Double Ratchet Algorithm

Jenish Raj Bajracharya and Ritul Satish

April 13, 2022

Contents

1 Introduction **1**

2 Project goals **1**

3 Technical background **1**
3.1 KDF Chain 1
3.2 Symmetric Key Ratchet 2
3.3 Diffie Hellman Ratchet 2

4 Extended Triple Diffie-Hellman(X3DH) **4**
4.1 Overview of X3DH 4

5 Double Ratchet **5**

6 Implementation specifics **5**

7 Timeline and Deliverable **6**

1 Introduction

The Double Ratchet Algorithm(DRA) is used for secure end-to-end encryption of messages between two users in popular messaging platforms like Whatsapp and Signal. The algorithm provides state of the art security as it generates unique session keys for every message. It guarantees perfect forward secrecy and post compromise security ie the attacker cannot obtain neither, earlier keys from later ones nor later keys from earlier ones.

2 Project goals

We aim to implement a end-to-end encrypted message system using the Double Ratchet Algorithm in python. The application will have a GUI which will allow two users to communicate securely with the guarantees of the DRA. This will also involve setting up a server that will relay the messages.

3 Technical background

3.1 KDF Chain

KDF chain is an important concept of the Double Ratchet algorithm.
A **KDF** is the cryptographic equivalent of a Pseudorandom Function **(PRF)**. The KDF takes

in two inputs, a secret and random key, an input data and returns a output data.

$$\texttt{KDF: } \{\texttt{input\_data}\} \times \{\texttt{KDF\_key}\} \rightarrow \{\texttt{output\_data}\}$$

The term **KDF chain** is used when some of the output from the KDF is used as an output key and some is used to replace the KDF key for the next iteration. The diagram below (Figure: 1) represents the working of a KDF chain.

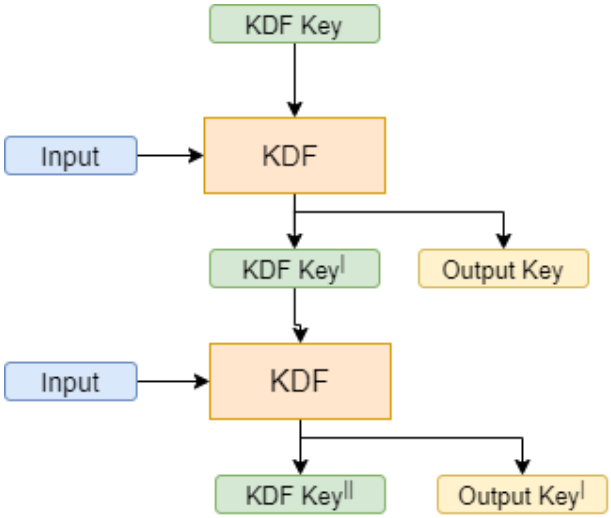

Figure 1: A KDF Chain

Since KDF resembles a PRF, it offers some cryptographic properties. A KDF maintains **resilience** to an adversary who controls the KDF inputs given that the KDF key is unknown. Similarly, KDF provides **forward security** as the output keys from the past appear random to an adversary who gets hold of a KDF key at some point in time. Furthermore, KDF provides **break-in-recovery** as the output data appears random given that the adversary has knowledge of the KDF key and the input.

In a Double Ratchet session between two parties the KDF chain is part of the root chain while the symmetric key ratchet is part of the sending and receiving chain.

3.2 Symmetric Key Ratchet

A Symmetric Key Ratchet is a KDF with the inputs to the KDF constant i.e. a keyed function. The output from this ratchet is a chain key and the message key. The chain key is used for another iteration of the KDF while the message key is used to encrypt the message.

$$\texttt{SKR}_{\texttt{constant}}\texttt{: } \{\texttt{Chain\_key}\} \rightarrow \{\texttt{output\_data}\}$$

The symmetric key ratchet is part of the sending and receiving chains. SKR ensures encryption of messages with a different key. Unlike KDF, these ratchets do not provide **break-in-recovery** as subsequent chain keys can be derived once a chain key is compromised thus revealing all future messages.

3.3 Diffie Hellman Ratchet

Since a symmetric-key ratchet cannot provide break-in-recovery, a Diffie-Hellman ratchet is combined on top to update chain keys.

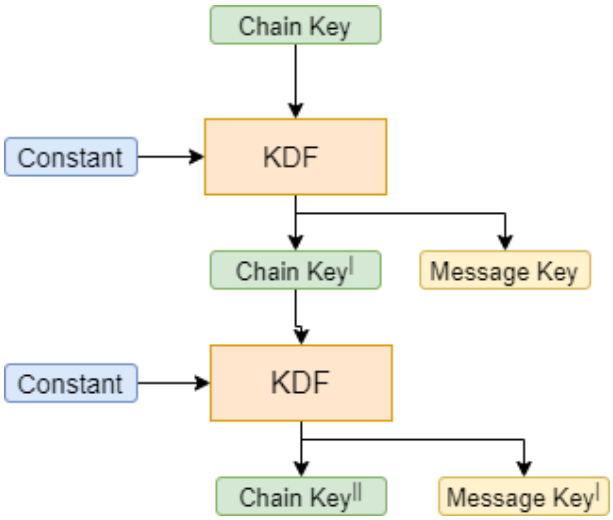

Figure 2: A Symmetric Key Ratchet

Each party generates a DH key pair which becomes the part of the current ratchet key. The header of the encrypted message contains the sender's DH public key. When a receiver receives a new DH public key, a DH ratchet step is performed which replaces the current ratchet key with the new key pair.

Assume Alice wants to communicate with Bob, Alice is first initialized with Bob's public key. Upon initialization, Alice performs DH calculation between her private key and Bob's public key to generate a DH output.

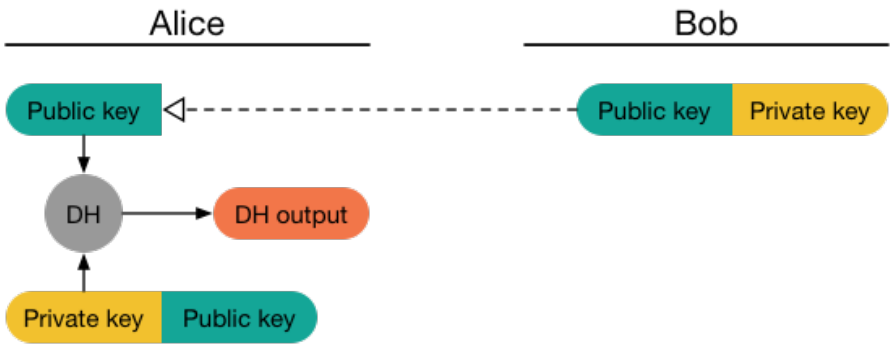

Figure 3: Alice's ratchet configuration

Alice then sends the encrypted message along with her public key to Bob. Once Bob receives one of these messages, Bob performers a DH ratchet step of his own. The DH output Bob get is equal to Alice's initial DH output. Bob then replaces his ratchet key pair and calculates a new DH output.

This results in a 'ping-pong' effects as parties take turns replacing ratchet key pairs. Even if one output is compromised, it is eventually replaced with a new uncompromised key.

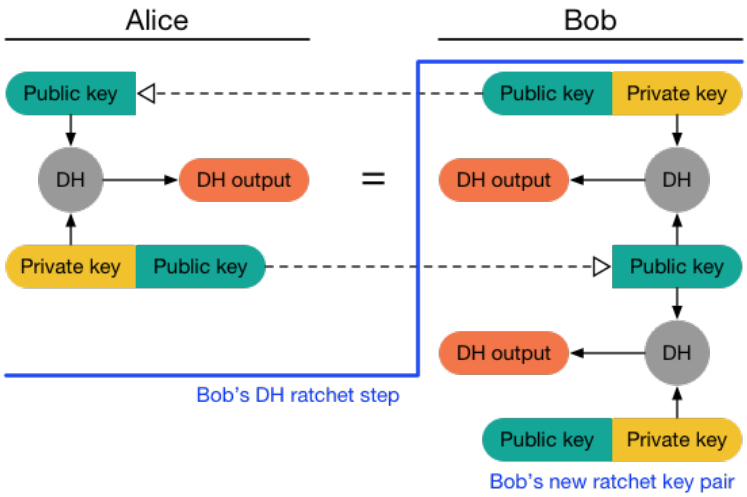

Figure 4: Bob's ratchet configuration

4 Extended Triple Diffie-Hellman(X3DH)

X3DH is a key agreement protocol used to establish a shared secret key (Root Key) between the users Alice and Bob based on their public key. This is done over a server. X3DH is designed for asynchronous settings as well. Assume Bob is offline and Alice wants to send him a message. Here Alice will retrieve information stored on the server by Bob to generate a shared key which will be used for future communication of encrypted data.

4.1 Overview of X3DH

- **Key Publishing:** Bob publishes an identity Key, signed prekey and a set of one time prekeys to the server.
- **Sending Message:** Alice fetches a prekey bundle from the server. this bundle contains- Bob's identity key, signed prekey and a one time pre-key.
  Alice verifies the prekey signature:

```
if verification fails:
   the protocol is aborted
else:
    Alice generates ephermal key (Ekr)
```

  After generating the ephermal key Alice sends Bob an initial message containing
  - His identity key
  - His ephermal key
  - identifier of Bob's prekeys used
  - initial ciphertext encrypted with AEAD.
- **Receiving Messages:** On receiving the first the intial message from Alice, Bob retrieves the Alice's identity key and ephermal key from the message. From this he extracts the

secret key(root key) and then decrypts the initial message using it and the identity keys of both his and Alice's.

5 Double Ratchet

Combining the DH ratchet with the symmetric-key ratchet gives the Double Ratchet. For any party, the double ratchet consist of a DH ratchet, a root chain, a sending chain and a receiving chain. The chains are configured with the symmetric-key ratchet. Receiver's receiving chain matches sender's sending chain and vice versa.

Consider a case when Alice is sending message (A1) to Bob. Using the X3DH both parities are initialized with the same root key. Further, Alice has been initialized with Bob's ratchet public key. Alice then generates a DH key pair of her own and feeds the DH output to the rook chain to calculate the new root key (RK) and the sending chain key (CK). CK acts as an input to Alice's sending chain resulting as a message key and a chain key. Message key will be us used to encrypt the message.

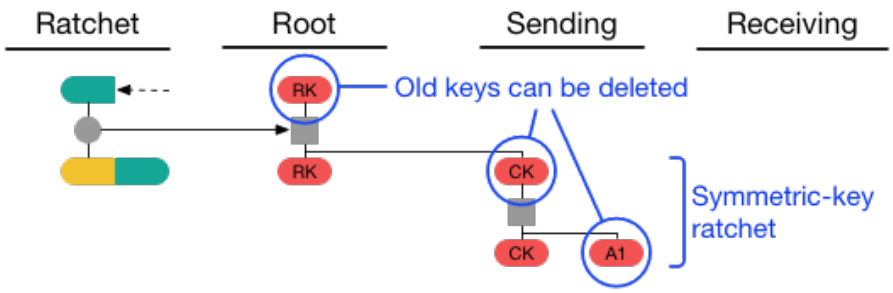

Figure 5: Alice's Double Ratchet

Upon receiving A1, Bob's Double Ratchet configuration will be the same as Alice's but , his receiving chain increases while his sending chain remains empty.
If Alice next receives a response B1 from Bob, a new ratchet configuration will start. Alice initiates DH ratchet to derive new receiving and sending chain keys, then applies a symmetric-key ratchet to the receiving chain to get the message key and thus decrypt the message. When back to back messages are sent at once, the DH ratchet is not updated. Only the symmetric key ratchet is ticked to generate new keys for encryption.

The following diagram (figure:6)shows Alice's Double ratchet configuration when the following message sequence is followed: $A1$, $B1$, $A2$, $B2$, $A3$, $A4$, $B3$, $B4$, $A5$. where A represents Alice sending message to Bob and B represents Alice receiving message from Bob.

6 Implementation specifics

We plan to do the following when implementing the algorithm-

1. Setup a server using a python server to enable communication between the two users.
2. Create a class for Bob and Alice which helps us do key generation and exchanges for the respective user. This will also keep track of the ratchets(root,send,recv) of both the users. This will also have the Diffie-Hellman ratchet.

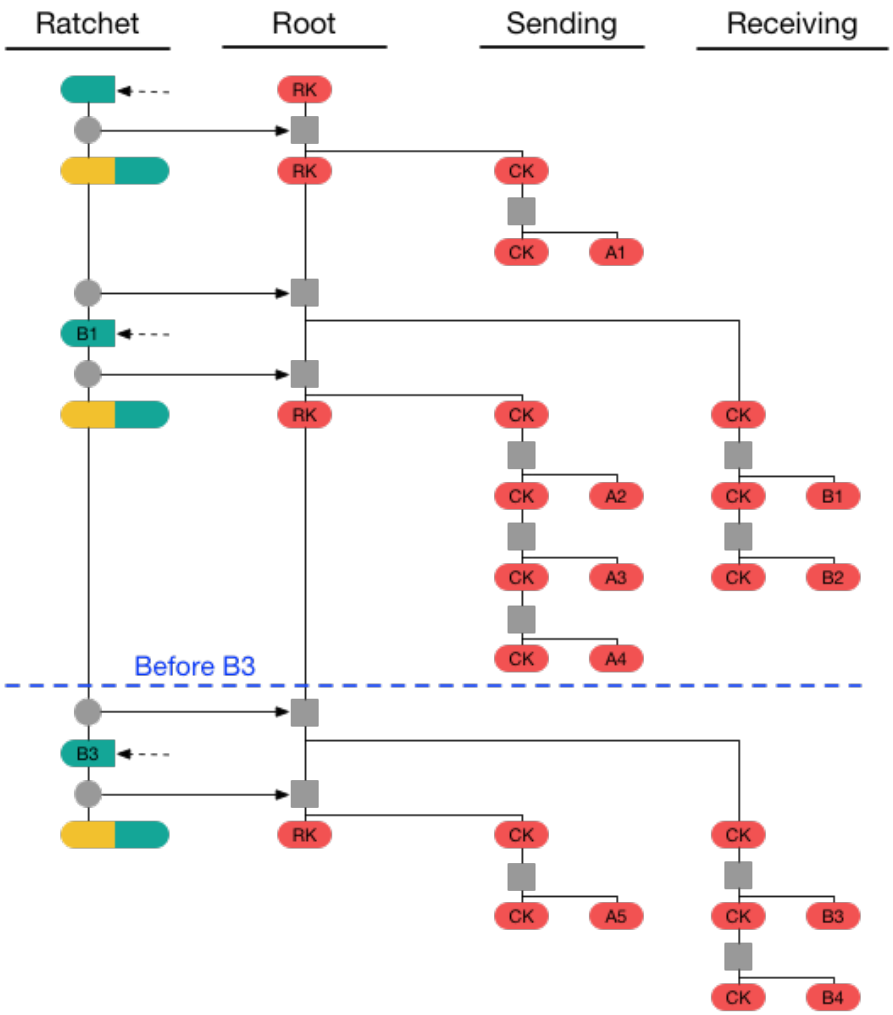

Figure 6: Alice's ratchet

3. We have a class for the Symmetric ratchet implementation which is a component of DRA.

- **Diffie-Hellman:** The generation and exchange of Diffie-Hellman keys is achieved by running the X25519 function used in the cryptography python library.
- **Key Derivation Function:** KDF for the root key is done using the HKDF function in which SHA256 function. KDF for the chain key(sending/receiving) is done using HMAC.
- **Encryption:** We will be using the AEAD encryption.

7 Timeline and Deliverable

Timeline	
Date	Work
April 15	Install Dependencies and start dummy project
April 16 - 18	Set up server and start writing user classes
April 19 - 21	Finish rest of the functions
April 22 - 25	Buffer period for project completion

References

[1] Marlinspike, M. (2016, November 20). The double ratchet algorithm. Signal Messenger. Retrieved April 14, 2022, from https://signal.org/docs/specifications/doubleratchet/

[2] Marlinspike, M. (2016, November 4). The X3DH key agreement protocol - signal. Retrieved April 13, 2022, from https://www.signal.org/docs/specifications/x3dh/x3dh.pdf

***