

Program: M.Sc

Course Code:MSCS 2310

Course Name: Cyber Security

Course Outcomes :

CO NUMBER	TITLE
MSCS2310_CO1	Knowledge of cyber security principles used to manage risks related to the use, Processing, storage and transmission of information or data.
MSCS2310_CO2	Implementation of Cryptographic Techniques.
MSCS2310_CO3	Apply the authentication methods and prevent cyber crime
MSCS2310_CO4	Apply the procedures and algorithms to stop cyber Attacks and Threats.
MSCS2310_CO5	Understand the various cyber security policies and how to apply them.
MSCS2310_CO6	Understand the latest trends and advances of cyber security



Course Prerequisites

Cryptography and Network Security

Syllabus

UNIT I Introduction

9 hour

Overview of Cyber Security, Internet Governance – Challenges and Constraints, Cyber Threats:- Cyber Warfare-Cyber Crime-Cyber terrorism-Cyber Espionage, Need for a Comprehensive Cyber Security Policy, Need for a Nodal Authority, Need for an International convention on Cyberspace.

UNIT II Cryptographic Techniques

9 hours

Symmetric key cryptographic techniques: Introduction to Stream cipher – Block cipher: DES – AES- IDEA. Asymmetric key cryptographic techniques: principles – RSA – ElGamal - Elliptic Curve cryptography – Key distribution and Key exchange protocols.

UNIT III Authentication and Cybercrime

9 hours

Hash functions – Secure Hash Algorithm (SHA) Message Authentication – Message Authentication Code (MAC) – Digital Signature Algorithm: RSA & ElGamal based Classification of cybercrimes – planning of attacks – social engineering: Human based – Computer based – Cyberstalking – Cybercafe and Cybercrimes

UNIT IV Cyber Threats, Attacks and Prevention

9 hours

Phishing – Password cracking – Keyloggers and Spywares – DoS and DDoS attacks – SQL Injection. Identity Theft (ID) : Types of identity theft – Techniques of ID theft.

UNIT V Cyber Security Policies and Practices

9 hours

What security policies are – determining the policy needs – writing security policies – Internet and email security policies – Compliance and Enforcement of policies- Review.

Unit VI Research

9 hours

The advances and the latest trends in the course as well as the latest applications of the areas covered in the course.

The latest research conducted in the areas covered in the course.

Discussion of some latest papers published in IEEE transactions and ACM transactions, Web of Science and SCOPUS indexed journals as well as high impact factor conferences as well as symposiums.

Discussion on some of the latest products available in the market based on the areas covered in the course and patents filed in the areas covered.

Recommended Books

- 1. Charles P. P fleeger, Shari Lawerance P fleeger, “Analysing Computer Security”, Pearson Education India.**
- 2. V.K.Pachghare, “Cryptography and information Security”, PHI Learning Private Limited, Delhi India.**
- 3. Sarika Gupta & Gaurav Gupta, Information Security and Cyber Laws, Khanna Publishing House**
- 4. Anshul Kaushik, Cyber Security, Khanna Publishing House**

Lecture10

UNIT-2 OBJECTIVES

After studying this module you will be able to :

- Data Privacy(confidentiality)
- Data Authenticity(it came from where it claims)
- Data integrity(it has not been modified on the way) in the digital world.



Lecture10

UNIT-2 OBJECTIVES

CONFIDENTIALITY

- Confidentiality is most commonly addressed goal
- The meaning of a message is concealed by encoding it
- The sender encrypts the message using a cryptographic key
- The recipient decrypts the message using a cryptographic key that may or may not be the same as the one used by the sender

Lecture10

UNIT-2 OBJECTIVES

DATA INTEGRITY

- Integrity Ensures that the message received is the same as the message that was sent
- Uses hashing to create a unique message digest from the message that is sent along with the message
- Recipient uses the same technique to create a second digest from the message to compare to the original one
- This technique only protects against unintentional alteration of the message
- A variation is used to create digital signatures to protect against malicious alteration

Lecture10

UNIT-2 OBJECTIVES

AUTHENTICATION

- A user or system can prove their identity to another who does not have personal knowledge of their identity
- Accomplished using digital certificates
- Kerberos is a common cryptographic authentication system

Lecture10

Vision

To build a secure and resilient cyberspace for citizens, business, and government and also to protect anyone from intervening in your privacy.

Mission

To protect information and information infrastructure in cyberspace, build capabilities to prevent and respond to cyber threat, reduce vulnerabilities and minimize damage from cyber incidents through a combination of institutional structures, people, processes, technology, and cooperation.

Lecture10

CRYPTOGRAPHIC TECHNIQUES



Lecture10

- Cryptography (crypto)– study of how to mathematically encode & decode messages
- Cryptographic primitive (low-level) = algorithm
- Applied Cryptography – how to use crypto to achieve security goals (e.g. confidentiality)
- Primitives build up higher-level protocols (e.g. digital signature – only constructible by signer)
- Symmetric Encryption: Alice, Bob use same key

Lecture10

INTRODUCTION TO CRYPTOGRAPHY

1. Goal: Confidentiality

ISHMEET



“My account number is 485853 and my PIN is 4984”

MUSTAFA



EVE



Lecture10

SUBSTITUTION CIPHERS

- ❖ Plaintext: meet me at central park
- ❖ Ciphertext: phhw ph dw fhqwudo sdun
- ❖ Plain: abcdefghijklmnopqrstuvwxyz
- ❖ Cipher: defghijklmnopqrstuvwxyzabc
- ❖ Key is 3, i.e. shift letter right by 3
- ❖ Easy to break due to frequency of letters
- ❖ Good encryption algorithm produces output that looks random: equal probability any bit is 0 or 1

Lecture10

INTRODUCTION TO CRYPTOGRAPHY

1. Goal: Confidentiality

ISHMEET



“My account number is 485853 and my PIN is 4984”

MUSTAFA



- Message “sent in clear”: Eve can overhear
- Encryption unintelligible to Eve; only MUSTAFA can decipher with his secret key (shared w/ ISHMEET)



EVE



Lecture10

NOTATION & TERMINOLOGY

- m = message (plaintext), c = ciphertext
- F = encryption function
- F^{-1} = decryption function
- k = key (secret number)
- $c = F(m,k) = F_k(m)$ = encrypted m
- $m = F^{-1}(c,k) = F^{-1}_k(c)$ = decrypted message
- Symmetric cipher: $F^{-1}(F(m,k), k) = m$, same key

} Cipher Text

Lecture 10

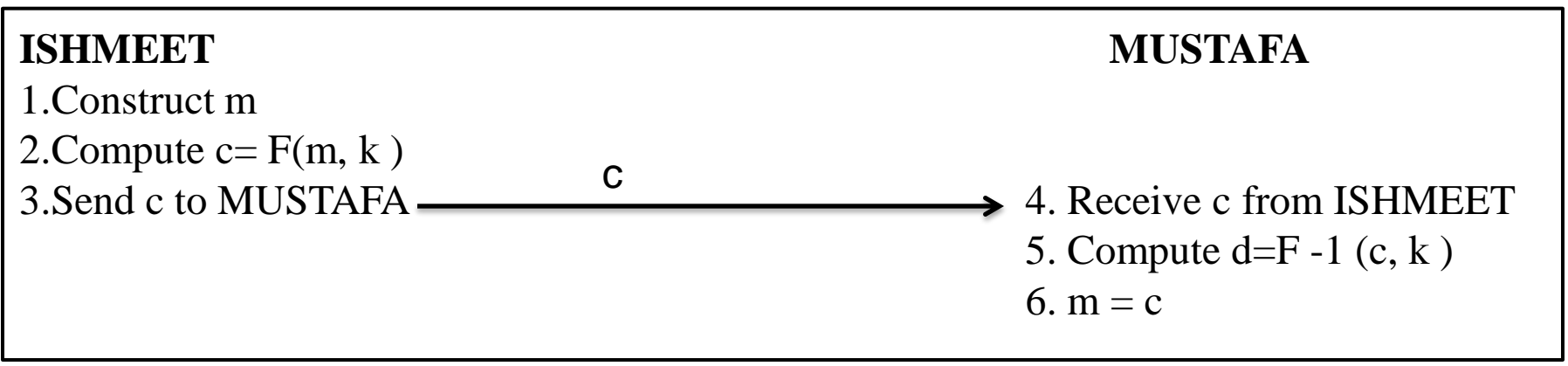
NOTATION & TERMINOLOGY

$m =$ message (plaintext)
 $\Rightarrow m =$ cyber security is an interesting subject
 $\Rightarrow c =$ ciphertext
 $\Rightarrow F =$ encryption function
 $F^{-1} =$ decryption function
 $\Rightarrow k =$ Key (secret number)
 $\Rightarrow c = F(m, k)$
 $\Rightarrow c = F(\text{cyber security is an interesting subject}, 1)$
 $= F_k(\text{cyber security is an interesting subject}, 1)$
 $\Rightarrow m = F^{-1}(c, k) = F^{-1}(\text{cyber security is an interesting subject}, 1)$
 Symmetric cipher

Lecture10

SYMMETRIC ENCRYPTION

- ISHMEET encrypts a message with the same key that MUSTAFA uses to decrypt.



- Eve can see c , but cannot compute m because k is only known to ISHMEET and MUSTAFA



Lecture10

STREAM CIPHER

- Much faster than block ciphers
- Encrypts one byte of plaintext at a time
- *Keystream*: infinite sequence (never reused) of random bits used as key
- Approximates theoretical scheme: one-time pad, trying to make it practical with finite keys

Lecture10

STREAM CIPHER

One-Time Pad

Key as long as plaintext, random stream of bits

Ciphertext = Key XOR Plaintext

Only use key once!

Impractical having key the same size as plaintext (too long, incurs too much overhead)

Theoretical Significance: “perfect secrecy” (Shannon) if key is random.

Under brute-force, every decryption equally likely

Ciphertext yields no info about plaintext (attacker’s a priori belief state about plaintext is unchanged)



Lecture10

STREAM CIPHER

RC4

- Most popular stream cipher: 10x faster than DES
- Fixed-size key “seed” to generate infinite stream
- State Table S that changes to create stream
- Ex: 256-bit key used to seed table (fill it)

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i], S[j])
t = (S[i]+S[j]) mod 256
K = S[t]
```



Lecture11

UNIT-2 OBJECTIVES

CONFIDENTIALITY

- Confidentiality is most commonly addressed goal
- The meaning of a message is concealed by encoding it
- The sender encrypts the message using a cryptographic key
- The recipient decrypts the message using a cryptographic key that may or may not be the same as the one used by the sender

Lecture11

UNIT-2 OBJECTIVES

DATA INTEGRITY

- Integrity Ensures that the message received is the same as the message that was sent
- Uses hashing to create a unique message digest from the message that is sent along with the message
- Recipient uses the same technique to create a second digest from the message to compare to the original one
- This technique only protects against unintentional alteration of the message
- A variation is used to create digital signatures to protect against malicious alteration

Lecture11

UNIT-2 OBJECTIVES

AUTHENTICATION

- A user or system can prove their identity to another who does not have personal knowledge of their identity
- Accomplished using digital certificates
- Kerberos is a common cryptographic authentication system

Lecture11

Vision

To build a secure and resilient cyberspace for citizens, business, and government and also to protect anyone from intervening in your privacy.

Mission

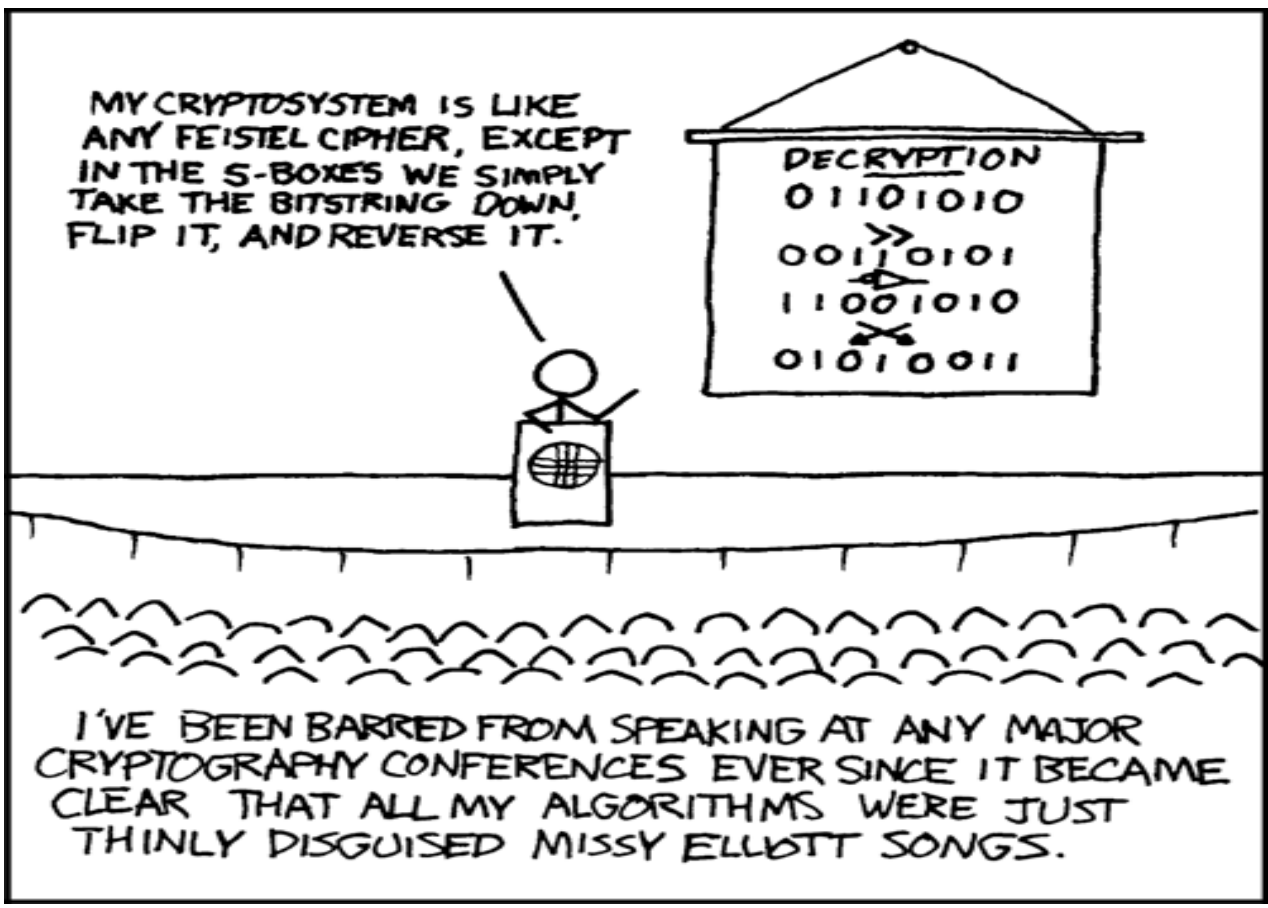
To protect information and information infrastructure in cyberspace, build capabilities to prevent and respond to cyber threat, reduce vulnerabilities and minimize damage from cyber incidents through a combination of institutional structures, people, processes, technology, and cooperation.

Lecture11

CRYPTOGRAPHIC TECHNIQUES

Lecture11

and other ciphers...



Lecture11

RC4 Pitfalls

- Never use the same key more than once!
- Clients & servers should use different RC4 keys!
 - C -> S: $P \oplus k$ [Eve captures $P \oplus k$] [Eve
 - S -> C: $Q \oplus k$ captures $Q \oplus k$]
 - Eve: $(P \oplus k) \oplus (Q \oplus k) = P \oplus Q!!!$
 - If Eve knows either P or Q, can figure out the other
- **Ex: Simple Mail Transfer Protocol (SMTP)**
 - First string client sends server is `HELO`
 - Then Eve could decipher first few bytes of response

Lecture11

More RC4 Pitfalls

- **Initial bytes of key stream are “weak”**
 - Ex: WEP protocol in 802.11 wireless standard is broken because of this
 - Discard first 256-512 bytes of stream

- **Active Eavesdropper**
 - Could flip bit without detection
 - Can solve by including MAC to protect integrity of ciphertext

Lecture12

UNIT-2 OBJECTIVES

After studying this module you will be able to :

- Data Privacy(confidentiality)
- Data Authenticity(it came from where it claims)
- Data integrity(it has not been modified on the way) in the digital world.



Lecture12

UNIT-2 OBJECTIVES

CONFIDENTIALITY

- Confidentiality is most commonly addressed goal
- The meaning of a message is concealed by encoding it
- The sender encrypts the message using a cryptographic key
- The recipient decrypts the message using a cryptographic key that may or may not be the same as the one used by the sender



Lecture11

UNIT-2 OBJECTIVES

DATA INTEGRITY

- Integrity Ensures that the message received is the same as the message that was sent
- Uses hashing to create a unique message digest from the message that is sent along with the message
- Recipient uses the same technique to create a second digest from the message to compare to the original one
- This technique only protects against unintentional alteration of the message
- A variation is used to create digital signatures to protect against malicious alteration

Lecture11

UNIT-2 OBJECTIVES

AUTHENTICATION

- A user or system can prove their identity to another who does not have personal knowledge of their identity
- Accomplished using digital certificates
- Kerberos is a common cryptographic authentication system

Lecture12

Vision

To build a secure and resilient cyberspace for citizens, business, and government and also to protect anyone from intervening in your privacy.

Mission

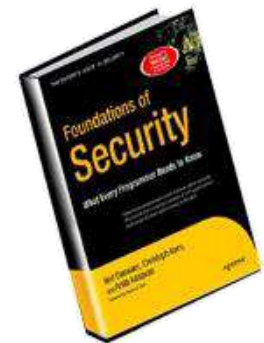
To protect information and information infrastructure in cyberspace, build capabilities to prevent and respond to cyber threat, reduce vulnerabilities and minimize damage from cyber incidents through a combination of institutional structures, people, processes, technology, and cooperation.

Lecture12

CRYPTOGRAPHIC TECHNIQUES

Symmetric Key Cryptography

Slides adapted from "Foundations of Security: What Every Programmer Needs To Know" by Neil Daswani, Christoph Kern, and Anita Kesavan (ISBN 1590597842; <http://www.foundationsofsecurity.com>). Except as otherwise noted, the content of this presentation is licensed under the Creative Commons 3.0 License.





Agenda

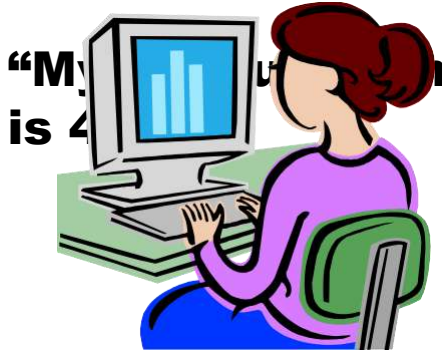
- *Cryptography* (crypto)– study of how to mathematically encode & decode messages
- *Cryptographic primitive* (low-level) = algorithm

- Applied Cryptography – how to use crypto to achieve security goals (e.g. confidentiality)
- Primitives build up higher-level protocols (e.g. *digital signature* – only constructible by signer)
- Symmetric Encryption: Alice, Bob use same key

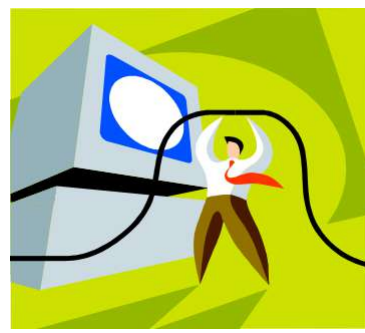
Introduction to Cryptography

- Goal: Confidentiality

Alice



“My account number is 485853 and my PIN is 4”



Eve

- Message “sent in clear”: Eve can overhear
- Encryption unintelligible to Eve; only Bob can decipher with his secret key (shared w/ Alice)



Substitution Ciphers

- Plaintext: `meet me at central park`
- Ciphertext: `phhw ph dw fhqwudo sdun`

- Plain: `abcdefghijklmnopqrstu`
`vwxyz`
- Cipher: `defghijklmnopqrstu`
`vwxyzabc`

- Key is 3, i.e. shift letter right by 3
- Easy to break due to frequency of letters
- Good encryption algorithm produces output that looks random: equal probability any bit is 0 or 1



Notation & Terminology

- m = message (plaintext), c = ciphertext
 - F = encryption function
 - F^{-1} = decryption function
 - k = key (secret number)
- } **Cipher**
- $c = F(m, k) = F_k(m)$ = encrypted message
 - $m = F^{-1}(c, k) = F^{-1}_k(c)$ = decrypted message
 - Symmetric cipher: $F^{-1}(F(m, k), k) = m$, same key

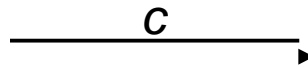


Symmetric Encryption

- Alice encrypts a message with the **same** key that Bob uses to decrypt.

Alice

1. **Construct** m
2. **Compute** $c = F(m, k)$
3. **Send** c to **Bob**



Bob

4. **Receive** c from **Alice**
5. **Compute** $d = F^{-1}(c, k)$
6. $m = d$

- Eve can see c , but cannot compute m because k is only known to Alice and Bob



Block Ciphers

- Blocks of bits (e.g. 256) encrypted at a time

- Examples of several algorithms:
 - Data Encryption Standard (DES)
 - Triple DES
 - Advanced Encryption Standard (AES) or Rijndael

- Internal Data Encryption Algorithm (IDEA), Blowfish, Skipjack, many more... (c.f. Schneier)

$m =$ message (plaintext)
 $\Rightarrow m =$ cyber security is an interesting subject
 $\Rightarrow c =$ ciphertext
 $\Rightarrow F =$ encryption function
 $F^{-1} =$ decryption function
 $\Rightarrow k =$ Key (secret number)
 $\Rightarrow c = F(m, k)$
~~gshmeet~~
 $\Rightarrow c = F(\text{cyber security is an interesting subject}, 1)$
 $= F_k(\text{cyber security is an interesting subject}, 1)$
~~Habib~~
 $m = F^{-1}(c, k) = F^{-1}(\text{cyber security is an interesting subject}, 1)$
 Symmetric cipher

ISHMEET 10110110 Message at sender side

KEY 01101101

11011011

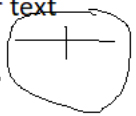


cipher text from ishmeet

HABIB 11011011 Cipher text

Key 01101101

10110110





DES

- Adopted in 1977 by NIST
- Input: 64-bit plaintext, 56-bit key (64 w/ parity)
- Parity Bits: redundancy to detect corrupted keys
- Output: 64-bit ciphertext
- Susceptible to Brute-Force (try all 2^{56} keys)
 - 1998: machine Deep Crack breaks it in 56 hours
 - Subsequently been able to break even faster
 - Key size should be at least 128 bits to be safe



TripleDES

- Do DES thrice w/ 3 different keys (slower)
- $c = F(F^{-1}(F(m, k_1), k_2), k_3)$ where $F = DES$
 - Why decrypt with k_2 ?
 - Backwards compatible w/ DES, easy upgrade
- Keying Options: Key Size (w/ Parity)
 - $k_1 \neq k_2 \neq k_3$: 168-bit (192-bit)
 - $k_1 = k_3 \neq k_2$: 112-bit (128-bit) 56-bit (64-bit)
 - $k_1 = k_2 = k_3$: (DES)



AES (Rijndael)

- ❑ Invented by 2 Belgian cryptographers
- ❑ Selected by NIST from 15 competitors after three years of conferences vetting proposals
- ❑ Selection Criteria:
 - ❑ Security, Cost (Speed/Memory)
 - ❑ Implementation Considerations (Hardware/Software)
- ❑ Key size & Block size: 128, 192, or 256 bits (much larger than DES)
- ❑ Rely on algorithmic properties for security, not obscurity



12.1.4. Security by Obscurity: Recap

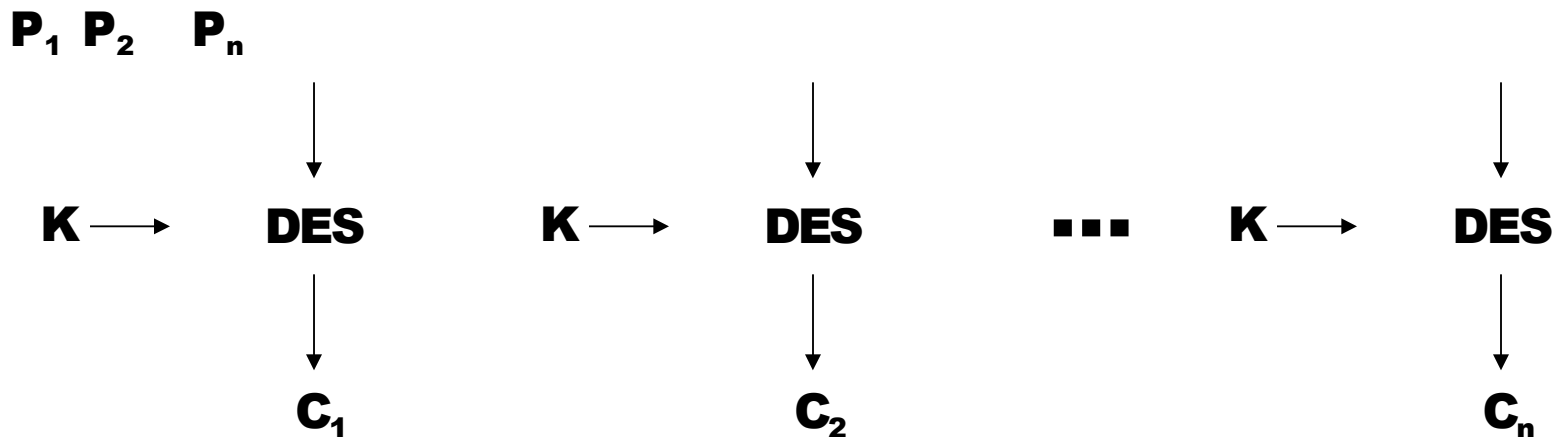
- Design of DES, Triple DES algorithms public
 - Security not dependent on secrecy of implementation
 - But rather on secrecy of key

- Benefits of Keys:
 - Easy to replace if compromised
 - Increasing size by one bit, doubles attacker's work

- If invent own algorithm, make it public! Rely on algorithmic properties (math), not obscurity.

Electronic Code Book

- Encrypting more data: ECB encrypt blocks of data in a large document



- Leaks info about structure of document (e.g. repeated plaintext blocks)

Review of XOR

- Exclusive OR (either x or y but not both)

x	y	$x \text{ XOR } y$
0	0	0

- Special Properties:

- $x \text{ XOR } y = z$

- $z \text{ XOR } y = x$

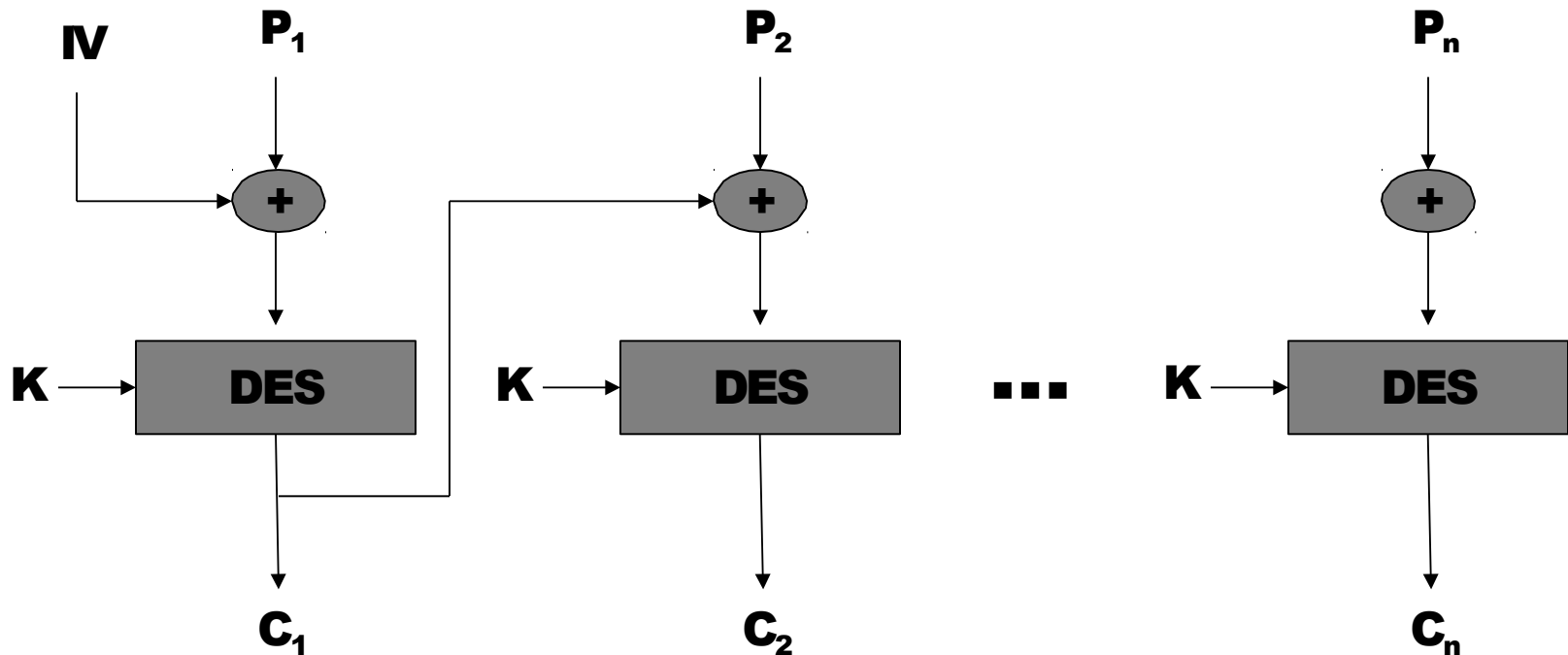
- $x \text{ XOR } z = y$

0	1	1
1	0	1
1	1	0

Cipher Block

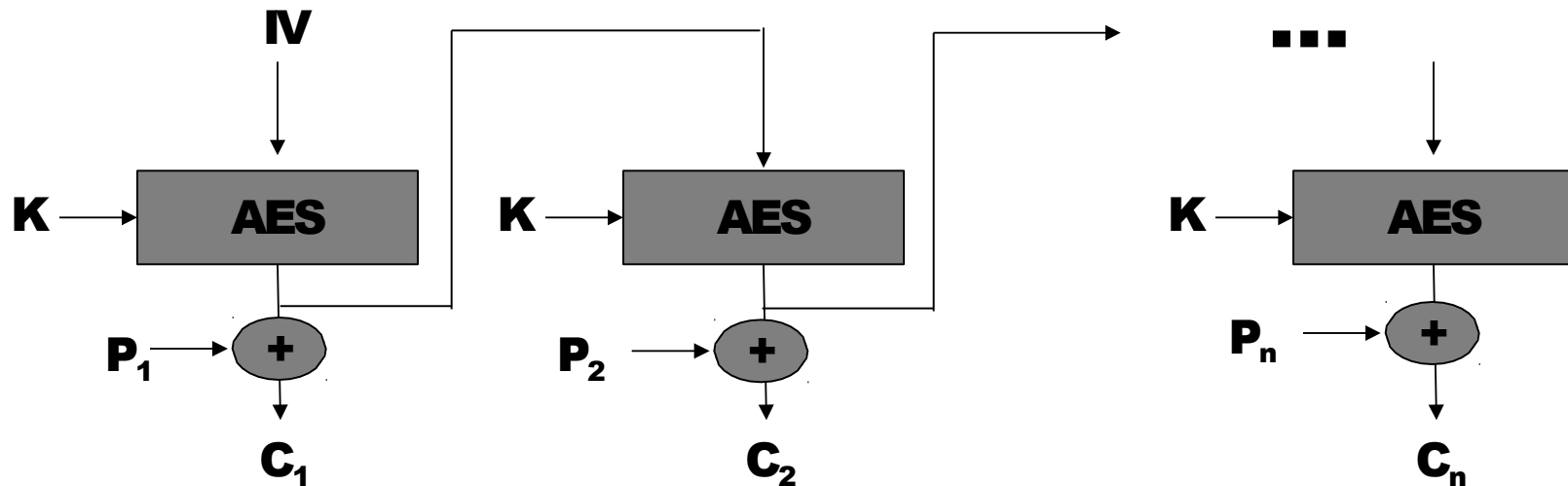
Chaining

- CBC: uses XOR, no patterns leaked!
- Each ciphertext block depends on prev blk



Output Feedback (OFB)

- Makes block cipher into stream cipher
- Like CBC, but do XOR after encryption



AES Code Example

- Example Java Class: `AESCrypter`
- Command-line utility:
 - Create AES key
 - Encrypt & Decrypt with key
 - AES in CBC mode
- Arguments: `<command> <keyfile>`
 - `command = createkey|encrypt|decrypt`
 - Input/output from `stdin` and `stdout`

Using AESEncrypter

- Alice generates a key and encrypts a message:

```
$ java AESEncrypter createkey mykey  
$ echo "Meet Me At Central Park" |  
java AESEncrypter encrypt mykey > ciphertext
```

- She gives Bob `mykey` over *secure* channel, then can send `ciphertext` over insecure channel

- Bob can decrypt Alice's message with `mykey`:

```
$ java com.learnsecurity.AESEncrypter decrypt mykey < ciphertext  
Meet Me At Central Park
```

AESCrypter: Members & Constructor

```
/* Import Java Security & Crypto packages, I/O library */

public class AESCrypter {
public static final int IV_SIZE = 16; // 128 bits public
static final int KEY_SIZE = 16; // 128 bits public static
final int BUFFER_SIZE = 1024; // 1KB
Cipher cipher; /* Does encryption and decryption */
SecretKey secretKey;
AlgorithmParameterSpec ivSpec; /* Initial Value - IV */
byte[] buf = new byte[BUFFER_SIZE];
byte[] ivBytes = new byte [IV_SIZE]; /* inits ivSpec */

public AESCrypter(SecretKey key) throws Exception {
cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
/* Use AES, pad input to 128-bit multiple */
secretKey = key;
}
// ... Methods Follow ...
}
```

AESCrypter: encrypt()

```
public void encrypt(InputStream in,
OutputStream out) throws Exception {
ivBytes = createRandBytes(IV_SIZE); // create IV & write to output
out.write(ivBytes);
ivSpec = new IvParameterSpec(ivBytes);
cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec);
// cipher initialized to encrypt, given secret key, IV

// Bytes written to cipherOut will be encrypted
CipherOutputStream cipherOut = new CipherOutputStream(out, cipher);

// Read in the plaintext bytes and write to cipherOut to encrypt  int
numRead = 0;
while ((numRead = in.read(buf)) >= 0) // read plaintext
cipherOut.write(buf, 0, numRead); // write ciphertext
cipherOut.close(); // padded to 128-bit multiple
}
```


AESDecryptor: decrypt()

```
public void decrypt(InputStream in,
OutputStream out) throws Exception {
// read IV first  System.in.read(ivBytes);
ivSpec = new IvParameterSpec(ivBytes);

cipher.init(Cipher.DECRYPT_MODE, secretKey, ivSpec);
// cipher initialized to decrypt, given secret key, IV

// Bytes read from in will be decrypted
CipherInputStream cipherIn = new CipherInputStream(in, cipher);

// Read in the decrypted bytes and write the plaintext to out  int
numRead = 0;
while ((numRead = cipherIn.read(buf)) >= 0) // read ciphertext
out.write(buf, 0, numRead); // write plaintext
out.close();
}
```

AESCryptor: main()

```
public static void main (String[] args) throws Exception {
    if (args.length != 2) usage(); // improper usage, print error
    String operation = args[0]; // createkey|encrypt|decrypt String
    keyFile = args[1]; // name of key file
    if (operation.equals("createkey")) {
        FileOutputStream fos = new FileOutputStream(keyFile); KeyGenerator
        kg = KeyGenerator.getInstance("AES"); kg.init(KEY_SIZE*8); // key
        size in bits
        SecretKey skey = kg.generateKey(); fos.write(skey.getEncoded()); //
        write key fos.close();
    } else {
        byte[] keyBytes = new byte[KEY_SIZE]; FileInputStream fis = new
        FileInputStream(keyFile); fis.read(keyBytes); // read key
        SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES");
        AESCryptor aes = new AESCryptor(keySpec); // init w/ key if
        (operation.equals("encrypt")) {
            aes.encrypt(System.in, System.out); // Encrypt
        } else if (operation.equals("decrypt")) {
            aes.decrypt(System.in, System.out); // Decrypt
        } else usage(); // improper usage, print error
    }
}
```

AESEncryptor: Helpers

```
/* Generate numBytes of random bytes to use as IV */ public
static byte[] createRandBytes(int numBytes)
throws NoSuchAlgorithmException {
byte[] bytesBuffer = new byte[numBytes];
SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
sr.nextBytes(bytesBuffer);
return bytesBuffer;
}
```

```
/* Display error message when AESEncryptor improperly used */
public static void usage () {
System.err.println("java com.learnsecurity.AESEncryptor " +
"createkey|encrypt|decrypt <keyfile>");
System.exit(-1);
}
```

AESCryptor Recap

- Java class `KeyGenerator` can be used to construct strong, cryptographically random keys
- `AESCryptor`: no integrity protection
 - Encrypted file could be modified
 - So in practice, should tag on a MAC
 - Use different keys for MAC and encryption
- Key Distribution is a challenge (c.f. Ch. 13-14)

Stream Ciphers

- Much faster than block ciphers
- Encrypts one byte of plaintext at a time
- *Keystream*: infinite sequence (never reused) of random bits used as key
- Approximates theoretical scheme: one-time pad, trying to make it practical with finite keys

One-Time Pad

- Key as long as plaintext, random stream of bits
 - Ciphertext = Key XOR Plaintext
 - Only use key once!
- Impractical having key the same size as plaintext (too long, incurs too much overhead)
- Theoretical Significance: “perfect secrecy” (Shannon) if key is random.
 - Under brute-force, every decryption equally likely
 - Ciphertext yields no info about plaintext (attacker’s a priori belief state about plaintext is unchanged)

RC4

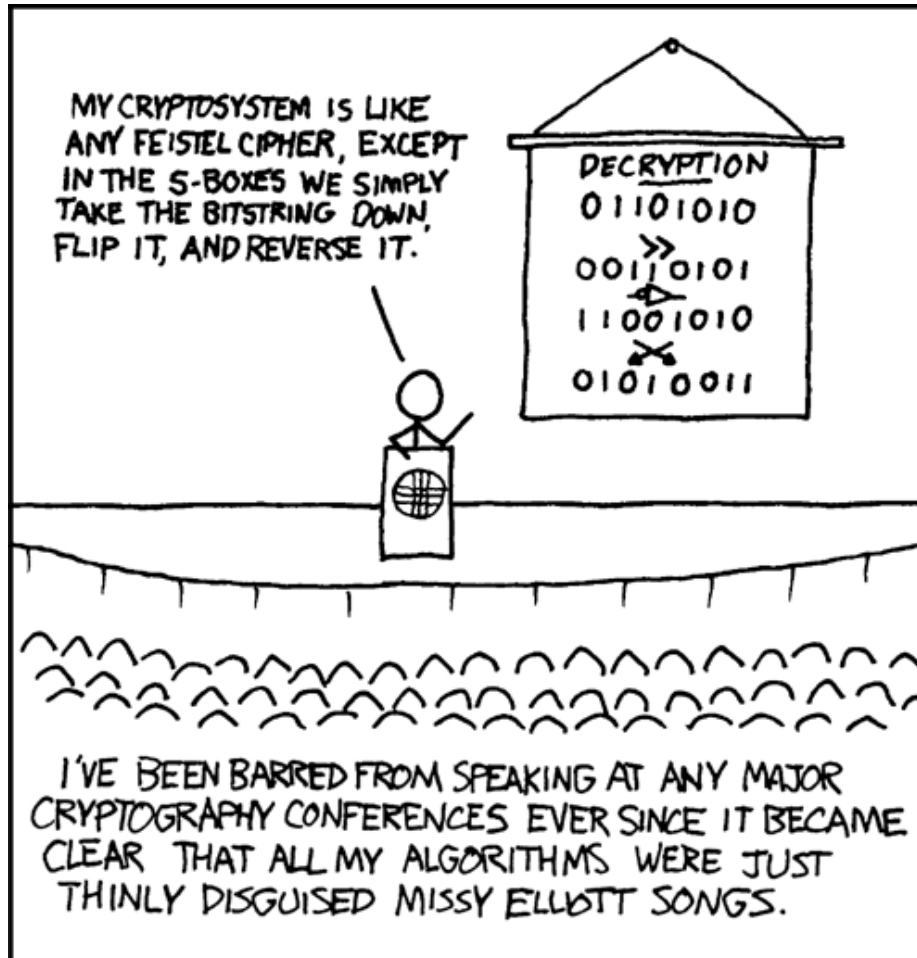
- Most popular stream cipher: 10x faster than DES
- Fixed-size key “seed” to generate infinite stream
- *State Table S* that changes to create stream
- Ex: 256-bit key used to seed table (fill it)

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i], S[j])
t = (S[i] + S[j]) mod 256
K = S[t]
```

```
t = (S[S[i] + S[j]])
```

```
K = S[t]
```

and other ciphers...



Source:
<http://xkcd.com/153/>

RC4 Pitfalls

- Never use the same key more than once!
- Clients & servers should use different RC4 keys
 - C -> S: $P \text{ XOR } k$ [Eve captures $P \text{ XOR } k$]
 - S -> C: $Q \text{ XOR } k$ [Eve captures $Q \text{ XOR } k$]
 - Eve: $(P \text{ XOR } k) \text{ XOR } (Q \text{ XOR } k) = P \text{ XOR } Q!!!$
 - If Eve knows either P or Q, can figure out the other
- Ex: Simple Mail Transfer Protocol (SMTP)
 - First string client sends server is `HELO`
 - Then Eve could decipher first few bytes of response

More RC4 Pitfalls



- Initial bytes of key stream are “weak”
 - Ex: WEP protocol in 802.11 wireless standard is broken because of this
 - Discard first 256-512 bytes of stream
- Active Eavesdropper
 - Could flip bit without detection
 - Can solve by including MAC to protect integrity of ciphertext

Steganography

- All ciphers transform plaintext to random bits
- Eve can tell Alice is sending sensitive info to Bob
- Conceal existence of secret message
- Use of a “covert channel” to send a message.

What is Steganography?

- Study of techniques to send sensitive info and hide the fact that sensitive info is being sent
- Ex: “All the tools are carefully kept” -> Attack
- Other Examples: Invisible ink, Hidden in Images
 - Least significant bit of image pixels
 - Modifications to image not noticeable by an observer
 - Recipient can check for modifications to get message

Red	Green	Blue	
00000000	00000000	00000000	
00000001	00000000	00000001	 → 101

Steganography vs. Cryptography

- Key Advantage: when Alice & Bob don't want Eve to know that they're communicating secrets
 - Disadvantages compared to encryption
 - Essentially relying on security by obscurity
 - Useless once covert channel is discovered
- High overhead (ratio of plain bits/secret bits high)
- Can be used together with encryption, but even more overhead (additional computation for both)



Summary

- Cryptography: encode & decode messages
- Applied to serve security goals (confidentiality)

- Symmetric Ciphers: Alice & Bob have same key
 - Block Ciphers: DES, AES (128-bit blocks at a time)
 - Stream Ciphers: OTP, RC4 (byte at a time, faster)
- Encrypting More Data: ECB & CBC

- Steganography: Attempt to hide that secrets are being communicated at all

ELGAMAL Cryptography-Asymmetric Key

1) Key Generation:

- i. Select large Prime No.(P) \rightarrow P=11
- ii. Select Decryption Key/Private Key (D)=3
- iii. Select Second part of encryption key or public key (E1)=2
- iv. Third part of the encryption key or public key (E2). $E2=E1 \bmod P \rightarrow 8$
- v. Public Key = (E1,E2,P), Private Key =D

2) Encryption:

- i. Select random ineteger (R) \rightarrow 4
- ii. $C1=E1^R \bmod P, C1 = 2^4 \bmod 11=5$
- iii. $C2=(PT * E2^R) \bmod P = (7 * 8^4) \bmod 11 = 28672 \bmod 11$
- iv. C.T=(C1,C2)

3) Decryption:

$$P.T=[C2*(C1^D)^{-1}] \bmod P$$



Thank You