FIB

Màster en Enginyeria Informàtica (MEI)

Internet, Seguretat i Distribució de Continguts Multimèdia (ISDCM)

Colección de problemas

Seguridad en Internet

SOLUCIONES

Curso 2024-25 Q2

Mayo 2025

Jaime Delgado

Dept. AC

Preguntas Test Cierto/Falso. Indicar si las siguientes afirmaciones son ciertas o falsas.

True/False Test questions. Indicate if the following sentences are true or false.

INTRODUCTION AND CRYPTOGRAPHY

1. Interception, Manipulation, Impersonation and Repudiation are typical threats to security in communications.

☐ True ☐ False

Answer: True.

2. AES and DES are symmetric key encryption mechanisms.

☐ True ☐ False

Answer: True.

3. DES and AES are examples of block-based ciphering mechanisms.

☐ True ☐ False

Answer: True.

4. The DES (*Data Encryption Standard*) for symmetric encryption is obsolete (it is not used) respect to the AES (*Advanced Encryption Standard*).

☐ True ☐ False

Answer:True.

5. It is not possible to use the AES algorithm for digital signature.

☐ True ☐ False

Answer: True.

6. In cryptography, the "confusion" principle is the one that provokes that a small change in the key achieves a big change in the ciphered text.

☐ True ☐ False

Answer: True.

7. In cryptography, the "diffusion" principle is the one that achieves that with a small change in the clear text, a big change in the ciphered text will happen (plain text vs. cipher text independence).

☐ True ☐ False

Answer: True.

8. SubBytes, ShiftRows and MixColumns are examples of permutations of the RSA algorithm for symmetric encryption.

☐ True ☐ False

Answer: False. It is for AES. RSA is asymmetric.

9. ShiftRows, MixColumns and AddRoundKey are examples of permutations of the DES algorithm for symmetric encryption.

☐ True ☐ False

Answer: False. They are for AES.

10. Diffie-Hellman allows sharing a secret key through the communication channel in a secure manner.

☐ True ☐ False

Answer: True.

11. In Diffie-Hellman, two machines A and B manage to share a secret key by interchanging the values α^a (sent by A) and α^b (sent by B), being a$\in G$ and $\alpha^a \in G$, and also b$\in G$ and $\alpha^b \in G$, where G is a *multiplicative finite group* with a generator $\alpha \in G$ known by A and B. On the other hand, a and b are only known by A and B, respectively.

☐ True ☐ False

Answer:True.

12. In Diffie-Hellman, two machines A and B manage to share a secret key by interchanging the values α^a (sent by A) and α^b (sent by B), being a$\in G$ and $\alpha^a \in G$, and also b$\in G$ and $\alpha^b \in G$, where G is a *multiplicative finite group* with a generator $\alpha \in G$ known by A and B. On the other hand, a and b are known by both A and B, but not by the rest of machines.

☐ True ☐ False

Answer: False. **a** and **b** are only known by A and B, respectively.

13. In "asymmetric encryption", the recipient's private key is used to encrypt a message.

☐ True ☐ False

Answer: False. The recipient's public key is used.

14. It is not useful ciphering with symmetric key and sending that key through a public key mechanism.

☐ True ☐ False

Answer: False. Ciphering with symmetric key is more efficient, and sending the symmetric key using PKI solves the key distribution problem.

15. An electronic signature is generated with the public part of the asymmetric key of the signer.

☐ True ☐ False

Answer: False. With the secret part.

16. In security, Hash algorithms are used to interchange symmetric keys.

☐ True ☐ False

Answer: False. They are used to "summarize" (without the possibility of recovering the original) the content of a message to sign.

17. In asymmetric encryption, the secret part of the key may be deduced from the public part of the key.

☐ True ☐ False

Answer: False.

18. In symmetric encryption, the public part of the key may be encrypted with the secret part.

☐ True ☐ False

Answer: False. There are no public keys in symmetric encryption.

19. In RSA, the e value of the public key must be coprime with the value of $\Phi(n)$.

☐ True ☐ False

Answer: True.

20. In RSA, the secret part of the key is calculated directly from the two values of the public part, e and n.

☐ True ☐ False

Answer: False. Module *n* is not used, but numbers *p* and *q* are used for the calculation.

21. In the ElGamal mechanism for asymmetric encryption, we need the secret key in order to calculate the public key.

☐ True ☐ False

Answer: True. The public key Kp is calculated as $Kp = \alpha^{Ks}$, being Ks the secret key, and α a known number.

22. In the ElGamal mechanism for asymmetric encryption, the secret key Ks is calculated as $Ks = \alpha^{Kp}$, being Kp the public key, and α a known number.

☐ True ☐ False

Answer: False. It is just the contrary; i.e., $Kp = \alpha^{Ks}$.

23. In the ElGamal mechanism for asymmetric encryption, the public key Kp is calculated as $Kp = \alpha^{Ks}$, being Ks the secret key, and α a known number.

☐ True ☐ False

Answer: True.

24. In the ElGamal mechanism for asymmetric encryption, to encrypt m we should calculate $c=m*(\alpha^a)^v \bmod g$, where v is a random number chosen by the sender, which is not sent.

☐ True ☐ False

Answer: True.

25. In the ElGamal asymmetric encryption mechanism, to encrypt m we should calculate $c=m*(\alpha^a)^v \bmod g$, where v is a random number chosen by the sender, that is also sent.

☐ True ☐ False

Answer: False. The random number v is not sent.

PUBLIC KEY INFRASTRUCTURE

1. Apart from other responsibilities, a Registration Authority (RA) verifies the information about the user to whom a certificate is to be given.

☐ True ☐ False

Answer: True.

2. In PKI, if we compare OCSP with SCVP, we can say that OCSP needs more complex clients.

☐ True ☐ False

Answer: True.

3. OCSP (Online Certificate Status Protocol) builds the *certification path* of a certificate in order to validate it.

☐ True ☐ False

Answer: False. This is SCVP.

4. SCVP (Server-Based Certificate Validation Protocol) is better that OCSP (Online Certificate Status Protocol) in that it allows clients not to worry about constructing the certification path.

☐ True ☐ False

Answer: True.

5. In SCVP (Server-Based Certificate Validation Protocol), the clients' software needs to implement more tasks than in OCSP (Online Certificate Status Protocol).

☐ True ☐ False

Answer: False. It is the contrary. In SCVP, the server builds the *certification path* of a certificate in order to validate it, thus simplifying the client.

6. SCVP (Server-Based Certificate Validation Protocol) builds the *certification path* of a certificate in order to validate it.

☐ True ☐ False

Answer: True.

7. To generate a trusted time stamp, a Time Stamping Authority uses Hash and PKI technology.

☐ True ☐ False

Answer: True. A hash from the data to be timestamped is calculated and the time stamp is digitally signed.

8. In order to confirm that the time is correct, a *Time Stamping Authority* needs to keep a copy of the document to which a time stamp is assigned.

☐ True ☐ False

Answer: False. It is enough with a Hash of the document.

9. The PKI distributed trust model does not use Certification Authorities.

☐ True ☐ False

Answer: True. It is based on the information collected by the users.

10. In a Plain trust model, the certificate of the CA is self-signed.

☐ True ☐ False

Answer: True.

11. In the case of a hierarchical trust model, a X.509 certificate includes the signature of all the certification authorities of the tree.

☐ True ☐ False

Answer: False. Only the signature of the CA that has issued the certificate.

12. In PKI, a Plain trust model is as a Hierarchical one, but the Plain only has the *root* CA and a unique level of CAs.

☐ True ☐ False

Answer: False. A Plain model only has one CA, the *root* one.

13. The PKI trust list model is controlled by the user.

☐ True ☐ False

Answer: True.

14. Without adding hybrid mechanisms, a user of a CA following the hierarchical model will not trust in a certificate coming from another CA following the plain model.

☐ True ☐ False

Answer: True.

15. The Bridge trust model only works with PKIs using the hierarchical model.

☐ True ☐ False

Answer: False. It may work to relate any kind of infrastructure.

16. The PKI bridge certification trust model is more efficient than the cross-certification one with respect to the number of needed certificates.

☐ True ☐ False

Answer: True.

17. The Bridge trust model implies adding a new Certification Authority.

☐ True ☐ False

Answer: True.

18. A X.509 certificate includes the signature of the owner of the certificate.

☐ True ☐ False

Answer: False. The only signature is that of the certificate issuer.

19. A X.509 certificate includes the public key of its owner.

☐ True ☐ False

Answer: True.

20. A digital certificate includes the signature of the Certification Authority issuing that certificate.

☐ True ☐ False

Answer: True.

21. ASN.1 is a protocol to interchange X.509 certificates.

☐ True ☐ False

Answer: False. ASN.1 is a data representation language used to formalize X.509 certificates.

22. ASN.1 is a data representation language used to formalize XML documents.

☐ True ☐ False

Answer: False. ASN.1 has nothing to do with XML. It is used to represent X.509 certificates.

23. ASN.1 is a data representation language used to formalize X.509 certificates that has been standardized by the IETF (Internet standard).

☐ True ☐ False

Answer: False. It is an ISO standard.

24. ASN.1, standardized by ISO, is the data representation language used to formalize X.509 certificates.

☐ True ☐ False

Answer: True.

25. PKCS#7 (Public-Key Cryptography Standards 7) specifies how to send encrypted documents and their signatures. Some of its concepts are applied in the security of the e-mail.

☐ True ☐ False

Answer: True.

26. In a PKCS#7 message of type `enveloped data`, it is possible to sign over a *hash* or *digest* of the message.

☐ True ☐ False

Answer: False. There is no signature in `enveloped data`.

27. In a PKCS#7 message of type `enveloped data`, the signature algorithm should be RSA.

☐ True ☐ False

Answer: False. There is no signature in `enveloped data`.

28. In a PKCS#7 message of type `signed data`, it is not possible to sign over a *hash* or *digest* of the message.

☐ True ☐ False

Answer: False. The norm recommends to encrypt a *digest*.

29. In a PKCS#7 message, the symmetric key is not sent, since it is supposed to be transfer by non-electronic means.

☐ True ☐ False

Answer: False. What PKCS#7 specifies is how to send, encrypted, the symmetric key.

30. There are PKCS#7 messages that define how to send a public key with a symmetric key.

☐ True ☐ False

Answer: False. PKCS#7 specifies how to send the symmetric key encrypted with an asymmetric key.

31. In PKCS#7, *signedAndEnvelopedData* is just adding *signedData* to *EnvelopedData*.

☐ True ☐ False

Answer: False. *signedAndEnvelopedData* is more complex, it is envelopedData + Doubly encrypted (signature encrypted with content key) message digest.

32. PKCS#7 (Cryptographic Message Syntax) is one of the few (less than 20) PKCS rules.

☐ True ☐ False

Answer: True. There are only rules numbered #1 to #15, and 3 are missing.

33. PKCS#7 (Cryptographic Message Syntax) is one of hundreds of PKCS rules.

☐ True ☐ False

Answer: False. There are only rules numbered #1 to #15, and 3 are missing.

APPLICATION LAYER SECURITY

1. The security of HTTPS may be achieved adding security to TCP.

☐ True ☐ False

Answer: True.

2. There is a new version of the TLS protocol (TLSv1.3) that reduces the number of steps in the *handshake* phase.

☐ True ☐ False

Answer: True.

3. Version 1.3 of the TLS protocol (TLSv1.3) increases, with respect to the previous version, the number of phases in the initial handshake in order to improve the security level.

☐ True ☐ False

Answer: False. It reduces the number of phases to reduce overhead.

4. In TLSv1.3, certificates are only mandatory for the server.

☐ True ☐ False

Answer: True.

5. In TLSv1.3 it is mandatory that the server presents a certificate, while the client's certificate is not always mandatory.

☐ True ☐ False

Answer: True.

6. In TLSv1.3, certificates are no longer used because the symmetric key is encrypted with Diffie-Hellman algorithms.

☐ True ☐ False

Answer: False. Certificates are used to identify parties.

7. TLSv1.3 supports 5 *cipher suites*. TLS_AES_128_GCM_SHA256 is an example of suite.

☐ True ☐ False

Answer: True.

8. TLSv1.3 supports 5 *cipher suites*. All of them are based on AES.

☐ True ☐ False

Answer: False. There is one suite based on CHACHA20 (stream encryption).

9. TLSv1.3 specifies a handshake protocol phase that includes authentication and "cipher suite" negotiation.

☐ True ☐ False

Answer: True.

10. In TLSv1.3, the symmetric key is encrypted with Diffie-Hellman algorithms.

☐ True ☐ False

Answer: True.

11. In TLSv1.3, "cipher suite" negotiation is part of the handshake protocol.

☐ True ☐ False

Answer: True.

12. At least two of the 5 *cipher suites* in TLSv1.3 support stream encryption (not only block encryption).

☐ True ☐ False

Answer: False. There is one suite based on CHACHA20, but no other stream encryptions.

13. TLSv1.3 specifies a handshake protocol phase that includes authentication, but "cipher suite" negotiation is left out of the protocol.

☐ True ☐ False

Answer: False. "Cipher suite" negotiation is part of the handshake protocol.

14. QUIC ("*A UDP-Based Multiplexed and Secure Transport*") does not provide its own security mechanism. Therefore, we need to add a security mechanism over it.

☐ True ☐ False

Answer: False. Security mechanisms similar to TLSv1.3 are provided in QUIC.

15. In order to have a secure protocol, we do not need to add a TLSv1.3 layer to a QUIC ("*A UDP-Based Multiplexed and Secure Transport*") implementation.

☐ True ☐ False

Answer: True. The equivalent security is already provided in QUIC.

16. QUIC standardizes how to combine it with any application protocol.

☐ True ☐ False

Answer: False. For the moment, it only standardizes how to combine with HTTP, which is HTTP/3.

17. One of the objectives in the design of QUIC has been to reduce the latency in the connection establishment phase.

☐ True ☐ False

Answer: True.

18. A limitation of QUIC ("*A UDP-Based Multiplexed and Secure Transport*") is that the security has to be added on top (for example, by adding TLSv1.3).

☐ True ☐ False

Answer: False. TLSv1.3 is included in QUIC.

19. TLSv1.3 functionalities are included in QUIC ("*A UDP-Based Multiplexed and Secure Transport*").

☐ True ☐ False

Answer: True.

20. HTTP/3 is "HTTP over QUIC". Therefore, TCP is not used.

☐ True ☐ False

Answer: True.

21. HTTP/3 is “HTTP over QUIC”. The problem is that the new features from HTTP/2 are lost.

☐ True ☐ False

Answer: False. HTTP/2 features are included.

22. S/MIME is MIME sent over PKCS#7 (`enveloped` data type).

☐ True ☐ False

Answer: True.

XML SECURITY

1. XML Encryption is a W3C Recommendation, but XML Signature it is not.

☐ True ☐ False

Answer: False. XML Signature is also a W3C Recommendation.

2. XML *Encryption* always provides *non-repudiation*.

☐ True ☐ False

Answer: False. We need something more, as a trusted third party.

3. The encrypted content obtained with XML Encryption is always included in the document resulting from the encryption process.

☐ True ☐ False

Answer: False. It could be external and referred to with the element `xenc:CipherReference`.

4. The encrypted content obtained with XML Encryption may be referenced from the XML document itself, instead of being included in the document resulting from the encryption process.

☐ True ☐ False

Answer: True.

5. If ciphered data are included in *XML Encryption*, they are inside the element *CipherValue*, coding in base64 the sequence of encrypted octets.

☐ True ☐ False

Answer: True.

6. A *detached* signature in XML *Signature* means that the `signature` element may be, optionally, kept out of the signed XML document.

☐ True ☐ False

Answer: False. It is always outside the document.

7. A *detached* signature in XML *Signature* means that the `signature` element is outside the signed XML document.

☐ True ☐ False

Answer: True.

8. In XML *Signature*, in the case of a *detached* signature, the `signature` element is at the root of the signed XML element.

☐ True ☐ False

Answer: False. It is outside the document.

9. In XML Signature, in the *detached* case, the `signature` element is not in the same XML document that it is signed.

☐ True ☐ False

Answer: True.

10. In a *detached* XML signature, the signed document is referenced from the `Reference` element, which is part of the `Signature` element.

☐ True ☐ False

Answer: True.

11. In XML Signature, the signature algorithm that generates the `SignatureValue` element is applied to the canonicalization of the `SignedInfo` element of the `Signature` element.

☐ True ☐ False

Answer: True.

12. In XML Signature, both in the *enveloped* and in the *enveloping* cases, the `signature` element is included in the signed XML document.

☐ True ☐ False

Answer: True.

13. In XML Signature, if an element `signature` is part of another element, this means that the signature is *enveloping*.

☐ True ☐ False

Answer: False. It is `enveloped`.

14. In XML Signature, the signature algorithm that generates the `SignatureValue` element is directly applied on the Digest of the XML document.

☐ True ☐ False

Answer: False. It is applied on the canonicalization of the element `SignedInfo`, which includes the canonicalization method, the signature method, the URI and the Digest itself.

15. In XML Signature, the signature algorithm that generates the `SignatureValue` element is applied to the `SignedInfo` element (its canonicalization), which includes more elements than just the Digest of the XML document to sign.

☐ True ☐ False

Answer: True. It is applied on the canonicalization of the element `SignedInfo`, which includes the canonicalization method, the signature method, the URI and the Digest itself.

16. In XML Signature, the signature algorithm that generates the `SignatureValue` element is applied to the canonicalization of the `SignedInfo` element, which includes the canonicalization method, the signature method, the URI and the Digest.

☐ True ☐ False

Answer: True.

17. In XML *Signature*, the *Digest* element is the data on which the signature algorithm is applied.

☐ True ☐ False

Answer: False. The signature algorithm is applied over the *SignedInfo* element, which includes the canonicalization and signature methods, the URI and the *Digest*.

18. In *XML Signature*, the *Digest* of the document to sign is included in the element *SignedInfo*.

☐ True ☐ False

Answer: True.

19. In *XML Signature*, the *Digest* of the document to sign is included in the element *SignedInfo* only when the signature is enveloped or enveloping.

☐ True ☐ False

Answer: False. It is included in all cases.

SPECIFIC SECURITY PROTOCOLS

1. It is not possible to express a SAML token in XML.

☐ True ☐ False

Answer: False. A SAML token (assertion) is expressed and interchanged in XML.

2. *SAML Assertions*" are data structures represented in XML.

☐ True ☐ False

Answer: True.

3. In SAML, users identify themselves in front of a Service Provider that, afterwards, communicates with an Identity Provider in the name of the user.

☐ True ☐ False

Answer: False. Users identify themselves in front of an Identity Provider independently of when they communicate with the Service Provider.

4. In SAML, the user is identified in front of an *Identity Provider*.

☐ True ☐ False

Answer: True.

5. In SAML, the user must initially connect to an Identity Provider before accessing the Service Provider.

☐ True ☐ False

Answer: False. The user may start either with the IdP or the SP.

6. In SAML, the Identity Provider and the Service Provider always communicate directly between them when a user wants access to a resource in the Service Provider.

☐ True ☐ False

Answer: False. In some cases, IdP and SP communication is through the user.

7. In SAML, when a user wants to Access to a resource in a Service Provider, the Identity Provider and the Service Provider communicate through the user.

☐ True ☐ False

Answer: True.

8. A browser implementing SAML will need to transfer information between an Identity provider and a Service provider. For this purpose, it will use the POST method when using HTTP.

☐ True ☐ False

Answer: True.

9. The OAuth 2.0 protocol is fully compatible with its previous version (OAuth 1.0 protocol).

☐ True ☐ False

Answer: False.

10. The OAuth 2.0 protocol is implemented over HTTP, and a response could include JSON information or XML data in its body.

☐ True ☐ False

Answer: True.

11. OAuth is an authorization protocol.

☐ True ☐ False

Answer: True.

12. In OAuth 2.0, the "Authorization code" is one type of "Authorization grant".

☐ True ☐ False

Answer: True.

13. In OAuth 2.0, the "redirect_uri" is part of the "Access token request".

☐ True ☐ False

Answer: True.

14. With the `OAuth 2.0` protocol, the password of the user is never shared with the application.

☐ True ☐ False

Answer: True.

15. The `OAuth 2.0` protocol protects user's password by encrypting it when shared.

☐ True ☐ False

Answer: False. The password is never shared.

16. The `OAuth 2.0` protocol shares the password of the users with the application that acts on their name, but this is done in a secure way so no one else may have access to the password.

☐ True ☐ False

Answer: False. The password is not shared.

17. The `OAuth 2.0` protocol does not normally share the password of the users with the application that acts on their name. When needed, this is done in encrypted mode.

☐ True ☐ False

Answer: False. The password is never shared.

18. Resource Owner, Resource Server and Client Application are examples of the roles defined by OAuth 2.0.

☐ True ☐ False

Answer: True.

19. Two of the features of the `OAuth 2.0` protocol are that allows users to approve an application to act on their behalf and that the password is not shared with the application.

☐ True ☐ False

Answer: True.

20. In OAuth 2.0, the "scope" is part of the "Authorization response".

☐ True ☐ False

Answer: False. It is part of the "Authorization request".

21. A *token endpoint* is used by the client to exchange an authorization grant for an access token, typically with client authentication.

☐ True ☐ False

Answer: True.

22. The body of an access token response in `OAuth 2.0` may include a JSON string.

☐ True ☐ False

Answer: True.

23. OpenID Connect is a simple layer to handle identity on top of OAuth 2.0.

☐ True ☐ False

Answer: True.

24. OpenID Connect provides authorization, so it is very useful to use in combination with OAuth 2.0, which provides authentication.

☐ True ☐ False

Answer: False. It is the other way around.

25. JSON Web Tokens (JWT) are intended as a simplification of XML.

☐ True ☐ False

Answer: False. They are simplifying SAML.

26. A JWT structure is a sequence of ASCII characters.

☐ True ☐ False

Answer: True. It is encoded with BaseUrl.

27. In JSON Web Tokens (JWT), information such as the "Signing/decrypting algorithm" is inside the Header, while the claims are part of the Payload.

☐ True ☐ False

Answer: True.

28. The JWT structure is Base64Url encoded.

☐ True ☐ False

Answer: True.

29. The Encrypted JWT standard recommends content encryption algorithms, but no key encryption algorithms.

☐ True ☐ False

Answer: False. It recommends both.

30. In the Encrypted JWT there is no protected header.

☐ True ☐ False

Answer: False.

31. As for XML, JSON has its own "JSON Web Signature" and "JSON Web Encryption", also standardized by W3C.

☐ True ☐ False

Answer: False. It is standardized by IETF.

32. A JWT structure contains a Header, the Payload and a Signature. All three are mandatory.

☐ True ☐ False

Answer: False. The Signature is optional.

33. An Encrypted JWT (JWE) contains a Protected header, an Encrypted key (symmetric), an Initialization vector, the Encrypted data (ciphertext) and an Authentication tag. The vector and the tag are optional, while the rest are mandatory.

☐ True ☐ False

Answer: True.

PRIVACY & ACCESS CONTROL

1. Privacy controls that reduce Privacy Identifiable Information (PII) are examples of Privacy Enhancing Technologies (PETs).

☐ True ☐ False

Answer: True.

2. Privacy Enhancing Technologies (PETs) help controlling access to Privacy Identifiable Information (PII).

☐ True ☐ False

Answer: True.

3. *Personally Identifiable Information* (PII) is the one that can be freely distributed, since it does not affect the privacy of their owners.

☐ True ☐ False

Answer: False. Just the contrary: it is the information that should be protected.

4. The name of a person may be considered *Personally Identifiable Information* (PII), while their identity card number is not considered PII.

☐ True ☐ False

Answer: False. Both are PII since they identify the user.

5. Health data are not considered Personally Identifiable Information (PII), so they should not be protected.

☐ True ☐ False

Answer: False. They are sensitive PII.

6. Anonymization and pseudonymization tools are examples of PETs (Privacy Enhancing Technologies).

☐ True ☐ False

Answer: True.

7. Anonymization is an example of PET (Privacy Enhancing Technology), while pseudonymization is not.

☐ True ☐ False

Answer: False. Pseudonymization is also a PET.

8. The PDP (*Policy Decision Point*) needs information from the PAP (*Policy Administration Point*) and the PIP (*Policy Information Point*) in order to take an access decision.

☐ True ☐ False

Answer: True.

9. The PEP (*Policy Enforcement Point*) is the one that controls the access to the resources, while the PDP (*Policy Decision Point*) is the one who takes the decision based on the policies and other information.

☐ True ☐ False

Answer: True.

10. In access control systems, the PEP (Policy Enforcement Point) does not always need a PDP (Policy Decision Point).

☐ True ☐ False

Answer: False. Both are modules that work together.

11. Users request access to the PEP (*Policy Enforcement Point*), but the module that works with the policies to allow or deny access is the PDP (*Policy Decision Point*).

☐ True ☐ False

Answer: True.

12. The Discretionary Access Control is based on the use of security labels (levels and categories).

☐ True ☐ False

Answer: False. This is the Mandatory Access Control. The DAC is based on access control lists.

13. The Mandatory Access Control is based on the use of security labels (levels and categories).

☐ True ☐ False

Answer: True.

14. Security labels are used in MAC (Mandatory Access Control), but not in DAC (Discretionary Access Control).

☐ True ☐ False

Answer: True.

15. RBAC means Record Based Access Control, and, as its name indicates, is based in "records" with information of the users.

☐ True ☐ False

Answer: False. The "R" refers to "Role".

16. ABAC is an Access Control mechanism based on attributes.

☐ True ☐ False

Answer: True.

17. XACML (eXtensible Access Control Markup Language) is a W3C standard.

☐ True ☐ False

Answer: False. It is from OASIS.

18. XACML defines elements such as Rule, Policy and PolicySet.

☐ True ☐ False

Answer: True.

19. XACML is useful for the RBAC model, but not for the ABAC one.

☐ True ☐ False

Answer: False.

20. XACML is a XML-based language used to represent licenses.

☐ True ☐ False

Answer: False. It is for representing rules, as privacy policies, for example.

21. XACML is a language to define privacy policies using rules.

☐ True ☐ False

Answer: True.

22. XACML is a standard that allows expressing rules for access control.

☐ True ☐ False

Answer: True.

23. With XACML we are able to specify the rules that control the access to a specific resource.

☐ True ☐ False

Answer: True.

24. XACML is software that allows to control access to the data.

☐ True ☐ False

Answer: False. It is not software, but a XML specification.

25. In XACML, the Rule Combining Algorithm allows to decide how to combine encryption mechanisms.

☐ True ☐ False

Answer: False. It allows to express how to combine several rules in a Policy.

26. XACML includes the element *Rule Combining Algorithm* inside a *Policy*, since there may be several different rules inside a policy.

☐ True ☐ False

Answer: True.

27. XACML is an extension of XML to add security to the communication.

☐ True ☐ False

Answer: False. XACML defines rules, as for example privacy rules.

E-HEALTH XACML

1. The following line of a XACML `Rule`:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
```

states, with the attribute `Effect`, that access will not be allowed if the `Rule` validates.

☐ True ☐ False

Answer: False. Access will be allowed.

2. The following line of a XACML `Policy`:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
```

Indicates with the value of the attribute `Effect` that even if what the rule specifies is not fulfilled, access will be allowed.

☐ True ☐ False

Answer: False. Access is not allowed if what the rule specifies is not fulfilled.

3. The following line of an XML document:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
```

is not a valid start of a XACML `Rule`, because the attribute `Effect` should be in a different XML element.

☐ True ☐ False

Answer: False. It is valid.

4. Given the following fragment of a XACML `Rule`:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Deny">
   …
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         Modify
       </AttributeValue>
       <AttributeDesignator MustBePresent="false"
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </Match>
   …
</Rule>
```

if we request the action “Modify”, it will be authorized.

☐ True ☐ False

Answer: False. The Effect is Deny, so it will not be authorized.

5. Given the previous fragment of a XACML `Rule`, the `Match` element specifies that the XACML Request should exactly have the “Modify” string as “action” in order to apply the indicated “Effect”.

☐ True ☐ False

Answer: True.

6. Given the following fragment of a XACML `Rule`:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
…
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Print
          </AttributeValue>
          <AttributeDesignator MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
…
</Rule>
```

if we request the action Print, it will be authorized.

☐ True ☐ False

Answer: True.

7. The following line of a XACML `Rule`:

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
```

states, with the attribute `Effect`, that access will not be allowed if the `Rule` validates.

☐ True ☐ False

Answer: False. It states that access will be allowed.

8. If we want to specify a XACML rule for a "Researcher" to get a file containing genomic information on a specific chromosome, the corresponding XACML rule could have 3 Match elements, one for the resource (chromosome information), another for the action (get) and another for the user (Researcher, as a Role).

☐ True ☐ False

Answer: True.

DRM

1. The only rights that could have exclusivity or could be remunerated are the economic rights.

☐ True ☐ False

Answer: True.

2. The exclusivity of the moral rights could be negotiated.

☐ True ☐ False

Answer: False. Moral rights could not be transferred, so "exclusivity" does not apply.

3. A difference between XACML and MPEG-21 REL is that only in the second case it is possible to use *grants* (or licenses).

☐ True ☐ False

Answer: True.

4. A Digital Rights Management (DRM) system allows to control the moral rights over a work ("creation").

☐ True ☐ False

Answer: False. Moral rights are inalienable. A DRM system controls exploitation rights.

5. The economic rights of a work include, between others: reproduction, distribution and public communication.

☐ True ☐ False

Answer: True.

6. The moral rights of a work are irrevocable ("unrenounceable").

☐ True ☐ False

Answer: True.

7. It is allowed to sell the economic rights of a work.

☐ True ☐ False

Answer: True.

8. It is allowed to sell the economic rights of a work, but not the moral rights.

☐ True ☐ False

Answer: True.

9. The economic rights could have exclusivity or could be remunerated.

☐ True ☐ False

Answer: True.

10. There are a few parts of the MPEG-21 standard that deal with the representation of licenses for rights management.

☐ True ☐ False

Answer: True.

11. A license, or *grant*, in a Rights Expression Language (REL) identifies a resource to which rights are assigned with XACML.

☐ True ☐ False

Answer: False. XACML is independent from a REL.

12. A *grant* (or License, in general) in a rights expression language is intended to identify a resource, which operations are allowed on it (and under which conditions), but it does not identify for whom the license is, which is done outside.

☐ True ☐ False

Answer: False. Apart from the resource and the operations, it also specifies for whom the license is.

13. A *grant* (or License, in general) in a rights expression language is intended to identify a resource, which operations are allowed on it (and under which conditions), and for whom the license is.

☐ True ☐ False

Answer: True.

14. A *grant* (or License, in general) in a rights expression language is intended to identify a resource, which operations are allowed on it (and under which conditions), but never identifies for whom the license is.

☐ True ☐ False

Answer: False. It also specifies for whom the license is.

15. A *grant* (or License, in general) in a rights expression language is intended to identify a resource, which operations are allowed on it and for whom the license is. However, conditions for the operations are expressed outside.

☐ True ☐ False

Answer: False. It also specifies the conditions.

16. A *grant* (in the context of the Rights Expression Language of MPEG-21) identifies a resource, which operations are allowed on it (and under which conditions), and for whom the grant is.

☐ True ☐ False

Answer: True.

17. W3C's EME (Encrypted Media Extensions) specifies a communication channel between web browsers and DRM (Digital Rights Management) agent software.

☐ True ☐ False

Answer: True.

18. The main purpose of W3C's EME (Encrypted Media Extensions) is to define how to manage keys in web browsers.

☐ True ☐ False

Answer: False. EME mainly defines how to manage DRM in a web environment.

19. The W3C's EME (Encrypted Media Extensions) limits the encryption algorithms to use. It just allows a few of them (less than 10) from a list in the standard. However, the list could be extended.

☐ True ☐ False

Answer: False. EME provides a mechanism to use any encryption algorithm (implemented outside the browser).

EXERCISE 1

Using RSA, B sends to A the result `c=11` of encrypting a message `m`, together with the result `s=18` of signing the same message with a hash function H(m)=m/2.

The public and private keys of A and B are: $(e_A, n_A) = (11, 35)$, $d_A = 11$; $(e_B, n_B) = (3, 22)$, $d_B = 7$

Note: Examples of RSA operations: $x = y^d \bmod n$; $y = x^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

Reasoned and briefly answer the following questions:

1) **Calculate (with the information that the recipient could have) the original message m.**

B has encrypted m with the public key from A, and A must decrypt with his/her private key:

$m = c^{dA} \bmod n_A$; $m = 11^{11} \bmod 35 = 16$.

2) **Verify (with the information that the recipient could have) if the signature is correct.**

B has signed H(m) with his/her private key. To verify the signature, A needs to decrypt s with the public key from B, and check if the result coincides with H(m), i.e. if $H(m)=s^{eB} \bmod n_B$.

$s^{eB} \bmod n_B = 18^3 \bmod 22 = \mathbf{2}$.

$H(m)=m/2$, $H(16)=\mathbf{8}$.

Therefore, the signature is not correct!

PROBLEMA 2

Un usuario A quiere firmar con RSA un mensaje m de sólo un octeto: **00001110**, que envía a otro usuario B.

Los datos disponibles son:

Claves públicas de los usuarios A y B: $(e_A, n_A) = (3, 22)$ $(e_B, n_B) = (11, 35)$

Nota: En la firma RSA, $s = m^d \bmod n$; $m = s^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

Calcular la firma que generará A.

Para calcular s necesitamos m, d y n. m será el octeto 00001110 = 14 en decimal. En este caso concreto en el que A firma con su clave secreta, necesitamos d_A y n_A. n_A vale 22, pero nos falta d_A, que es su clave secreta. La podríamos calcular si tuviésemos $\Phi(n) = (p-1)*(q-1)$, siendo p y q los factores de n. Como n (n=p*q, siendo ambos primos) es muy pequeño, podemos deducir que n=22=2*11 y, por tanto $\Phi(n) = 1*10=10$.

Por tanto, $d_A = e_A^{-1} \bmod \Phi_A(n) = 3^{-1} \bmod 10$. Y calculamos el inverso con la “magic box”:

b	d	k
0	10	-
1	3	3
-3	1	

por lo que el inverso es igual a 10 – 3 = 7, y $d_A = 7 \bmod 10 = 7$.

Ahora simplemente queda aplicar la fórmula $s = m^d \bmod n$

Con los valores que tenemos:

$s = 14^7 \bmod 22 = 105413504 \bmod 22 = 20$

EXERCISE 3

A and B use RSA to exchange encrypted messages.

Their public keys are: $(e_A, n_A) = (11, 35)$ $(e_B, n_B) = (3, 22)$

C manages to find out, for both A and B, the values "p" and "q" with which "n" has been obtained (because the selected prime number is too small, so C has managed to factorize "n").

C intercepts an encrypted message that A sends to B. Specifically, c = 14.

Reasoned and briefly answer the following question:

What is the original message m that has been sent and which C is able to decrypt? Detail all the calculations that C must do to obtain m.

Note: In RSA, $c = m^e \bmod n$, $m = c^d \bmod n$, $d = e^{-1} \bmod \Phi(n)$.

The message c intercepted by C had been calculated as:

$c = 20^3 \bmod 22 = 8000 \bmod 22 = 14$

The values p and q of A and B are:

$n = p * q$; in A: $n = 35 = 5 * 7$; in B: $n = 22 = 2 * 11$

And $\Phi(n)$:

$\Phi(n) = (p-1)*(q-1)$; in A: $\Phi_A(n) = 4*6 = 24$; in B: $\Phi_B(n) = 1*10 = 10$

Taking into account that A sends to B, we calculate m in this way:

$m = 14^{d_B} \bmod 22$, so we need to calculate d_B.

Applying the formula with our values:

$d_B = e_B^{-1} \bmod \Phi_B(n) = 3^{-1} \bmod 10$

And we calculate the inverse with the "magic box":

b	d	k
0	10	-
1	3	3
-3	1	

So the inverse is equal to 10 – 3 = 7, and $d_B = 7 \bmod 10 = 7$.

Then we now have all values to calculate m:

$m = 14^7 \bmod 22 = 105413504 \bmod 22 = 20$

PROBLEMA 4

Recibimos un mensaje encriptado c y su firma s. En concreto, nos llega c=16 y s=14. Usamos RSA.

Las claves pública y privada de nuestro interlocutor son: (e, n) = (11, 35) d = 11

Nuestras claves pública y privada son: (e, n) = (3, 22) d = 7

Nota: Ejemplo de operaciones RSA: $x = y^d \bmod n$; $y = x^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

Calcular el mensaje m original y verificar si la firma es correcta. La función de Hash es H(m)=m.

El originador de m lo ha encriptado con nuestra clave pública y deberemos desencriptar con nuestra clave privada:

$m = c^d \bmod n;\ m = 16^7 \bmod 22 = 14.$

El originador ha firmado H(m) con su clave privada. Para verificar la firma, hemos de desencriptar s con su clave pública y ver si el resultado coincide con H(m)=m:

$m = s^e \bmod n;\ m = 14^{11} \bmod 35 = 14.$

EXERCISE 5

In a RSA environment, we want to impersonate user A by signing a message m=14 as if we were user A.

The public key of A is: (e, n) = (11, 35)

Our public and private keys are: (e, n) = (3, 22) d = 7

Note: Examples of RSA operations: $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n).$

Calculate the private key of user A and the result of the signature.

We calculate the private key d of user A:

$n=p*q;\ 35=5*7.\ \Phi(n)=(p-1)*(q-1)=4*6=24.$

$d = e^{-1} \bmod \Phi(n) = 11^{-1} \bmod 24.$ With magic box:

b	d	k
0	24	-
1	11	2
-2	2	5
11	1	

$$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$$

$d = 11 \bmod 24 = 11.$

Now we sign with A's private key:

$s = m^d \bmod n;\ m = 14^{11} \bmod 35 = 14.$

PROBLEMA 6

Usando RSA, A envía a B el resultado c=5 de encriptar un mensaje m, y el resultado s=4 de firmar el mismo mensaje con una función de Hash H(m)=m/3.

Las claves pública y privada de A y B son: $(e_A, n_A) = (11, 35), d_A = 11; (e_B, n_B) = (3, 22), d_B = 7$

Nota: Ejemplo de operaciones RSA: $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n).$

1) **Calcular (con los datos que pueda tener el receptor) el mensaje m original.**

A ha encriptado m con la clave pública de B, y B debe desencriptar con su clave privada:

$m = c^{dB} \bmod n_B;\ m = 5^7 \bmod 22 = 3.$

2) **Verificar (con los datos que pueda tener el receptor) si la firma es correcta.**

A ha firmado H(m) con su clave privada. Para verificar la firma, B ha de desencriptar s con la clave pública de A, y ver si el resultado coincide con H(m):

H(m) vale H(m)=m/3, H(3)=1.

Calculamos H(m) desencriptando s:

$H(m) = s^{eA} \bmod n_B;\ s = 4^{11} \bmod 35 = 9.$

Como la desencriptación de s es distinta al cálculo de H(m), la firma no es correcta!

EXERCISE 7

We encrypt using ElGamal with a generator $\alpha=3 \in Z_{31}$. The secret key of the sender is 14 and the one for the recipient is 10. To encrypt, v=2 is chosen.

Note: ElGamal expressions: Encrypt: $c = m^*(\alpha^a)^v \in G$ Decrypt: $m = c^*(\alpha^{va})^{-1} \in G$

Reasoned and briefly answer the following question:

If the encrypted message c=27 is received, calculate the original message m.

$\alpha^v = 3^2 \bmod 31 = 9.$

$m= 27 * (9^{10})^{-1} \bmod 31 = 27 * 5^{-1} \bmod 31$, ya que $9^{10} \bmod 31 = 5$

We calculate $5^{-1} \bmod 31$ with "magic box":

```
  b |   d | k
----|----|---
  0 |  31 | -
  1 |   5 | 6
 -6 |   1 |
```

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

Therefore, the result of the magic box is 31-6=25

$m = 27 * 25 \bmod 31 = 24$

EXERCISE 8

We encrypt using ElGamal with a generator $\alpha=3 \in Z_{31}$. Our secret key is 10. As a result of the encryption we send the values **c** and 9.

Note: ElGamal calculations: Encrypt: $c = m^*(\alpha^a)^v \in G$ Decrypt: $m = c^*(\alpha^{va})^{-1} \in G$

Calculate the encrypted message c that we would send if our clear text message were m=24.

$\alpha^v = 9$. Therefore, since $3^v \bmod 31 = 9$, this means that v=2.

$c= 24^*(9^{10}) \bmod 31 = 24 * 5 \bmod 31 = 27$

PROBLEMA 9

Suponer que encriptamos usando ElGamal con un generador $\alpha=3 \in Z_{31}$. La clave secreta de un usuario A es a=17 y la de un usuario B es b=10. B recibe de A: $(\alpha^v, c) = (13, 5)$.

Nota: Fórmulas ElGamal: Encriptar: $c = m^*(\alpha^a)^v \in G$ Desencriptar: $m = c^*(\alpha^{va})^{-1} \in G$

Calcular el valor del mensaje m que A ha enviado.

$m= 5^*(13^{10})^{-1} \bmod 31 = 5 * 5^{-1} \bmod 31 = 5 * 25 \bmod 31 = 1$

ya que $13^{10} \bmod 31 = 5$, y $5^{-1} \bmod 31 = 31\text{-}6 = 25$ con magic box.

```
  b |   d | k
----|----|---
  0 |  31 | -
  1 |   5 | 6
 -6 |   1 |
```

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

EXERCISE 10

We encrypt using ElGamal with a generator $\alpha=2 \in Z_{31}$. The sender's secret key is a = 14 and that of the receiver is a = 9. To encrypt we choose v = 3.

<u>Note</u>: ElGamal calculations: Encrypt: $c = m*(\alpha^a)^v \in G$ Decrypt: $m = c*(\alpha^{va})^{-1} \in G$

If the encrypted message c = 3 is received,

Reasoned and briefly answer the following questions:

1) What other value will we receive in addition to c?

$\alpha^v = 2^3 \bmod 31 = 8.$

2) What is the public key of the receiver that has been used to send the encrypted message?

$\alpha^a = 2^9 \bmod 31 = 16.$

3) Calculate the original message m.

$m= 3 * (8^9)^{-1} \bmod 31 = 3 * 4^{-1} \bmod 31$, since $8^9 \bmod 31 = 4$

We calculate $4^{-1} \bmod 31$ with “magic box”:

```
   b  |  d  | k
  ----|-----|---
   0  | 31  | -
   1  |  4  | 7
  -7  |  3  | 1
   8  |  1  |            bi = bi-2 - (ki-1 * bi-1)
```

Therefore, the result of the *magic box* is 8

$m = 3 * 8 \bmod 31 = 24$

PROBLEMA 11

Tenemos una infraestructura PKI, que llamamos PKIa, que sigue un modelo de confianza plano (su CA se llama CAa), y otra llamada PKIb que sigue un modelo jerárquico. En concreto, PKIb tiene una CAroot de la que dependen dos CAs (CA1a y CA1b). A su vez, de CA1a cuelgan CA2a y CA2b.

Suponer un certificado C con el siguiente contenido:

```
DATA:
      Version: 3 (0x2)
      Serial Number: 7829 (0x1e95)
      Signature Algorithm: md5WithRSAEncryption
      Issuer: I
      Validity: …
      Subject: S
      Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key:
                  Modulus: n
                  Exponent: e

SIGNATURE:
      Certificate Signature Algorithm: md5WithRSAEncryption
      Certificate Signature: s
```

En este certificado C no tenemos los valores I (issuer), S (subject), n (módulo de la clave pública), e (exponente de la clave pública) y s (firma).

Suponer que CA2b emite un certificado C1 para un usuario U1, y que CA1b emite C2 para U2.

Suponer las siguientes claves públicas y secretas:

U1:	n= 35 (5*7)	e= 11	d= 11
U2:	n= 299 (13*23)	e= 13	d= 61
CA1b:	n= 319 (11*29)	e= 3	d= 257
CA2b:	n= 22 (2*11)	e= 3	d= **¿?**

Nota: En RSA, $c = m^e \bmod n$, $m = c^d \bmod n$, $d = e^{-1} \bmod \Phi(n)$.

Contestar razonada y brevemente a las siguientes preguntas:

1) ¿Qué dos campos contiene el elemento Validity del certificado?

"Not before" y "Not after".

2) Calcular **d** para CA2b.

$d = e^{-1} \bmod \Phi(n)$; $\Phi(n) = (p-1)*(q-1) = (2-1) * (11-1)=10$;

Con Magic Box:

```
 b  | d    | k
---|----|--
 0  | 10  | -
 1  |   3 | 3
-3  |   1 |
```

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

Por tanto, d = 10 - 3 = 7.

Suponer que el certificado C es el certificado C1:

3) ¿Cuáles serían los valores **I** y **S**? ¿De quién sería la firma **s**?

I sería CA2b, que es quien emite ("issue") C1. S sería U1, para quien es el certificado ("subject").

La firma s sería de CA2b (firma la CA que emite).

4) ¿Cuáles serían los valores **n** y **e**?

Serían los correspondientes a la clave pública de U1, el dueño del certificado; es decir, n=35 y e=11.

5) ¿Cómo calcularíamos **s**? ¿Cuánto valdría la clave necesaria?

Es la firma de CA2b sobre el certificado C1. Sería algo como $s = C1^d \bmod n$, *siendo n y d la clave privada de CA2b (n=22 y d=7; calculada antes).*

Suponer ahora que U1 quiere enviar un mensaje **m**=2 a U2:

6) ¿Cuánto valdrá el mensaje encriptado **c** enviado?

$c = m^e \bmod n$, *siendo n y e la clave pública de U2 (n=299 y e=13).*

Por tanto, $c = 2^{13} \bmod 299 = 119$.

7) ¿Cuánto valdrá la firma del mensaje **m** (sin aplicar ninguna función de Hash)?

$s = m^d \bmod n$, *siendo n y d (clave secreta) los correspondientes a U1 (n=35 y d=11).*

Por tanto, $s = 2^{11} \bmod 35 = 18$.

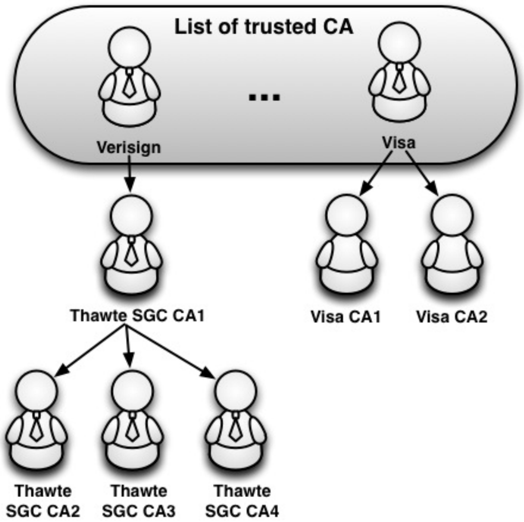
8) Teniendo en cuenta la estructura de las CAs, indicar qué certificados serán verificados y por quién.

U1 tiene que verificar el certificado C2 de U2, porque U1 usa la clave pública de U2. Para ello verifica la firma del issuer, es decir de CA1b. Como no reconoce esa firma, mira quien ha firmado el certificado de CA1b. Al ver que es CAroot, lo acepta, pues el certificado de U1 viene de CAroot también (C1 está firmado por CA2b, el certificado de CA2b está firmado por CA1a, y el de esta última por CAroot).

PROBLEMA 12

Ens volem connectar a una tenda online per comprar una sèrie de productes fent servir el protocol HTTPS. Imaginar que la CA 'ThawteSGC CA2' ha signat el certificat de la tenda online. Un cop el nostre carret és ple de productes, comencem el procés de pagament. Llavors la tenda ens re-adreça a la plana web d'un banc online, que no coneixem, fent servir de nou el protocol HTTPS, però en aquest cas la CA encarregada de signar el certificat del banc és 'Visa CA2'.

Considerant la següent estructura de PKI instal·lada al nostre navegador web:



descriure les operacions de verificació de firma que s'han de dur a terme per autenticar tots dos servidors. Suposar que tots els certificats a recuperar tenen una data d'expiració correcta.

Verificación del servidor 1 (tienda), que tiene un certificado firmado por SGC CA2:

- Verifica que la firma de SGC CA2 es correcta con la clave pública que hay en su certificado emitido por SGC CA1.
- Verifica que la firma de SGC CA1 es correcta con la clave pública que hay en su certificado emitido por Verisign.
- Acepta el certificado de la tienda ya que tiene a Verisign en su lista de Trusted CAs.

Verificación del servidor 2 (banco), que tiene un certificado firmado por Visa CA2:

- Verifica que la firma de Visa CA2 es correcta con la clave pública que hay en su certificado emitido por Visa.
- Acepta el certificado del banco ya que tiene a Visa en su lista de Trusted CAs.

PROBLEMA 13

Tenemos una infraestructura PKI, que llamamos PKIa, que sigue un modelo de confianza plano (su CA se llama CAa), y otra llamada PKIb que sigue un modelo jerárquico. En concreto, PKIb tiene una CAroot de la que dependen 2 CAs (CA1a y CA1b). A su vez, de CA1a cuelgan CA2a y CA2b.

Un usuario U1 envía a U2 un certificado emitido por CAa. U2 tiene un certificado emitido por CA2a. A su vez, U2 envía su certificado a U3, que trabaja con CA1b.

SE PIDE:

1) ¿Qué operaciones (detallar qué firmas se calculan o verifican y con qué claves) hará U2 para decidir si confía en el certificado que le envía U1? ¿Qué decisión tomará?

U2 recibe un certificado de U1 emitido y firmado por CAa.
U2 confía en CA2a, quien le ha emitido su certificado y, por tanto, también confía en CA1a y CAroot (en la PKIb).
U2 verificará la firma del certificado que recibe de U1, es decir la de CAa de PKIa. Como su modelo de confianza es plano, CAa se firma su propio certificado y como U2 no confía en CAa, decidirá no aceptar el certificado.

2) Lo mismo para U3.

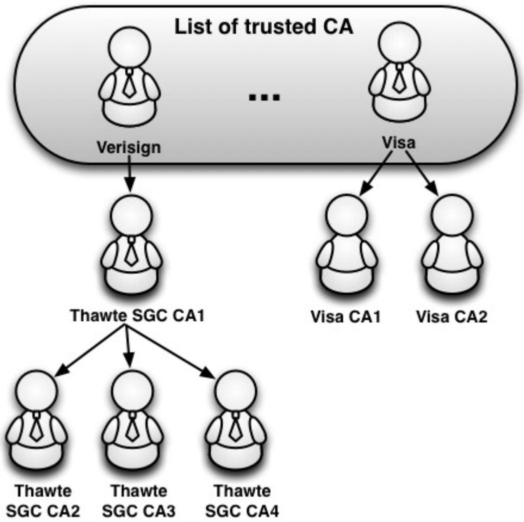
U3 recibe un certificado de U2 emitido y firmado por CA2a.
U3 confía en CA1b, quien le ha emitido su certificado y, por tanto, también confía en CAroot (en la PKIb).
U3 verificará la firma del certificado que recibe de U2, es decir la de CA2a. Como el certificado de CA2a está firmado por CA1a, verifica entonces su firma, viendo que la certifica CAroot, en quien U3 confía, pues está en su árbol. Por tanto, sí aceptará el certificado.

3) Si es U1 quien se comunica con U3, éste no le aceptará el certificado. Sin embargo, U3 confía en U1. ¿Qué puede hacer U3 para que su software acepte el certificado de U1?

Como la solución que se pide sólo puede ser sobre el software de U3 (no a nivel de reconocimiento entre PKIa y PKIb), hemos de usar el modelo "hierarchical trust list" e incluir el certificado de U1 en la lista de certificados de confianza de U3.

PROBLEMA 14

Supónganse las estructuras jerárquicas de Autoridades de Certificación de la figura. La "List of trusted CA" la tienen todos los usuarios mencionados posteriormente.



El usuario A tiene un certificado emitido por Thawte SGC CA2, el usuario B uno de VisaCA1 y el usuario C uno de otra CA ("otherCA") que tiene estructura plana y no está en "List of trusted CA".

Contestar razonada y brevemente a las siguientes preguntas:

1) ¿Qué información contiene el certificado del usuario A?

Básicamente: Emisor (Tawte SGC CA2), Validez, Propietario (A), Clave pública de A, Firma del emisor.

2) El usuario B quiere validar la clave pública de A. ¿Qué certificados y firmas tendrá que verificar? ¿Qué decisión tomará?

Primero verificará la firma de Tawte SGC CA2, quien ha firmado el certificado de A. Para ello tendrá que acceder al certificado de Tawte SGC CA2, donde verá que le firma Tawte SGC CA1 y repetirá el mismo proceso. Finalmente irá al certificado de Verisign. Una vez verificado que la firma es correcta, pasará a validarlo positivamente, pues Verisign está en la lista de Trusted CAs de B.

3) ¿Por qué B no aceptará el certificado de C? Enumerar 3 maneras diferentes de conseguir que B pueda aceptar el certificado de C.

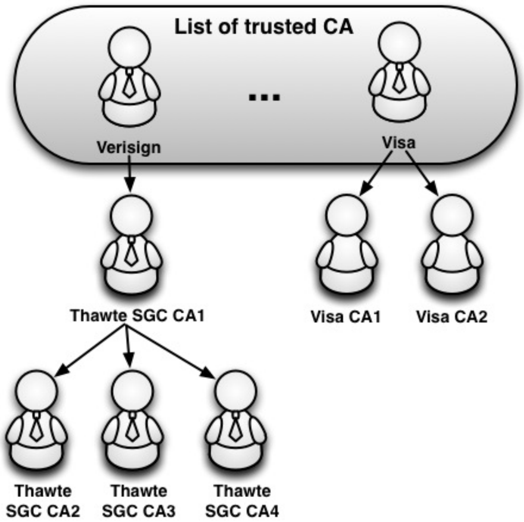
Porque el certificado de C lo firma otherCA, que se autofirma, y no está en la lista de CAs de confianza de B.

Hay 3 opciones para poder aceptar:

- Añadir otherCA a la lista de Trusted CAs de B.
- Añadir una CA Bridge entre Visa CA1 y otherCA.
- Hacer una cross-certification entre Visa CA1 y otherCA.

PROBLEMA 15

Tenemos una infraestructura PKI con un modelo de lista de confianza de un usuario U1. Suponer que tenemos dos estructuras jerárquicas en la lista, tal como indica la siguiente figura.



Suponer además un certificado C con el siguiente contenido (incompleto):

```
DATA:
      Version: 3 (0x2)
      Serial Number: 7829 (0x1e95)
      Signature Algorithm: md5WithRSAEncryption
      Issuer: I
      Validity: Not before: …, Not after: …
      Subject: S
      Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key:
                  Modulus: n
                  Exponent: e
SIGNATURE:
      Certificate Signature Algorithm: md5WithRSAEncryption
      Certificate Signature: s
```

Suponer que “Visa CA2” emite un certificado C1 para un usuario U1, y que “Thawte SGC CA3” emite C2 para U2.

Suponer las siguientes claves públicas y secretas:

U1:	n= 35	e= 11	d= 11
U2:	n= 299	e= 13	d= 61
Visa CA2:	n= 319	e= 3	d= 257
Thawte SGC CA3:	n= 22	e= 3	d= 7

Contestar razonada y brevemente a las siguientes preguntas:

Suponer que el certificado C es el certificado C1:

1) ¿Cuáles serían los valores **I** y **S**? ¿De quién sería la firma **s**?

I sería Visa CA2, que es quien emite (“issue”) C1. Su sería U1, para quien es el certificado (“subject”).

La firma s sería de Visa CA2 (firma la CA que emite).

2) ¿Cuáles serían los valores **n** y **e**?

Serían los correspondientes a la clave pública de U1, el dueño del certificado; es decir, n=35 y e=11.

Suponer ahora que U1 quiere enviar un mensaje **m** a U2:

3) ¿Quién generaría la firma del mensaje **m**?

U1, que es quien envía el mensaje.

4) ¿Quién generaría la clave pública de U2 con la que U1 encripta el mensaje **m**?

La CA que ha emitido el certificado de U2, es decir Thawte SGC CA3.

5) Teniendo en cuenta la “List of Trusted CA” del principio:
   a. Indicar qué certificados y firmas serán verificados y por quién.
   b. ¿Qué decisión tomará U1? Es decir, ¿enviará o no el mensaje **m** a U2?

U1 tiene que verificar el certificado C2 de U2, porque U1 usa la clave pública de U2. Para ello verifica la firma del issuer, es decir de Thawte SGC CA3. Como no reconoce esa firma, mira quién ha firmado su certificado. Ve que es Thawte SGC CA1, a quien tampoco reconoce. Entonces verifica el certificado de quien ha firmado el de Thawte SGC CA1, que es Verisign. Entonces lo acepta, pues está en su lista de confianza.

PROBLEMA 16

Tenemos un sistema S que permite enviar mensajes XML como éste:

```
<?xml version='1.0'?>
      <PasswordInfo xmlns='http://example.org/password'>
            <Name>Josep Roca</Name>
            <Password>
                  <Alphanumeric>1234ab</Alphanumeric>
                  <Reminder>Home town</Reminder>
            </Password>
      </PasswordInfo>
```

Queremos que el usuario A del sistema S envíe dichos mensajes XML con el *valor (contenido) del elemento* `Alphanumeric` encriptado. El destinatario de los mensajes es el usuario B.

Contestar razonada y brevemente a las siguientes preguntas:

Parte I

1) Si usamos XML Encryption, dar la estructura del fichero XML que recibirá el destinatario del mensaje.

```
<?xml version='1.0'?>
<PasswordInfo xmlns='http://example.org/password'>
      <Name>Josep Roca</Name>
```

```
        <Password>
              <Alphanumeric>
                    <EncryptedData
                    Type='http://www.w3.org/2001/04/xmlenc#Content'
                    xmlns='http://www.w3.org/2001/04/xmlenc#'>
                          <CipherData>
                                <CipherValue>xxxxxx</CipherValue>
                          </CipherData>
                    </EncryptedData>
              </Alphanumeric>
              <Reminder>Home town</Reminder>
        </Password>
  </PasswordInfo>
```

2) ¿Cuáles serían los caracteres concretos que se habrían encriptado con el ejemplo anterior?

El valor alfanumérico del password: “1234ab”.

Parte II

Suponer que la encriptación se realiza usando ElGamal con un generador α=3 $\in Z_{31}$. La clave secreta de A es a=10 y la de B es b=17. B recibe de A: (α^v, c) = (13, 5).

Nota: Fórmulas ElGamal: Encriptar: $c = m*(\alpha^a)^v \in G$ Desencriptar: $m = c*(\alpha^{va})^{-1} \in G$

3) Calcular el mensaje (irreal para el caso concreto de XML Encryption) que A ha enviado.

$m= 5*(13^{17})^{-1} \bmod 31 = 5 * 17^{-1} \bmod 31 = 5 * 11 \bmod 31 = 24$

ya que $13^{17} \bmod 31 = 17$, y $17^{-1} \bmod 31 = 11$ con magic box.

b	d	k
0	31	-
1	17	1
-1	14	1
2	3	4
-9	2	1
11	1	

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

Parte III

Suponer que A quiere firmar con XML Signature el nuevo fichero XML resultante de la XML encryption.

Nota: La estructura de la firma XML es:

```
<Signature Id="..." xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>...</SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```

4) ¿Qué información es la que se firma y por tanto se incluirá dentro del elemento SignedInfo?

```
<SignedInfo>
  <CanonicalizationMethod Algorithm="..."/>
  <SignatureMethod Algorithm="..."/>
  <Reference URI="...">
    <Transforms>
      <Transform Algorithm="..."/>
    </Transforms>
    <DigestMethod Algorithm="..."/>
    <DigestValue>.../DigestValue>
  </Reference>
</SignedInfo>
```

Parte IV

Imaginemos el caso, también irreal, de que la información a firmar sólo contiene el octeto **00001110**, y supongamos que se realiza una firma RSA con los siguientes datos:

Claves públicas de los usuarios A y B: $(e_A, n_A) = (3, 22)$ $(e_B, n_B) = (11, 35)$

Nota: En la firma RSA, $s = m^d \bmod n$; $m = s^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

5) Calcular la firma RSA que generará A.

Para calcular s necesitamos m, d y n. m será el octeto 00001110 = 14 en decimal. En este caso concreto en el que A firma con su clave secreta, necesitamos d_A y n_A. n_A vale 22, pero nos falta d_A, que es su clave secreta. La podríamos calcular si tuviésemos $\Phi(n) = (p-1)*(q-1)$, siendo p y q los factores de n. Como n (n=p*q, siendo ambos primos) es muy pequeño, podemos deducir que n=22=2*11 y, por tanto $\Phi(n) = 1*10=10$.
Por tanto, $d_A = e_A^{-1} \bmod \Phi_A(n) = 3^{-1} \bmod 10$. Y calculamos el inverso con la “magic box”:

```
       b    |     d     |     k
-----------------------------------
       0    |    10     |     -
       1    |     3     |     3
      -3    |     1     |
```

por lo que el inverso es igual a 10 – 3 = 7, y d_A = 7 mod 10 = 7.

Ahora simplemente queda aplicar la fórmula $s = m^d \bmod n$

Con los valores que tenemos:

$$s = 14^7 \bmod 22 = 105413504 \bmod 22 = 20$$

6) ¿En qué elemento de la XML Signature se transmitirá el valor calculado en el apartado anterior?

`SignatureValue`

PROBLEMA 17

Tenemos un sistema S que permite enviar mensajes XML como éste:

```
<?xml version='1.0'?>
      <PasswordInfo xmlns='http://example.org/password'>
            <Name>Josep Roca</Name>
            <Password>
                  <Alphanumeric>1234ab</Alphanumeric>
                  <Reminder>Home town</Reminder>
            </Password>
      </PasswordInfo>
```

Queremos que un usuario del sistema S envíe dichos mensajes XML con el *elemento* `Alphanumeric` encriptado.

Contestar razonada y brevemente a las siguientes preguntas:

1) Si usamos XML Encryption, dar la estructura del fichero XML que recibirá el destinatario del mensaje.

```
<?xml version='1.0'?>
<PasswordInfo xmlns='http://example.org/password'>
      <Name>Josep Roca</Name>
      <Password>
         <EncryptedData
           Type='http://www.w3.org/2001/04/xmlenc#Element'
           xmlns='http://www.w3.org/2001/04/xmlenc#'>
                <CipherData>
                      <CipherValue>xxxxxx</CipherValue>
                </CipherData>
         </EncryptedData>
```

```
            <Reminder>Home town</Reminder>
        </Password>
    </PasswordInfo>
```

2) ¿Cuáles serían los caracteres concretos que se habrían encriptado con el ejemplo anterior?

El elemento alfanumérico del password: “<Alphanumeric>1234ab</Alphanumeric>”.

PROBLEMA 18

Suponer que se quiere firmar un mensaje m con XML Signature.

Nota: La estructura de la firma XML es:

```
<Signature Id="..." xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>...</SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```

¿Qué información es la que se firma (es decir, qué campos contiene) y por tanto se incluirá dentro del elemento `SignedInfo` de `Signature`?

```
  <SignedInfo>
    <CanonicalizationMethod Algorithm="..."/>
    <SignatureMethod Algorithm="..."/>
    <Reference URI="...">
      <Transforms>
        <Transform Algorithm="..."/>
      </Transforms>
      <DigestMethod Algorithm="..."/>
      <DigestValue>.../DigestValue>
    </Reference>
  </SignedInfo>
```

EXERCISE 19

Given the following XML document resulting from encryption with XML Encryption:

```
<?xml version='1.0'?>
<PasswordInfo xmlns='http://example.org/password'>
      <Name>John Smith</Name>
      <Password>
            <Alphanumeric>
                  <EncryptedData
                  Type='http://www.w3.org/2001/04/xmlenc#Content'
                  xmlns='http://www.w3.org/2001/04/xmlenc#'>
                        <CipherData>
                              <CipherValue>xxxxxx</CipherValue>
                        </CipherData>
                  </EncryptedData>
            </Alphanumeric>
            <Reminder>My dog</Reminder>
      </Password>
</PasswordInfo>
```

¿Which element or value has been encrypted? Reason the answer indicating the taken assumptions. **Provide an example of a possible version of this XML document before encryption.**

```
<?xml version='1.0'?>
<PasswordInfo xmlns='http://example.org/password'>
      <Name>John Smith</Name>
      <Password>
            <Alphanumeric>4321z</Alphanumeric>
            <Reminder>My dog</Reminder>
      </Password>
</PasswordInfo>
```

Since the `Type` attribute of the `EncryptedData` element is `Content`, we know that the **value** of the `Alphanumeric` element has been encrypted.

PROBLEMA 20

Suponer que se quiere firmar con XML Signature un documento XML en el que se ha encriptado con XML Encryption uno de sus elementos.

La estructura de dicha firma XML es:

```
<Signature Id="..." xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>...</SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
</Signature>
```

1) ¿De qué tipo de firma (Detached, Enveloped o Enveloping) se trata? ¿Cómo lo podemos saber?

   Detached o Enveloped. No tenemos el elemento Object.

2) ¿Dónde estaría dentro de la Signature el documento XML encriptado que estamos firmando?

   La URI del elemento `Reference` de `SignedInfo` apuntaría al documento a firmar.

```
<SignedInfo>
  <CanonicalizationMethod Algorithm="..."/>
  <SignatureMethod Algorithm="..."/>
  <Reference URI="...">
    <Transforms>
      <Transform Algorithm="..."/>
    </Transforms>
    <DigestMethod Algorithm="..."/>
    <DigestValue>.../DigestValue>
  </Reference>
</SignedInfo>
```

EXERCISE 21

We have a hospital system HS that allows sending XML documents like this one:

```
<?xml version='1.0'?>
      <HealthRecord xmlns='http://example.org/healthrecord'>
            <Name>Ami Ill</Name>
            <Status>
                  <IllnessStandardCode>AB123</IllnessStandardCode>
                  <DaysInHospital>3</DaysInHospital>
            </Status>
      </HealthRecord>
```

Hospital H wants to send the health record of patient P to her using system HS. In order to ensure confidentiality, the value *(content)* of the `Status` element is encrypted inside the XML document.

Reasoned and briefly answer the following questions:

Part I

1) If we use XML Encryption, provide the structure of the XML document that patient P will receive. Include at least the "type" attribute and the "CipherValue" element of the "EncryptedData" element.

```
<?xml version='1.0'?>
< HealthRecord xmlns='http://example.org/healthrecord'>
     <Name>Ami Ill</Name>
     <Status>
          <EncryptedData
               Type='http://www.w3.org/2001/04/xmlenc#Content'
               xmlns='http://www.w3.org/2001/04/xmlenc#'>
                    <CipherData>
                         <CipherValue>xxxxxx</CipherValue>
                    </CipherData>
          </EncryptedData>
     </Status>
</HealthRecord>
```

Part II

Assume that we encrypt using ElGamal with a generator $\alpha=2 \in Z_{31}$. The secret key of the sender is 14 and the one for the recipient is 9. The received information is: $(\alpha^v, c) = (8, 3)$.

Note: ElGamal expressions: Encrypt: $c = m*(\alpha^a)^v \in G$ Decrypt: $m = c*(\alpha^{va})^{-1} \in G$

2) **Calculate** the message sent.

$m= 3*(8^9)^{-1} \bmod 31 = 3 * 4^{-1} \bmod 31 = 3 * 8 \bmod 31 = 24$

given that $8^9 \bmod 31 = 4$, and $4^{-1} \bmod 31 = 8$ with magic box.

b	d	k
0	31	-
1	4	7
-7	3	1
8	1	

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

3) **Justify** why the message sent calculated is not realistic for the case introduced in Part I.

We are encrypting several characters. In particular, those corresponding to the content of the *status* element. For the example given in the introduction it would contain more than 80 characters.

Part III

Still in the system HS environment, imagine that we want to sign the *HealthRecord* XML document, which in a particular case is a set of characters that correspond to the decimal number 200. Let's assume that we apply a RSA signature with the following data:

Public keys of H and P: $(e_H, n_H) = (13, 299)$ $(e_P, n_P) = (3, 319)$

Note 1: In RSA signature, $s = m^d \bmod n;\ m = s^e \bmod n;\ d = e^{-1} \bmod \Phi(n)$.

Note 2: 299 = 13*23; 319=11*29

4) **Calculate** the RSA signature generated when signing the *HealthRecord* XML document.

To calculate s we need m, d and n. m is 200. The hospital H is who signs. Therefore, its private key is used, so we need d_H and n_H. n_H is 299, but we need d_H, i.e., the secret key. We could calculate it with $\Phi(n) = (p-1)*(q-1)$, but we need the p and q factors of n. Since n (n=p*q, being p and q primes) is, in this case, very small, we have n=299=13*23. So, $\Phi(n) = 12*22=264$. Then, $d_H = e_H^{-1} \bmod \Phi(n_H) = 13^{-1} \bmod 264$. We calculate the inverse with the "magic box":

```
b   |  d   | k
----|------|----
  0 | 264  | -
  1 |  13  | 20
-20 |   4  | 3
 61 |   1  |                bi = bi-2 – (ki-1 * bi-1)
```

The result is $d_H = 61$.
Finally, to calculate the signature, $s = m^d \bmod n$
With the values we already have:

$$s = 200^{61} \bmod 299 = 96$$

EXERCISE 22

We have a hospital system HS that allows sending XML documents like this one:

```
<?xml version='1.0'?>
      <HealthRecord xmlns='http://example.org/healthrecord'>
            <Name>Ami Ill</Name>
            <Status>
                  <IllnessStandardCode>AB123</IllnessStandardCode>
                  <DaysInHospital>3</DaysInHospital>
            </Status>
      </HealthRecord>
```

Hospital H1 has sent one of the records of patient P to her. In turn, patient P, not satisfied with the treatment in Hospital H1, wants to send her health record to Hospital H2 in order to get a second opinion. She also uses system HS. In order to ensure confidentiality, the value *(content)* of the `HealthRecord` element is encrypted using XML Encryption.

Reasoned and briefly answer the following questions:

Part I

1) Provide the structure of the XML document that patient P sends. Include at least the "type" attribute and the "CipherValue" element of the "EncryptedData" element.

```
<?xml version='1.0'?>
< HealthRecord xmlns='http://example.org/healthrecord'>
      <EncryptedData
            Type='http://www.w3.org/2001/04/xmlenc#Content'
            xmlns='http://www.w3.org/2001/04/xmlenc#'>
                  <CipherData>
                        <CipherValue>xxxxxx</CipherValue>
                  </CipherData>
      </EncryptedData>
</HealthRecord>
```

Part II

We encrypt using ElGamal with a generator $\alpha=2 \in Z_{31}$. The secret key of Patient P is 14, the one for Hospital H1 is 11, and the one for Hospital H2 is 9. The received information is: $(\alpha^v, c) = (8, 3)$.

Note: ElGamal expressions: Encrypt: $c = m*(\alpha^a)^v \in G$ Decrypt: $m = c*(\alpha^{va})^{-1} \in G$

2) **Calculate** the message ("health record") sent.

$m = 3*(8^9)^{-1} \bmod 31 = 3 * 4^{-1} \bmod 31 = 3 * 8 \bmod 31 = 24$

given that $8^9 \bmod 31 = 4$, and $4^{-1} \bmod 31 = 8$ with magic box.

```
  b |  d  | k
----|-----|---
  0 | 31  | -
  1 |  4  | 7
 -7 |  3  | 1
  8 |  1  |
```

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

Part III

Still in the system HS environment, imagine that we want to sign the *HealthRecord* XML document with XML Signature. We want to generate the signature in the *enveloped* mode. We are going to use RSA as signature algorithm.

3) If the specific encrypted XML document that we are going to sign is the one obtained in Part I of this exercise, **indicate** where is the XML Signature going to be included?

Since we are in the enveloped mode, the *Signature* element should be inside the original XML document. For example, it could be a new element inside the *HealthRecord* one, added after the *EncryptedData* element.

4) **Describe** the exact data on which the RSA algorithm is going to be applied: Is the XML document of the example? Is a hash/digest of that document? Something different (if so, please specify)?

The RSA algorithm is applied over a canonicalization of the *SignedInfo* element, which includes the *CanonicalizationMethod*, the *SignatureMethod* and the *Reference* (an URI and the Digest of the XML document).

5) Let's assume that the information to be signed is a message m=14. Calculate the obtained signature s using RSA, with the following data:

Public keys of hospitals and patient: $(e_{H1}, n_{H1}) = (11,35)$ $(e_{H2}, n_{H2}) = (13,299)$ $(e_P, n_P) = (3, 22)$

Note: In RSA signature, $s = m^d \bmod n;\ m = s^e \bmod n;\ d = e^{-1} \bmod \Phi(n)$.

To calculate s we need m, d and n. m is 14. The patient P is who signs. Therefore, its private key is used, so we need d_P and n_P. n_P is 22, but we need d_P, i.e., the secret key. We could calculate it with $\Phi(n) = (p-1)*(q-1)$, but we need the p and q factors of n. Since n (n=p*q, being p and q primes) is, in this case, very small, we have n=22=2*11. So, $\Phi(n) = 1*10=10$.
Then, $d_P = e_P^{-1} \bmod \Phi(n_P) = 3^{-1} \bmod 10$. We calculate the inverse with the "magic box":

```
 b  |  d   | k
----|------|----
  0 |  10  | -
  1 |   3  | 3
 -3 |   1  |
```

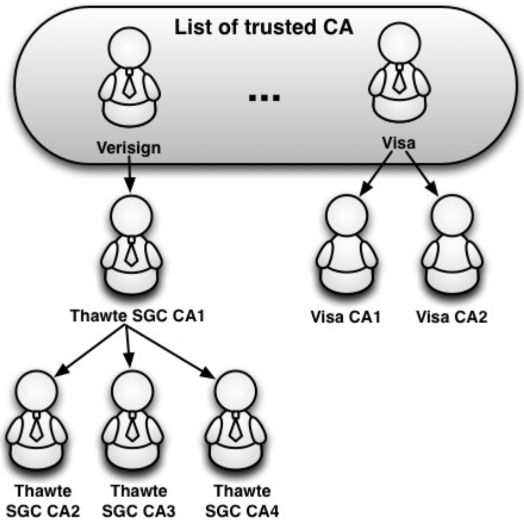
$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

The result is $d_P = 10-3=7$.
Finally, to calculate the signature, $s = m^d \bmod n$
With the values we already have:

$s = 14^7 \bmod 22 = 105413504 \bmod 22 = 20$

Part IV

We have a PKI with users U1 and U2 having a trust list model. Let's assume that we have two hierarchical structures in the list, as indicated in the following figure.



“Visa CA2” issues a certificate C1 for U1, while “Thawte SGC CA3” issues C2 for U2. U2 wants to send an encrypted message to U1.

6) Who needs to accept a certificate from another one?

U2 needs the certificate C1 from U1 since it needs to encrypt the message with the public key of U1.

7) With the trusted CAs list of the previous figure, will U2 decide to send the message? Why?

Yes, because the CAs that issued both certificates have their roots in the trusted list.

EXERCISE 23

We are sending XML documents, such as the one in the following example, with confidential information:

```
<?xml version='1.0'?>
      <AccountInfo xmlns='http://example.org/bank'>
            <Name>John Smith</Name>
            <AccountNumber>12345678</AccountNumber>
            <Balance>
                  <Available>
                        <Amount>999</Amount>
                        <Currency>EUR</Currency>
                  </Available>
                  <Date>2015/12/18</Date>
            </Balance>
      </AccountInfo>
```

We want to encrypt part of this information to be sent through the network. We decide to just encrypt the `Balance` element.

Reasoned and briefly answer the following questions:

Part I

1) If we use XML Encryption, provide the resulting XML document. Assume the unknown values, but at least include the “Type” attribute.

```
<?xml version='1.0'?>
<AccountInfo xmlns='http://example.org/bank'>
      <Name>John Smith</Name>
      <AccountNumber>12345678</AccountNumber>
      <EncryptedData
            Type='http://www.w3.org/2001/04/xmlenc#Element'
            xmlns='http://www.w3.org/2001/04/xmlenc#'>
```

```
                    <CipherData>
                         <CipherValue>xxxxxx</CipherValue>
                    </CipherData>
          </EncryptedData>
     </AccountInfo>
```

Part II

We want to sign the previously encrypted document with XML Signature.

Note: The XML signature structure is:

```
<Signature Id="..." xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>...</SignedInfo>
  <SignatureValue>...</SignatureValue>
  <KeyInfo>...</KeyInfo>
  <Object>...</Object>
</Signature>
```

2) In which of the elements of the XML signature structure is the encrypted document included? In which of the elements of the XML signature structure is the digest (result of applying a Hash function) of the encrypted document included? In case the selected element has, in turn, an internal structure, then indicate the specific **sub-element**.

```
The sub-elements of the SignedInfo element are:
<SignedInfo>
  <CanonicalizationMethod Algorithm="..."/>
  <SignatureMethod Algorithm="..."/>
  <Reference URI="...">
    <Transforms>
      <Transform Algorithm="..."/>
    </Transforms>
    <DigestMethod Algorithm="..."/>
    <DigestValue>.../DigestValue>
  </Reference>
</SignedInfo>
```

The encrypted document is in the Object element in the case of Enveloping signature. In any other case, it is out of the Signature element.

The digest is in the DigestValue sub-element of the SignedInfo element.

Part III

Let's assume that the encrypted XML document is sent by user A to user B. Let's also assume that it can be represented as a message **m** and that the result of the encryption of m is **c=18**. Let's finally assume that we use RSA.

The public keys of A and B are: $(e_A, n_A) = (3, 22)$ $(e_B, n_B) = (11, 35)$

Note: Examples of RSA operations, $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n)$.

3) B has lost his private key, but he is able to calculate it. How could he do it?

We need to calculate d_B, the secret key. For this, we need $\Phi(n) = (p-1)*(q-1)$, being p and q the factors for n. Since n (n=p*q, both prime) is very small, we can deduct that n=35=5*7 and, therefore $\Phi(n) = 4*6=24$.
Then, $d_B = e_B^{-1} \bmod \Phi(n_B) = 11^{-1} \bmod 24$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

```
  b  |  d  | k
-----|-----|----
  0  |  24 | -
  1  |  11 | 2
 -2  |   2 | 5
 11  |   1 |
```

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

So the MMI is equal to 11, and $d_B = 11 \bmod 24 = 11$.

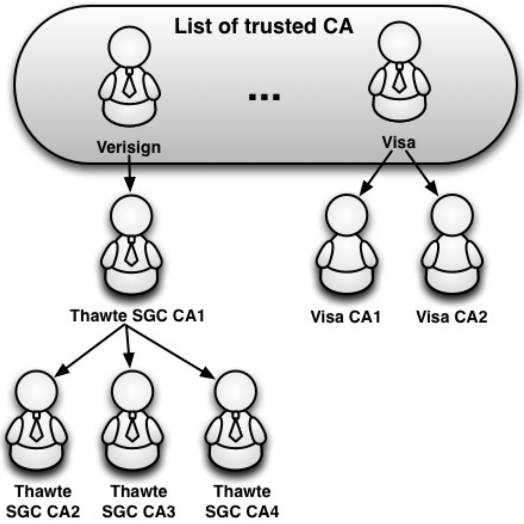
4) Calculate the value of the original message m.

We need to evaluate $m = c^{d_B} \bmod n_B$

So: $m = 18^{11} \bmod 35 = 64268410079232 \bmod 35 = 2$

Part IV

Let's assume that we have the following PKI, and another PKI following a Plain trust model.



5) Describe the steps that (the software of) a user whose certificate has been issued by Visa CA2 will follow to validate a certificate issued by Thawte SGC CA1.

- Check the correctness of the certificate's signature with the certificate of Thawte SGC CA1.
- If correctly signed, since we do not know Thawte SGC CA1, look for the certificate of the signer of Thawte SGC CA1, i.e. Verisign.
- Check the correctness of the certificate's signature with the certificate of Verisign, which is self-signed.
- Since we trust on Verisign because it is in our List of trusted CAs, then we can accept the original certificate.

6) Describe the steps that (the software of) a user whose certificate has been issued by Visa CA2 will follow to validate a certificate issued by the Plain trust model CA.

- Check the correctness of the certificate's signature with the certificate of the CA, which is self-signed.
- Even if correctly signed, since we do not know that CA and it is not possible to follow a certificates chain (CA is self-signed), we should not accept the original certificate.

EXERCISE 24

We are sending XML documents, such as the one in the following example, with confidential information:

```
<?xml version='1.0'?>
      <AccountInfo xmlns='http://example.org/bank'>
            <Name>John Smith</Name>
            <AccountNumber>12345678</AccountNumber>
            <Balance>
                  <Available>
                        <Amount>999</Amount>
                        <Currency>EUR</Currency>
                  </Available>
                  <Date>2015/12/18</Date>
            </Balance>
      </AccountInfo>
```

We want to encrypt part of this information to be sent through the network. We decide to just encrypt the `Available` element.

Reasoned and briefly answer the following questions:

1) If we use XML Encryption, provide the resulting XML document. Assume the unknown values, but at least include the "Type" attribute.

```
<?xml version='1.0'?>
<AccountInfo xmlns='http://example.org/bank'>
     <Name>John Smith</Name>
     <AccountNumber>12345678</AccountNumber>
          <Balance>
               <EncryptedData
               Type='http://www.w3.org/2001/04/xmlenc#Element'
               xmlns='http://www.w3.org/2001/04/xmlenc#'>
                    <CipherData>
                         <CipherValue>xxxxxx</CipherValue>
                    </CipherData>
               </EncryptedData>
               <Date>2015/12/18</Date>
          </Balance>
</AccountInfo>
```

Let's assume that the encrypted XML document is sent by user A to user B. Let's also assume that it can be represented as a message **m** whose value is equal to **2**. Let's finally assume that we use RSA.

The public keys of A and B are: $(e_A, n_A) = (11, 35)$ $(e_B, n_B) = (3, 22)$

Note: Examples of RSA operations, $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n)$.

2) B has lost his private key, but he hires us to calculate it. How could we do it?

We need to calculate d_B, the secret key. For this, we need $\Phi(n) = (p-1)*(q-1)$, being p and q the factors for n. Since n ($n=p*q$, both prime) is very small, we can deduct that $n=22=2*11$ and therefore $\Phi(n) = 1*10=10$.

Then, $d_B = e_B^{-1} \bmod \Phi(n_B) = 3^{-1} \bmod 10$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

```
 b  |  d  | k
----|-----|----
  0 |  10 | -
  1 |  3  | 3
 -3 |  1  |
```

$$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$$

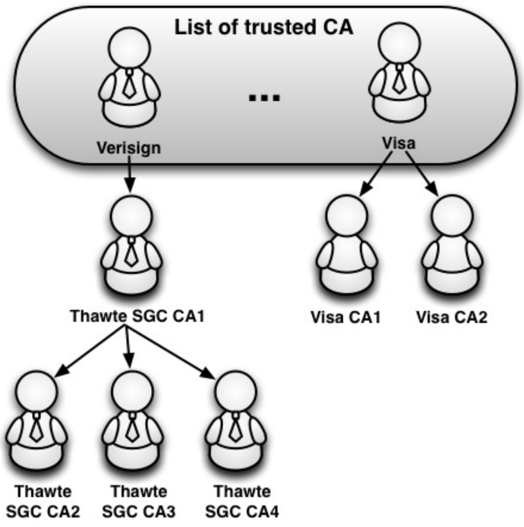
So the MMI is equal to 10-3=7, and $d_B = 7 \bmod 10 = 7$.

3) Calculate the value of the encrypted message c that A will send to B.

We need to evaluate $c = m^{e_B} \bmod n_B$

So: $c = 2^3 \bmod 22 = 8 \bmod 22 = 8$

Let's assume that we have the following PKI, and another PKI following a Plain trust model.

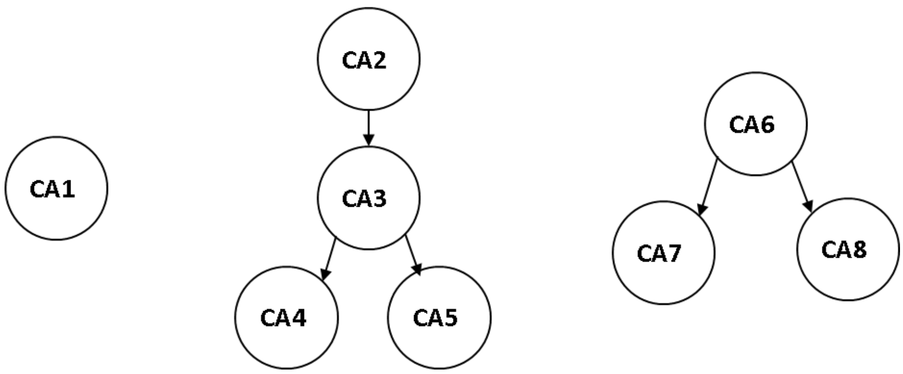


4) Describe the steps that (the software of) a user whose certificate has been issued by Visa CA2 will follow to validate a certificate issued by the Plain trust model CA.

- Check the correctness of the certificate's signature with the certificate of the CA, which is self-signed.
- Even if correctly signed, since we do not know that CA and it is not possible to follow a certificates chain (CA is self-signed), we should not accept the original certificate.

Problema 25

Tenemos varias infraestructuras PKI independientes. Se representan en la siguiente figura.



Suponer que tenemos un certificado C con el siguiente contenido (incompleto):

```
DATA:
        Version: 3 (0x2)
        Serial Number: 7829 (0x1e95)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: I
        Validity: Not before: …, Not after: …
        Subject: Su
        Subject Public Key Info:
                Public Key Algorithm: rsaEncryption
                RSA Public Key:
                        Modulus: n
                        Exponent: e
SIGNATURE:
        Certificate Signature Algorithm: md5WithRSAEncryption
        Certificate Signature: s
```

Suponer que CA1 emite un certificado C1 para un usuario U1, y que CA7 emite C2 para U2.

Suponer las siguientes claves públicas:

U1:	n= 15	e= 3
U2:	n= 14	e= 5

Contestar razonada y brevemente las siguientes preguntas:

Parte I

Si el valor **I** de un certificado C es igual a CA2,

1) ¿A cuál de los 2 certificados C1 o C2 corresponde?

A ninguno, porque C1 ha sido emitido por CA1 y C2 por CA7.

Si C es igual a C2,

2) ¿Cuáles serían los valores de **Su**, **n** y **e**?

Su = U2, y n y e serían los correspondientes a su clave pública: n = 14 y e = 5.

Suponer ahora que U2 quiere enviar un mensaje **m** a U1:

3) ¿Qué información (**I**, **Su**, **n**, **e**) contendrá el certificado que U1 presenta a U2? ¿Quién lo habrá firmado?

I = CA1, Su = U1, n = 15, e = 3: La firma será de CA1.

4) ¿Cómo puede U2 conseguir que su software acepte el certificado de U1?

Necesitará añadir CA1 (la CA que firma el certificado de U1) a su lista de Trusted CAs.

Parte II

Supongamos que el mensaje que U2 envía a U1 es **m=8**. Supongamos también que tanto U1 como U2 han perdido sus claves secretas. Finalmente, supongamos asimismo que encriptamos con RSA.

<u>Nota</u>: Ejemplos de operaciones RSA: $x = y^d \bmod n$; $y = x^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

1) Calcular el mensaje encriptado **c** que U2 envía a U1.

We need to calculate $c = m^{e_{U1}} \bmod n_{U1}$

So: $c = 8^3 \bmod 15 = 2$

2) Cuando U1 recibe el mensaje encriptado **c**, quiere desencriptarlo para calcular el mensaje original **m**. ¿Cuáles son los cálculos que tendrá que hacer y sus resultados?

We need to evaluate $m = c^{d_{U1}} \bmod n_{U1}$

We need to calculate first d_{U1}, the secret key. For this, we need $\Phi(n) = (p-1)*(q-1)$, being p and q the factors for n. Since n (n=p*q, both prime) is very small, we can easily deduct that n=15=3*5 and, therefore $\Phi(n) = 2*4=8$.
Then, $d_{U1} = e_{U1}^{-1} \bmod \Phi(n_{U1}) = 3^{-1} \bmod 8$. We calculate the modular multiplicative inverse (MMI) with "magic box":

b	d	k
0	8	-
1	3	2
-2	2	1
3	1	

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

So the MMI is equal to 3, and therefore d_{U1} = 3 mod 8 = 3.

So: $m = 2^3 \bmod 15 = 8$

3) Si U2 quisiera firmar el mensaje encriptado, ¿cuál sería la firma obtenida s? Detallar las operaciones a realizar.

Since we will be signing the encrypted c message with U2's secret key, we need to calculate $s = c^{d_{U2}} \bmod n_{U2}$

Similarly with what we did before, we need to calculate the secret key d_{U2}. We need $\Phi(n) = (p-1)*(q-1)$. We can deduct that n=14=2*7 and, therefore $\Phi(n)$ = 1*6=6.

Then, $d_{U2} = e_{U2}^{-1} \bmod \Phi(n_{U2}) = 5^{-1} \bmod 6$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

```
  b |  d | k
------------
  0 |  6 | -
  1 |  5 | 1
 -1 |  1 |       bi = bi-2 − (ki-1 * bi-1)
```

So the MMI is equal to 6-1=5, and therefore d_{U2} = 5 mod 8 = 5.

So: $s = 2^5 \bmod 14 = 4$ (we sign the encrypted message)

Parte III

Suponer ahora que encriptamos usando ElGamal con un generador $\alpha=3 \in Z_{31}$. La clave secreta del usuario U2 es CSu2=17 y la de un usuario U1 es CSu1=10. U1 recibe de U2: $(\alpha^v, c) = (13, 5)$.

Nota: Fórmulas ElGamal: Encriptar: $c = m*(\alpha^a)^v \in G$ Desencriptar: $m = c*(\alpha^{va})^{-1} \in G$

Calcular el valor del mensaje m que U2 ha enviado.

$m = 5*(13^{10})^{-1} \bmod 31 = 5 * 5^{-1} \bmod 31 = 5 * 25 \bmod 31 = 1$

ya que $13^{10} \bmod 31 = 5$, y $5^{-1} \bmod 31 = 31-6 = 25$ con magic box. *[Note: 13^{10} is a big number]*

```
  b |  d  | k
----|-----|---
  0 | 31  | -
  1 |  5  | 6
 -6 |  1  |          bi = bi-2 − (ki-1 * bi-1)
```

Problema 26

Suponer el siguiente documento XML (simplificado) que incluye *XML Encryption*.

```
<?xml version='1.0'?>
<UserProfile xmlns="http://example.org/system">
      <Name>
            <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Content"
                  <CipherData>
                        <CipherValue>xxxxxx</CipherValue>
                  </CipherData>
            </EncryptedData>
      </Name>
```

```
        <Nickname> Pep </Nickname>
        <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
              <CipherData>
                    <CipherValue>yyyyyy</CipherValue>
              </CipherData>
        </EncryptedData>
  </UserProfile>
```

Contestar razonada y brevemente las siguientes preguntas:

1) ¿Qué diferencia hay entre los dos elementos `EncryptedData`?

El primer EncryptedData es de un valor (Content), mientras que el segundo es de un elemento entero.

2) Una vez desencriptado, ¿qué elementos (y subelementos) tiene el *root element* `UserProfile` del documento XML? Dar un ejemplo de un posible resultado de desencriptar el documento XML.

User Profile tiene 3 elementos: Name, Nickname y un tercero que no conocemos porque está encriptado.

Podemos dar cualquier valor al contenido del elemento Name y cualquier nombre al tercer elemento (y su valor). Por ejemplo:

```
<?xml version='1.0'?>
<UserProfile xmlns='http://example.org/system'>
      <Name> Josep Roca </Name>
      <Nickname>Pep</Nickname>
      <Password> abc </Password>
</UserProfile>
```

Exercise 27

From the following privacy policy in XACML, build a new Policy to regulate that the video `urn:bbc:mdocum:Planets.mp4` is only played by users in the United Kingdom.

It is only necessary to indicate the changes with respect to the given Policy.

```
<Policy  xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
         xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
         xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
                        http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
         PolicyId="urn:isdcm:policyid:1"
         RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable"
         Version="1.0">
<Description> Policy for playing video A </Description>
<Target/>
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleVideoA" Effect="Permit">
  <Description> Any user may play the video urn:mvideo:videoA.mp4 before the end of the year </Description>
  <Target>
    <AnyOf>
      <AllOf>
        <!-- Which resource -->
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">urn:mvideo:videoA.mp4 </AttributeValue>
            <AttributeDesignator   MustBePresent="false"
                                   Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                                   AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                                   DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
```

```
        <!-- Which action  -->
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"> play </AttributeValue>
            <AttributeDesignator       MustBePresent="false"
                                       Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                       DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
        <AttributeDesignator    MustBePresent="false"
                                Category="urn:oasis:names:tc:xacml:3.0:date"
                                AttributeId="accessDate"
                                DataType="http://www.w3.org/2001/XMLSchema#date"/>
      </Apply>
      <AttributeValue DataType=http://www.w3.org/2001/XMLSchema#date> 2020-01-01 </AttributeValue>
    </Apply>
  </Condition>
</Rule>
</Policy>
```

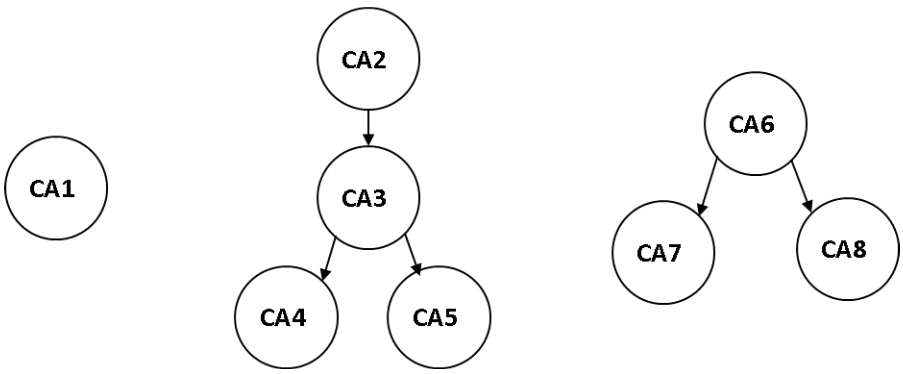
Changes in yellow:

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
                      http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
        PolicyId="urn:isdcm:policyid:2"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable"
        Version="1.0">
<Description> Another restriction </Description>
<Target/>
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleUk" Effect="Permit">
  <Description> Only the users from United Kingdom are allowed to play the video urn:bbc:mdocum:Planets.mp4
  </Description>
  <Target>
    <AnyOf> <AllOf>
        <!-- Which resource -->
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue
                DataType="http://www.w3.org/2001/XMLSchema#string"> urn:bbc:mdocum:Planets.mp4
          </AttributeValue>
          <AttributeDesignator MustBePresent="false"
                               Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                               AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                               DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
        <!-- Which action  -->
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType=http://www.w3.org/2001/XMLSchema#string> play </AttributeValue>
          <AttributeDesignator   MustBePresent="false"
                                 Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                                 DataType="http://www.w3.org/2001/XMLSchema#string"/>
```

```
          </Match>
        </AllOf> </AnyOf>
    </Target>
    <Condition>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0: function:string-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <AttributeDesignator MustBePresent="false"
                               Category="urn:oasis:names:tc:xacml:3.0:country"
                               AttributeId="country"
                               DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"> UK </AttributeValue>
      </Apply>
    </Condition>
  </Rule>
</Policy>
```

Problema 28

Tenemos varias infraestructuras PKI independientes. Se representan en la siguiente figura.



Suponer que tenemos un certificado C con el siguiente contenido:

```
DATA:
      Version: 3 (0x2)
      Serial Number: 7829 (0x1e95)
      Signature Algorithm: md5WithRSAEncryption
      Issuer: CA7
      Validity: Not before: …, Not after: …
      Subject: U2
      Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key:
                  Modulus: 14
                  Exponent: 5
SIGNATURE:
      Certificate Signature Algorithm: md5WithRSAEncryption
      Certificate Signature: s
```

Suponer los usuarios U1, U2 y U3, cada uno de ellos con un certificado emitido por una de las 8 CAs de la figura (CA1 a CA8).

Contestar razonada y brevemente las siguientes preguntas:

Parte I

Suponer que las encriptaciones en este apartado son RSA.

Nota: Ejemplos de operaciones RSA: $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n)$.

Teniendo en cuenta el certificado C anterior,

1) ¿Quién lo ha emitido y para qué usuario?

CA7 para U2.

Suponer que U1 (cuyo certificado ha emitido CA8) quiere enviar un mensaje **m_1**=8 a U2:

2) Calcular el mensaje encriptado c que U1 envía a U2.

We need to calculate $c = m_1^{e_{U2}} \bmod n_{U2}$

So: $c = 8^5 \bmod 14 = 8$

Suponer ahora que U2 quiere enviar, encriptándolo, un mensaje **m_2**=2 a U3:

3) Con la información de que se dispone, describir el certificado que será necesario para poder encriptar m_2. Indicar sólo los campos que seguro que cambian de valor respecto al certificado C anterior. ¿Quién lo habrá firmado?

```
DATA:
      Version: 3 (0x2)
      Serial Number: 7829 (0x1e95)
      Signature Algorithm: md5WithRSAEncryption
      Issuer: CAx (la que haya emitido el certificado de U3)
      Validity: Not before: …, Not after: …
      Subject: U3
      Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key:
                  Modulus: x
                  Exponent: y
SIGNATURE:
      Certificate Signature Algorithm: md5WithRSAEncryption
      Certificate Signature: s
```

La firma será de CAx.

4) Dada la estructura de CAs de la que disponemos (CA1 a CA8 de la figura), ¿qué ha de ocurrir para que U2 pueda enviar con total seguridad el mensaje m_2 a U3?

U2 tiene que aceptar el certificado de U3. Por tanto, ha de aceptar a la CA que haya emitido su certificado. Dada la estructura jerárquica donde está CA7 (la CA que ha emitido el certificado de U2), los emisores del certificado de U2 sólo pueden ser CA6, CA7 o CA8). Una alternativa sería que el software de U2 incluyese en su lista de *Trusted CAs* la CA que firma el certificado de U3.

5) Desde el punto de vista de la estructura de CAs, justificar por qué sí se podía enviar **m_1**.

El certificado de U1 está emitido por CA8, mientras que el de U2 está emitido por CA7. CA7 y CA8 reconocen a CA6 (su root), por lo que U1 aceptará sin problemas el certificado de U2.

Parte II

Suponer ahora que para el envío del mensaje **m_1**=8 de U1 a U2 encriptamos usando ElGamal con un generador α=2 $\in Z_{11}$. U1 elije un valor aleatorio **v**=4. La clave secreta del usuario U2 es Ks=7.

Nota: Fórmulas ElGamal: Encriptar: $c = m*(\alpha^a)^v \in G$ Desencriptar: $m = c*(\alpha^{va})^{-1} \in G$

1) Calcular los valores que U1 enviará a U2.

En ElGamal se envía c y α^v.

$\alpha^v = 2^4 \bmod 11 = \mathbf{5}$.

$c = m_1 * (\alpha^{Ks})^v \bmod 11 = 8 * (2^7)^4 \bmod 11 = 8 * (7)^4 \bmod 11 = 8 * 3 \bmod 11 = \mathbf{2}$

2) Verificar que el resultado anterior es correcto calculando $\mathbf{m_1}$ a partir de los valores que U2 ha recibido.

$$m_1 = c*(\alpha^{v\,Ks})^{-1} \bmod 11 = 2*(5^7)^{-1} \bmod 11 = 2 * 3^{-1} \bmod 11 = 2 * 4 \bmod 11 = 8$$

ya que $5^7 \bmod 11 = 3$, y $3^{-1} \bmod 11 = 4$ con magic box.

b	d	k
0	11	-
1	3	3
-3	2	1
4	1	

$$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$$

Exercise 29

We have a XACML privacy policy at the end of the Exercise.

Reasoned and briefly answer the following questions about that policy:

1) What is the purpose of the attribute-value pair `Effect="Permit"`? What are the other possible values for the attribute `Effect`?

`Effect="Permit"` indicates that the specified action is only executed when the rule is fulfilled.

The attribute `Effect` may also have the value `Deny`.

2) In the `AttributeValue` of the first element `Match`, what is the URN identifying? What would change in the policy if we would modify its value?

It identifies the resource that may be played.

If we change the URN, the resource whose playing we are authorizing will change.

3) The description of the rule says "`Any user may play urn:mvideo:videoA.mp4 before end of the year`" In which specific elements of the policy is the concept "`before end of the year`" specified?

In the element Condition.

In particular, with

<Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal">

we indicate the limit condition ("less or equal"), and in

<AttributeValue DataType=http://www.w3.org/2001/XMLSchema#date> 2019-12-31 </AttributeValue>

we include the specific limit date.

The rest of elements define the used types.

4) What should we modify if we would like that the limit date for playing is end of July 2020?

<AttributeValue DataType=http://www.w3.org/2001/XMLSchema#date> 2019-12-31 </AttributeValue>

should be:

<AttributeValue DataType=http://www.w3.org/2001/XMLSchema#date> 2020-07-31 </AttributeValue>

Privacy Policy in XACML

```xml
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
                           http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
        PolicyId="urn:isdcm:policyid:1"
        RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-
applicable"
        Version="1.0">
<Description> Policy play video A </Description>
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:ejemplo:RuleVideoA" Effect="Permit">
   <Description> Any user may play urn:mvideo:videoA.mp4 before end of the year </Description>
   <Target>
      <AnyOf>
         <AllOf>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    urn:mvideo:videoA.mp4
               </AttributeValue>
               <AttributeDesignator MustBePresent="false"
                              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
                              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
                              DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </Match>
            <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                    Play
               </AttributeValue>
               <AttributeDesignator         MustBePresent="false"
                              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
                              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
                              DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </Match>
         </AllOf>
      </AnyOf>
   </Target>
   <Condition>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-only">
            <AttributeDesignator    MustBePresent="false"
                           Category="urn:oasis:names:tc:xacml:3.0:date"
                           AttributeId="accessDate"
                           DataType="http://www.w3.org/2001/XMLSchema#date"/>
         </Apply>
         <AttributeValue DataType=http://www.w3.org/2001/XMLSchema#date>
             2019-12-31
         </AttributeValue>
      </Apply>
   </Condition>
</Rule>
</Policy>
```

Exercise 30

We interchange XML documents with confidential information using XML Encryption. We receive the following document (the `EncryptedData` part has been simplified):

```xml
<?xml version='1.0'?>
<AccountInfo xmlns='http://example.org/bank'>
      <Name>
            <EncryptedData
            Type='http://www.w3.org/2001/04/xmlenc#Content'
                  <CipherData>
                        <CipherValue>xxxxxx</CipherValue>
                  </CipherData>
            </EncryptedData>
      </Name>
      <AccountNumber>12345678</AccountNumber>
```

```
        <Balance>
              <Available>
                    <Amount>
                          <EncryptedData
                          Type='http://www.w3.org/2001/04/xmlenc#Content'
                                <CipherData>
                                      <CipherValue>xxxxxx</CipherValue>
                                </CipherData>
                          </EncryptedData>
                    </Amount>
                    <Currency>EUR</Currency>
              </Available>
              <Date>2018/12/18</Date>
        </Balance>
  </AccountInfo>
```

Reasoned and briefly answer the following questions:

Part I

1) If the original values of the elements `Name` and `Amount` were "John Smith" and "1000", respectively, provide the XML document before encryption.

```
<?xml version='1.0'?>
<AccountInfo xmlns='http://example.org/bank'>
      <Name>John Smith</Name>
      <AccountNumber>12345678</AccountNumber>
      <Balance>
            <Available>
                  <Amount>1000</Amount>
                  <Currency>EUR</Currency>
            </Available>
            <Date>2018/12/18</Date>
      </Balance>
</AccountInfo>
```

Part II

Let's suppose that the encrypted XML document is sent from user A to user B. Let's also suppose that the document is represented as a message **m** with value **m=8**. Let's finally assume that we use RSA.

The public keys of A and B are: $(e_A, n_A) = (5, 14)$ $(e_B, n_B) = (3, 15)$

Note: Examples of RSA operations: $x = y^d \bmod n;\ y = x^e \bmod n;\ d = e^{-1} \bmod \Phi(n).$

2) Calculate the encrypted message **c** that A sends to B.

We need to calculate $c = m^{e_B} \bmod n_B$

So: $c = 8^3 \bmod 15 = 2$

3) When B receives the encrypted message **c**, she wants to decrypt it to calculate the original message **m**. Provide the needed operations and results in order to obtain it.

We need to evaluate $m = c^{d_B} \bmod n_B$

We need to calculate d_B, the secret key. For this, we need $\Phi(n) = (p-1)*(q-1)$, being p and q the factors for n. Since n (n=p*q, both prime) is very small, we can easily deduct that n=15=3*5 and, therefore $\Phi(n) = 2*4=8$.
Then, $d_B = e_B^{-1} \bmod \Phi(n_B) = 3^{-1} \bmod 8$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

b	d	k
0	8	-
1	3	2
-2	2	1
3	1	

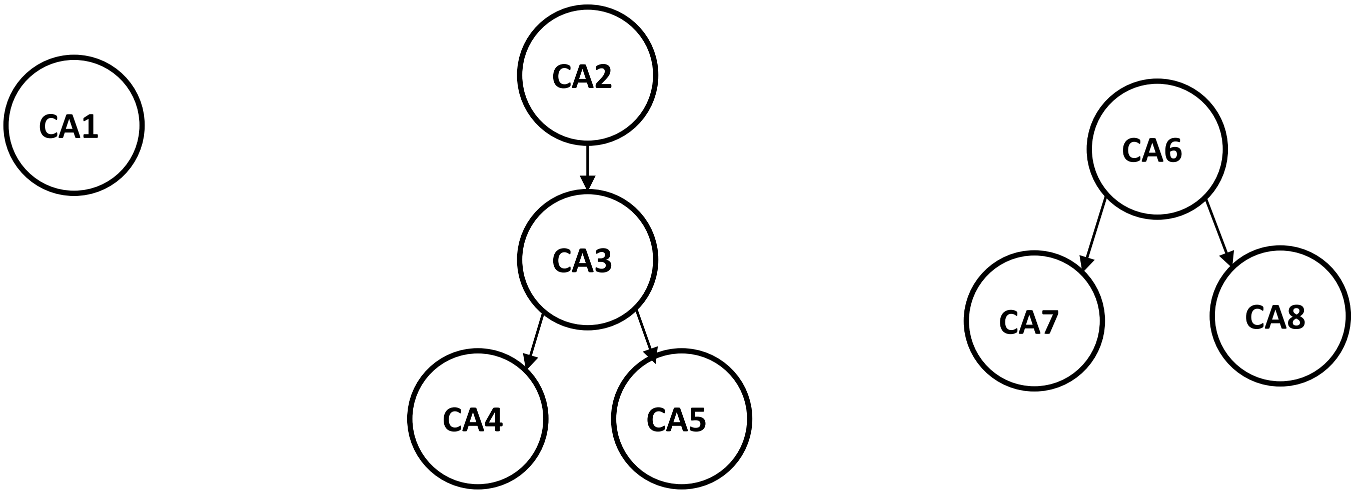
$$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$$

So the MMI is equal to 3, and therefore $d_B = 3 \bmod 8 = 3$.

So: $m = 2^3 \bmod 15 = 8$

Part III

We have some CAs organized as in the following figures; i.e., we have both Plain and Hierarchical structures.



We also have the following certificates:

C1: issued by CA1; C2: issued by CA5; C3: issued by CA7; C4: issued by CA8.

4) Which signature (or signatures) would include C1 and C4?

   C1 will include the signature of its issuing CA, i.e. CA1.
   C4 will include the signature of its issuing CA, i.e. CA8.

5) Which signature (or signatures) would include the certificates of CA1, CA4 and CA6?

   CA1's certificate will include its own signature (self-signed certificate).
   CA4's certificate will include the signature of its parent CA, i.e. CA3.
   CA6's certificate will include its own signature (self-signed certificate).

6) Why (the software of) the user identified by certificate C4 will accept certificate C3? What are the steps followed?

   - Check the correctness of C3's signature.
   - If correctly signed, since we do not know the certificate's issuer (CA7), look for the certificate of its signer: CA6.
   - Check the correctness of the signature of CA6's certificate.
   - CA6's certificate is self-signed, but it is the issuer of CA8's certificate, being CA8 the issuer of C4, so the software will accept certificate C3.

Exercise 31

We send documents with confidential information from A to B.

Reasoned and briefly answer the following questions:

Part I

Let's suppose that the document is represented as a message **m**. Let's also assume that we use RSA.

The public keys of A and B are: $(e_A, n_A) = (3, 15)$ $(e_B, n_B) = (5, 14)$

Note: Examples of RSA operations: $x = y^d \bmod n$; $y = x^e \bmod n$; $d = e^{-1} \bmod \Phi(n)$.

1) When B receives the encrypted message **c=8**, he wants to decrypt it to calculate the original message **m**. Provide the needed operations and results in order to obtain it.

We need to evaluate $m = c^{d_B} \bmod n_B$
We need to calculate d_B, the secret key. For this, we need Φ(n) = (p-1)*(q-1), being p and q the factors for n. Since n (n=p*q, both prime) is very small, we can easily deduct that n=14=2*7 and, therefore Φ(n) = 1*6=6.
Then, $d_B = e_B^{-1} \bmod \Phi(n_B) = 5^{-1} \bmod 6$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

b	d	k
0	6	-
1	5	1
-1	1	

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

So the MMI is equal to 6-1=5, and therefore $d_B = 5 \bmod 6 = 5$.
So: $m = 8^5 \bmod 14 = 8$

2) If A would like to sign the encrypted message, which would be its result **s**? Detail the needed operations.

Since we will be signing the encrypted c message with A's secret key,
we need to calculate $s = c^{d_A} \bmod n_A$
Similarly with what we did before, we need to calculate d_A, the secret key. We need Φ(n) = (p-1)*(q-1). We can deduct that n=15=3*5 and, therefore Φ(n) = 2*4=8.
Then, $d_A = e_A^{-1} \bmod \Phi(n_A) = 3^{-1} \bmod 8$. We calculate the modular multiplicative inverse (MMI) with the "magic box":

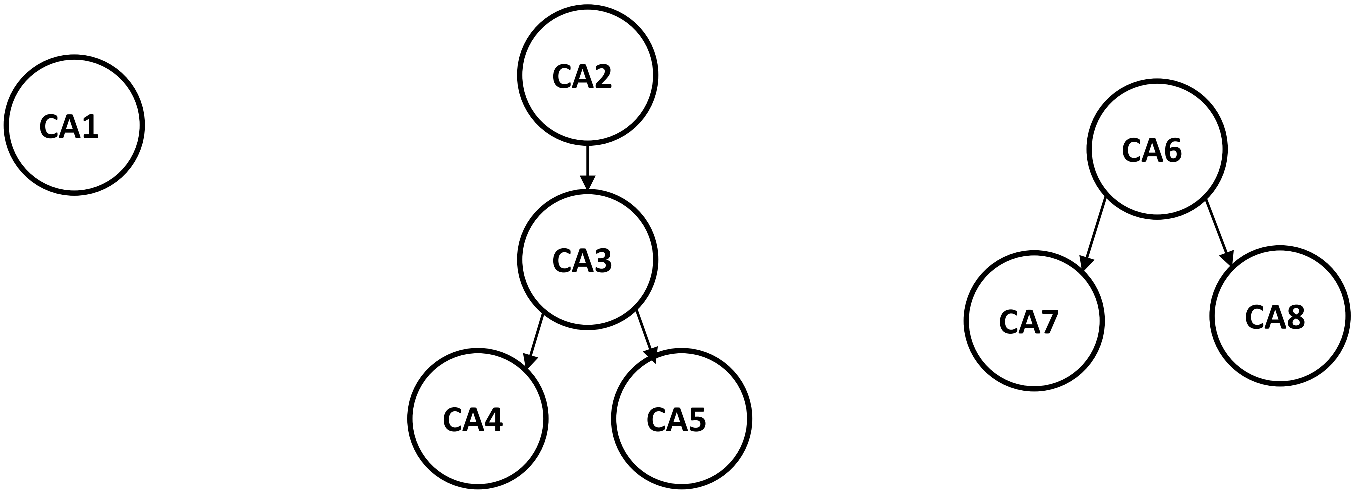
b	d	k
0	8	-
1	3	2
-2	2	1
3	1	

$b_i = b_{i-2} - (k_{i-1} * b_{i-1})$

So the MMI is equal to 3, and therefore $d_A = 3 \bmod 8 = 3$.
So: $s = 8^3 \bmod 15 = 2$

Part II

We have some CAs organized as in the following figures; i.e., we have both Plain and Hierarchical structures.



We also have the following certificates:

C1: issued by CA1; C2: issued by CA5; C3: issued by CA7.

3) Which signature (or signatures) would include C1 and C2?

   C1 will include the signature of its issuing CA, i.e. CA1.
   C2 will include the signature of its issuing CA, i.e. CA5.

4) Which signature (or signatures) would include the certificates of CA7?

   CA7's certificate will include the signature of its parent CA, i.e. CA6.

5) Describe the steps that (the software of) the user identified by certificate C2 will follow to validate certificate C3, just received.

   - Check the correctness of the certificate's signature.
   - If correctly signed, since we do not know the certificate's issuer (CA7), look for the certificate of its signer: CA6.
   - Check the correctness of the certificate's signature.
   - Since CA6's certificate is self-signed and we do not know CA6 (so we do not trust on it), we should reject certificate C3.

6) The software of a user whose certificate has been issued by CA1 will not accept a certificate issued by CA8. Without adding new CAs, what could we do to have the software accepting that certificate?

   We could add CA8 to the list of Trusted CAs. Cross-certification or bridges is not controlled by the user, but by the CA.

Exercise 32

These are 2 fragments of XACML v3.0 rules.

Fragment 1

```
<Target>
 <AnyOf>
  <AllOf>
   <Match
     MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
         urn:hospital:lab_result
       </AttributeValue>
```

```
        <AttributeDesignator
          MustBePresent="false"
          Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
          AttributeId="urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
          DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
    </Match>
   </AllOf>
  </AnyOf>
</Target>
```

Fragment 2

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
   …
     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         Modify
       </AttributeValue>
       <AttributeDesignator MustBePresent="false"
         Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </Match>
   …
</Rule>
```

Reasoned and briefly answer the following questions:

1) **a)** What is the type of the XACML attribute being used in fragment 1? **b)** Which operation(s) should be performed with that attribute? **c)** What is the value of that attribute?

a) The attribute is of type `anyURI`.

b) It should compare (`anyURI-equal`) with an attribute of category `resource` of type `anyURI`.

c) The value of the attribute is `urn:hospital:lab_result`.

2) Fragment 2 defines a rule. **a)** What is its "effect"? Where is it defined? **b)** The rule contains an element of type `Match`, what is the XML type of the attribute it specifies? **c)** What is the XACML operation to be performed associated to that "Match"? Where is it defined? **d)** Which action is being authorized with this rule?

a) The rule defines the effect `Permit` in the attribute `Effect`.

b) Type `string`, to be compared with an attribute of category `action`.

c) The operation is a comparison (`string-equal`), defined in the attribute `MatchId` of the element `Match`.

d) The authorized `action` is `Modify`.

Exercise 33

These are 2 fragments of XACML v3.0 rules.

Fragment 1

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Deny">
   …
```

```
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Modify
          </AttributeValue>
          <AttributeDesignator MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
      …
   </Rule>
```

Fragment 2

```
<Rule RuleId="urn:oasis:names:tc:xacml:3.0:RuleSAM" Effect="Permit">
…
        <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            Print
          </AttributeValue>
          <AttributeDesignator MustBePresent="false"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action"
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </Match>
…
</Rule>
```

Reasoned and briefly answer the following questions:

1) Fragment 1 defines a rule. **a)** What is its "effect"? **b)** The rule contains an element of type `Match`, what is the XML type of the attribute it specifies? **c)** Which action is being authorized with this rule?

e) The rule defines the effect `Deny` in the attribute `Effect`.

f) Type `string`, to be compared with an attribute of category `action`.

g) The operation is a comparison (`string-equal`), defined in the attribute `MatchId` of the element `Match`. The non-authorized `action` is `Modify`.

2) Fragment 2 defines a rule. **a)** What is its "effect"? **b)** Which action is being authorized with this rule?

a) The rule defines the effect `Permit` in the attribute `Effect`.

b) The operation is a comparison (`string-equal`), defined in the attribute `MatchId` of the element `Match`. The authorized `action` is `Print`.

Exercise 34

A is sending to B an encrypted message c=27. We encrypt using ElGamal with a generator $\alpha=3 \in Z_{31}$. B's secret key is 10. A uses v=2.

Note: ElGamal calculations: Encrypt: $c = m*(\alpha^a)^v \in G$ Decrypt: $m = c*(\alpha^{va})^{-1} \in G$

Reasoned and briefly answer the following questions:

1) Should A send something more than c=27? If so, what else?

In ElGamal, c is accompanied with α^v.

In this case, $\alpha^v = 3^v \bmod 31 = 9$

2) Calculate the original message before encryption.

For decryption we need to calculate $m = c*(\alpha^{va})^{-1} \in G$

$m = 27 * (9^{10})^{-1} \bmod 31) = 27 * (3.486.784.401 \bmod 31)^{-1} \bmod 31) =$
$= 27 * (5^{-1} \bmod 31)$

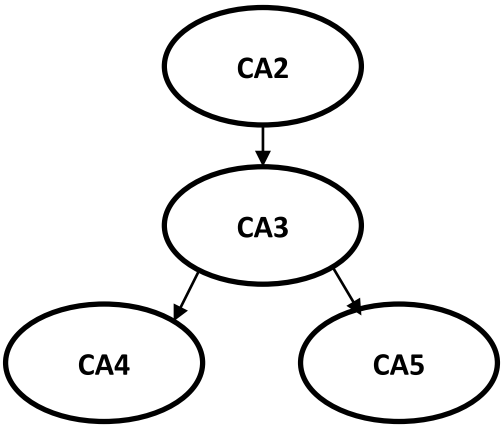
For $5^{-1} \bmod 31$ we calculate the modular multiplicative inverse (MMI) with the "magic box":

```
        b |   d | k
-------------
        0 |  31 |  -
        1 |   5 |  6
-6 |    1 |        bi = bi-2 - (ki-1 * bi-1)
```

So the MMI is equal to 31-6=25.
Then: $m = 27 * (5^{-1} \bmod 31) = (27 * 25) \bmod 31 = 24$

Exercise 35

Given the following hierarchical structure of Certification Authorities:



CA4 issues certificate C1 for UserA and CA5 issues certificate C2 for UserB.

Reasoned and briefly answer the following questions:

If UserA sends protected content to UserB, **a)** how many certificates (and which ones) needs to validate UserA? **b)** how many certificates (and which ones) needs to validate UserB? **c)** What could be a reason for UserA to reject UserB's certificate?

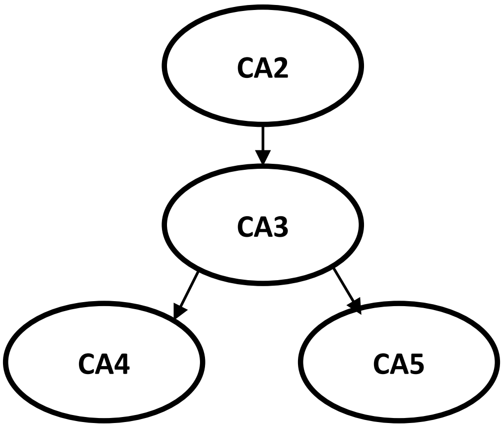
a) UserA needs first to validate C2 (UserB's certificate issued by CA5). Then, it has to validate CA5's and CA3's certificates, and this is enough because both users trust in CA3.

b) UserB does not need to validate any certificate, since it is receiving (not sending) information, so it is not using others' public keys.

c) Since UserA and UserB have certificates from the same CAs hierarchy, trust should not be a problem. Therefore, the reason should be, for example, a revoked, expired or corrupted certificate.

Exercise 36

Given the following hierarchical structure of Certification Authorities:



CA4 issues certificate C1 for UserA and CA5 issues certificate C2 for UserB. On the other hand, C3 is the certificate of UserC, which is issued by CA6 that is a CA following the plain model.

If UserA sends encrypted content to UserC,

reasoned and briefly answer the following questions:

a) which certificates need to be validated and by whom? **b)** What could be a reason for UserA to reject UserC's certificate? **c)** Repeat question "a)" if UserA sends encrypted content to UserB.

a) C3 (UserC's certificate issued by CA6). Since CA6's certificate is self-signed and unknown to UserA, it will be rejected by UserA.

b) It is rejected because both users are in independent trust hierarchies.

c) C2 (issued by CA5), i.e. the certificate of UserB, since it is the recipient of the message that needs to be encrypted with its public key. Then, it has to validate CA5's and CA3's certificates, and this is enough because both users trust in CA3.