comp38411 - cryptography and network security

available at JTANG.DEV/RESOURCES

(1) introduction

confidentiality: preventing disclosure of information
integrity: preventing unauthorised modifications
availability: ensuring timely and reliable access
authenticity: being genuine and able to be verified
accountability: actions of an entity can be traced

masquerade: impersonation of another entity
replay: subsequent transmission of captured data unit
modification of messages: alteration of messages
denial of service: prevent normal use of facilities

(2) introduction to cryptography

symmetric encryption: $C = E(K, P),\ P = D(K, C)$

confusion: each bit of the ciphertext has a non-linear relationship with each bit of the plaintext bits and key bits
diffusion: changes to plaintext bits or key bits spreads and affects many ciphertext bits

stream ciphers: encrypt one bit or one byte at a time, uses XOR operator

unconditionally secure: cannot be broken no matter what. only one-time pad is unconditionally secure - but is highly impractical
provably secure: the difficulty of breaking the system is at least as difficult as a known hard problem, such as integer factorisation. examples would be RSA and DSA
ad-hoc security: not provably secure, but we trust it as many skilled people have failed to break it, such as AES

ciphertext-only attack: attacker only has ciphertext itself to analyse, hard to do much due to lack of info
known-plaintext attack: attacker has plaintext-ciphertext pairs at their disposable
chosen-plaintext attack: attacker is able to choose plaintext to encrypt and deliberately picks patterns to reveal structure

(3) - conventional cryptography

security goals of encryption algorithm
(1) completeness - every bit of the output should depend on every bit of the input and every bit of the key
(2) avalanche effect - small change in either plaintext or the text produces significant change in the ciphertext
(3) statistical independence - the ciphertext and plaintext appear to be statistically independent (no patterns

block ciphers: work with blocks of data at a time, defined by the block size. the larger the block size, the larger the memory footprint, and so less ideal for embedded hardware, say.

(3) - conventional cryptography

DES: a **16 round** feistel cipher, which takes a **64 bit input**, and a **56 bit key**. operations include permutation, XOR and substitution

AES: a **10/12/14 round** cipher, which takes a **128 bit input** with **128/192/256 bit key**. operations include AddRoundKey, SubBytes, ShiftRows, MixColumns.

why is AES the way it is?
AddRoundKey: ensures encryption depends on key
SubBytes: provides non-linear relationship (confusion)
ShiftRows: changes in a given column affect other columns (diffusion / avalanche)
MixColumns: changes in a byte affect other bytes in the state (diffusion)
KeyExpansion: otherwise the same key is used in every round, thus being vulnerable to a slide attack

ECB mode: plaintext blocks are processed independently, *extremely insecure*. no error propagation onto subsequent blocks as blocks are processed independently.

CBC mode: chaining of blocks by the means of feedback, which aids in concealing the patterns of plaintext. errors in one block will propagate to subsequent blocks. encryption is not parallelisable, but decryption is.

CTR mode: encrypts a nonce along with an incrementing counter value, which serves as a key for XOR'ing with the plaintext block. can be used to turn a block cipher into a stream cipher.

(4) public key cryptography

asymmetric encryption: makes use of a pair of keys (public and private). $C = E(K_E, P)$ and $P = E(K_D, C)$. asymmetric encryption is more expensive than symmetric!

rsa: can be used for encryption and signatures
key generation
(1) choose two large primes $p, q, p \neq q$
(2) calculate $n = pq$ and $\varphi(n) = (p-1)(q-1)$
(3) choose $e \in \mathbb{Z}, gcd(\varphi(n), e) = 1, e \in (1, \varphi(n))$
(4) calculate $d \equiv e^{-1} \mod \varphi(n)$
private key = {d, n} and public key = {e, n}
encryption is $C \equiv M^e \mod n$ $(M \in [0, n-1])$
decryption is $M \equiv C^d \mod n$

(5) mac and hash functions

hash function: takes an input of any length and produces a fixed size output, known as the hash value.

(5) mac and hash functions

four properties of a good hash function
(1) compression: takes an input of any size and produced a fixed length output
(2) one-way: easy to compute $h = H(x)$, but impossible to compute x from $H(x)$
(3) second preimage: given m_1, it is impossible to find $m_2 \neq m_1$ wit h $H(m_1) = H(m_2)$
(4) strong collision: it is impossible to find $m_1 \neq m_2$ with $H(m_1) = H(m_2)$
if H is not second preimage resistant, signature forgery is possible. if H is not strong collision resistant, we don't have non-repudiation

MAC: a function which protects the integrity and authenticity of a message, by producing a tag
fresh message: not a replayed message, can be achieved by sequence numbers, timestamps and nonces
keyed hash: hashing the message and key, $h = H(K||M)$
HMAC: MAC construction based on hash function

types of authenticated encryption: hash-then-encryption, MAC-then-encryption, encrypt-then-MAC, encrypt-and-MAC. only encrypt-then-MAC allows the recipient to verify the tag before decrypting - anti-DoS.

CMAC: MAC based on CBC mode, take the last output block as the tag (technically, the MSBs of this last block)

(6) digital signatures

digital signatures should be message-dependent (non-reusable / integrity), signer-independent (unforgeable / authenticity) and verifiable (origin authenticity)

DSA: signature algorithm based on discrete logarithms. does not provide encryption like RSA. message digest cannot be recovered from signature, unlike RSA. requires an ephemeral key for each message.

(7) public key infrastructure

PKI: set of protocols, policies and procedures to enable practical deployment of public key cryptography

end entity: end-users, devices or any identifiable entity
certificate authority: issuer of certificate and certificate revocation lists, should be a trustworthy source
mandatory fields of a x.509 certificate
version number, serial number, signature algorithm ID, issuer, validity period, subject, subject public key, signature

certificate revocation list: list of blacklisted certificates before its expiry date. reasons for revocation:
(1) user's private key compromised
(2) user is no longer certified by the CA
(3) CA's certificate has been compromised

(7) public key infrastructure

verifying a certificate: follow the issuance chain in a bottom-up manner, until you find a certificate with a signature you trust. number of levels required to check can be reduced by means of cross certification.

digital certificate: an document which binds the public key of an entity to the entity itself

(8) key management

key management issues
(1) generation: are the generated keys of sufficient length
(2) storage: are keys stored in a secure manner
(3) transportation: are keys delivered in a securely
(4) establishment: how do two parties establish keys
(5) revocation: how are keys revoked and replaced?

key generation: the key space should be sufficiently large, else a exhaustive search is feasible. keys should be generated using psuedo-random generators, not some lame-ass random number generator.

key storage: keys need to be stored securely. key wrapping is encrypting the key with another key - keys should never appear unencrypted outside the encryption device. hardware tokens should require a PIN to access from secure memory.

session keys: use session keys so that adversaries have less ciphertext for cryptanalysis. shorter lifetime of session key limits the exposure in the event of key compromise.

anonymous diffie-hellman: public-key exchange protocol. both parties know g and p (prime). each party chooses a random exponent, say a and b, computes g^a and g^b respectively, and sends it to the other party. upon receipt, the shared key can calculate by $(g^a)^b$ and $(g^b)^a$. this is vulnerable to **man-in-the-middle** attack.

authenticated diffie-hellman: the communicating parties sign the g^a and g^b values, so that a man in the middle cannot modify these values whilst retaining a valid signature

key distribution without PKC: use a trusted third party (KDC) which is assumed to have shared master secrets with the two parties which wish to communicate. an example would be needham-schroeder protocol.

nonces vs timestamps: nonces need to be (at least partially) contributed from the recipient - so we cannot do this if say, the recipients is a large and/or unknown group. nonces should be psuedo-random and from a large key space, else we will have nonce reuse. timestamps require communicating principles to have their clocks in sync. both are used to achieve freshness.

(9) entity authentication

entity authentication is the process whereby one party is assured of the identity of a second party.

basis of identification
(1) something known (passwords, pins)
(2) something possessed (chip cards) not demons!
(3) something inherent (fingerprints, retinal patterns)

AAA services
authorisation: linking access rights to particular identities
authentication: verifying the identity of a claimant
accounting: using a users identity for logging security-related events

how unix authentication works
(1) user signs up with a password, unix computes a salt based on the current time, hashes the password and the timestamp, to produce a hashed password. the timestamp is also stored
(2) when the user tries to login, unix will hash the user input, along with the timestamp, and compare it against the hashed password stored

how unix protects users passwords
(1) hashing: plaintext passwords are not stored
(2) password salting: the salt alters the DES expansion function, meaning each password in a dictionary attack requires 4096 trials. in addition, off-the-shelf DES chips cannot be used to assist hardware based attacks
(3) shadow files: the hashed password is stored in a shadow file which only is accessible with root privileges

replay attacks on unix
unix is not protected against replay attacks, attackers can eavesdrop on a network and later replay it to gain access. we assume LAN is secure and that secures do not bring their own client software in, and often overlook this issue

one-time passwords: safe from eavesdroppers and subsequent impersonation. can be done using lamport's OTP scheme, hashing-based OTP, etc

(10) ip security

IPSec provides security at the IP level, by the means of **authentication header (AH)** or **encapsulating security payload (ESP)**

AH provides access control, connectionless integrity, data origin authentication and rejection of replayed packets
ESP provides access control, confidentiality, limited traffic confidentiality
ESP with authentication provides everything AH and ESP provide

security association (SA): a set of agreed attributes (authentication mechanism, encryption algorithm, etc) between two parties that wish to communicate

transport mode: regular mode for packets to travel from source to destination in network and **only applicable to host implementations**

tunnel mode: can be used if the endpoints do not have the ability to carry out security checks on the packets, then the IPSec Server is responsible for doing the checks

anti-replay: achieved using sequence numbers, attached by the sender. upon receipt (if they ever arrive), we use a *window* to help reorder the packets

combining SA's
as a single SA can only implement either AH or ESP **but not both**, sometimes we employ multiple SA's to achieve certain IPSec services. for example, transport-tunnel bundle is when you apply AH transport first, then ESP tunnel. benefit of transport-tunnel over ESP with authentication is that more fields are protected, namely the original IP header

full notes

detailed notes: `https://comp38411.jtang.dev`