Exercises II

Computer and Network Security

Emilio Coppa

Exercise: block cipher modes of operation

Evaluate the truth of the following assertions:

- ECB is insecure for encrypting one single block of plaintext
- ECB is parallelizable
- CBC-encryption is parallelizable
- CBC-decryption is parallelizable
- In CBC decryption: a bit flip in the ciphertext corrupts only the current block
- CBC-MAC is insecure for variable-length plaintext messages

Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- ECB is insecure for encrypting one single block of plaintext

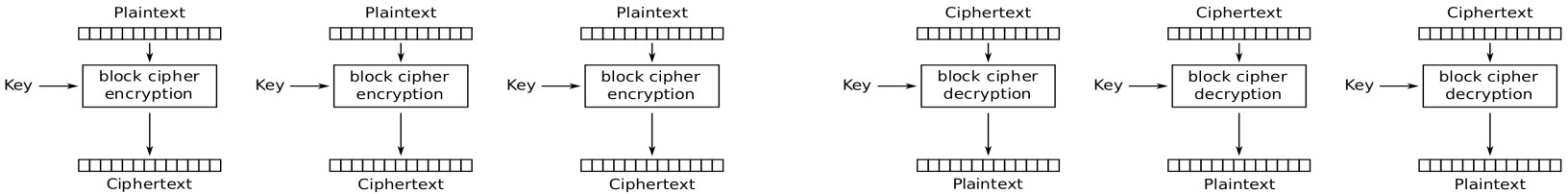
ECB on a single block is equivalent to using just the block cipher over the block, hence the security mainly depend on this component. However, the encrypted block will not be randomized, hence an attacker is able to recognize when you send the same message twice.

Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- ECB is parallelizable

True since blocks are processed independently from each other.

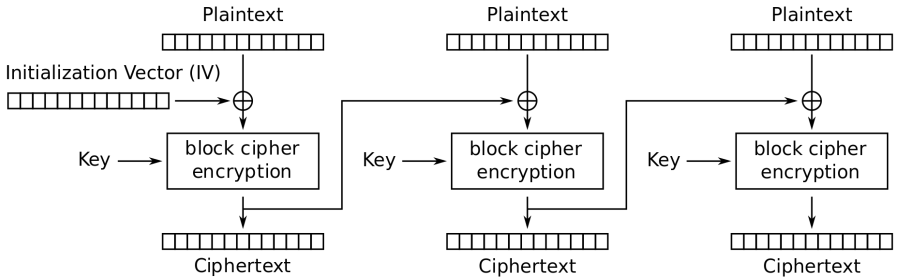


Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- CBC-encryption is parallelizable

False since there is a dependency between subsequent blocks.

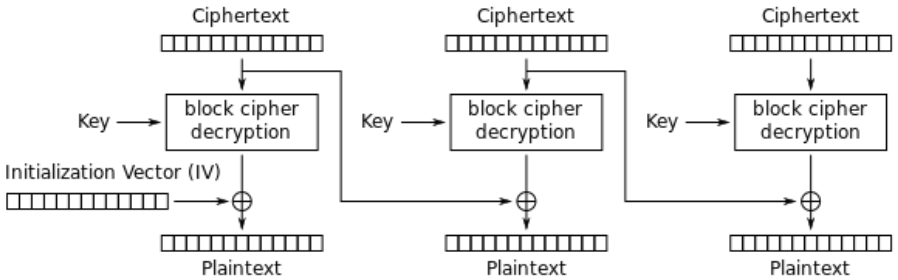


Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- CBC-decryption is parallelizable

True since to get plaintext block i we just need ciphertext block (i-1). We do not need to have previous plaintext blocks.

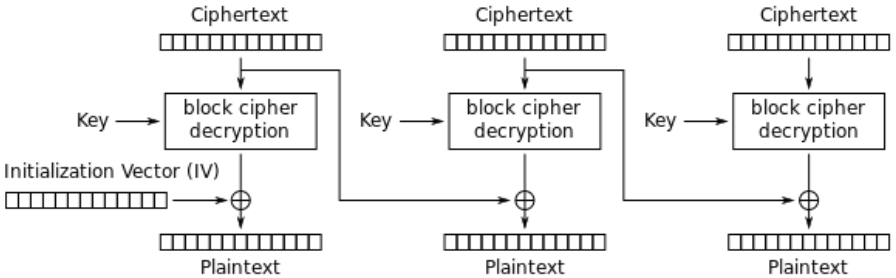


Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- In CBC decryption: a bit flip in the ciphertext corrupts only the current block

False since if there a bit flip in ciphertext block i then it will (unpredictably, due to the block cipher avalanche effect) affect plaintext i and it will (predictably due to bitwise XOR) affect plaintext (i+1)



Exercise: block cipher modes of operation (solution)

Evaluate the truth of the following assertions:

- CBC-MAC is insecure for variable-length plaintext messages

True, from slides on MAC:

If attacker knows correct message-tag pairs (x, t) and (x', t'), where x has L blocks and x' has L' blocks, then he can generate a third (longer) message x" whose tag will also be t':

$$x'' = x_1 \,||\ldots||x_L\,||\,(x'_1 \oplus t)\,||x'_2\,||\ldots||x'_{L'}$$

XOR first block of x' with t and then concatenate x with this modified x'. The resulting (x", t') is valid pair. This works because the xor operation with t "cancel out" the contribution from x. See slides for the proof.

Exercise: AES and CBC mode

1. Describe at your best the meaning of AES128-CBC.
2. If the key is compromised, but the IV has been kept secret, is AES128-CBC still secure? Discuss.
3. How would you change answer to Q2 if the encryption was AES256-CBC?

Exercise: AES and CBC mode (solution)

- Describe at your best the meaning of AES128-CBC.

Block cipher: Advanced Encryption Standard (AES), a cipher originally called Rijndael selected during an evaluation process started by NIST. It runs several rounds (e.g., 11 rounds with 128 key length) and each round performs four main steps: SubBytes, ShiftRows, MixColumn, and addRoundKey. Several of these steps perform computations over Galois Fields.

Key length: AES is using a key length of 128 bits

Operation mode: the block cipher is applied following the operation mode CBC, i.e., Cipher Block Chaining, that:

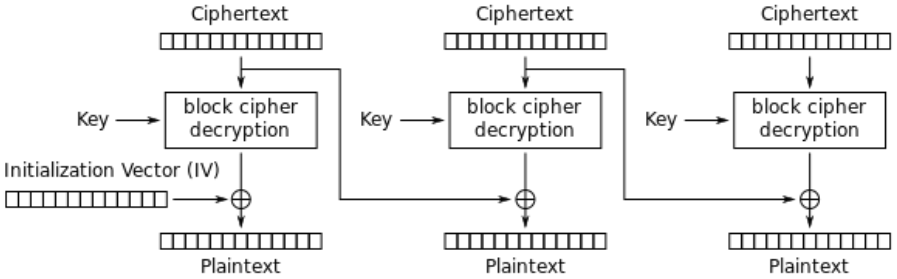
- during encryption: before encrypting a plaintext block with the block cipher, perform a XOR between the plaintext block and the previous ciphertext block (a IV is used when processing the first plaintext block)

- during decryption: after decrypting a ciphertext block with the block cipher, perform a XOR between the decrypted ciphertext block and the previous ciphertext block.

Exercise: AES and CBC mode (solution)

- If the key is compromised, but the IV has been kept secret, is AES128-CBC still secure?

The security is compromised since the attacker is able to decrypt all blocks except the first one:



Given a ciphertext block, we can decrypt it given the key (compromised) and the previous ciphertext block. The first block cannot be decrypted without the IV (although you could still bruteforce it).

Exercise: AES and CBC mode (solution)

- How would you change answer to Q2 if the encryption was AES256-CBC?

If the key is compromised, there is no difference between AES128-CBC and AES256-CBC. Keep in mind that the block size in AES is always 128 bits, regardless the key length. This means that the IV in CBC will be 128 bits, regardless the key length.

Exercise: iterated symmetric encryption

- Describe what iterated encryption (aka iterated cipher) is and describe a general model. Discuss the security improvements that it is possible to obtain, and under what circumstances.
- Describe the Meet-in-the-Middle attack and show how it makes double encryption as secure as single encryption. Can it be used in the case of multiple levels of encryption? Discuss.

Exercise: iterated symmetric encryption (solution)

- Describe what iterated encryption (aka iterated cipher) is and describe a general model. Discuss the security improvements that it is possible to obtain, and under what circumstances.

The main idea is to apply encryption or decryption in sequence using different keys (at least between operation i and i+1). Two popular approaches are:

- EEE mode: $y = E_{k_1}(E_{k_2}(E_{k_3}(x)))$
  E.g. 3-DES. It performs three times encryption in sequence using three different keys.

- EDE mode: $y = E_{k_1}(D_{k_2}(E_{k_1}(x)))$
  It perform encryption with a key #1, decryption with a key #2 (hence this decryption does not “reverse” the previous step), and then again encryption with key #1. EDE mode is used for compatibility.

Exercise: iterated symmetric encryption (solution)

- Describe what iterated encryption (aka iterated cipher) is and describe a general model. Discuss the security improvements that it is possible to obtain, and under what circumstances.

If a block cipher is broken, then iterated encryption does not help significantly. On the other hand, if the block cipher is weak due to a limited key length then this approach may be a way of making it stronger. Using the same key for subsequent step (operation i and i+1) does not increase the robustness.

However, the “increase” of security must be considered carefully: given P keys of size N, the approach will NOT have a “key strength” equivalent to P*N. For instance, 2-DES has a key strength of just 2^57 due to MITM, even when using two keys of size 56 bits, while 3-DES has a key strength of 2^112.

Exercise: iterated symmetric encryption (solution)

- Describe the Meet-in-the-Middle attack and show how it makes double encryption as secure as single encryption. Can it be used in the case of multiple levels of encryption? Discuss.

Meet-in-the-Middle Attack. Assuming to have a pair (x, y):

- 2^56 attempts: $A = \{ \forall\, k_1 \ : E_{k_1}(x) \}$
- 2^56 attempts: $B = \{ \forall\, k_2 \ : D_{k_2}(y) \}$
- Find matching: (a, b) such that $a = b$ where $a \in A, b \in B$
- first key is the one that have generated a, second one is the one that generated b
- Total attempts: 2^56+2^56=2^57

Yes, MITM attack can be performed even with multiple levels of encryption, e.g., for 3-DES see this discussion.

Exercise: about data integrity

1. What are the benefits of introducing keyed hashing?
2. Can the non-repudiation be guaranteed by keyed hashing? Discuss.
3. Is the function $h(x) = x \bmod 2^{256}$ a cryptographic hashing function? (Answers without a proper motivation will be discarded).

Exercise: about data integrity (solution)

- What are the benefits of introducing keyed hashing?

The idea behind keyed hashing is to use a (cryptographic) hashing function to process an input that is given by the “combination” of a message and a secret shared key. This approach allows to implement for instance a Message Authentication Code (MAC) which provides:

- data integrity: a modification of the message is detected since the hash value would be quite different even when changing a single bit
- data origin: the message has been generated by a person owning the secret key. However, we do not know whether the message is “fresh” (hence replay attacks are possible).

Other questions at the exam may ask how to implement keyed hashing, pros and cons of different common approaches, and real-world solutions.

Exercise: about data integrity (solution)

- Can the non-repudiation be guaranteed by keyed hashing? Discuss.

No, keyed hashing cannot provide non-repudiation. Indeed, the approach is based on a secret but shared key hence any person owning the key can generate a checksum for a message. To get non-repudiation, we likely need a scheme based on asymmetric ciphers, as in digital signature approaches.

Exercise: about data integrity (solution)

- Is the function $h(x) = x \bmod 2^{256}$ a cryptographic hashing function?

We need to consider the security requirements:

- Preimage resistance (or one wayness): or a given output $y = h(x)$, it is computationally infeasible to find any input x such that $h(x) = y$, i.e., $h(x)$ is one-way.

  PROBLEM: this function is “easy” to revert since we can easily compute a set of input values that will generate a specific hash value. Modular arithmetic does no make it easy to identify exactly the correct input within this set but we may have domain knowledge to exploit (e.g., if know that the message $< 2^{256}$ then the set contains a single input).

- Second preimage resistance: Given x_1, and thus $h(x_1)$, it is computationally infeasible to find any x_2 such that $h(x_1) = h(x_2)$.
  Collision resistance: It is computationally infeasible to find any pairs (x_1, x_2) where $x_1 \mathrel{!}= x_2$ such that $h(x_1) = h(x_2)$.

  PROBLEM: we can easily find two input with same hash, e.g., v and $(v + 2^{256})$. This can be used to prove no second preimage resistance and no collision resistance.

Exercise: DHKE with three parties

1. Design a sequence of messages, inspired to the classic Diffie-Hellman key exchange, so that three independent parties A, B and C can agree a common session key.
2. Analyze the scheme above with respect to man-in-the-middle attacks (MITM)

Exercise: DHKE with three parties (solution)

- Design a sequence of messages, inspired to the classic Diffie-Hellman key exchange, so that three independent parties A, B and C can agree a common session key.

For example, Alice, Bob, and Carol could participate in a Diffie-Hellman agreement as follows, with all operations taken to be modulo p:

1. The parties agree on the algorithm parameters p and g.
2. The parties generate their private keys, named a, b, and c.
3. Alice computes g^a and sends it to Bob.
4. Bob computes $(g^a)^b = g^{ab}$ and sends it to Carol.
5. Carol computes $(g^{ab})^c = g^{abc}$ and uses it as her secret.
6. Bob computes g^b and sends it to Carol.
7. Carol computes $(g^b)^c = g^{bc}$ and sends it to Alice.
8. Alice computes $(g^{bc})^a = g^{bca} = g^{abc}$ and uses it as her secret.
9. Carol computes g^c and sends it to Alice.
10. Alice computes $(g^c)^a = g^{ca}$ and sends it to Bob.
11. Bob computes $(g^{ca})^b = g^{cab} = g^{abc}$ and uses it as his secret.

Exercise: DHKE with three parties (solution)

- Analyse the scheme above with respect to man-in-the-middle attacks (MITM)

The previous protocol is not secure against an active active that performs a MITM attack. The main issue is that, similar to traditional DHKE, there is no authentication of the public keys sent by {A, B, C}. Hence an attacker can intercept messages from one or more parties, replacing with new messages and the other involved parties have no way of detecting that the attacker is involved in the key exchange.

If authentication is performed at least for some of the entities (e.g., C is a server), then the attacker cannot impersonate these entities but only the remaining ones. This could be a trade-off that some applications may consider.

Exercise: digital signatures

Evaluate the truth of the following assertions:

- A digital signature is obtained by encrypting a message digest by the public key of the signer
- ElGamal signature uses a temporary pair of public/private keys
- DSS signature uses a temporary pair of public/private keys

Exercise: digital signatures (solution)

Evaluate the truth of the following assertions:

- A digital signature is obtained by encrypting a message digest by the public key of the signer

False, since if we compute the signature using the public key of an entity then everybody is able to compute it and thus force the signatures of even never-signed messages.

Exercise: digital signatures (solution)

Evaluate the truth of the following assertions:

- ElGamal signature uses a temporary pair of public/private keys

False, although Elgamal uses a different ephemeral key for each message, it can reuse the same public/private key for different messages since the ephemeral key plays the role of randomizer. Hence the keys are said “per-user”.

Exercise: digital signatures (solution)

Evaluate the truth of the following assertions:

- DSS signature uses a temporary pair of public/private keys

False, for a similar argument as in ElGamal. In DSA (DSS), keys are "per-user". An ephemeral random value is used to randomize the signing process of a message.

Exercise: authentication

1. Alice and Bob agreed the special message T = "The cat is on the moon" and on a 256b secret key K. For one-way authentication Alice sends to Bob $Enc_K(T)$: Bob decrypts, recognizes T and authenticates Alice. Discuss the security of the protocol and possible improvements .
2. Alice and Bob discuss about mutual authentication (not based on third parties). Bob claims that mutual authentication can be simply implemented by sequencing two (opposite) uni-directional authentications; Alice insists that it is better to implement a protocol authenticating both parts, because of possible attacks in the time interval between the two authentications. Describe and motivate your position.

Exercise: authentication (solution)

- Alice and Bob agreed the special message T = "The cat is on the moon" and on a 256b secret key K. For one-way authentication Alice sends to Bob $Enc_K(T)$: Bob decrypts, recognizes T and authenticates Alice. Discuss the security of the protocol and possible improvements.

A crucial weakness of this protocol is that there is not proof on the freshness of the message generated by Alice for Bob. An attacker can easily intercept the message and replay it later. Hence, a first improvement could be to introduce a timestamp and/or a challenge-response mechanism. Also mutual authentication could be desirable to make to protocol more robust.

Another problem is that there is not identify of the initiator and responder in the message. Hence, this message could be reused.

Exercise: authentication (solution)

- Alice and Bob discuss about mutual authentication (not based on third parties). Bob claims that mutual authentication can be simply implemented by sequencing two (opposite) uni-directional authentications; Alice insists that it is better to implement a protocol authenticating both parts, because of possible attacks in the time interval between the two authentications. Describe and motivate your position.

There is no easy and unique answer as it depends on how we design the protocol. In general, we have to require freshness of the messages. However, this could not be enough: e.g., ISO/IEC 9798-2:

- uni-directional authentication:

$$A \rightarrow B : E_K(ts_A, B)$$

- mutual authentication with two unilateral authentications:

$$A \rightarrow B : E_K(ts_A, B)$$
$$B \rightarrow A : E_K(ts_B, A)$$

Exercise: authentication (solution)

$$A \to B : E_K(ts_A, B)$$

vs

$$A \to B : E_K(ts_A, B)$$
$$B \to A : E_K(ts_B, A)$$

We can see that both protocols ensure freshness of the messages by using a timestamp, hence in theory they prevent replay attacks. However, the second protocol can be attacked by C (where C(X) means that C is impersonating X):

$$(1)\ A \to B : E_K(ts_A, B)$$
$$(2)\ B \to C(A) : E_K(ts_B, A)$$
$$(1')\ C(B) \to A : E_K(ts_B, A)$$
$$(2')\ A \to C(B) : E_K(ts'_A, B)$$

A is involved into two sessions:

- in the first one as the initiator
- in the second one as the responder

The attacker is exploiting the confusion in the protocol between the two roles and reuse messages from one session into the other. To fix this problem, the protocol has to provide “injective agreement”, for instance by adding in some way the role of the creator of a message into the message. During the course, we said that for instance the initiator can use K and the responder K+1

Exercise: authentication (solution)

Using a (true) mutual authentication protocol that “binds” the initiator challenge with the responder challenge can mitigate attacks based on role mixup. For instance, in ISO/IEC 9798-2:

$$
\begin{aligned}
&B \to A : N_B \\
&A \to B : E_K(N_A, N_B, B) \\
&B \to A : E_K(N_B, N_A)
\end{aligned}
$$

Exercise: Kerberos

1. Discuss what a "ticket" is and what benefits it provides.
2. Discuss what a "ticket-granting ticket" is and what benefits it provides.
3. What is a realm? Describe when and how it is possible to get services from some principal belonging to another realm.

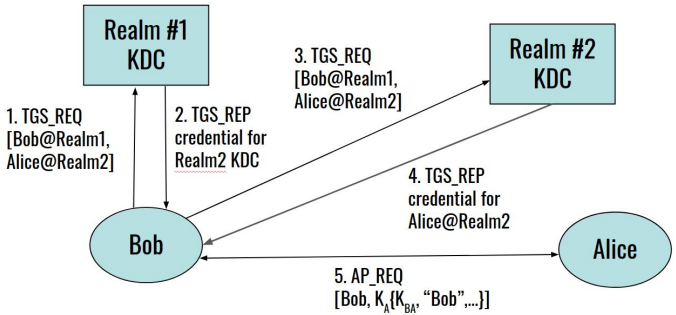
Exercise: Kerberos (solution)

A ticket is a message containing a secret and other pieces of information (such as a ts, a validity of the secret/session/etc) that is returned to client to later obtain a service. The client cannot decrypt the ticket but can only forward to the right entity to submit a request. For instance, in kerberos:

- when the client performs a login, it obtains a ticket-granting ticket from the KDC: this ticket is needed by the client to obtain tickets from the Ticket Granting Server (TGS). Hence, the ticket-granting ticket is sent to client but is “used” by the TGS to obtain a session key for securely replaying to the client.
- when the client wants to use a service, it sends a request to the TGS. The TGS replies with a ticket that the client has to sent to the service when making a request. The ticket can only be decrypted by the service and contain, e.g., the shared secret key between the client and the service.

Exercise: Kerberos (solution)

Since having the same unique “domain” for different networks is not practical both for performance and security (trustiness) reasons, then Kerberos defines the concept of a “realm”: each realm is independent with a different KDC/TGS, however in presence of specific agreements a user of one realm can authenticate and use a service of another realm. Hence a user can perform login with his KDC without ever sending his password to another KDC. See Kerberos slides for details.



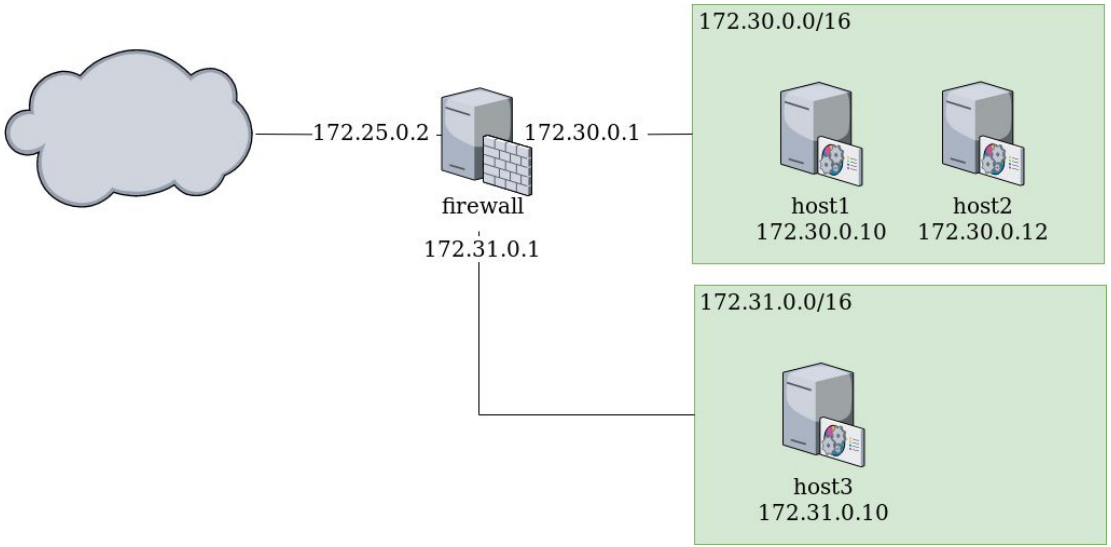
In recent version of Kerberos, realms can be structured in a hierarchy.

For a better discussion of questions 1-3, see the slides.

Exercise: firewall

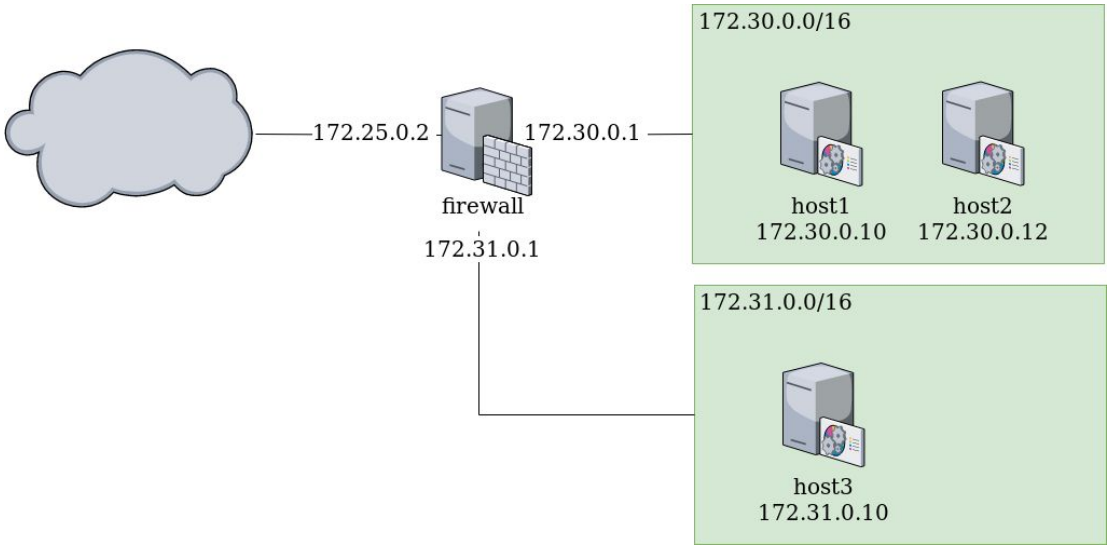
This exercise is taken from SecGroup@Unive: Lab 5 Firewall

Using a Linux VM: install docker (see this) and complete tasks #1 and #2. Do not do it only on paper!



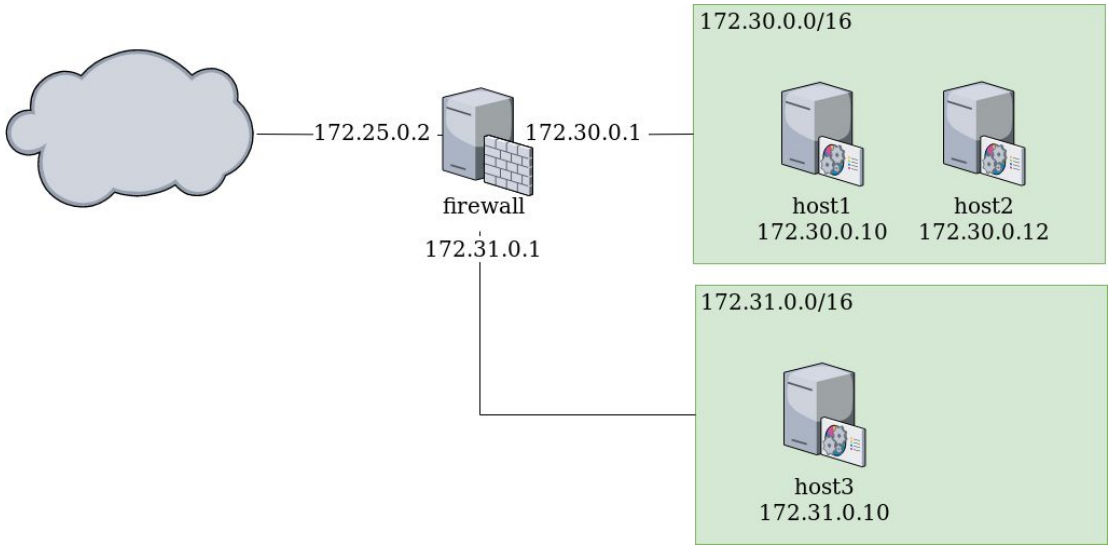
Exercise: firewall

Task #1: suppose that the firewall is forwarding a packet coming from host1 to host2. What chains of tables nat and filter are traversed by the packet?



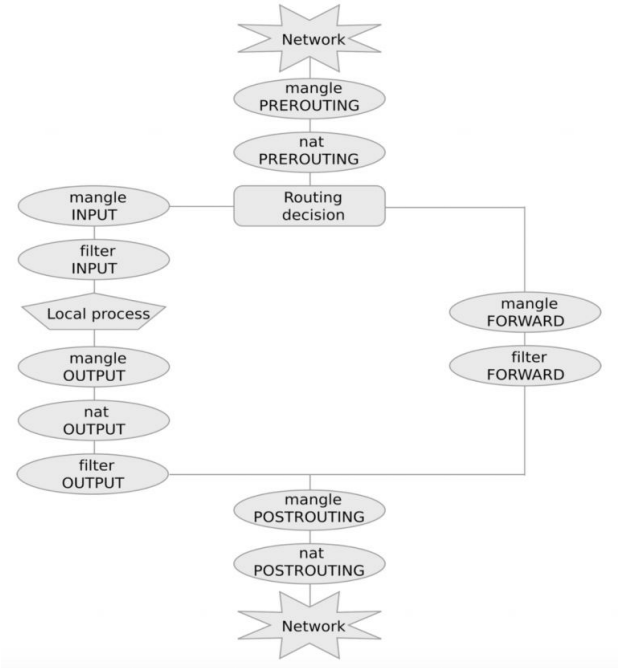
Exercise: firewall

Task #2: Write the rule that drops all tcp connections from host3 (172.31.0.10) to host1 (172.30.0.10) on port 8080.



Exercise: firewall (solution)

Task #1: suppose that the firewall is forwarding a packet coming from host1 to host2. What chains of tables nat and filter are inspected?



Answer:

(nat,PREROUTING),(filter,FORWARD),(nat,POSTROUTING)

Exercise: firewall (solution)

Task #2: Write the rule that drops all tcp connections from host3 (172.31.0.10) to host1 (172.30.0.10) on port 8080.

```
iptables -A FORWARD -p tcp -s 172.31.0.10 -d 172.30.0.10 --dport 8080 -j DROP
```