Includes material from: (checklist)

My Security+ notes (only crypto section specifically),

CASP\_Notes\_Condensed doc- including anything having to do with applications OF crypto and hashes

CEH Topics- Class notes.rtf

CSRF - forged HTTP requests trick a victim into submitting them - use browsers existing credentials to act

XSS - application takes untrusted data and sends it to a web browser without proper validation or escaping (allowing injection)

Topics: (Outline to leave intact)

Intro

- Hashes vs Crypto

- General cryptography concepts: base types (diffusion, confusion, etc)

- Stream vs Block

- Public key vs Shared key/ Asymmetric vs Symmetric

Public Key, Asymmetric Key Pair - how it works

Applied Crypto for covering all the bases

- Sym for key exchange, Private for Comms,

- Hashing for integrity

- Shared key for key exchange

- Public key encryption for ultimate confidentiality, non-repudiation

- Public key signing for authenticity

Types of stream ciphers

- Commonly used

- Others

Types of block ciphers

- Commonly used

- Others

Tweaks (where to put?)

- Key stretching/widening, whitening, nonces, salts, etc.

- PBKDF

- MAC, HMAC

Protocols and applications of Cryptography:

- IPSEC

- OCSP

Important concepts

- State of data: (data at rest, data in transit, data in place (use), etc.

Application: Transport encryption: SSL/TLS, SSH, IPSEC, WPA

Application: Data at rest: Full disk encryption, file-based, etc

- Latency and encryption

- Weakness in mis-application, mis-implementation

LAN manager and Microsoft

WEP and wireless, also

Problems and solutions:

- Perfect forward secrecy (PFS)

- Key management: certificates, escrow,

**Case Study: Listing of Algorithms Selections/ Availability in pfSense (early 2017)**

**Encryption Algorithms**

AES-128-CBC | CFB | CFB1 | CFB8 | OFB (128-bit)

AES-192-CBC | CFB | CFB1 | CFB8 | OFB (192-bit)

AES-256-CBC | CFB | CFB1 | CFB8 | OFB (256-bit)

BF-CBC | CFB | OFB (128-bit)

CAMELLIA-128-CBC | CFB | CFB1 | CFB8 | OFB (128-bit)

CAMELLIA-192-CBC | CFB | CFB1 | CFB8 | OFB (192-bit)

CAMELLIA-256-CBC | CFB | CFB1 | CFB8 | OFB (256-bit)

CAST5-CBC | CFB | OFB (128-bit)

DES-CBC | CFB | CFB1 | CFB8 (64-bit)

DES-EDE-CBC | CFB | OFB (128-bit)

DES-EDE3-CBC | CFB | CFB1 | CFB8 | OFB (192-bit)

DES-OFB (64-bit)

DESX-CBC (192-bit)

IDEA-CBC | CFB | OFB (128-bit)

RC2-40-CBC (40-bit)

RC2-64-CBC (64-bit)

RC2-CBC | CFB | OFB (128-bit)

RC5-CBC | CFB | OFB (128-bit)

SEED-CBC | CFB | OFB (128-bit)

None (No Encryption)

**Auth digest algorithms**

DSA | DSA-SHA | DSA-SHA1 | SHA1-old (160-bit)

MD4 (128-bit)

MD5 (128-bit)

MDC2 (128-bit)

RIPEMD160 (160-bit)

RSA-MD4 | MD5 | MDC2 (128-bit)

RSA-RIPEMD160 | SHA | SHA1 | SHA1-2 (160-bit)

RSA-SHA224 (224-bit)

RSA-SHA256 (256-bit)

RSA-SHA384 (384-bit)

RSA-SHA512 (512-bit)

SHA (160-bit)

SHA1 (160-bit)

SHA224 (224-bit)

SHA256 (256-bit)

SHA384 (384-bit)

SHA512 (512-bit)

ecdsa-with-SHA1 (160-bit)

whirlpool (512-bit)

None (No Authentication)

**Cryptographic Methods - CASP documentation**

**Block Cipher Example: AES Rijndael - pronounced REIN DULL**

Advanced Encryption Standard (AES) is the most popular block cipher and is used throughout organizations and governments worldwide. It is based on the Rijndael algorithm developed by Belgian cryptographers Joan Daemen and Vincent Rijmen and was adopted by the U.S. government as its standard for symmetric encryption. It has no feasible attacks published against it, and it is very efficient in terms of Random Access Memory (RAM) usage and performance. It can operate with key sizes of 128, 192, and 256 bits.

**Cryptographic Design Considerations/ Principles**

Strength Longer keys lead to stronger encryption.

Performance Longer keys add to performance overhead. Asymmetric is slower than symmetric.

Feasibility to implement Older systems may not support modern encryption protocols. A costly upgrade may be required.

Asymmetric may also be difficult and require the use of a third party PKI system.

Interoperability Numerous types of devices means different types of encryption protocols and standards.

Test all devices for their ability to operate with your preferred encryption implementation.

**Transport encryption**

Protects against:

Passive attacks - Example: eavesdropping - VoIP or wardriving a wireless network

Active attacks - Example: man-in-the-middle - replay credentials from previous session - rogue web servers or proxy servers with self-signed SSL certs that trick users into trusting them. All web traffic can then be successfully decrypted by the attacker.

**Transport Encryption Protocols**

SSL/TLS - web-based protocols for combining digital certificates with public key encryption.

SSH - remote login protocol for encrypting sessions.

IPSec - open standard for encrypted transmissions across networks.

WPA/WPA2 - wireless networking encryption.

Data at rest encryption: Full disk, File, and Database encryption

Hash Functions (should have):

Pre-image resistance - can't generate plaintext data based on a certain hash value

***Second pre-image resistance - can't modify plaintext data without also changing its hash value***

Collision resistance - produce two different plaintext input values that have the same resulting hash

See HashCalc program for quick hashing/dehashing of different algorythms (MD5, SH256, etc)

**Key Stretching** - using algorithm to increase key size to increase difficulty to crack

**Digital Sigs - matching hashes**

Digital signatures are particularly valuable in an enterprise environment where authenticity, data integrity, and non-repudiation are vital to the handling of sensitive documents. For example, a financial institution must be absolutely certain that any request to change a customer's account is legitimate. Furthermore, the arbiter of this request must be able to verify that its contents have not been tampered with (for example, requesting that funds be transferred to an attacker's account instead of the intended recipient's). And, for liability purposes, the arbiter must have a guarantee that the sender has no grounds to later deny sending the request.

**Encryption of the Hash**

It is important to remember that a digital signature is a hash that is then itself encrypted. Without the second round of encryption, another party could easily:

Intercept the file and the hash ---> Modify the file ---> Re-create the hash ---> Send the modified file to the recipient.

**Key-Derivation Functions for Key Stretching**

One popular approach to key stretching is to use a key-derivation function:

**Password-Based Key Derivation Function 2 (PBKDF2) is part of the Public Key Cryptography Standards from RSA Laboratories.**

This key derivation function uses five input parameters to create a derived key:

A pseudorandom function such as a hash, cipher, or hash message authentication code (HMAC).

The master password used to generate derived keys.

A cryptographic salt, or random data added to a password or passphrase to counter against certain attacks.

A specified number of iterations for the function to loop.

The desired length of the derived key.

**Bcrypt is a key-derivation function based on the Blowfish cipher.**

Like PBKDF2, it uses a cryptographic salt, but it also adapts over time by increasing the iteration count. There are implementations of bcrypt for Ruby, Python, C, C#, Perl, PHP, Java, and other languages.

**Pseudorandom Number Generation**

A pseudorandom number generator (PRNG) is an algorithm that can produce pseudorandom numbers

These are based on an initial state, called the seed state- a number that defines what will be the first stage of the number generation.

The seed is passed through a mathematical formula to output a pseudorandom number.

Truly random numbers take a long time to gather enough entropy; require integration into software systems; and/or can be cost prohibitive.

A PRNG will always produce the same pseudorandom number sequence when given the same seed state.

So it must be as random as possible and hidden. If a seed for the cryptographic key is compromised, the key can be regenerated.

Best known PRNGs are the Mersenne Twister and Fortuna. The Mersenne Twister is not considered safe for cryptographic purposes, while Fortuna is able to produce pseudorandom number streams suitable for cryptographic operations

**Perfect forward secrecy (PFS) = "NO, you can't go back" - Session keys aren't re-used.**

A characteristic of session encryption that ensures if a key used during a certain session is compromised, it should not affect previously encrypted data. This is a very desirable trait, as long-term keys, such as public/private key pairs, shared keys, and others, are at risk of exposing all data encrypted with them if the key is compromised.

Key compromise should not affect previously encrypted data. Without PFS, decryption of previous HTTPS traffic will compromise web server.

Obtaining PFS requires that long-term keys are only used to derive **ephemeral (per-session)** keys and that the same ephemeral key is never used twice to generate other keys. By never using the same key twice, an attacker who compromises a session key or long-term key can only decrypt one piece of information, such as one packet of a network stream or one conversation.

PFS is a standard function of the SSH and Off-the-Record Messaging (OTR) protocols, and **is optional in the IPSec and TLS protocols**. Despite its significant benefit to security, the vast majority of websites that use TLS do not fully incorporate PFS. There is minor overhead involved in PFS communication, but it is usually not prohibitive enough to justify leaving it out of a TLS-enabled website.

**Heartbleed and the Importance of PFS**

April 2014 OpenSSL vulnerable. An attacker could use this exploit to retrieve plaintext user names, passwords, and even the cryptographic keys themselves.

17% of the Internet's trusted, secure web servers were affected. Although it would not have solved the problem and attackers would still be able to glean sensitive information, PFS would have greatly mitigated its effect. Any keys an attacker could have gleaned with the exploit would only be valid for that particular session. Without PFS, an attacker could use these keys to unlock encrypted communications from the past, putting a great deal more information at risk.

**Guidelines for Choosing Cryptographic Techniques**

Implement transport encryption:

Use SSL/TLS on web servers, SSH when remotely logging in, IPSec for network communications like VPN, WPA2 for wireless.

Implement data at rest encryption:

Encrypt the entire disk when everything on the storage medium needs to be secured; databases that store sensitive information and individual files and folders when only specific data needs to be secured.

Use encryption suites with key stretching algorithms especially if the default key length is not of sufficient length (128 bits).

Use key-derivation functions like PBKDF2 and bcrypt to strengthen key generation in apps that your enterprise develops.

Use a strong hash function like SHA-2, and avoid flawed functions like MD5 to store passwords.

Salt all password hashes to mitigate the threat of rainbow tables and other cracking attempts.

Digitally sign sensitive documents to ensure authenticity and integrity when they are transmitted across your organization.

Sign the code of developed apps to guarantee its legitimacy.

Choose a cryptographic algorithm that uses a "truly random" seed-state PRNG.

Implement secure communication protocols (TLS and SSH) using a suite that fully supports perfect forward secrecy.

Ensure you are running a post-Heartbleed fix of OpenSSL (1.0.1g or higher).

**Rainbow tables:**

Demonstration in Kali used "crunch 3 5 abcd123 >mypasswords" for 3-5 characters of those characters only.

DRM - Controls how digital content can and cannot be used: unauthorized copying of works; share purchased product. Easily bypassed.

Digital Watermarking

Steganography to enforce copyright protection. Validates the media’s authenticity; alerts distributors or users to unauthorized content.

Certain software can remove digital watermarks.

**SSL/TLS**

Public key cryptography used to authenticate, usually with a CA. To encrypt and decrypt information sent from client to server.

Uses MACs to provide data integrity.

The CA is the weak point. If it is compromised, the entire trust is as well. Keep the main CA offline and just use it to update the secondaries.

SSH PGP/GPG - Slides have nothing

PGP is a viable method of ensuring authenticity and integrity in email communications across an enterprise.

PGP generally requires additional end-user plugins and software, and may make it less attractive to a large enterprise that needs to account for significant end-user integration and management.

**S/MIME - Secure/Multipurpose Internet Mail Extensions** is easier to administer/ easier for the users than PGP

Adds digital signatures and public key cryptography to MIME communications. "MIME defines advanced characteristics of emails" **(is this an exam soundbyte?)**

S/MIME differs from PGP: Both parties rely on same CA - More centralized and easily administrated.

**Guidelines for Choosing Cryptographic Implementations**

Implement DRM to protect intellectual property from unauthorized use and sharing.

Keep in mind that DRM can be bypassed.

Use watermarking to embed identity information in a media file.

Keep in mind that watermarking can also be bypassed.

Use SSL/TLS for secure web communications, preferably the latest versions of TLS.

Use SSH instead of a protocol like Telnet to execute remote commands on a host.

Implement PGP and GPG for email encryption, but be aware of the extra software and plugins necessary.

Use S/MIME email encryption to interface with a CA and streamline administration.

**My Notes for Security+**

**Hashes**

Hashes are not reversible.

Ciphers can be reverse engineered but hashes cannot when attempting to re-create a data file.

Hashing is not the same as encryption; hashing is the digital fingerprint, so to speak, of a group of data.

Hashing is used in databases for indexing and file retrieval and is used to protect the confidentiality of data in database tables. It is faster and easier to use than encryption methods.

**MD5 - Message Digest - 1991 - 128-bit**

Weaker than SHA, was ultimately compromised in mid 1990s and considered useless.

Much later, the 2012 malware “Flame” easily fakes a MS digital signature.

MD5 - 128-bit, chance of duplicating same hash after file has been changed is 1 in 2^128 (230 billion billion billion billion)

When CRC is used to fingerprint a file, hash is 32-bit, chance of duplicating same hash after file has been changed is 1 in 2^32 (4,294,967,296)

md5 filename.txt

**SHA - Secure Hash Algorithm**

SHA-0: 1993 - 160-bit hash, unsecure and quickly replaced with SHA-1

SHA-1: Designed by NSA for DSA, in 2010 rendered obsolete

SHA-2: SHA-256 (32 words) SHA-512 (64 words). Also truncated versions -224 and -384.

SHA-3: formerly called “Keccak”; 2012 public competition with non-NSA designers.

Internal structure significantly different from other SHA types, but uses same hash lengths.

Note: CompTIA Cert prep says SHA-2 has 512 bits, SHA-1 160 bit

NTLM (NT LAN Manager) is used by Microsoft as an authentication protocol and a password hash.

The LANMAN hash is a deprecated cryptographic hash function that breaks the password into two parts, the first of which is only seven characters. NTLMv2 (even though it uses MD5) is recommended over LANMAN's hash.

LM hash is used with passwords of 14 or fewer characters.

If you use a password of 15 characters or more on newer versions of Windows, the OS will store a constant string as the LM hash, which is effectively a null password (?), and thereby uncrackable(??2015- this is CompTIA's own language). The real password will be stored as an NTLM2 hash and (in this case calculated with MD5) will be used solely.

***Still Vulnerable!***

Generic Metasploit NTLM Relayer was demonstrated at Defcon 2012

https://www.blackhat.com/html/bh-us-12/bh-us-12-arsenal.html

**Key Exchange**

*The Diffie-Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communication channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.*

*The ElGamal encryption system is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie-Hellman key exchange. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems.*

**Encryption - Symmetrical Secret Key**

[[[RC4 Paragraph was here]]]]

3DES- 1998-9 - Feistel network - symmetric

Data Encryption Standard (DES) cipher key was 56-bit was fine for the time (1977) but later could be brute-forced. Triple-DES applies DES 3 times to each data block

DES uses a key bundle of 3 keys -

ciphertext = Ek1(Dk2(Ek3(plaintext))) - encrypt with key 3, then decrypt with key 2, encrypt with key 1

plaintext = Ek3(Dk2(Ek1(ciphertext))) - reverse (sort of) to decrypt

3 options:

1 - all 3 independent (168-bit - strongest, but only 112-bit effective\*)

2 - 2 of the keys identical (112-bit - next to useless)

3 - 3 keys all identical (56-bit - useless)

\*A meet-in-the-middle (sic) attack on the algorithm, assume that n keys be compromised

Each key bundle is 24, 16, or 8 bytes

NIST considers 3DES option1 to be strong enough till about 2030 due to decrypt calculations.

DES-X (or DESX) is a variant on the DES (Data Encryption Standard) symmetric-key block cipher intended to increase the complexity of a brute force attack using a technique called key whitening.

The original DES algorithm was specified in 1976 with a 56-bit key size: 256 possibilities for the key. There was criticism that an exhaustive search might be within the capabilities of large governments, particularly the United States' National Security Agency (NSA). One scheme to increase the key size of DES without substantially altering the algorithm was DES-X, proposed by Ron Rivest in May 1984.

The algorithm has been included in RSA Security's BSAFE cryptographic library since the late 1980s.

DES-X augments DES by XORing an extra 64 bits of key (K1) to the plaintext before applying DES, and then XORing another 64 bits of key (K2) after the encryption. The key size is thereby increased to 56 + (2 × 64) = 184 bits.

However, the effective key size (security) is only increased to 56+64-1-lb(M) = 119 - lb(M) = ~119 bits, where M is the number of chosen plaintext/ciphertext pairs the adversary can obtain, and lb denotes the binary logarithm. Moreover key size drops to 88 bits given 232.5 known plaintext and using advanced slide attack. (Because of this, some implementations actually make K2 a strong one way function of K1 and K.)

DES-X also increases the strength of DES against differential cryptanalysis and linear cryptanalysis, although the improvement is much smaller than in the case of brute force attacks. It is estimated that differential cryptanalysis would require 261 chosen plaintexts (vs. 247 for DES), while linear cryptanalysis would require 260 known plaintexts (vs. 243 for DES.) Note that with 264 plaintexts (known or chosen being the same in this case), DES (or indeed any other block cipher with a 64 bit block size) is totally broken via the elementary codebook attack.

---------------------

Key whitening

From Wikipedia, the free encyclopedia

In cryptography, key whitening is a technique intended to increase the security of an iterated block cipher. It consists of steps that combine the data with portions of the key.

The most common form of key whitening is xor-encrypt-xor -- using a simple XOR before the first round and after the last round of encryption.

The first block cipher to use a form of key whitening is DES-X, which simply uses two extra 64-bit keys for whitening, beyond the normal 56-bit key of DES. This is intended to increase the complexity of a brute force attack, increasing the effective size of the key without major changes in the algorithm. DES-X's inventor, Ron Rivest, named the technique whitening.

The cipher FEAL (followed by Khufu and Khafre) introduced the practice of key whitening using portions of the same key used in the rest of the cipher. Obviously this offers no additional protection from brute force attacks, but it can make other attacks more difficult. In a Feistel cipher or similar algorithm, key whitening can increase security by concealing the specific inputs to the first and last round functions. In particular, it is not susceptible to a meet-in-the-middle attack. This form of key whitening has been adopted as a feature of many later block ciphers, including AES, MARS, RC6, and Twofish.

----------------------------

Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in a large number of cipher suites and encryption products. Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date. However, the Advanced Encryption Standard (AES) now receives more attention.

Schneier designed Blowfish as a general-purpose algorithm, intended as an alternative to the aging DES and free of the problems and constraints associated with other algorithms. At the time Blowfish was released, many other designs were proprietary, encumbered by patents or were commercial or government secrets. Schneier has stated that, "Blowfish is unpatented, and will remain so in all countries. The algorithm is hereby placed in the public domain, and can be freely used by anyone."[citation needed]

Notable features of the design include key-dependent S-boxes and a highly complex key schedule.

Blowfish has a 64-bit block size and a variable key length from 32 bits up to 448 bits.[2] It is a 16-round Feistel cipher and uses large key-dependent S-boxes. In structure it resembles CAST-128, which uses fixed S-boxes.

----------------------------

AES - Advanced Encryption Standard - 1998 - Standardized 2001-2

Substitution-permutation network Rijndael cipher

Symmetric key algorithm - 128, 192, and 256-bit

First publicly open and accessible cipher approved by NSA for Top Secret use.

RC5 and it's successor RC6 were both AES candidates

IDEA - 1991

Int’l Data Encryption Algorithm. 128-bit block encryption, symmetric key

Meant as a replacement for DES. By 2012 was broken.

**Encryption - Asymmetrical - Public Key Infrastructure**

Public key algorithms, unlike symmetric key algorithms, do not require a secure initial exchange of one (or more) secret keys between the parties.

The X.509 system, which uses a hierarchical approach based on certificate authority mostly replaced the older "web-of-trust". Current versions of PGP encryption include both options through an automated key management server.

Key escrow maintains a secured copy of the user’s private key for the purpose of recovering the key if it is lost.

When creating key pairs, PKI has two methods: centralized and decentralized.

Centralized is when keys are generated at a central server and are transmitted to hosts.

Decentralized is when keys are generated and stored on a local computer system for use by that system.

A RA (registration authority) is used to verify requests for certificates from a certificate authority or multiple certificate authorities.

A CRL is a certificate revocation list; if for some reason a certificate cannot be verified by any parties involved and the issuer of the certificate confirms this, the issuer needs to revoke the certificate. The certificate is placed in the CRL that is published.

-----------------------------------------

Last week, one of the users in your organization encrypted a file with a private key. This week the user left the organization, and unfortunately the systems administrator deleted the user’s account. What are the most probable outcomes of this situation?

----> The data is not recoverable and the data can be decrypted using the recovery agent.

The file can be decrypted with a PKI.

The former user’s account can be re-created to access the file.

The most probable outcome is that the data is not recoverable and the data can be decrypted using the recovery agent. Many systems have a recovery agent that is designed just for this purpose.

If the account that encrypted the file is deleted, it cannot be re-created (without different IDs and therefore no access to the file), and the recovery agent will have to be used. If there is no recovery agent (which in some cases needs to be configured manually), then the file will be unrecoverable.

This file was encrypted with a private key and needs to be decrypted with a private key—PKI is a system that uses asymmetric key pairs (private and public).

The root user account does not have the ability to recover files that were encrypted by other users.

-----------------------------------------

RSA

One of the first PKIs. Algorithms proposed 1973, described again in 1977. MIT granted patent in 1983, wasn’t declassified till 1997; released to public domain in 2000.

Based on 2 prime numbers and an auxiliary value; If used properly they will be large prime numbers that are difficult or impossible to factor.

Anyone can use Bob's public key to send him encrypted messages. In order to verify the origin of a message, RSA can also be used to sign a message.

A message can be signed with a hash from the private key to prove identity:

Alice encrypts a message to Bob using Bob's public key, and signs it with hers.

Bob can check the hash signature with Alice's public key and know it originated from her private key.

ECC - Elliptic Curve Cryptography

Based on the difficulty of solving certain math problems that generate keys by graphing specific points on a curve, is calculated against a finite field. It uses smaller key sizes than most other asymmetrical methods but is as strong

RSA and Diffie-Hellman require more computational power due to the increased key length. DHE especially uses more CPU power because of the ephemeral aspect. (ECDHE would be the solution in that respect.)

PGP

PGP encryption uses a serial combination of hashing, data compression, symmetric-key cryptography, and finally public-key cryptography; each step uses one of several supported algorithms. Versions that support newer features and algorithms are able to create encrypted messages that older PGP systems cannot decrypt, even with a valid private key. Therefore it is essential that partners in PGP communication understand each other's capabilities or at least agree on PGP settings

PGP combines symmetric-key encryption and public-key encryption. The message is encrypted using a symmetric encryption algorithm, which requires a symmetric key. Each symmetric key is used only once and is also called a session key. The message and its session key are sent to the receiver. The session key must be sent to the receiver so they know how to decrypt the message, but to protect it during transmission, it is encrypted with the receiver's public key. Only the private key belonging to the receiver can decrypt the session key.

Cryptoanalysis is unrealistic for cracking up-to-date PGP implementations, leaving vulnerability to “rubber-hose” (torture or coercion) or “black-bag” (burglary or bugs, keystroke loggers or trojan horse) methods.

---------------------

PBKDF2 (Password-Based Key Derivation Function 2) is a key derivation function that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It replaces an earlier standard, PBKDF1, which could only produce derived keys up to 160 bits long.

PBKDF2 applies a pseudorandom function, such as a cryptographic hash, cipher, or HMAC to the input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations. The added computational work makes password cracking much more difficult, and is known as key stretching. When the standard was written in 2000, the recommended minimum number of iterations was 1000, but the parameter is intended to be increased over time as CPU speeds increase. Having a salt added to the password reduces the ability to use precomputed hashes (rainbow tables) for attacks, and means that multiple passwords have to be tested individually, not all at once. The standard recommends a salt length of at least 64 bits.

TOTP, HOTP - Time-based and hash based one-time passwords. Hash-based last longer.

SPA (Secure Password Authentication) is a Microsoft protocol used to authenticate e-mail clients.

S/MIME and PGP can be used to secure the actual e-mail transmissions.

S/MIME (Secure/Multipurpose Internet Mail Extensions) enables users to send both encrypted and digitally signed e-mail messages, enabling a higher level of e-mail security.

**CASP Material**

**Ch 5 - Implementing Cryptographic Techniques**

Confidentiality, Integrity, Non-repudiation, Entropy

Confusion - Caesar-style cipher shown, Diffusion - graphic mostly nonsensical

Steganography - another useless slide - OPENStego, BPMSecrets

Chain of Trust (shows CA machines, root to subordinate to end workstation)

In the real world, the root CA gets turned off to let the subordinate do the work, to protect the root CA)

Root of Trust

Guarantees integrity of a hardware environment supporting:

Full disk encryption, DRM, Detecting and preventing unauthorized software changes

Can be the starting point in a chain of trust.

Advanced PKI Concepts

Wild card- Special character in a certificate used to cover subdomains in a domain. \*.google.com

"there are security concerns associated with using wild cards to secure your domains.

Because the same private key is stored on multiple systems, there is a greater risk that it can be accessed"

**CRL-** A list of certificates that were **revoked before the set expiration date.**

**OCSP- Online Certificate Status Protocol**

An HTTP-based alt to a CRL that checks the status of certificates - for internet-use to obtain revocation status of X.509 certs

"OCSP servers, also called responders, accept a request to check a specific certificate's status. The responder uses the certificate's serial number to search for it in the CA's database. The server then sends the certificate's status to the requester. OCSP benefit over a CRL is that it lowers overhead on the client side. OCSP responses for specific certificate requests contain less data than entire revocation lists. However, because OCSP does not by default encrypt these standard HTTP transmissions, an attacker may be able to glean that a network resource used a specific certificate at a specific time during this OCSP transaction

**Key escrow-** A method for backing up private keys to protect them while allowing trusted third parties (key escrow agent) to access the keys under certain conditions.

Issuance to entities- Automating certificate requests, using wild card certificates, certificate lifespan.

Enterprise have its own PKI and CA? Use public certificates? Both? Group Policy automates cert requests.

**Issuance to entity subgroups**

Users: Shortest expiration time, submission and issuance can be automatically managed.

Systems: Longer certificate expiration, easier to centrally manage (5-30 years!).

Applications: One to three year expiration, any secure communication with external entities may necessitate public certificates (SSL/TLS, email)

"Applications certificates are usually managed centrally. If the applications will only be used by internal entities, an internal CA can manage the application certificates. If the server requires secure communication with any external entities, then the enterprise would need to purchase public certificates."

**Cryptographic Applications**

Proper implementations:

Industry standard scheme like AES or RSA.

Strong key length (128-bit or more for symmetric encryption, 2048 or more for asymmetric).

Store keys in a key management system.

Regulate access to this system.

Employ perfect forward secrecy in public key cryptography.

Encryption should cover all relevant environments in your enterprise.

Weigh the benefits with increased cost and overhead.

Improper implementations:

Obsolete scheme like DES or rolling your own.

Weak key length (56-bit or less for private encryption, 1024 or less for public WEAK).

Storing data in insecure physical or virtual space.

Failing to account for increased cost and overhead. Failure to realize you need to budget a dedicated CA machine

Not covering all relevant environments in the enterprise. "You missed a spot"

**Block Cipher Example: AES Rijndael - pronounced REIN DULL**

Advanced Encryption Standard (AES) is the most popular block cipher and is used throughout organizations and governments worldwide. It is based on the Rijndael algorithm developed by Belgian cryptographers Joan Daemen and Vincent Rijmen and was adopted by the U.S. government as its standard for symmetric encryption. It has no feasible attacks published against it, and it is very efficient in terms of Random Access Memory (RAM) usage and performance. It can operate with key sizes of 128, 192, and 256 bits.

**Cryptographic Design Considerations/ Principles**

Strength Longer keys lead to stronger encryption.

Performance Longer keys add to performance overhead. Asymmetric is slower than symmetric.

Feasibility to implement Older systems may not support modern encryption protocols. A costly upgrade may be required.

Asymmetric may also be difficult and require the use of a third party PKI system.

Interoperability Numerous types of devices means different types of encryption protocols and standards.

Test all devices for their ability to operate with your preferred encryption implementation.

it is shown that Blowfish and AES have the best performance among others. And both of them are known to have better encryption (i.e. stronger against data attacks) than the other two.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **MB (2^20 bytes) Processed** | **Time Taken** | **MB/Second** |
| **Blowfish** | 256 | 3.976 | 64.386 |
| **Rijndael (128-bit key)** | 256 | 4.196 | 61.010 |
| **Rijndael (192-bit key)** | 256 | 4.817 | 53.145 |
| **Rijndael (256-bit key)** | 256 | 5.308 | 48.229 |
| **Rijndael (128) CTR** | 256 | 4.436 | 57.710 |
| **Rijndael (128) OFB** | 256 | 4.837 | 52.925 |
| **Rijndael (128) CFB** | 256 | 5.378 | 47.601 |
| **Rijndael (128) CBC** | 256 | 4.617 | 55.447 |
| **DES** | 128 | 5.998 | 21.340 |
| **(3DES)DES-XEX3** | 128 | 6.159 | 20.783 |
| **(3DES)DES-EDE3** | 64 | 6.499 | 9.848 |

**Transport encryption**

Protects against:

Passive attacks - Example: eavesdropping - VoIP or wardriving a wireless network

Active attacks - Example: man-in-the-middle - replay credentials from previous session - rogue web servers or proxy servers with self-signed SSL certs that trick users into trusting them. All web traffic can then be successfully decrypted by the attacker.

**Transport Encryption Protocols**

SSL/TLS - web-based protocols for combining digital certificates with public key encryption.

SSH - remote login protocol for encrypting sessions.

IPSec - open standard for encrypted transmissions across networks.

WPA/WPA2 - wireless networking encryption.

Data at rest encryption: Full disk, File, and Database encryption

Hash Functions (should have):

Pre-image resistance - can't generate plaintext data based on a certain hash value

***Second pre-image resistance - can't modify plaintext data without also changing its hash value***

Collision resistance - produce two different plaintext input values that have the same resulting hash

See HashCalc program for quick hashing/dehashing of different algorythms (MD5, SH256, etc)

**Key Stretching** - using algorithm to increase key size to increase difficulty to crack

**Digital Sigs - matching hashes**

Digital signatures are particularly valuable in an enterprise environment where authenticity, data integrity, and non-repudiation are vital to the handling of sensitive documents. For example, a financial institution must be absolutely certain that any request to change a customer's account is legitimate. Furthermore, the arbiter of this request must be able to verify that its contents have not been tampered with (for example, requesting that funds be transferred to an attacker's account instead of the intended recipient's). And, for liability purposes, the arbiter must have a guarantee that the sender has no grounds to later deny sending the request.

**Encryption of the Hash**

It is important to remember that a digital signature is a hash that is then itself encrypted. Without the second round of encryption, another party could easily:

Intercept the file and the hash ---> Modify the file ---> Re-create the hash ---> Send the modified file to the recipient.

**Key-Derivation Functions for Key Stretching**

One popular approach to key stretching is to use a key-derivation function:

**Password-Based Key Derivation Function 2 (PBKDF2) is part of the Public Key Cryptography Standards from RSA Laboratories.**

This key derivation function uses five input parameters to create a derived key:

A pseudorandom function such as a hash, cipher, or hash message authentication code (HMAC).

The master password used to generate derived keys.

A cryptographic salt, or random data added to a password or passphrase to counter against certain attacks.

A specified number of iterations for the function to loop.

The desired length of the derived key.

**Bcrypt is a key-derivation function based on the Blowfish cipher.**

Like PBKDF2, it uses a cryptographic salt, but it also adapts over time by increasing the iteration count. There are implementations of bcrypt for Ruby, Python, C, C#, Perl, PHP, Java, and other languages.

**Pseudorandom Number Generation**

A pseudorandom number generator (PRNG) is an algorithm that can produce pseudorandom numbers

These are based on an initial state, called the seed state- a number that defines what will be the first stage of the number generation.

The seed is passed through a mathematical formula to output a pseudorandom number.

Truly random numbers take a long time to gather enough entropy; require integration into software systems; and/or can be cost prohibitive.

A PRNG will always produce the same pseudorandom number sequence when given the same seed state.

So it must be as random as possible and hidden. If a seed for the cryptographic key is compromised, the key can be regenerated.

Best known PRNGs are the Mersenne Twister and Fortuna. The Mersenne Twister is not considered safe for cryptographic purposes, while Fortuna is able to produce pseudorandom number streams suitable for cryptographic operations

**Perfect forward secrecy (PFS) = "NO, you can't go back" - Session keys aren't re-used.**

A characteristic of session encryption that ensures if a key used during a certain session is compromised, it should not affect previously encrypted data. This is a very desirable trait, as long-term keys, such as public/private key pairs, shared keys, and others, are at risk of exposing all data encrypted with them if the key is compromised.

Key compromise should not affect previously encrypted data. Without PFS, decryption of previous HTTPS traffic will compromise web server.

Obtaining PFS requires that long-term keys are only used to derive **ephemeral (per-session)** keys and that the same ephemeral key is never used twice to generate other keys. By never using the same key twice, an attacker who compromises a session key or long-term key can only decrypt one piece of information, such as one packet of a network stream or one conversation.

PFS is a standard function of the SSH and Off-the-Record Messaging (OTR) protocols, and **is optional in the IPSec and TLS protocols**. Despite its significant benefit to security, the vast majority of websites that use TLS do not fully incorporate PFS. There is minor overhead involved in PFS communication, but it is usually not prohibitive enough to justify leaving it out of a TLS-enabled website.

**Heartbleed and the Importance of PFS**

April 2014 OpenSSL vulnerable. An attacker could use this exploit to retrieve plaintext user names, passwords, and even the cryptographic keys themselves.

17% of the Internet's trusted, secure web servers were affected. Although it would not have solved the problem and attackers would still be able to glean sensitive information, PFS would have greatly mitigated its effect. Any keys an attacker could have gleaned with the exploit would only be valid for that particular session. Without PFS, an attacker could use these keys to unlock encrypted communications from the past, putting a great deal more information at risk.

**Guidelines for Choosing Cryptographic Techniques**

Implement transport encryption:

Use SSL/TLS on web servers, SSH when remotely logging in, IPSec for network communications like VPN, WPA2 for wireless.

Implement data at rest encryption:

Encrypt the entire disk when everything on the storage medium needs to be secured; databases that store sensitive information and individual files and folders when only specific data needs to be secured.

Use encryption suites with key stretching algorithms especially if the default key length is not of sufficient length (128 bits).

Use key-derivation functions like PBKDF2 and bcrypt to strengthen key generation in apps that your enterprise develops.

Use a strong hash function like SHA-2, and avoid flawed functions like MD5 to store passwords.

Salt all password hashes to mitigate the threat of rainbow tables and other cracking attempts.

Digitally sign sensitive documents to ensure authenticity and integrity when they are transmitted across your organization.

Sign the code of developed apps to guarantee its legitimacy.

Choose a cryptographic algorithm that uses a "truly random" seed-state PRNG.

Implement secure communication protocols (TLS and SSH) using a suite that fully supports perfect forward secrecy.

Ensure you are running a post-Heartbleed fix of OpenSSL (1.0.1g or higher).

**Rainbow tables:**

Demonstration in Kali used "crunch 3 5 abcd123 >mypasswords" for 3-5 characters of those characters only.

DRM - Controls how digital content can and cannot be used: unauthorized copying of works; share purchased product. Easily bypassed.

Digital Watermarking

Steganography to enforce copyright protection. Validates the media’s authenticity; alerts distributors or users to unauthorized content.

Certain software can remove digital watermarks.

**SSL/TLS**

Public key cryptography used to authenticate, usually with a CA. To encrypt and decrypt information sent from client to server.

Uses MACs to provide data integrity.

The CA is the weak point. If it is compromised, the entire trust is as well. Keep the main CA offline and just use it to update the secondaries.

SSH PGP/GPG - Slides have nothing

PGP is a viable method of ensuring authenticity and integrity in email communications across an enterprise.

PGP generally requires additional end-user plugins and software, and may make it less attractive to a large enterprise that needs to account for significant end-user integration and management.

**S/MIME - Secure/Multipurpose Internet Mail Extensions** is easier to administer/ easier for the users than PGP

Adds digital signatures and public key cryptography to MIME communications. "MIME defines advanced characteristics of emails" **(is this an exam soundbyte?)**

S/MIME differs from PGP: Both parties rely on same CA - More centralized and easily administrated.

**Guidelines for Choosing Cryptographic Implementations**

Implement DRM to protect intellectual property from unauthorized use and sharing.

Keep in mind that DRM can be bypassed.

Use watermarking to embed identity information in a media file.

Keep in mind that watermarking can also be bypassed.

Use SSL/TLS for secure web communications, preferably the latest versions of TLS.

Use SSH instead of a protocol like Telnet to execute remote commands on a host.

Implement PGP and GPG for email encryption, but be aware of the extra software and plugins necessary.

Use S/MIME email encryption to interface with a CA and streamline administration.

**HSM - Hardware Security Module**

USB, hardware security solution, prevents the execution of programs

Verify the integrity of data on a computer through message authentication codes (MACs). For example, when an authorized user makes changes to an employee's salary in a database full of employee financial information, the HSM creates a MAC of that valid input.

If a user or program asks the HSM to scan a salary value and its MAC is incorrect, the HSM will identify that the database has been tampered with. The HSM itself secures the key that it uses to construct the MAC, so it should not be possible to reproduce that valid MAC. This makes it important for the HSM to conduct secure key management to both create and maintain cryptographic keys.

HSMs typically have a standardized user interface for easy programming and customization.

**Note: some of this refers to HSM appliances - Network-Attached HSMs mentioned later - There isn't a line drawn here to differenciate the two.**

HSMs can be used as a secure key generator and storage container for a certificate authority (CA) in a public key infrastructure.

HSMs can do key generation for SSL/TLS, support scalability in an expanding enterprise infrastructure through clustering and load balancing, can be prohibitively expensive for smaller organizations.

**TPM**

**Trusted Platform Module (TPM)** is a hardware-based encryption specification that allows secure cryptoprocessors to generate cryptographic keys to be used to authenticate hardware, encrypt disks, enforce digital rights management (DRM), or for any other encryption-enabled application. TPM can also be used as a BIOS security method and by full disk encryption apps such as BitLocker® to secure the operating system's volume. Several major PC manufacturers provide TPM-integrated microprocessors with their products.

**Virtual Trusted Platform Module - vTPM**

Extension of TPM to software running in VMs. Host hardware needs a real TPM

If VM migrates, so too should the vTPM instance, retain confidentiality and integrity of data.

Software might not be able to differentiate between a physical and virtual TPM.

**Secure Boot**

Component of UEFI that stops "unwanted processes" from executing at boot.

Secure boot **checks digital signatures against a database before boot**; if it can’t validate process’s signature, it will not execute.

Can limit what you add to boot loaders.

**May need to update signature database** or turn secure boot off in BIOS.

**Cloud Provider Hash Matching**

Hash matching runs a cryptographic hash function on data that the client sends to the service. The resulting hash of each piece of data is compared with known hash values to detect any matches:

Cloud antivirus and anti-spam solutions can use the enterprise's Internet connection to remotely scan data on a host and look for hash matches. The hashed data is compared to known virus or email content values in a signature-based analysis. If matches are detected, the cloud software can alert an administrator or take steps to remove the virus or block the spam.

Beyond basic antivirus and anti-spam are more in-depth vulnerability scanning services to check host operating systems, firmware, apps, and more for known weak points. These weak points are usually matched to a vulnerability database like NVD, and to predefined industry standards and best practices.

The enterprise can enlist a cloud provider to take on the overhead involved in running this software for each host:

- cloud antivirus and anti-spam is always current and doesn't require you to constantly download signature database updates.

- a more complete security analysis of their environments without needing to configure or assign resources to a local solution.

- These cloud solutions have the potential to expose vulnerability- transport encryption is particularly crucial to keep communications with the cloud confidential.

**Inline Network Encryptor**

The INE device is situated just outside the secure network, and any data sent by the secured network through the unsecured network must pass through the device first. The INE encrypts the data, then passes it to a perimeter router, which then passes it to the unsecured network. The unsecured network then sends the data to the destination network's perimeter router, where it passes the data on to a receiving INE device. This device decrypts the data before it passes it on to the destination network

- While similar in function, virtual private networks (VPNs) tend to connect individuals to networks whereas INEs connect whole networks to other networks

- Packets over IP communications often use the Internet Protocol Security (IPSec) protocol to encapsulate data in a way that is invisible to the end user. The National Security Agency (NSA), for example, develops and uses **High Assurance Internet Protocol Encryptor (HAIPE)** devices based on an enhanced version of IPSec that uses strong cryptographic algorithms and can quickly be de-provisioned to destroy cryptographic keys.

- Latency an issue - some compensate but are more expensive

**Network-Attached Hardware Security Module (HSM)**

Can provide encryption for cloud computing and virtualization deployments.

Centralizes how keys for many connected hosts are managed. Used as a root of trust.

Often partition themselves to provide isolated encryption containers for hosts to work with specific ports or implementations

Related things to remember

Investigate how much of a bottleneck an INE will be on your network before implementing it.

Determine if multiple hosts would benefit from a centralized network-attached HSM, rather than multiple host-based solutions.

Establish secure protocols like SSH and SSL/TLS for remote access.

Enable transport encryption on remote applications like RDP and VNC.

Implement strong authentication methods like multifactor and mutual in your network's design.

Implement 802.1X, especially in wireless networks, to force users to authenticate before they can even connect to your network.

Construct your network around a mesh topology if you can meet the financial and administrative costs.

Secure connections between controller and forwarding device in SDN architectures.

Avoid using cloud network management solutions that put the controller in the cloud.

**Data Flow Enforcement**

Attackers may still be able to take advantage of transport encryption to disrupt the flow of data. This is why you must implement more complex network security solutions for data flow. Sensors and monitors, for instance, can capture the size and frequency of all packets transmitted over the network, giving you the network flow data you need to analyze the traffic in your enterprise. Additionally, network monitors can record all transmission on a host-by-host level. A forensic investigation will benefit from this analysis, as you can accurately pinpoint the source and destination of relevant traffic over a given time period. Likewise, network flow data will help you craft a network baseline and detect anomalous traffic usage that deviates from this baseline.

Flow data is useful for seeing your network from a high level, but what if you need to enforce a security policy in individual transmissions? For this, you'd need a solution that scans the actual content of encrypted data, not just its metadata. One such solution is **SSL inspection**, a technique of monitoring the contents of HTTPS traffic to detect threats in the encrypted payload. SSL inspection technologies decrypt a transmission before it reaches its destination, inspect its data for malware or anything else that does not conform to pre-set policy, and then re-encrypts the data and passes it along to its destination. SSL inspection can be used to trigger alerts based on certain keywords, reject or allow certain packets over others, or other related forms of content filtering.

SSL inspection can be considered a white hat (ethical) version of a man-in-the-middle attack.

Microsoft Forefront offers data flow enforcement in enterprise networks.

**Network Device Configuration**

**Transport security (TLS)**

Cryptographic process for data transmitted over the network.

Ensures communication integrity and confidentiality.

Alternative to IPSec for VPNs.

Comparable method uses SSH to manage access to network devices.

**Storage Encryption**

Disk, Block, File, Record (DBs), Port (443, ssh tunnels, forwarding)

Consider which storage encryption method is the best fit for your needs, given your performance and confidentiality requirements.

**Network Authentication** - Kerberos, Radius, 802.1x EAP

802.1x

Design network to require authentication for users and devices that want to attach. WPA Enterprise for wireless environments.

Encapsulates frames in EAP to provide to a RADIUS server. User needs to authenticate, and 802.1X carries this request to the service.

If the service verifies authentication, traffic is enabled for the user.

Without authentication, user is only allowed to communicate over the 802.1x protocol, effectively locking them out.

Mesh networks can reliably transfer information from node to node. If one node malfunctions, the rest of the nodes in the network can fall back on a secondary routing path that bypasses the affected node. This keeps the malfunctioning node from interrupting data transfer, as the rest of the network will still be able to transmit the data to every other healthy node using a different route. Likewise, mesh networks can ensure that each node is carrying its own weight, which allows the nodes to keep the network available in high-traffic situations. The nodes in the mesh also have dedicated lines of communication instead of needing to constantly switch, so they can achieve a measure of security as information will more reliably reach its intended destination and no other.

Despite these advantages, mesh networks can be very costly and difficult to administer, especially in wired implementations. Even wireless mesh networks are difficult to construct, and often require that each node install accompanying software. Your ability to build and maintain your network in terms of cost and labor should influence whether or not you decide to go with a mesh topology.

Examples

Real-world examples of mesh networks include: Devices using the ZigBee specification, City-wide wireless networks (WMANs??), laptops created for the One Laptop per Child (OLPC) initiative, U.S. military in field operations.

**Out-of-Band Communication**

Communication outside of normal channels.

Can configure out-of-band NICs, ACLs, management and data interfaces.

Provide the following functionality not used in-band:

Reset host if main communications fail/ Rebooting a host that is shut down.

Reinstall a host’s OS / Mounting physical media / Accessing host’s BIOS.

Monitoring hardware components.

Isolating these functions may guard hosts if attackers compromise in-band communications.

**Peripheral Restrictions**

Certain hosts need peripherals; others could do without this additional attack surface.

Bluetooth - bluejacking (unsolicited messages or transfers to you) bluesnarfing (unauth access into info)

Firewire: some implementations can map physical memory spaces in hardware, an attacker might retrieve crypto keys stored there.

Full Disk Encryption

**Guidelines for Selecting Host Hardware and Software**

Implement an HSM on hosts that need strong assurances of integrity, like databases.

Use an HSM to generate and store keys for a CA in a public key infrastructure.

Use an HSM to take on SSL/TLS key generation overhead in order to accelerate the process.

**---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Ch 4 - Integrating Advanced Authentication and Authorization Techniques**

Authentication - Name and password - Something you have, know, are, etc. Authorization- access to. Access based on authentication

Certificate-Based Authentication - Client validates to CA, which issues to certificate holder (CH), which presents to client to use

SSO - single sign-on - Users authenticate once to several related but independent systems.

Good: Credentials quickly regained. No continuously logging in with different credentials.

Bad: Compromise of single credentials can lead to the compromise of many systems.

If one authentication server is unavailable, many systems may be affected.

OAuth

Minimize exposure of credentials by issuing tokens.

Empowers an app to act on its behalf.

Accesses secure APIs without sharing passwords.

Token is a short string combined with a secret string.

The OAuth Process

1. App **request** for a service and user provides credentials at endpoint.

2. Credentials **authorizorization** sent to service provider.

3. Another request made to **upgrade** the authorized token to an access token so user can access resources without credentials.

======================

OAuth is an open authorization framework that enables users to empower an app to act on their behalf, accessing secure APIs (and the resources accessed through those APIs) without sharing their password. OAuth uses a token that is essentially a short string combined with a corresponding secret string.

Traditional web-based applications typically require that users provide a user name and password to authenticate. ***There are, however, situations when it is not a good idea to use a user name and password for authentication in apps that your enterprise develops. For example, when an app seeks a web or cloud API service, it is not safe to permit the app to store the web service user name and password, because this extends the attack surface to include both the app and the web APIs. Also, passing credentials across the network each time the app makes an API call or establishes a session will also increase risk***.

OAuth's token-based authentication helps mitigate this risk.

Many major websites and cloud services, like Facebook, Twitter, and Amazon, have adopted OAuth and provide developer documentation on accessing their services through OAuth. In an enterprise setting, this open authorization framework will help limit the exposure of your app users' credentials.

**OAuth 2.0 is the most current version, and is not backward compatible with version 1.0. However, OAuth 2.0 may be less secure than version 1.0 because it leaves many factors open to implementation instead of using narrowly defined security protocols.**

**The OAuth Process**

An OAuth token goes through three general steps during the initialization process, each of which has an associated URL (called an endpoint) at the OAuth provider's website, with which the user and app interact to obtain the token. The steps include:

1. Request: A client app contacts an OAuth cloud-based provider service to request access to a particular service (called the "scope"). The service has your app redirect the user to the request endpoint, where the user is prompted to provide user credentials.

2. Authorization: The user enters login credentials for the service provider website, which will be granted to the client app. The credentials are sent directly to the service provider. If the user has entered the correct credentials, the server replies with a response that authorizes the token and gives the app an authorized request token.

3. Upgrade: With the authorized token, the client app issues another request to the service provider. The service provider upgrades the authorized token to an access token, which it returns to the client app. The client app may now use the access token to gain access to the user's resources on the service, within the permitted scope, until the token is revoked by the user.

An HTTP request is sent to the appropriate endpoint to obtain a token or manipulate it. As an example, Google's OAuth endpoints are:

• Request endpoint: www.google.com/accounts/OAuthGetRequestToken

• Authorization endpoint: www.google.com/accounts/OAuthAuthorizeToken

• Upgrade endpoint: www.google.com/accounts/OAuthGetAccessToken

Once acquired, the access token can be used until revoked by the user, which might be a very long time.

======================

**XACML - eXtensible Access Control Markup Language**

A declarative access control policy language implemented in XML

A processing model describing how to evaluate access requests according to the rules defined in policies.

Three-level hierarchy:

Rules- rule-based access:

Policies - can contain any number of rule elements

PolicySet - can contain any number of policy elements and policy set elements

Policies, policy sets, rules and requests all use these elements:.

Subject - the entity requesting access

Resource - a data, service or system component

Action - defines the type of access requested on the resource

(An environment element can optionally provide additional information)

Extensible, can easily integrate with new and changing policies.

Reduces inconsistencies in policy implementation.

XACML describes policy and request/response languages:

**The policy language** describes general access control requirements

**The request/response language** allows users to create queries about whether a request should be permitted.

**XACML is comprised of SAML (a policy language) to handle authentication (SSO) and SPML (request/response language) to provide resources and services.**

**SPML uses a Requesting Authority (creates service provisioning requests) to send requests to PSP (Provisioning Service Point), which processes the request and creates or modifies a user account in the target system.**

*On research outside the given material for the test it is difficult to determine from the information given where exactly SAML and SPML begin and end. RA and PSP are not further defined, or seem to precisely fit into the explanation of xACML below. Deductions can be made with the idea that "the client is decoupled from the access decision" - yet notice that all "points" below begin with "Policy". Where authorization and authentication begin and end is hard to tell, thus where to draw the line on SAML and SPML. I doubt that on the exam questions will pry into details beyond the RA and PSP as presented before. Regardless, this came from Wikipedia and Oasis:*

Attribute Based Access Control system (ABAC), where attributes (bits of data) associated with a user or action or resource are inputs into the decision of whether a given user may access a given resource in a particular way. Role-based access control (RBAC) can also be implemented in XACML as a specialization of ABAC. xACML supports and encourages the separation of the access decision from the point of use. When access decisions are baked into client applications (or based on local machine userids and Access Control Lists (ACLs)), it is very difficult to update the decision criteria when the governing policy changes. When the client is decoupled from the access decision, authorization policies can be updated on the fly and affect all clients immediately.

When users want to access resources, they will request the Policy Enforcement Point (PEP), which is an entity that protects the resources.

The PEP in turn will create a request based on the availability of the resource and the action required, and send it to a Policy Decision Point (PDP).

The PDP will analyze the request, compare it with some policies, and provide a solution regarding whether access should be granted.

The response is sent to the PEP, which allows or denies access to the request.

PAP Policy Administration Point - Point which manages access authorization policies

PDP Policy Decision Point - Point which evaluates access requests against authorization policies before issuing access decisions

PEP Policy Enforcement Point - Point which intercepts user's access request to a resource, makes a decision request to the PDP to obtain the access decision (i.e. access to the resource is approved or rejected), and acts on the received decision

PIP Policy Information Point - The system entity that acts as a source of attribute values (i.e. a resource, subject, environment)

PRP Policy Retrieval Point - Point where the XACML access authorization policies are stored, typically a database or the filesystem.

======================

SAML - Security Assertion Markup Language

Exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

SAML addresses web browser SSO, and problems migrating it from intranet to internet

Another more recent approach to addressing the browser SSO problem is OpenID

SPML - Service Provisioning Markup Language

Automates and manages provisioning of resources across networks and organizations.

Uses a Requesting Authority sends requests to PSP.

[Provisioning Service Point (PSP), processes the request and creates or modifies a user account in the target system.]

Exchanging user, resource and service provisioning information (setup, amend and revoke user or system access) between cooperating organizations.

When access to resources by a particular entity needs to be updated or terminated, SPML-enabled applications will allow management to quickly determine all resources currently available to the entity and change the provisioning automatically.

**Note: this language may be used on the test to refer to these technologies:**

Implement XACML to **streamline access control** policy integration across the enterprise.

Implement SPML to **securely automate resource provisioning**.

OASIS Security Services Technical Committee develops both of these language specs.

--------------------------------------------------------

**Trust Models**

Certificate-based authentication is an example of hierarchical trust. (parent-child)

In a peer trust model, there is no one centralized authority that can verify all resources in the model. Instead, the resources establish a transitive relationship: if resource A trusts resource B, and resource B trusts resource C, then resource A trusts resource C. For example, if you request remote access to a host, and the host can verify that you have already established trust with a separate web server that requires the same level of access, then the remote host will trust you. Peer trust models can avoid a single point of failure like in a hierarchical model using a ticketing system, cached credentials, or other technologies, but they are typically more complex and take more time to operate. Active Directory is an example of technology that can implement peer trust between forests and domains

--------------------------------------------------------

**RADIUS**

When a network contains several remote access servers, you can configure one of the servers to be a RADIUS server, and all of the other servers as RADIUS clients. The RADIUS clients will pass all authentication requests to the RADIUS server for verification. User configuration, remote access policies, and usage logging can be centralized on the RADIUS server.

**RADIUS Authentication:**

PAP - Sends user IDs as plaintext. Remote client is connecting to a non-Windows server that does not support strong password encryption. Legacy protocol.

CHAP - Along with a challenge-response mechanism, CHAP hashes passwords with MD5 instead of plaintext. MD5 is insecure, it is also not suitably secure.

EAP - Uses a variety of plug-ins for authentication (extensible). Allows integration with future authentications.

PEAP - Encapsulates EAP in an encrypted TLS tunnel. This strengthens authentication by protecting the exchange from man-in-the-middle attacks.

LEAP - Mutual client-server authentication, as well generating WEP keys for wireless. WEP is insecure, so you should also avoid LEAP.

**Diameter**

- An authentication protocol that improves upon RADIUS by strengthening some of its weaknesses.

- Diameter has a failover mechanism because it is TCP-based, and RADIUS does not being UDP-based.

- RADIUS does not mandate confidentiality per packet, whereas Diameter does by requiring IPSec and TLS.

- Name is from claim that Diameter is twice as good as RADIUS.

- Diameter is stronger, but not as widespread in its implementation due to the lack of products using it.

**Network Policy Server (NPS)** is a Microsoft Windows Server 2012 implementation of a RADIUS server. It helps in administrating VPNs and wireless networks. NPS was known as Internet Authentication Service (IAS) in Windows Server® 2003.

**Network Access Protection (NAP)** is a Windows Server technology that uses RADIUS to evaluate the health state of a host client. Health requirements could mandate that a host be running a particular operating system version, that the host has the latest anti-malware signatures installed, that the host's firewall is enabled, and so on.

--------------------------------------------------------

**LDAP**

LDAP clients authenticate to the LDAP service, and the service's schema defines the tasks that clients can and cannot perform

while accessing a directory database, the form the directory query must take, and how the directory

server will respond. The LDAP schema is extensible, which means you can make changes or add on to it.

Secure LDAP (LDAPS often is SLDAP) is a method of implementing LDAP using SSL/TLS encryption protocols to prevent eavesdropping and man-in-the-middle attacks. LDAPS forces both the client and server to establish a secure connection before any transmissions can occur, and if the secure connection is interrupted or dropped, LDAP likewise closes. The server implementing LDAPS requires a signed certificate issued by a certificate authority, and the client must accept and install the certificate on their machine.

--------------------------------------------------------

Microsoft Forefront Identity Manager (FIM) now simply known as "Microsoft Identity Manager."

FIM embeds self-help tools in Microsoft Outlook so end users can manage routine aspects of identity and access such as resetting their own passwords without requiring help desk assistance. FIM also alllows end users to create their own security and email distribution lists and decide who to include in those lists. IT administrators can use FIM to manage digital certificates and smart cards. FIM also provides administrative and automation tools.

"allows administrators to synchronize identities directly to the cloud, while also supporting on-premises systems that use and exchange identity information, according to a Microsoft blog post announcing the preview availability.

"integrates with Active Directory and Exchange Server to provide identity synchronization, certificate management, user password resets and user provisioning from a single interface."

"Customers can enable single sign-on to an application that supports Azure Active Directory by sharing the right attributes and values with Azure Active Directory to enable those new scenarios," ... "The new Identity Manager also offers a customizable password-management option with multi-factor (aka two factor) authentication and deeper attribute integration with Active Directory, Microsoft's directory service, the post added.

--------------------------------------------------------

**Kerberos**

Authentication service based on time-sensitive ticket-granting system.

Used in Active Directory. Developed by MIT to use as an SSO method.

Manages access control to different services from one authentication server.

Protects against MitM and replay attacks.

**In the Kerberos process:**

1. A user logs on to the domain.

2. The user requests a ticket granting ticket (TGT) from the authenticating server.

3. The authenticating server responds with a time-stamped TGT.

4. The user presents the TGT back to the authenticating server and requests a service ticket to access a specific resource.

5. The authenticating server responds with a service ticket.

6. The user presents the service ticket to the resource.

7. The resource authenticates the user and allows access.

**Guidelines for Implementing Authentication and Authorization**

Research the pros and cons of using each authentication and authorization scheme.

Implement certificate-based authentication in web servers through SSL/TLS.

Implement an SSO scheme in directory services to streamline user sign in and to ease the burden of having to remember many passwords.

Implement SSO with multiple levels and factors of authentication to mitigate unauthorized access.

Use OAuth to secure client credentials that your enterprise app uses to communicate.

Implement XACML to streamline access control policy integration across the enterprise.

Implement SPML to securely automate resource provisioning.

Configure a RADIUS server to use a secure protocol like LEAP, and not insecure protocols like PAP and PEAP.

Encrypt LDAP communications with SSL/TLS to prevent man-in-the-middle attacks.

Implement Kerberos in an Active Directory domain to ensure mutual authentication and protect against man-in-the-middle and replay attacks.

**Attestation records/ software holds the user accountable for performing certain task on systems or the network.**

Verifying that least privilege is being upheld.

Attestation agent is accountable for what they attest to.

User-focused:

Monitors the privileges specific users have.

Removal or additional based on changes in employment.

Resource-focused:

Monitor system for which users have which privileges.

Emphasis is on security of specific resources.

When denying a supervisor to keep permissions on something when promoted to a different area:

"If you keep priviledges to that area, you would be a suspect if the people with that privilege get investigated"

Attestation is supposed to help with that, being that it can specify who specifically did what..

Identity Propagation

Another useless slide graphic showing user --> webApp frontend --> DB. transitive permissions issue

Expand this to users having priv to enter DB data but not query it. App security issue!

Identity Federation

Another useless slide graphic showing that xbox, Onedrive, Outlook, Windows all sign in with the same microsoft account

Windows uses SSO but not all federated identity systems do and will ask for reauthetication

**Identity Federation Methods**

Security Assertion Markup Language (SAML)

SAML is an XML-based framework for exchanging security-related information such as user authentication, entitlement, and attributes. This information is communicated in the form of assertions over a secure HTTP connection, which conveys the identity of subjects and authorization decisions about the access level of the subjects. SAML contains components such as assertions, protocol, and binding. Authentication assertions contain information about any acts of authentication or user identity validation, attribute assertions contain information about users, and authorization assertions contain information about the level of access for each user. Clients request assertions from SAML authorities and get a response from them using the protocol defined by SAML.

OpenID

OpenID is a method of authenticating users with certain sites that participate in an OpenID system. This allows them to retain a single account for all participating sites. A user will register with an OpenID system in a given domain like they would with any other account. A site under this OpenID domain will then give the user the option to sign in using this system. The site then contacts its external OpenID provider in order to verify that the login credentials supplied by the user are correct. Internet companies such as Google and Yahoo! use their own OpenID systems. (also LastPass)

Shibboleth

Shibboleth is a federated identity method **based on SAML** that is often employed by universities or public service organizations. In a Shibboleth implementation, a user attempts to retrieve resources from a Shibboleth- enabled website, which then sends SAML authentication information over URL queries. The user is then redirected to an identity provider with which they can authenticate using this SAML information. The identity provider then responds to the service provider (the Shibboleth-enabled website) with the proper authentication information. The site validates this response and grants the user access to certain resources based on their SAML information.

Where Are You From (WAYF)

WAYF is an SSO implementation that is centered around asking users what institution they are from before they are allowed access to a service provider. In WAYF, a user connects to a web resource, which then refers to a WAYF identity management system. The system asks which institution the user is from, which mandates that the user log in to their institution if they are not already. After the user is successfully logged in to their own institution, the WAYF informs the user what identity information will be sent to the service provider. The user must consent before this information is sent. The service provider then decides, based on this information, whether to allow the user access.

**Guidelines for Implementing Advanced Identity Management**

Implement user-focused attestation in environments where personnel are routinely switching job functions.

Implement resource-focused attestation in environments where sensitive resources need consistent privilege verification.

Implement identity propagation in systems that incorporate a mix of vendor processes, especially if those processes are built on different frameworks.

Ensure that identity propagation has fine-grained control over whose identity gets propagated where to conform to access control policies.

Implement identity federation to streamline user interaction and identity management within the enterprise.

Implement identity federation to reduce the cost and the risk associated with multiple identity management systems.

Use a SAML-based federation framework like Shibboleth to securely manage individual identities.

Q26

...The telephone and Internet services portions of the company will each be separate subsidiaries of the parent company....all three companies must share customer data for the purposes of accounting, billing, and customer authentication. The solution must use open standards, and be simple and seamless for customers, while only sharing minimal data between the companies. Which of the following solutions is BEST suited for this scenario?

A. The companies should federate, with the parent becoming the SP, and the subsidiaries becoming an IdP.

B. The companies should federate, with the parent becoming the IdP, and the subsidiaries becoming an SSP.

C. The companies should federate, with the parent becoming the IdP, and the subsidiaries becoming an SP.

D. The companies should federate, with the parent becoming the ASP, and the subsidiaries becoming an IdP.

Answer: C

Explanation: Is mostly provided above. parent IdP and subsidiaries provide the services.

IdP - Identity provider; SP - Service provider; SSP - Storage service provider; ASP - Application service provider

**---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**New Encryption Additions**

**Camellia**

Camellia is a Feistel cipher with either 18 rounds (when using 128-bit keys) or 24 rounds (when using 192 or 256-bit keys). Every six rounds, a logical transformation layer is applied: the so-called "FL-function" or its inverse. Camellia uses four 8 × 8-bit S-boxes with input and output affine transformations and logical operations. The cipher also uses input and output key whitening. The diffusion layer uses a linear transformation based on a matrix with a branch number of 5.[citation needed]

Security analysis

Camellia is considered a modern safe cipher. Even using the smaller key size option (128 bits), it's considered infeasible to break it by brute-force attack on the keys with current technology. There are no known successful attacks that weaken the cipher considerably. The cipher has been approved for use by the ISO/IEC, the European Union's NESSIE project and the Japanese CRYPTREC project. The Japanese cipher has security levels and processing abilities comparable to the AES/Rijndael cipher.[1]

Camellia is a block cipher which can be completely defined by minimal systems of multivariate polynomials.[vague][3]

The Camellia (as well as AES) S-boxes can be described by a system of 23 quadratic equations in 80 terms.[4]

The key schedule can be described by 1,120 equations in 768 variables using 3,328 linear and quadratic terms.[3]

The entire block cipher can be described by 5,104 equations in 2,816 variables using 14,592 linear and quadratic terms.[3]

In total, 6,224 equations in 3,584 variables using 17,920 linear and quadratic terms are required.[3]

The number of free terms is 11,696, which is approximately the same number as for AES.

Theoretically, such properties might make it possible to break Camellia (and AES) using an algebraic attack, such as Extended Sparse Linearisation, in the future (provided that the attack becomes feasible). With today's technology, such an attack would take years to compute.

**CAST5 and 6**

In cryptography, CAST-128 (alternatively CAST5) is a symmetric-key block cipher used in a number of products, notably as the default cipher in some versions of GPG and PGP. It has also been approved for Government of Canada use by the Communications Security Establishment. The algorithm was created in 1996 by Carlisle Adams and Stafford Tavares using the CAST design procedure.[1]

Another member of the CAST family of ciphers, CAST-256 (a former AES candidate) was derived from CAST-128. According to some sources, the CAST name is based on the initials of its inventors, though Bruce Schneier reports the authors' claim that "the name should conjure up images of randomness".[2]

CAST-128 is a 12- or 16-round Feistel network with a 64-bit block size and a key size of between 40 and 128 bits (but only in 8-bit increments). The full 16 rounds are used when the key size is longer than 80 bits.[3]

Components include large 8×32-bit S-boxes based on bent functions, key-dependent rotations, modular addition and subtraction, and XOR operations. There are three alternating types of round function, but they are similar in structure and differ only in the choice of the exact operation (addition, subtraction or XOR) at various points.

Although Entrust holds a patent on the CAST design procedure, CAST-128 is available worldwide on a royalty-free basis for commercial and non-commercial uses.

CAST-256 (or CAST6) is a symmetric-key block cipher published in June 1998. It was submitted as a candidate for the Advanced Encryption Standard (AES); however, it was not among the five AES finalists. It is an extension of an earlier cipher, CAST-128; both were designed according to the "CAST" design methodology invented by Carlisle Adams and Stafford Tavares. Howard Heys and Michael Wiener also contributed to the design.

CAST-256 uses the same elements as CAST-128, including S-boxes, but is adapted for a block size of 128 bits – twice the size of its 64-bit predecessor. (A similar construction occurred in the evolution of RC5 into RC6). Acceptable key sizes are 128, 160, 192, 224 or 256 bits. CAST-256 is composed of 48 rounds, sometimes described as 12 "quad-rounds", arranged in a generalized Feistel network.

In RFC 2612, the authors state that, "The CAST-256 cipher described in this document is available worldwide on a royalty-free and licence-free basis for commercial and non-commercial uses."

Currently, the best public cryptanalysis of CAST-256 in the standard single secret key setting that works for all keys is the zero-correlation cryptanalysis breaking 28 rounds with 2246.9 time and 298.8 data.[1]

**BH**

The Boneh–Franklin scheme is an identity-based encryption system proposed by Dan Boneh and Matthew K. Franklin in 2001

**IDEA**

International Data Encryption Algorithm (IDEA), originally called Improved Proposed Encryption Standard (IPES), is a symmetric-key block cipher designed by James Massey of ETH Zurich and Xuejia Lai and was first described in 1991. The algorithm was intended as a replacement for the Data Encryption Standard (DES). IDEA is a minor revision of an earlier cipher, Proposed Encryption Standard (PES).

The cipher was designed under a research contract with the Hasler Foundation, which became part of Ascom-Tech AG. The cipher was patented in a number of countries but was freely available for non-commercial use. The name "IDEA" is also a trademark. The last patents expired in 2012, and IDEA is now patent-free and thus completely free for all uses.[2][3]

IDEA was used in Pretty Good Privacy (PGP) v2.0, and was incorporated after the original cipher used in v1.0, BassOmatic, was found to be insecure.[4] IDEA is an optional algorithm in the OpenPGP standard.

IDEA operates on 64-bit blocks using a 128-bit key, and consists of a series of eight identical transformations (a round, see the illustration) and an output transformation (the half-round). The processes for encryption and decryption are similar. IDEA derives much of its security by interleaving operations from different groups — modular addition and multiplication, and bitwise eXclusive OR (XOR) — which are algebraically "incompatible" in some sense. In more detail, these operators, which all deal with 16-bit quantities, are:

Bitwise eXclusive OR (denoted with a blue circled plus ⊕).

Addition modulo 216 (denoted with a green boxed plus ⊞).

Multiplication modulo 216+1, where the all-zero word (0x0000) in inputs is interpreted as 216 and 216 in output is interpreted as the all-zero word (0x0000) (denoted by a red circled dot ⊙).

After the eight rounds comes a final “half round”, the output transformation illustrated below (the swap of the middle two values cancels out the swap at the end of the last round, so that there is no net swap)

(see Wikipedia)

The key can be recovered with a computational complexity of 2126.1 using narrow bicliques. This attack is computationally faster than a full brute force attack, though not, as of 2013, computationally feasible.

**CEH Materials**

**Module5 - System Hacking**

***Meaning of "offline" and "online" here***

"Online attacks means directly communicating to victim machine - it DOES NOT MEAN NETWORK "ONLINE" - USB is "ONLINE"

Similarly "Offline attack" includes DNA (distributed network attack) because you are not accessing that machine yourself directly

**Part 1 - Cracking Passwords**

Non-electronic Attacks (don't need tech knowledge): Shoulder surfing, social engineering, dumpster diving

Active online attacks (directly communicating to victim machine): dictionary and brute forcing, rule-based (there is some info about password), hash injection and phishing, trojan/spyware/keyloggers (run in background and collect login info), password guessing

Passive online attacks (not communicating w/ victim): Wire sniffing, manin the middle, replay

Offline attack (on password file): precompiled hashes (rainbow tables), distributed network

**Active online attack: password guessing** - less frequent, high failure rate

Find a valid user, list possible passwords, rank from high to low, try them.

Default Passwords: a form of password guessing

cirt.org, default-password.info, defaultpassword.us, passwordsdatabase.com, w3dt.net, virus.org, open-sez.me, securityoverride.org, routerpasswords.com, fortypoundhead.com

Active **online attack**: USB drive:

PassView, pass hacking tool - put on USB; create autorun.inf to launch.bat (contents: start pspv.exe/stext pspv.txt); insert usb and program autoruns in background, saves passwords to .txt file on drive.

Active online attack: hash injection

User logs in (logged-on hashes are stored in the SAM file), attacker compromises server with local or remote exploit, grabs a logged-on domain admin account hash, then *injects the compromised hash into a local session to validate to network resources (log into the domain controller)*

"Pass the hash" is giving the hash in place of password for LM/NTLM authentication access

**Passive online attack: Wire sniffing**

Packet sniffer tools on LAN to get raw traffic

- for sensitive info like passwords or email

("computationally complex, hard to perpetrate, tools available" in graphic of slide)

wire sniffing with wireshark

**Passive online attack: Main in the Middle, Replay attacks**

MITM- Attacker gains/acquires access to session between victim and server to extract info, etc

Replay- packets/auth tokens captured using sniffer, tokens placed back on network to gain access

"hard to perpetrate, (?)trusted by one or both sides, can sometimes be broken by invalidating traffic"

Offline attack: Rainbow tables

Rainbow table defined as "a precomputed table which contains word lists like dict files and brute force lists and their hash values"

Capture the hash of password, compare to precomputed hash table, find match, password cracked.

Create rainbow tables with rtgen (project-rainbowcrack.com) and winrtgen (oxid.it)

Example: rtgen hash\_algorithm charset plaintext\_len\_min plaintext\_len\_max

table\_index chain\_len chain\_num part\_index

- winrtgen supports supports LM, FastLM, NTLM, LMCHALL, HalfLMCHALL, NTLMCHALL, MSCACHE, MD2, MD4, MD5, SHA1, RIPEMD160, MySQL323, MySQLSHA1, CiscoPIX, ORACLE, SHA-2 (256), SHA-2 (384) and SHA-2 (512) hashes.

**Offline attack:** DNA **(distributed network attack**)

"Recovering passwords from hashes or password protected files using the unused processing power of machines across the network to decrypt (like a botnet or like SetiOnline, and runs in the BG)

DNA Manager is installed at a central location to run a net of DNA clients to do passwd cracking.

It coordinates the attack and allocates small portions of the key search/hunt to each of the clients

Elcomsoft Distributed Password Recovery (elcomsoft.com)

Is a DNA solution. LAN internet or both. Plug-in architecture for additional file formats

Scheduling and load balancing.

Install and remove agents remotely.

Communication is encrypted.

**Microsoft Authentication**

Security Accounts Manager (SAM) database

- AD database in domains passwords stored as hashes

NTLM Authentication in NTLM and LM authentication protocols

- different hashing methods

Kerberos

- new default since Win2k Server- replacing authentication for client/server

- NTLM still supported in Server2003, and even later

- Hash is null-padded to 14 bytes; is split into two 7-byte halves If LM hash is from a pass of less than 8 chars, the second half is filled with 0xAAD3B435B51404EE

**c:\windows\system32\config\SAM**

Format of SAM entries:

user:UID:LM Hash:NTLM Hash:::

LM hashes disabled in Vista and later- LM will be blank on those systems

NTLM Auth Process:

User types password --> runs through hash algorithm --> PC sends login request to DC

DC sends login challenge --> PC sends hash --> DC compares to stored copy, and if matches, success

**Kerberos**

- Time-sensitive ticket-granting system.

- Originally developed by MIT for SSO method, access control to different svcs from one auth server.

- Protects against MitM and replay attacks.

- Kerberos employs a Key Distribution Center (KDC) containing a Authentication Server (for user authentication), and a Ticket Granting Server (TGS) to give service tickets for access to Application Server services, and a database in the backend for the entire KDC

In the Kerberos process:

- User logs on to the domain, requests ticket granting ticket (TGT) from the auth server (AS).

- Authentication Server responds with time-stamped TGT.

- User presents the TGT to the Ticket Granting Server for a service ticket to access a specific resource.

- User presents the service ticket to the resource, the resource authenticates the user and allows access.

Password Salting

- random string of characters is added to a password before hashing is calculated

- make it more difficult to reverse the hash and defeat pre-computed hash attasks

- Slide makes special note that Windows password hashes are *NOT* salted

pwdump7 (tarasco.org)

- grabs LM/NTLM hashes of local accounts from the SAM DB

fgdump (foofus.net)

- works like pwdump but also gets cached credentials and allows remote network execution

fgdump.exe -h 192.168.0.10 -u AdminUserName -p myPassword

- logs in as user and dumps remote machines goods

L0phtCrack - password auditing and recovery- features scheduling, hash extraction from 64-bit Windows versions, network monitoring and decoding. l0phtcrack.com

Ophcrack - Windows password cracker based on Rainbow tables. GUI and multiple platform versions - ophcrack.sourceforge.net

Cain & Able - oxid.it

sniffing the network, cracking encrypted passwords using dictionary, brute-force and cryptanalysis attacks

RainbowCrack project - rainbowcrack.com

cracks with rainbow tables "uses a time-memory tradeoff algorithm to crack hashes

Password Cracking Tools

LSASecretsView (nirsoft.net), Proactive System Password Recovery (elcomsoft.com), Offline NT Password and Registry Editor (pogostick.net), WinPassword (lasbit.com), PasswordUnlocker.com Bundle, Passware Kit (lostpassword.com), PasswordsPro (insidepro.com), John the Ripper (openwall.com), LPC (lpcsoft.com), Windows-Password-Cracker.com, RecoverWindowsPassword.com, Password Cracker (amlpages.com), Windows Password Recovery (passcape.com), CloudCracker.com, top-password.com Recovery Bundle, WindowsPasswordRecovery.com Tool, krbpwguess (cqure.net), HashSuite.openwall.NET, THC-Hydra (thc.org), InsidePro.com

Mobile: FlexiSPY Password Grabber - (flexispy.com - expensive and not on KAT) - captures the security pattern used to access the phone itself and crack the password used to unlock the iPhone, plus the actual passwords they use for social messaging. Allows you to login to their FB, Skype, Twitter, Pininterest, LinkedIn, GMail and other accounts directly from your computer.

Defense on password cracking

Enable information security audit to monitor and track password attacks

Enable SYSKEY with strong password to encrypt SAM DB

Monitor server logs for brute force, Account lockout on too many password guesses.

Passwords - don't share, no dictionary words or defaults, and no storing in an unsecured location

No cleartext protocols or weak encryption, password change policy every 30 days, account lock on 3 password attempts

Make sure no apps store or write passwords to disk in clear text

Use a random string (salt) with password before encrypting

Don't use same password on password change. Never use DOB, pet's or kid's name, etc.

Ch 14 - Wireless Networks

46% total mobile data traffic was offloaded onto fixed wifi rather than LTE/3G

BSSID basic service set

ISM Band set freq for industrial scientific medical

orthogonal frequency-division multiplexing OFDM - encoding data on multiple carrier frequencies

Direct-sequence spread spectrum DSS - orig signal multiplied with pseudo rand noise spreading code

Frequency hopping SS - rapidly hopping carrier frequency channels

Disadvantages to wifi - security might not meet expectations

more users on same network= bandwidth suffers

wifi enhancements may require new hardware electronic interference

BSSID on one access point

ESSID on multiple APs

Hotspot 3g/4g

wireless stds chart 802.11a-n 2.4/5, frequencies, modulation, speed range

802.11i wpa2-ent/wpa2-personal

802.16 wimax, 30 miles, 70-1000 mbps, 10-66ghz

higher the freq, shorter range, better speed

Whats an SSID single shared identifier

AP broadcast - if SSID of netwk changed reconfiguration of SSID on every host required

human readable text

wifi auth modes

open system auth vs psk

AAA radius tacacs, diameter used for centralized auth server solution

global auth key encrypted with per-station unicast session key (AP to client)

War walking/driving/flying/chalking defined

antennae:

directional, omnidir, parabolic grid( like satellite) can be used to focus on one signal better), yagi(unidir, UHF/VHF), dipole (bidirectional)

Types of wireless encryptions

CCMP 128bit keys, 48bit IV for replay detection

WPA 48bit IV 32 bit CRC, TKIP

24 bit IVto form stream cipher RC4

WEP 64 - 40bit key

128 - 104

256 - 232 bit key

IV+wep key= wep seed

WPA 128-bit temporal key TKIP

temporal keys explained- slides went too fast for notetaking

<http://insights.dice.com/2014/03/20/wpa2-security-cracked-without-brute-force/>

https://latesthackingnews.com/2015/02/21/cracking-wpawpa2-psk-encryption/

aircrack-ng suite slide- good overview of all the tools in one slide.

note to self: review Cisco password recovery process in light of CEH

**Topics: (Outline to fill in)**

Emphasize and define this: [Confusion and Diffusion "Types of crypto THIS", Asym/Sym types of crypto THIS (aka public and secret key), stream and block are types of crypto "THIS",

Intro

- Hashes vs Crypto

**General concepts**

- General cryptography concepts: base types (diffusion, confusion, etc)

**Rotor Machines**

In the 1920s, various mechanical encryption devices were invented to automate the process of encryption. Most were based on the concept of a rotor, a mechanical wheel wired to perform a general substitution.

A rotor machine has a keyboard and a series of rotors, and implements a version of the Vigenère cipher. Each rotor is an arbitrary permutation of the alphabet, has 26 positions, and performs a simple substitution. For example, a rotor might be wired to substitute “F” for “A,” “U” for “B,” “L” for “C,” and so on. And the output pins of one rotor are connected to the input pins of the next.

Figure 1.4 Columnar transposition cipher.

For example, in a 4-rotor machine the first rotor might substitute “F” for “A,” the second might substitute “Y” for “F,” the third might substitute “E” for “Y,” and the fourth might substitute “C” for “E”; “C” would be the output ciphertext. Then some of the rotors shift, so next time the substitutions will be different.

It is the combination of several rotors and the gears moving them that makes the machine secure. Because the rotors all move at different rates, the period for an n-rotor machine is 26n. Some rotor machines can also have a different number of positions on each rotor, further frustrating cryptanalysis.

The best-known rotor device is the Enigma. The Enigma was used by the Germans during World War II. The idea was invented by Arthur Scherbius and Arvid Gerhard Damm in Europe. It was patented in the United States by Arthur Scherbius [1383]. The Germans beefed up the basic design considerably for wartime use.

The German Enigma had three rotors, chosen from a set of five, a plugboard that slightly permuted the plaintext, and a reflecting rotor that caused each rotor to operate on each plaintext letter twice. As complicated as the Enigma was, it was broken during World War II. First, a team of Polish cryptographers broke the German Enigma and explained their attack to the British. The Germans modified their Enigma as the war progressed, and the British continued to cryptanalyze the new versions. For explanations of how rotor ciphers work and how they were broken, see [794,86,448,498,446,880,1315,1587,690]. Two fascinating accounts of how the Enigma was broken are [735,796].

**Simple XOR**

XOR is exclusive-or operation: ‘^’ in C or ⊕ in mathematical notation. It’s a standard operation on bits:

0 ⊕ 0 = 0

0 ⊕ 1 = 1

1 ⊕ 0 = 1

1 ⊕ 1 = 0

Also note that:

A ⊕A = 0

A ⊕ B ⊕ B = A

(A ⊕ B) ⊕ C = A ⊕ (B ⊕ C)

A ⊕ (A ⊕ B)= B ⊕ 0 = B

The symmetric XOR algorithm is like a Vigenère polyalphabetic cipher. It’s prevalent in commercial software packages with plaintext XORed with a keyword to generate ciphertext. Since XORing the same value twice restores the original, encryption and decryption use exactly the same program: P ⊕ K = C, C ⊕ K = P

There’s no real security here- it's trivial to break, even without computers

Despite this, the list of software vendors that tout this toy algorithm as being “almost as secure as DES” is staggering. It is the algorithm (with a 160-bit repeated “key”) that the NSA finally allowed the U.S. digital cellular phone industry to use for voice privacy.

Assume the plaintext is English. Furthermore, assume the key length is any small number of bytes. Here’s how to break it:

1. Discover the length of the key by a procedure known as counting coincidences [577]. XOR the ciphertext against itself shifted various numbers of bytes, and count those bytes that are equal. If the displacement is a multiple of the key length, then something over 6 percent of the bytes will be equal. If it is not, then less than 0.4 percent will be equal (assuming a random key encrypting normal ASCII text; other plaintext will have different numbers). This is called the index of coincidence. The smallest displacement that indicates a multiple of the key length is the length of the key.

2. Shift the ciphertext by that length and XOR it with itself. This removes the key and leaves you with plaintext XORed with the plaintext shifted the length of the key. Since English has 1.3 bits of real information per byte (see Section 11.1), there is plenty of redundancy for determining a unique decryption.

**One-Time Pads**

Believe it or not, there is a perfect encryption scheme. It’s called a one-time pad, and was invented in 1917 by Major Joseph Mauborgne and AT&T’s Gilbert Vernam. Classically, a one-time pad is nothing more than a large nonrepeating set of truly random key letters, written on sheets of paper, and glued together in a pad. In its original form, it was a one-time tape for teletypewriters. The sender uses each key letter on the pad to encrypt exactly one plaintext character. Encryption is the addition modulo 26 of the plaintext character and the one-time pad key character.

Each key letter is used exactly once, for only one message. The sender encrypts the message and then destroys the used pages of the pad or used section of the tape. The receiver has an identical pad and uses each key on the pad, in turn, to decrypt each letter of the ciphertext. The receiver destroys the same pad pages or tape section after decrypting the message. New message- new key letters.

For example, if the message is ONETIMEPAD, and the key sequence from the pad is TBFRGFARFM then the ciphertext is IPKLPSFHGQ (O + T mod 26 = I, N + B mod 26 = P, E + F mod 26 = K etc.

Assuming an eavesdropper can’t get access to the one-time pad used to encrypt the message, this scheme is perfectly secure. A given ciphertext message is equally likely to correspond to any possible plaintext message of equal size.

Since every key sequence is equally likely (remember, the key letters are generated randomly), an adversary has no information with which to cryptanalyze the ciphertext. Since every plaintext message is equally possible, there is no way for the cryptanalyst to determine which plaintext message is the correct one. A random key sequence added to a nonrandom plaintext message produces a completely random ciphertext message and no amount of computing power can change that.

There are two conditions to OTP: the key letters have to be generated truly randomly (no pseudo-random number generators) and you can never use the key sequence again, ever. Even if you use a multiple-gigabyte pad, if a cryptanalyst has multiple ciphertexts whose keys overlap, he can reconstruct the plaintext. He slides each pair of ciphertexts against each other and counts the number of matches at each position. If they are aligned right, the proportion of matches jumps suddenly- the exact percentages depend on the plaintext language. It’s like the index of coincidence, but with just two “periods” to compare.

With data, we use a one-time pad of bits and rather than adding the plaintext to the one-time pad, use an XOR. To decrypt, XOR the ciphertext with the same one-time pad. Everything else remains the same and the security is just as perfect.

There are a few problems. Since the key bits must be random and can never be used again, the length of the key sequence must be equal to the length of the message. A one-time pad might be suitable for a few short messages, but it will never work for a 1.544 Mbps communications channel. You can store 650 megabytes worth of random bits on a CD-ROM, but there are problems. First, you want exactly two copies of the random bits, but CD-ROMs are economical only for large quantities. And second, you want to be able to destroy the bits already used. CD-ROM has no erase facilities except for physically destroying the entire disk. Digital tape is a much better medium for this sort of thing.

Even if you solve the key distribution and storage problem, you have to make sure the sender and receiver are perfectly synchronized. If the receiver is off by a bit (or if some bits are dropped during the transmission), the message won’t make any sense. On the other hand, if some bits are altered during transmission (without any bits being added or removed—something far more likely to happen due to random noise), only those bits will be decrypted incorrectly. But on the other hand, a one-time pad provides no authenticity.

1.6 Computer Algorithms

There are many cryptographic algorithms. These are three of the most

common:

— DES (Data Encryption Standard) is the most popular computer encryption algorithm. DES is a U.S. and international standard. It is a symmetric algorithm; the same key is used for encryption and decryption.

— RSA (named for its creators—Rivest, Shamir, and Adleman) is the most popular public-key algorithm. It can be used for both encryption and digital signatures.

— DSA (Digital Signature Algorithm, used as part of the Digital Signature Standard) is another public-key algorithm. It cannot be used for encryption, but only for digital signatures.

These are the kinds of stuff this book is about.

**Confusion and Diffusion**

***"The two basic techniques for obscuring the redundancies in a plaintext message," ...***

***"two properties of the operation of a secure cipher"***

***- Schneier, 1996; Shannon; "Communication Theory of Secrecy Systems" 1949.***

Confusion (substitution)

- obscures the relationship between the plaintext, the ciphertext, and the key to thwart finding redundancies, statistical relationships and patterns that are exploited by linear and differential cryptanalysis. Good confusion makes ciphertext too complicated so powerful cryptanalytic tools won’t work. The easiest method is substitution

The simple Caesar Cipher, in which every identical letter of plaintext is substituted by the character alphabetically three to the right modulo 26, is easy to break since the ciphertext is a rotation of the plaintext alphabet and not an arbitrary substitution. In ROT13 (not intended for security), every letter is rotated 13 places (A is replaced by N, etc). Encrypting a file twice with ROT13 restores the original file. P = ROT13 (ROT13 (P))

In the secure modern ciphers a long block of plaintext is substituted for a different block of ciphertext, and the mechanics of the substitution change with each bit in the plaintext or key. The famous German Enigma code in World War II was a complex substitution cipher

Diffusion (transposition/ permutation).

- dissipates the redundancy or other influence of individual plaintext or key bits over as much of the ciphertext as possible. The simplest way to cause diffusion is through transposition (aka permutation). The old columnar transposition simply rearranges the letters of the plaintext, but modern permutation ciphers use other forms of diffusion that can diffuse parts of the message throughout the entire message.

Generally, diffusion alone is easily cracked (double transposition is good but not enough)

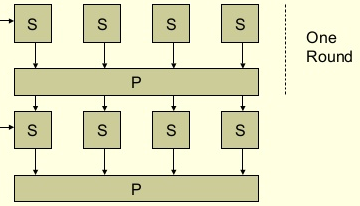
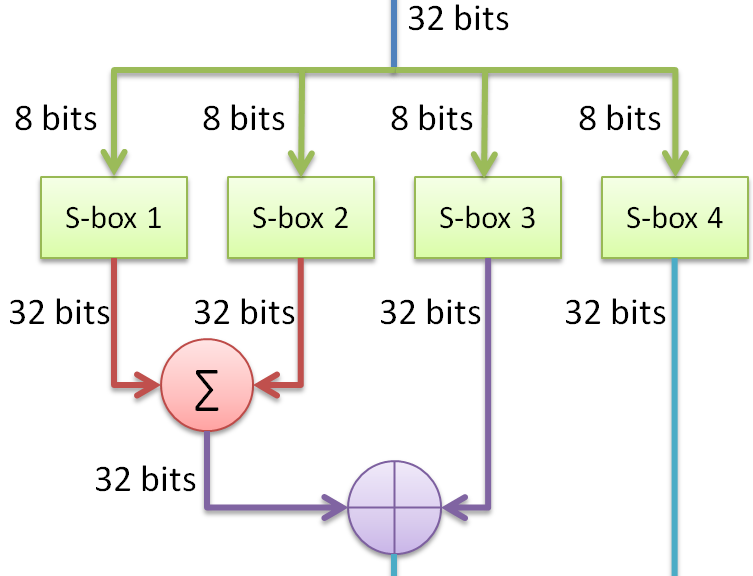
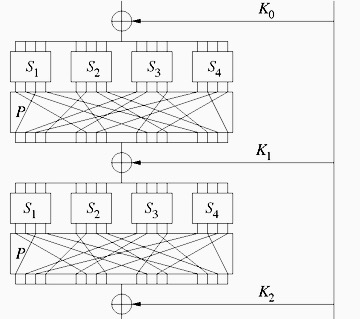
***- Confusion provides a secure encryption component. Adding diffusion makes cryptanalysis harder***

***- Stream ciphers generally rely on confusion alone, although some feedback schemes add diffusion.***

***- Block algorithms generally use both confusion and diffusion.***

Product Ciphers- (putting components together efficiently)

Product ciphers use alternating substitution and transposition phases (rounds) to achieve both confusion and diffusion respectively. When we diagram out the operations of a cipher, it's components are often arranged in a Feistel function/network, which first divides the input of bits into chunks, which have their own paths through the cipher's algorithm. Down the path, confusion is performed/contained in substitution S-boxes, while diffusion/permutation is performed/contained in P-boxes. Each "box" uses a lookup table for the corresponding output values, to pass to the next function, which could be another round through these boxes, with possibly an XOR or other function in between them. The Feistel network containing all this may also be referred to as a substitution-permutation network, or SP network, and it wraps the functions together in a way that can be used to both encrypt and decrypt.

These structures don't only provide strong encryption, but break down tasks into realistic memory chunks to make things possible. A single key-dependent lookup table of 64 bits of plaintext to 64 bits of ciphertext would be strong algorithm, but the table uses 10^20 bytes (100 Exabytes) - large lookup tables require lots of memory to implement. The whole point of block cipher design is to create something that looks like a large lookup table, but with much smaller memory requirements.

***[preamble: Stream vs block - A Bogus Premise]***

The distinction between block and bit stream ciphers is not always clear-cut: in some modes of operation, a block cipher primitive is used so it acts effectively as a stream cipher, which is how CFB and OFB work

*This is precisely where information developed for certification tests makes people not understand the way this all works: it makes an artificial brick wall between the two types as the basis, an oversimplification so that the questions can be answered on the test*

**Algorithm Types: block and stream ciphers**

Two types of symmetric algorithms: block and (bit) stream ciphers

"Block ciphers operate on data with a fixed transformation on large blocks of plaintext data; (while)

stream ciphers operate with a time-varying transformation on individual plaintext digits." [Rueppel]

- Stream ciphers generally rely on confusion alone, although some feedback schemes add diffusion.

- Block algorithms generally use both confusion and diffusion.

Although block and stream ciphers are very different, block ciphers can be implemented as stream ciphers and stream ciphers can be implemented as block ciphers.

Main differences are in implementation.

- Stream ciphers can be more suitable for hardware because the device sees individual bits.

- Block ciphers can be easier to implement in software, often avoiding time-consuming bit-by-bit operations

- Encrypting the link between the keyboard and the CPU in software sounds ideal for a stream, but generally an encryption block could be at least the width of the data bus.

In the real world, block ciphers seem to be more general (i.e., they can be used in any of the four modes) and stream ciphers seem to be easier to analyze mathematically.

Block ciphers:

- "Operate on blocks of plaintext, ciphertext is usually 64 bits but sometimes longer

- "Same plaintext block will always encrypt to the same ciphertext block, using the same key

Stream ciphers:

- "Streams of plaintext or ciphertext one bit or byte (sometimes even one 32-bit word) at a time"

- "The same plaintext bit or byte will encrypt to a different bit or byte every time it is encrypted"

RC4 is the most used and well-known

- Was an RSA trade secret 1987, leaked to general use in 1994 (aka ARC4 for “alleged” due to trademark.

- RC for "Rivest Cipher" also informally stands for "Ron's Code"

- Used in Wired Equivalent Privacy (WEP), BitTorrent, Microsoft's Remote Desktop Protocol (RDP)

- The prevalent example is WEP, deprecated in 2004 due to its implementation of RC4.

- Used in SSL(1995)/TLS(1999) and WEP. Was chosen for speed and simplicity - usage not advised

- Use for TLS was already frowned upon, but in Feb 2015 was finally prohibited in RFC 7465.

- Pseudorandom generator used. Proposed new random number generators are often compared to RC4's.

- A few OS's use RC4 for random number generation

- Several attacks on RC4 are able to distinguish its output from a random sequence.

- Other official variants/ versions includie RC2, RC5, RC6 (RC4 prevails)

- In 2014 Ron Rivest introduced an updated redesign called Spritz, which is slower and not adopted much

- Other attempts to strengthen RC4 include RC4A, VMPC, and RC4+

- Attacks on RC4 include Klein, Royal Holloway, Bar-mitzvah (often WEP-implementation-specific)

- Fluhrer, Mantin and Shamir attack specifically exploited the first 1024 bits of the key

- Numerous Occurrence MOnitoring & Recovery Exploit (NOMORE) attack against TLS and WPA

----

Other Stream ciphers: A5/1, A5/2, Chameleon, FISH, Helix, ISAAC, MUGI, Panama, Phelix, Pike, SEAL, SOBER, SOBER-128 and WAKE. Comparative table: https://en.wikipedia.org/wiki/Stream\_cipher

***Block Cipher Confidentiality Modes (of Operation) Part 1***

- ECB, CBC, OFB, and CFB date back to 1981, specified in FIPS 81, DES Modes of Operation

- "Usually combines the basic cipher, some sort of feedback, and some simple operations" (Schnier)

- "Operations are simple because the security is a function of the underlying cipher and not the mode"

- The mode "should not compromise the security of the underlying algorithm"

- 2001, NIST added CTR mode in SP800-38A

- 2010, NIST Addendum to SP800-38A - Outlines CBC-CS1, -CS2, and -CS3 (aka CTS "CipherText Stealing mode")

- Other confidentiality modes exist, not approved by NIST.

***Basic Block Cipher modes (ECB and CBC)***

**Electronic Code Book (ECB)**:

Easy, fast, but also weakest. Weak algorithm and is vulnerable to block replay attacks. Very flawed mode- avoid.

Only suitable for encrypting short and random data, like other keys. These uses aren't affected by ECB's problems.

- Each block is encrypted by itself. More than one message can be encrypted with the same key.

- File doesn’t have to be encrypted linearly (do 10 blocks at the end, then the start, etc.)

- Each occurrence of a particular word is encrypted exactly the same\*

- Two identical plaintext blocks will always generate the same ciphertext block.

- Same plaintext block will always encrypt to same ciphertext - susceptible to brute-force; deletion and insertion

- Plaintext patterns not concealed; input to block cipher not randomized (same as plaintext)

- Plaintext is easy to manipulate, blocks can be removed, repeated, or interchanged.

- Padding the plaintext makes it fit ciphertext fixed-block size (usually 64-bit in ECB)

- Processing is parallelizable, but preprocessing is not possible.

- A ciphertext error affects one full block of plaintext. Synchronization error is unrecoverable.

- If a ciphertext bit is accidentally lost or added, all subsequent ciphertext will decrypt incorrectly unless there is some kind of frame structure to realign the block boundaries

*- \*If a DB is encrypted with ECB, then any record can be added, deleted, encrypted, or decrypted independently of any other record, assuming that a record consists of a discrete number of encryption blocks*

**Cipher Block Chaining (CBC)**:

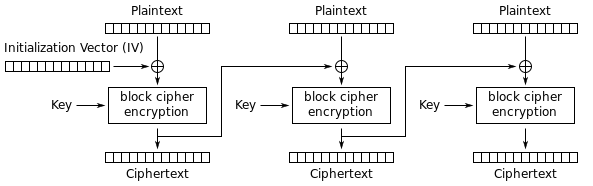
- Invented by IBM in 1976

- encryption of each block depends on all the previous blocks (a feedback mechanism)

- plaintext block is XORed with the previous ciphertext block, before encrypted with algorithm and key

- after plaintext block is encrypted, resulting ciphertext is spit out and also piped to the next block's process

The first block doesn't have a previous block to use, so an IV the same size as the plaintext block kicks it off.



Decrypting is similar- ciphertext decrypted normally and also saved in feedback register; next block decrypted, XORed with the feedback register, etc.

Some messages have a common header: a letterhead, or a “From” line, or whatever. While block replay would still be impossible, this identical beginning might give a cryptanalyst some useful information. An initialization vector (IV) throws some random data as the first block; helps make identical plaintext messages encrypt to different ciphertext messages

- sometimes it's just a timestamp, should be unique for each message encrypted with the same key, it is not required.

When the receiver decrypts this block, it's used to fill the feedback register.

The IV need not be secret in CBC; it can be transmitted in the clear with the ciphertext. If this seems wrong, consider the following argument. Assume that we have a message of several blocks: *B1, B2,..., Bi. B1* is encrypted with the IV. B2 is encrypted using the ciphertext of B1 as the IV. B3 is encrypted using the ciphertext of *B2* as the IV, and so on. So, if there are n blocks, there are n-1 exposed IVs, even if the original IV is kept secret. So there’s no reason to keep the IV secret; the IV is just a dummy ciphertext block- you can think of it as B0 to start the chaining.

- Easy and fast; best for encrypting files; Almost always the best choice if application is software-based

- More than one message can be encrypted with the same key.

- Plaintext patterns concealed by XORing with previous ciphertext block.

- An IV is used to encrypt the first block, since there is no "previous ciphertext block"

- Processes 64-bit blocks; if last block does not reach the block boundary, it is filled with padding.

- Ciphertext is up to one block longer than the plaintext, not counting the IV.

- A ciphertext error affects one full block of plaintext and the corresponding bit in the next block.

- Almost never synchronization errors, a synchronization error is unrecoverable.

- No preprocessing. Encryptions not parallelizable; decryption is parallelizable and has a random-access property.

- Plaintext is somewhat difficult to manipulate; blocks can be removed from the beginning and end of the message, bits of the first block can be changed, and repetition allows some controlled changes.

The Propagating Cipher Block Chaining (P-CBC) variant of CBC provides error-checking; integrity check. Kerberos 4 used it but it was found to be flawed and dismissed.

**Padding in ECB and CBC**

ECB requires 64-bit blocks. Most messages don’t divide neatly into 64-bit (or whatever size) encryption blocks; there is usually a short block at the end. Padding is the way to deal with this problem.

- Pad the last block with some regular pattern- zeros, ones, alternating ones and zeros- to make it a complete block.

- If you need to delete the padding after decryption, add the number of padding bytes as the last byte of the last block.

Example: assume the block size is 64 bits and the last block consists of 3 bytes (24 bits).

Five bytes of padding are required to make the last block at 64 bits; add 4 bytes of padding and a final byte with the number 5. On decryption, it is understood to delete the last 5 bytes of the block.

For this method to work correctly, every message must be padded.

Even if the plaintext ends on a block boundary, you have to pad one complete block.

Otherwise, you can use an end-of-file character to denote the final plaintext byte, and then pad after that character.

Padding in CBC works like ECB, but in sometimes the ciphertext has to be exactly the same size as the plaintext.

Suppose a plaintext file has to be encrypted and then replaced in the exact same memory location.

In this case, you have to encrypt the last short block differently.

Assume the last block has J bits. After encrypting the last full block, encrypt the ciphertext again, select the left-most J bits of the encrypted ciphertext, and XOR that with the short block to generate the ciphertext.

Figure 9.4 illustrates this.

The weakness here is that while Mallory cannot recover the last plaintext block, he can change it systematically by changing individual bits in the ciphertext. If the last few bits of the ciphertext contain essential information, this is a weakness. If the last bits simply contain housekeeping information, it isn’t a problem.

**Ciphertext stealing** is a better way (see Figure 9.5) [402]. Pn-1 is the last full plaintext block, and Pn is the final, short, plaintext block. Cn-1 is the last full ciphertext block, and Cn is the final, short, ciphertext block. C’ is just an

intermediate result and is not part of the transmitted ciphertext. The benefit of this method is that all the bits of the plaintext message go through the encryption algorithm.

**CBC Error Propagation**

CBC mode can be characterized as "feedback of the ciphertext at the encryption end and feedforward of the ciphertext at the decryption end". A single bit error in a plaintext block will affect that ciphertext block and all subsequent ciphertext blocks. This isn’t significant because decryption will reverse that effect, and the recovered plaintext will have the same single error.

Ciphertext errors are more common. They can easily result from a noisy communications path or a malfunction in the storage medium. In CBC mode, a single-bit error in the ciphertext affects one block and one bit of the recovered plaintext. The block containing the error is completely garbled. The subsequent block has a 1-bit error in the same bit position as the error.

Error extension: property of taking a small ciphertext error and converting it into a large plaintext error.

Blocks after the second are not affected by the error, so CBC mode is self-recovering.

Two blocks are affected, but the system recovers and continues to work correctly for all subsequent blocks.

CBC is an example of a block cipher being used in a self-synchronizing manner, but only at the block level.

While CBC mode recovers quickly from bit errors, it doesn’t recover at all from synchronization errors.

If a bit is added or lost from the ciphertext stream, then all subsequent blocks are shifted one bit out of position and decryption will generate garbage indefinitely.

*Any cryptosystem that uses CBC mode must ensure that the block structure remains intact, either by framing or by storing data in multiple-block-sized chunks*.

**Structural Vulnerabilities of CBC:**

Since a ciphertext block affects the following block in a simple way:

1. Someone can add blocks to the end of an encrypted message without being detected.

- it will probably decrypt to gibberish, but in some situations this is undesirable.

Solution: structure your plaintext so that you know where the message ends and can detect the addition of extra blocks.

2. Someone can alter a ciphertext block to introduce controlled changes in the following decrypted plaintext block

- Toggle a single ciphertext bit, the entire block will decrypt incorrectly, but the following block will have a 1-bit error in the corresponding bit position. There are situations where this is desirable.

- Solution: the entire plaintext should include some kind of controlled redundancy or authentication.

3. (Hypothetical) Although plaintext patterns are concealed by chaining, very long messages will still have patterns. The birthday paradox predicts that there will be identical blocks after 2m/2 blocks, where m is the block size. For a 64-bit block size, that’s about 34 gigabytes. A message has to be pretty long before this is a problem.

***Stream ciphers***

- Stream of plaintext is processed with a corresponding bit of a pseudorandom cipher one bit at a time

- Fast, require less overhead - one bit at a time perfect for hardware. Output ciphertext is same size as plaintext.

- Good where plaintext quantities are of unknowable length (like a wireless connection)

- Fewer errors, but no guarantee of message integrity;

- Secrecy, but no authentication (traffic can be tampered with, impersonated by MitM)

- As each bit's encryption depends on the state of the keystream generator, the term "state cipher" is also used.

- With binary digits being added to the keystream there is also the name "binary additive stream cipher"

With stream ciphers the fault is that they try to do something proven to work, but completely the wrong way!

- Based on the one-time pad (OTP, Vernam cipher); unbreakable one-time pre-shared key the same size as plaintext

- OTP uses a keystream of truly random digits XORed with plaintext digits one at a time to produce ciphertext

- OTP also requires keystream at least the length of plaintext that will not be used more than once.

A stream cipher fails in 3 ways - uses a key of limited length; repeats the keystream; it's pseudorandom- not random.

**Best practices with stream ciphers**

- Remember that most stream ciphers provide privacy, but no authenticity: messages in transit can still be modified

- Remember that information can be in certificates may not break the cipher, yet can reveal other weaknesses.

- Never reuse the same keystream twice (generally means a different nonce or key be supplied each time

- Use a method that discards the first few bytes of the keystream

- All keystreams (except one-time pad) are "periodic" -they repeat at some point. The longer this period is, the better

- Although keystream generators work with pseudorandom seeds, the closer to random you can get the better

- Avoid weak keys: as much as possible the keystream should be free of any patterns or detectable relationships that correspond to keys or nonces. The ideal is that transmissions would be indistinguishable from random noise.

**Synchronous and Self-Synchronizing: Two categories of Stream Ciphers**

Both primary types of stream ciphers are easiest explained by describing the synchronous stream cipher first.

- In a synchronous cipher, a keystream generator creates a keystream from the key (aka "running key"), then the keystream is XORed with the plaintext, and the ciphertext is spit out.

- A generator has an internal state, which is basically which part of the key is used at that moment to make the keystream. Two keystream generators with the same key and same internal state will produce the same keystream.

- On the decoding end, a keystream in the same state generates the identical keystream, then XORs it with the ciphertext to produce the plaintext.

|  |  |
| --- | --- |
|  | - A stream cipher is defined as *synchronous* simply by the fact that the keystream is generated without the use of either the plaintext or the ciphertext- meaning the keystream is ONLY a function of the key.  - In a synchronous stream cipher both the sender and receiver must be synchronized in that each must be at exactly the same position in their shared key. Any loss or insertion of bits means that they must resynchronize. Big problem since it can force repeating the keystream!. Luckily, errors usually return a corrupted bit, with no interruption. |

*"Synchronous" stream cipher just means that the keystream generation is independent of the plaintext and ciphertext*

A few problems with synchronous stream ciphers:

- Have some ciphertext and it's corresponding plaintext? Just XOR both and you get the keystream. Decrypt the rest.

- Have two ciphertexts from the same keystream? XOR both and you have 2 plaintexts XOR'd together. XOR that with one of the ciphertexts and you can get the keystream. Rotating keys can help minimize this vulnerability.

- Even though a loss or addition of bits requires resynchronization, a corrupted or changed bit will not propagate to other bits, and may not even be detected! Useful when transmission error rate is high, but bad for error detection.

- Also a huge attack vector - a change in a ciphertext digit, can make possible predictable changes to corresponding plaintext bit; flipping a bit in the ciphertext causes the same bit to be flipped in the plaintext.

**Self-Synchronizing Stream Ciphers**

In a synchronous stream's keystream generator, we had an internal state working with the key put into the output function to spit out the keystream. Simultaneously, a next-state function took the old internal state to come up with the next one to use. Instead of setting up the internal state like that, *self-synchronizing* stream ciphers grab the ciphertext that was just generated, and plug it into making the next internal state

|  |  |
| --- | --- |
|  | In a *self-synchronizing* (or *asynchronous*) stream cipher, the keystream is generated as a function of the key and a fixed number of previous ciphertext bits.  Since the internal state is derived from the previous ciphertext bits, synchronization is automatic after *x* bits have been received. Good implementations start things off with a random header to mark the length of ciphertext bits to use |

Problems with self-synchronizing stream ciphers:

Error propagation - for every bit of garbled transmission in the ciphertext, there will be corresponding length of bits decrypted incorrectly until the garbled bits get out of the internal state of the keystream generator.

Vulnerable to playback attack - if the same key is used in a stream, you can play back packet-captured traffic and after resynchronizing into the internal state it will decrypt normally. Unless timestamps are used the other end wouldn't be aware of a replay- (imagine repeatedly replaying traffic resulting in granting bank credits).

Finally, if the stream cipher utilizes plaintext in the keystream generation, it is called *nonsynchronous*

In the military, synchronous stream ciphers are also called "Key Auto-Key" (KAK), while self-synchronous are called "Ciphertext Autokey" (CTAK).

Synchronous stream attack -> pg202-203 9.7 insertion attack example

The length of the period depends on the application - a T1 2^37 per\_\_ (see example in book which means X per day )

**STREAM CIPHER LEFTOVER MATERIAL:**

Efforts to make the keystream more complex include

- employing block ciphers like DES (64-bit) with output feedback mode, but there is still a repetition. When not using full feedback, the resulting stream has a period of around 232 blocks on average; encryption at a rate of 8 MBps, a stream of period 232 blocks will repeat after about 30 min.

- Bit-shifting the key is also done but is largely ineffective

- Nonces can also be employed.

------------------------------------------------

Some applications using the stream cipher RC4 are attackable because of weaknesses in RC4's key setup routine; new applications should either avoid RC4 or make sure all keys are unique and ideally unrelated (such as generated by a well-seeded CSPRNG or a cryptographic hash function) and that the first bytes of the keystream are discarded.

---------

Inside the keystream generator, linear feedback shift registers (LFSRs) are usually used since their structure works fast on hardware. On their own they are insufficient to provide good security. Various schemes have been proposed to increase the security of LFSRs, generally attempts to remove linearity (add complexity).

Non-linear combining functions use multiple LFSRs in parallel, their outputs combined using a non-linear Boolean function. In another approach, clock-controlled generators sequence LFSRs irregularly to change the order each pass. Examples include the alternating step, stop-and-go, and shrinking generator types, that use different methods to rotate the use and output of 2 or 3 LFSRs and combine their results to produce the output. Finally, the option of filter generators pass the state of one LFSR into a non-linear filtering function such as a T-function before producing output.

-----------------------

**Usage**

If a block cipher (not operating in a stream cipher mode) were to be used in this type of application, the designer would need to choose either transmission efficiency or implementation complexity, since block ciphers cannot directly work on blocks shorter than their block size. For example, if a 128-bit block cipher received separate 32-bit bursts of plaintext, three quarters of the data transmitted would be padding. Block ciphers must be used in ciphertext stealing or residual block termination mode to avoid padding, while stream ciphers eliminate this issue by naturally operating on the smallest unit that can be transmitted (usually bytes).

Another advantage of stream ciphers in military cryptography is that the cipher stream can be generated in a separate box that is subject to strict security measures and fed to other devices such as a radio set, which will perform the xor operation as part of their function. The latter device can then be designed and used in less stringent environments.

RC4 is the most widely used stream cipher in software[citation needed]; others include: A5/1, A5/2, Chameleon, FISH, Helix, ISAAC, MUGI, Panama, Phelix, Pike, SEAL, SOBER, SOBER-128 and WAKE.

------------------------

BACK TO BLOCK CIPHERS- CFB, OFB,

-----------------------

http://www.crypto-it.net/eng/theory/modes-of-block-ciphers.html

https://en.wikipedia.org/wiki/Block\_cipher\_mode\_of\_operation

http://www.garykessler.net/library/crypto.html

http://www.umich.edu/~x509/ssleay/fip81/fip81.html

*Schneier suggests in Applied Cryptography: "Stay away from the weird modes. One of the four basic modes- ECB, CBC, OFB, and CFB- is suitable for almost any application. These modes are not overly complex and probably do not reduce the security of the system. While it is possible that a complicated mode might increase the security of a system, most likely it just increases the complexity. None of the weird modes has any better error propagation or error recovery characteristics." This was written prior to NIST approval of CBC-CS/CTS.*

***Block Cipher Confidentiality Modes (of Operation) Part 2:***

***Block Modes Implemented as Stream Ciphers (CFB, OFB, CTR)***

**CFB - Cipher Feedback mode**

*A block mode implemented as a self-synchronizing stream cipher*

- Allows encryption of partial blocks rather than requiring full blocks for encryption (padding not necessary)

*- Often you will see options for CFB, CFB-1, and CFB-8;* 8-bit blocks allow for one ascii character at a time

- Remember that the number of bits there is for the mode itself, separate from how many the block cipher employs.

The example below shows using 8-bit CFB, working with a 64-bit block algorithm

- The rule is, any size of *n*-bits where the *n* is less than or equal to the block size (in this case n=8bits)

- CFB uses a shift register queue the size of the block size (64 bits)

|  |  |
| --- | --- |
|  | - First, an IV is put in the queue (like CBC)  - Queue is encrypted with the key, and the contents dumped into an output buffer.  - Leftmost 8 bits of output buffer are for use, the remainder is discarded.  - The selected 8 bits are XORed with 8 bits of the plaintext to output 8-bits of ciphertext.  - This ciphertext can now be transmitted, a copy is also pushed into the shift register, moving the pre-existing bits up in the queue.  - Bits left in the output buffer are dumped.  - The next plaintext bits are encrypted in the same manner, and it continues |

- Plaintext patterns are concealed; input to the block cipher is randomized.

- Ciphertext is the same size as the plaintext, not counting the IV.

- More than one message can be encrypted with the same key provided that a different IV is used.

- An attacker with some of the plaintext can toggle bits in a given block and decrypt

- if IV is not unique for each message "session" the plaintext is vulnerable. Can be any incrementing number that doesn't repeat (like serial numbers). Data in storage could even use a function of the data lookup index.

- Self-recovering: A single bit error in ciphertext causes single error in plaintext, but also corrupts the shift register until it flushes out. Where m=block size, in n-bit CFG, a single ciphertext error will affect m/n-1 blocks (in 8-bit mode, 9 bytes garbled by a single bit ciphertext error). Also applies to synchronization. (see *self-synchronizing stream ciphers).*

- 1-bit CFB can recover from the addition or loss of single bits.

- Unlike other modes, synchronization errors of full block sizes are recoverable.

- Some preprocessing is possible before a block is seen; the previous ciphertext block can be encrypted.

- Encryptions are not parallelizable; however, decryption is parallelizable and has a random-access property.

CFB is like CBC in that:

- Links plaintext together so the ciphertext depends on the preceding plaintext

- Input to the block algorithm needs to be kicked off with an IV, and the IV need not be secret

- In CFB, the IV needs to be unique, in CBC it should be but it isn't required

CFB (specifically 8-bit CFB) is the mode of choice for encrypting streams of characters when each character has to be treated individually, as in a link between a terminal and a host. However- for high-speed synchronous systems where error propagation is intolerable and preprocessing is required, OFB is the better option.

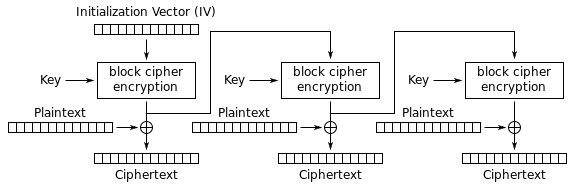
You can use CFB1-bit to do one bit at a time, but using one complete encryption with a block cipher on a single bit can be useless overhead. A stream cipher might be a better idea (plus, reducing the number of rounds to speed things up isn't recommended either!)

Plaintext is somewhat difficult for an attacker to manipulate; blocks can be removed from the beginning and end of the message, bits of the first block can be changed, and repetition allows some controlled changes.

**OFB - Output Feedback mode**

*A block mode implemented as a synchronous stream cipher*

The function of OFB is much like CFB, but instead of using the ciphertext block from the previously encrypted bits in it's shift register queue, it uses what the output function spits out *before* it gets XORed with the plaintext. It is sometimes called internal feedback since the feedback mechanism is *independent of the plaintext and ciphertext* (recall that in the definition of a synchronous stream cipher). Since it doesn't need the plaintext or ciphertext to do anything, most of the work can occur offline, before the plaintext message even exists. When the message finally arrives, it can be XORed with the output of the algorithm to produce the ciphertext.



- Because it has no chaining dependencies, OFB doesn't suffer error propagation.

- A ciphertext error affects only the corresponding bit of plaintext

- Synchronization error is unrecoverable. Like with synchronized stream ciphers you need an external mechanism to detect synch loss and resych.

- Plaintext patterns are concealed. Input to the block cipher is randomized.

- Ciphertext is the same size as the plaintext, not counting the IV

- Processing is possible before the message is seen; is not parallelizable; counter processing is parallelizable.

- Plaintext is very easy to manipulate, any change in ciphertext directly affects the plaintext.

- The IV should be unique, doesn't need to be secret

- More than one message can be encrypted with the same key, provided that a different IV is used.

----------

- It's recommended that OFB be used only when feedback size is same as block size (64-bit algorithm only in 64-bit OFB). Even though US Gov't authorizes other feedback sizes for DES, they should be avoided.

- OFB XORs a keystream with the text and the keystream will eventually repeat (the keystream's period).

- If it repeats using the same key then "there is no security"

- *"When the feedback size equals the block size, the block cipher acts as a permutation of m-bit values (where m is the block length) and the average cycle length is 2m-1. For a 64-bit block length, this is a very long number. When the feedback size n is less than the block length, the average cycle length drops to around 2m/2. For a 64-bit block cipher, this is only 232 - not long enough." - Schneier, Applied Cryptography*

|  |  |
| --- | --- |
|  | *OFB operation as a stream cipher*  OFB works slightly different when being used as a stream cipher First time in, internal state is given the IV (no previous key interaction) and is passed to the output function of the keystream generator without interacting with the key first. Simultaneously, that same content that was passed to the output function is incorporated with the key (in the next-state function) and placed in the new internal state (bumping the bits over like in CFB). Second time through the internal state contains the key-interacted previous internal state and part of the IV, and this is the first output containing key-involved stuff going to the output function. |

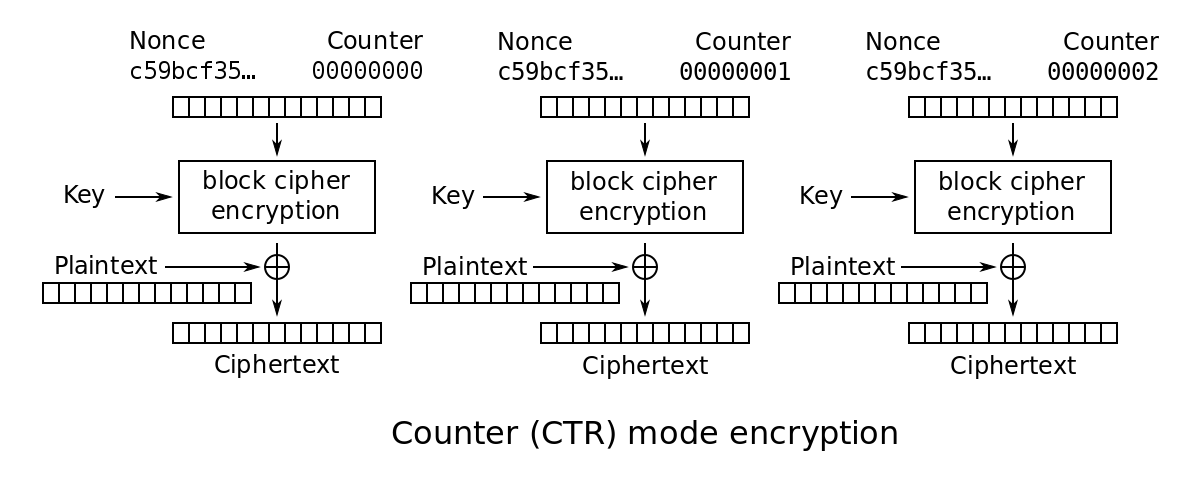
Ci = Pi ⨁ Si; Si = EK(Si-1)

------------------------------------------------

**Counter (CTR):**

Like OFB, but uses a nonce and incrementing counter instead of an IV and encryption output to fill the register.

After each block encryption, the counter increments by some constant (often one, but designed to allow for less predictable options). Random-sequence generators can be used as input to the block algorithm, rather than more obvious sequences



- solves the OFB mode problem of n-bit output where n is less than the block length.

- synchronization and error propagation characteristics of this mode are identical to those of OFB.

- both encryption and decryption can be performed using many threads at the same time.

If one bit of a plaintext or ciphertext message is damaged, only one corresponding output bit is damaged as well.

Thus, it is possible to use various correction algorithms to restore the previous value of damaged parts of received messages.

Sometimes called Segment Integer Counter (SIC) mode.

- This incrementing counter ensures each block is encrypted with a unique keystream (same content encrypts to different value), and provides the best performance.

Stream ciphers in counter mode have simple next-state functions and complicated output functions dependent on the key. The next-state function can be something as simple as a counter, adding one to the previous state.

With a counter mode stream cipher, it is possible to generate the ith key bit, ki, without first generating all the previous key bits. Simply set the counter manually to the ith internal state and generate the bit. This is useful to secure random-access data files; you can decrypt a specific block of data without decrypting the entire file.

As usual, a key should be changed after using it for encrypting an appropriate number of sent messages- but the CTR mode does allow for less frequent key changes. For example, AES in CTR should have a key change after about 264 plaintext blocks.

- 2001, NIST revised its list of approved modes by including AES as a block cipher and adding CTR mode in SP800-38A

- Jan 2010, NIST added XTS-AES in SP800-38E, CBC-CS/CTS in SP800-38A-Addendum " Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode

**Confidentiality Modes Lack Authenticity, Integrity - Enter Message Authentication Code**

- Traditional modes don't protect against accidental modification or malicious tampering,

- Detect with separate message authentication code such as CBC-MAC, or a digital signature.

- CBC-MAC is secure only for fixed-length messages

- Needed for dedicated integrity assurances, NIST approved HMAC, CMAC, and GMAC.

- HMAC (Keyed-Hash MAC) was approved in 2002 as FIPS 198

- CMAC (Cipher-based MAC) was released in 2005 under SP800-38B, aka OMAC (One-key MAC)

- GMAC (Galois MAC) was formalized in 2007 under SP800-38D,

Compositing a confidentiality mode with an authenticity mode proved to be difficult and error prone

Solution: combine confidentiality and data integrity into a single cryptographic primitive.

The modes are referred to as authenticated encryption, AE or "authenc".

- Examples of AE modes are

- CCM (Counter with CBC-MAC) - only defined for ciphers with a block length of 128 bits

- https://en.wikipedia.org/wiki/CCM\_mode

- GCM (Galois/Counter Mode),

- CWC,

- EAX,

- IAPM, and OCB.

**---------------------------------------------------------------------------------------------**

Types of block ciphers

- Commonly used

- Others

**Block cipher**

- Encrypts in blocks.

- Lower performance, More secure.

-

- Different modes of operation. Some can ensure message integrity.

-------------------

Look back at function f of DES. The expansion permutation and P-box perform diffusion; the S-boxes perform confusion. The expansion permutation and P-box are linear; the S-boxes are nonlinear. Each operation is pretty simple on its own; together they work pretty well.

DES also illustrates a few more principles of block cipher design.

- an iterated block cipher (taking a simple round function and iterating it multiple times)

- P-boxes (permutaion/confusion and S-boxes

Two-round DES isn’t very strong; it takes 5 rounds before all of the output bits are dependent on all of the input bits and all of the key bits. Sixteen-round DES is strong; 32-round DES is even stronger.

Cipher Feedback (CFB) mode is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. If we were using one-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted. At the receiving side, the ciphertext is decrypted and the extra bits in the block (i.e., everything above and beyond the one byte) are discarded. CFB mode generates a keystream based upon the previous ciphertext (the initial key comes from an Initialization Vector [IV]). In this mode, a single bit error in the ciphertext affects both this block and the following one.

Output Feedback (OFB) mode is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the same ciphertext block by using an internal feedback mechanism that generates the keystream independently of both the plaintext and ciphertext bitstreams. In OFB, a single bit error in ciphertext yields a single bit error in the decrypted plaintext.

Counter (CTR) mode is a relatively modern addition to block ciphers. Like CFB and OFB, CTR mode operates on the blocks as in a stream cipher; like ECB, CTR mode operates on the blocks independently. Unlike ECB, however, CTR uses different key inputs to different blocks so that two identical blocks of plaintext will not reuslt in the same ciphertext. Finally, each block of ciphertext has specific location within the encrypted message. CTR mode, then, allows blocks to be processed in parallel — thus offering performance advantages when parallel processing and multiple processors are available — but is not susceptible to ECB's brute-force, deletion, and insertion attacks.

**History of substitution ciphers (not needed)**

Simple substitution ciphers can be easily broken because the cipher does not hide the underlying frequencies of the different letters (only 26) of the plaintext

Polygram substitution ciphers group letters together. Playfair was used by the British during World War I, and encrypts pairs of letters together. The Hill cipher is another example, Huffman coding used as a cipher; this is an insecure polygram substitution cipher.

Polyalphabetic substitution ciphers were invented by Leon Battista in 1568. They were used by the Union army during the American Civil War. Despite the fact that they can be broken easily many commercial computer security products use ciphers of this form (WordPerfect used it) The Vigenère and Beaufort ciphers are also examples of polyalphabetic substitution ciphers.

Polyalphabetic substitution ciphers have multiple one-letter keys, each of which is used to encrypt one letter of the plaintext. The first key encrypts the first letter of the plaintext, the second key encrypts the second letter of the plaintext, and so on. After all the keys are used, the keys are recycled. If there were 20 one-letter keys, then every twentieth letter would be encrypted with the same key. This is called the ***period*** of the cipher. In classical cryptography, ciphers with longer periods were significantly harder to break than ciphers with short periods. There are computer techniques that can easily break substitution ciphers with very long periods.

The running-key cipher- sometimes called a book cipher—in which one text is used to encrypt another text, is another example of this sort of cipher. Even though this cipher has a period the length of the text, it can also be broken easily

**Transposition cipher** the plaintext remains the same, but the order of characters is shuffled around. In a simple columnar transposition cipher, the plaintext is written horizontally onto a piece of graph paper of fixed width and the ciphertext is read off vertically (see Figure 1.4). Decryption is a matter of writing the ciphertext vertically onto a piece of graph paper of identical width and then reading the plaintext off horizontally.

Cryptanalysis of these ciphers is discussed in [587,1475]. Since the letters of the ciphertext are the same as those of the plaintext, a frequency analysis on the ciphertext would reveal that each letter has approximately the same likelihood as in English. This gives a very good clue to a cryptanalyst, who can then use a variety of techniques to determine the right ordering of the letters to obtain the plaintext. Putting the ciphertext through a second transposition cipher greatly enhances security. There are even more complicated transposition ciphers, but computers can break almost all of them.

The German ADFGVX cipher, used during World War I, is a transposition cipher combined with a simple substitution. It was a very complex algorithm for its day but was broken by Georges Painvin, a French cryptanalyst [794].

Although many modern algorithms use transposition, it is troublesome because it requires a lot of memory and sometimes requires messages to be only certain lengths. Substitution is far more common.

--------------------------------------------

Feistel Networks

Most block algorithms are Feistel networks. This idea dates from the early 1970s [552,553]. Take a block of length n and divide it into two halves of length n/2: L and R. Of course, n must be even. You can define an iterated block cipher where the output of the ith round is determined from the output of the previous round:

Li = Ri - 1

Ri = Li - 1 ⨁ f(Ri - 1,Ki)

Ki is the subkey used in the ith round and f is an arbitrary round function.

You’ve seen this concept in DES, Lucifer, FEAL, Khufu, Khafre, LOKI, GOST, CAST, Blowfish, and others. Why is it such a big deal? The function is guaranteed to be reversible. Because XOR is used to combine the left half with the output of the round function, it is necessarily true that

Li - 1 ⨁ f(Ri - 1,Ki) ⨁ f(Ri - 1,Ki) = Li - 1

A cipher that uses this construction is guaranteed to be invertible as long as the inputs to f in each round can be reconstructed. It doesn’t matter what f is; f need not be invertible. We can design f to be as complicated as we please, and we don’t have to implement two different algorithms—one for encryption and another for decryption. The structure of a Feistel network takes care of all this automatically.

------------------------------------

- Public key vs Secret key/ Asymmetric vs Symmetric

Public Key, Asymmetric Key Pair - how it works

Applied Crypto for covering all the bases

- Sym for key exchange, Private for Comms,

- Hashing for integrity

- Shared key for key exchange

- Public key encryption for ultimate confidentiality, non-repudiation

- Public key signing for authenticity

Tweaks (where to put?)

- Key stretching/widening, whitening, nonces, salts, etc.

- PBKDF

- MAC, HMAC

Protocols and applications of Cryptography:

- IPSEC

- OCSP

Important concepts

- State of data: (data at rest, data in transit, data in place (use), etc.

Application: Transport encryption: SSL/TLS, SSH, IPSEC, WPA

Application: Data at rest: Full disk encryption, file-based, etc

- Latency and encryption

- Weakness in mis-application, mis-implementation

LAN manager and Microsoft

WEP and wireless, also

Problems and solutions:

- Perfect forward secrecy (PFS)

- Key management: certificates, escrow,

TRUST MODELS

4.1. PGP Web of Trust

4.2. Kerberos

4.3. Public Key Certificates and Certification Authorities

http://www.nku.edu/~christensen/diffusionandconfusion

http://www.garykessler.net/library/crypto.html

https://www.tutorialspoint.com/cryptography/cryptography\_quick\_guide.htm

https://www.tutorialspoint.com/cryptography/cryptosystems.htm

www.networksorcery.com/enp/data/encryption.htm

cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/

http://docstore.mik.ua/orelly/networking\_2ndEd/fire/appc\_02.htm

http://docstore.mik.ua/orelly/bookshelfs.html

https://en.wikipedia.org/wiki/Product\_cipher