Networks and Internet Applications

Continuous Assessment Test – CA2

Nicolas D'Alessandro Calderon

Answers

1. **Polyalphabetic encryption**

C1 - Caesar cipher (offset +7)

Plaintext Letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Letter:	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g

C2 - Monoalphabetic cipher

Plaintext Letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Letter:	p	g	i	r	k	u	m	f	b	t	y	v	o	z	q	x	h	l	e	j	a	c	w	n	s	d

Polyalphabetic cipher using C1 and C2 with repetition pattern [C1, C1, C1, C2, C2, C1, C2]

Plaintext Letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Alphabet:	C1	C1	C1	C2	C2	C1	C2	C1	C1	C1	C2	C2	C1	C2	C1	C1	C1	C2	C2	C1	C2	C1	C1	C1	C2	C2
Ciphertext Letter:	h	i	j	r	k	m	m	o	p	q	y	v	t	z	v	w	x	l	e	a	a	c	d	e	s	d

Plaintext:	n	i	c	o	l	a	s		d	a	l	e	s	s	a	n	d	r	o
Alphabet:	C1	C1	C1	C2	C2	C1	C2		C1	C1	C1	C2	C2	C1	C2	C1	C1	C1	C2
Ciphertext:	u	p	j	q	v	h	e		k	h	s	k	e	z	p	u	k	y	q

Plaintext:	n	i	c	o	l	a	s	d	a	l	e	s	s	a	n	d	r	o
Ciphertext:	u	p	j	q	v	h	e	k	h	s	k	e	z	p	u	k	y	q

Source: Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition,
Chapter 8: Security in Computer Networks, Section 8.2.1 Symmetric Key Cryptography,

2.

a) In ~~stream encryption~~, the data is split and encrypted into blocks.

In **a block cipher**, the data is split and encrypted into blocks.

The learning material mentions that "In a block cipher, the message to be encrypted is processed in blocks of k bits."
Source: Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.2.1 Symmetric Key Cryptography, p. 644.

b) The best way to get source authentication of a message is to use ~~symmetric encryption~~.

The best way to get source authentication of a message is to use **digital signatures with asymmetric encryption.**

The learning material mentions that "[…] digital signatures also provide message integrity, allowing the receiver to verify that the message was unaltered as well as the source of the message."
Source: Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.3.3 Digital Signatures, p. 660.

c) Between Alice and Bob, the session keys mechanism uses ~~the same~~ symmetric key each time (connection/session/message).

Between Alice and Bob, the session keys mechanism uses **a new** symmetric key each time (connection/session/message).

*"First Alice chooses a key that will be used to encode the data itself; this key is referred to as a **session key** […]"*
Source: Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.2.2 Public Key Encryption, p. 652.

d) The length of the hash of a message is ~~always approximately equal to~~ the length of the original message.

The length of the hash of a message is **fixed regardless of** the length of the original message.

Source: "[…] a hash function takes an input, m, and computes a fixed-size string H(m) known as a hash."
Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.3.1 – Cryptographic Hash Functions, page 655.

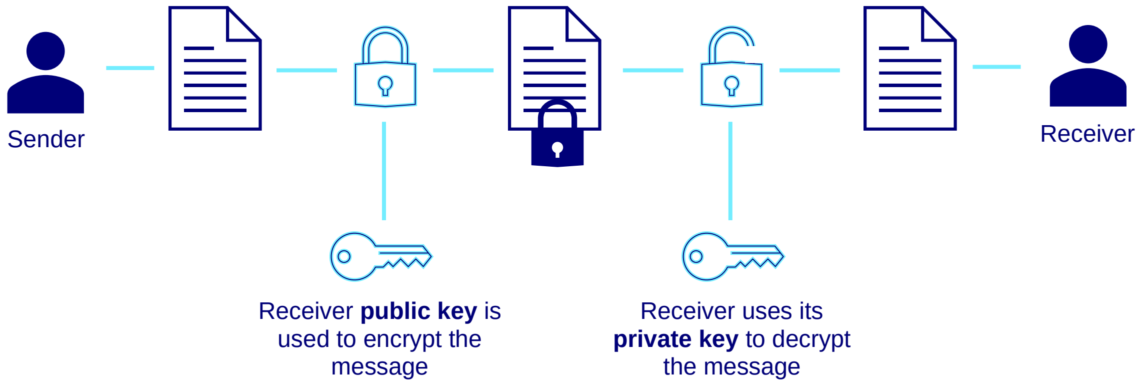
3.

a) RSA is an encryption algorithm that uses a pair of keys, a public key and a private key. The public key can be openly shared while the private key should be kept secret.
These keys are mathematically related, meaning that the message encrypted by the public key can only be decrypted using the corresponding private key and also the other way around.

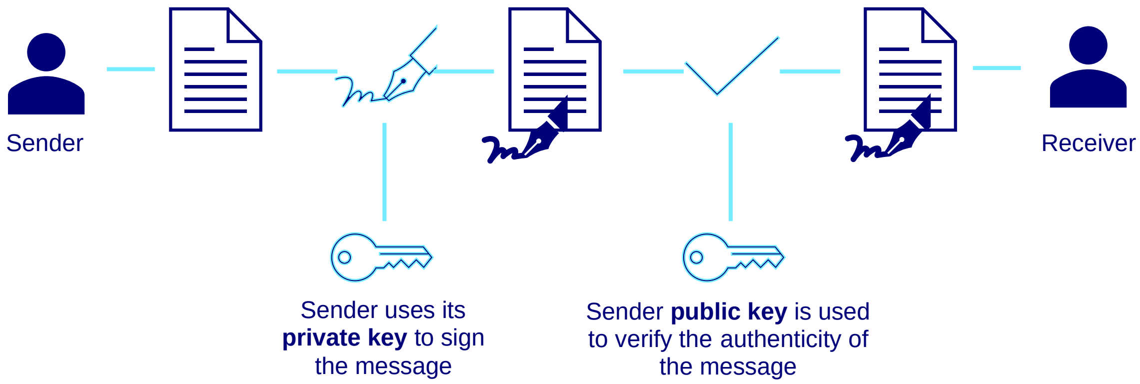
In simple words, this is achieved by using modular exponential operations with very big numbers (the larger the numbers the more difficult to break). A summary of the mathematical operations and steps described in the learning material could be:

1. Choose two large prime numbers $\mathbf{p}$ and $\mathbf{q}$
2. Calculate $\mathbf{n = p * q}$ and a function called $\boldsymbol{\phi(n)}$
3. Choose a number **e** (public exponent) small and coprime with $\boldsymbol{\phi(n)}$
4. Calculate **d**, which is the inverse of **e** module $\boldsymbol{\phi(n)}$
5. Finally, the **public key** is $(\mathbf{e, n})$ and the **private key** is $(\mathbf{d, n})$.

To encrypt a message $\mathbf{m}$ is used $\mathbf{c = m^e \bmod n}$ (with public key) To decrypt $\mathbf{m = c^d \bmod n}$ (with private key)

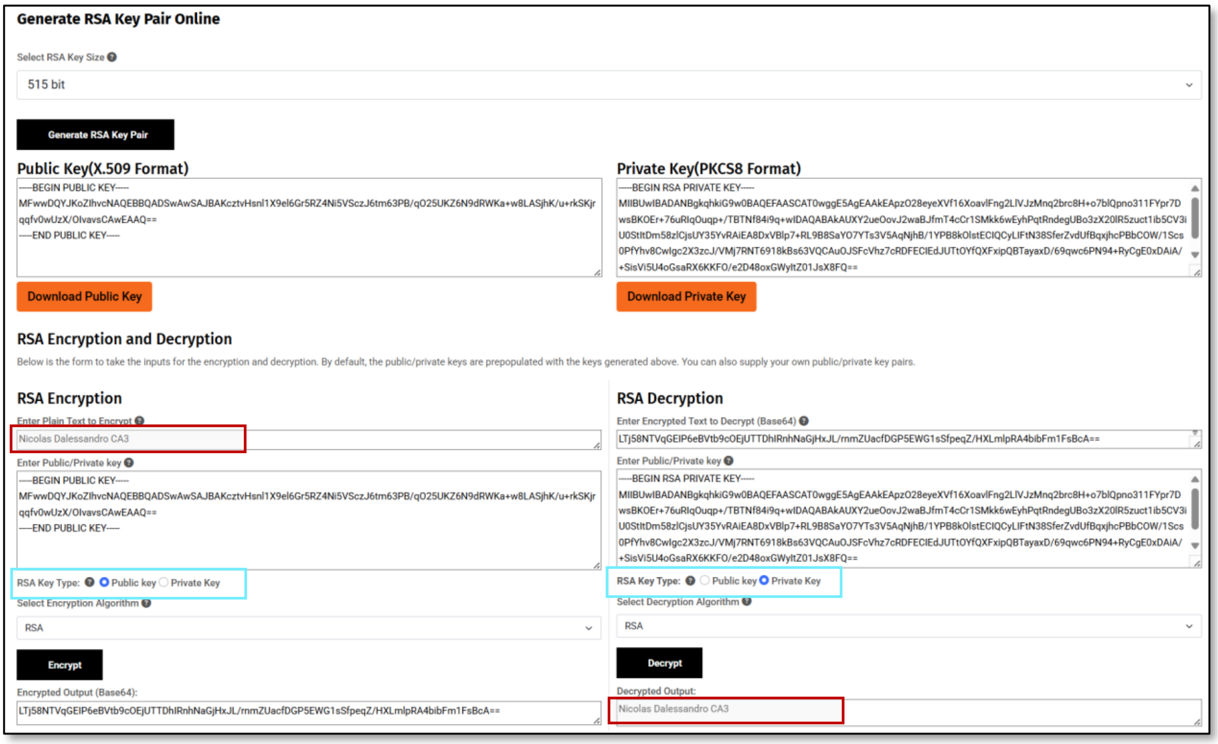


Additionally, a message $\mathbf{m}$ can be digitally signed using $\mathbf{f = m^d \bmod n}$ (with private key) Then, to verify the sign $\mathbf{m = f^e \bmod n}$ (with public key)



RSA is powerful because it enables both **secure communication (confidentiality)** and secure **authentication and integrity (digital signatures),** depending on how the key pair is used.

b)



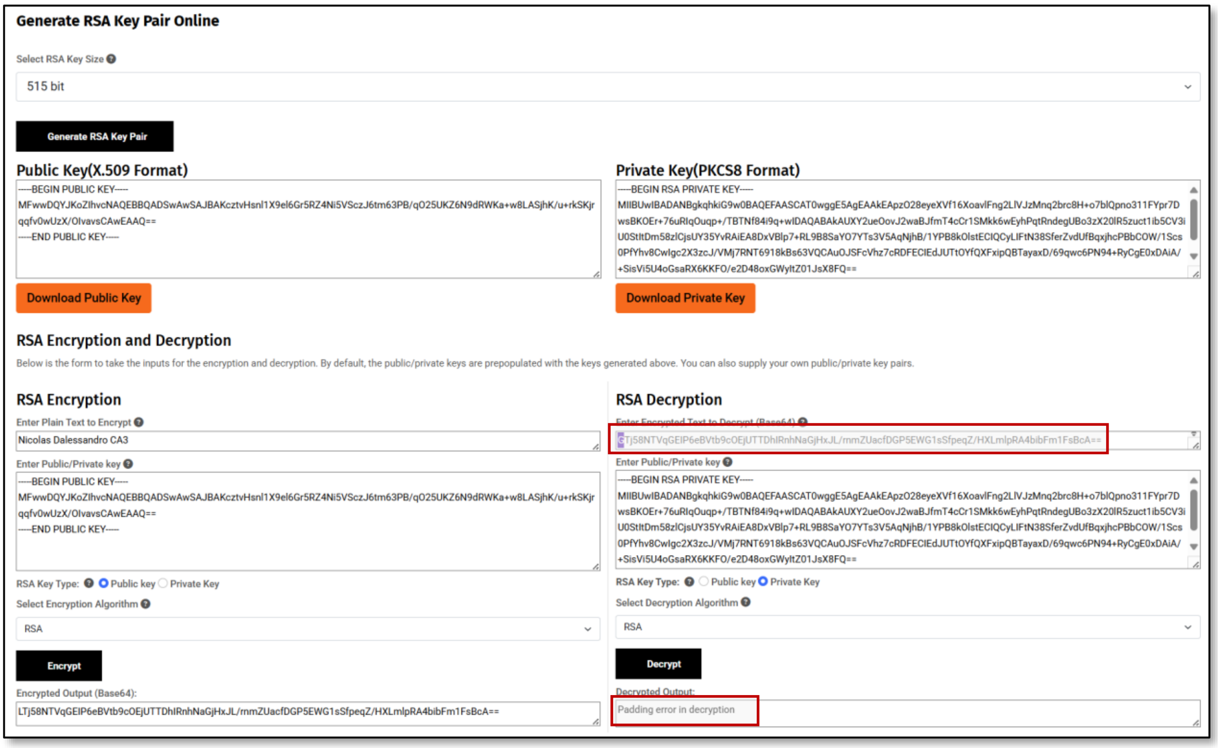
c) As we explain in the section a), there are **two valid** (secure) options for using the RSA keys, each offering different security options.

1. In the image above I encrypted the message using the *public key* and then decrypted it using the *private key*. This ensures **Confidentiality**, since anyone can encrypt the message with the openly shared public key, but only the receiver (who have the private key) can decrypt the message.

2. Additionally, as we also explained in section a), the sender could sign a message using *it private key*, so anyone can verify that signature using the corresponding and openly available *public key* . This ensures **Authenticity and Integrity** because only the owner of the private key could have signed the message, and if a single bit is altered after it has been signed, the public key will no longer be able to correctly verify the signature.

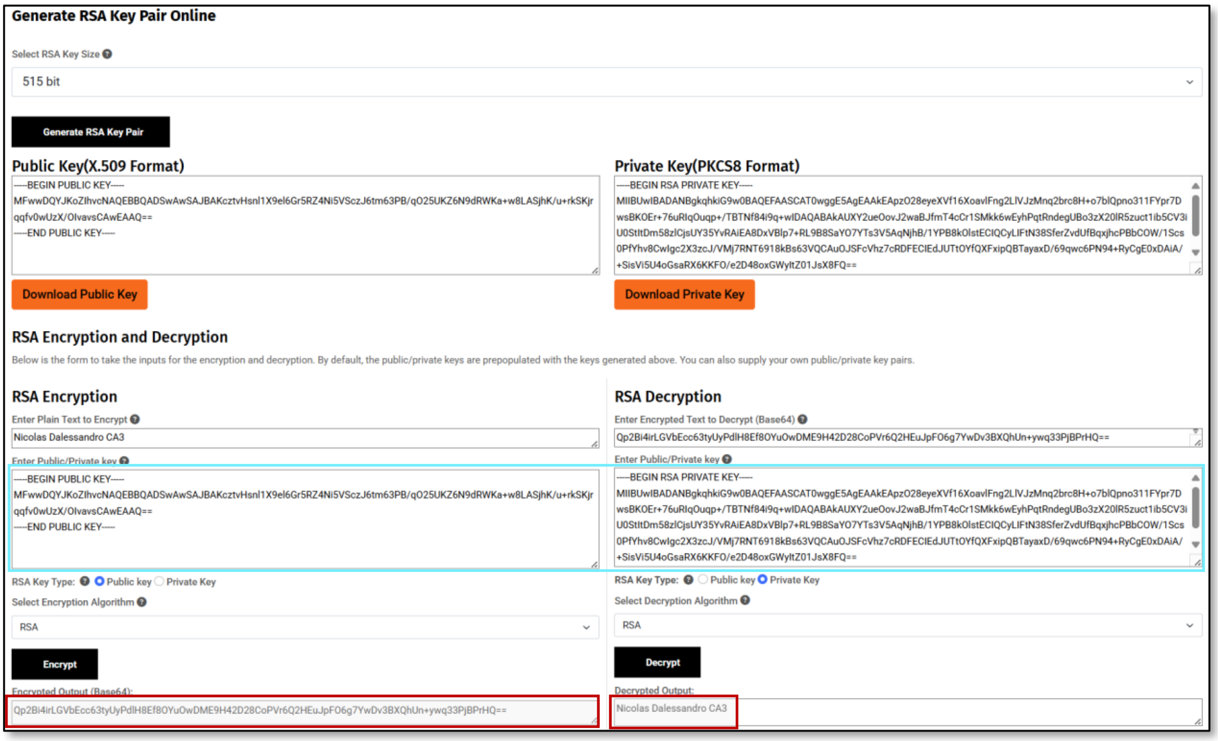
d) Because the RSA output size always matches the size of the key and not the size of the input. If we use the formula described above, $\mathbf{c = m^e \bmod n}$, the result **c** is a number less than **n**. If the RSA key has a length of 512 bits as in our example, the output will always be 512 bits, no matter the size of the input message.
In our case, even the input message was short, the encrypted output is always 512 bits (88 characters in base64, with 2 '=' at the end).

e)



I changed the first character of the requested input field, and as we can see in the image above, the error obtained is "**Padding error in decryption**". This demonstrates that the message cannot be altered, if someone modifies the ciphertext, even just one character as I did, the decryption process fails, ensuring that the message integrity is preserved.

f)



As we can see, when we click **Encrypt** again using the same *public key*, the encrypted output changes but the message can be still decrypted successfully using the same *private key*.
This happens because the RSA encryption process uses a padding schema that includes some random bytes, for example:

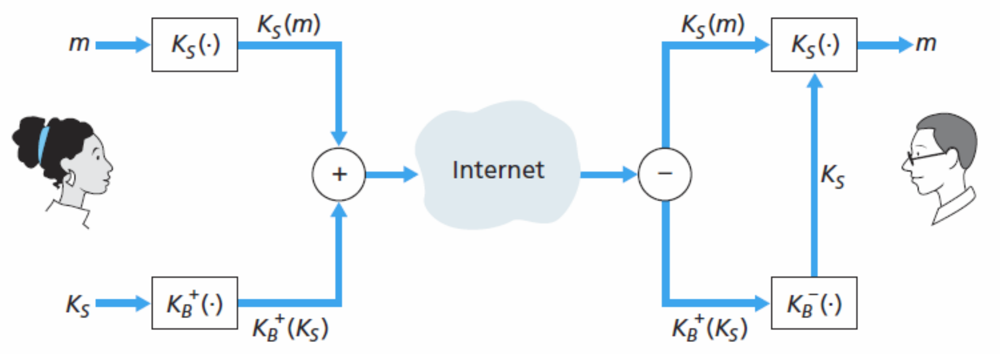
0x00 0x02 [some non-zero bytes] 0x00 [message bytes]

This new "message with padding" is what it gets encrypted. When we use the private key to decrypt the message, the system knows how to identify and remove this padding (because is a standard structure), so that we can recover the original message.

Sources:

- *Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.2.2 – Public Key Encryption, page 650-652.*
- *https://medium.com/asecuritysite-when-bob-met-alice/so-how-does-padding-work-in-rsa-6b34a123ca1f*
- *https://www.splunk.com/en_us/blog/learn/rsa-algorithm-cryptography.html*

4.



This chapter covers different approaches in developing a secure email communication between Alice and Bob, applying the different cryptographic principles described in previous units (also covered in previous questions of this practice).

This figure 8.19 shows how Alice can send Bob a secure e-mail using a combination of symmetric encryption (secret key $\mathbf{K_s}$) and asymmetric encryption (public key $\mathbf{K_B^+}$):

Alice point of view (sender):

1. Choose a random secret key $\mathbf{K_s}$ that will be only used for this message.
2. Encrypts the message **m** using that key **=>** $\mathbf{K_s(m)}$.
3. Encrypts the secret key $\mathbf{K_s}$ with Bob´s public key $\mathbf{K_B^+}$ **=>** $\mathbf{K_B^+(K_s)}$
4. Concatenates both results **=>** $\mathbf{K_s(m) + K_B^+(K_s)}$.
5. Send the packet trough internet to Bob.

Bob point of view (recipient):

1. Receives $\mathbf{K_s(m)}$ and $\mathbf{K_B+(K_s)}$.
2. Decrypts $\mathbf{K_B^+(K_s)}$ with his private key $\mathbf{K_B^-}$ **=> gets the session key** $\mathbf{K_s}$.
3. Use the obtained secret key $\mathbf{K_s}$ to decrypt $\mathbf{K_s(m)}$ **=> gets the message m.**

As explained in the book, this figure **provides only confidentialit**y. Using a combination of symmetric encryption $\mathbf{K_s(m)}$ and asymmetric encryption to protect the session key $\mathbf{(K_B+(K_s))}$**,** ensures **confidentiality** because only Bob with his private key can recover $\mathbf{K_s}$ and read the message **m**, but it **does not guarantee integrity** because there is no verification that the message has not been modified along the way and also **there is no guarantee that it was Alice who send the message (authenticity).**

To achieve this, the book introduces two new concepts covered in figures 8.20 and 8.21. This last one combines the concepts explained in figures 8.19 + 8.20 to achieve the three security services required.

Figures 8.20 uses a Hash of the message **H(m)** and a digital signature with Alice private key **=>** $\mathbf{K_A^-(H(m))}$**.** Bob can verify the **authenticity** and **integrity** of the message using Alice public key $\mathbf{(K_A^+)}$, but as explained in the book, this process **does not provide confidentiality**.

Finally, figure 8.21 combines these two approaches to provide the complete security services Confidentiality, Authenticity and Integrity:

1. Alice creates a $\mathbf{m + K_A\text{-}(H(m))}$ to **guarantee authenticity and integrity** as in figure 8.20**.**
2. Encrypts everything with the symmetric secret key $\mathbf{K_s}$ to **provide confidentiality.**
3. Additionally, $\mathbf{K_s}$ is protected with $\mathbf{K_B^+}$ (Bob public key) as in figure 8.19.

5. Main fields in a X.509 certificate from the section 8.4 of the learning material:

Field Name	Functionality and Justification
Version	Specifies the version or standard of the X509 used. Ensures compatibility with the recipient software.
Serial number	Unique identifier assigned by the CA that helps to reference or revoke the certificate.
Signature	Indicates the algorithm used by the CA to digitally sign the certificate
Issuer name	DN of the CA that signed the certificate.
Validity period	Start and end date that indicates how long the certificate is considered valid and trusted.
Subject name	DN of the certificate owner (person, organization, server, etc.).
Subject public key	Public key that its being certificated and the associated algorithm.

**CA: Certification Authority*
**DN: Distinguished Name*

A **digital signature** is a cryptographic technique where the CA signs a hash of the certificate data using its private key.
The signature is added to the certificate, and anyone with the CA public key can verifies it.
This signature is used to ensure authenticity and integrity and is what turns a public key into a **trusted certificate**.

To generate the **digital signature** (in the case of a certificate), the subject sends its public key and identification data to a Certification Authority. The CA generates a hash (summary) of the certificate content and then encrypts the hash using its private key (this is the digital signature).
The resulting certificate contains the subject original information, the digital signature and metadata signed by the CA.

The CA digital signature is what transforms a document with public key into **a trusted certificate**. Anyone with the CA public key can verify the signature. If the signature is valid, we can trust that the public key belongs to the claimed proprietary (a web server, a person, etc.).

Sources:

- *Computer Networking: A Top-Down Approach, Kurose & Ross, 8th edition, Chapter 8: Security in Computer Networks, Section 8.4 – End-Point Authentication, page 664.*
- *https://www.geeksforgeeks.org/x-509-authentication-service/*

6. IPsec mechanism:

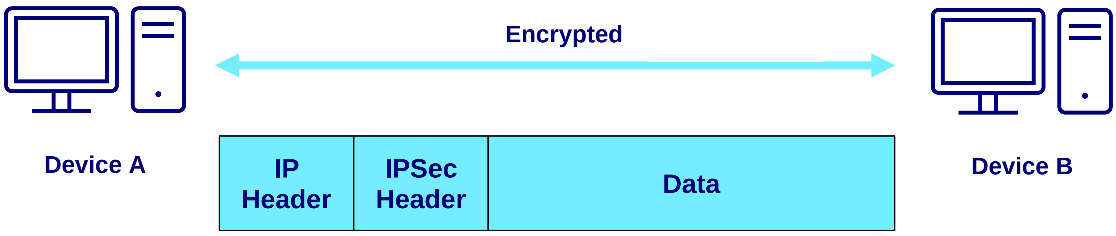
a) TLS/SSL and IPSec are both used to protect data, but they work at different layers of the network.

1. **TLS** operates at the transport layer. It protects data in applications such as web browsers, emails, chats, etc. For example, when visiting a website with HTTPS, TLS is being used.
2. **IPSec** operates at the IP network layer, and it is aimed to protect traffic between two devices. As the book explain, it is often used to create VPNs.

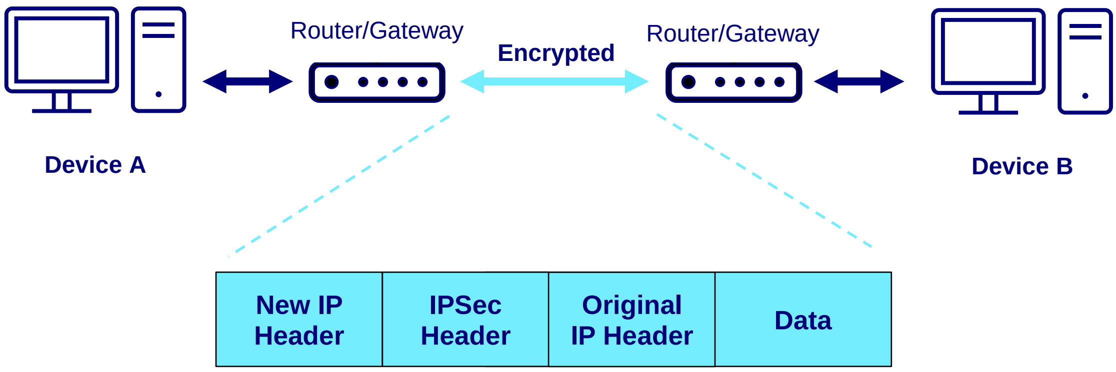
In summary, **TLS works at the transport layer and protects specific apps** (like browsers). IPsec **works at the network layer and protects all data between two IP addresses**.

b) IPsec Modes:

1. **Trasport Mode**: In transport mode the security is provided to the traffic end to end. Only the data part of the IP packet is encrypted. The original IP address is visible. Commonly used when the sender and receiver are the final devices.



2. **Tunnel Mode**: In tunnel mode the protection is provided from the router or gateway of one network, to the router or gateway of another network. The whole IP packet (also the IP address) is encrypted.



Source: https://www.omscs-notes.com/information-security/ipsec-and-tls/

c) Selected IPSec mode:

1. **Sender <=IPSec=> Receiver**
   Similar to our example, the best choice is **Transport Mode** because both ends are the final devices.

2. **Sender <-IP-> Router 1 <=IPSec=> Router 2 <-IP-> Receiver**
   Also similar to the above diagram, the best choice here is Tunnel Mode because the encryption is between two routers.

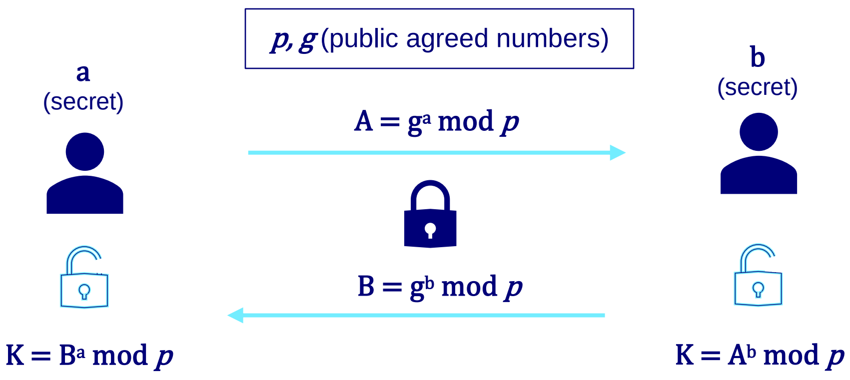
3. **Sender <=IPSec=> Router <-IP-> Receiver**
   The best choice here is Tunnel Mode because the encryption is only between the sender and a router.

7. "Diffie–Hellman" mechanism:

a) This algorithm is a protocol for exchanging keys, allowing two parts to communicate in a public channel (such as the internet) to establish a mutual secret.
In networking, this **allows two devices to agree on a secret key** without the need to send it directly. A mathematical technique is used to derive the key, that will be then used to encrypt messages.

b) How does this work:

1. Two public keys (numbers) are agreed, p (a prime number) and g (generator number).
2. Each part chooses a secret number (private key) a and b (integers and less than p).
3. Each part sends a computed value using the formula:
   $$A = g^a \bmod p$$
   $$B = g^b \bmod p$$
4. Both parts calculate the shared secret key using the formula:
   $$K = B^a \bmod p$$
   $$K = A^b \bmod p$$
5. Both get the same result K which is the shared key.



c) **Ephemeral Diffie-Hellman** is when the key exchange is done only once per session and then discarded to make the connection safer.
In **HTTPS** this method is used to create **session keys** that change every time, so if one key is stolen, other sessions are safe.

d) Example of the calculation:

- Public numbers: p (17) and g (3).
- Each part secret number (private key) a (12) and b (6).

Step 1: Values to be sent:

$$A = g^a \bmod p = 3^{12} \bmod 17 = 4$$
$$B = g^b \bmod p = 3^6 \bmod 17 = 16$$

Step 2: Calculate the shared key

$$K = B^a \bmod p = 16^{12} \bmod 17 = 1$$
$$K = A^b \bmod p = 4^6 \bmod 17 = 1$$

Both get the same result $K=1$ which is the shared key.

Sources:

- *https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/subhasish-sarkar1/2020/07/09/understanding-diffiehellman-key-exchange-method?communityKey=8bcc5745-f7d5-43e3-a7ef-a4801c68c752*
- https://security.stackexchange.com/questions/45963/diffie-hellman-key-exchange-in-plain-english

8. FYI