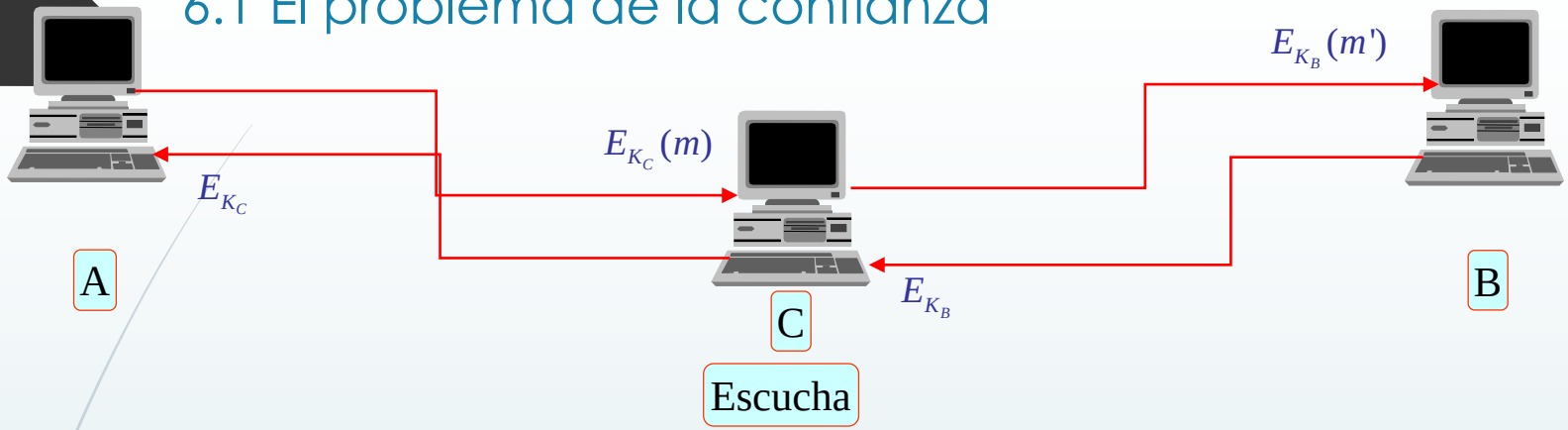
6.- Infraestructuras de clave pública

6.1 El problema de la confianza
6.2 Autoridad certificadora
6.3 Protocolos de comunicación segura

6.1 El problema de la confianza



- El uso de la criptografía de clave pública no garantiza la seguridad de las comunicaciones, ya que estas se pueden atacar por un intermediario.
- El escucha se hace pasar por los usuarios A y B y facilita sus claves públicas (no las de A y B).
- De esta manera puede descifrar y alterar toda la información que se comuniquen A y B.
- Para evitar este tipo de ataque se hace uso de una tercera parte de confianza que certifica la identidad de cada usuario.

6.1 El problema de la confianza

- Para solucionar el problema de la **Autentificación** en las transacciones por Internet se buscó algún sistema **identificativo único** de una entidad o persona.

- Ya existían los sistemas criptográficos de clave asimétrica, mediante los cuales una persona disponía de dos claves, una pública, al alcance de todos, y otra privada, sólo conocida por el propietario.

- Cuando deseara enviar un mensaje confidencial a otra persona, bastaría con cifrarlo con la clave pública del destinatario, y así estaría seguro de que sólo el destinatario correcto podría obtener el mensaje en claro.

- Ahora bien, el **problema era estar seguro de que efectivamente la clave pública que nos envía el receptor corresponde a la persona correcta** y no a un suplantador.

6.1 El problema de la confianza

- La solución a este problema la trajo la aparición de los **Certificados Digitales** o Certificados Electrónicos, documentos electrónicos basados en la criptografía de clave pública y en el sistema de firmas digitales.

- La misión principal de un Certificado Digital es **garantizar con toda confianza el vínculo existente entre una persona, entidad o servidor con la clave que se hace pública** de la pareja de claves correspondientes a un sistema criptográfico asimétrico.

6.2 Autoridad certificadora

CONCEPTO DE CERTIFICADO DIGITAL

- Un certificado digital **es un documento electrónico que contiene datos identificativos de una persona o entidad** (empresa, servidor web, etc.) y la clave pública de la misma, haciéndose responsable de la autenticidad de los datos que figuran en el certificado otra persona o entidad de confianza, denominada **Autoridad Certificadora** (AC).

- Si el certificado es auténtico y confiamos en la AC, entonces, podemos confiar en que el sujeto identificado en el certificado digital posee la clave pública que se señala en dicho certificado.

- ***Así pues, si un sujeto firma un documento y anexa su certificado digital, cualquiera que conozca la clave pública de la AC podrá autentificar el documento***

6.2 Autoridad certificadora

CONCEPTO DE CERTIFICADO DIGITAL

- El formato de los certificados digitales es estándar, siendo X.509 v3 el recomendado por la Unión Internacional de Comunicaciones (ITU) y el que está en vigor en la actualidad.

- Los datos que figuran generalmente en un certificado son:
  - **Versión**: versión del estándar X.509, generalmente la 3, que es la más actual.
  - **Número de serie**: número identificador del certificado, único para cada certificado expedido por una AC determinada.
  - **Algoritmo de firma**: algoritmo criptográfico usado para la firma digital.
  - **Autoridad Certificadora**: datos sobre la autoridad que expide el certificado.
  - **Fechas de inicio y de fin de validez del certificado**: Definen el periodo de validez del mismo, que generalmente es de un año.
  - **Propietario**: persona o entidad vinculada al certificado. Dentro de este apartado se usan una serie de abreviaturas para establecer datos de identidad.
  - **Clave pública:** representación de la clave pública vinculada a la persona o entidad (en hexadecimal), junto con el algoritmo criptográfico para el que es aplicable.
  - **Algoritmo** usado por la **Autoridad Certificadora** firmar la clave pública.
  - **Firma de la Autoridad Certificadora**, que asegura la autenticidad del mismo.
  - **Información adicional**, como tipo de certificado, etc.

6.2 Autoridad certificadora

CONCEPTO DE CERTIFICADO DIGITAL

- El **certificado digital vincula**, pues, indisolublemente a una **persona o entidad** con una **clave pública**, y mediante el sistema de firma digital se asegura que el certificado que recibimos es realmente de la persona o entidad que consta en el mismo.

- Los procesos de validación de certificados, obtención de resúmenes, descifrados y comprobación de coincidencia se realizan por el software adecuado del navegador web o programa de seguridad particular de forma transparente al usuario, por lo que éste será informado sólo en el caso de que el certificado no sea válido.

Visualizar certificados con mmc.
Observar que en el campo clave pública contiene el identificador de RSA y a continuación el módulo n de 2048 bits y el exponente e en hexadecimal.
Habitualmente $e=010001_{(16}=65537$

6.2 Autoridad certificadora

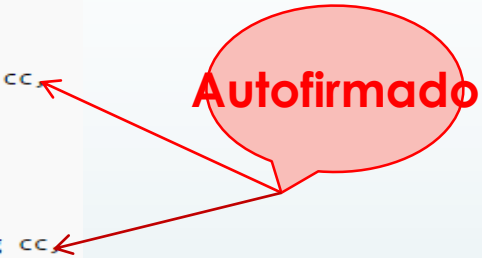
CONCEPTO DE CERTIFICADO DIGITAL

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/Email=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/Email=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

6.2 Autoridad certificadora

CONCEPTO DE CERTIFICADO DIGITAL

```
Certificate:
   Data:
       Version: 3 (0x2)
       Serial Number: 1 (0x1)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/Email=server-certs@thawte.com
       Validity
           Not Before: Aug  1 00:00:00 1996 GMT
           Not After : Dec 31 23:59:59 2020 GMT
       Subject: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
                OU=Certification Services Division,
                CN=Thawte Server CA/Email=server-certs@thawte.com
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:d3:a4:50:6e:c8:ff:56:6b:e6:cf:5d:b6:ea:0c:
                   68:75:47:a2:aa:c2:da:84:25:fc:a8:f4:47:51:da:
                   85:b5:20:74:94:86:1e:0f:75:c9:e9:08:61:f5:06:
                   6d:30:6e:15:19:02:e9:52:c0:62:db:4d:99:9e:e2:
                   6a:0c:44:38:cd:fe:be:e3:64:09:70:c5:fe:b1:6b:
                   29:b6:2f:49:c8:3b:d4:27:04:25:10:97:2f:e7:90:
                   6d:c0:28:42:99:d7:4c:43:de:c3:f5:21:6d:54:9f:
                   5d:c3:58:e1:c0:e4:d9:5b:b0:b8:dc:b4:7b:df:36:
                   3a:c2:b5:66:22:12:d6:87:0d
               Exponent: 65537 (0x10001)
       X509v3 extensions:
           X509v3 Basic Constraints: critical
               CA:TRUE
   Signature Algorithm: md5WithRSAEncryption
       07:fa:4c:69:5c:fb:95:cc:46:ee:85:83:4d:21:30:8e:ca:d9:
       a8:6f:49:1a:e6:da:51:e3:60:70:6c:84:61:11:a1:1a:c8:48:
       3e:59:43:7d:4f:95:3d:a1:8b:b7:0b:62:98:7a:75:8a:dd:88:
       4e:4e:9e:40:db:a8:cc:32:74:b9:6f:0d:c6:e3:b3:44:0b:d9:
       8a:6f:9a:29:9b:99:18:28:3b:d1:e3:40:28:9a:5a:3c:d5:b5:
       e7:20:1b:8b:ca:a4:ab:8d:e9:51:d9:e2:4c:2c:59:a9:da:b9:
       b2:75:1b:f6:42:f2:ef:c7:f2:18:f9:89:bc:a3:ff:8a:23:2e:
       70:47
```



6.2 Autoridad certificadora

- El uso de la criptografía asimétrica plantea el problema de cómo **asegurar que la clave pública** de un usuario **corresponde** realmente al mismo y no ha sido falsificada por otro.

- La solución más ampliamente adoptada consiste en recurrir a una **tercera parte confiable**, erigida en la figura de una **Autoridad Certificadora (AC)**.

- La **función básica** de una **AC** consiste en **verificar la identidad** de los solicitantes de certificados, **crear los certificados** y **publicar listas de revocación** cuando éstos son inutilizados.

- El certificado **contiene** de forma estructurada información acerca de la **identidad de su titular, su clave pública y la AC que lo emitió**.

- La **confianza** de los **usuarios** en la **Autoridad Certificadora** es fundamental para el buen funcionamiento del servicio.
  - Caso sysmatec

6.2 Autoridad certificadora

- El entorno de seguridad (control de acceso, cifrado, etc.) de la AC ha de ser muy fuerte, en particular en lo que respecta a **la protección de la Clave Privada** que utiliza para firmar sus emisiones.

- Si este secreto se viera comprometido, toda la infraestructura de Clave Pública (PKI) se vendría abajo.

6.2 Autoridad certificadora

- Con el tiempo, una Autoridad Certificadora puede verse fácilmente desbordada si cubre un área geográfica muy extensa o muy poblada, por lo que a menudo delega en las llamadas ***Autoridades Registradoras*** (**AR**) la labor de verificar la identidad de los solicitantes.

- Las **AR** pueden abrir multitud de oficinas regionales dispersas por un gran territorio, llegando hasta los usuarios en los sitios más remotos, mientras que la AC se limitaría así a certificar a todos los usuarios aceptados por las AR dependientes de ella.

- **Gracias a esta descentralización se agiliza el proceso de certificación y se aumenta la eficacia en la gestión de solicitudes.**

6.2 Autoridad certificadora

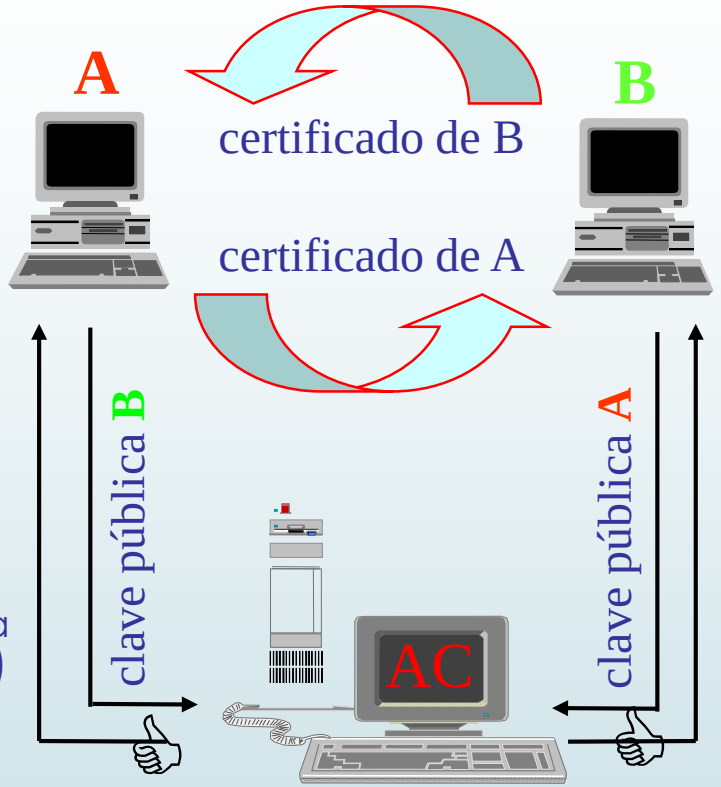
- La AC se encarga de realizar las siguientes tareas:
  - **Emisión de los certificados de usuarios** registrados y validados por la Autoridad Registradora.
  - **Revocación de los certificados** que ya no sean válidos (CRL - lista de certificados revocados). Un certificado puede ser revocado por que los **datos han dejado de ser válidos**, la **clave privada ha sido comprometida** o el c**ertificado ha dejado de tener validez** dentro del contexto para el que había sido emitido.
  - **Renovación de certificados**.
  - **Publicar certificados** en el directorio repositorio de certificados.

6.2 Autoridad certificadora

- Las **Autoridades Registradoras** realizan las siguientes tareas:
  - **Validar solicitudes de certificado en base a determinados procedimientos de identificación**, apropiados a los niveles de seguridad que ofrece cada categoría de certificado (políticas de seguridad).
  - **Mandar las peticiones de generación de certificados a la Autoridad de Certificación**, para que esta los firme con su clave privada.
  - **Recibir los certificados solicitados** a la Autoridad de Certificación.
  - **Entregar físicamente los certificados a los solicitantes**, por cualquier medio (e-mail, disquete, …)
  - **Informar a los usuarios de la necesidad de la renovación** de su certificado.
  - **Petición de revocación de un certificado** (también puede solicitarlo el propio usuario).

6.2 Autoridad certificadora

- Autoridad de Certificación es un ente u organismo que, de acuerdo con unas políticas y algoritmos, certificará (por ejemplo) claves públicas de usuarios o servidores.
- El usuario **A** enviará al usuario **B** su certificado (clave pública y otros datos firmados por AC) y éste comprobará con esa autoridad su autenticidad.
- Lo mismo en sentido contrario.



Autoridad de Certificación AC

6.2 Autoridad certificadora

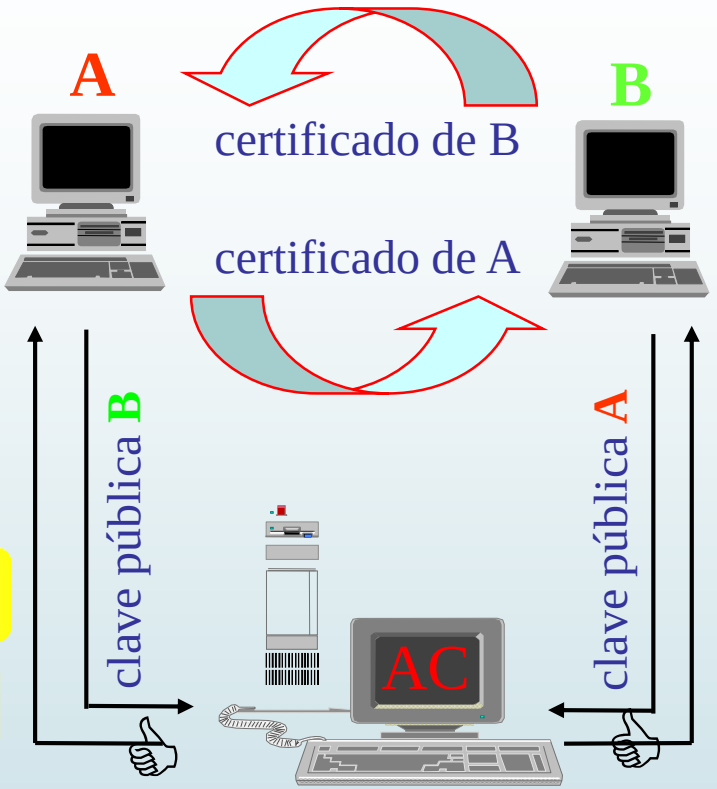
A K_A k_A E_{K_A} D_{k_A}

B K_B k_B E_{K_B} D_{k_B}

AC K_{AC} k_{AC} $E_{K_{AC}}$ $D_{k_{AC}}$

Certificado de A $c_A = D_{k_{AC}}(K_A)$

Certificado de B $c_B = D_{k_{AC}}(K_B)$



Autoridad de Certificación AC

Autentificación de A $E_{K_{AC}}(c_A) = K_A$

Autenticación de B $E_{K_{AC}}(c_B) = K_B$

6.2 Autoridad certificadora - RSA

A n_A e_A d_A

B n_B e_B d_B

AC n_{AC} e_{AC} d_{AC}

Certificado de A, $c_A = D_{k_{AC}}(e_A) = e_A^{\,d_{AC}} \bmod n_{AC}$

Certificado de B, $c_B = D_{k_{AC}}(e_B) = e_B^{\,d_{AC}} \bmod n_{AC}$

Autentificación de A $E_{K_{AC}}(c_A) = e_A$

Autenticación de B $E_{K_{AC}}(c_B) = e_B$

6.2 Autoridad certificadora

EJEMPLO RSA

Una autoridad certificadora (AC) tiene clave pública RSA e_{AC}= 47, siendo
n_{AC}= 899.

Benito (B) tiene clave pública RSA e_B= 611, siendo n_B= 851, y un certificado de e_B expedido por la autoridad AC con valor c_B= 530.

Alicia desea enviar un mensaje cifrado a Benito y quiere tener la seguridad de que la clave pública de Benito es e_B.

¿Cómo puede verificar que la clave pública de Benito es auténtica?

Compruébalo aplicando el protocolo correspondiente y efectuando los cálculos pertinentes.

6.2 Autoridad certificadora

EJEMPLO RSA

Sabemos que $c_B = D_{k_{AC}}(e_B) = 530$

Para verificar que la clave de Benito es auténtica, Alicia debe comprobar que

$e_B = E_{K_{AC}}(c_B)$

Apliquemos el protocolo con los cálculos pertinentes para comprobar que la clave es auténtica.

$E_{K_{AC}}(c_B) = c_B^{e_{AC}} \bmod n_{AC} = 530^{47} \bmod 899 = \mathbf{611} = e_B$

Por lo que la clave pública de Benito es auténtica.

530^1	mod	899 =					=	**530**	**1**		
530^2	mod	899 =	530^2	mod	899 =	280900 mod 899 =		**412**	**1**		
530^4	mod	899 =	412^2	mod	899 =	169744 mod 899 =		**732**	**1**		
530^8	mod	899 =	732^2	mod	899 =	535824 mod 899 =		**20**	**1**		
530^{16}	mod	899 =	20^2	mod	899 =	400 mod 899 =		400	**0**		
530^{32}	mod	899 =	400^2	mod	899 =	160000 mod 899 =		**877**	**1**		

530 . **412**	mod	899 =	802
802 . **732**	mod	899 =	17
17 . **20**	mod	899 =	340
340 . **877**	mod	899 =	**611**

$47_{(2)} = \mathbf{101111}$

6.3 - Protocolos de comunicación segura

Ejemplo de protocolo de comunicación segura

- Signal Protocol
  - Desarrollado por Open Whisper Systems en 2013
  - Inicialmente para la app open-source ***TextSecure***, que finalmente se llamó ***Signal***.
- Usado por múltiples sistemas de comunicación:
  - WhatsApp
  - Facebook Messenger
  - Google Allo
- El 5 de abril de 2016, WhatsApp publicó un whitepaper explicando su implementación del protocolo.

- Cifrado punto a punto
- Protege las confidencialidad de las comunicaciones de terceras partes y de los servidores de WhatsApp.
- Si las claves de un dispositivo son comprometidas, no pueden ser usadas para descifrar mensajes del pasado.

Terms

Public Key Types

- `Identity Key Pair` – A long-term Curve25519 key pair, generated at install time.
- `Signed Pre Key` – A medium-term Curve25519 key pair, generated at install time, signed by the `Identity Key`, and rotated on a periodic timed basis.
- `One-Time Pre Keys` – A queue of Curve25519 key pairs for one time use, generated at install time, and replenished as needed.

Session Key Types

- `Root Key` – A 32-byte value that is used to create `Chain Keys`.
- `Chain Key` – A 32-byte value that is used to create `Message Keys`.
- `Message Key` – An 80-byte value that is used to encrypt message contents. 32 bytes are used for an AES-256 key, 32 bytes for a HMAC-SHA256 key, and 16 bytes for an IV.

Client Registration

At registration time, a WhatsApp client transmits its public `Identity Key`, public `Signed Pre Key` (with its signature), and a batch of public `One-Time Pre Keys` to the server. The WhatsApp server stores these public keys associated with the user's identifier. At no time does the WhatsApp server have access to any of the client's private keys.

Initiating Session Setup

To communicate with another WhatsApp user, a WhatsApp client first needs to establish an encrypted session. Once the session is established, clients do not need to rebuild a new session with each other until the existing session state is lost through an external event such as an app reinstall or device change.

To establish a session:

1. The initiating client ("initiator") requests the public `Identity Key`, public `Signed Pre Key`, and a single public `One-Time Pre Key` for the recipient.
2. The server returns the requested public key values. A `One-Time Pre Key` is only used once, so it is removed from server storage after being requested. If the recipient's latest batch of `One-Time Pre Keys` has been consumed and the recipient has not replenished them, no `One-Time Pre Key` will be returned.
3. The initiator saves the recipient's `Identity Key` as $I_{recipient}$, the `Signed Pre Key` as $S_{recipient}$, and the `One-Time Pre Key` as $O_{recipient}$.
4. The initiator generates an ephemeral Curve25519 key pair, $E_{initiator}$.
5. The initiator loads its own `Identity Key` as $I_{initiator}$.
6. The initiator calculates a master secret as `master_secret` = $ECDH(I_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, I_{recipient}) \,||\, ECDH(E_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, O_{recipient})$. If there is no `One Time Pre Key`, the final ECDH is omitted.
7. The initiator uses HKDF to create a `Root Key` and `Chain Keys` from the `master_secret`.

- **ECDH:** Elliptic-curve Diffie-Hellman
- **HKDF:** simple key derivation function basado en HMAC
- **HMAC:** hash-based message authentication

Receiving Session Setup

After building a long-running encryption session, the initiator can immediately start sending messages to the recipient, even if the recipient is offline. Until the recipient responds, the initiator includes the information (in the header of all messages sent) that the recipient requires to build a corresponding session. This includes the initiator's $E_{initiator}$ and $I_{initiator}$.

When the recipient receives a message that includes session setup information:

1. The recipient calculates the corresponding `master_secret` using its own private keys and the public keys advertised in the header of the incoming message.
2. The recipient deletes the `One-Time Pre Key` used by the initiator.
3. The initiator uses HKDF to derive a corresponding `Root Key` and `Chain Keys` from the `master_secret`.

Exchanging Messages

Once a session has been established, clients exchange messages that are protected with a `Message Key` using `AES256` in CBC mode for encryption and `HMAC-SHA256` for authentication.

The `Message Key` changes for each message transmitted, and is ephemeral, such that the `Message Key` used to encrypt a message cannot be reconstructed from the session state after a message has been transmitted or received.

The `Message Key` is derived from a sender's `Chain Key` that "ratchets" forward with every message sent. Additionally, a new ECDH agreement is performed with each message roundtrip to create a new `Chain Key`. This provides forward secrecy through the combination of both an immediate "hash ratchet" and a round trip "DH ratchet."

Calculating a `Message Key` from a `Chain Key`

Each time a new `Message Key` is needed by a message sender, it is calculated as:

1. `Message Key = HMAC-SHA256(Chain Key, 0x01)`.
2. The `Chain Key` is then updated as `Chain Key = HMAC-SHA256(Chain Key, 0x02)`.

This causes the `Chain Key` to "ratchet" forward, and also means that a stored `Message Key` can't be used to derive current or past values of the `Chain Key`.

Calculating a Chain Key from a Root Key

Each time a message is transmitted, an ephemeral Curve25519 public key is advertised along with it. Once a response is received, a new Chain Key and Root Key are calculated as:

1. ephemeral_secret = ECDH($\text{Ephemeral}_{\text{sender}}$, $\text{Ephemeral}_{\text{recipient}}$).

2. Chain Key, Root Key = HKDF(Root Key, ephemeral_secret).

A chain is only ever used to send messages from one user, so message keys are not reused. Because of the way Message Keys and Chain Keys are calculated, messages can arrive delayed, out of order, or can be lost entirely without any problems.

Transmitting Media and Other Attachments

Large attachments of any type (video, audio, images, or files) are also end-to-end encrypted:

1. The WhatsApp user sending a message ("sender") generates an ephemeral 32 byte `AES256` key, and an ephemeral 32 byte `HMAC-SHA256` key.
2. The sender encrypts the attachment with the `AES256` key in CBC mode with a random IV, then appends a MAC of the ciphertext using HMAC-SHA256.
3. The sender uploads the encrypted attachment to a blob store.
4. The sender transmits a normal encrypted message to the recipient that contains the encryption key, the HMAC key, a SHA256 hash of the encrypted blob, and a pointer to the blob in the blob store.
5. The recipient decrypts the message, retrieves the encrypted blob from the blob store, verifies the SHA256 hash of it, verifies the MAC, and decrypts the plaintext.

Group Messages

Traditional unencrypted messenger apps typically employ “server-side fan-out” for group messages. A client wishing to send a message to a group of users transmits a single message, which is then distributed N times to the N different group members by the server.

This is in contrast to “client-side fan-out,” where a client would transmit a single message N times to the N different group members itself.

Messages to WhatsApp groups build on the pairwise encrypted sessions outlined above to achieve efficient server-side fan-out for most messages sent to groups. This is accomplished using the “Sender Keys” component of the Signal Messaging Protocol.

The first time a WhatsApp group member sends a message to a group:

1. The sender generates a random 32-byte `Chain Key`.

2. The sender generates a random Curve25519 `Signature Key` key pair.

3. The sender combines the 32-byte `Chain Key` and the public key from the `Signature Key` into a `Sender Key` message.

1. The sender generates a random 32-byte `Chain Key`.

2. The sender generates a random Curve25519 `Signature Key` key pair.
3. The sender combines the 32-byte `Chain Key` and the public key from the `Signature Key` into a `Sender Key` message.
4. The sender individually encrypts the `Sender Key` to each member of the group, using the pairwise messaging protocol explained previously.

For all subsequent messages to the group:

1. The sender derives a `Message Key` from the `Chain Key`, and updates the `Chain Key`.
2. The sender encrypts the message using `AES256` in CBC mode.
3. The sender signs the ciphertext using the `Signature Key`.
4. The sender transmits the single ciphertext message to the server, which does server-side fan-out to all group participants.

The “hash ratchet” of the message sender’s `Chain Key` provides forward secrecy. Whenever a group member leaves, all group participants clear their `Sender Key` and start over.

Call Setup

WhatsApp voice and video calls are also end-to-end encrypted.
When a WhatsApp user initiates a voice or video call:

1. The initiator builds an encrypted session with the recipient (as outlined in Section *Initiating Session Setup*), if one does not already exist.
2. The initiator generates a random 32-byte SRTP master secret.
3. The initiator transmits an encrypted message to the recipient that signals an incoming call, and contains the SRTP master secret.
4. If the responder answers the call, a SRTP encrypted call ensues.

Verifying Keys

WhatsApp users additionally have the option to verify the keys of the other users with whom they are communicating so that they are able to confirm that an unauthorized third party (or WhatsApp) has not initiated a man-in-the-middle attack. This can be done by scanning a QR code, or by comparing a 60-digit number.

The QR code contains:

1. A version.
2. The user identifier for both parties.
3. The full 32-byte public `Identity Key` for both parties.

When either user scans the other's QR code, the keys are compared to ensure that what is in the QR code matches the `Identity Key` as retrieved from the server.

Transport Security

All communication between WhatsApp clients and WhatsApp servers is layered within a separate encrypted channel. On Windows Phone, iPhone, and Android, those end-to-end encryption capable clients use Noise Pipes with Curve25519, AES-GCM, and SHA256 from the Noise Protocol Framework for long running interactive connections.

This provides clients with a few nice properties:

1. Extremely fast lightweight connection setup and resume.

2. Encrypts metadata to hide it from unauthorized network observers. No information about the connecting user's identity is revealed.

3. No client authentication secrets are stored on the server. Clients authenticate themselves using a Curve25519 key pair, so the server only stores a client's public authentication key. If the server's user database is ever compromised, no private authentication credentials will be revealed.