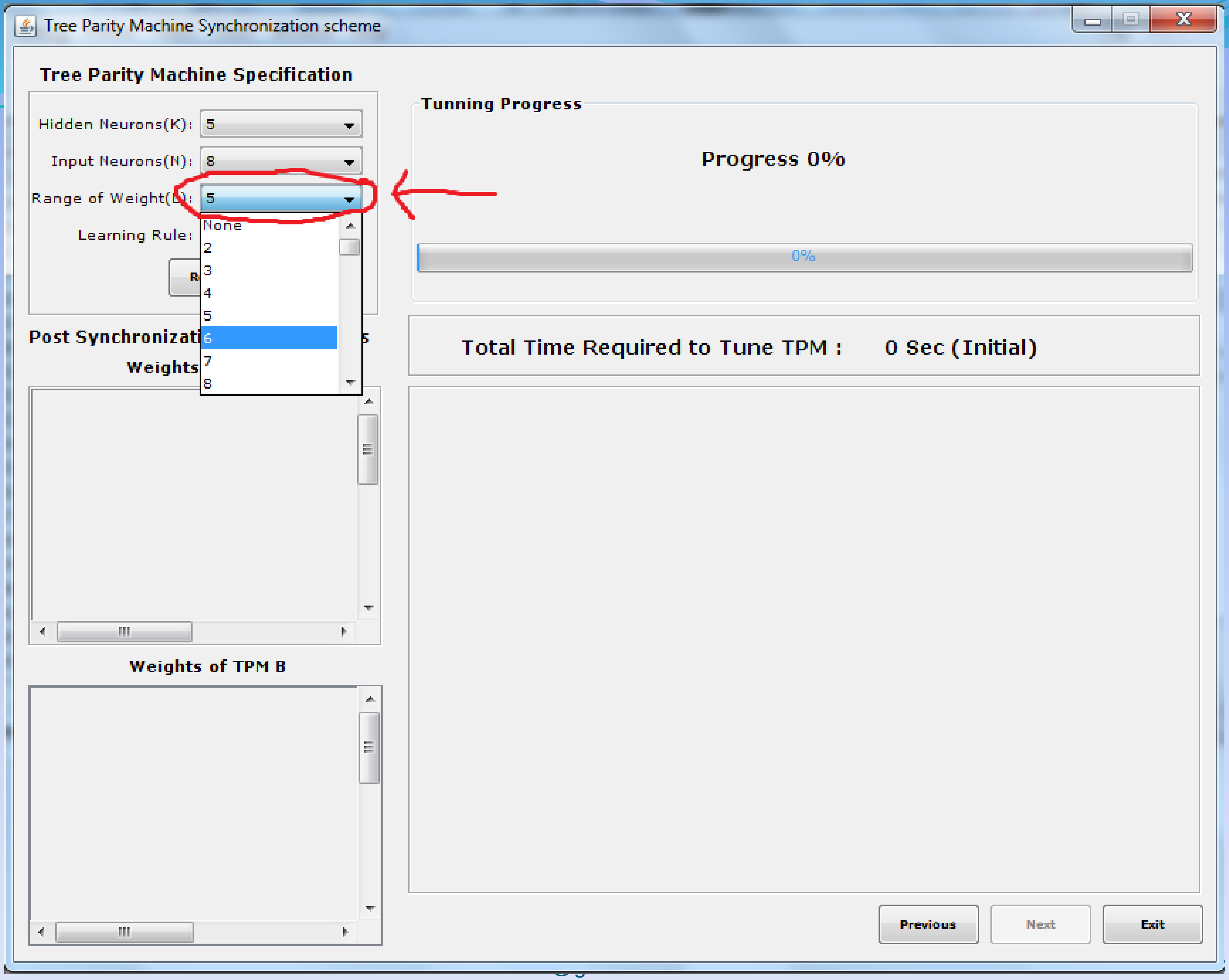
In recent years mobile ad hoc networks have received tremendous attention because of their self-configuration and self-maintenance capabilities. While early research effort assumed a friendly and cooperative environment and focused on problems such as wireless channel access and multihop routing, security has become a primary concern in order to provide protected communication between nodes in a potentially hostile environment.

SIMULATIONS

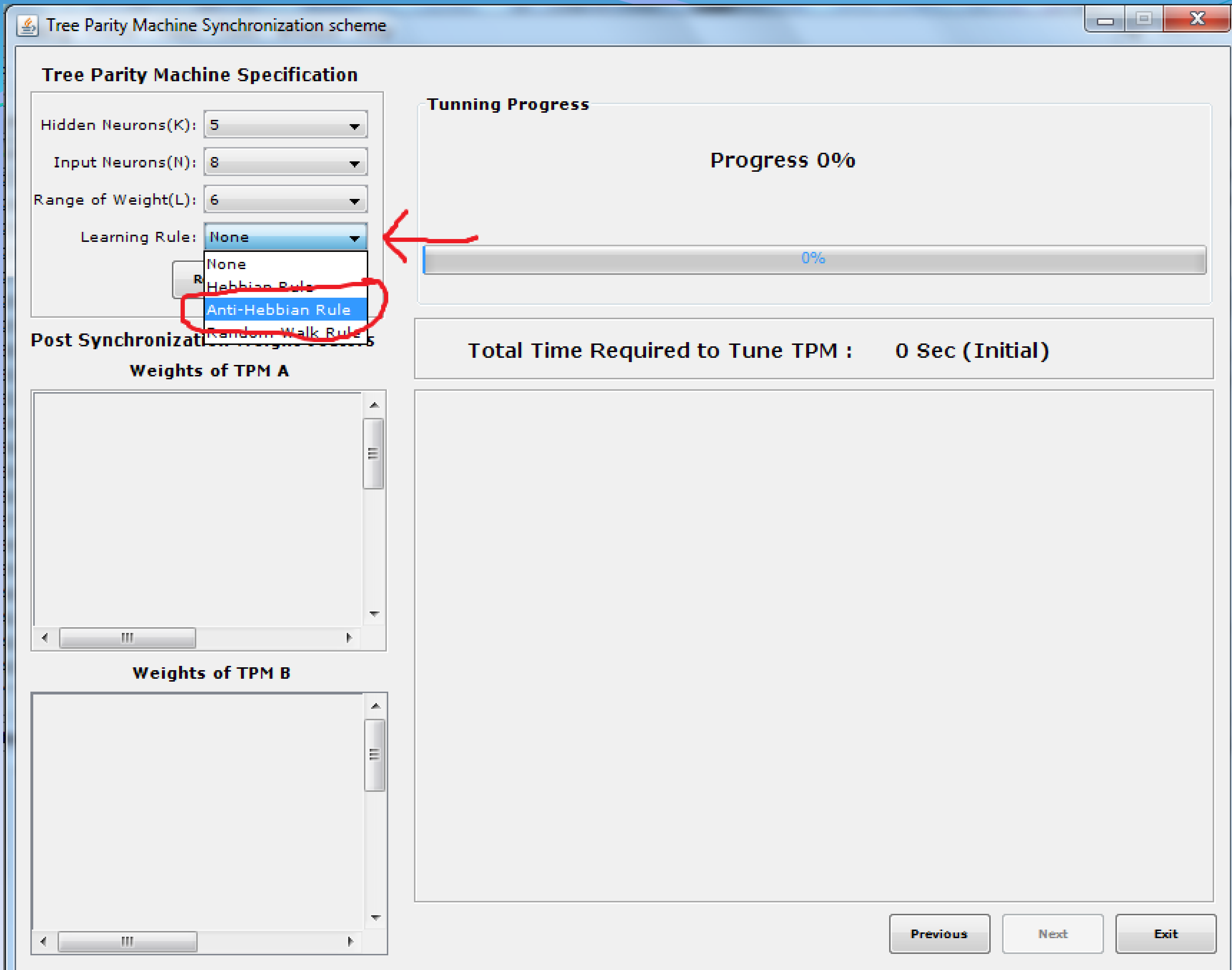

Tree Parity Machine Synchronization scheme
Tree Parity Machine Specification
Hidden Neurons(K): 5
None
3
4
5
6
Input Neurons(N):
Range of Weight(L):
Learning Rule:
Refresh
Tuning
Tunning Progress
Progress 0%
0%
Total Time Required to Tune TPM : 0 Sec (Initial)
Post Synchronization weight vectors
Weights of TPM A
Weights of TPM B
Previous
Next
Exit

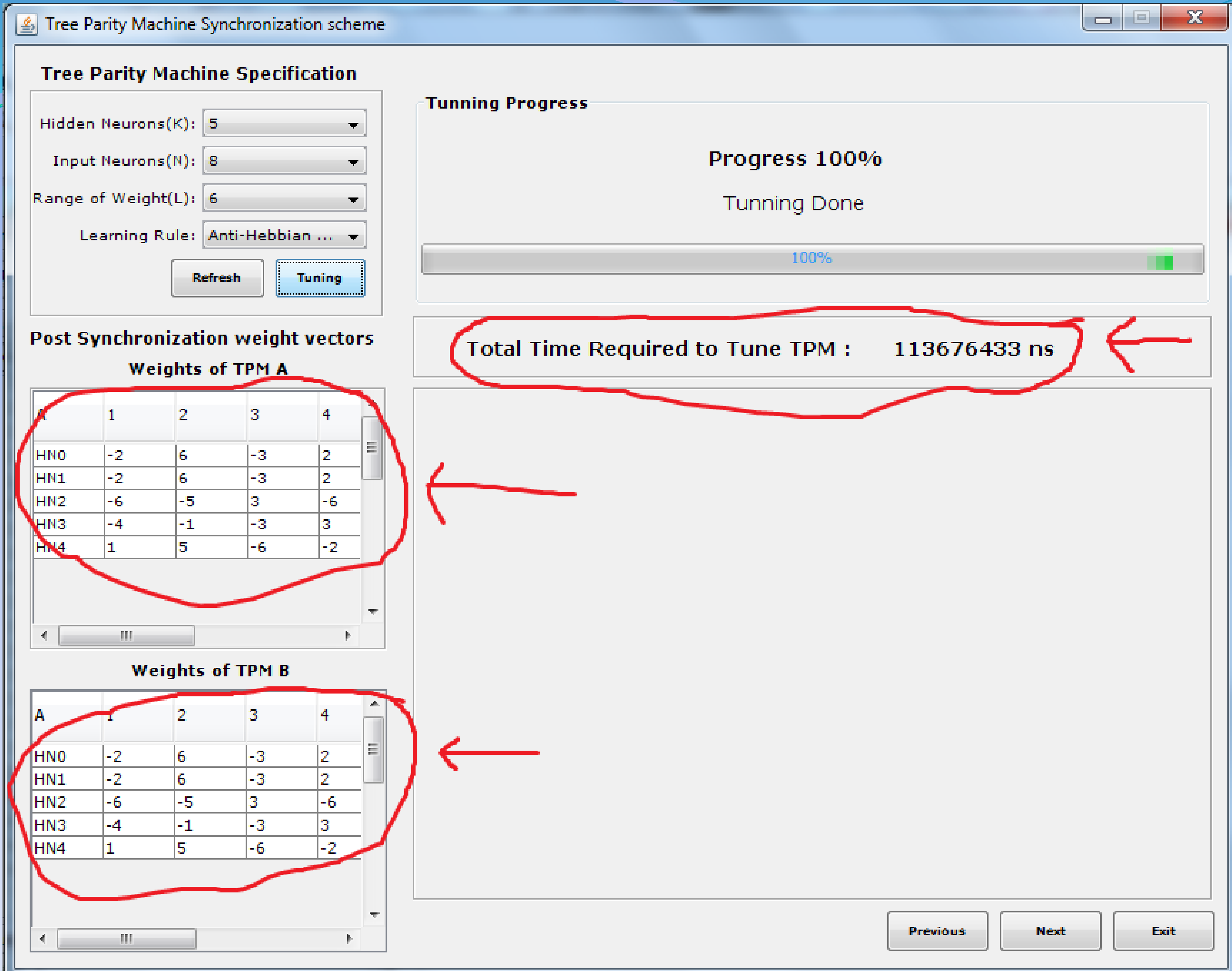


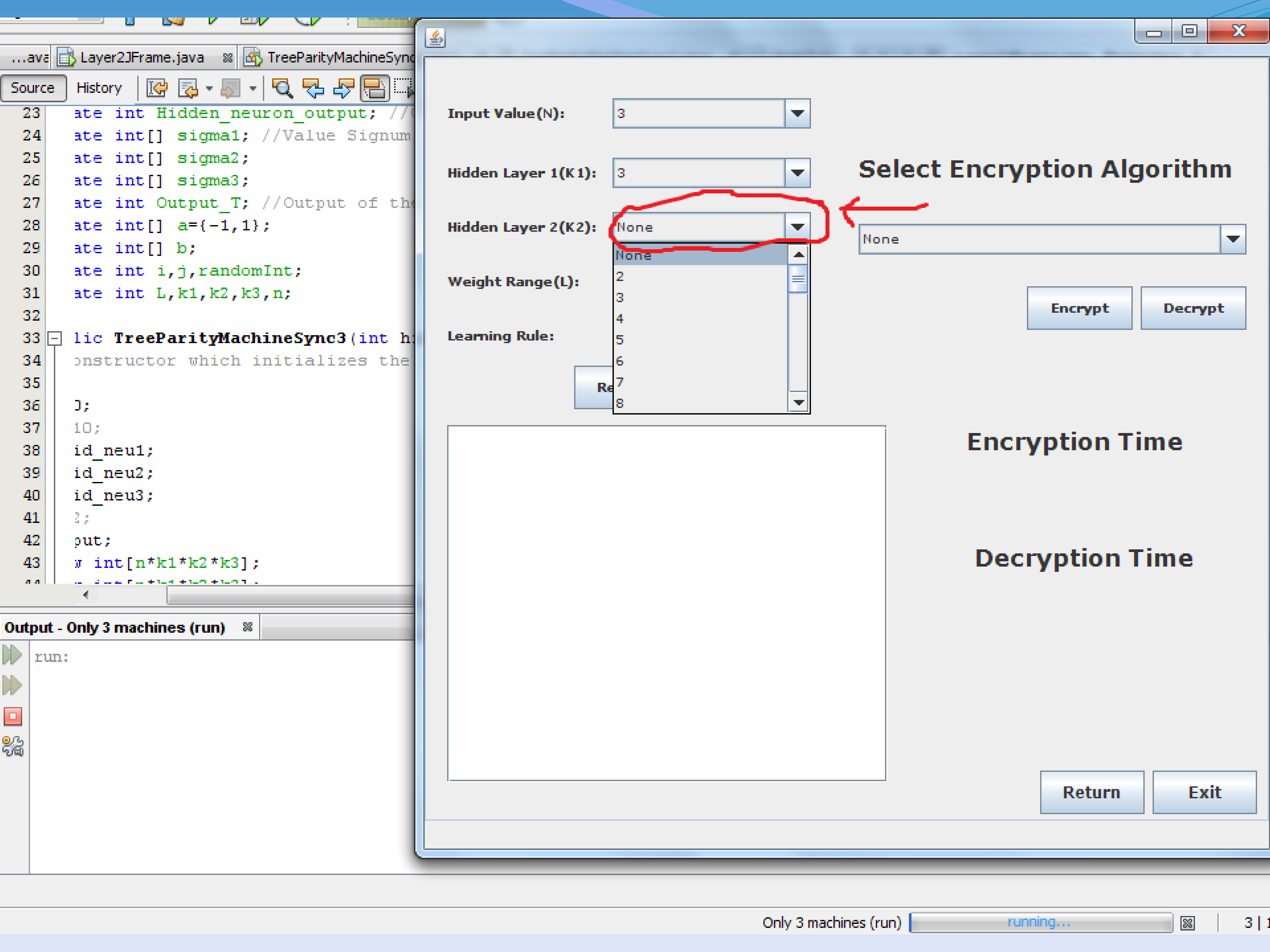
Tree Parity Machine Synchronization scheme
Tree Parity Machine Specification
Hidden Neurons(K): 5
Input Neurons(N): 4
6
7
8
9
10
Range of Weight(L):
Learning Rule:
Tunning Progress
Progress 0%
0%
Post Synchronization weight vectors
Weights of TPM A
Total Time Required to Tune TPM : 0 Sec (Initial)
Weights of TPM B
Previous
Next
Exit



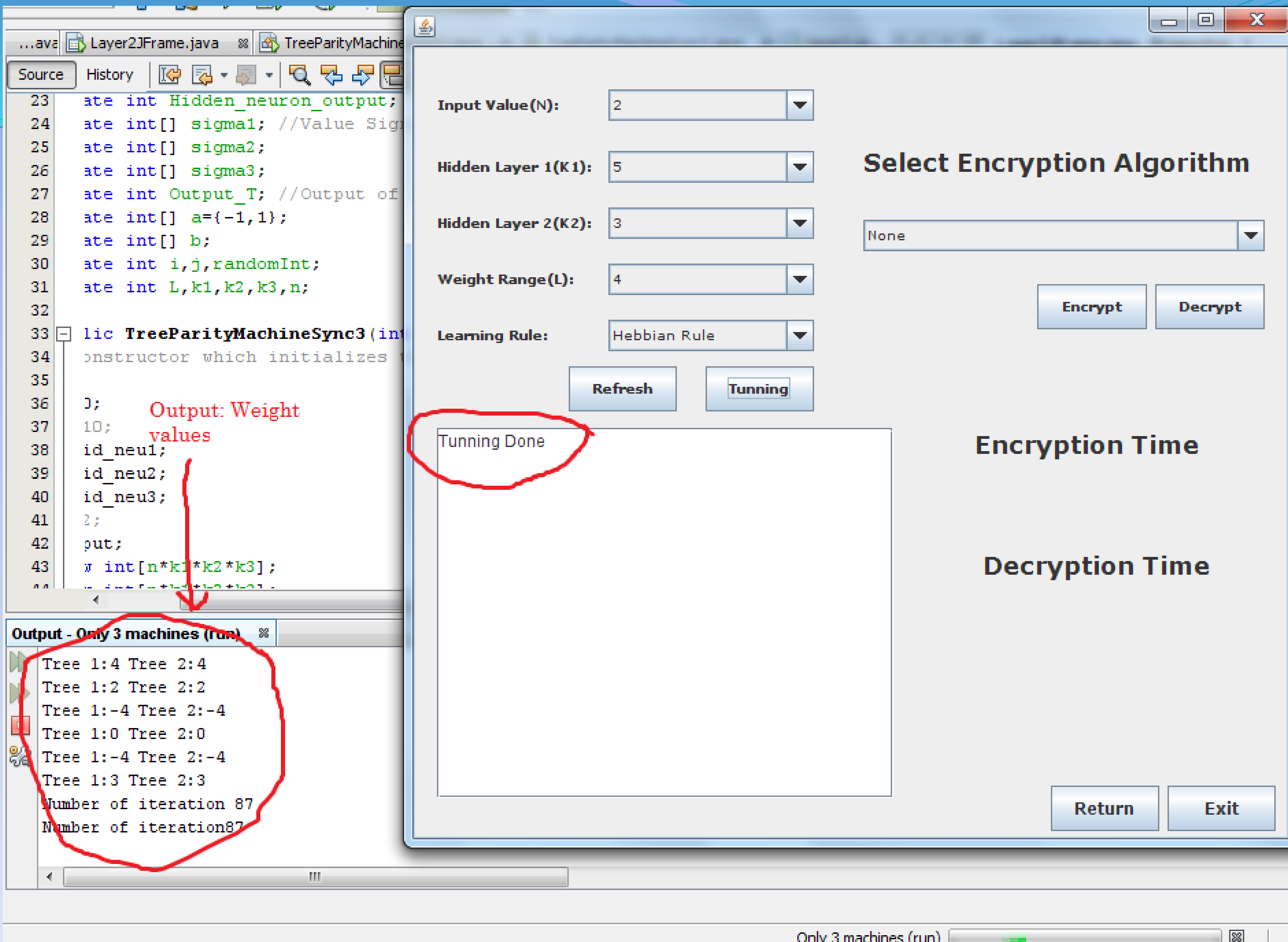
Tree Parity Machine Synchronization scheme
Tree Parity Machine Specification
Hidden Neurons(K): 5
Input Neurons(N): 8
Range of Weight(L): 5
Learning Rule:
None
2
3
4
5
6
7
8
Post Synchronizati
Weights
Weights of TPM B
Tunning Progress
Progress 0%
0%
Total Time Required to Tune TPM : 0 Sec (Initial)
Previous
Next
Exit


Tree Parity Machine Synchronization scheme
Tree Parity Machine Specification
Hidden Neurons(K): 5
Input Neurons(N): 8
Range of Weight(L): 6
Learning Rule: None
None
Hebbian Rule
Anti-Hebbian Rule
Random Walk Rule
Post Synchronization
Weights of TPM A
Weights of TPM B
Tunning Progress
Progress 0%
0%
Total Time Required to Tune TPM : 0 Sec (Initial)
Previous
Next
Exit

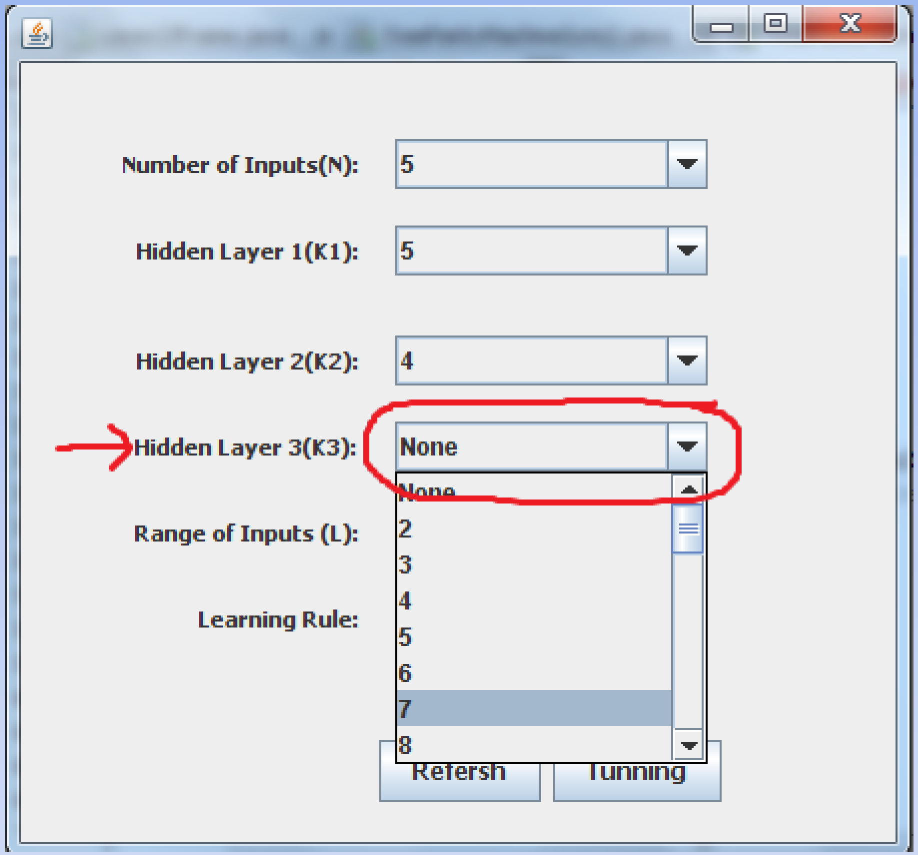

Tree Parity Machine Synchronization scheme
Tree Parity Machine Specification
Hidden Neurons(K): 5
Input Neurons(N): 8
Range of Weight(L): 6
Learning Rule: Anti-Hebbian ...
Refresh
Tuning
Tunning Progress
Progress 100%
Tunning Done
100%
Total Time Required to Tune TPM : 113676433 ns
Post Synchronization weight vectors
Weights of TPM A
A 1 2 3 4
HN0 -2 6 -3 2
HN1 -2 6 -3 2
HN2 -6 -5 3 -6
HN3 -4 -1 -3 3
HN4 1 5 -6 -2
Weights of TPM B
A 1 2 3 4
HN0 -2 6 -3 2
HN1 -2 6 -3 2
HN2 -6 -5 3 -6
HN3 -4 -1 -3 3
HN4 1 5 -6 -2
Previous
Next
Exit

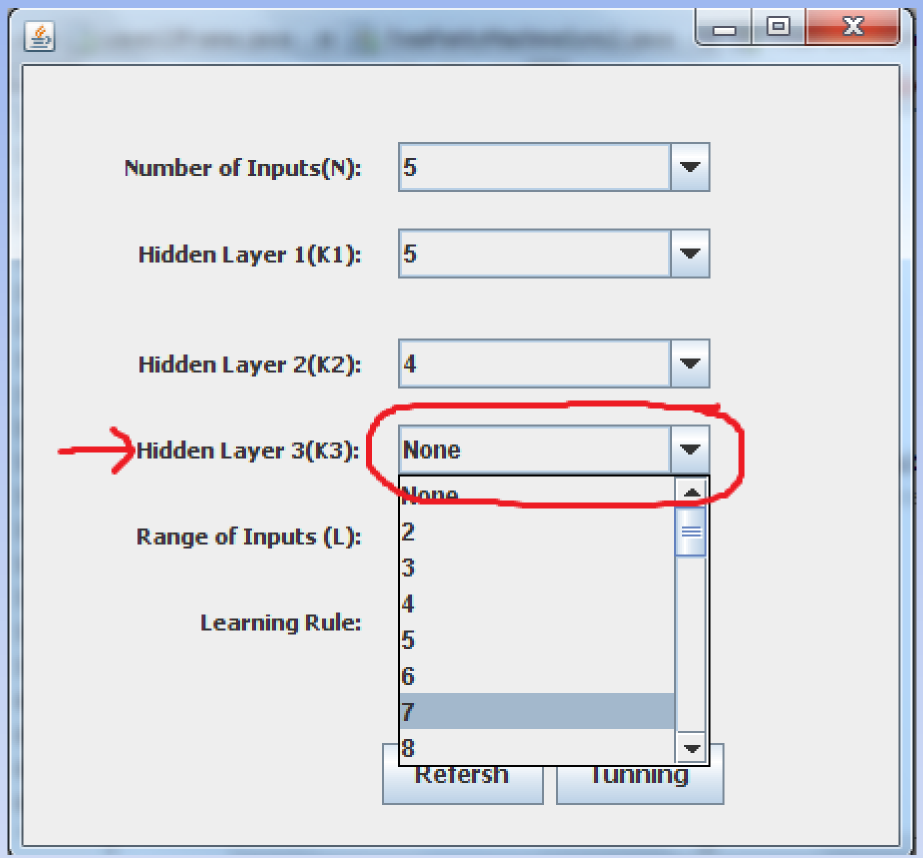


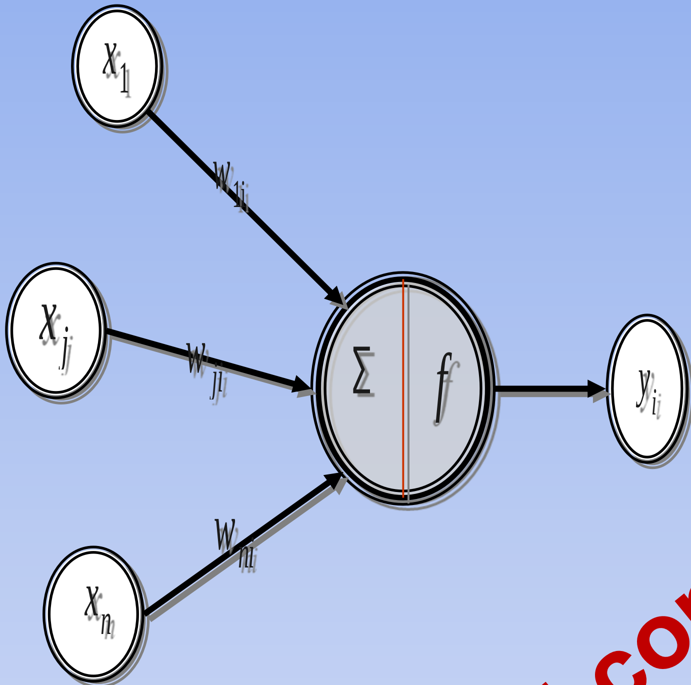
Layer2JFrame.java
TreeParityMachineSync
Source
History
ate int Hidden_neuron_output; //
ate int[] sigma1; //Value Signum
ate int[] sigma2;
ate int[] sigma3;
ate int Output_T; //Output of the
ate int[] a={-1,1};
ate int[] b;
ate int i,j,randomInt;
ate int L,k1,k2,k3,n;
lic TreeParityMachineSync3(int h
onstructor which initializes the
id_neu1;
id_neu2;
id_neu3;
put;
Output - Only 3 machines (run)
run:
Input Value(N):
3
Hidden Layer 1(K1):
3
Hidden Layer 2(K2):
None
None
2
3
4
5
6
7
8
Weight Range(L):
Learning Rule:
Select Encryption Algorithm
None
Encrypt
Decrypt
Encryption Time
Decryption Time
Return
Exit
Only 3 machines (run)
running...



Layer2JFrame.java
TreeParityMachine
Source
History
ate int Hidden_neuron_output;
ate int[] sigma1; //Value Sig
ate int[] sigma2;
ate int[] sigma3;
ate int Output_T; //Output of
ate int[] a={-1,1};
ate int[] b;
ate int i,j,randomInt;
ate int L,k1,k2,k3,n;
lic TreeParityMachineSync3(int
onstructor which initializes
Output: Weight values
id_neu1;
id_neu2;
id_neu3;
put;
int[n*k1*k2*k3];
Output - Only 3 machines (run)
Tree 1:4 Tree 2:4
Tree 1:2 Tree 2:2
Tree 1:-4 Tree 2:-4
Tree 1:0 Tree 2:0
Tree 1:-4 Tree 2:-4
Tree 1:3 Tree 2:3
Number of iteration 87
Number of iteration87
Input Value(N): 2
Hidden Layer 1(K1): 5
Hidden Layer 2(K2): 3
Weight Range(L): 4
Learning Rule: Hebbian Rule
Refresh
Tunning
Tunning Done
Select Encryption Algorithm
None
Encrypt
Decrypt
Encryption Time
Decryption Time
Return
Exit
Only 3 machines (run)

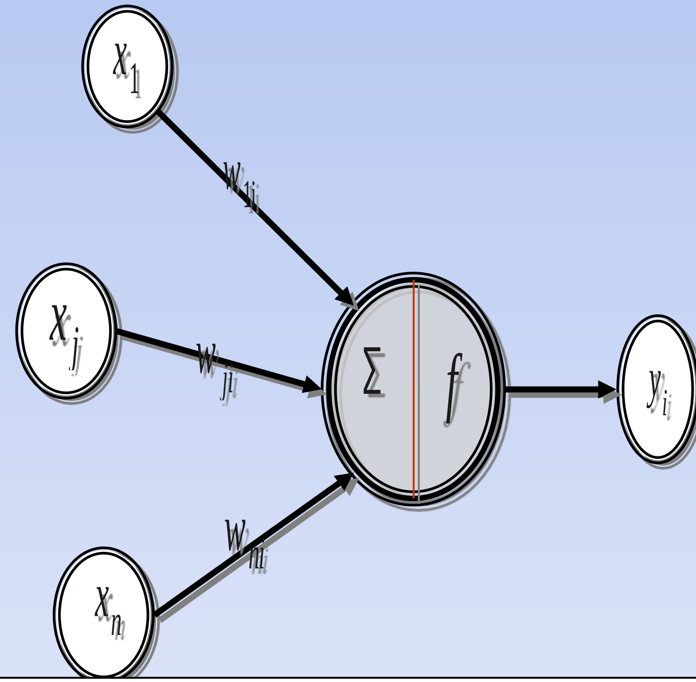

Number of Inputs(N):
5
Hidden Layer 1(K1):
5
Hidden Layer 2(K2):
4
Hidden Layer 3(K3):
None
None
2
3
4
5
6
7
8
Range of Inputs (L):
Learning Rule:
Refersh
Tunning

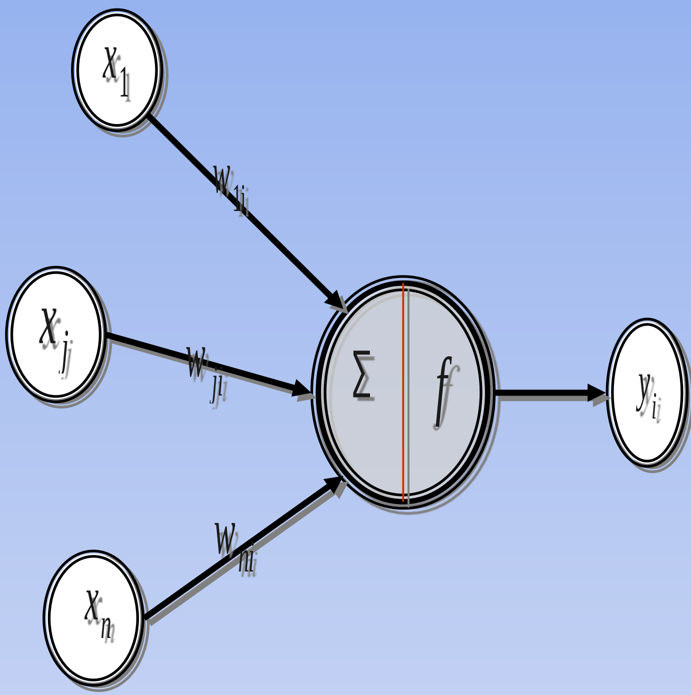

Number of Inputs(N):
5
Hidden Layer 1(K1):
5
Hidden Layer 2(K2):
4
Hidden Layer 3(K3):
None
Range of Inputs (L):
Learning Rule:
None
2
3
4
5
6
7
8
Refersh
Tunning



Questions?

jkm.cse@gmail.com





Thanks