WhatsApp Encryption

Harpreet Singh
15313778

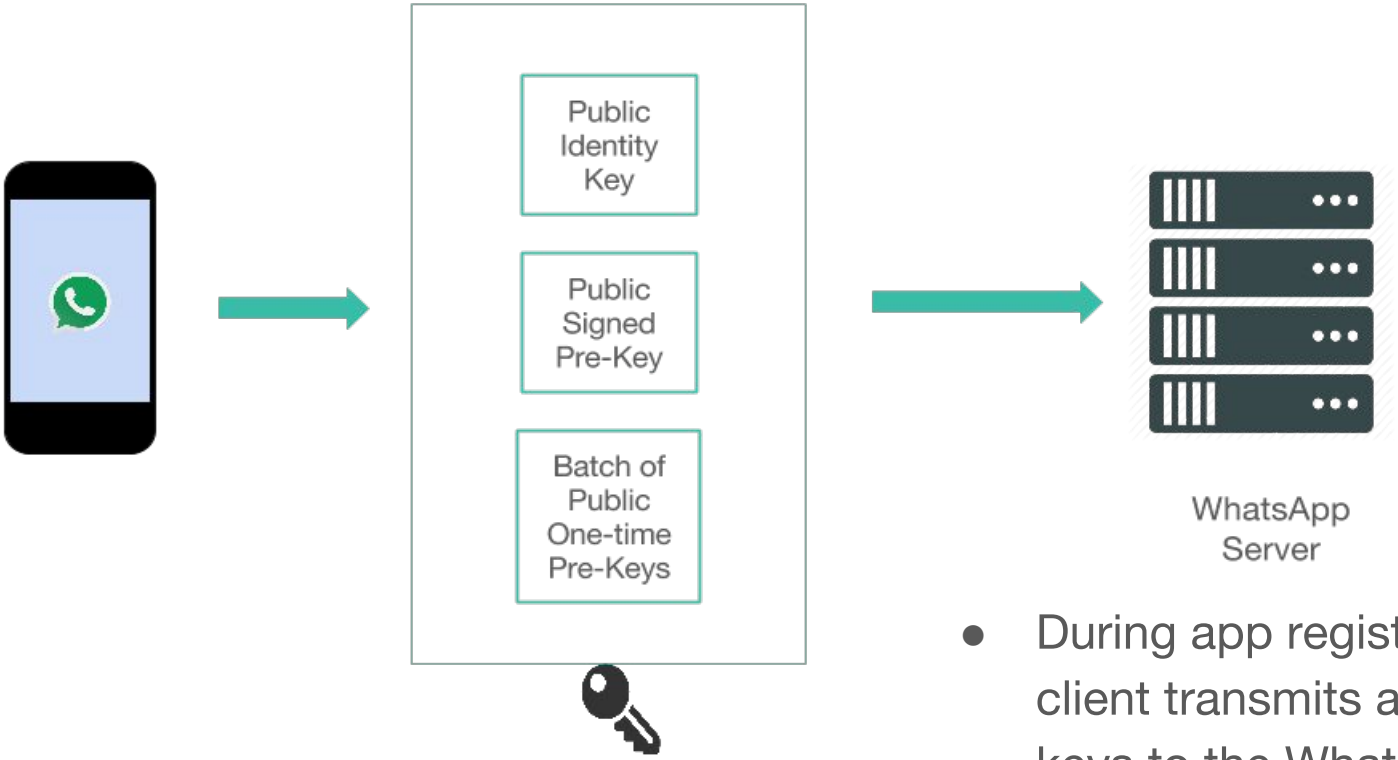
Introduction

- WhatsApp enabled end to end Encryption in April, 2016 for all communication via the application
- It uses the Signal Protocol from Open Whisper Systems, which is open source at https://github.com/WhisperSystems

Public Key Types

- The protocol uses 3 types of public keys
- Identity Key Pair- Elliptic Curve (Curve25519) key pair, generated during app installation.
    - Has a Long-term Usage
- Signed Pre Key- Elliptic Curve (Curve25519) key pair, generated during app installation and signed by the Identity Key
- One-Time Pre Keys - A queue of Elliptic Curve (Curve25519) key pairs for one time usage.
    - Generated during install time and get replenished as required

Client Registration



- During app registration, the client transmits all its public keys to the WhatsApp Server

Initiating Session Setup

- In order to communicate with another WhatsApp user, the WhatsApp client needs to establish an Encrypted Session with the user
- Once this encrypted session is established, the users will not build a new session, and the same session is used throughout
- New Session is only built if the app is reinstalled

Calculating Master Secret

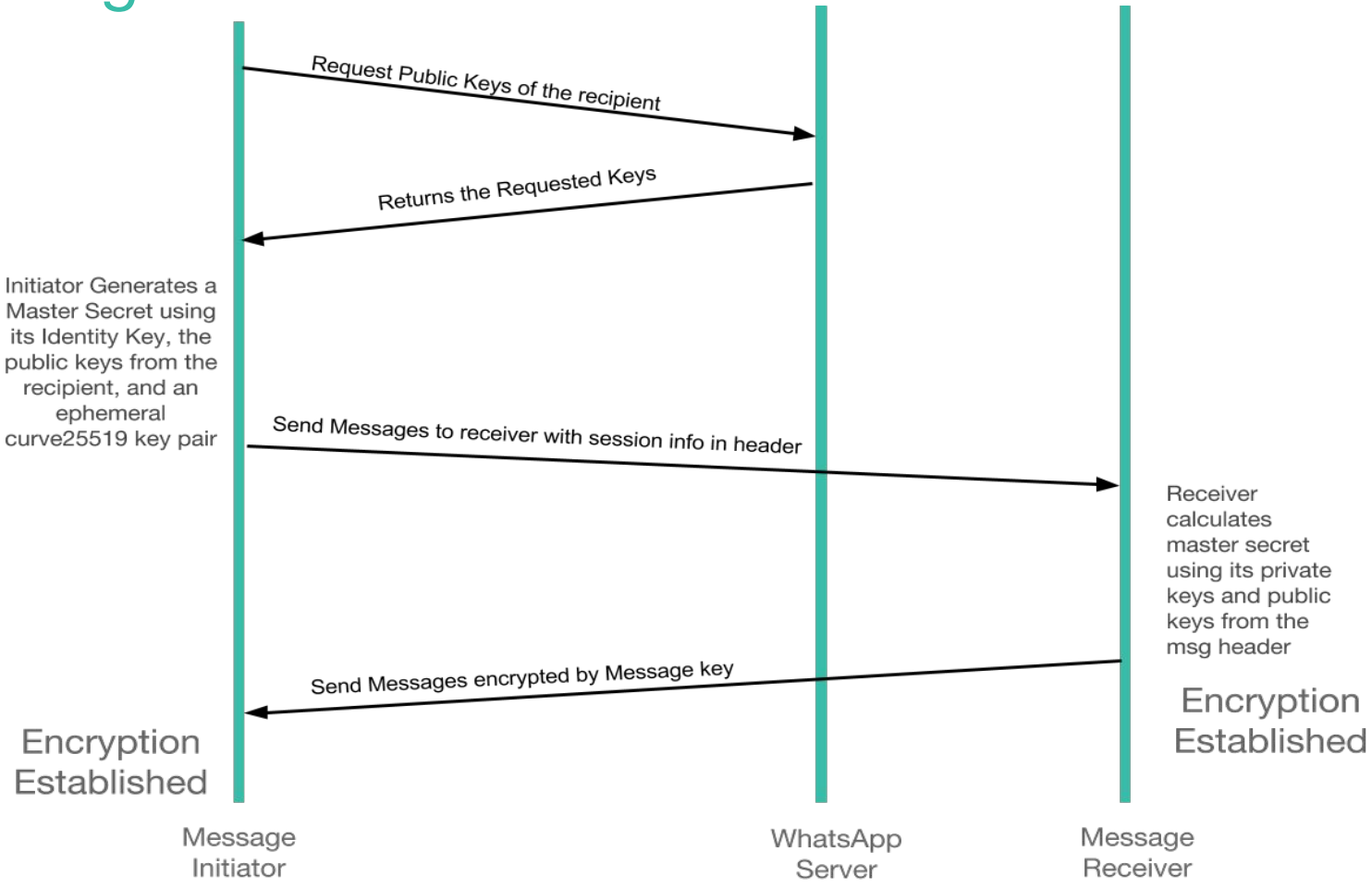
- The initiator requests the receiver's Public Identity key ($I_{recipient}$), Signed Pre-Key ($S_{recipient}$) and One time Pre Key ($O_{recipient}$) from the WhatsApp Server
- Next, the initiator generates an ephemeral Curve25519 key pair, ($E_{initiator}$)
- Initiator generates the master secret as, master_secret =

$$ECDH(I_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, I_{recipient}) \,||$$

$$ECDH(E_{initiator}, S_{recipient}) \,||\, ECDH(E_{initiator}, O_{recipient})$$

- ECDH = Elliptic Curve Diffie-Hellman, $I_{initiator}$ = Initiator's public key

Establishing Session

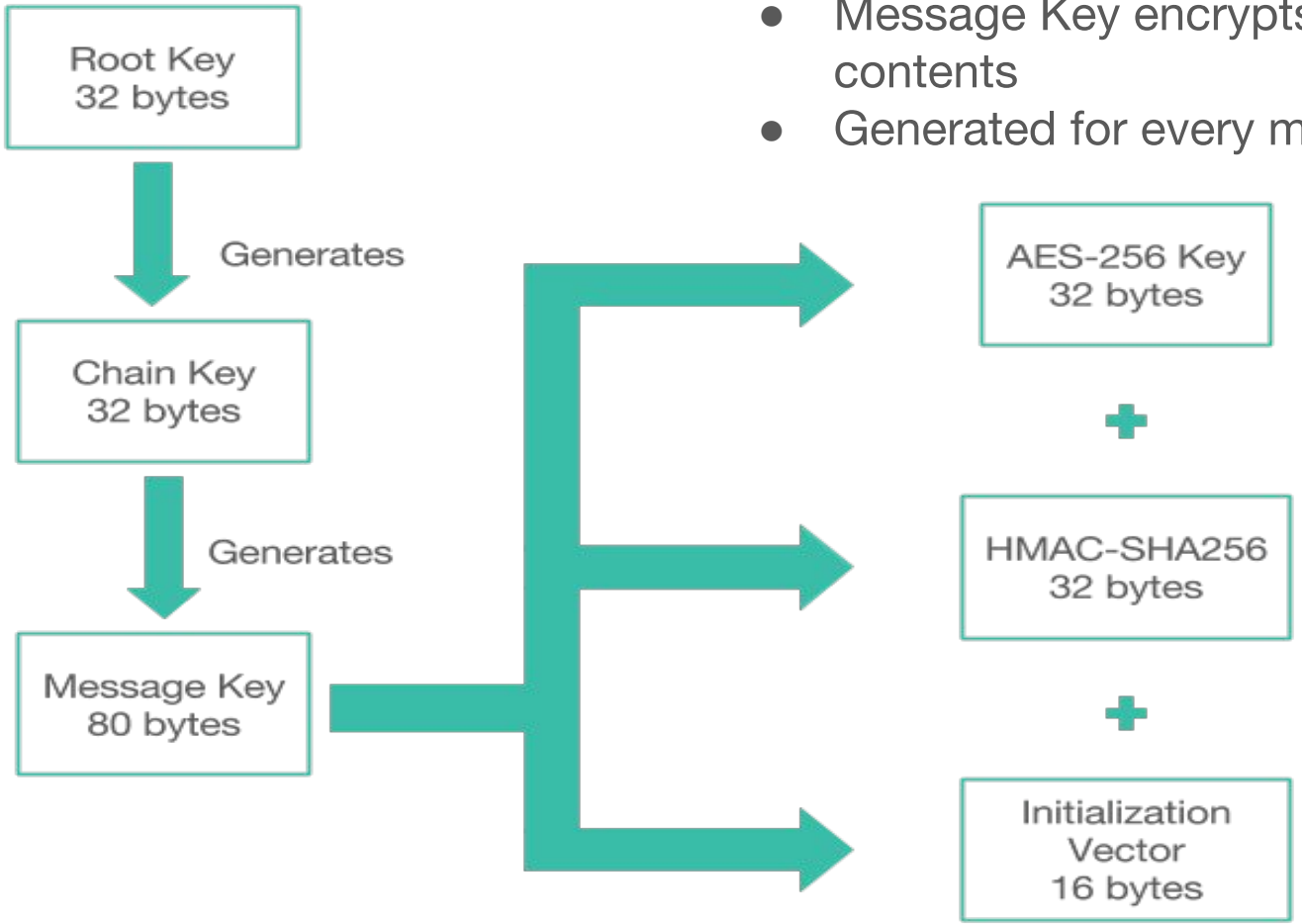


Forward Secrecy

- Root Key and Chain key are derived by the initiator from the master_secret using HKDF
- A new ECDH agreement is performed for every message round trip in order to create a new Chain Key
- The Message key is derived from the sender’s Chain Key
- The Message key changes for every message transmitted and “ratchets” forward with every message sent
- This ensures forward secrecy

Session Key Types

- Message Key encrypts the message contents
- Generated for every message transmitted



Exchanging Messages

- Once encrypted session is established, clients exchange messages encrypted with the Message Key
- Every new message is encrypted by generating a new Message Key
- Message key is of 80 bytes and uses a randomly generated Initialization vector of 16 bytes to encrypt messages using AES256 in Cipher Block Chain Mode, each block is of 32 bytes
- Authentication is Done via 32 bytes HMAC-SHA256, which is also a part of the message key

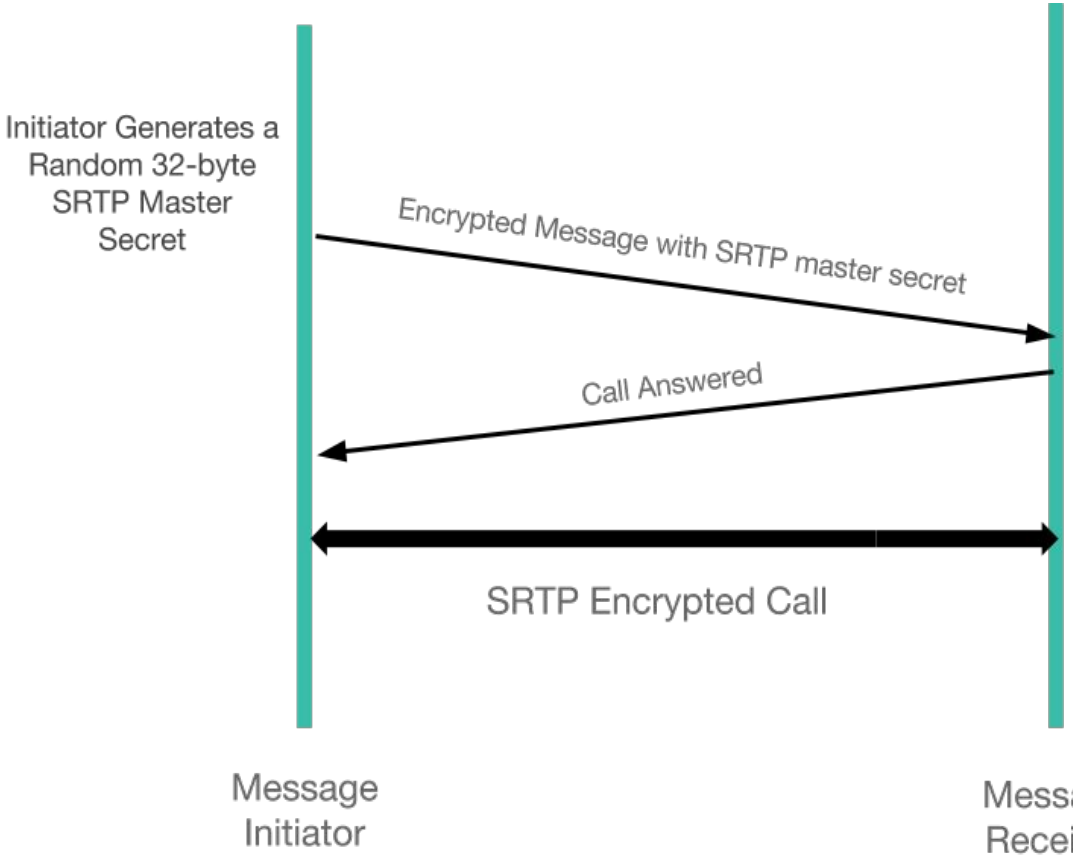
Calculating Chain Key from Root Key

- An ephemeral Curve25519 Public key is advertised with every message transmission
- After a response is received, the new Chain Key and Root key is calculated by the initiator as:
  a. ephemeral_secret = ECDH($Ephemeral_{sender}$,$Ephemeral_{recipient}$)
  b. Chain Key, Root Key = HKDF(Root Key, ephemeral_secret)
- HKDF = HMAC-based Extract-and-Expand Key Derivation Function (RFC 5869)

Calculating Message Key from Root Key

- The message key is generated from the Chain key as-
  - Message Key = HMAC-SHA256(Chain Key, 0x01)
- The Chain Key is then updated as-
  - Chain Key =HMAC-SHA256(Chain Key, 0x02)
- This causes the Chain Key to ratched forward, ensuring “forward secrecy”

Call Setup



- The initiator builds an encrypted session as outlined in the previous slides, if it doesn't exist
- Subsequently, the encrypted call is achieved using SRTP (RFC 3711)

Transport Security

- Communication between WhatsApp clients and is done via Noise Pipes with Curve25519, AES-GCM, and SHA256
- Achieved using the Noise Protocol Framework, see http://noiseprotocol.org/noise.html
- Server stores only the public Curve25516 keys, preventing leakage of the Client's private Authentication keys
- This encrypts metadata, preventing revelation to Eavesdroppers like the NSA

Conclusion

- WhatsApp uses strong end to end encryption based on the signal protocol that employs AES256 in CBC mode, HMAC-SHA256 and Curve25516
- WhatsApp doesn’t store private keys on its servers, which will prevent Government agencies to try and get private keys in order to decrypt messages as happened with the case of Lavabit
- Forward Secrecy is maintained via new ECDH agreements for every message exchanged
- This is a great step forward in the fight to curtain the pervasive monitoring powers of various governments

References

"WhatsApp". *WhatsApp Encryption Overview - Technical white paper.*

Levison, Ladar. "Lavabit". Lavabit.com. N.p., 2016. Web. 13 Apr. 2016.