Krypto Oppsummering

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7f4c26ed-17d1-4bd4-b75c-8e502a4baf45/infosec.pdf

Lecture 1:
- The CIA Triad:
- Passive Threats
- Active threats
- Security services and mechanisms
- Main security service
- Main security mechanisms
- Risk management

Lecture 2:
- Basic number theory
  - Basic properties of factors
  - Division algorithm
- GCD
  - Euclidean algorithm & extended euclidean algorithm(EEA)
- Modular arithmetic
  - Residue class
  - Notation: a mod n
- Groups
  - Cyclic groups
- Computing inverses modulo n
- Modular inverses using Euclidean algorithm
  - An actual example of modular inverses.
- the set of residues $\mathbb{Z}_p^*$
- Finding a generator for $\mathbb{Z}_p^*$
- Fields
  - Finite fields GF(p)
  - Boolean values

        - Negation
- Lecture 3:
    - Terminology:
    - Confidentiality and authentication
    - Cryptosystems
    - Symmetric and asymmetric cryptography
    - Notation for symmetric encryption algorithms
    - Methods of cryptoanalysis
    - Exhaustive key search
    - Attack classification
    - Which attacks should be prevented?
    - Kerckhoffs' Principle
    - Alphabets
    - Statistical attacks
    - Basic cipher operations
    - Ceasar cipher
    - Random simple substitution cipher
    - Polyalphabetic substitution
    - Vigenère cipher
    - Autokey cipher
    - Running key cipher
    - Rotar machine
- Lecture 4
    - Hill cipher
    - Stream ciphers
    - One time padding
    - Perfect secrecy (Shannon)
- Lecture 5 Block Ciphers
    - Block cipher
    - Product cipher
    - Iterated ciphers
    - 1. Feistel ciphers
    - 2. SPN
    - Differential Cryptoanalysis
    - Linear cryptoanalysis
    - Avalanche effect
    - DES
    - Double encryption
    - Meet in the middle attack

- AES
- Lecture 6
  - Efficency
  - Padding
  - ECB (Electronic Code Block)
  - CBC (Cipher Block Chaining)
  - CTR (Counter mode)
  - Randomness
- Lecture 7
  - Security
  - Iterated hash function
  - Merkel Damgård construction
  - SHA
    - MDx
    - SHA-0 and SHA-1
    - SHA-2
    - Uses of hash
    - Mac (Message authentication code)
    - Authenticated encryption
    - Authenticated encryption with assocated data (AEAD)
    - GCM (Galois Counter Mode)
- Lecture 8
  - Chinese remainder theorem (CRT)
  - Euler function ϕ
  - Properties of $\phi(n)$
  - Fermat theorem
  - Euler theorem
  - Fermat primality test
  - Miller-Rabin test
  - Complexity
- Lecture 9
  - One-way functions
  - Trapdoor one-way functions
  - Public and private keys
  - Hybrid encryption
  - RSA
- Lecture 10
  - RSA
    - Generating p and q

        - Selecting e
        - Fast computation
        - Padding
        - PKCS
        - OAEP (Optimal Asymmetric Encryption Padding)
        - Factorising RSA modulus
        - Miller’s algo
        - Side channel attacks
- Lecture 11
    - Diffie-Hellman key exchange
    - Egemal cryptosystem
    - Elliptic curve
    - Lecture 12
    - RSA signature
    - Elgamal signature in $\mathbb{Z}_p^*$
    - Digital signature algorithm (DSA)
    - ECDSA
    - Digital certificates
    - Public key infrastructure (PKI)
    - CA (Certificate Authority)
    - X.509 standard
- Lecture 13
    - Key managment
    - Forward secrecy
    - Key pre-distribution
    - Signed Diffie-Hellman
    - Needham–Schroeder protocol
    - Freshness
    - Tickets
    - Kerberos
- Lecture 14
    - TLS
    - Alert and change cipher spec protocol
    - Ciphersuites
    - Record protocol
    - Handshake protocol
    - Ephemeral Diffie–Hellman handshake variant
    - RSA handshake variant:
    - Generating session keys:

Attacks
Lecture 15
TLS 1.3
Lecture 16
Email architecture

Lecture 1:

The CIA Triad:

- **Confidentiality**: preventing unauthorised disclosure of information
- **Integrity**: preventing unauthorised (accidental or deliberate) modification or destruction of information
- **Availability**: ensuring resources are accessible when required by an authorised user

Passive Threats

- **Eavesdropping**: The attacker monitors the communication, for example by sniffing packets or tapping a telephone
  wire.
- **Traffic analysis**: The attacker monitors the amount, source and destination of communication.

Active threats

- **Masquerade**: the attacker claims to be a different entity.
- **Replay**: the attacker sends a message which has already been sent.
- **Modification of messages**: the attacker changes messages during transmission.
- **Denial of service**: the attacker prevents legitimate users from accessing resources

Security services and mechanisms

- **Security service**: a processing or communication service to give a specific kind of protection to system resources
- **Security mechanism**: a method of implementing one or more security services

Main security service

- **Peer entity authentication** provides confirmation of the claimed identity of an entity.
- **Data origin authentication** provides confirmation of the claimed source (origin) of a data unit (message).
- **Access control** provides protection against unauthorized use of resources. Access control service is usually provided in combination with authentication and authorisation services.
- **Data confidentiality** protects data against unauthorised disclosure.
- **Traffic flow confidentiality** protects disclosure of data which can be derived from knowledge of traffic flows.
- **Data integrity** detects any modification, insertion, deletion or replay of data in a message or a stream of messages.
- **Non-repudiation** protects against any attempt by the creator of a message to falsely deny creating the data or its contents.
  X.800 talks about nonrepudiation of origin to protect against denial by the sender of a message, and nonrepudiation of receipt to protect against denial by the recipient of a message.
- **Availability service** protects a systems against denial of service. It is not listed in X.800 as a separate service.

Main security mechanisms

- **Encipherment** is the transformation of data in order to hide its information content. Later in the course we look at both public-key and symmetric-key encryption.
- **Digital signature** mechanisms are cryptographic algorithms which transform data using a signing key. The essential property is that signed data can only be created with the signing key. We will look at standard signature schemes.
- I X.800 describes a variety of access control mechanisms including access control lists, passwords, or tokens, which may be used to indicate access rights.

- X.800 describes data integrity mechanisms as “corruption detection techniques” which can be used with “sequence information”. We will look at the example of message authentication codes.
- **Authentication exchange** mechanisms are protocols which exchange information to ensure identity of protocol participants. We will study examples such as TLS later.
- **Traffic padding** is spurious traffic generated to protect against traffic analysis. Traffic padding is typically used in combination with encipherment,
- **Routing control mechanism** is the use of specific secure routes.
- The **notarization mechanism** uses a trusted third party to assure the source or receipt of data. The trusted third party is sometimes called a notary.

Risk management

A key tool in information security management.

1. Identify threats
2. Classify all threats according to likelihood and severity
3. Apply security controls based on cost benefit analysis

Lecture 2:

Basic number theory

Z denotes the set of integers
a divides *b* if there exists a *k* in $\mathbb{Z}$ such that $ak = b$

$ak = 3 * 2 = 6 = b$

An integer is prime if the only positive divisors are 1 and *p*
checking primality for a number *n* can be done by trial division up to $sqrt(n)$.

Basic properties of factors

1. if *a* divides *b* and *a* divides *c* then *a* divides *b*+*c*
2. if *p* is a prime and *p* divides *ab*, then *p* divides *a* or *b*

Example:
$6|18$ and $6|24 \rightarrow 6|42$

Division algorithm

> for a and b in $\mathbb{Z}$, a > b, there exists q and r in Z such that $a = bq + r$ where 0 $\leq$ r < b.

GCD

The value d is the GCD of a and b if all hold:

1. d divides a and b
2. if c divides a and b then c divides d (the greatest)
3. d > 0, by definition of integers

> a and b are relatively prime if $gcd(a, b) = 1$

Euclidean algorithm & extended euclidean algorithm(EEA)

Euclidean algorithm is for finding gcd.
See slides 2 for pseudo-code if you need it
EEA finds integers x and y in $ax + by = d$, we're interested in the case where a and b are
co-prime (x and y = 1)

Modular arithmetic

> b is the residue of a modulo n if $a - b = kn$ for some integer k.

$a \equiv b(\mod n) \leftrightarrow a - b = kn$

Given $a \equiv b(\mod n)$ and $c \equiv d(\mod n)$ then

1. $a + c \equiv b + d(\mod n)$
2. $ac \equiv bd(\mod n)$
3. $ka \equiv kb(\mod n)$

Note:
This means we can always reduce the inputs modulo *n* before performing additions or multiplications

Residue class

> Definition:
> The set $\{r_0, r_1, ...r_{n-1}\}$ is called a complete set of residues modulo n if, for every
> integer a, $a \equiv r_i(\mod n)$ for exactly one r_i

We usually denote this set as the complete set of residues and denote it $\mathbb{Z}$

Notation: a mod n

we write a mod n to denote the value a' in the complete set of residues with
$a\prime \equiv a(\mod n)$
$a = k * n + a\prime$
$0 \leq a\prime < a$

Groups

> a group is a set, G, with a binary operation $\cdot$ satisfying the following properties:

1. Closure: $a \cdot b \in G, \forall a, b \in G$
2. identity: there exists an element 1, so that $a \cdot 1 = 1 \cdot a = a, \forall a \in G$
3. inverse: for all *a*, there exists an element *b* so that $a \cdot b = 1, \forall a \in G$
4. associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c), \forall a, b, c \in G$ (Doesn't matter where you put the parentheses)

In this course we will only consider commutative groups, which are also commutative:

1. $\forall a, b \in G, a \cdot b = b \cdot a$ (order of operands doesn't matter)

Cyclic groups

- The order of a group, G, often written |G|, is the number of elements in G
- we write g^k to denote repeated application of g using the group operation.
- a group element g is a generator for G if $|g| = |G|$
  - the order of an element g, written |g|, is the smallest integer k with $g^k = 1$
- a group is cyclic if it has a generator

Computing inverses modulo n

> the inverse of a , if it exists,, is a value x such that $ax \equiv 1(\mod n)$ and is written $a^{-1} mod\, n$

In cryptosystems, we often need to find inverses so we can decrypt, or undo, certain operations

> Theorem:
> Let $0 < a < n$. Then a has an inverse modulo n iff $gcd(a, n) = 1$.
> (a and n are co-prime)

Modular inverses using Euclidean algorithm

to find the inverse of *a* we can use the Euclidean algorithm, which is very efficient.
Since gcd(a,n) = 1, we can find ax + ny = 1 for integers x and y by Euclidean algorithm.

An actual example of modular inverses.

Since there are really bad resources for this:
From exam 2018:

$8^{-1} \mod 21$

Set up the equation:

$21 = 8(factor) + remainder$

$21 = 8(2) + 5$

shift numbers one to the left

$8 = 5(factor) + remainder$

$8 = 5(1) + 3$
keep shifting till the remainder is 1

$15 = 3(1) + 2$

$3 = 2(1) + 1$

Now, for each line exchange the equation so that the remainder is alone on its side
Labelling each equation in parentheses.

$21 + 8(-2) = 5$ (eq 4)
$8 + 5(-1) = 3$ (eq 3)
$5 + 3(-1) = 2$ (eq 2)
$3 + 2(-1) = 1$ (eq 1)

Look at equation (1). You see it uses the number 2, which is defined in equation (2). Substitute equation (2) in (1):

$3 + (5 + 3(-1))(-1) = 1 \mod 21$

$3 + (5(-1) + 3) = 1 \mod 21$

$3(2) + 5(-1) = 1 \mod 21$

Now we see the number 3, which can be substituted using equation (3):

$(8 + 5(-1))(2) + 5(-1) = 1 \mod 21$
$8(2) + 5(-3) = 1 \mod 21$

Do the same for the number 5, using equation (4):
$8(2) + (21 + 8(-2))(-3) = 1 \mod 21$
$8(2) + 21(-3) + 8(6) = 1 \mod 21$
$8(8) + 21(-3) = 1 \mod 21$

3 is not representable in mod 21, but its absolute value is smaller than our modulus of 21. Substitute -3 with $21 - 3 = 18$.

$8(8) + 21(18) = 1 \mod 21$

We have 21(18), which is 18 times the modulus. Anything multiplied with the modulus is 0:\

$8(8) = 1 \mod 21$

This is our solution. Modular inverses should satisfy $XX^{-1} = 1 \mod n$, and we see that $8 * 8 = 1 \mod 21$.

the set of residues $\mathbb{Z}_p^*$

a complete set of resides modulo any prime *p* with the 0 removed forms a group under multiplication
denoted $\mathbb{Z}_p^*$. It has some interesting properties:

- The order of $\mathbb{Z}_p^*$ is $p-1$
- it is also cyclic.
- it has many generators

Finding a generator for $\mathbb{Z}_p^*$

A generator of $\mathbb{Z}_p^*$ is an element of order $p-1$. To find a generator, we can choose a value and test it like so:

- compute all the distinct prime factors of $p-1$, denoted $f_1, f_2 ... f_r$
- g is a generator as long as $g^{\frac{(p-1)}{f_i}} \not\equiv 1 \mod p$, for $i = 1, 2, .., r$

Should you be tasked to find the order of a generator for $\mathbb{Z}_n^*$ where *n* is not prime
you need to factorize *n* into its prime factors *pq* and then use the rule
$|g| = (p-1)(q-1)$ to find the order.

Fields

> a field is a set, F, with two binary operations
> + and $\cdot$, satisfying:

1. F is commutative group under the + operation, with identity element denoted 0
2. F {0} is a commutative group under the dot operation
3. distributive, $\forall (a, b, c) \in F$

Finite fields GF(p)

For secure communications, we only care about fields with a finite number of elements. \
a famous theorem says that

> Finite fields exist of size p^n for any prime
> jesus christ hun dama går fort frem.
> See slides from lecture 2

- often written Z_p,instead of GF(p)
- multiplication and a addition are done modulo p
- Multiplicative group is exactly Z_p^*
- used in digital signature schemes

For finite fields of order 2^n can use polynomial arithmetic:

$00101101 = x^5 + x^3 + x^2 + 1$

the field is represented by use of a primitive polynomial m(x).
Addition and multiplication is defined by polynomial addition and multiplication modulo m(x).
Division is done efficiently by hardware using shifts.

Boolean values

A Boolean variable x takes the values of 1 or 0 representing true or false

Logical AND: equivalent to multiplication modulo 2:

X_1	X_2	$X_1 \wedge X_2$
1	1	1
1	0	0
0	1	0
0	0	0

Logical OR:

X_1	X_2	$X_1 \vee X_2$
1	1	1
1	0	0
0	1	0

X_1	X_2	$X_1 \vee X_2$
0	0	0

Negation

x	$\neg x$
1	0
0	1

We can also write $\neg x = x \wedge 1$

Lecture 3:

Terminology:

cryptography - the study of designing cryptosystems

cryptanalysis - the study of breaking cryptosystems

Confidentiality and authentication

- Cryptography is the science of secret writing. It concerns transformations of data which depend on a secret called the key
- Cryptography can be used to provide confidentiality and to provide authentication (or integrity)
- When used for confidentiality a key is needed in order to read the message
- When used for authentication a key is needed in order to write the message

Cryptosystems

Consist of:

a set of plaintexts (holding the original message);
a set of ciphertexts (holding the encrypted message);
a set of keys;

a function which transforms plaintext into ciphertext (called encryption or encipherment); an inverse function which transforms ciphertext back into plaintext (called decryption or decipherment).

Symmetric and asymmetric cryptography

Symmetric key cipher (also known as secret key cipher):

- Encryption and decryption keys known only to the sender and receiver.
- Requires a secure channel for transmission of the cryptographic key

Asymmetric key cipher (also known as public key cipher)

- Each participant has a public key and a private key.
- May allow for both encryption of messages and creation of digital signatures.
- We study public key ciphers in a later lecture

Notation for symmetric encryption algorithms

E = Encryption function
D = Decryption function
X = Message or Plaintext
Y = Cryptogram or Ciphertext
K = Shared secret key
Encryption: Y = E(K, X)
Decryption: X = D(K, Y)

Methods of cryptoanalysis

What resources the adversary has available. This includes the computational capability of the adversary. It may also include access to various inputs and outputs of the system.

What the adversary is aiming to achieve. This may be to retrieve the whole of the secret key or it may be as little as distinguishing two message (such as yes or no.)

Exhaustive key search

Can find key with brute-force attack, in which the adversary tries all possible keys. A basic system should have enough keys to make sure this is impossible. To prevent this

is minumun standard

Attack classification

1. **Ciphertext Only attack**: The attacker has available only the intercepted ciphertext.
2. **Known Plaintext attack**: The attacker knows a small amount of plaintext and its ciphertext equivalent.
3. **Chosen Plaintext attack**: The attacker can obtain the ciphertext equivalent of some plaintext which can be selected by the attacker; i.e. the attacker has an “inside encryptor” available.
4. **Chosen Ciphertext attack**: The attacker can obtain the plaintext equivalent of some ciphertext which can be selected by the attacker; i.e. the attacker has an “inside decryptor” available.

Which attacks should be prevented?

A cryptosystem which can be practically attacked using only ciphertext, is generally considered to be highly insecure.
The modern standard is that a cryptosystem should be secure against chosen plaintext and chosen ciphertext attacks.
History shows that chosen ciphertext attacks can often be practical to set up for an attacker.

Kerckhoffs’ Principle

> This principle says that an attacker has complete knowledge of the how the cryptosystem works. The decryption key is the only thing unknown to the attacker.

Alphabets

Roman alphabet: A, B, C, ..., Z. Can use both upper and lower case and sometimes space

Cipher map letter to number, A = 1, B = 2, etc

Statistical attacks

Use the statistical distribution of letters and two and three letter combinations to match with knows statistics in ex. the english language

Basic cipher operations

Transposition: the characters in the plaintext are mixed up with each other (permuted).

> A transposition cipher permutes characters usually in a fixed period d and permutation f.
>
> The key is a pair of d and f. Each block of d is re-ordered using permutation f. There are d! permuations of length d. d = 10 $\rightarrow$ 10! = 3, 628, 800 keys
>
> The distribution of characters is the same for the ciphertext and plaintext. Helps identifying transposition cipher.
>
> if d is small, can be solved by hand using process of anagramming

Substitution: each character (or set of characters) is replaced by a different character (or set of characters).

> Simple substitution ciphers are also called monoalphabetic substitution ciphers.
>
> substitution ciphers permute alphabet characters

Ceasar cipher

moves a letter j places.

Encryption: $c_i = (a_i + j) \bmod n$
Decryption: $a_i = (c_i - j) \bmod n$

Random simple substitution cipher

A cipher which assigns a random character of the alphabet to another character of the alphabet.

Use frequency analysis to crack random simple sub

Map highest frequency to highest freq in english language and start looking for words

Polyalphabetic substitution

use multiple mappings from plaintext to ciphertext

Used to hinder direct frequency analysis

Basiclly splitt plaintext into d blocks and generate d simple substitution and use one on each block

Vigenère cipher

popular form of periodic substitution cipher based on shifted alphabets

ex:

M: AT∇T HE∇T IME∇
K: LOCK LOCK LOCK
E(K, M): LGBC SSBC T∇GJ

Crypto analysis: Identify the period length, Attack separately d different substitution tables

Autokey cipher

the autokey cipher starts off as the Vigenère cipher but once the alphabets defined by the key have been used once, uses the plaintext to define subsequent alphabets. Therefore the autokey cipher is not periodic.

Running key cipher

the running key cipher uses a (practically) infinite set of alphabets from a shared key. In practice the shared key can be extract from a book, when it is often called a book cipher.

Rotar machine

Enigma

Lecture 4

Hill cipher

Polygram cipher using the extended alphabet. Major weakness is that its linear. Easy with known plaintext attacks

Encryption:

Using a $d\ x\ d$ matrix K by block of plaintext P

$C = KP$

Decryption:

Involves multiplying the matrix $K^{-1}$by the block of ciphertext C

$P = K^{-1}C$

Cryptoanalysis

Known plaintext attack. With Cipher text and plaintext we can compute K

Stream ciphers

generation of a keystream of any length.

Stream ciphers are usually symmetric key ciphers: sender and receiver share the same key and can generate the same keystream given the same initialisation value

Synchronous stream cipher:

Sender and receiver generate same keystream and sync its usage

Binary sync stream cipher

for each timeintervall t we define binery sequence s(t) called keystream, binary plaintext p(t) and binary cipertext c(t)

Encryption: $c(t) = p(t) \oplus s(t)$

Decruption$p(t) = c(t) \oplus s(t)$

One time padding

key is truly random seq of characters, all independently generated. Each char is used one time

is non-periodic binary sync stream cipher, is perfectl secrecy

Must have as many keys as messages, is unbreakable. big problem is managing all completely random keys.

Perfect secrecy (Shannon)

messages M, and ciphers C. Pr(M|C) is probability that M was encrypted given C was observed.

Then perfect secrecy is Pr(M|C)=Pr(M)

Lecture 5 Block Ciphers

Block cipher

Symmertric key ciphers where each block of plaintext is encrypted with the same key.

Block ciphers must be **confusion**; involves substitution to make relationship between the key and ciphertext as complex as possible. **Diffusion**; tranformations that dissipate the statistical properties of the plaintext across the ciphertext.

Product cipher

cipher with encruption is formed by applyting several sub-encruption functions

$C = E(P, K) = f_r(...(f_2(f_1(P, K_1), K_2)...), K_r)$

Iterated ciphers

Encryption is divided into r rounds, sub-encryption on all rounds with same round function g, with round keys derived from master key

$W_0 = P, W_1 = g(W_0, K_1), W_2 = g(W_1, K_2), ..., W_r = g(W_{r-1}, K_r), C = W_r$

each round function g must have g^{-1}, where $g^{-1}(g(W, K_i), K_i) = W$.

Two types of iterated ciphers:

1. Feistel ciphers

Splitt plaintext $P = W_0$ into two halves $W_0 = (L_0, R_0)$, for each round perform: $L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, then $C = (L_r, R_r)$

For decryption: $C = (L_r, R_r)$, perform rounds: $L_{i-1} = R_i \oplus f(L_i, K_i), R_{i-1} = L_i$, then $P = (L_0, R_0)$

2. SPN

Block length n must split into m sub-blocks of length l so that $n = lm$

$\pi_s : \{0,1\}^l \to \{0,1\}^l$, operates on sub-blocks of size l

π_P swaps the inputs from $\{1, ..., n\}$, this is similar to the transposition cipher.

MORE STUFF HERE

Differential Cryptoanalysis

chosen plaintext attack.

difference between two input plaintexts can be correlated to the difference between two output ciphertexts

Linear cryptoanalysis

known plaintext attack

Avalanche effect

key avalance (confision): a small change in the key should result in large change in ciphertext

plaintext avalanche (diffusion): small change in plaintext should result in large change in ciphertext

DES

DES is a 16-round Feistel cipher with key length of 56 bits and data block length of 64 bits.

Enc:

1. The 64 bits of P are permuted according to an initial fixed permutation, denoted by IP.
2. After the permutation, 16 rounds of a Feistel operation are applied, denoted by function f. A different 48 bit subkey is used for each round of the f function.

3. After the 16 round operations, a final fixed inverse permutation, denoted by IP−1 , is applied.

→ Output is C

Each round:

1. expand 32 to 48 bits
2. Bitwise modulo two add 48 bits to 48 bit subkey for round
3. Break 48 bits into eight blocks of six bits each
4. Put block i into substitution table i resulting in block of length four
5. apply permutation to resulting 32 bits

Brute force:

Test all 2^kkeys in order to find key K. DES uses 2^{56} keys and right key can be found in 2^{55} trails

Double encryption

With two keys we can encrypt double by: $C = E(E(P, K_1), K_2)$

Meet in the middle attack

We have a single cipher/plain text pair

For every random keys K, K': Check if $E(P, K) = D(C, K')$

Tripple encrypt

with three keys, immune agains meet in the middle attacks. can use $K_1 = K_3, K_2$and still be good

AES

Big dick advance stuff

Lecture 6

Modes can be designed to provide confidentiality and authentication for data

Efficency

some modes allow parallell processing, encrypt and decrypt in parallel.

Padding

ECB and CBC require complete blocks

1. append a single ‘1’ bit to the data string
2. pad the resulting string by as few ‘0’ bits, as needed to complete the block

ECB (Electronic Code Block)

Encryption: $C_t = E(P_t, K)$, Decryption: $P_t = D(C_t, K)$

This can be done in parallel

Properties:

- Randomised: No
- Padding: Required
- Error propagation: error propagation within blocks
- IV: None
- Parallel enc: Yes
- Paralell dec: Yes

CBC (Cipher Block Chaining)

Chains blocks together, Random ‘IV’ is selected and sent with C

Enc: $C_t = E(P_t \oplus C_{t-1}, K)$, where $C_0 = IV$

Dec: $P_t = D(C_t, K) \oplus C_{t-1}$, where $C_0 = IV$

Properties:

- Randomised: Yes

- Padding: Required
- Error propagation: Errors propagate within blocks and into specific bits of next block
- IV: Must be random
- Parallel enc: No
- Paralell dec: Yes

CTR (Counter mode)

Synchronous stream cipher, choose a random nonce N

$O_t = E(T_t, K)$, where $T_t = N||t$ in the concatenation of then nonce

Enc: $C_t = O_t \oplus P_t$

Dec: $P_t = O_t \oplus C_t$

Properties:

- Randomised: Yes
- Padding: Not required
- Error propagation: Errors occur in specific bits of current block
- IV: Nonce must be unique
- Parallel enc: Yes
- Paralell dec: Yes

Randomness

Randomness is difficult. We think in generators that give us random strings:

True random number generator (TRNG) gives a random string with equial probability.

entropy sources

pseaudo random number generator (PRNG) is ish random

deterministic random bit generator (DRBG)

takes seed as input and outputs

security:

backtracking resistance

An attacker who obtains the current state of the DRBG should not be able to distinguish between the output of earlier calls to the DRBG generate function and random strings.

forward prediction resistance

An attacker who obtains the current state of the DRBG should not be able to distinguish between the output of later calls to the DRBG generate function and random strings.

Lecture 7

Hash function H is a public function, so that is is simple and fast to compute and takes as input message m of any length and outputs messages digest H(m) of fixed length

Security

Second-preimage resistant: given a value x1 it should be infeasible to find a different value x2 with H(x1) = H(x2).
Collision resistant: It should be infeasible to find any two different values x1 and x2 with H(x1) = H(x2).
One-way (or preimage resistant): Given a value y it should be infeasible to find any input x with H(x) = y

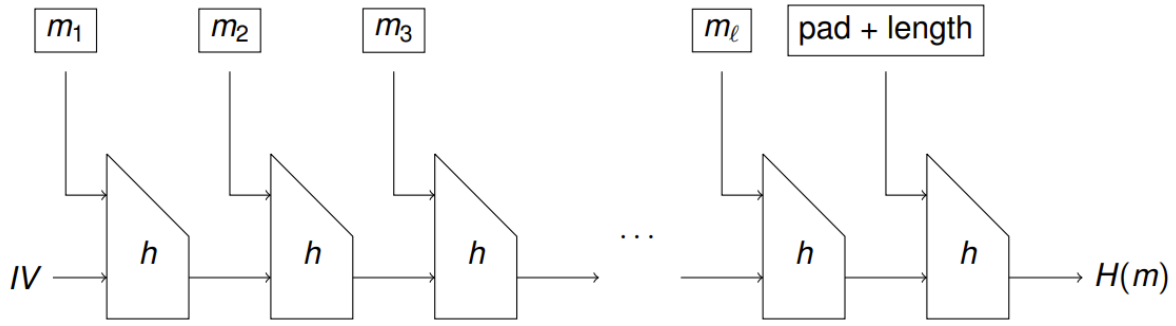
Iterated hash function

splits input into blocks of fixed size and operates on each block seq using the same function

Merkel Damgård construction

Compression function h: takes in two n-bit input string x1 and x2 and produces an n bit output string y

Break message into blocks and pass into h where m_1 is merged with IV and then the messages are merged all the way



Securtity:

if compression function h is collision-resistant, then hash function H is collision resistant

weakness:

Length extension attack: once you have one collision, easy to find more.

Second-preimage attacks not as hard as they should be.

Collisions for multiple messages can be found without much more difficulty than collisions for 2 messages

SHA

MDx

Old and weak

SHA-0 and SHA-1

both are broken, quite recently

SHA-2

	Hash size	Block size	Security match
SHA-224	224 bits	512 bits	2 key 3DES
SHA-512/224	224 bits	1024 bits	2 key 3DES
SHA-256	256 bits	512 bits	AES-128
SHA-512/256	256 bits	1024 bits	AES-128
SHA-384	384 bits	1024 bits	AES-192
SHA-512	512 bits	1024 bits	AES-256

padding:

64-bits when block length is 512-bits, 128 bits when block length is 1024 bits

Uses of hash

Hash functions are not encrypted, not dependent on key and cannot be decrypted

Can be used to authenticate data.

Good for storing passwords: Store salt and password, h=H(pw, salt), store (salt, h)

Mac (Message authentication code)

used for message integrity and authentication

Sender and reciever have same key. $T = MAC(M, K)$, sends M and T then reciever checks that $MAC(M, K) = T$' and verifies that $T = T'$

This is unforgeable, cannot produce T without knowing K

$HMAC(M, K) = H((K \oplus opad)||H((K \oplus ipad)||M))$

opad: fixed string, ipad: fixed string, || denotes concatenation of bit string

HMACs are secure if H is collision resistant or if H is a pseudorandom function. It is designed to resist length extension attacks, even if H is a Merkle-Damgård construction (which are vulnerable to such attacks).

HMAC is often used as a pseudorandom function for deriving keys (since they are deterministic but seem random)

Authenticated encryption

How do you ensure both confidentiality (no one can read your messages) and integrity (the messages aren't tampered with)? A proposed solution is to split your assumed established shared key, K, into two parts - one for encryption and one to obtain a MAC.

There are three possible ways to combine encryption and MACs:

1. Encrypt-and-MAC
   • Encrypt message, apply MAC to message and send the two results
   • C ← Enc(M, K1)
   • T ← MAC(M, K2)
   • Send C||T
2. MAC-then-encrypt
   • Apply MAC to message to get tag. Then encrypt message concatenated with tag and
   send the ciphertext.
   • T ← MAC(M, K1)
   • C ← Enc(M||T, K2)
   • Send C
3. encrypt-then-MAC
   • encrypt message to get ciphertext. Then apply MAC to ciphertext and send the two results
   • C ← Enc(M, K1)
   • T ← MAC(C, K2)
   • Send C||T

Encrypt-then-MAC is the safest of the three. Encrypt-and-MAC is the worst because the MAC cannot guarantee that it wont leak information. MAC-then-encrypt has the same weakness, but at least it encrypts the MAC. Some schemes do, however, provide both confidentiality and integrity with one key

Authenticated encryption with assocated data (AEAD)

AEAD algo is symmetric key

inputs are Message M, Associated data A and shared Key K

AEAD output O contains ciphertext and tag, sender sends O and A, receiver outputs either a M or reportsa fail. Conf for M and Auth for M and A

GCM (Galois Counter Mode)

Block cipher providing AEAD, used in TLS. Combines CTR mode on block cipher with a special keys hash function called GHASH

Algo:

- GHASH uses multiplication in the finite field $GF(2^{128})$
- Inputs are plaintext P, authenticated data A, and nonce N
- Values u and v are the minimum number of 0s required to expand A and C to complete blocks
- Outputs are ciphertext C and tag T. The length of A, len_A, and the length of C, len_C, are 64-bit values
- In TLS the length of T is t = 128 bits and the nonce N is 96 bits The initial block input to CTR mode of E (denoted CTR in diagram) is $J_0 = N \| 0^{31} \| 1$
- The function inc_{32} increments the right-most 32 bits of the input string by 1 modulo 2^{32}

Dec:

- The elements transmitted to the receiver are the ciphertext C, the nonce N, the tag T and the authenticated data A.
- All elements required to recompute the tag T are available to the receiver who shares key K. The tag is recomputed and checked with received tag. If tags do not match then output is declared invalid.
- If the tag is correct then the plaintext can be recomputed by generating the same key stream, from CTR mode, as is used for encryption.

Lecture 8

Chinese remainder theorem (CRT)

Let $d_1, d_2, ..., d_r$ be pairwise relatively prime and $n = d_1, d_2, ..., d_r$. Given any integers c_i tehre exists a unique integer x with $0 \leq x < n$ so that. $x \equiv c_1 (mod\ d_1), x \equiv c_2 (mod\ d_2), x \equiv c_r (mod\ d_r)$

Euler function ϕ

For a + integer n, Euler function $\phi(n)$denoted the number of + integers less then n and relatively prime to n. a and b is relatively prime is the same as $gcd(a,b) = 1$.

Properties of $\phi(n)$

1. $\phi(p) = p - 1$ for prime
2. $\phi(pq) = (p-1) * (q-1)$ for p and q distinct primes

Fermat theorem

$a^{p-1} mod\ p = 1$ for all $1 < a < p - 1$

Euler theorem

$a^{\phi(n)} mod\ n = 1$ if $gcd(a,n) = 1$

Fermat primality test

if a number p is prime then $a^{p-1} mod\ p = 1$ for all a with $gcd(a,p) = 1$

If we examine a number n and find that $a^{n-1} mod\ n \neq 1$ then we know that n is not prime

```
fermat_primality_test(n:test for primality, k:nr of times to test primality):
  for loop k:
    a = random(1, n-1)
    if (a^(n-1) mod n != 1):
      return composite
  return probable prime
```

Miller-Rabin test

```
miller_rabin_test(n):
  u, v = find_u_v(n) # n-1=2^(v)*u, u has to be odd
  a = random(1, n-1)
  b = a^(u) mod n
  if b==1:
    return probable prime
  for loop v-1:
    if b==-1:
```

```
      return probable prime
    b = b^(2) mod n
  return composite
```

Complexity

Algorithm complexity: how long it takes to run a particular algorithm

$f(m) = O(g(m))$

Polynomial: $f(m) = O(m^t), t > 0$

Exponential$f(m) = O(b^m), b > 1$

Brute force is exponential

Problem complexity: what is the best algoritm to solve the problem

classified acoording to the min time and space needed to solve the hardest instance of the problem

Two important problems

Integer factorisation: given an tineger find its prime factors

D**iscrete algorithm problem**: given a prime p and an integer y with $0 < y < p$, find x so that $y = g^x mod\ p$

Lecture 9

One-way functions

If $f(x)$ for any given x is easy to compute but $f^{-1}(y) = x$ for any given y is computationally hard

Trapdoor one-way functions

A trapdoor one-way function f is a one-way function such that given additional information (the trapdoor) it is easy to compute f^{-1}

Public and private keys

public key cryptography = asymmetric cryptography

encryption and decryption keys are different,

encryption public,

decryption private

finding public key from private key most be computationally difficult

Public keys make key management easy, hide one key and send one key out to the world. can contain digital signature

Hybrid encryption

public key crypto is often computationally difficult, and more expencive then symmetric key crypto

typical use of public key crypto is encrypt random key for symmetric key excryption. and then encrypt message with symkey

RSA

keygen

- p, q random distinct primes
- compute n = pq
- select e randomly so that $gcd(e, \phi(n)) = 1$
- compute $d = e^{-1} mod\ \phi(n)$
- public key is pair n and e
- private key is p, q, and d

enc:

- using public key PK=(n, e)
- input any M where M < n
- compute $C = E(M, PK) = M^e mod\ n$

dec:

- using private key SK=(d)
- compute $D = (C, SK) = C^d\ mod\ n = M$

> Proving that RSA works

Lecture 10

RSA

Generating p and q

prime p and q should be random of a chosen length, recommended 1536 bits

easy way of selecting prime. choose a random odd int of correct length, and check if prime. If not imcrement by 2 and check if prime

check prime at size x. → $ln(2^x)$ = (1 prime in of y numbers)

Selecting e

$e = 2^{16} + 1$ is popular choice, but small e is good when practecing for easy computation

Fast computation

square and multiply algorithm

use CRT (chinese remainder theorem) for faster decryption using q and p

- We can use the Chinese Remainder Theorem to decrypt ciphertext C faster with regard to p and q separately.
- First compute:

$$\begin{aligned} M_p &= C^{d \bmod p-1} \bmod p \\ M_q &= C^{d \bmod q-1} \bmod q \end{aligned}$$

- Solve for $M \bmod n$ using the Chinese remainder theorem.

$$\begin{aligned} M &\equiv M_p \pmod p \\ M &\equiv M_q \pmod q \end{aligned}$$

$M = q \times (q^{-1} \bmod p) \times M_p + p \times (p^{-1} \bmod q) \times M_q \bmod n$

Why it works

Note that $d = d \bmod (p-1) + k(p-1)$ for some k.

$$\begin{aligned} M \bmod p &= (C^d \bmod n) \bmod p \\ &= C^d \bmod p \\ &= C^{d \bmod p-1} C^{k(p-1)} \bmod p \\ &= C^{d \bmod p-1} \\ &= M_p \end{aligned}$$

- Similarly $M \bmod q = M_q$
- Therefore $M \bmod n$ is the unique solution to the above two equations.

Padding

unpadded plaintext are weak and can easily be hacked

thererfore we prepair plaintext with padding

typical attack

- Håstad attack
  - if we have three encrupted messages without padding and all use e=3 then we can solve usign CRT

PKCS

Can use PKCS with format [00, 02, Psudo random string, 00, Message]

OAEP (Optimal Asymmetric Encryption Padding)

uses k-bit (often 256) random r and constant d. data, m, is padded withat least byte to make |n|-2k-8 bits where n is rsa modulus n. now merge all this together look at image in lecture

Factorising RSA modulus

if one can factorise n into p and q er have found the private key

Miller’s algo

define u, v so that $ed - 1 = 2^v u$, where u is odd

choose a random a < n and generate sequnce $a^u, a^{2u}, ..., a^{2^v u} (mod\ n)$

notice that $a^{2^v u} \equiv a^{ed-1} \equiv a^{ed} a^{-1} \equiv a a^{-1} \equiv 1 (mod\ n)$

find the square root????????????

Side channel attacks

Timing attacks

- Attacker measures distribution of timing of $a * b\ mod\ n$ on target platform
- Use different size C to find value in crypto
- there are some countermeasures, dummy calculation, randomize message being sent in and other stuff

Lecture 11

Diffie-Hellman key exchange

A and B have both two random values a and b. Using a generator they send g^a and g^b over an insecure channel. now that both have generator value A can take $g^b$and make $(g^b)^a$ and B can make $(g^a)^b$ both resulting in secret key $Z = (g^{ab})$

example:

The public parameters are: $p = 181, g = 2$.

- Alice selects $a = 50$
- Bob selects $b = 33$
- Alice sends $g^{50} \pmod{181} = 116$ to Bob
- Bob sends $g^{33} \pmod{181} = 30$ to Alice
- Alice computes $Z = 30^{50} \pmod{181}$
- Bob computes $Z = 116^{33} \pmod{181}$
- Shared secret is then $Z = 49$

With no authentication we can set up man in the middle attack where one makes two keys and sends to A and B. Can then relay messages between them while still reading all messages

Egemal cryptosystem

uses Diffie-Hellman as base

KeyGen:

select prime p and generator g of $\mathbb{Z}_p^*$

select long-term private key x where x < p-1

compute $y = g^x (mod\ p)$

public key is (p, g, y)

enc:

PuK=(p, g, y)

for any message M, where M < p

choose a random k and compute $g^k (mod\ p)$

$C = Enc(M, PuK) = (g^k (mod\ p), My^k (mod\ p))$

dec:

private key is PrK=x with $y = g^x (mod\ p)$

let $c = (c_1, c_2)$

compute $c_1^x (mod\ p)$

$Dec(c, PrK) = c_2 * (c_1^x)^{-1} (mod\ p) = M$

security:

if discrete log problem is solved then $g^x (mod\ p)$ is found

Elliptic curve

pair (x, y) that satisfy $y^2 = x^3 + ax + b\ (mod\ p)$

Lecture 12

MAC (Message authentication codes) only allow entity with shared secret to generate a valid MAC, providint data integrity and authentication

Only owner of private key can generate a correct signature

Digital signature has three algos:

1. key gen
2. signature gen
3. signature val

Keygen outputs private signing key K_s and public signature verification key K_v

Message is signed with K_s to create $\sigma = SIG(M, K_s)$ and can be checked with a boolean function $Ver(M, \sigma, K_v) = True/False$. Anyone can use K_v

Correctness and unforgablility

weakness/crack attemt: key recovery, selective forgery, esitential forgery

RSA signature

signature keys are genereated same as enc keys.

$n = pq$, then $ed\ mod \phi(n) = 1$, priv key $K_s = (d, p, q)$ and public ver $K_v = (e, n)$

we also need a public hash function h

signing: $\sigma = h(m)^d mod\ n$

veification: $h' = h(m)$, check if $\sigma^e mod\ n = h'$

Elgamal signature in $\mathbb{Z}_p^*$

large p with generator g. Private signing key is x where $x < p - 1$

pub key: $y = g^x mod\ p$, values y, p, g are public

signing: select random $k, k < p - 1$, compute $r = g^k mod\ p$, compute $s = k^{-1}(m - xr) mod (p - 1)$, signing pair $\sigma = (r, s)$

verification: given m and $\sigma = (r, s)$ and verification key y, verify that $g^m \equiv y^r r^s mod\ p$

Digital signature algorithm (DSA)

find prime modulus p of L bits

find prime divisor q of p-1 of N bits

Combinations of L and N are (L = 1024, N = 160), (L = 2048, N = 224), (L = 2048, N = 256), (L = 3072, N = 256)

$g = h^{\frac{p-1}{q}} mod\ p$

keygen:

choose a ranom $x, x < q$

x is the secret key

$y = g^x mod\ p$ is public key

signing:

choose random $k, k < q$

set $r = (g^k mod\ p) mod\ q$

set $s = k^{-1}(H(m) - xr) mod\ q$

signature is pair $\sigma = (r, s)$

verification:

check that $r < q$ and $s < q$

compute $w = s^{-1} mod\ q$

$u_1 = H(m)w\ mod\ q$

$u_2 = rw \ mod \ q$

check whether $(g^{u_1} y^{-u_2} \mod p) \mod q = r$

ECDSA

DSA using elliptic curves, using q from elliptic curve group and p is replaced by elliptic curve group operation. same size signature as DSA and public keys are shorter in ECDSA

Digital certificates

need to validate that public key and owner binding are correct. Can be done by using a CA certification authority with validating that a public key is owned by this validated owner

Public key infrastructure (PKI)

converned with the lifecycle of crypto keys (generation, distribution, storage and destruction)

CA (Certificate Authority)

Create, issue and revoke certificates for subscribers and other CAs

CPS (Certification Practice Statement)

X.509 standard

standard allowing flexible extensions

important fields:

- Version number
- Serial number (provided by CA)
- Signature algo identifier
- Issuer
- Subject
- Public key info
- Validity period

- Digital signature by CA

Validatin certificate is done by checking that the CA siganture is valid and other conditions are correct

Certification path, CA0 which is not known or trusted can be certified by another CA1. And CA1 is certified by CA2, then we get the path $CA_0 \leftarrow CA_1 \leftarrow CA_2$

This can create a hierarchi of PKI. Multiple of these are loaded in the browser. When accessing a new CA they add there PK to the browser.

Revoking Certificate: Checking the Certificate Revocation lists (CRL) periodictlly. And check Online certificate status profocol (OCSP)

Lecture 13

Key managment

We have:

- longterm keys
- Ephemeral keys: one time use
- session keys: session bases

Longterm and ephemeral are used to establish session keys

Hackers are able to eavesdrop, alter messages in a protocol using any info, reroute any message to any user and obtain the value of the session key K_{AB} used in any previous run of the protocol

Forward secrecy

A key agreement protocol proveices perfect forward secrecy if compromises of long-term private keys does not reveal session keys previously agreed using those long-term keys

Key pre-distribution

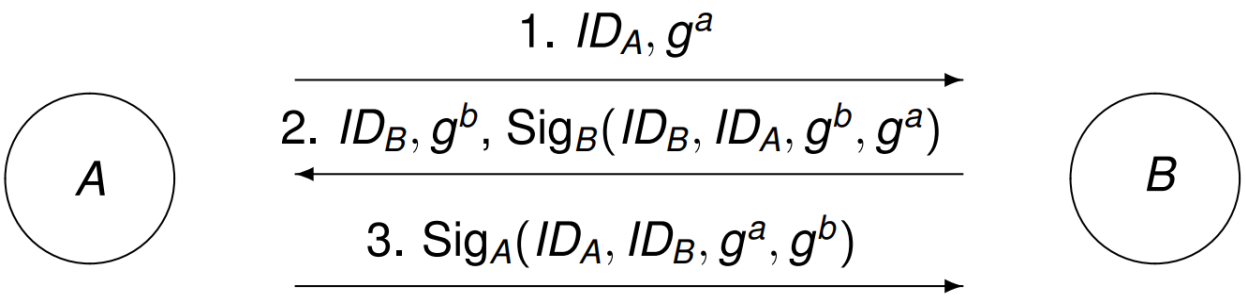
trusted authority (TA) generates and distributes long-term keys to all users when joining system. Does only work on pre-distribution, and are shut off afterwards

Signed Diffie-Hellman

Two parties, A and B, both with identities ID_A and ID_B. Takes place in Group G with generator g. a, b are random values chosen by A and B in the range up to the order of G

$Sig_A(m)$ is a digital signature on message m by A, same for B

Both parties need each others public signature verification keys. This is forward secrecy because the long-term singnig keys are only used for authentication



Needham–Schroeder protocol

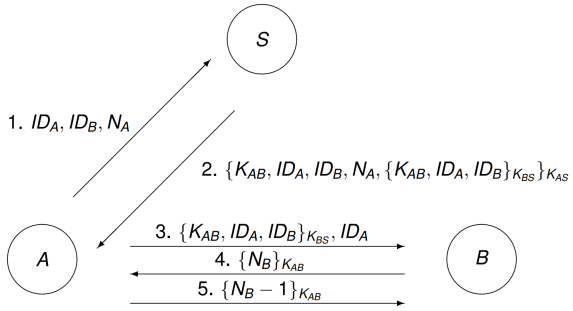
Parties/Principals: A, B, S
- A and B are two parties who want to establish a session key S is the trusted authority

Shared Secret Keys: K_{AS}, K_{BS}, K_{AB}
- A and S share long-term key K_{AS}
- B and S share long-term key K_{BS}
- K_{AB} is the new session key generated by S

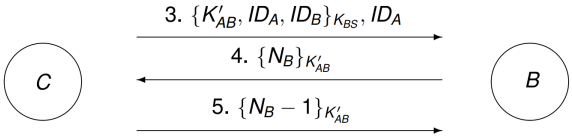
Nonces: N_A, N_B
- Randomly-generated for one-time use (n-once)

$S \to A : M$
- S sends message M to A

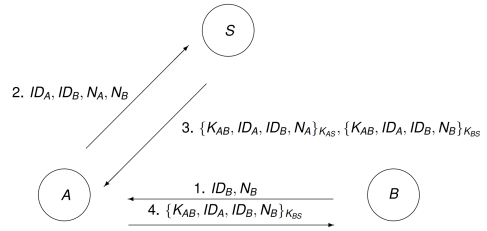
$\{X\}_K$
- Authenticated encryption of message X using the shared secret key K



Replay attack: attacker C obtains a session key K'_{AB} previously established between A and B, C can then predent to be A and is this able to persuade B to use the old key K'_{BA}



TO fix this we can use this format instead



Freshness

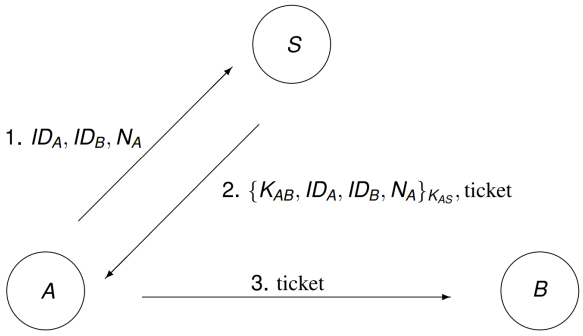
To protect against an replay attack we have to establishe fresh keys for each session

Three basic mechanismes to achieve fresheness:

1. Random challenges (nonce)
2. timestamps
3. counters

Tickets

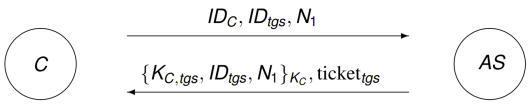
if A wishes to access B it can aske authentication server S for a ticket to allow access to B. Then the ticket can be formated as $\{K_{AB}, ID_A, ID_B, T_B\}_{K_{BS}}$, there T_B is a timestamp. A can access B as long as T_B is valid



Kerberos

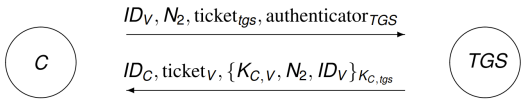
SSO - Single Sign On where user only enters username password to get session access to multiple services using indicitual tickets.

Level 1: Client C interacts with authentication server AS in order to obtain a ticket-granting ticket – happens once for a session (maybe a working day)



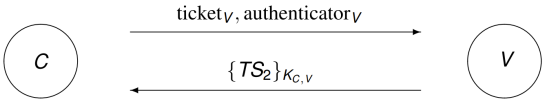
- $\text{ticket}_{tgs} = \{K_{C,tgs}, ID_C, T_1\}_{K_{tgs}}$ for some validity period T_1
- Result: user has ticket-granting ticket which can be used to obtain different service-granting tickets

Level 2: Client C interacts with ticket-granting server TGS in order to obtain a service ticket – happens once for each server during the session



- $\text{ticket}_V = \{K_{C,V}, ID_C, T_2\}_{K_V}$ for some validity period T_2
- $\text{authenticator}_{TGS} = \{ID_C, TS_1\}_{K_{C,tgs}}$ for some timestamp TS_1
- Result: user has service-granting ticket which can be used to obtain access to a specific server

Level 3: Client C interacts with application server V in order to obtain a service – happens each time the client requires service during the session



- $\text{authenticator}_V = \{ID_C, TS_2\}_{K_{C,V}}$ for some timestamp TS_2
- Result: user has secure access to a specific server V

Limitations: Scalability; to add a new service we have to give the key to each other service

Lecture 14

TLS

TLS is a crypto services protocol based on PKI and commonly used on the internett. Used to etablise secure sessions with web servers

secure reliable end-to-end service over TCP

3 protocols:

- TLS Handshake protocol to set up sessions
- TLS alert protocol to signal events such as failures
- TLS change cipher spec protocol to change the cryptographic algorithms

Alert and change cipher spec protocol

Alert protocol handels connections by sending an alert message of various degrees, between “warning alerts”, “close_notify alerts” and “fatal alerts”

change cipher spec prot is used after the handshake prot to indivate commencements of secure traffic

Ciphersuites

is the public algorithms used in the handshake prot and the symmetric algo used in the record prot

Record protocol

Provides two services for TLS:

- Message confidentiality: Ensure that the message contents cannot be read in transit
- Message integrity: Ensure that the receiver can detect if a message is modified in transit

These services can be provided by a symmetric encryption algorithm and a MAC

From TLS 1.2, these services are often provided with authenticated encryption with associated data (AEAD) modes CCM or GCM

The handshake protocol establishes symmetric keys (session keys) to use with these mechanisms

Operation:

- Fragmentation: Each application layer message is fragmented into blocks of 2^{14} bytes or less
- Compression: Optionally applied – default compression algorithm is null
- Authenticated data: consists of the (compressed) data, the header, and an implicit record sequence number
- Plaintext: Compressed data and the MAC, if present
- Session keys for the MAC and encryption algorithms, or AEAD algorithm, are established during the handshake protocol
- The encryption and MAC algorithms are specified in the negotiated *ciphersuite*

Crypto algoritems:

MAC: The algorithm used is HMAC in all TLS versions using a negotiated hash function. SHA-2 is allowed only from TLS 1.2.

Encryption: Either a negotiated block cipher in CBC mode or a stream cipher. Most common block ciphers are AES and 3DES. RC4 originally supported in TLS 1.2. For block ciphers, padding is applied after the MAC to make a multiple of the cipher block size.

AEAD: Allowed instead of encryption and MAC in TLS 1.2. Usually AES in CCM or GCM modes. Authenticated additional data is the header and implicit record sequence number.

Handshake protocol

purpose:

negotiate verion of TLS and crypto algo

establishe shared session key for use in record protocol

authentication of server and client

Four phases:

1. Client and server negotiate version, cipher suite and compression and exchange nonces
2. Server sends certificate and key exchange message (if needed)
3. Cerver sends certificate and key exchange message
4. Client and server start secure communications

Ephemeral Diffie–Hellman handshake variant

Server key exchange: Diffie–Hellman generator and group parameters and server ephemeral Diffie-Hellman value, all signed by server

Client key exchange: Client ephemeral Diffie-Hellman value. This is optionally signed by the client if the client certificate is used

Pre-master secret pms is the shared Diffie–Hellman secret

Provides forward secrecy and therefore recommended today

RSA handshake variant:

No server key exchange

Client key exchange: key transport of pre-master secret, client selects pms, client encrypts pms with servers PuK and sends the cipher text to the server. Server gets pms

Generating session keys:

master secret is defined using the pre master secret: $ms = PRF(pms, \text{“}master\ secret\text{”}, N_c||N_s)$

keying material is generated: $k = PRF(ms, \text{”}key\ expansion\text{”}, N_s||N_c)$

Attacks

BEAST attack: allows attacker to recover plaintext byte by byte by caining IV in CBC mode enc.

CRIME and BREACH attack: side channel attack based on compression.

A padding oracle is a way for an attacker to know if a message in a ciphertext was correctly padded

POODLE attack:

Lecture 15

TLS 1.3

handshake: similar to tls1.2 but fewer round, round record and altert protocol is same as tls 1.2

Some items removed:

- static RSA and DH key exchange
- renegotiation
- SSL 3.0 negotiation
- DSA in finite fields
- data compression
- non-AEAD cipher suites

Some items added:

- Encrypted content type
- 0-RTT mode (from pre-shared key)
- post-handshake client authentication through “certificate verify” signature
- more AEAD ciphersuites

Lecture 16

Email architecture

Message user agent (MUA) connects client to mail system. Uses SMTP to send mail to message submission agent (MSA) and POP or IMAP to retreive mail from message store (MS).

Message handling system (MHS) transfers message from MSA to MS via one or more message transfer agent (MTA)