CS2107: Intro to InfoSec

Encryption

General

- Encryption scheme (cipher) has two algorithms: **encryption** & **decryption**
- **Correctness of encryption** $\forall x, k, D_k(E_k(x)) = x$

Ciphers

Classic Ciphers

- **Substitution** – one-one onto function
- **Permutation/transposition** – shuffling of characters in a block of the plaintext
- **One time pad** – encrypt n-bit plaintext by XOR-ing with n-bit key (that is used once)

Modern Ciphers

- Takes into consideration common attacks like: known-plaintext attack & frequency analysis

Stream Ciphers

- **Extension** of one time pad
- If key length < plaintext, need to generate key so that XOR can be done
- Use a **cryptographically secure pseudorandom sequence** generated by the secret key
- **Initial Value (IV)** is used to start the generation of the sequence. Must be truly random for it to be secure.
- If IV is same, $c_1 \oplus c_2 = p_1 \oplus p_2$

Transparency of Algorithm

- **Kerckhnoffs's Principle**
  A system should be secure even if everything about the system, except the secret key, is public knowledge.
- **Security through Obscurity**
  To hide the design of the system in order to achieve security.

Authentication

Password

Authentication System Flow

- **Step 1: Bootstrap**
  Server & user establish common password, server records in file
  - **Examples** – server/user chooses password and sent through another communication channel or by using a default password
- **Step 2: Authenticate**
  Server authenticates entity.

Attacks

Bootstrapping attack

- Interception of password during bootstrapping (e.g. steal mail with password)
- Default password may be widely known (e.g. WiFi router)

Searching the Password

- **Guessing** password from social information
- **Dictionary attack**
  Search from a large collection of probable passwords (e.g. words, compromised passwords, etc)

Stealing the Password

- **Sniffing**
  - **Shoulder surfing** – look-over-the-shoulder
  - **Sniffing communications** (e.g. wireless keyboards)
  - **Virus & Keyloggers** – capturing/recording keystrokes/input and send back to attacker (usually via **covert channel**)
- **Phishing**
  Disguise as **trustworthy entity in communications**
  - **Login spoofing**
  - **Spear phishing** – targeted to a particular group of users (social engineering)
- **Cache**
  Browser/application may cache password
- **Insider attack**
  Trusted user steal password (e.g. system administrator stealing user password)

Prevention

- **Strong Password**
  Examples – random, mnemonic, altered passphrases or combining/altering word (e.g. replacing a with @)
- **Password Policy**
  - **Delay** between login attempts
  - Checking/preventing **weak passwords**
  - Password **expiry**
- **Protection of Password Files**
  - **Hash** (with salt) the passwords that are stored
  - Authenticate by comparing hashes

Security Questions

- Example of **fallback authentication**
- Enhances **usability** but weakens security
- **Choice of Questions**
  - **Memorable** – answers can be easily remembered
  - **Consistent** – answer should not change over time
  - **Nearly universal** – question should apply to wide audience
  - **Safe/Private** – not something easily guessed or researched (e.g. public info)

Biometric

- Utilises **unique physical characteristics** of person
- **Enrollment** – **template** of user's biometric data is captured/stored
- **Verification** – biometric data captured and compared with template using a **matching algorithm**
- Inevitable **noise** in biometric data

	Accept	Reject
Genuine	A	C
False	B	D

- **Matching decisions**
  - False Match Rate = $\frac{B}{B+D}$
  - False Non Match Rate = $\frac{C}{B+C}$
  - Matching algo's threshold can be adjusted
- **Equal error rate (EER)** – when FNMR = FMR

- **False-to-enrol rate (FER)** – some users' biometric data cannot be captured (e.g. injury)
- **Failure-to-capture rate (FTC)** – fail to capture biometric data during auth (e.g. skin dry, dirty)

n-factor Authentication

- **Three factors**
  - Something you **know** (password)
  - Something you **have** (e.g. token, card, phone)
  - **Who** you are (biometric)
- **One Time Password tokens** – hardware that generates one time password. Two kinds
  - **Time-based** – based on shared secret and time interval
  - **Sequence-based** – event (button) triggers change in password

Authenticity

General

- **Symmetric-key encryption** uses same key for encryption and decryption (e.g. AES)
- **Public key (symmetric-key)** scheme uses different key for encryption and decryption (e.g. RSA)
- **# keys involved**
  Symmetric – $\frac{n(n-1)}{2}$
  Public key – n public, n private

Comparisons

- Public key may be slower than symmetric
- Security is about protecting the plaintext (from any means of attack)

Hash

- **Hash** – function that takes an arbitrarily sized message and outputs a fixed size **digest**
- **Security requirement** – difficult to find messages m_1, m_2 s.t. $h(m_1) = h(m_2)$ (**collision-resistant**)
- **Keyed-hash (MAC)** – same as hash, but takes in a **secret key**

Data Integrity/Authenticity

- **Unkeyed hash** – if message F' has the same digest as F (which is the actual message), high confidence that F' is same as F

Attacks

Birthday Attack

- **Variables**
  - H – # of possible values
  - n – # items chosen
  - p – proba at least one value chosen more than once
- $p(n, H) \approx 1 - e^{-n^2/(2H)}$
- $n(p, H) \approx \sqrt{2H \ln \frac{1}{1-p}}$
- $n(0.5, H) \approx 1.1774\sqrt{H}$
- **Variant** (set of K distinct elements of n-bits & select M n-bit items)
  $p(n, K) \approx 1 - e^{-KM2^{-n}}$

 # keys needed

- About half of the keyspace in order to get lucky (e.g. 2^{127} for 128-bit keys)

PKI + Channel Security

Public Key Distribution

- Need **secure channel** to distribute (broadcast) public key
- **Three** possible methods
  - Public Announcement – not standardised, have to trust website
  - Publicly available directory – maintainer have to verify information is correct, entity maintaining needs to be trusted
  - Public Key Infrastructure – standardised system, mechanism for "trust" to be extended, from the root

Public Key Infrastructure

Certificate

- Contains at least the following items (can also contain other information)
  - Identity of owner
  - Public key of owner
  - Period of validity
  - Signature of CA
- Certifies public key ownership
- Certificates removes the need to have publicly available directory

CA & Trust Relationship

- CA responsible for verifying information is correct (e.g. domain name)
- Most OS, browsers have few CA's loaded (root CA)
- Someone can send certificate chain, so that verification can start from a CA that is "trusted"

Limitations/Attack on PKI

-

Network Security

Network Layers

Layer	Example
Application	FTP, SMTP, HTTP
Transport	TCP, UDP
Network	IP, Routing
Link	Ethernet, 802.11, PPP
Physical	actual wire

- Conceptually, **peer entities** on same layer communicate through executing same layer protocol
- Layer N protocol built on top of virtual connection at layer $N-1$
- **Protocol Data Unit (PDU)** – message from a layer

Name Resolution

Basics

- Each peer entity has name – each layer can have different names (e.g. domain name, IP, MAC)
- **Name resolution** is the process of finding corresponding name to utilise virtual connection at a lower layer
- **Examples**
  - Address Resolution Protocol (ARP) – resolves MAC given IP address
  - Domain Name System (DNS) – resolves IP addr given domain name

Attacks on DNS

- DNS has **minimal authentication** – query ID (QID) used to

match request to response
- **No encryption/MAC** involved
- **Attack** – through Man-In-The-Middle (MITM), intercept packet and send response before actual response reaches
- **"Single point-of failure"** – attack on availability of DNS server instead of actual web server

Denial of Service (DoS)

General

- DoS is attack on **availability**
- Mostly done by **flooding** victims with overwhelming # of data/requests
- **Example** in application layer (HTTP) – flooding web server with HTTP requests (MyDoom worm targeting SCO's website)

ICMP/Smurf Flood

- Attacker sends `ICMP PING` broadcast request to router with source IP of victim
- Every entity receiving the ping request will send `echo reply` to source, thus flooding it (and the network)
- **Prevented** by disabling broadcast and/or ping feature

Botnet

- Large collection of connected bots communicating via **covert channels**
- Command-and-control mechanisms – individual can control to carry out DDoS or SPAM

Useful Tools

- **Wireshark** – used to analyse packets captured in the **link layer**
- **nmap** – port scanning tool

Protection (Cryptography)

General

- Different security protocols operate at every layer
- Security at layer N will protect layer N and above

Examples

- **SSL/TLS** – transport layer, protects confidentiality and authenticity
- **WiFi Protected Access II (WPA2)** – link and physical layer, not all link layer info is protected
- **IPSec** – network layer, only integrity/authenticity protection of IP address (sniffer able to see IP address)

Protection (Firewall)

General

- Controls what traffic can flow into (ingress filtering) or out (egress filtering) of the network
- **Demilitarized Zone (DMZ)** – sub-network that contains services to be exposed to the (untrusted) Internet

Access Control

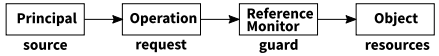
Overview

Access Control Model

General

- Different application domains have different requirements on access control
- Model/System gives a way to **specify and enforce** such restrictions

Principal, Operation, Object



- *Principal/subject* wants to access *object* with some *operation*. **Reference monitor** grants/denies the access.
- **Example** – **Student** (source) wants to **submit** (request) **forum post** (object)
- **Principals vs Subject**
  - **Principals** — human users
  - **Subjects** – entities in system that operate on behalf of principals
- **Classification of accesses to objects**
  - **Observe** – reading file
  - **Alter** – writing, deleting, changing properties
  - **Action** – execution

Ownership

- Two options on deciding access rights to object
  - Owner of object decides rights (discretionary access control)
  - System-wide policy (mandatory access control)

Access Control Matrix

	file1	**file2**
root	{r,w}	{r,x}
Alice	{r,w}	{r,x,o}
Bob	{r,w,o}	{}

- Table is just an **abstract concept**, not explicitly stored
- Matrix represented in two ways
  - **ACL** – stores access rights to an object as list (column)
  - **Capabilities** – subject given list of capabilities, each capability is the access rights to an object (row)
- ACL cannot easily generate list of files user has rights, while the converse is true for capabilities

Intermediate Control

General

- **Group** – subjects in same group have same access rights
- **Role-based access control** – role of subject, role has a collection of procedures
- **Least privilege principle** – access rights not required to complete role will not be assigned

Protection Rings

- Object (data) and subject (process) are assigned numbers
- Lower ring number means **higher privilege**
- Subject cannot access (read/write) object with a **smaller number**
- **Unix** – 2 rings (superuser and user)

Examples

- Both models classify this way:

Level 2 (most secure)
Level 1
Level 0 (least secure)

  where **higher level means higher "security"** (opposite of protection rings)
- **Bell-LaPadula (Confidential)**
  - No read up (prevent access of info in higher security levels)
  - No write down (prevents passing down of confidential info to lower levels)

- **Biba Model (Integrity)**
  - No write up (prevents compromising integrity of higher level)
  - No read down (prevents reading forged information)

UNIX

Scope

- Objects of access control includes files, directories, memory and I/O devices. (Treated as files)

Permissions

- File permissions grouped into **three triples** – read, write and execute access for owner, group & others (world)
- Read (r), Write + delete (w), Execute (x), Execute as owner (s)

Principals & Subjects

- Principals are **user-identities** (UIDs) and **group-identities** (GIDs)
- Subjects are **processes**, that has process ID (PID)

Objects & Access Control

- Objects are **files**
- When user (subject) access file (object), check in this order:
  1. Can use **Owner** permissions?
  2. Can use **Group** permissions?
  3. **Other/World** permissions

Searchpath Issues

- If path not specified, UNIX searches directories specified in **searchpath**
- Prevent attack by **specifying full path**

Controlled Invocation

- Certain resources in UNIX can only be accessed by superuser (e.g. listening at trusted ports), not advisable to change user to superuser
- Set **owner** execute permission to **s**, all executing program will have **effective UID** of owner

Computer Architecture

Background

- von Neumann architecture – code and data stored together in memory

Program Counter

- Register that stores address of next instruction
- PC can be changed in many ways
  - Increment by 1 as instructions get fetched
  - Direct branch – replaced by constant value in memory
  - Indirect branch – replaced by value fetched from memory

Stack

- Stores control flow information and used to pass parameters

Control Flow Integrity

General

- Compromise execution integrity by
  - Modifying code
  - Modifying control flow
- Hard to distinguish malicious code from data

Attacks

- Overwriting execution code with malicious code
- Overwrite control flow information
  - Overwrite actual instruction
  - Overwrite register that is used by code

Software Security

printf

- First argument is the formatted text to be printed
- If value is not specified, printf will use the parameter "stack"
- printf vulnerable if user can supply string that includes malicious formatting
- printf is based on null termination

Data Representation

Strings

- Strings can be stored with or without NULL termination
- Possible exploit – SSL, where certificate is verified based on non-NULL, while comparison is based on NULL termination

UTF-8

- Multi-byte representation – able to represent shorter character with more bytes
- Possible inconsistency between verification and actual use

Buffer Overflow

- Writing beyond buffer's boundary
- E.g. `a[5]` when array size is smaller than 6

strcpy(dest, src)

- Copies whole string from source, even though dest is smaller (based on NULL termination of source)
- Potentially writing over memory or modify computation
- Use **strncpy**

strncpy(dest, src, n)

- Copy at most n items from source to dest
- However, **does not** ensure NULL terminator if there's no space in dest

strlen

- Count **does not** include NULL

Stack Overflow (Stack Smash)

- Buffer overflow that targets **stack**
- If return address modified, control flow of execution will be changed

Stack Overrun (Stack smasking)

- Modification of stack item (e.g. return address)

Integer Overflow

- Integer arithmetic based on modulo arithmetic
- Not enough bits to represent large numbers – wraparound occurs

Script Injection

- Mixing code and data is potentially unsafe

SQL Injection

- Data mixed with SQL query
- Malicious data may be treated as though it is a query command

TOCTOU

General

- Time-of-check-time-of-use (TOCTOU)
- Occurs due to **race condition** between checking and usage of

file/resource

Prevention

- Avoid separate system calls. Instead, open file, lock file and use file handler/descriptor
- Leave access-control checks to OS

Defensive Measures

General

- There's no fool-proof method

Input Validation/Filtering

- Two ways of filtering
  - **Whitelist** – legitimate input may be blocked, no assurance that ALL malicious inputs will be blocked
  - **Blacklist** – malicious input that are not on blacklist may slip through

Use "safe" functions

- For instance, `strncpy` instead of `strcpy`

Bounds Checking & Type Safety

- Check offset being accessed is within the bounds of the data/buffer
- Two types of type safety checks
  - **Dynamic type check** – during runtime
  - **Static type check** – when compiled

Canaries & Memory Protection

- Secret values inserted at selected memory locations
- Check if values are modified during runtime
- Detects **overflow** (especially stack), since consecutive memory locations have to be over-run
- **Address space layout randomization (ASLR)** – prevents attacking based on address stored in same location

Code Inspection

- **Manual checking** – tedious
- **Automated checks** – taint analysis

Testing

- White, black and grey-box testing
- Discover intentional attacks – test for inputs that rarely occurs
- **Fuzzing** – malformed inputs to discover vulnerability

Principle of Least Privilege

- Be conservative in elevating program privilege
- Do not give users more access rights than necessary

Patching

- Lifecycle of vulnerability
  Discovery → Fixing → Testing → Deployment of Patch → Patch applied
- Number of successful attacks goes up after patch announced, since more attackers will be aware of exploit (able to see what is being patched)
- Patches may affect application and thus, organization operations

Web Security

Background

Complications

- Browsers run with same privileges as user
- Supports rich command set and controls for content provider
- Browsers keep user's info and secrets (cookies)
- Many servers (IP addresses) provide content
- Many browsers support add-ons by third parties
- Users can update content in server
- Sensitive information may be in web/cloud

Threat Model

- Attacker as user (end system)
- Attacker as MITM (usually IP layer)
- Difficult to classify different web attacks

SSL/TLS Attacks

Description

- Pre-condition of attack – MITM between browser and server, able to sniff/spoof packets at TCP/IP layer

Examples

- FREAK attack, Superfish, Heartbleed
- Re-negotiation attack, BEAST attack

Misleading Users

URL

- Delimiter used to separate hostname & path can also be characters in path
- Malicious website could use characters that looks similar to delimiter in the hostname
- **Solution** – hostname should be highlighted in URL bar in order to distinguish it

Address Bar Spoofing

- Address bar is the only indicator of the URL to the user
- Address bar could be modified by attacker to trick user to visit malicious URL

Cookies & Same Origin Policy

Cookies

- HTTP cookie is a piece of data sent by server & permanently stored by browser
- Browser sends cookie back to server every time the client revisits its website

Same-Origin Policy

- Script in webpage A can access cookies by webpage B, only if A and B have same origin
- Definition of origin – same protocol, hostname and port number

Why Use Cookies?

- Token-based authentication – browser needs to just present correct token to server, instead of performing password authentication again

Cookie Choices

- Randomly generated number – have to keep track of all tokens
- MAC (two part token) – secure
- Meaningful information concatenated with sequence number – insecure, can easily be forged

Cross-site Scripting (XSS)

- Client can enter data s in browser, which is sent to server and server responds with HTML containing s
- XSS exploits client's trust in

server/website

Cross-site Request Forgery

- Request that are made without client's knowing (e.g. clicking on malicious link), that leads to actions that may have consequences (e.g. withdraw money)
- CSRF exploits server's trust in client
- Defended by including authentication information into request (e.g. one-time token)

Case Studies

FREAK Attack

General

- Man-in-the-Middle attack on the unencrypted cipher suite negotiation
- Due to old regulations limiting RSA moduli to 512-bits (RSA Export Keys)

Method

- Intercept cipher negotiation and change to `EXPORT_RSA`
- When client encrypts pre-master secret, attacker will try to factor the RSA modulus to recover decryption key
- Able to recover any communication between the hosts

Superfish

- Universal self-signed CA installed to allow MITM attack to embed ads
- Same private keys for CA on all laptops
- Eavesdropper able to intercept/modify HTTPS traffic – private key of self-signed certificates in memory

CS2107: Terminologies

General

- **Confidentiality** – property that information is not made available or disclosed to unauthorized individuals, entities or processes
- **Integrity** – maintaining and assuring accuracy & completeness of data over its entire life-cycle. Data cannot be modified in an unauthorized or undetected manner.
- **Availability** – information must be available when it is needed (system's perspective, compromised data **does not** imply compromised availability)
- **Usability** – ease-of-use of system (e.g. 2FA reduces usability)
- **Authenticity** – identity of subject or resource is the identity claimed
- **(Cryptography) Cryptology** – study of techniques for secure communication
- **Cryptanalysis** – study of analysing information systems in order to study hidden aspects of system
- **Common Vulnerabilities and Exposures (CVE)**
  Reference of publicly knwon infosec vulnerabilities and exposures
- **Zero-day (0-day)**
  Vulnerability that is undisclosed

Encryption

- **End-to-end encryption** – only communicating users can read messages
- **Hardware RNG**
  Random numbers from a physical process (e.g. atmospheric noise)
- **Authenticated encryption**
  Form of encryption that provides confidentiality, integrity and authenticity assurances on the data

Authentication

- **Graphical password**
  Select from images, in a specific order, presented in GUI
- **Mutual Authentication**
  Authenticates both parties
- **Phishing**
  Attempt to obtain sensitive information through disguising as trustworthy entity in electronic communication
- **Single sign on**
  Access control of multiple related, but independent software systems
- **Strong Authentication**
  Authentication that is resistant to **replay attacks**
- **Smishing**
  SMS phishing
- **Unilateral Authentication**
  Authenticates a single party
- **Vishing**
  Voice phishing

Cryptography

- **Certificate** – electronic document used to prove the ownership of a **public key**
- **Initial Value/Initialization Vector (IV)** – fixed-size input to ensure randomness
- **Kerckhoffs's principle** – system should be secure if everything about system (except secret) is public knowledge
- **Message Authentication Code (MAC)** – protects message's data integrity & authenticity
- **Non-repudiation**
  Party cannot deny having received a transaction nor can the other party deny having sent a transaction (authorship)
- **Public Key Infrastructure**
  Set of roles, policies and procedures needed to create, manage, distribute, store and revoke digital certificates
- **Public Key** – key that is disseminated widely
- **Private Key** – key that only owner possesses
- **Revocation List**
  List of certificates that are published by CA periodically, stating the certificates that are revoked
- **Root Certificate**
  PKC that identifies as root CA. Self-signed
- **Symmetric Key** – key that is used for both encryption/decryption
- **Key escrow** – keys held in escrow, so that third party may gain access if needed (e.g. employer)
- **Salt** – random data that is used as an additional input to one-way function, defend against dictionary attacks or rainbow tables (see **nonce**)
- **Self-signed Certificate** – identity certificate that is signed by the same entity whose identity it certifies
- **Extended Validation Certificate**
  Certificate that proves the legal entity controlling website
- **Signature** – demonstrates the authenticity of digital messages or documents (authentication, non-repudiation & integrity)
- **Semantic Security**
  Determines if cryptosystem cannot be attacked by any method that has a probabilistic, polynomial-time algorithm (PPTA). That is to say, knowledge of ciphertext (and length) does not reveal additional info that can be feasibly extracted.
- **Trapdoor function**
  A function that is easy to compute in one direction, yet difficult to compute in other direction
- **Web of Trust**
  Establishes authenticity of binding between public key and owner in decentralised trust models (e.g. PGP, GnuPG). Can have many individual web of trusts.

Attacks

- **Birthday attack**
  Attack that exploits **birthday problem** in probability
- **Chosen-plaintext attack**
  Attack model where attacker can obtain the ciphertexts for arbitrary plaintexts. (viz. black box)
- **Covert channel attack**
  Attack that allows transfer of information objects between processes that are not supposed to be allowed to communicate by the computer security policy
- **Denial-of-Service**
  Perpetrator seeks to make a ma-

chine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host

- **Man-in-the-middle**
  Attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other
- **Side-channel attack**
  Attack based on info gained from physical implementation (e.g. timing, power consumption, electromagnetic & sound)
- **Known-plaintext attack**
  Attack model where the attacker has access to both the plaintext & ciphertext (of a specific plaintext)
- **Frequency analysis**
  Study of frequency of letters or group of letters in ciphertext
- **Replay attack**
  Sniffed information from the communicated channel can be used to impersonate user

Access Control

- **Principle of least privilege**
  In an abstraction layer of computing environment, every module must only be able to access only the information and resources that are necessary
- **Mandatory Access Control**
  OS dictates the ability of subject to perform operation on object
- **Discretionary Access Control**
  Subject with certain access permissions is able to pass the permission (indirectly) onto any other subject
- **Role-based Access Control**
  Permissions dictated by the role(s) that the subject possess

Software Security

- **Buffer Overflow**
  Reading/Writing data beyond the boundaries of buffer
- **Integer Overflow**
  Attempting to create numeric value that is outside the range that can be represented with a given number of bits

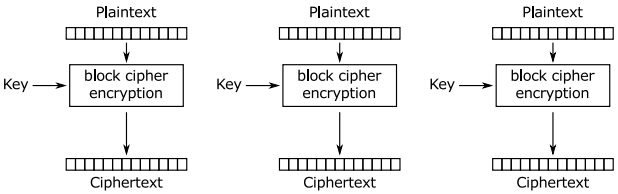
Web

- **Typosquatting**
  Cybersquatting that is dependent on typological errors that are made by humans
- **Cross Site Scripting (XSS)**
  Attacker injects client-side scripts into web pages viewed by others
- **Cross Site Request Forgery (CSRF)**
  Website makes unauthorized commands transmitted from user that web application trusts (exploits trust a site has in user's browser)
- **Drive-by-Download**
  - Authorized download, but without understanding consequence of download
  - Download without user's knowledge
- **Web beacon/bug/tag**
  Object embedded in webpage/email that unobtrusively/invisibly allows checking whether user has accessed content (e.g. analytics)
- **Clickjacking**
  Tricking Web user into clicking on something different from what user think they are clicking (e.g. hidden element)
- **CAPTCHA**
  Type of challenge-response test to determine whether user is human or not
- **Pharming**
  Attack intended to redirect website's traffic to another (e.g. hosts file, DNS exploits)

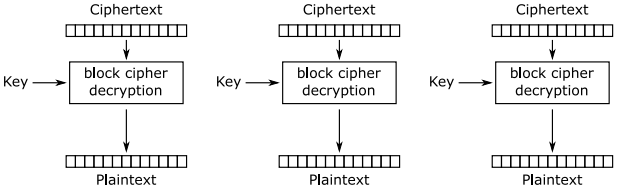
CS2107: Diagrams

Block cipher mode of operation

Electronic Codebook (ECB)

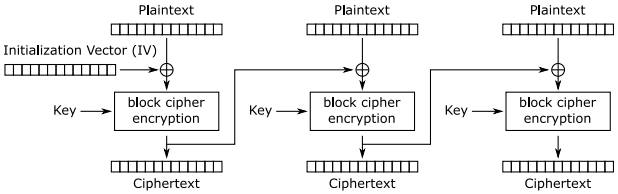


Electronic Codebook (ECB) mode encryption

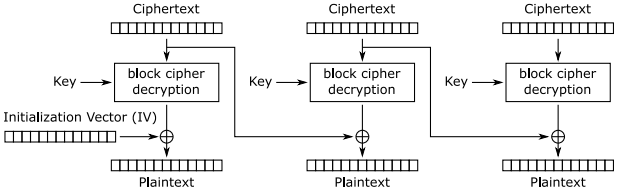


Electronic Codebook (ECB) mode decryption

Cipher Block Chaining (CBC)

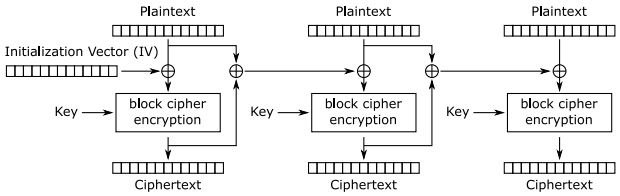


Cipher Block Chaining (CBC) mode encryption

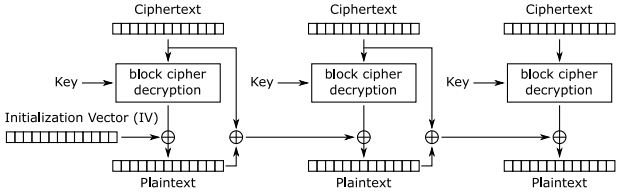


Cipher Block Chaining (CBC) mode decryption

Propagating Cipher Block Chaining (PCBC)

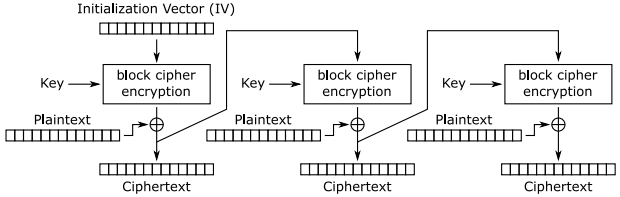


Propagating Cipher Block Chaining (PCBC) mode encryption

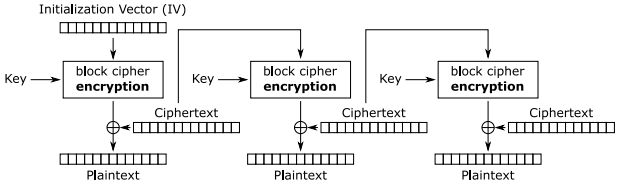


Propagating Cipher Block Chaining (PCBC) mode decryption

Cipher Feedback (CFB)

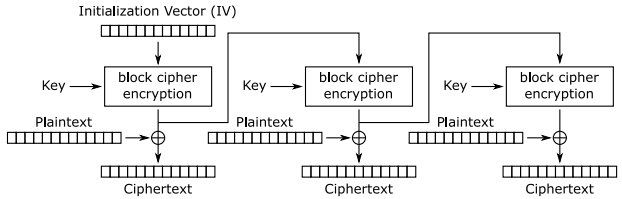


Cipher Feedback (CFB) mode encryption

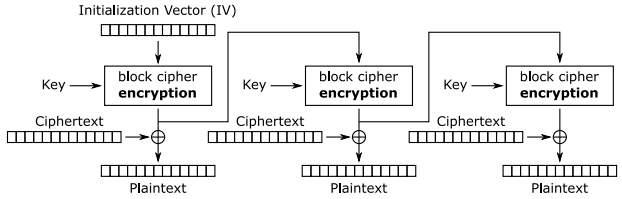


Cipher Feedback (CFB) mode decryption

Output Feedback (OFB)

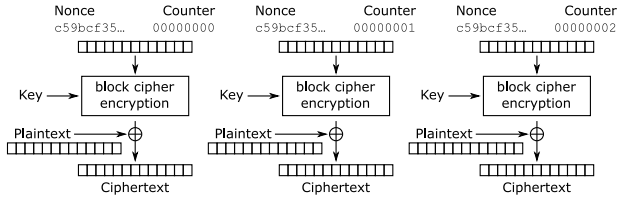


Output Feedback (OFB) mode encryption

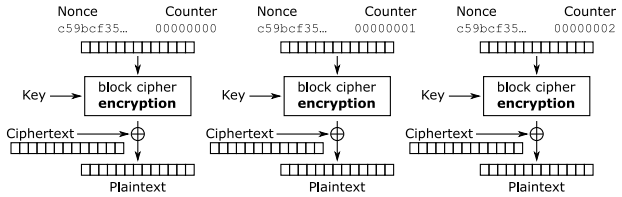


Output Feedback (OFB) mode decryption

Counter (CTR)



Counter (CTR) mode encryption



Counter (CTR) mode decryption

