UNIK4250 - Security in Distributed Systems

Tanusan Rajmohan - tanusanr@ulrik.uio.no

UNIVERSITY OF OSLO

Spring 2018

Contents

1 Learning outcome

After completing the course you will be able to:

- understand the threats against distributed systems and how to protect against them
- have a foundation for designing and developing secure distributed systems, and for evaluating the security of existing solutions
- have knowledge of standards, security protocols, technologies, principles, methods and cryptographic mechanisms applicable for securing modern distributed systems
- have knowledge of common mistakes leading to insecurities in distributed systems

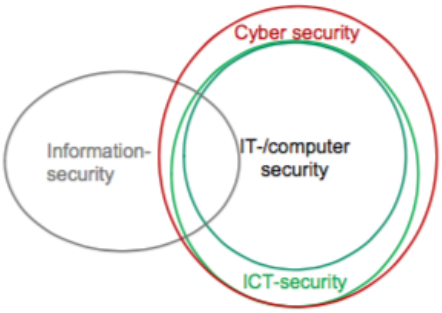
2 Introduction and Basic Concepts

What is a distributed system?

A **distributed system** is one in which components located at networked computers communicate and coordinate their actions only by passing messages. *Examples*: Spotify, Google, Airplane systems, Car systems, ATM, Facebook, MinID, Phones etc.

There is a lack of widely recognized definitions - different terms are often used as synonyms with slight (and partly claimed) differences.

While information security includes non-digital information (e.g., paper documents), we here say that cyber security ≈ ICT-security ≈ IT-/computer security

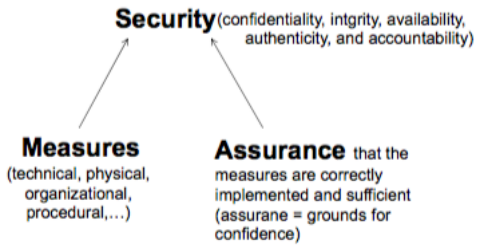


The traditional definition of computer security is to have preservation of the CIA objectives:

- **C**onfidentiality: preventing unauthorized disclosure of data
- **I**ntegrity: preventing unauthorized modification or destruction of data
  - Data integrity
  - System integrity
- **A**vailability: ensuring resources (i.e., services and data) are accessible and usable upon demand by authorized entities.

In addition to the CIA objectives, the following two objectives are also often included:

- Authenticity: ensuring genuineness
  - **Entity (user) authentication**: ensuring that the identity of an entity is as claimed
  - **Data origin authentication (message authentication**: ensuring that the source of the data/message is as claimed
- Accountability (supports nonrepudiation and traceability: ensuring that the actions of an entity can be traced uniquely to that entity (e.g., cannot later falsely deny sending/receiving a message)

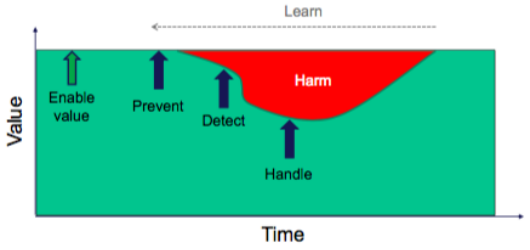


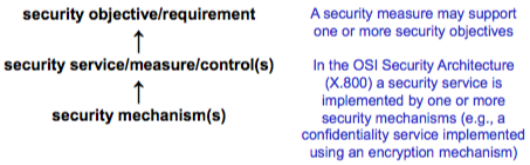
Security depends both on what security measures are implemented and the assurance that these measures are effective and sufficient.

The higher the risk, the higher the requirement for assurance - fit for purpose

The security requirements must be correct, and the security measures must fulfill the requirements under all relevant circumstances. Whether a system is secure or not is relative to its (correct) security requirements.

Security is about applying appropriate security measures to enable value and minimize harm (i.e., reduce risk to an acceptable level)





A security objective may be supported by one or more security measures.

A single security mechanism may support multiple security services and a single security service may support multiple security objectives

2.1 Threats, Vulnerabilities, and Attacks

Threat (The term has inconsistent usages):

- A potential event or circumstance that would violate security objectives (i.e., potentially causing adverse effects)
  - E.g., leakage of confidential data
- The capability and motivation/intent of the threat actor(s) (i.e., a measure of the threat level)
  - E.g., highly skilled and motivated

Vulnerability: Weakness that could allow a threat to cause harm

- e.g., lack of confidentiality protection

Attack: Deliberate attempt (i.e., a sequence of actions) to realize a threat (i.e., event) by exploiting vulnerabilities

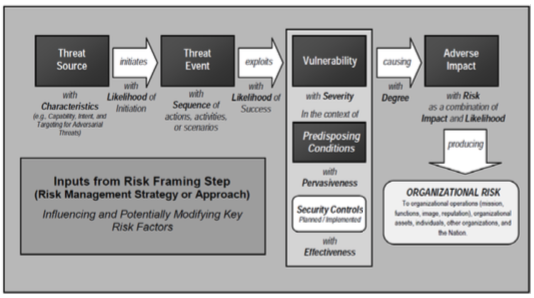
- e.g., eavesdropping on communication

2.1.1 Generally

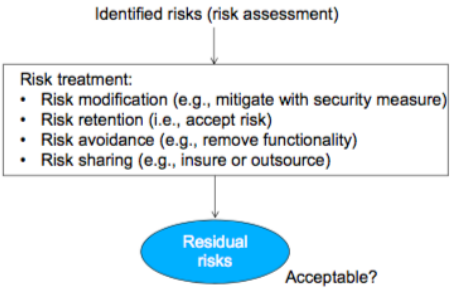
Risk = **Probability** * **Consequence**. For computer security, often use **Risk** = **Threat** * **Vulnerability** * **Impact**

- The aim is to provide acceptable risk
  - Zero risk requires one of the factors being zero...
  - How to handle risks with low probability but severe consequences?
- Risk is dynamic and dependent on many factors
  - Including a systems contextual value and environment
- In the context of risk assessment, probability is often subjective.
  - i.e., someone's belief about how likely something is to occur

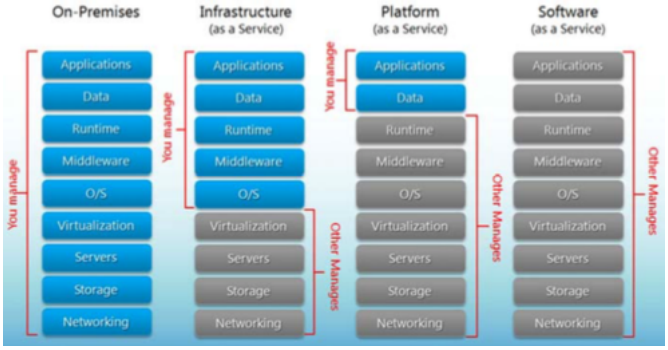
Risk model with key factors



Treating risks



Transferring responsibility does not necessarily transfer the consequences



Targeted attacks are more difficult to protect against than non-tareted attacks.
Typical phases of targeted attacks:

- Survey target (e.g., using open information sources, DNB, WHOIS, social networks, port scanning, social engineering) to identify vulnerabilities that can be exploited
- Deliver (e.g., utilizing spear phishing or a USB stick) exploit(s) to breach target
- Achieve attack objective(s) (affect)

Advanced Persistent Threats (APT) refers to the ability to use advanced attack techniques and persistence in time and in the face of adversity instead of moving on to weaker targets.

Threat modeling is useful to ensure that the security requirements of a system are fulfilled and to identify missing requirements.
STRIDE (as used in Microsoft SDL) provides one approach:

- **S**poofing (violates authenticity)
- **T**ampering (violates integrity
- **R**epudiation (violates accountability/non-repudiation)
- **I**formation disclosure (violates confidentiality)
- **D**enial-of-service (violates availability)
- **E**levation of privilege (violates authorization)

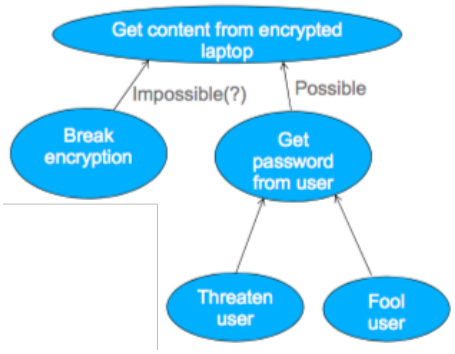
To help identify potential threats the system is modeled (using data flow diagrams). Identified threats needs to be mitigated, eliminated, or accepted.
An **attack tree** shows the goal of the attacker (threat) as the root node and the ways of achieving that goal as child nodes.

Child nodes may be *OR* nodes (default) or *AND* nodes.

May assign a value to each node, e.g.,

- Possible / impossible
- Cost
- Probability
- Required expertise/tools

Treating risks



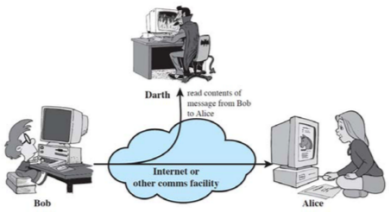
Information is considered to exist in three possible states: **Storage**, **Transmission** & **Processing**(use)

2.2 Active and passive attacks

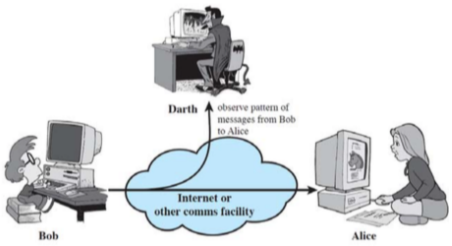
A **passive attack** is one where the attack eavesdrops or observes but does not modify the message stream in any way. This is *difficult to detect*, and we *aim to prevent* this.

An **active attack** is one where the attacker transmits messages, replays old messages, modifies messages, and/or delete selected messages. This is *difficult to prevent completely*, but we *aim to detect and recover*. ***One can use an active attack to enable a passive attack.***

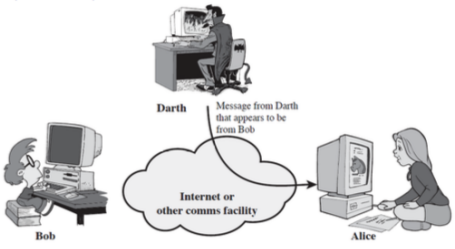
Release of message content (interception / information disclosure) (passive attack)



Traffic analysis (passive attack)

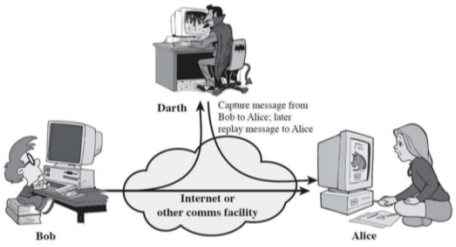


Spoofing/masquerade/fabrication/impersonation (active attack)



Man-in-the middle attacks may be seen as a special case of masquerade (where Darth masquerades as Bob to Alice and as Alice to Bob)

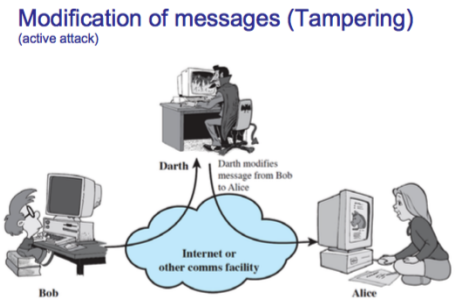
Replay (active attack)



Denial-of-Service attacks (active attacks)
Causing DoS by for example:

- Suppressing selected messages etc.
- depleting, disabling or overloading (network or end-system) resources

Includes distributed denial-of-service (DDoS) attacks
Overly restrictive or resource expensive security mechanisms may contribute to DoS



These attacks mainly focus at the network communication, however, end-systems and users may be subject to attack as well. If the attacker can observe or tamper with the data before or after it is transmitted it does not help much that the communication is secure.

3 Symmetric Encryption

3.1 Terminology

- Plaintext (P) - original message / data
- Ciphertext (C) - coded message / data
- Cipher - algorithm for transforming plaintext to chipertext or chipertext to plaintext
- Key (K) - info used in cipher known only to sender / receiver
- Encipher (encrypt) (E) - converting plaintext to ciphertext
- Decipher (decrypt) (D) - recovering plaintext from ciphertext
- Cryptography - study of encryption principles / methods
- Cryptanalysis (code breaking) - study of principles / methods of recovering key or deciphering ciphertext *without* knowing key
- Cryptology - field of both cryptography and cryptanalysis

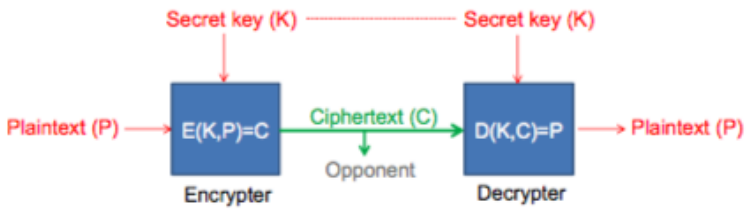
3.2 Ciphers

Main cryptographic cipher types:

- Symmetric (one key, i.e., shared secret key) → Stream, Block

- Asymmetric (two keys, i.e., public/private key)

Model of symmetric cryptosystem
(i.e., the sender and receiver share a secret key)



The secret key (used for both encryption and decryption) must be distributed over a secure channel, while the encryption algorithm is assumed to be publicly known-

3.2.1 The one-time pad (the Vernam cipher)

$C = E(K,P) = K \oplus P$
$P = D(K,C) = K \oplus C$

\+ provides *perfect secrecy* (and is fast)
\- Requires a random *one-time key as long as the plaintext*

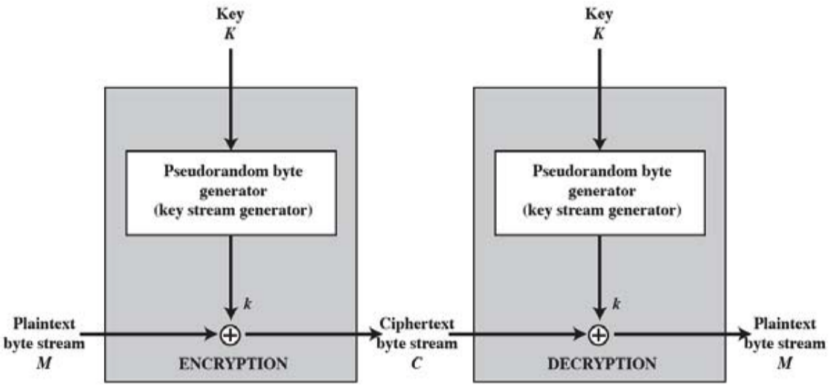
Notion of cryptographic security

Unconditional security - The system cannot be broken even with infinite computational resources

Computational security - It is impossible to break the system in practice due to the computational resources required by the best known algorithms for breaking the system

Provable security - Breaking the system is equivalent to solving a difficult problem (factoring, discrete logarithm)

Stream ciphers use pseudo-random number generators to generate a keystream that is xored with the plaintext/ciphertext



Stream ciphers can be realized using a blockcipher in a "stream mode" or by dedicated stream ciphers (e.g., RC4)

Designed to be efficient to implement in software (as opposed to typical stream ciphers intended to be implemented in HW)

Has been widely used, including:

- SSL / TLS
  - Sees less support due to attack demonstrated in 2013
    - Enabled by biases in the start of the RC4 keystream
    - The attack is not very practical yet but...
- WEP / WPA
  - The attack on TLS with RC4 also applies to WPA / TKIP
  - The vulneravilities in WEP were not due to RC4 itself, but the way it was used

3.2.2 RC4

Start with a key K of length ≤ 256:

for i = 0 to 255 do:

S[i] = i

T[i] = K[i mod keylength]

S is now initialized with all numbers from 0-255. T is initialized with K (where K is repeated if necessary to generate T of length 256).

Use T to shuffle S:

j = 0

for i = 0 to 255 do:

j = (j + S[i] + T[i]) mod 256

swap(S[i], S[j])

S forms the internal state of the cipher.

RC4 Keystream generation - encryption/decryption

For each byte plaintext/ciphertext: shuffle S and generate keystream value that is XORed with plaintext/ciphertext byte:

i = j = 0

for each plaintext byte P_i do

i = (i + 1) (mod 256)

j = (j + S[i]) (mod 256)

swap(S[i], S[j])

t = (S[i] + S[j]) (mod 256)

$C_i = P_i \bigoplus S[t]$ (Decryption: $P_i = C_i \bigoplus S[t]$)

Small deviations or wrong assumptions can cause insecurity

(we here use the One-Time Pad for illustration)

Stream ciphers do not provide integrity:

- E.g., OTP: D(K, C $\bigoplus$ i) = P $\bigoplus$ i (i.e., changes to C are not detected and results in predictable changes to P)
- Lesson: only depend on cryptographic mechanism for its intended purpose(s)!

A stream cipher is insecure if the same keystream is used twice:

- E.g., OTP: C1 $\bigoplus$ C2 = (K $\bigoplus$ P1) $\bigoplus$ (K $\bigoplus$ P2)= P1 $\bigoplus$ P2
- Lesson: Only use keys for their intended purpose and duration!

The keystream must be completely unpredictable!

- May otherwise become vulnerable to known plaintext attacks

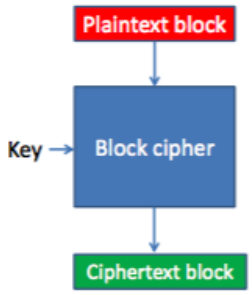
Random numbers

- Many applications of random numbers in cryptography and security (e.g., key generation, keystreams, nonces,..)
- Critical that these values are statistically random (uniform distribution and independence) and that future values are unpredictable.
  - Improper random number generation is a common source of security vulnerabilities.
- Often use a Pseudorandom Number Generator (PRNG):
  - Deterministic sequences of outputs, given a seed (e.g., the secret key) as input
  - Such pseudorandom numbers are not truly random but can pass as many tests of randomness
  - May be based on e.g., symmetric / asymmetric ciphers or hash functions

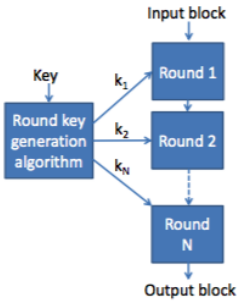
3.3 Block ciphers

Symmetric block ciphers maps a fixed size input block to a fixed size output block

- Block size: Number of bits taken as input/output (AES: 128 bits)
- Key size: Larger keys are more secure but may reduce speed (AES: 128, 192 or 256 bits)
- Block ciphers can be used in different modes of operation



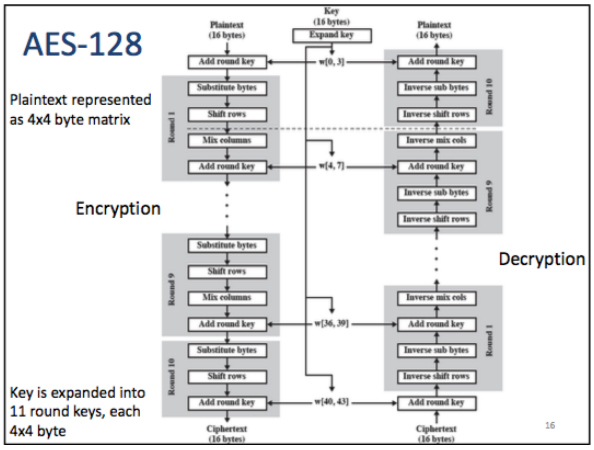
Block ciphers typically iterate a weaker round function



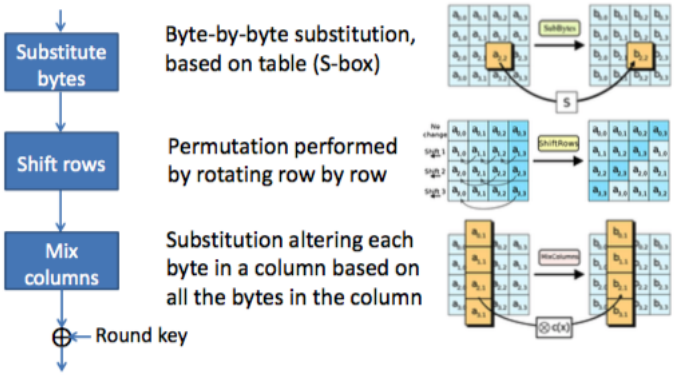
- The key is expanded into a sequence of round keys
- AES-128: 10 rounds
- AES-192: 12 rounds
- AES-256: 14 rounds
- DES: 16 rounds

3.4 Advanced Encryption Standard

AES uses the Rijndael block cipher



Rijndael/AES round function uses four invertible operations



AES Instruction Set and Intel's AES-NI

- Extensions to x86 instruction set providing hardware support for AES
- Provided by Intel and AMD, used by many libraries and applications
- Hardware support for AES is also available on other platforms

3.5 Data Encryption Standard (DES)

Issued as a standard by NIST in 1997

- Block size is 64 bits
- Key is 56 bits - too short today!

- Variation of a Feistel network

DES is expired and should no longer be used! (use AES instead)

Average Time Required for Exhaustive Key Search

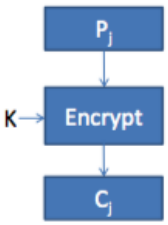
Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Block Cipher Modes of Operation specifies how to use symmetric block ciphers for practical applications

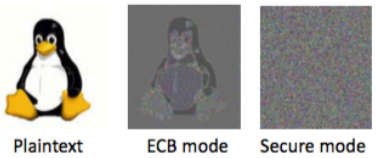
- NIST SP 800-38A specifies five modes of operation:
  - ECB
  - CBC
  - CFB
  - OFB
  - CTR

    ↑ Confidentiality modes (do not ensure integrity)
- SPs 800-38 B - F specifies additional modes of operation, including authenticated encryption modes such as GCM and modes intended for storage encryption

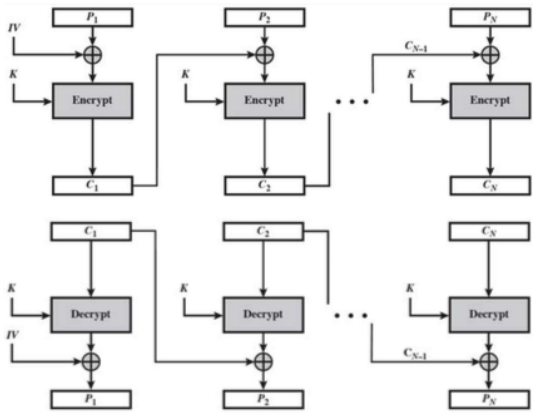
Using Electronic Codebook (ECB) mode, each block is encrypted/decrypted independently



Identical plaintext blocks (encrypted with the same key) result in identical ciphertext blocks - may be insecure

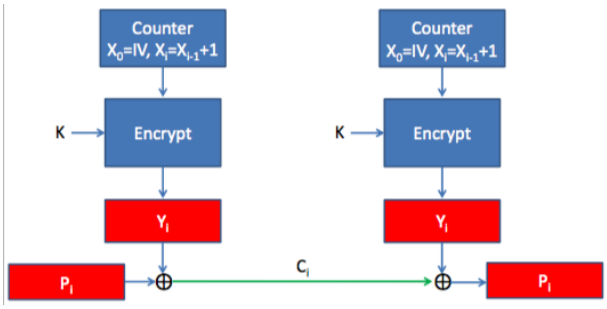


3.6 Cipher Block Chaining (CBC) mode



- The IV must be unpredictable (but does not need to be secret)
- Does not provide integrity protection
- Correct decryption depends on correct receipt of the corresponding and previous ciphertext block
- Can not be parallelized well (decryption can to some extent)
- Needs to pad last block if the plaintext is not a multiple of the block size (can be avoided using ciphertext stealing)

3.7 Counter (CTR) mode



- Hardware and software efficiency:
  - Encryption/decryption can be done in parallel
  - Preprocessing - The underlying encryption algorithm does not depend on plaintext or ciphertext input
- Random access to ciphertext/plaintext blocks
- Only requires implementation of the encryption algorithm and not the decryption algorithm
- Does not provide integrity protection

3.8 Galois Counter Mode (GCM)

- Mode of operation that combines encryption and authentication (i.e., authenticated encryption)
- To be used with 128-bit clock cipher (typically AES)
- Uses a variation of CTR mode encryption for confidentiality
- Uses a keyed hash function to create the authentication tag
- Suitable for use with e.g., IPSEC and TLS

3.9 Cryptanalysis

- Objective is to find the key or some unknown plaintext
- Brute-force attack
  - On average half the keys must be tried
  - Must be able to recognize valid plaintext
  - Mitigated by sufficient key length
- Cryptanalytic attack → Weaknesses may result in much less resources/effort being required than for a brute-force attack

3.9.1 Cryptanalytic attacks

- **Ciphertext only** - only know algorithm and ciphertext
- **Know plaintext** - know/suspect plaintext and ciphertext
- **Chosen plaintext** - attacker select plaintext and obtain the corresponding ciphertext
- **Chosen ciphertext** - attacker select ciphertext and obtain the corresponding plaintext

4 Hash functions, message authentication, and asymmetric cryptography

4.1 Cryptographic hash functions

Hash functions have many applications, e.g., integrity checks, digital signatures, message authentication codes, key derivation, pseudorandom number generators, password protection, etc.

Practical requirements:

- Can be applied to data of "any" size
- Produces a fixed-length output
- H(x) is relatively easy to compute

Data object (e.g., message) $\longrightarrow$ Hash function $\longrightarrow$ Fixed length hash value (digest/fingerprint/imprint)

Requirements for cryptographic hash functions

- **Preimage resistance** (one-way): For any given y, it is computationally infeasible to find x such that H(x) = y
- **Second preimage resistance** (weak collision resistance): For any given x, it is computationally infeasible to find y $\neq$ x where H(y) = (x)
- **(Strong) collision resistance**: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y)

Bryte-force preimage and second preimage attacks require 2^n effort, while a collision attack only requires $2^{n/2}$ effort (i.e., due to the birthday paradox) for Hash code of length n.

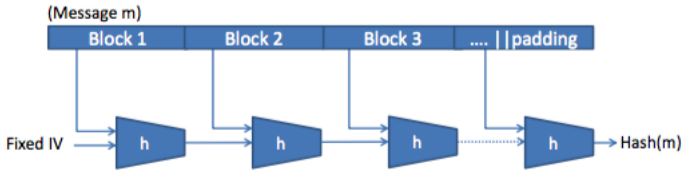
Common cryptographic hash functions

Secure Hash Standard (SHA):

- SHA-1 (1995): 160 bit digest
  - 2011: Deprecated by NIST
  - 2013: 2^{61} theoretical collision attack estimated
  - 2017: First known collision for full SHA-1 was found ($2^{61,1}$ effort)
- **SHA-2** (2002): 224 (2008), 256, 384, or 512 bit digests
- **SHA-3** (2015): 224, 256, 384, or 512 bit digests

SHA-1 and SHA-2 both make iterative use of a compression function

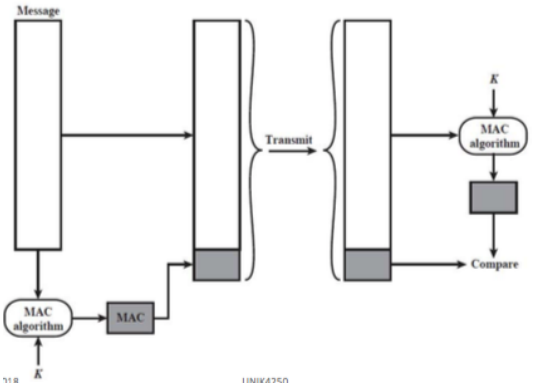
(Merkle-Damgard construction)



Padding: Single 1 followed by the required number of 0's and the message length
(e.g., 10000 || 64/128-bit message length field)

Block size: 512 (1024 for SHA-384/512)

4.2 Message Authentication Codes (MAC)

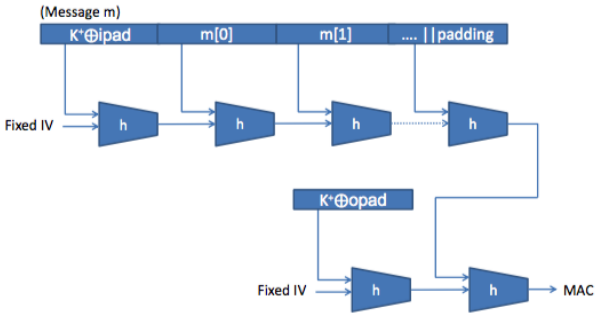


Message authentication serves to protect the integrity of a message and authenticate its originator
Alternative mechanisms:

- Message authentication code (MAC)
- Digital signature
- Authenticated encryption (e.g., GCM)

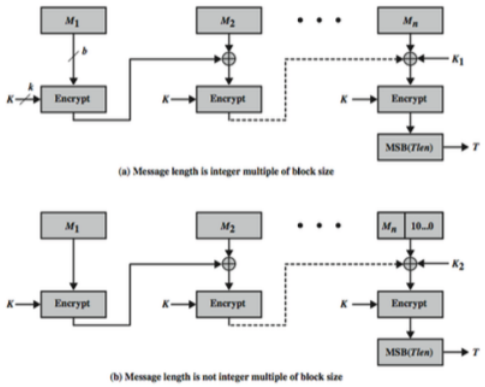
Digital signatures can also provide non-repudiation
MAC are based on the use of a shared secret key (i.e., do not provide non-repudiation)

HMAC (e.g., using SHA-1/2)



HMAC(K,M) = Hash[(K⁺⊕opad) ‖ Hash[(K⁺⊕ipad) ‖ M]] (where K⁺ is K extended with 0s)

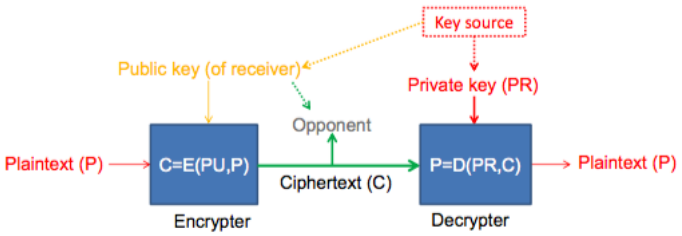
CMAC (Cipher-based Message Authentication Code)



CMAC is a more secure variation of CBC-MAC (which in its basic form encrypts the message in CBC mode and uses the last block as the tag)

4.3 Asymmetric cryptography

Model of asymmetric cryptosystem (used for encryption/decryption)



The authenticity (and validity) of the public key must be assured.
Because asymmetric cryptography is slow it is typically used to encrypt a secret key, that is used to encrypt the data using symmetric encryption.

4.3.1 RSA

Published in 1977 by Rivest, Shamir, and Adleman. It is widely used for both key-exchange (encryption) and digital signatures. RSA makes use of the efficiency of modular exponentiation and that it is computationally infeasable to compute the corresponding root (i.e., given message M, it is easy to compute $C = M^e (mod n)$, but har to find the e-th root of C). The security of RSA is based on the difficulty of the factoring problem.

4.3.2 RSA ("textbook version")

Generate two large random primes *p* and *q*, and compute

$n = p * q$

Choose integers *e* and *d* such that:

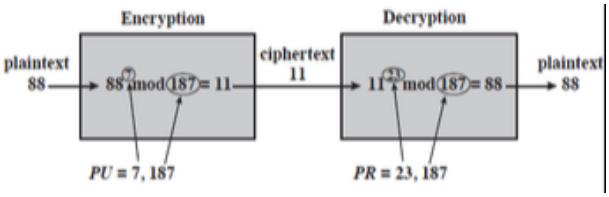
e*d = 1 (mod $(p$-1)$(q$-1))

The resulting keypair is:

Public key = (e,n) and Private key = (d,n)

Encryption:
C = M^e (mod n)

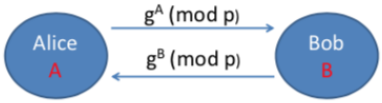
Decryption:
M = D^d (mod n)



4.3.3 Diffie-Helleman (DH) key exchange

It is computationally infeasible for an eacesdropper to find the private keys/exponents A and B (discrete logarithm problem)

Authentication of the public keys ($g^A and g^B$ (mod p)) is required to prevent man-in-the-middle attacks! (i.e., anonymous DH is insecure)



$K_{AB} = (g^A)^B \pmod p = (g^B)^A \pmod p = g^{AB} \pmod p$

4.3.4 Security levels

Symmetric	80	112	128	192	256
RSA & DH	1024	2048	3072	7680	15360
ECC	161	224	256	384	512

Comparable strengths of keys in bits (NIST sp800-57, part1_rev3)

- A 1024-bit RSA key provides insufficient security for general use
- Successful quantum computing attacks (e.g., base on Shor's algorithm) would render these asymmetric cryptosystems ineffective
- NSA, August 2015: "Unfortunately, the growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, which has made it clear that elliptic curve cryptography is not the long term solution many once hoped it would be"

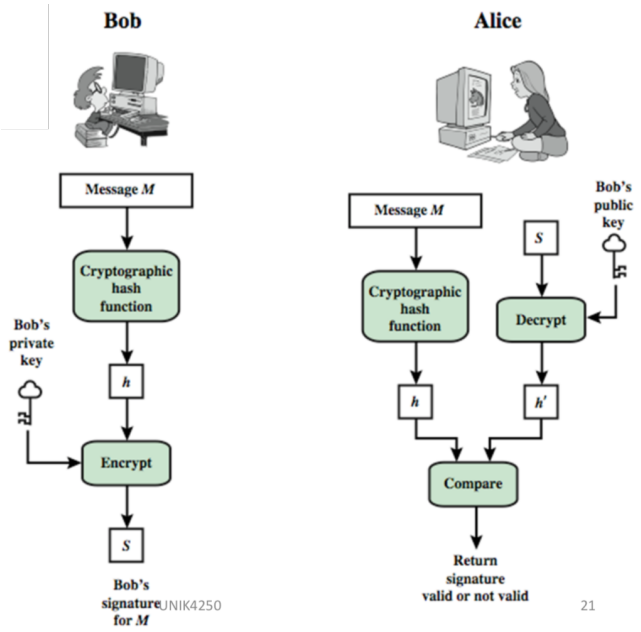
Given quantum computers successfully implementing Shors algorithm, all current common asymmetric ciphers will be rendered useless.

- Some believe that large-scale quantum computers may be realized within the next 20 years.

- There are ongoing efforts to establish quantum-safe asymmetric primitives, e.g.:
  - NIST Post-quantum crypto project
  - ETSI Quantum-safe cryptography

4.4 Digital signatures

Digital signature generation and verification



5 Key management and entity authentication

Key management involves generation, storage, distribution, use and destruction of keys.

A cryptographic system is no stronger than its key management

Keys should be periodically changed

Suggested cryptoperiods for key types, NIST SP 800-57 (part1, rev.4, 2016)

Longer or shorter cryptoperiods may be warranted

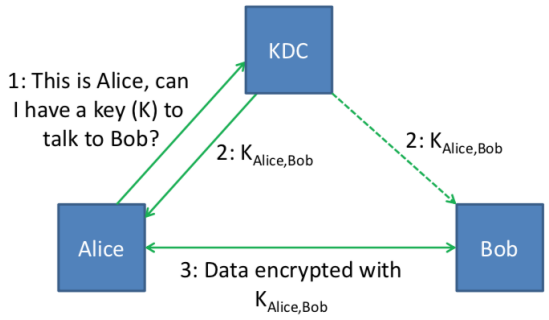
8.2.2018

Key Type	Crytoperiod	
	Originator-Usage Period (OUP)	Recipient-Usage Period
1. Private Signature Key	1 to 3 years	–
2. Public Signature-Verification Key	Several years (depends on key size)	
3. Symmetric Authentication Key	≤ 2 years	≤ OUP + 3 years
4. Private Authentication Key	1 to 2 years	
5. Public Authentication Key	1 to 2 years	
6. Symmetric Data Encryption Keys	≤ 2 years	≤ OUP + 3 years
7. Symmetric Key Wrapping Key	≤ 2 years	≤ OUP + 3 years
8. Symmetric RBG Keys	See [SP800-90]	–
9. Symmetric Master Key	About 1 year	–
10. Private Key Transport Key	≤ 2 years[16]	
11. Public Key Transport Key	1 to 2 years	
12. Symmetric Key Agreement Key	1 to 2 years[17]	
13. Private Static Key Agreement Key	1 to 2 years[18]	
14. Public Static Key Agreement Key	1 to 2 years	
15. Private Ephemeral Key Agreement Key	One key-agreement transaction	
16. Public Ephemeral Key Agreement Key	One key-agreement transaction	

5.1 Key distribution can be achieved in many ways

- Physical delivery (or secure out-of-band delivery by other means)
- A new key may be encrypted using an existing shared key
  - Requires an existing shared key, and the compromise of one key would compromise all following keys
- Encrypt the new key using the receiver's public key
  - Must be able to verify the authenticity of the receiver's public key
- Use authenticated Diffie-Hellman key exchange
  - Must be able to verify the authenticity of the public key(s)
- If both have a secure channel to a trusted third party, this third party could deliver a key to both

Basic idea of a third party Key Distribution Center (KDC): Alice wants to communicate with Bob but they don't share a secret key, instead both share a secret key with a trusted KDC



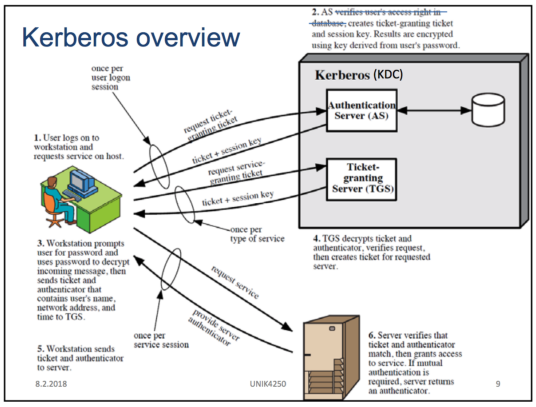
5.1.1 Kerberos provides entity (e.g., user or server) authentication and key distribution using a KDC approach

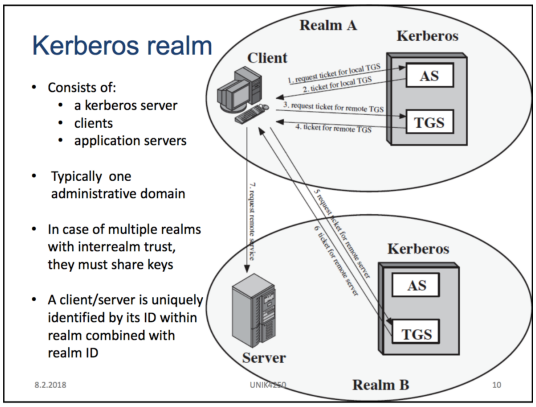
- Kerberos itself does not enforce authorization, but may provide a basis to do so
- Primarily based on symmetric cryptography - but public key extensions are available
- Version 4 is beyond end-of-life, should use an up-to-date implementation of Kerberos 5
- Supported by many products (E.g., default authentication in Windows domains)

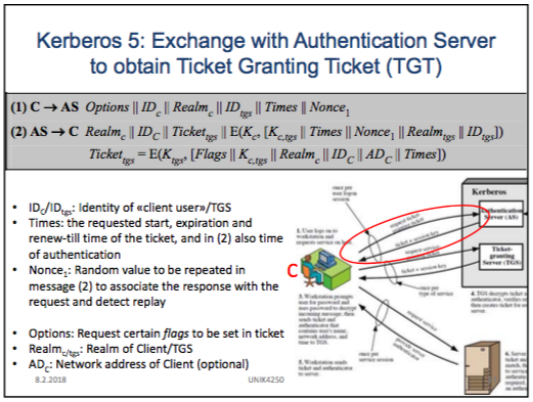
5.1.2 Kerberos is a third-party authentication and key distribution scheme

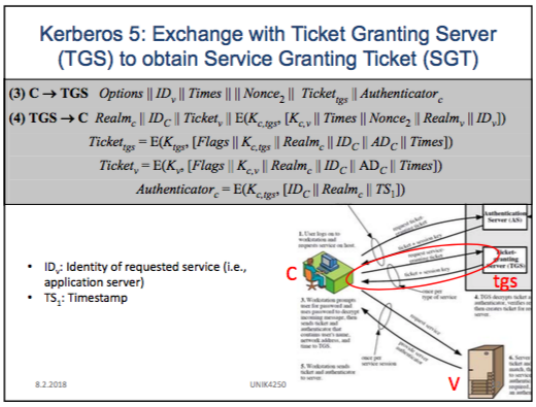
- Authentication Server (AS)
  - User initially authenticate to AS
  - AS issues Ticket Granting ticket (TGT)
- Ticket Granting Server (TGS)
  - Users use TGT to access TGS
  - Users request TGS for access to other services
  - TGS issues Service Granting Ticket (SGT) to be used to access requested service Oten both servers/services are provided by the same server

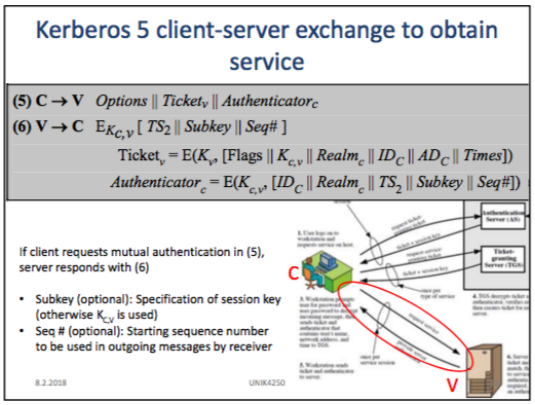
5.1.3 Kerberos











Delegation can be used to enable services to act on behalf of a principal (e.g., user)

Proxy
The PROXIABLE flag tells the Ticket-granting server that it is OK to issue a new ticket (but not a TGT) with a different network address based on this TGT

- Set if requested by the client on initial authentication
- Can be used by client to allow a service to perform an operation on its behalf (i.e., taking on the identity of the client)
- The use of the "proxy ticket" may be restricted to a particular purpose

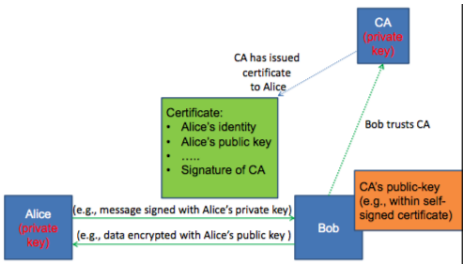
Forwarded TGT
An instance of a proxy, but the FORWARDABLE flag tells the TGS that it is OK also to issue a new TGT with a different network address

5.2 Management of asymmetric keys

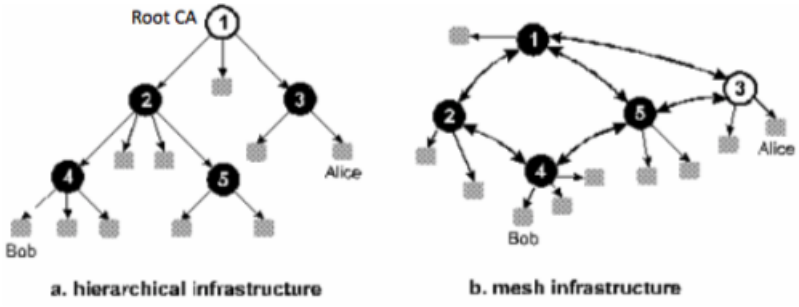
5.2.1 Key management and asymmetric cryptography

- Public-key encryption can be used to distribute secret keys (Requires authenticity of public-key)
- Public-key infrastructures
  - Certificates enable the authenticity of public-keys to be verified, binding a public-key and the ID of its owner
  - Certificates must be issued by a trusted Certificate Authority (CA)

Public-key certificates are issued to authenticate the association between a public-key and an identity, and assumes that the issuing CA is trusted



PKI architectures - a single CA is not practical for large communities



Relying parties typically know the public key of the Root CA and can verify certificate paths to the root CA

Cross certification between "independent" CAs. Relying parties know the public key of one (or more) CA used to verify certificate paths

Reality may be a bit more ugly ..

Hundreds (!) of root CAs are trusted by default by common OSs/browsers

A trusted root CA can typically issue certificates for *any* site

There are several examples of trusted CAs incorrectly issuing certificates

- DigiNotar (2011) - security breach resulted in illicit issuing of certificates used to perform MiTM attacks against Iranian Gmail users
- Comodo (2011) – hackers managed to issue nine illegitimate certificates for sites such as Google, Yahoo, Mozilla, Skype, and Microsoft

- Trustwave (2012) – issued a **CA certificate** to a customer who used it to decrypt traffic. First became known when revoked by Trustwave, announcing they would not issue such certificates again
- Turktrust (2013) – two certificates accidentally marked as CA certificates, one of the customers used it to lcoally intercept encrypted traffic. Detected when Chrome's certificate pinning reported unexpected certificate for google.com.
- India's National Informatics Centre (2014) – "compromised issuance process" resulted in fake certificates (including google and yahoo domains)
- Many incidents due to poor certificate issuing processes, e.g., Thawte certificate for live.com obtained by user with e-mail: sslcertficates@live.com

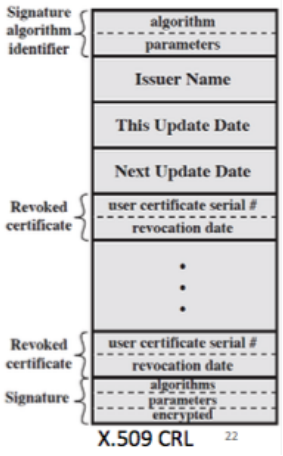
5.2.2 Certificate Transparency CT

- Publicly accessible, append only log issued certificates
- Everyone can monitor log entries to detect unauthorized certificates
  - To be effective, someone (e.g., domain owners,..) must actually do this
- Certificates that are not logged should not be trusted
  - Log inclusion proof can be included in e.g., certificate or TLS handshake
  - May require certificates to be present in multiple logs
- Chrome will require CT for all newly issued, publicly trusted certificates from April 2018

Revocation - withdrawing certificates (e.g., due to private key being compromised)

Alternative approaches:

- Certificate Revocation List (CRL)
  - Centralized or replicated
- Online certificate verification
  - Online Certificate Status Protocol (OCSP)
  - Server-based Certificate Validation Protocol (SCVP)
- Fast expiration (e.g., certificate lifetime between 5 minutes and 24 hours)



In Practice, revocation status is often not properly cheched

What if CRL or online status can't be obtained?

- Display warning/error (hard fail)
- Or, assume the certificate is not revoked ("soft fail")

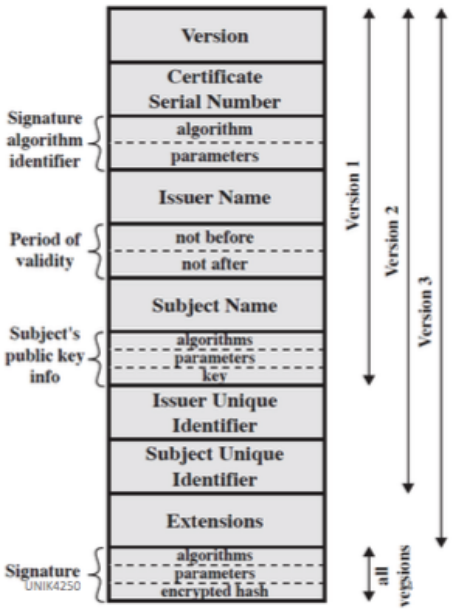
Online certificate verification may increase delay and reduce privacy

- Can be mitigated through OCSP stapling - OCSP status presented together with certificate
- Should be combined with OCSP Must-Staple flag in the certificate

5.2.3 X.509 certificates

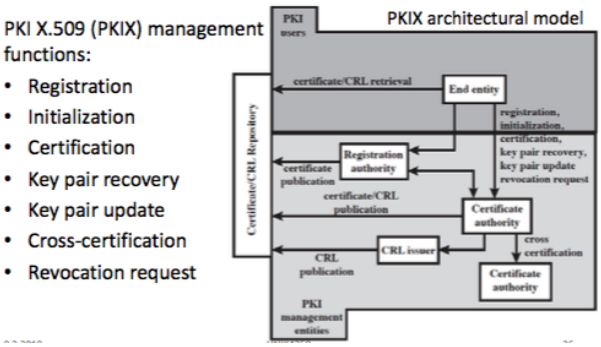
Various Extensions to X-509 certificates can be used, including:

- **Certificate policies**: Rules that indicate the applicability of a certificate to a particular community and/or class of application
- **Key usage**: Restricts the purposes for which the public key may be used (digital signature, non-repudiation, key encryption, data encryption, key agreement, CA signature verification on certificates and/or CRLs)
- **Private-key usage period**: Indicates the period of use of the private key corresponding to the public key
- **Subject/Issuer alternative name** or **Subject directory attributes**
- **Certification path constraints** - restrict the type of certificates that may be issued by the subject or that may occur subsequently in a certification chain



(The authenticity of a certificate is ensured by the CAs signature (i.e., certificates do not need to be specially protected)

Public-Key Infrastructure: The set of HW, SW, people, policies and procedures needed to create, manage, store, distribute and revoke digital certificates



6 Transport Layer Security

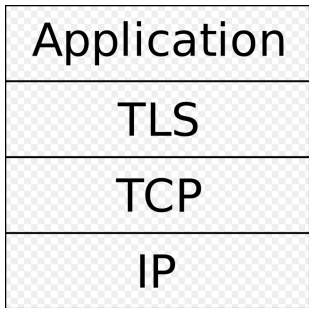
6.1 TLS - Transport Layer Security

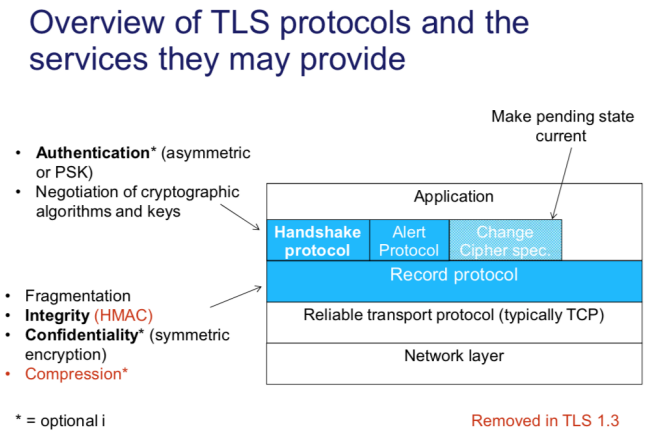
Secure Socket Layer (SSL) derives its name from the Berkeley *sockets* API, its descendant was standardized as Transport Layer Security (TLS)

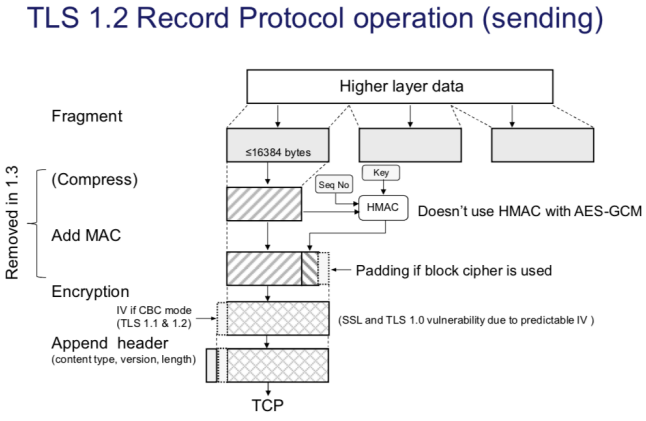
- SSLv1: Never released (Netscape)
- SSLv2: Netscape Navigator (1995) - Multiple vulnerabilities, insecure
- PCT (Private Communication Technology, Microsoft)
- SSLv3: Netscape (1996)
- TLS: IETF (v1.0 1999, v1.1 2006, v1.2 2008) - Ongoing standadization of v1.3

TLS resides at, or arguably on top of the transport layer
Implications:

- Not dependent on being implemented withing OS stack → Widespread deployment, many implementations
- Applications must use TLS API instead of TCP API → Not transparent to applications
- The underlying reliable transport makes TLS simpler
- Lower layer protocols (IP, TCP) are not protected
- Easy NAT traversal, etc.







An implication of utilizing TCP for providing a reliable transport is that the connection must be broken if the TLS MAC verification fails

- If TLC MAC verification fails, there is no way for TLS to make TCP retransmit the segment
- TCP only provides a non-cryptographic checksum that can be easily bypassed by an attacker
- Means a TLS connection can be broken by modifying (or inserting) a single packet

A TLS session may be shared by multiple connections, thereby avoiding the overhead of establishing each connection from scratch

Connection

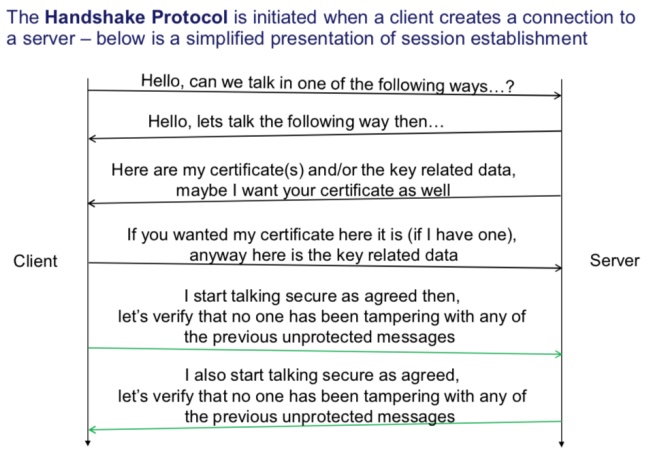
- A transient, application-to-application, communication channel
- Associated with one session

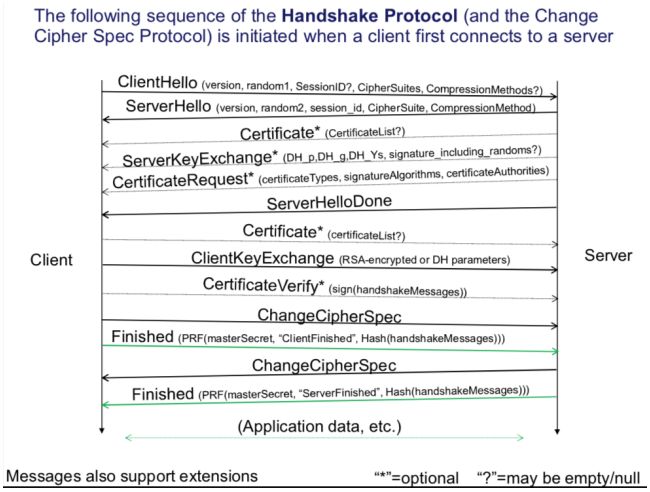
Session

- An association between client & server
- Created by the Handshake Protocol
- May be shared by multiple connections

The Handsake Protocol is Used to negotiate a session consiting of

- Session identifier - arbitrary byte sequence
- Peer certificate - peer's C509v3 certificate (may be null)
- Compression method (typically none)
- Chiper spec (PRF, key exchange (authentication) algorithm, bulk encryption algorithm, MAC hash function, ...)
- Master secret - 48 byte shared secret
- Is resumable - whether session can be used to initiate new connections

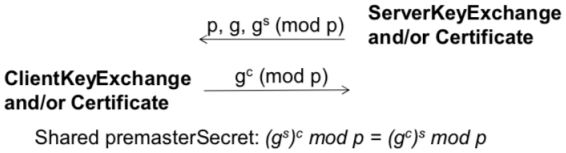




Key Exchange (i.e., establishing the premasterSecret) **RSA**: Client generates 48-byte premasterSecret and sends it encrypted with the servers public key in ClientKeyExchange

Diffie-Hellman: Client/server has the following alternatives:

- Send certificate containing the public DH parameters
- Send public values in (unsigned) key exchange messages (Anonymous DH - vulnerable to MitM)
- Server sends signed DH public value and parameters, clients send its public value (and optionally authenticates using CertificateVerify) - Can provide perfect forward secrecy



Perfect Forward secrecy (PFS)

1. Eavesdropper records conversation
2. Later gains access to the (long term) secret
3. If it is still impossible for the eavesdropper to decrypt the conversation, the protocol is said to provide PFS
    - Also provides protection against passive attackers after the long term key has been compromised.

Supported in TLS through *ephemeral Diffie-Hellman (DHE)*, i.e., where the DH parameters are signed with a certificate so that the parameters can be changed each time.

Creating the master secret and generating key material

masterSecret = PRF(premasterSecret, "master secret", ClientHello.random + ServerHello.random)

Key material can then be generated by repeating PRF(masterSecret, "key expansion", ServerlHello.random + ClientHello.random) until sufficient material has been generated

RFC7525: Recommendations for secure use of TLS or DTLS (v1.2)

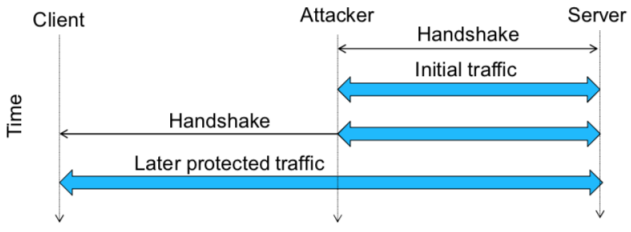
- Never use/support SSL. (D)TLS 1.2 should be used.
- Use strict TLS (to protect against TLS stripping)
- TLS compression should be disabled (to protect against compression attacks)
- Use a cipher suite with forward secrecy (i.e., ephemeral DH using at least 2048-bit keys), AES-GCM 128/256, and SHA 256/384
- The host name and certificate path must both be properly validated

Some main changes in the current TLS 1.3 draft specification

- Only supports authenticated encryption modes such as AES-GCM (more specifically Authenticated Encryption with Assosiated Data is used)
- Static RSA and DH have been removed - all public-key based key exchange now provide PFS
- All data following the initial hello messages are encrypted
- A 1-RTT mode for establishment new connections
- A 0-RTT mode when resuming previous connections - should not be used by default due to security weaknesses
- Session renegotiation is prohibited, except that the server may request client authentication at any time - authentication binds to initialization handshake

The handshake process may be repeated to perform renegotiation (forbidden in TLS 1.3)

Because there ie no secure binding between the initial handshake and the renegotiation, applications that don't differentiate between the intial and later traffic (time-of-check vs. time-of-use) may be vulnerable to a type of restricted MitM attack.



RFC5746 (2010) defines an extension to provide secure renegotiation

The Change Cipher Spec Protocol is used to signal a switch from the current to the pending connection state

- Consist of a fixed one byte message (encrypted and compressed under the current connection state)
- The sender of the ChangeCipherSpec message makes the pending write state current after sending the message
- The receiver of the ChangeCipherSpec message makes the pending read state current after receiving the message
- Resets the corresponding sequence numbers

The Alert Protocol is used to convey TLS related alerts to the peer entity

Close notify: Notifies the recipient that the sender will not send any more messages on this connection (To prevent truncation attacks)

Error Alerts:

- Fatal: the connection is closed upon transmission/receipt (the session can continue but can not be used to establish new connections) (e.g., malformed message or failed MAC verification)
- Warning: the connection can generally continue, if this is not acceptable the receiver should sent a fatal alert (e.g., alerts related to certificate verification problems)

Negotiation provides flexibility to meet different and future requirements, but also increases complexity

- Makes the protocol more difficult to fully understand/verify
  - E.g., renegotiation vulnerability first discovered after many years
- Some negotiated sessions provide minimal security
  - E.g., no authentication, weak certificate chains, no confidentiality protection...
  - May downgrade to previous weaker version (if allowed by configuration)
- Users/developers/administrators can easily make incorrect assumptions about what security is actually provided
- The application may need to determine how to relate to different negotiation outcomes (e.g., whether it is acceptable or not)

HTTPS

- HTTPS (HTTP over SSL) (combination of HTTP & SSL/TLS to secure communications between browser & server)
- Uses https:// URL rather than http:// (and port 433 rather than 80)
- Protects (URL, document contents, form data, cookies, HTTP headers)

6.2 DTLS - Datagram Transport Layer Security

Datagram Transport Layer Security (DTLS) is based on TLS but does not require reliable or in-order delivery

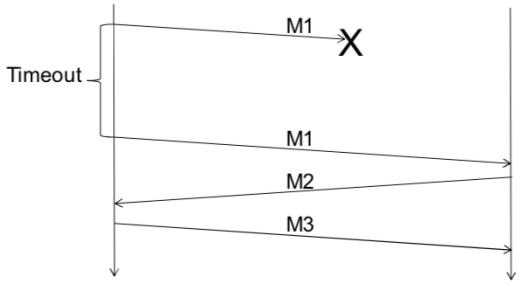
To enable reuse and reduce the risk of introducing new vulnerabilities,
DTLS is only different from TLS where required

While **TLS** is dependent on a reliable transport to ensure that records are not lost or re-ordered, DTLS is insensitive to re- ordering or loss of records

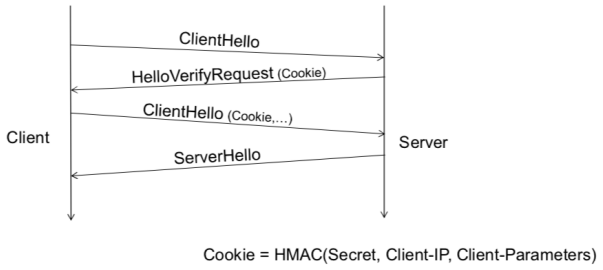
DTLS achieves this by:

- Not allowing stream ciphers (i.e., RC4) so that records can be decrypted independently
- Including explicit sequence numbers in the records
  - TLS only include implicit sequence numbers in the MAC (to provide replay and re-ordering protection)

DTLS use timeouts and retransmissions to provide reliability for the Handshake Protocol

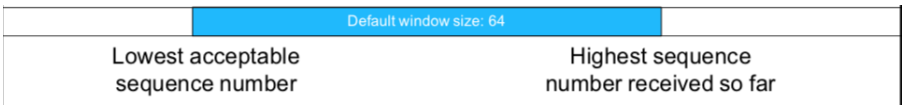


DTLS servers use stateless cookies to protect against DoS attacks



DTLS optionally supports replay detection using a sliding window mechanism (as in IPsec)

1. If the sequence number is lower than the left edge of the window, the record is discarded
2. If the sequence number falls within the window and is already marked as received, the record is discarded
3. If the MAC verification fails, the record is discarded
4. If the MAC verifies, the sequence number is higher than the right edge of the window, or falls within the window and is not marked as received, the window is updated



In summary, we have considered both TLS and DTLS

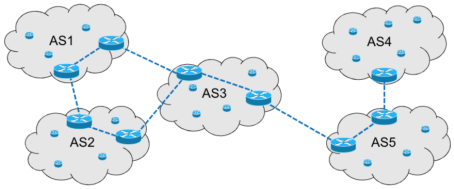
- TLS is relatively easy to utilize, but its apparent simplicity can be deceptive
  - E.g., easy to create insecure configurations or implementations
  - Depends on certificates for authentication
- The newer versions are generally more secure
- DTLS is a good alternative for applications not requiring a reliable transport
- (D)TLS 1.3 is currently in draft version

7 Protocol security

The protocols considered in this lecture may impact the security of many distributed systems and may also be considered distributed systems themselves, illustrating the potential difficulty of securing such systems.

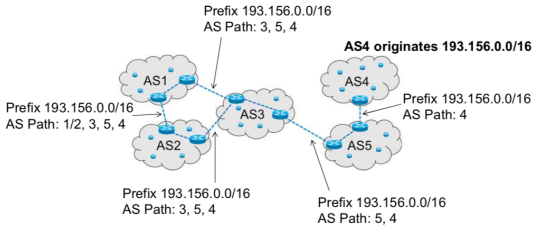
7.1 Border Gateway Protocol (BGP)

As internet is an interconnection of autonomous systems (aSs), and the Border Gateway Protocol (BGP) is the defacto interdomain (i.e., inter-AS) routing protocol.



There are about 55,000 ASs in the Internet routing system, while BGP to a large extent is still based on mutual trust

BGP is a Path Vector routing protocol, and an AS can announce or withdraw address prefixes through updates.



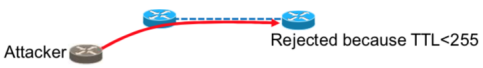
What can go wrong?

Neighboring BGP peers are not necessarily connected by a direct link, and perform incremental synchronization using a long lasting TCP session, making BGP vulnerable to attacks on TCP

- Confidentiality
  - BGP data is "generally public" (although business relations may be considered sensitive)
  - Information about TCP sequence numbers could simplify an attack
- Integrity
  - Attacker may affect routing decisions by inserting, modifying, or dropping BGP messages
- Availability
  - Attacker may reset TCP session between BGP peers, forcing complete resynchronization

Attacks on BGP's TCP communication can be mitigated by local solutions between peers (i.e., do not require a common global solution)

- E.g., using Generalized TTL Security Mechanism (Most effective when BGP neighbors are connected by direct link, low cost)

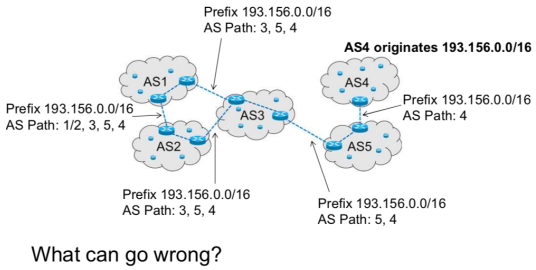


- E.g., using TCP-AO (Authentication Option) or IPSec (May require new hardware)

Preventing unauthorized prefix orgination is a more complex problem, requiring a common solution

- Prefix hijacking, deliberate or misconfiguration
  - Can typically gain widespread preference by using longer prefix
  - E.g., YouTube - Pakistan Telecom (2008)

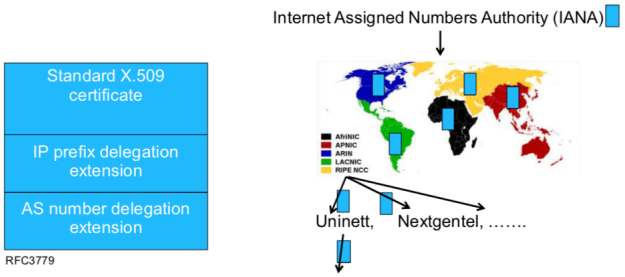
- Consequences/applications:
  - DoS (e.g., Black holes or loops)
  - Eavesdropping
  - Masquerade, MITM
  - Message modification



What can go wrong?

- May also involve unassigned prefixes and/or AS numbers
  - E.g., used by spammers/botnets to complicate traceability

The Resource Public Key Infrastructure (RPKI) provides a framework for trusted delegation of AS numbers and IP prefixes. The resource certificate hierarchy mirrors the resource distribution structure.



End Entity certificates are used to sign Route Origin Authorizations (ROA) (1 to 1), which may be used for creating better BGP route filters

Use of Resource Public Key Infrastructure (RPKI) and Resource Origin Authorizations (ROA) for route filtering when only partially deployed.

Filtering/prioritizing routes based on Resource Origin Authorizations (ROAs):

- if address prefix and AS matches valid ROA → valid origination
- if address prefix, but not AS, matches valid ROA → invalid origination
- prefix does not match any ROA → unknown validity

A valid ROA is a ROA whose corresponding End Entity certificate can be validated in the RPKI

Origin validation based on RPKI is not sufficient, as the authenticity of the entire AS path must be validated.

BGPsec: validate that the sequence of ASs in the AS Path represents the actual propagation of the BGP route announcement.

- Each AS on the path signs:
  - the signature of the received BGP update,
  - the local AS number
  - the next AS number
  - the hash of the signing router's public key
- The AS originating the route additionally signs
  - the address prefix
  - the expiry time of the route

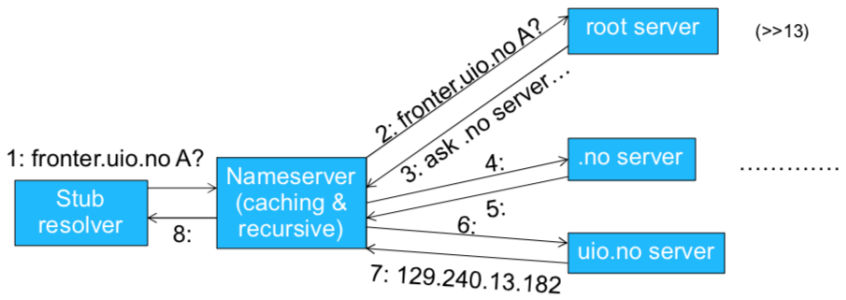
RPKI and BGPsec will take a long time to reach full deployment (if ever)

- BGPsec specifications were published in September 2017
- Partially deployed solution will likely be used for a long time
  - BGPsec to some extent changes the operation of BGP, but only where BGPsec is deployed to enable backward compatibility

BGPsec and RPKI do not aim to ensure that data packets are correctly forwarded (e.g., a router may still drop or incorrectly forward user traffic)

7.2 Domain Name System (DNS)

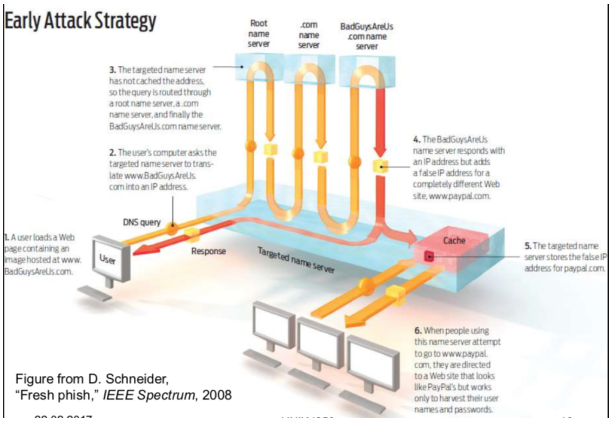
Overview (somewhat simplified as there are alternative configurations)

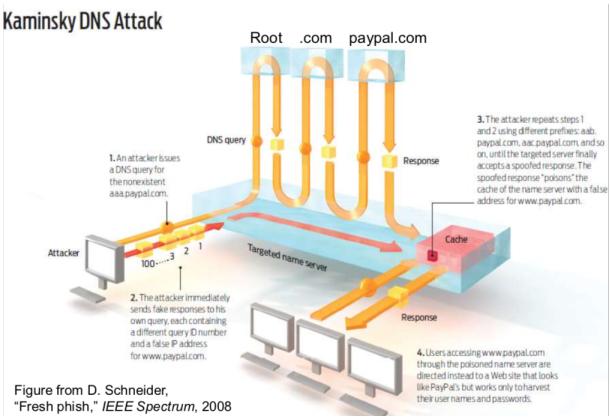


DNS is basically a distributed hierarchical database of Resource Records (RR) for various types of data, e.g., address (A) RRs

However, this attack was prevented by bailiwick checking introduced in the 90s

- Any extra information added to a DNS response is ignored if it pertains to a domain that is different from the one that was asked about in the first place.
- Can cache poisoning still be performed?





DNS cache poisoning (DNS spoofing) may affect a high number of clients

Very easy when attacker can see the nameserver's query (e.g., LAN)

- Simply respond before the authentic response

Otherwise, must guess 16 bit transaction ID (Kaminsky attack)

- Query <random value>.domain.com to avoid cached records and make server perform many queries for the given domain, increasing chance of guessing correctly
- For each query, send many spoofed responses telling the server to redirect the domain name targeted by the attack to the fake address

Randomization of server source port number makes off-path cache poisoning more difficult (another 16 bit value to guess)

- But still feasible (e.g., see A. Herzberg and A. Shulman, "Socket overloading for fun and cache-poisoning," ACSAC 2013)

DNS Security Extensions (DNSSEC)

- Delegated chain of trust from the root and down - Similar to RPKI
- Authentication through signed public keys and signing of all data
- No confidentiality or availability protection
- First published in 1997, but widespread deployment takes a longer time
  - December 2014: DNSSEC can be used for .no domains
  - Many DNS resolvers don't enforce DNSSEC validation

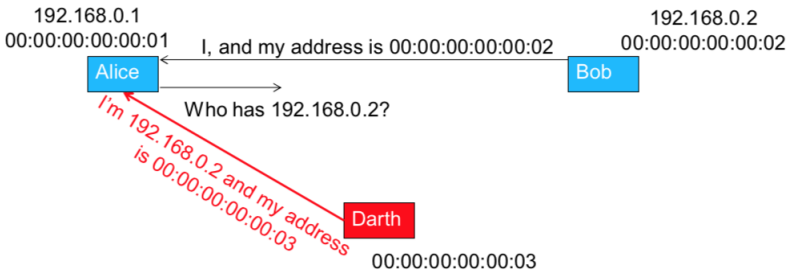
The increased size of DNS records results in higher amplification factor when used in DNS amplification attacks

DNB-based Authentication of Named Entities (DANE)

- TLS/Web security typically relies on a high number of trusted CAs that can issue certificates for any domain
  - Have been several serious security incidents due to this
- DANE allows for specifying within DNS records what certificate should be used for a given site, or what CA is allowed to issue certificates for that site
  - Relies on DNSSEC for authenticating DNS records
  - May replace or augment the use of CAs

7.3 Address Resolution Protocol (ARP) for IPv4 and Neighbor Discovery (ND) for IPv6

ARP is used to map from IP addresses to link layer addresses, but is vulnerable to ARP poisoning (e.g., used for eavesdropping on switched network, DoS, or masquerade)



Neighbor Discovery Protocol in IPv6 principally works the same way

Secure Neighbor Discovery Protocol (SEND) for IPv6 provides a solution, but support/deployment is so far limited. (Use of IPsec would require manual key distribution due to IKE bootstrapping problem)

- Proof of address ownership trough use of Cryptographically Generated Addresses (CGA) (I.e., address is generated from the nodes public key)
- Message authentication by signing messages using the private key corresponding to the public key used for generating the CGA
- Replay protection through nonce (solicited) or timestamp (unsolicited)
- Validation of router authorization based on certificate issued by third party

Summary

- Both BGP, DNS and ARP/ND have security vulnerabilities that can be exploited by an attacker
- Performing security improvements in such widely deployed systems is a slow and difficult process
- Use of for instance TLS for application traffic can mitigate many of the attacks (but does not prevent DoS)
  - (And how many actually notice that HTTPS is not enabled if entering www.bankname.com and then being diverted to an identical looking but masquerading website?)

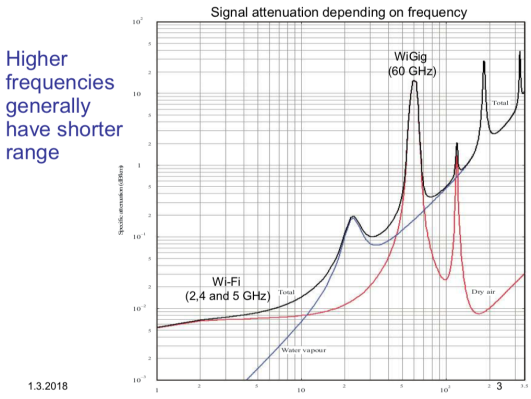
8 Wireless network security

- Passive attacks
  - Eavesdropping
  - Traffic analyses

  These are *non-invasive* and basically *impossible to detect*

- Active attacks
  - Masquerade (including rouge AP)
  - Replay
  - Message modification
  - Denial of service (including jamming)
  - Unauthorized use (misappropriation)

  These are *hard to trace*



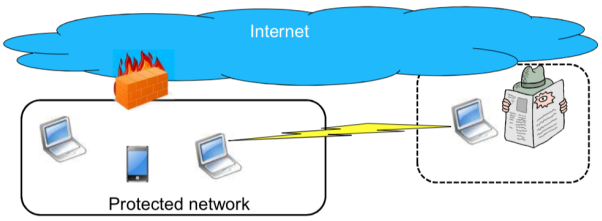
Optical Wireless Communication is limited by line of sight (may utilize reflections)

- LiFi - LED-lamps
- pureLiFis "LiFlame" - Ceiling Unit and Desktop Unit

Emission security is concerned with loss of confidentiality due to unintended compromising emanations

- Data may be reconstructed from electromagnetic emanations from monitors, computers and other electrical devices
- Such emanations may also be amplified by a nearby radio transmitter (e.g., wlan or cell phone)
- This is also referred to by the codenames TEMPEST and NONSTOP respectively
- Leakage of information may also occur through sound or vibration

Wireless devices may also pose a vulnerability to wired networks by introducing uncontrolled connections



Where relevant, dual connections should automatically be disabled

Mobile and wireless devices may pose an increased security risk

- **Lack of physical security controls** – may be easier for an attacker to steal, tamper with, or access a mobile device
- **Untrusted mobile devices** (e.g., employees personal devices not controlled by the organization) and **use of applications created by unknown parties**
- **Use of untrusted networks** – e.g., more susceptible to eavesdropping and MITM attacks and **exposure to untrusted content** – e.g., mobile devices may be exposed to other content (e.g., QR codes identifying a URL) than other computing devices

- **Interaction with other systems** – e.g., automatic device synchronization may result in data being stored in untrusted external location
- **Use of location services** – e.g., location data may be of value to an attacker

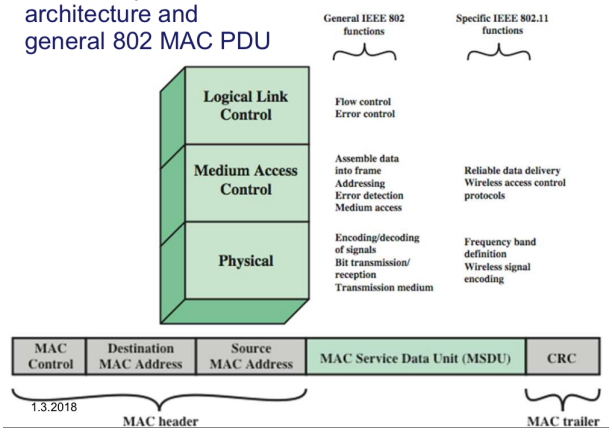
Darkhotel APT – attacks on selected high profile guests using hotel networks

- Entered last name and room number to access network
- Login portal was used to redirect to the phony installers who informed user to install software update
- Update contained digitally signed Darkhotel backdoors
  - Broke weak certificates (512-bit keys)
  - Also used 2048-bit certificates, stolen?

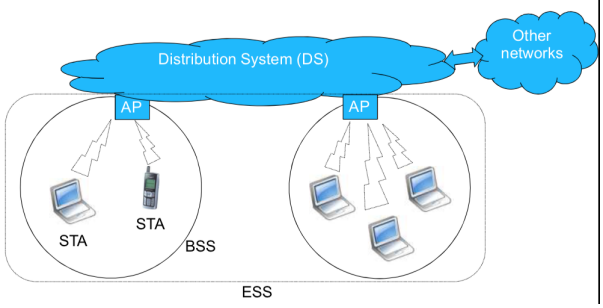
The focus of this lecture is on WLAN security as specified by **IEEE 802.11i** (also known as Wi-Fi Protected Access 2)

- IEEE Wired Equivalent Privacy (WEP) – part of 802.11 standard (1999)
  - Flawed authentication
  - Weak/flawed encryption (key reuse due to 24-bit IV)
  - Flawed integrity (RC4 encrypted CRC)
- Wi-Fi WPA interoperability certification (2003)
  - Interim solution based on subset of 802.11i draft
  - Based on WEP, but using Temporal Key Integrity Protocol (TKIP)
- IEEE IEEE 802.11i standard (2004) – Robust Security Network
  - Amendment to 802.11 standard
  - CCMP (AES)
- Wi-Fi WPA2 - interoperability certification for 802.11i implementations (2004)
- IEEE IEEE 802.11ad (2012) adds support for GCM Protocol (GCMP)
  - WiGig (60 GHz)
  - IEEE 802.11ac (2013) extends GCMP with support for 256-bit keys

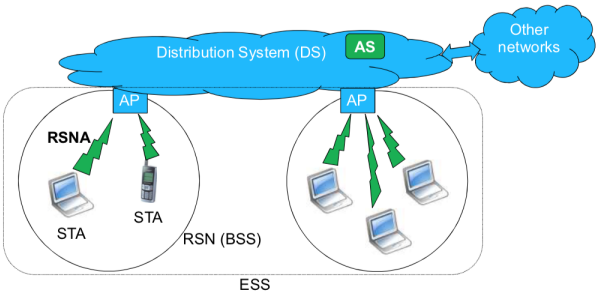
IEEE 802 protocol architecture and general 802 MAC PDU



802.11 WLAN architecture: Station (STA), access point (AP), distribution system (DS), Basic Service Set (BSS), and Extended Service Set (ESS)

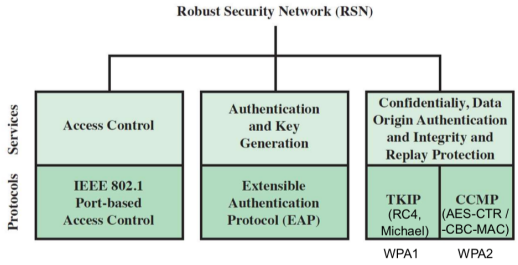


802.11i additionally introduces the Robust Security Network (RSN), the RSN Association (RSNA), and the Authentication Server (AS)

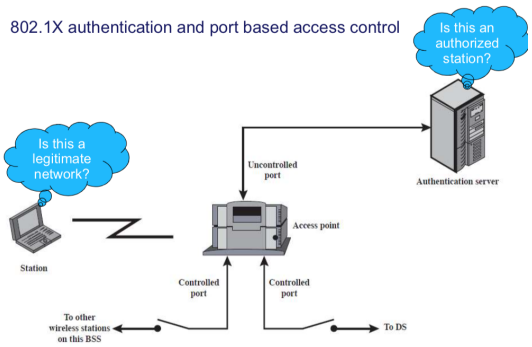


Only provides link security (end-to-end security must be provided at a higher layer)

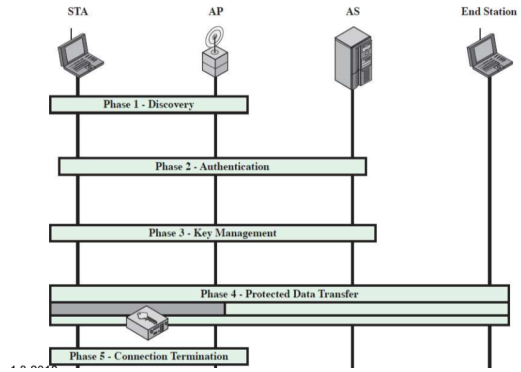
802.11i Robust Security Network (RSN) - Services and Protocols



802.1X authentication and port based access control



802.11i RSN phases of operation

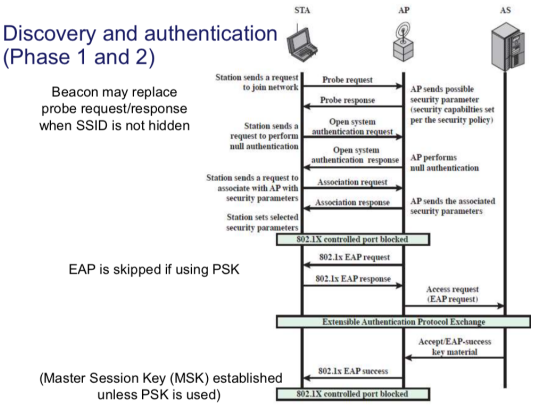


Discovery and authentication (Phase 1 and 2)

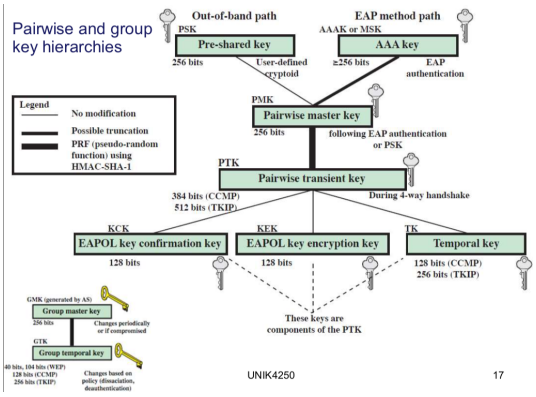
Beacon may replace probe request/response when SSID is not hidden

EAP is skipped if using PSK

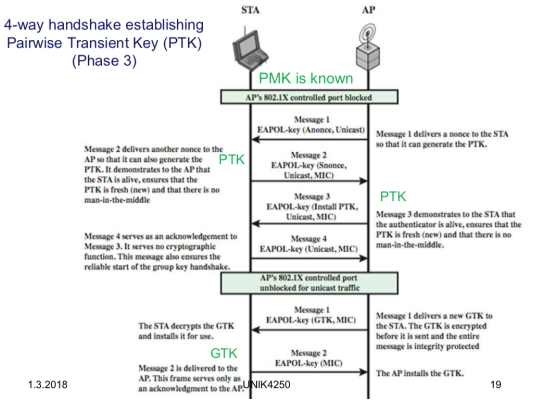
(Master Session Key (MSK) established unless PSK is used)



Pairwise and group key hierarchies



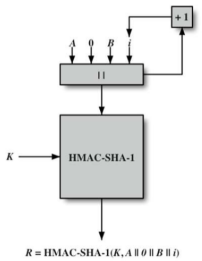
4-way handshake establishing Pairwise Transient Key (PTK) (Phase 3)



802.11i Protected Data Transfer Phase – alternative protocols

- Temporal Key Integrity Protocol (TKIP) (optional)
  - Only software changes from WEP in order to support legacy devices
  - Michael MIC
  - RC4 encryption, with new key for each frame
  - Transition solution
  - Also known as WPA
- Counter mode with CBC-MAC Protocol (CCMP)
  - Confidentiality, message authentication, and replay prevention
  - based (128-bit key)
  - Provides stronger security than TKIP
  - Also known as WPA2

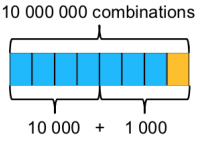
Pseudorandom function



CCMP:
PTK = PRF(PMK, «Pairwise key expansion» || AP&STA MAC adresses || nonces,384)

Wi-Fi Protected Setup (WPS): providing easy WPA/WPA2 key configuration for Alice, Bob, ... and Eve

- 8 digit PIN, where last digit is checksum
- The validity of the first and second half is acknowledged independently



- Depending on implementation: unrestricted number of PIN attempts

4-way handshake key reinstallation attacks forcing nonce reuse

- Replay of handshake message results in key being reinitialized, including resetting nonce/IV - resulting in reuse of the keystream
- CCMP: attacker can replay and decrypt packets
- GCMP: Attacker can replay and decrypt packets, and forge packets in both Directors

Implementation	Re. Msg3	Pt. EAPOL	Quick Pt.	Quick Ct.	4-way	Group
OS X 10.9.5	✓	✗	✗	✓	✓	✓
macOS Sierra 10.12	✓	✗	✗	✓	✓	✓
iOS 10.3.1 [c]	✗	N/A	N/A	N/A	✗	✓
wpa_supplicant v2.3	✓	✓	✓	✓	✓	✓
wpa_supplicant v2.4-5	✓	✓	✓	✓[a]	✓[a]	✓
wpa_supplicant v2.6	✓	✓	✓	✓[b]	✓[b]	✓
Android 6.0.1	✓	✗	✓	✓[a]	✓[a]	✓
OpenBSD 6.1 (rum)	✓	✗	✗	✗	✗	✓
OpenBSD 6.1 (iwn)	✓	✗	✗	✓	✓	✓
Windows 7 [c]	✗	N/A	N/A	N/A	✗	✓
Windows 10 [c]	✗	N/A	N/A	N/A	✗	✓
MediaTek	✓	✓	✓	✓	✓	✓

[a] Due to a bug, an all-zero TK will be installed, see Section 6.3.
[b] Only the group key is reinstalled in the 4-way handshake.
[c] Certain tests are irrelevant (not applicable) because the implementation does not accept retransmissions of message 3.

Given a weak passphrase, brute force or dictionary attacks are fully practical against WPA/WPA2-PSK

- Choose a strong passphrase (e.g., xFe>RLv6&s=@Q6q%&- 'q7CGdI9)
- May use an uncommon SSID to mitigate use of rainbow tables in brute force attacks to find PMK/PTK (PSK is generated from SSID and password – slow process)
- If applicable, disable WPS
- If AP configuration is performed online etc. (e.g., using ISP website), use a strong password..

Disabling of identifier (SSID) broadcasting and MAC address filtering provides negligible protection

- An implication of disabling SSID broadcasting at access points is that clients periodically must send queries for the SSID to discover it
  - The client machine may become more exposed and an attacker is able to discover the SSID anyway
- MAC addresses are sent unencrypted and are easy to spoof

Monitoring and auditing is an important part of WLAN security (and network security in general)

- Unauthorized WLAN devices (AP and STA)
- WLAN devices that are misconfigured or use weak protocols/implementations
- Unusual usage patterns, e.g.,
  - High numbers of client devices using a particular AP
  - Abnormally high volumes of WLAN traffic involving a particular client device
  - Many failed attempts to join the WLAN in a short period of time
- Active WLAN scanners
- DoS attacks
- Masquerade (e.g., address spoofing)

Summary - wireless network security

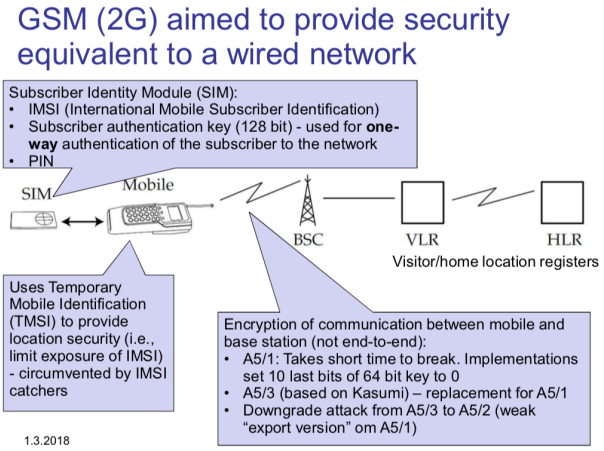
- WPA2 (i.e,. CCMP / AES) is the choice for securing WLANs today
- PSK is not suitable/scalable beyond home networks
- WLAN monitoring can be used for additional control

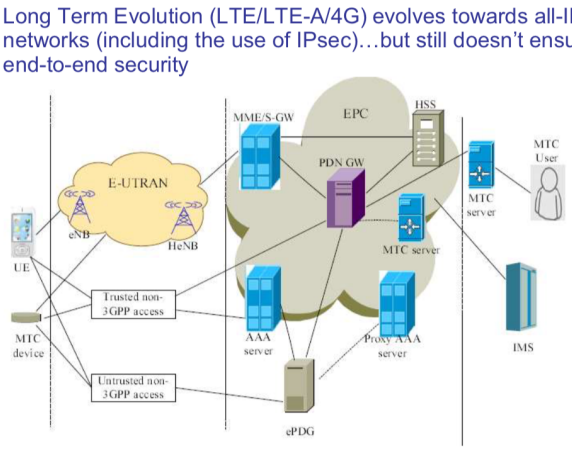
The first two are higher layer security (e.g., VPN), and may be used if satisfactory alternatives are not available.

3G (3gpp/UMTS) has the following security advantages compared to 2G

- A5/1, A5/2 etc. are replaced by various modes of the Kasumi block cipher
- All keys are 128 bit
- Two-way authentication
- Protects integrity (in addition to confidentiality) of both message content and signaling between mobile and operator network

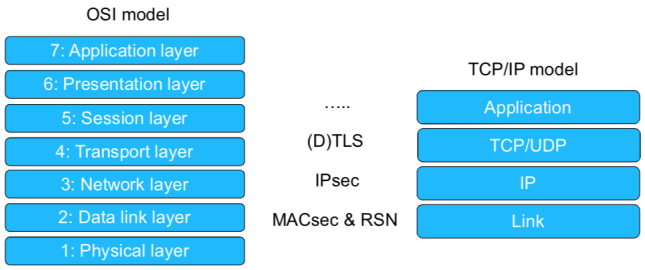
May perform downgrade attack to GSM (2G) unless disabled





9 IPsec and MACsec

Security may be provided at different layers in the network stack



9.1 Media Access Control (MAC) security - MACsec

- Specified in ANSI/IEEE 802.1ae (2006) - Using GCM-AES-128
- Several amendments and related specifications - Including support for GCM-AES-256 in 802.1AEbn (2011)
- Supported by numerous switches and network interface cards, and from 2016 also by Linux kernel (4.6 and above)

IPsec vs. MACsec

Supports	MACsec	IPsec
Confidentiality	Yes	Yes
Connectionless integrity	Yes	Yes
Access control	To network only	Yes (policy)
Traffic flow conf.	Normally not	Yes (limited)
Replay protection	Yes	Yes
Transparent to applications	Yes	Yes
Works across (layer 3) routers	No	Yes
Protection of link layer protocols (e.g., ARP)	Yes	No

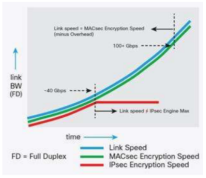
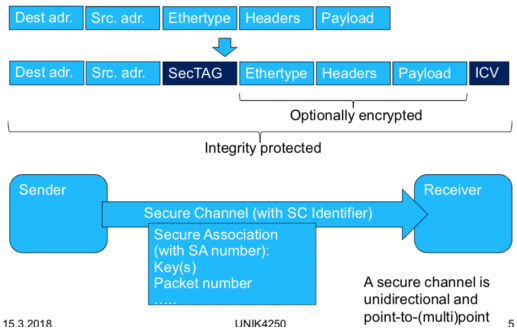
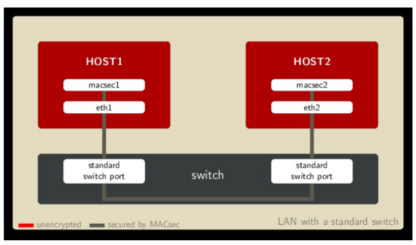


Figure from: Cisco, «Innovations in Ethernet Encryption (802.1AE - MACsec) for Securing High Speed (1-100GE) WAN Deployments White Paper», 2018

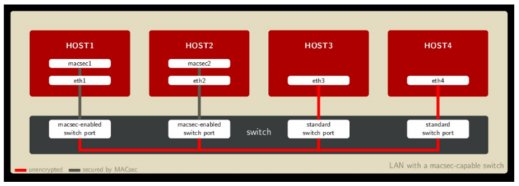
MACsec adds the SecTag and Integrity Check Value fields to the Ethernet frame, and provides optional encryption



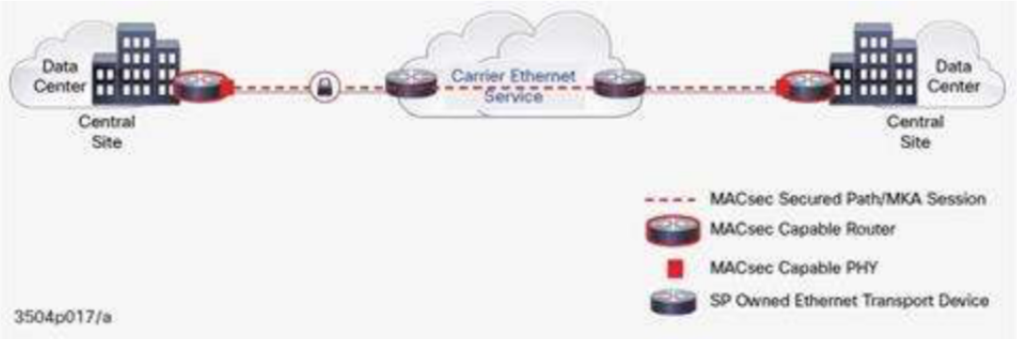
MACsec use case: host-to-host (switch is not MACsec aware)



MACsec use-case: host-to-switch (MACsec capable switch)



MACsec use case - high speed WANs



9.2 IP security (IPsec)

Reading: ANSSI, "Recommendations for securing networks with IPsec," 2015

If needed, refer to the respective RFCs identified on slide 8 for more detailed descriptions, including:

- RFC 4301 "Security Architecture for the Internet Protocol"

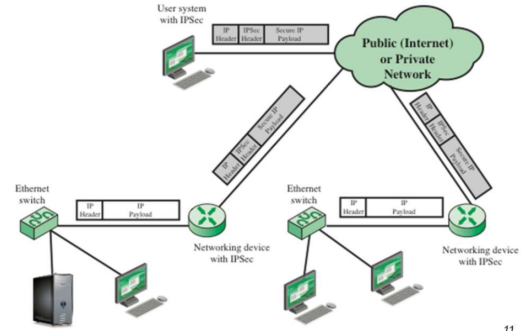
IPsec provides both authentication, confidentiality, and key management

- Applied at the IP (network) layer - i.e., to IP packets
- Applicable for use over (W)LANs, Internet, WANs, etc.
- Applicable with both IPv4 and IPv6
  - Support for IPsec was originally mandatory for IPv6 compliance, but IPsec support is now only recommended
- Can be used with multicast, but we only consider unicast traffic in this lecture

IPsec security services at the IP layer

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality
- Limited traffic flow confidentiality

IPsec can be used to provide security between both hosts and networks, transparent to higher layers



IPsec Advantages:

- An IPsec gateway can provide strong and non-bypassable security applied to all traffic crossing the perimeter
- Is transparent to applications as it resides at the network layer
- Can be transparent to end users
- Can provide security for individual users (e.g., off-site)
- Can be used to secure routing protocols
- Protects the transport layer protocols

Main IPsec protocols

Encapsulating Security Payload (ESP)

- Extension header and trailer providing encryption and optionally also authentication of payload

Authentication Header (AH)

- Extension header to provide message authentication of payload and immutable header fields
- Use of ESP is favored before less supported AH

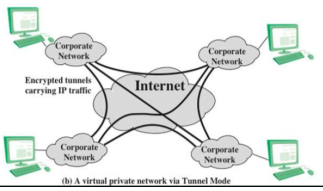
Internet Key Exchange

- The Internet Key Exchange (IKEv2) protocol is used to establish IPsec Security Associations (SA) with associated keys etc.

Both AH and ESP supports two modes: Transport and Tunnel mode

Tunnel mode (typically used for gateway-to-gateway/host security, VPN):

- ESP: Protects the entire inner IP packet (including header), but not the outer IP header
- AH: Authenticates the entire inner IP packet (including header), and selected parts of the outer IP header (including selected IPv6 extension headers)



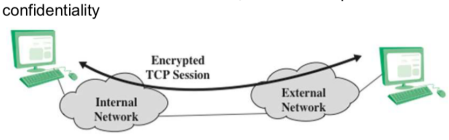
15.3.2018 17

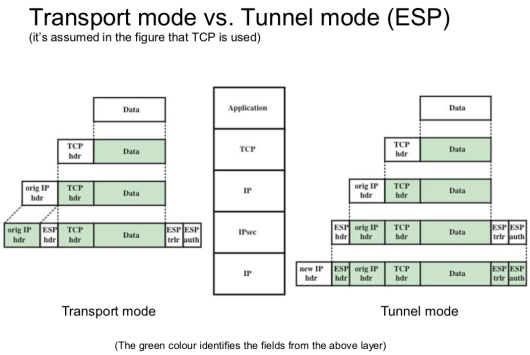
(b) A virtual private network via Tunnel Mode

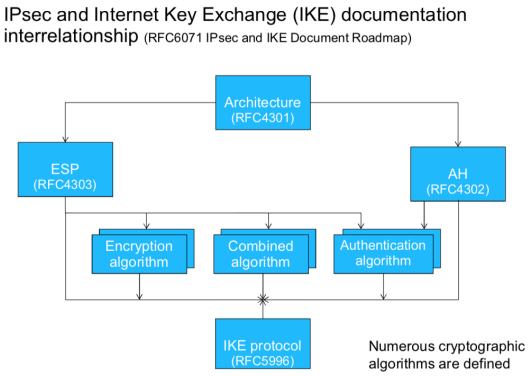
Both AH and ESP supports two modes: Transport and Tunnel mode

Transport mode (typically used for end-to-end security):

- ESP: Protects the IP payload, but not the IP header (except IPv6 extension headers following the ESP header)
- AH: Authenticates the IP payload and selected parts of the IP header (including selected IPv6 extension headers)
- Less overhead than tunnel mode, but does not provide traffic flow confidentiality







A Security Association (SA) is a one-way logical connection

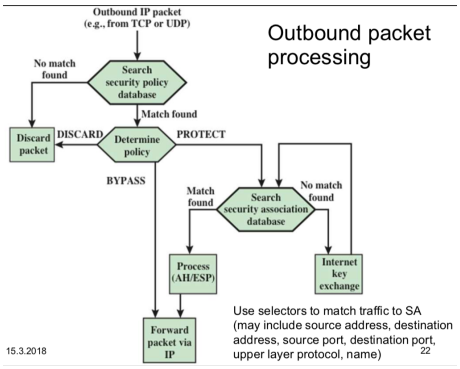
Consists of various parameters, e.g.:

- **Security Parameters Index** (SPI): 32-bit identifier selected by receiver
- **IP destination address**
- **Security protocol identifier: AH or ESP**
- Sequence number counter (64/32-bit)
- Sequence number overflow: flag indicating whether counter wraparound is allowed
- Anti-replay window
- AH/ESP information: algorithms, keys, key lifetimes, and related information (e.g., ESP initialization values)
- Lifetime of security association
- IPsec Protocol Mode: Tunnel, transport or wildcard
- Path MTU

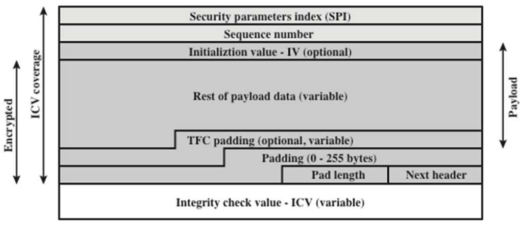
The SAs are contained in the Security Association Database

The Security Policy Database (SPD) is used to control whether and how IPsec is applied to different traffic, and what traffic to block/discard (the rules are processed in order)

Protocol	Local IP	Port	Remote IP	Port	Action	Comment
UDP	1.2.3.101	500	*	500	BYPASS	IKE
ICMP	1.2.3.101	*	*	*	BYPASS	Error messages
*	1.2.3.101	*	1.2.3.0/24	*	PROTECT: ESP intransport-mode	Encrypt intranet traffic
TCP	1.2.3.101	*	1.2.4.10	80	PROTECT: ESP intransport-mode	Encrypt to server
TCP	1.2.3.101	*	1.2.4.10	443	BYPASS	TLS: avoid double encryption
*	1.2.3.101	*	1.2.4.0/24	*	DISCARD	Others in DMZ
*	1.2.3.101	*	*	*	BYPASS	Internet

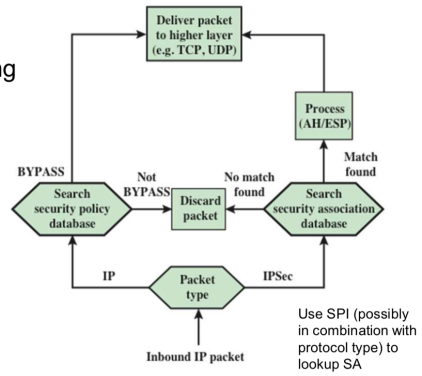


Encapsulating Security Payload (ESP) can provide message content confidentiality, data origin authentication, connectionless integrity, anti-replay and (in tunnel mode) limited traffic-flow confidentiality (depending on options selected in SA)

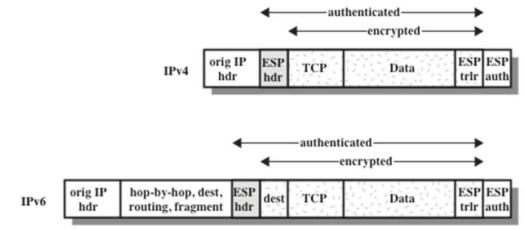


If an authenticated encryption mode is used the Integrity Check Value (ICV) may be omitted. The ICV is also omitted if integrity protection is not selected.

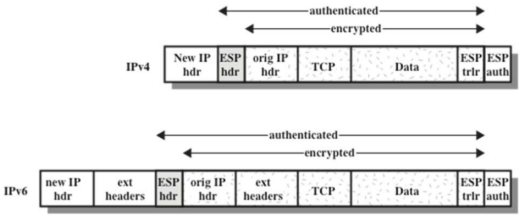
Inbound packet processing



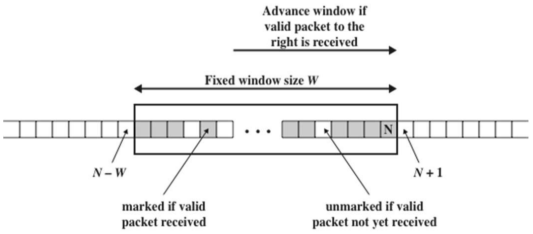
Transport mode ESP



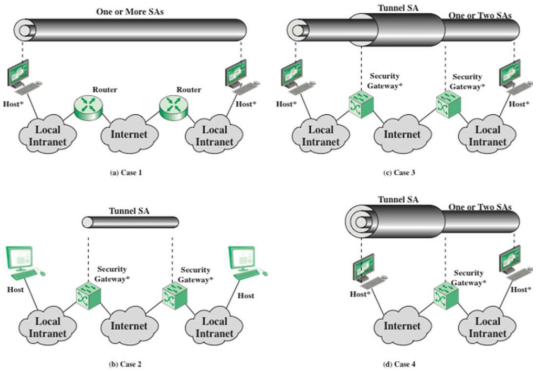
Tunnel mode ESP



Anti-replay service based on sequence numbers



(The same anti-replay mechanism is also used in DTLS)



Combining security associations

- **Transport adjacency**: Applying more than one security protocol/association to the same IP packet without using tunneling (i.e., using both ESP and AH between two hosts)
- **Iterated tunneling**: Applying multiple layers of security protocols through tunneling (where each tunnel can originate and terminate at different nodes on the path).

A transport adjacent SA bundle may again be subject to (iterated) tunneling

IPsec key management

- Manual (not recommended): the keys of each system are manually configured
- Automated: On-demand creation of keys for SAs - Internet Key Exchange (IKEv2) protocol

May need two pairs of keys in each direction (encryption + authentication)

Internet Key Exchange (IKE) provides key exchange and is used in establishing SA

- Based on Diffie-Hellman
- Employs cookie mechanism to mitigate clogging/DoS attacks (similar to what we saw for DTLS)
- Uses nonces to protect against replay attacks
- Authenticated - E.g., signature or MAC (using certificate or pre-shared key)
- Provides for negotiating SA attributes

(a) Initial exchanges

1. IKE header, cryptographic algorithms the initiator supports for the IKE SA, initiators public Diffie-Hellman value, initiators nonce
2. IKE header, cryptographic suite chosen by responder, responders public Diffie-Hellman value, responders nonce

Each party can now generate SKEYSEED, from which keys are derived. The following messages are encrypted and integrity protected

3. IDi: Initiator asserts its identity, AUTH: initiator authenticates and integrity protects the contents of the first message + Nr, SAi2: starts negotiation of a child SA, TSi/TSr: traffic selectors
4. IDr: responder asserts its identity, AUTH: responder authenticates and integrity protects the second message + Ni, SAr2: sets up child SA

(First) IPsec child SA established!

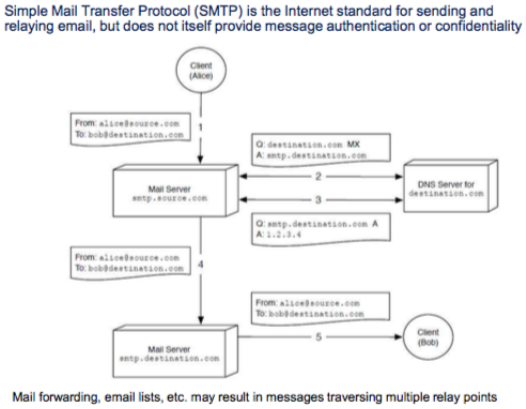
10 Application layer security

10.1 SMTP/email security

10.1.1 Email (in)security

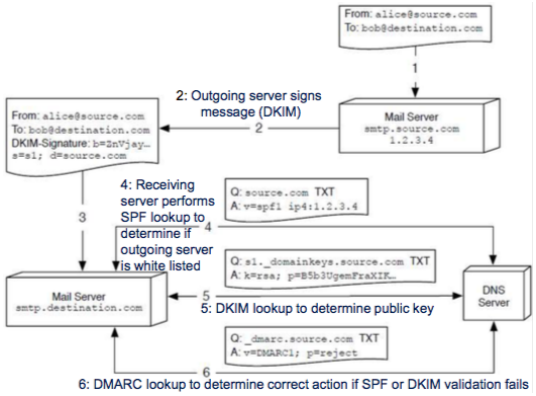
Mainly based on: Z. Durumeric et al., "Neither snow nor rain nor MITM... an empirical study of email delivery security, "Proc. ACM Internet Measurement Conference (IMC), 2015

Inscription on James Farley US Post Office Building in New York: "Neither snow nor rain nor gloom of night stays these couriers from the swift completion of their appointed rounds"



Several security extensions have therefore been developed for SMTP

- STARTTLS: An SMTP command initiating a TLS handshake
  - Relay-to-relay (i.e., hop-by-hop) opportunistic encryption
  - Provides protection against passive eavesdroppers
    - Draft RFC for SMTP MTA Strict Transport Security
  - Also defined for other protocols (e.g., IMAP and POP)
- Domain Keys Identified Mail (DKIM): Sender appends a DKIM- signature to the email enabling the receiver to authenticate the sending domain – public key published through DNS
- Sender Policy Framework (SPF): allows an organization to publish a range of servers that are authorized to send mail for its domain
- Domain-based Message Authentication, Reporting and Conformance (DMARC) builds on DKIM and SPF
  - Provides for senders publishing a DNS record specifying whether it supports DKIM and/or SPF, thereby suggesting a policy for authenticating received mail



Some observations from the paper "Neither snow nor rain nor MITM...an empirical study of email delivery security"

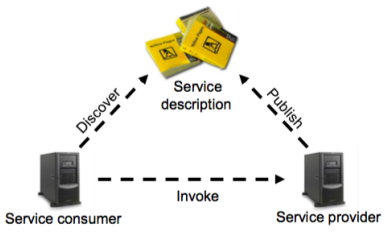
- In April 2015 60% of inbound and 80% of outbound mail to/from Gmail was encrypted (i.e., using TLS) and 94% of incoming messages were authenticated using DKIM and/or SPF
- 82% of the SMTP servers of the Alexa Top Million domains were found to support TLS, but less than 35% of these servers could be authenticated in any form
  - Smaller organizations lag behind, potentially due to some major SMTP server implementations not enabling STARTTLS by default
- STARTTLS provides no protection against MITM attacks, and STARTTLS stripping is found to happen
- End-to-end encryption (e.g., PGP or S/MIME) protects message content but message metadata (e.g., subject, sender, and receiver) remains unencrypted

10.2 XML, SOAP Web services and SOA security

10.2.1 XML and SOAP Web services security

- SOA and Web services
- XML Encryption
- XML Signature
- XML Key Management Specification (XKMS)
- WS-Security and WS-SecureConversation
- WS-Trust
- WS-Policy and WS-SecurityPolicy
- Security Assertion Markup Language (SAML)
  - Including applications for federated identity management
- eXtensible Access Control Markup Language (XACML)
  - Attribute based access control (ABAC)

Service Oriented Architectures (SOA), often implemented using Web services, seek to facilitate flexible and efficient integration across systems

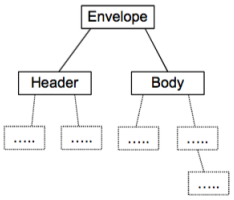


SOAP based Web services exchange messages using the SOAP protocol, which is based on XML, typically using HTTP(S) as transport

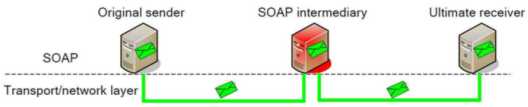
SOAP message structure:

```
<Envelope>
    <Header>
    .....
    </Header>
    <Body>
    .....
    </Body>
</Envelope>
```

XML has a tree structure:

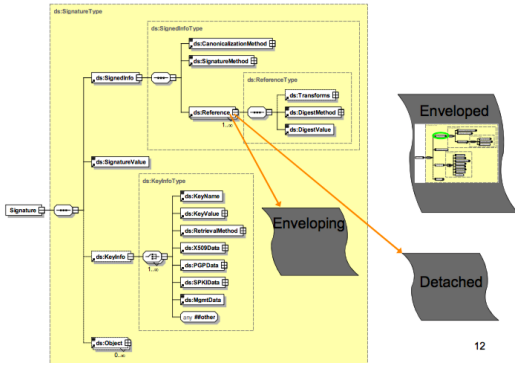


Lower layer security mechanisms may not be sufficient to provide end-to-end security for SOAP messages



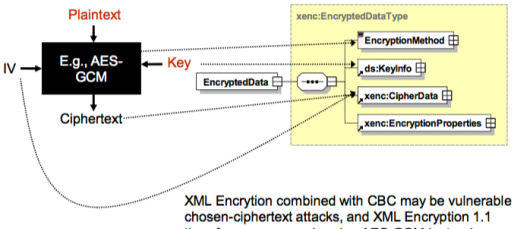
XML Encryption and XML Signature may also be used to provide security during storage at the end-nodes

XML Signature defines how to represent digital signatures (or MACs) in XML documents, and can be used to sign XML documents (or selected parts thereof) and binary resources



XML Encryption defines how to represent encrypted data in an XML document and can be used to encrypt binary resources or selected parts of an XML document

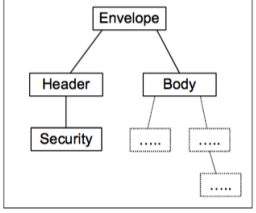
Typically uses a block-cipher (e.g., AES) for encryption:



XML Encrytion combined with CBC may be vulnerable to chosen-ciphertext attacks, and XML Encryption 1.1 therefore recommends using AES-GCM instead

WS-Security defines a SOAP security header that may be used to apply XML Signature and XML Encryption to selected parts of SOAP messages, providing confidentiality, integrity and/or message authentication

```
<Envelope>
    <Header>
        <Security>
        …
        </Security>
    </Header>
    <Body>
        .....
    </Body>
</Envelope>
```



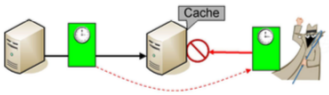
Integrity/confidentiality protection may be applied to selected elements

The SOAP security header may also be used to include other security related information, including:

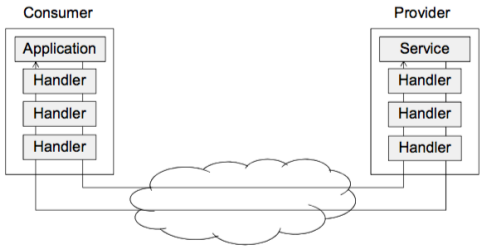
Security tokens for authentication and authorization

e.g., Username, X.509, Kerberos, SAML tokens

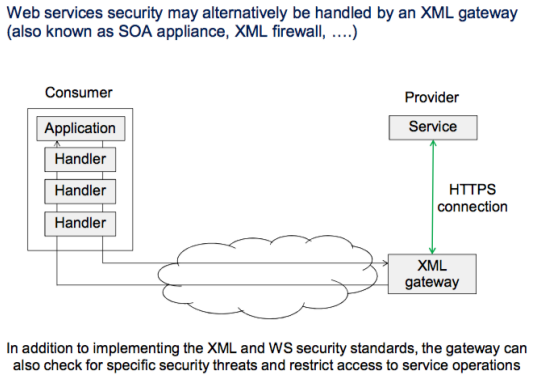
Timestamp/validity time for replay protection

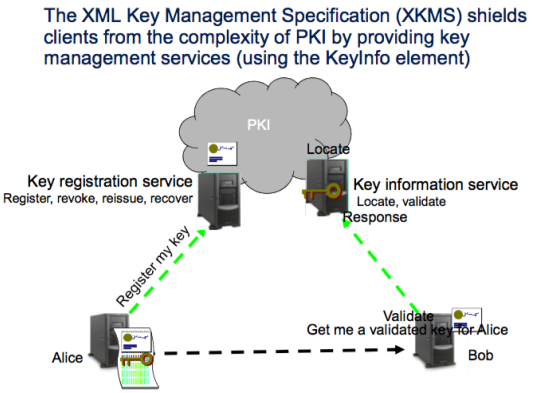


Web services platforms typically deal with different headers (e.g. security) using handlers



If a header has a must-understand attribute, it must raise an error if not being able to handle that header

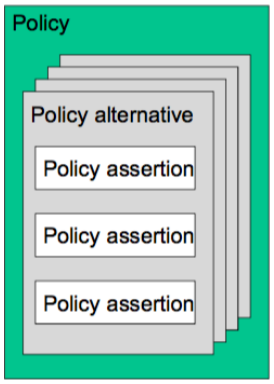




WS-Policy may be used by service providers and consumers to express interoperability requirements and capabilities

WS-SecurityPolicy defines policy assertions for use with WS-Policy to express security interoperability requirements and capabilities:

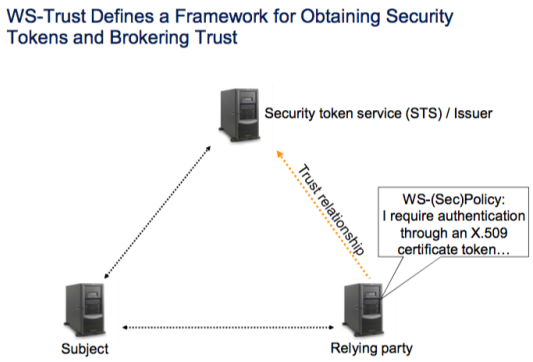
- Signed parts/elements
- Encrypted parts/elements
- Required parts/elements
- Token assertions
  - Request security token template

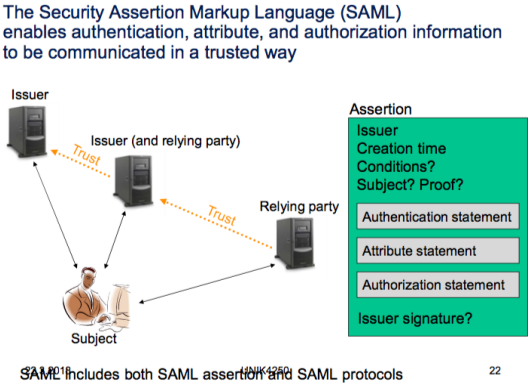


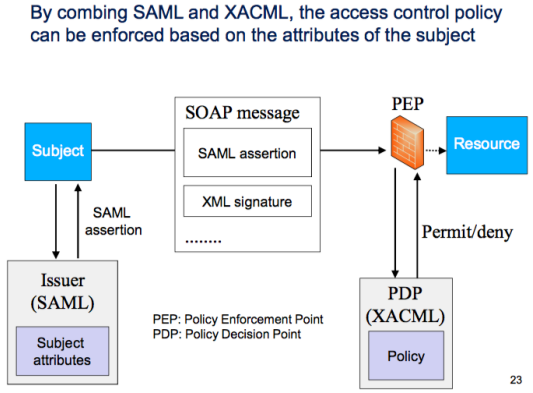
WS-SecureConversation builds on WS-Security and WS-Trust to provide a security context

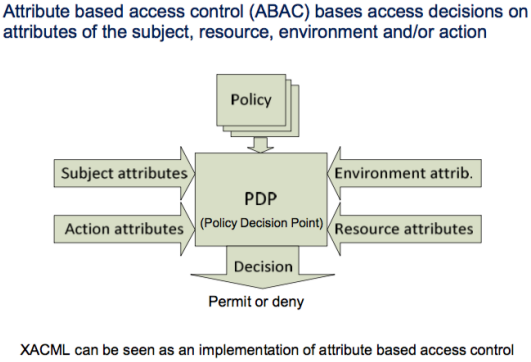
- An established authenticated context
- Derivation of keys from shared secret
- The security context is identified by a URI in the context token
- The context token can be obtained from a security token service, be created by one of the communicating parties, or through negotiation

WS-Security by itself has no notion of a communication session (i.e., it is only concerned with a SOAP request/response or a single message)









XACML rules are the basic building blocks for defining policies in the eXtensible Access Control Markup Language (XACML)

A XACML rule may contain:

- A **target**: Defines the subjects, resource, actions, and environment to which the rule applies.
- An **effect** : permit or deny (mandatory).
- A **condition**
- (and in XACML 3.0 also an **obligation**)

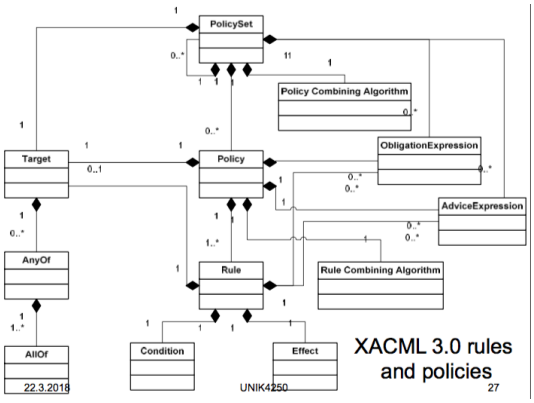
Put another way: Who may, or may not, do what to which resource given the environment

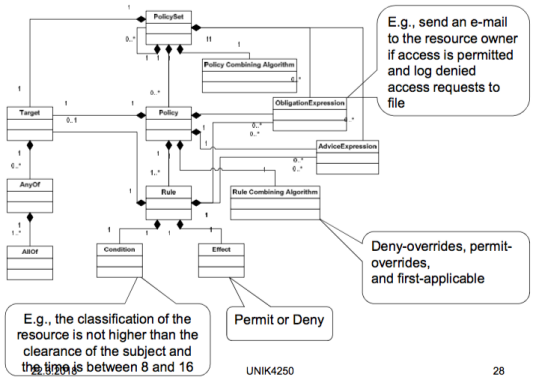
Example: *Alice* may *write* to drive *F* when *logged on locally*

A XACML rule may contain a condition further restricting the applicability of the rule A condition:

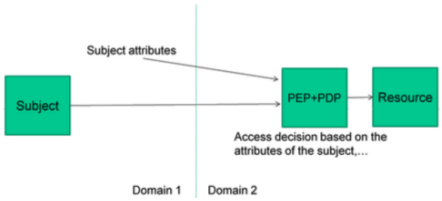
- May involve attributes of the subjects, resource, actions, and environment
- Can make use of arithmetical, comparative, set, and Boolean operators

Example: The rule only applies when the time is between 8-16 and when the subject is authorized for Top Secret.

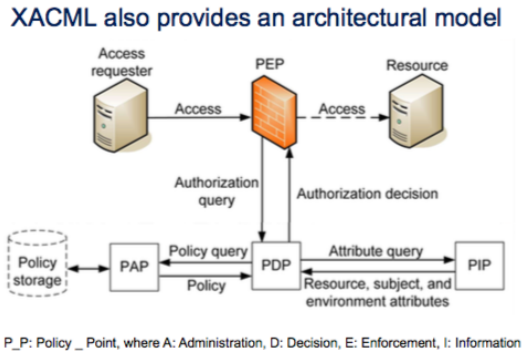


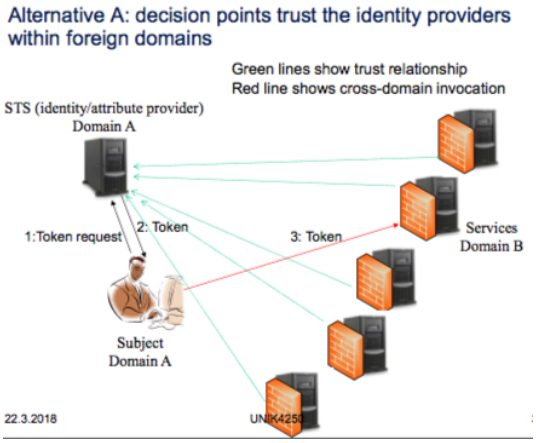


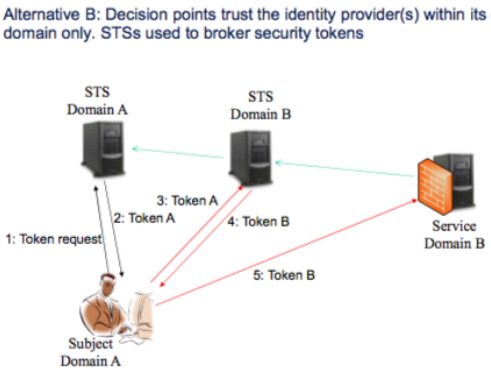
To be able to perform access control, the attributes of the subject must be communicated to the decision point within the domain of the resource

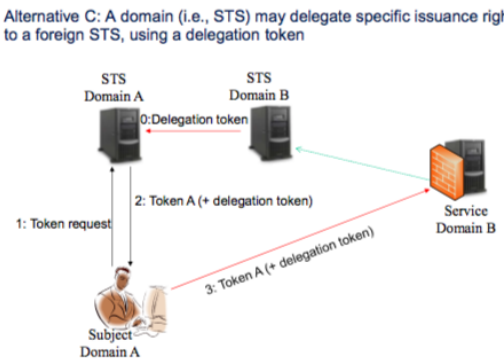


Many ways in which this can be achieved...









Consistency, Availability, and Partition tolerance (CAP)

- The so called CAP theorem (brewer's theorem) says that you can simultaneously gurantee at most two of these, but not all three
- Has implications for distributed systems in general, including security solutions depending on shared state

10.3 REST and API security

REpresentational State Transfer (REST) is an architectural style based on six constraints → RESTful APIs / Web services

1. Client-server - i.e., separation of concerns between client and server
2. Statelessness - i.e., server does not maintain client state
3. Cacheability - i.e., response data cacheability implicitly or explicitly given
4. Uniform interface - i.e., a uniform interface based on four interface constraints
5. Layered system - i.e., a component cannot see beyond the component it's interacting directly with
6. Code-on-demand (optional) - e.g., client may download and execute script

REST is not as defined/standardised as SOAP based Web services, e.g., there is no REST equivalent to WS-Security etc.

- Most RESTful APIs make use of HTTP (e.g., GET, POST, PUT, DELETE)
  - I.e., TLS should typically be used for protecting the communication
- Example:
  - Request: GET https://api.twitter.com/1.1/search/tweets.json?q=REST

  - Response: "errors":["code":215,"message":"Bad Authentication data."]

- API may be rate-limited (to protect against DoS, brute-force attacks, content scraping, etc.) and may require authentication

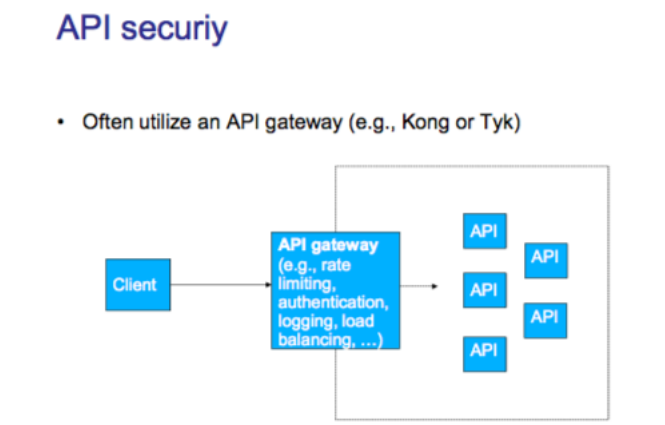
Response formats **JSON**

Requires authentication? **Yes**

Rate limited? **YES**

Request / 15-min window (user auth) **180**

Requests / 15-min window (app auth) **450**



11 Security design

Background: Saltzer and Schroeder (1975) identified eight design principles for security mechanisms

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design
- Separation of privilege
- Least privilege
- Last common mechanism
- Psychological acceptability

Much has changed since 1975, but the principles are generally still valid.

Security design principles generally have long validity, although their implementation may change over time

Saltzer & Schroder (1975)	Forskrift om informasjonssikkerhet (2001)
Economy of mechanism	Minimalism
Least privilege	Least privilege
	Self-protection
Complete mediation	Controlled dataflow
Open design (Kerckhoff 1883, Shannon 1948)	
	Defense in depth
Separation of privilege	
Least common mechanism	
Psychological acceptability (Kerckhoff 1883)	
	Redundancy
	Balanced strength
Fail-safe defaults	

Background: Rushby and Randell (1983) found that the design freedom of a distributed system can be used to secure systems in general

- The challenge is to find ways of structuring the system so that the separation provided by physical distribution is fully exploited to simplify security enforcement without destroying the coherence of the overall system
- All information channels between components must be mediated and the mediation mechanisms must be kept separate from untrusted system components.

- Separation can be physical, temporal, logical, or cryptographic
- MILS (originally an acronym for Multiple Independent Levels of Security) can be seen as a modern realization of this idea

11.1 What is a secure design?

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies." - C.A.R. Hoare, "The Emperor's Old Clothes," Communications of the ACM, 1981

A secure design is relative to the security requirements (i.e., objectives), which also must be correct!

Another perspective on secure design is provided by the requirements for a reference monitor, which are more generally applicable

The reference validation mechanism:

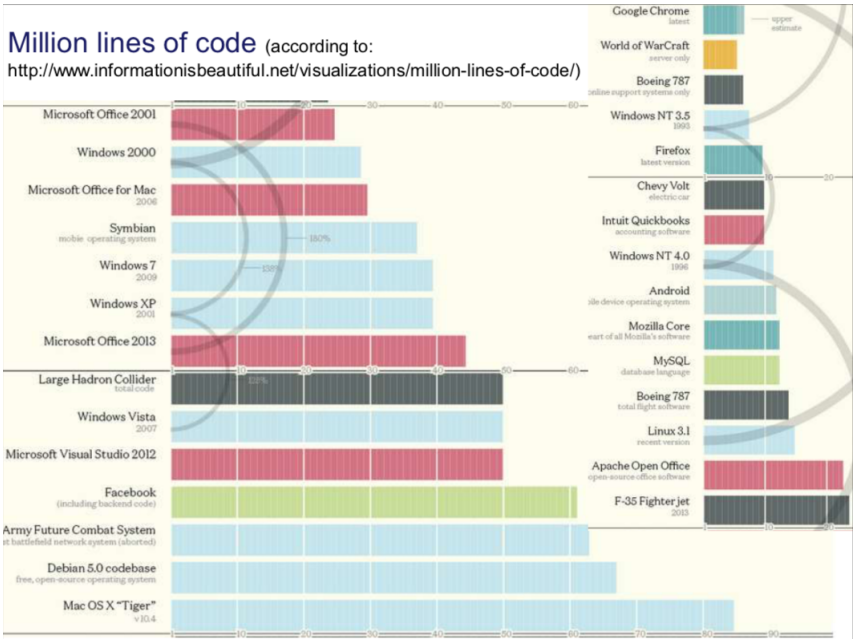
1. must be tamper proof
2. must always be invoked
3. must be small enough to be subject to analysis and tests to assure that it is correct

11.2 Avoid unnecessary complexity (economy of mechanism)

Common weakness: unnecessary complexity results in protection mechanism not being correctly *understood*, *modeled*, *configured*, *implemented*, *used*

Applies to any aspect of a system

- particular to security mechanisms as complexity makes thorough analysis and testing more difficult (security weaknesses are often not discovered during normal use or basic functional testing)



Modern systems are often inherently complex Calls for a modular architecture, where:

- Trusted modules (e.g., security mechanisms) are protected from manipulation
- Unnecessary complexity is avoided in trusted modules
- Required communication between modules is provided through well defined interfaces

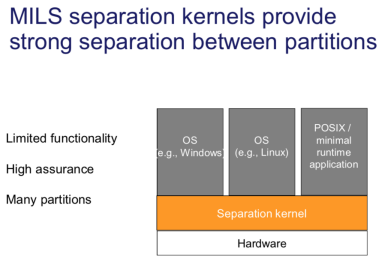
Separation provides a basis for the design of secure systems consisting of both trusted and untrusted components

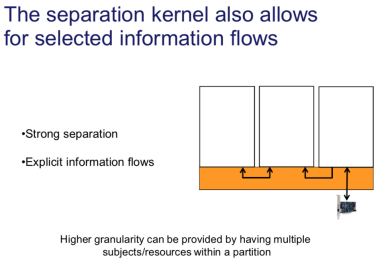
Separation can be

- Physical - e.g., separate networks, computers, or devices
- Logical - e.g., OS provided, programming language provided, virtual machines, virtual network, sandbox, or separation kernel
- Cryptographic - e.g., encryption or signature
- Temporal

Different approaches differ in strength (e.g., separation mechanisms in (or on top of) typical OS provide limited assurance)

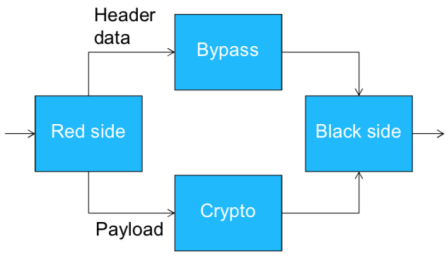
11.3 MILS separation kernels provide strong separation between partitions





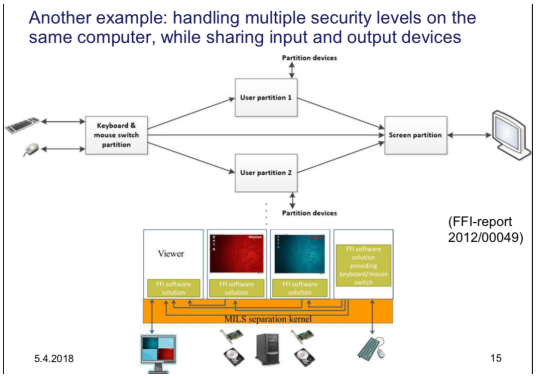
Example of modular design: Simple encryption device

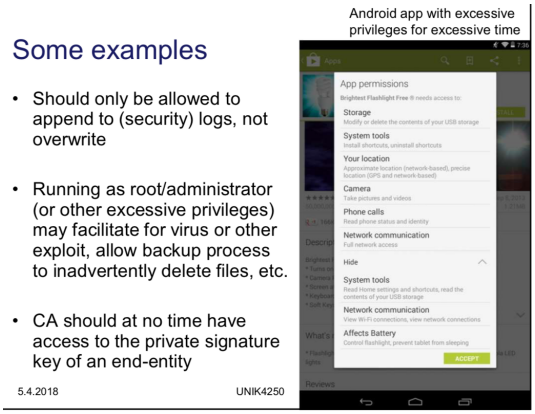
(communication between components can only occur according to the arrows) Can be implemented using different



- **Red side**: Provides network stack and forwards payload and header data
- **Bypass**: Verifies that forwarded data is legitimate header data
- **Crypto**: Performs encryption
- **Black side**: attach header to encrypted payload and provides network stack

separation mechanisms (e.g., separation kernel), but some security critical applications may require custom/separate hardware





Least privilege principle: each entity (user, component, application etc.) should only have the privileges necessary and only for the duration required

- Limits potential damage and may also reduce the potential interactions between trusted components
- May be enforced by access control mechanisms, and by modularizing and structuring a system and its mediated information flow as illustrated in the previous examples

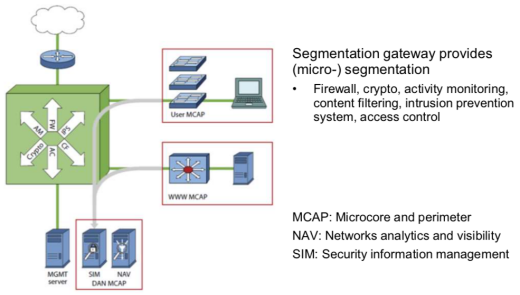
Common weakness: Excessive privileges causes weaknesses or amplifies the consequences of other weaknesses

11.4 The Zero Trust Model

Fundamental concepts:

1. Ensure all resources are accessed securely regardless of location
2. Adopt a least privilege strategy and strictly enforce access control
3. Inspect and log all traffic

Zero trust: no trusted or untrusted interfaces, networks, users, network traffic, etc.



Complete mediation: Authorization should be verified on every access (implies that the source of the request must be authenticated)

- May not be fully feasible in distributed systems (e.g., due to unacceptable performance or availability impacts)
- Should be enforced to the extent practically possible, and if required the potential consequences of deviations should be mitigated
- Time-of-check to time-of-use (TOCTOU) is an issue

Common weakness: Changes in the privileges of an entity is not reflected when later accessing a resource because an old authorization decision is relied upon

A related issue is that security should not depend upon untrusted inputs and all untrusted input should be validated

- E.g., relying on (unauthenticated) source IP addresses for authentication (e.g., DNS cache poisoning).
- Don't trust what is not trustworthy, e.g.,
  - a client application should not be trusted to enforce server access control or perform input validation
  - input from users/clients etc. must be validated
- The requirement for input validation favors simple data formats, interfaces, and protocols, as they are easier to validate
  - Insufficient input validation is a cause of a range of common vulnerabilities (e.g., buffer overflows, SQL injection,...), e.g.:
    SELECT * FROM db WHERE username='<USERNAME>' AND password ='<PASSWORD>'

Assuming a user/application may provide arbitrary input (<>), what may happen if this input is not validated? (may be mitigated by parameterized queries)

11.5 Fail-safe defaults (failing securely)

- E.g., access control or firewall should deny as default
  - Easy to overlook something otherwise and errors in systems that permit as default may go unnoticed in normal use
- Do not provide sensitive information about error to potential attacker, always check return values for errors, fail/recover to a secure state, etc.
- In distributed systems, due to interactions, it may not always be obvious what is a "safe default"

Security is a system (not just a component) requirement, thus, a system of secure components is not necessarily secure

- (Complexity → modularization →) interactions between components → source of insecure interactions and dependencies
- Strive for independent components, with clearly defined interfaces
  - Must understand their interdependencies, also in the case of unexpected events
  - Once something changes, previous assumptions may no longer hold
- «Insecure components» may be secure when used within the constraints of a larger system

Defense in depth may be used to provide stronger security than would be obtained by any of the security measures alone

- True defense in depth requires that the security measures provide overlapping (i.e., redundant security) and that the breach of one is independent from breaching the others!
  - E.g., antivirus scanning performed both in gateway (i.e., network traffic/e- mail) and on host machines, potentially using products from different vendors? Or, antivirus scanning used in combination with network monitoring and log analysis?
- One way to look at it when developing a system is to assume that the other defensive layers (e.g., firewall etc.) have been breached

From a security perspective – is it preferable with homogeneity or heterogeneity?

- In an analogy to biological systems, it is argued that monoculture is dangerous as one exploit may take them all
- While diversity avoids monoculture it may increase complexity, and provides an attacker with a larger potential attack surface for exploiting some system(s)
- Given less diversity, more resources may potentially be concentrated on securing each component/system
- A defense-in-depth strategy is likely more effective given diversity between the layers

Separation of privilege: two or more (independent) conditions must be met for e.g., access to be granted

- May prevent single accident, deception or breach from breaking security
- Related to separation of duty: e.g., the entity to perform an action should not be the same as the one approving it or the one monitoring/auditing

11.6 Final comments

- Modularization/separation provides a basic building block for secure systems (resulting in a distributed systems view)
- Secure system design is not an exact science, there are no universal rules/principles that can be followed to ensure security in complex systems (often also in complex environments)
- Need to have the security requirements right, otherwise there is not much help in fulfilling them!

12 Web (browser) security

12.1 Web browser security

Browsers are complex, highly exposed and among the applications most often targeted

Browser	CHROME	FIREFOX	INTERNET EXPLORER
Vulnerabilities (2015)	187	178	231
Market share	69%	18.6%	6.2%
Users that have the latest version installed	0.7%	0.2%	0.9%
Users that have older versions installed	68.3%	18.4%	5.3%

Vulnerabilities by Common Vulnerability Scoring Systems (CVSS) scores (2015)
(numbers from www.cvedetails.com)

	CVSS Score	Chrome	Firefox	Internet Explorer
Low	0-1			
	1-2			
	2-3		3	
	3-4		1	
Medium	4-5	28	32	36
	5-6	30	26	2
	6-7	23	31	11
High	7-8	98	55	
	8-9			
	9-10	8	30	182
	Total	187	178	231

Vulnerabilities by type (2015)

	Chrome	Firefox	Internet explorer
# of vulnerabilities	187	178	231
DoS	124	78	181
Code execution	8	83	181
Overflow	37	63	107
Memory corruption	13	41	181
XSS	5	6	5
Bypass something	31	30	19
Gain information	5	31	22
Gain privileges	2	6	16
CSRF		2	

Numbers from www.cvedetails.com

Reward programs

Chrome reward program (https://www.google.com/about/appsecurity/chrome-rewards/):

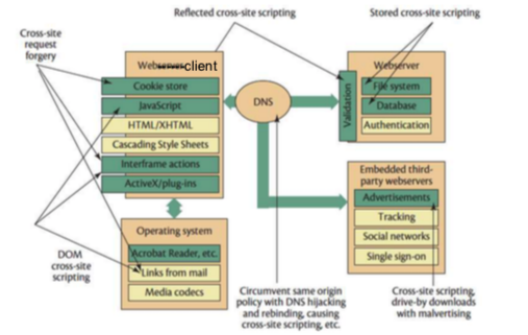
	High-quality report with functional exploit [1]	High-quality report [2]	Baseline [3]	Low-quality report [4]
Sandbox Escape [5]	$15,000	$10,000	$2,000 - $5,000	$500
Renderer Remote Code Execution	$7,500	$5,000	$1,000 - $3,000	$500
Universal XSS (local bypass or equivalent)	$7,500	$5,000	N/A	N/A
Information Leak	$4,000	$2,000	$0 - $1000	$0
Download Protection bypass [6]	N/A	$1,000	$0 - $500	$0

[1] A high-quality report with a reliable exploit that demonstrates that the bug reported can be easily, actively and reliably used against our users.
[2] A report that includes a minimized test case and the versions of Chrome affected by the bug. You will also demonstrate that exploitation of this vulnerability is very likely (e.g. good control of EIP or another CPU register). Your report should be brief and well written with only necessary detail and commentary.
[3] A minimized test case or output from a fuzzer that highlights a security bug is present without establishing that the issue is exploitable.
[4] A report submitted with only a crash dump, without a Proof of Concept (PoC) or with a poor quality PoC (e.g. a 1MB fuzz file dump with no attempt at reduction) that is later verified to be a legitimate issue.
[5] Escaping any layer of the sandbox (including the NaCl sandbox) will be considered as a sandbox escape.
[6] Landing a blacklisted test binary (malware example, UwS example) on disk where a typical user could execute it, on Mac or Windows. The file type on disk must lead to non-sandboxed code execution after minimal user interaction with the file. See the FAQ below for more information.

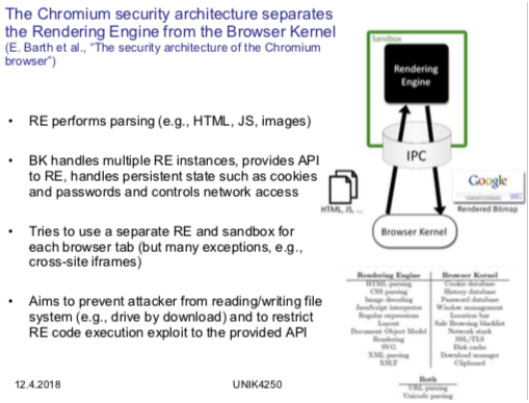
E.g., up to $15,000 for critical remote code execution and design flaws in Edge (in Windows insider preview)

12.1.1 High level threats to Web browsing

- Communication is compromised
  - Plaintext communication?
  - TLS connection compromised?
  - Certificate trust chain compromised?
- Underlying OS is compromised
- Browser is "compromised" (E.g., "man-in-the-browser": modification, fabrication, eavesdropping, ...)
- Web page/server is compromised or intentionally malicious



B. Stritter et al., "Cleaning up Web 2.0's security mess – at least partly", IEEE Security & Privacy, 2017



12.1.2 Security goals of the Chromium *security architecture*

Preventing:

- Persistent malware (malware must not survive browser close)
- Transient keylogger (keylogger must not survive browser close)
- File theft/writing

Out of scope:

- Phishing
- Origin isolation (an attacker who compromise the rendering engine may act on behalf of any web site)
- Firewall circumvention (supposed to be prevented by SOP)
- Web site vulnerabilities (e.g., XSS or CSRF)

The browser enforced same origin-policy (SOP) prevents a malicious web site from interacting within the user's session with another web site

- Origin is defined by protocol/scheme, hostname and port number
- A script in a webpage is only allowed to access content from the same origin (as that webpage)
- It does not prevent cross-domain inclusion of content (e.g., images or javascript) through HTML-tags in the webpage

The same origin-policy may be relaxed, e.g., using the Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header

Cross-site scripting (XSS) vulnerabilities are due to insufficient input validation in cases where user input is used as part of output

Attacker injects script into web page:

- Reflected (input from request used in response)
  - E.g., old vulnerability in google when using specific encoding:
    If http://www.google.com/url?q=USER_INPUT (e.g., provided in e-mail from attacker) was requested, the output would be:
    - "Your client does not have permission to get URL /url?q=USER_INPUT from this server.", where USER_INPUT includes an (non-escaped) reference to a script on attacker controlled server
- Persistent (stored)
  - E.g., attacker post (non-escaped) script reference within a comment to a news article

A Cross-Site Request Forgery (CSRF) aims to invoke a side effect by tricking the browser into sending a specific HTTP request

- Exploits the authorization/authentication of the user at the target site
- E.g., https://sikkerbank.no/transfer.do?amount=99999999&to=nils, e.g., hidden in link or in <img src> tag etc.
- Should use random session/CSRF token to protect state changing operations, but this may be defeated by XSS/XSSI

12.1.3 Cross-Site Script Inclusion (XSSI)

- One third of the 150 top-ranked domains were found to use server side generated JavaScript, 80% of these were susceptible to remote script inclusion attacks
- Remote scripts included using the HTML <script> tag are exempt for the same origin policy
- If a dynamic JavaScript is generated within a user's authenticated Web session, the script may contain sensitive user data that may potentially be leaked. (E.g., deanonymization, targeted phishing, compromise of user account,...)

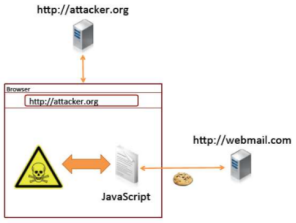
The response inherits the origin of the *including* document

- User authenticate to webmail.com
- Navigates to attacker.org, who includes webmail.com/script.js
- Global variables are accessible to attacker.org controlled scripts, and local variables are accessible through function overwriting

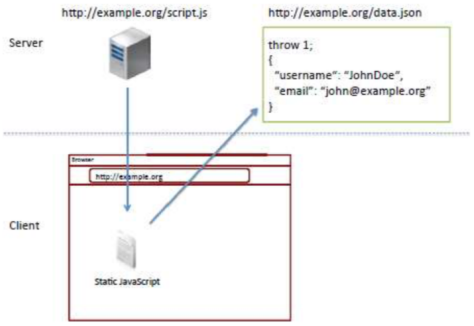
```
// Attacker's script overwriting a global function
JSON.stringify = function(data){
  sendToAttackerBackend(data);
}

-------------------------------------------------
//Within the dynamic script
function myFunction() {
  var myVar = { secret: "secret value"};

  // Calling a predefined global function
  return JSON.stringify(myVar);
}
```



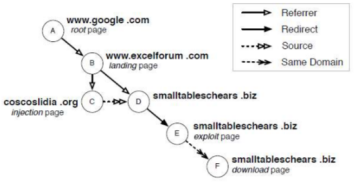
One mitigation is to place the data in a file separate from the script (and protect the data file using SOP/CORS)



Malware download paths (T. Nembs et al., "Webwitness: investigating, categorizing, and mitigating malware download paths," USENIX Security Symposium 2015)

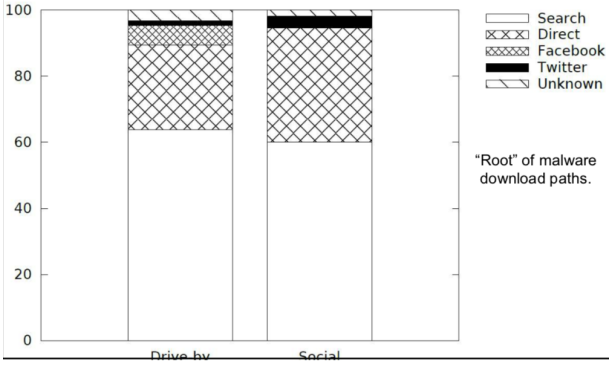
- "Social engineering": explicit user action required
- Drive-by-downloads: transparent using browser exploit

174 376 executable downloads, gave 164 drive-by and 41 social Events after filtering



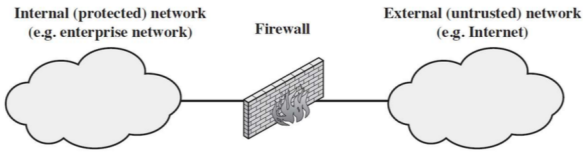
	Unique domains		Days from detect to blacklist		
	observed	blacklisted	Min.	Med.	Mean
Exploit/Download	152	9	1	20	29
Injection	52	6	20	31	36

More than 60 % of download paths started with a search. Malicious ads were involved in about 25% of drive-by and 40% of social engineering ones, but only one among Alexa top 500 domains



12.2 Firewalls

12.2.1 Firewalls are an important part of perimeter security



- All traffic from inside to outside, and vice versa, must pass through
- The security policy defines what traffic is allowed to pass
- The firewall itself should be immune to penetration

If everything on the internal network is secure, firewalls might not be needed ... but generally firewalls are required

Advantages:

- May mitigate the consequences of configuration errors, user mistakes, etc.
- Provides some protection against vulnerabilities being exploited before being patched/fixed
- Provides protection against port scanning etc.
- Facilitates network monitoring and intrusion detection (by "reducing noise" on the internal network)

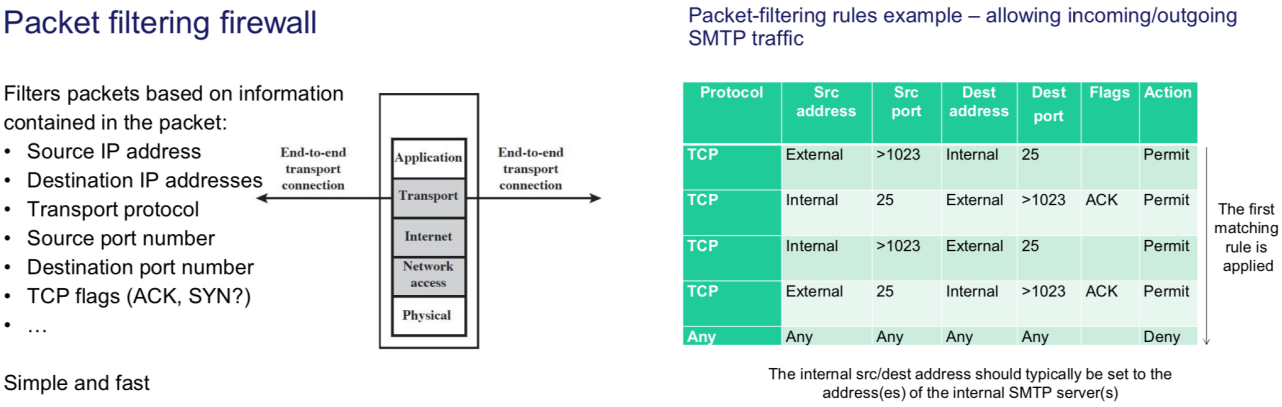
Limitations:

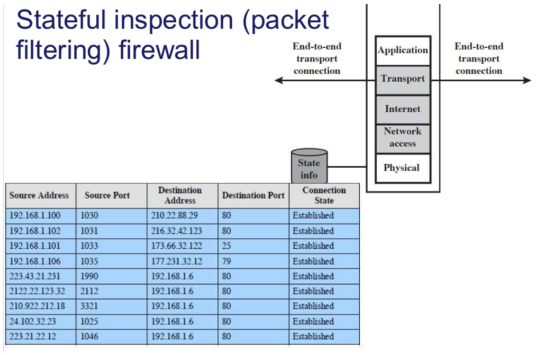
- Provides limited protection against many threats (e.g., inside threats, malicious content, phising, ...)
- Can not protect against attacks bypassing the firewall filter (e.g., through WLAN, cellular network, SSH/VPN tunnel, ...)
- Firewall "friendly protocols, UPnP, HTTPS (port 433), ...

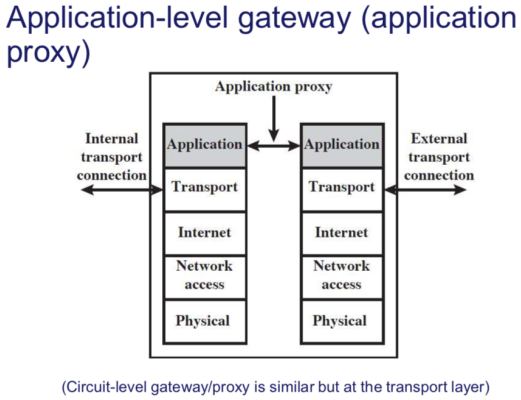
12.2.2 Firewall policy default action

- Allow biased: all packets except those meeting specified criteria are allowed to pass (that which is not expressly prohibited is permitted)
- Deny biased: only packets meeting specified criteria is allowed to pass (that which is not expressly permitted is prohibited)

The latter is generally the more secure approach

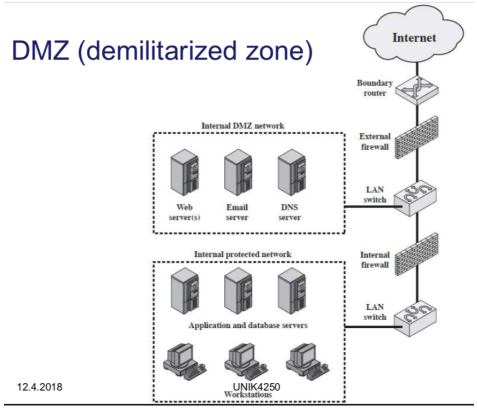






12.2.3 Network Address (Port) Translation

- Public address (and port) ↔ private address (and port)
- Conceals details of the internal network
- Restricts incoming traffic



12.2.4 Information flow control

- Diode - provides one-way information flow only
  - Network pump – similar to a diode but providing TCP acknowledgements in reverse direction
- Guard - may provide functionality similar to an application level gateway/proxy, but typically also makes use of a security label for release decisions
  - Security label: a set of metadata providing information on the security handling of a data object (e.g., a documents classification). The security label, the data object to which it applies, and their association may be protected , e.g., by a digital signature.

13 Monitoring and detection

13.1 Why intrusion detection systems?

- Support those performing security monitoring by providing alerts
- "Manual" security monitoring alone is inefficient and does not scale
- For most networks/systems, intrusions are likely to happen (even with proper security)
- Security monitoring may potentially discourage malicious actions
- The sooner an intrusion is detected the better are the changes of limiting damage
- Collected information about an intrusion can provide important knowledge for improving security and for establishing what has actually happened

13.2 Often differentiate between network- and host-based intrusion detection systems

- Network-based Intrusion detection system (N*IDS*) - Monitor one or more network segments to detect intrusions
- Host-based Intrusion detection system (H*IDS*) - Monitor events (e.g., logs) at host to detect intrusion

A single IDS may include both network sensors and host agents. An IDS may also utilize logs from appliances etc. and "network sensors" may include sandbox technology to vet files etc. Thus, IDS may not necessarily be classified as purely NIDS or HIDS.

13.3 Types of network sensor deployment

- Inline - the traffic pass through the sensor
  - E.g., for intrusion prevention purposes (or a router providing flow information, thereby acting as a sensor)
- Passive - the sensor receives a copy of the traffic
  - Spanning port (port mirroring)
  - Network tap
  - IDS load balancer

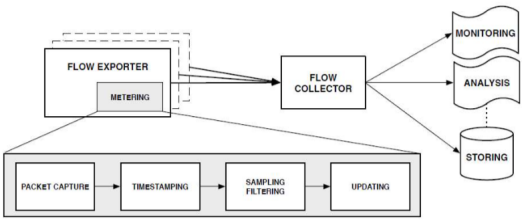
Sensors may differ in what is returned, e.g., full content data, traffic metadata, alerts, etc.
In host based IDSs, the term "agent" is typically used in place of "sensor"

13.4 Classification of detection approaches (NIST SP 800-94)

- Signature-Based
  - Check against signatures of known attacks
  - Blacklists of IP addresses, domain names, certificates etc. may be seen as a variation of this
- Anomaly-Based
  - Threshold detection (thresholds for frequency of events)
  - Profile based (deviation from trained static or dynamic profile of normal behavior/activity)
- Stateful Protocol Analysis
  - "Rules" for how given protocols should be used
- Network Behavior Analysis (NBA)
  - Makes use of information about flows (i.e., a set of packets passing an observation point with a set of common properties, e.g., src, dst, port, protocol)

13.4.1 An IP flow is a set of IP packets passing an observation point in the network during a certain time interval

All packets in the flow have a set of common properties such as source and destination addresses and ports.



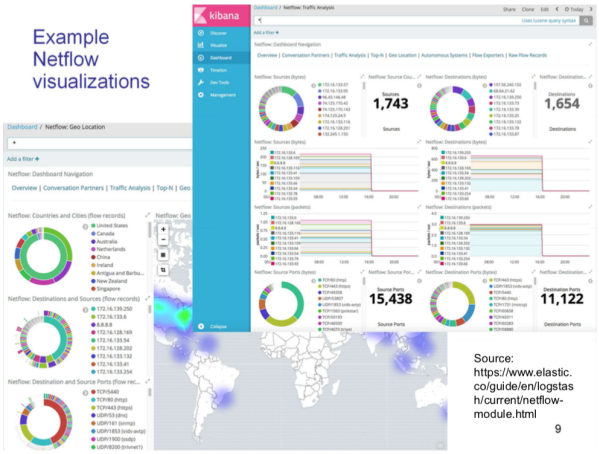
Aggregated information about a flow (e.g., times of data transfer, number of packets and bytes, next layer protocol, flags, etc.) is exported as a flow record (NetFlow or IP Flow Information Export (IPFIX)) when the flow is finished, idle or reaches an allowed lifetime.

13.5 Flow based intrusion detection

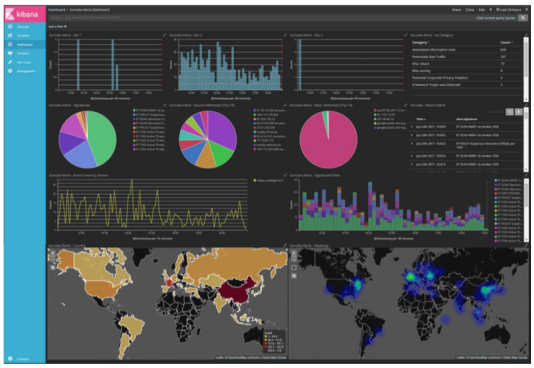
- Complements payload based intrusion detection
  - Increased use of encryption (also by malware and attackers) challenges the use of payload/packet based detection

  - Payload based detection results in high resource consumption for high bandwidths

- May be used to detect for instance:
  - DoS attacks (e.g., detected by anomalies in traffic volume or SYN/ACKs)
  - Scans (e.g., detected by high number of new flows with same source and different destination address/port)
  - Worms (e.g., similar to scan detection, honeypots, unusual connections)



Example visualization/summarization of alerts



Source: https://extelligenceblog.it/2017/07/27/elasticstack-elk-suricata-and-pfsense-firewall-part-4-kibana-visualizations-and-dashboards-pretty-pictures/

13.6 Advantages and disadvantages of different detection approaches

- Signature-based:
  + Effective at detecting known attacks (or known improper events)
  - May be fooled by evasion techniques
  - Does not consider more complex interrelations between events
  - May require access to encrypted payload
- Anomaly-based:
  + May detect previously unknown intrusions/attacks
  - May create a high amount of false positives
  - Profile (dynamic or static) may be trained
- Stateful protocol analysis:
  + Identifies unexpected use of protocols (but that's it)

- Resource intensive
- Proprietary protocols may not be covered

- Network behavior analyses (flow based):
  + Can support high data rates and encrypted traffic is not a problem
  - Not a replacement for other intrusion detection approaches

More and more traffic (malicious and non-malicious) is encrypted, either requiring the traffic to be decrypted or rendering payload (e.g., signature) based detection approaches less effective

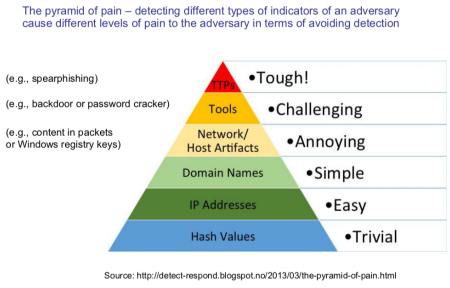
TLS inspection:

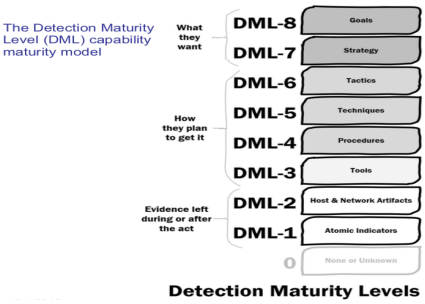
- Incoming connections can be decrypted with access to the internal server's privaate key (may require acting as MitM)
- Outgoing connections can be decrypted by inspection point acting as MitM and dynamically creating certificates for external server (requires that internal client trusts inspection point as Certificate Authority)

An alternative is to use an application to circuit level proxy that serves as the end-point of the TLS connection

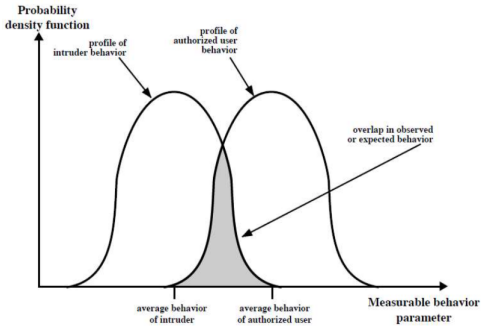
13.7 TLS 1.3 complicates TLS inspection and network security monitoring

- TLS 1.3 provides PFS by through the user of ephemeral DH (i.e., static RSA and DH cipher suites are no longer supported)
- Middleboxes are unable to disengage TLS inspection for a TLS 1.3 connection as the hash of the handshake message is included in the key derivation
- In TLS 1.3 the server certificate is sent encrypted





There is a trade-off between false positives and false negatives



The base-rate fallacy is to assume value of Pr(a|d) without considering Pr(a) (i.e., the base-rate)

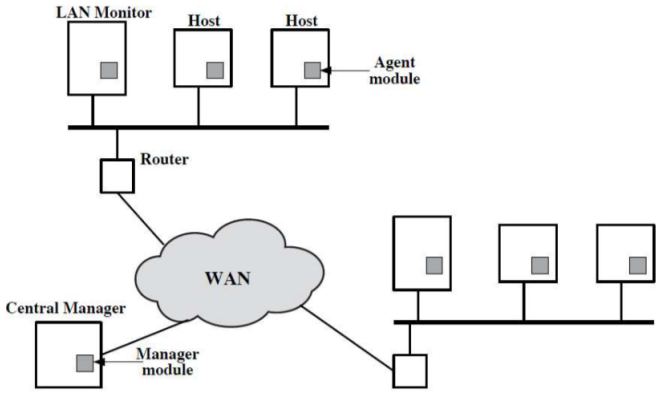
Example:

a: attack occurs (Pr(a)=1%) d: detects attack (Pr(d|a) = 87%)

$$\Pr(a|d) = \frac{\Pr(d|a)\Pr(a)}{\Pr(d|a)\Pr(a)+\Pr(d|\neg a)(1-\Pr(a))} = \frac{0.87*0.01}{0.87*0.01+0.13*0.99} = 6{,}3\%$$

Given an alert in this example, the probability of it being an actual attack is only 6,3%

In a distributed IDS, events from sensors in different systems/networks may be correlated



13.8 An Intrusion Prevention System (IPS) is basically an IDS that tries to stop detected intrusions

- End TCP session by sending TCP reset
- Block attacker (firewall or router)
- Block access to resource under attack
- Throttle bandwidth usage
- Sanitize (e.g., remove malicious e-mail attachement)

False positives may result in denial of service for authorized users

13.9 Honeypots/-nets may serve multiple purposes

- Divert attackers from critical systems
- Help detect intrusion attempts, as any connection to a honeypot can be viewed as unauthorized
- Observe intrusion to gain information (about attacker, about intrusion method)

13.10 (Data loss detection/prevention

- E.g., using flow information to detect unusual and/or excessive outbound traffic, or data gathering internally
- Detecting sensitive content, using, e.g.,
  - Document fingerprinting techniques
  - Dirty-words
  - Regular expressions
  - Machine learning
  - Information retrieval

13.11 Machine learning is gaining increased attention for intrusion and data loss detection

Typical machine learning approach:

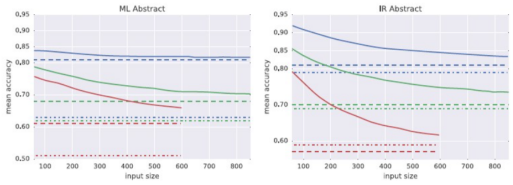
- Identify classes and features to be used for classification
- learn the model using training data (unsupervised, semi-supervised or supervised learning)
- Use the trained model to classify unknown data

The relevance and quality of the trained model is highly dependent on the training data

Example: Detection of transformed data leaks using machine learning and information retrieval (Kongsgård et al., "Data loss detection based on text classification in controlled environments," ICISS 2016)

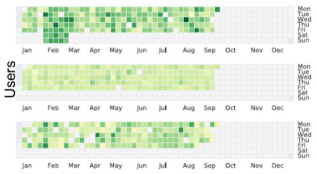
Controlled (user) environment:

- Know all imported documents and their sensitivity (*input*)
- Use this to infer sensitivity of new/modified document using machine learning - *ML* - (or information retrieval - *IR*) trained on the imported documents
- Provides improved performance for detecting transformed data leaks, that are more difficult to detect, compared to using a global classifier



While such solutions may be inefficient for detecting single data leaks, due to the number of false positives, aggregated results may be used to detect malicious/incompetent users and/or compromised systems (Kongsgård et al., "An internal/insider threat score for data loss prevention and detection," IWSPA 2017)

- The basic idea is to infer the likeliness that a user/system is leaking data from the aggregated discrepancies between the sensitivity claimed by the user, and that estimated by the data loss detection solution, for exported documents over time
- May be used to identify the entities who are most likely leaking data and better prioritize alerts



19.4.2018 24

13.12 Machine learning based detection approaches may be subject to specific attacks

- Causative/poisoning attacks: The attacker tries to influence the training set of the classifier in order to cause later misclassification
  - especially a concern when continuous training is performed
- Evasion attack: the attacker (potentially with knowledge of the training set and classifier?) tries to shape the data to be classified in such a way as to escape detection
- Model data leakage: The attacker may be able to infer data from the training set by probing the classifier – especially a concern if the training data is sensitive

14 Availability/DoS & review

14.1 Availability - ensuring resources (i.e., services and data) are accessible and usable upon demand by authorized entities

- Unintended compromise of availability:
  - Misconfiguration
  - System errors and component failures
  - Natural disasters, fire etc.
  - Power outages and other dependencies
  - ...

Typically mitigated by redundancy, high reliability, and/or fault-tolerance

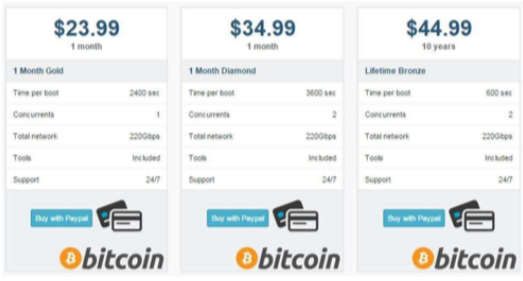
"An increasing percentage of the overall Internet ecosystem relies on a decreasing number of highly popular services"

#	Cloud		CDN		DNS		Email		ISP	
					Alexa Top 1 Million					
1	Amazon EC2	60%	Akamai	4%	Cloudflare	11%	Gmail	10%	hetzner.com	4%
2	Amazon ELB	5%	CloudFront	3%	DomainControl	7%	Secureserver.net	10%	linode.com	3%
3	GSuite	4%	Google CDN	3%	AWS DNS	6%	outlook.com	4%	endurance.com	2%
4	Office365	2%	Cloudflare	2%	DNSMadeEasy	2%	Yandex	2%	centrulink.com	2%
5	Amazon S3	2%	Incapsula	1%	DNSPod	1%	qq.com	1%	teliacarrier.com	2%
					Alexa Top 1,000					
1	Amazon EC2	94%	Akamai	46%	AWS DNS	30%	Gmail	19%	teliacarrier.com	39%
2	Amazon ELB	35%	CloudFront	30%	dynect.net	17%	sendgrid.net	10%	xo.com	36%
3	GSuite	17%	Google CDN	15%	Cloudflare	16%	secureserver.net	10%	centurylink.com	35%
4	Amazon S3	15%	fastly.net	12%	akadns.net	14%	outlook.com	5%	pccwglobal.com	33%
5	AWS other	11%	edgecast.com	12%	ultradns.com	9%	psmtp.com	2%	verio.com	33%
					.gov domains					
1	Amazon EC2	93%	Akamai	20%	akam.net	8%	outlook.com	18%	centurylink.com	13%
2	Amazon ELB	18%	CloudFront	5%	akadns.net	8%	Gmail	6%	zayo.com	10%
3	Office 365	14%	Cloudflare	5%	AWS DNS	7%	secureserver.net	6%	xo.com	10%
4	lync.com	12%	Google CDN	3%	domaincontrol.com	5%	sendgrid.net	2%	verio.com	10%
5	GSuite	6%	Incapsula	2%	dhhs.gov	4%	pphosted.com	2%	above.net	9%

Denial of Service (DoS) attacks intentionally aim to disturb availability

May in principle make use of any means, physical, logical, insider,....

DDoS as a service (Booter/Stresser)



DoS attacks may potentially be used as a smokescreen for some other attack...

The term DoS attack is most commonly used to refer to attacks taking place over a network, often making use of spoofed source addresses

Typically aim to disturb/prevent legitimate use by exhausting either:

- Network resources
- Server resources
  - State (disk, memory, sockets)
  - Processing resources

DoS attacks where some vulnerability is exploited by sending malformed packet(s) (e.g., ping of death) are relatively less of a problem today

14.2 May often classify network based DoS attacks as:

- Application layer attacks, exploiting a weakness at the application layer (E.g., HTTP flood)
- Protocol attacks, exploiting a weakness in the (network/transport layer) protocol (E.g., SYN flooding)
- Volumetric (or volume based) attacks, exhausting the bandwidth capacity of the target through high volumes of traffic (E.g., DDoS flooding)

14.3 Volumetric attacks are often distributed in nature (i.e., DDoS), potentially utilizing botnets

- Reflection based flooding attacks
  - Attacker sends forged requests to the reflectors, triggering their responses to the victim
- Amplification based flooding attacks
  - Reflectors are used to generate multiple or larger responses to the attackers' requests

Amplification attack on GitHub on February 28, 2018

- Internet exposed memcached servers provided high amplification factor
  - Supports UDP requests - easy to spoof source address (i.e., specifying the targets IP address)
  - No authentication required
- Akamai Prolexic became intermediary to handle the high traffic volume - filtering all traffic originating from UDP port 11211 (i.e., the default port of memcached) (source: https://blogs.akamai.com/2018/03/memcached-fueled-13-tbps-attacks.html)

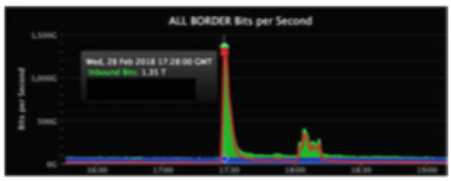
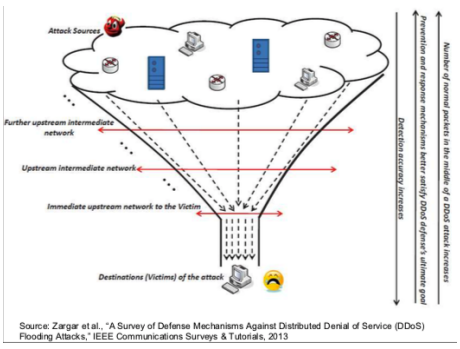


Figure: https://githubengineering.com/ddos-incident-report/



Source: Zargar et al., "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," IEEE Communications Surveys & Tutorials, 2013