# The Signal Protocol & The Double Ratchet

Robert Picciotti

# Table of Contents

# Abstract

Open Whisper System's Signal Protocol has become a very commonly used cryptographic protocol. It is in use in multiple apps with millions of users, and some with over a billion. However, the protocol is still very young, having been released less than five years ago. As a result it has not received the same in depth analysis that many other widely used protocols have been scrutinized with [4]. This paper, while not analyzing the security itself, will aim to give the reader an understanding of the protocol and how it functions. In particular this paper will look into the Double Ratchet Protocol, which is used in the Signal Protocol, and is also developed by Open Whisper Systems.

# Introduction

## Open Whisper Systems

The Signal Protocol's origins start with a company called Whisper Systems. In 2010, Whisper Systems released two apps RedPhone and TextSecure. These two apps aimed to provide secure calling and texting respectively and were proprietary in nature. In 2011 the company was acquired by Twitter who initially moved to shut down the services. However, after public outcry the company released the apps under the GPLv3 open source license.

In 2012 the founder of Whisper Systems, Moxie Marlinspike, left Twitter. He founded a new company Open Whisper Systems. This company aimed to continue development of the two apps. In 2014 the company would release an app for iOS called Signal, it supported the RedPhone calling protocol. Later the TextSecure Protocol would be ported to iOS and merged into the Signal app as well. Around the same time the RedPhone and TextSecure apps for Android were merged into a single app also called Signal. To complete the rebranding the TextSecure Protocol was rebranded as the SIgnal Protocol.

## The Signal Protocol

The Signal Protocol is developed by Open Whisper Systems. The project aims to create a secure messaging protocol for use in the modern world. The protocol is used in their own app, Signal, along with WhatsApp, Facebook Messenger, and Google Allo. The Signal app and WhatsApp both use the protocol for all messages, whereas Facebook Messenger and Google Allo only use it for "secret messages." Both WhatsApp and Facebook Messenger claim to have over a billion users. WhatsApp is especially significant since all messages are encrypted with the protocol. [4]

The protocol was originally released in 2013 under the name of the TextSecure Protocol. It was later renamed to the Signal Protocol. After its original release an update was made in 2014. The second version brought in several new features and began the use of the Axolotl ratchet. The Axolotl ratchet would late be renamed the Double ratchet and is a key focus of this paper.

## The Double Ratchet

The Double Ratchet is the key to the Signal Protocol's key exchange mechanism. This paper will go into depth on the matter, but a brief description will be given here. In short, the Double Ratchet aims to expand on the Diffie Hellman ratched used in Off-The-Record (OTR) messaging. This protocol had aimed to allow for secure key exchange in instant messaging conversations.

The Double Ratchet aims to go further than OTR. Open Whisper Systems identified the long term nature of modern text conversations as a significant weakness. [3] The compromise of a key could reveal years of conversation. This contrasts with the generally short sessions that existed for old Instant Messaging platforms. This has resulted in Open Whisper Systems creating a protocol that rapidly moves through "ephemeral" keys, short lived keys that are kept for only as long as needed. [3]

This results in a property that Open Whisper Systems call "self-healing." [3] The compromise of any ephemeral key will only compromise a short part of the conversation. All parts of the conversation before and after the compromised key was used will remain secure.

# Features of the Double Ratchet

The Double Ratchet aims to provide various properties to the key exchange aspect of communication. The most notable of the features are: forward secrecy, future secrecy, asynchronicity, and message unlinkability.

## Forward Secrecy

One of the major features the Double Ratchet aims to provide is protection from a future compromise. If your current situation is secure, then a later message being compromised should have no effect on the current message. This is achieved through the use of "ephemeral" keys. [2]

## Future Secrecy

Future secrecy is, to an extent, the opposite of forward secrecy. Whereas forward secrecy protects the current message from the compromise of a future one, future secrecy protects a future message from a compromise of the current message.

Open Whisper Systems has dubbed this property "self-healing." If a third party compromises the current keys, then the replacement of keys over time will make it impossible for the third party to continue to listen in on the conversation. [3]

## Asynchronicity

The Signal Protocol and the Double Ratchet are primarily designed for use with phones and similar platforms. Phones are subject to frequent disconnection, from loss of signal, power, or other means. As a result it is not guaranteed that both parties will be available at any one moment. The Double Ratchet aims to allow key exchanges even if both parties are not present, messages are dropped, or received out of order. [3]

## Message Unlinkability

Continuing in the vein of the above properties, message unlinkability aims to remove the connection between different messages. For example, it should not be possible to tell if multiple messages were part of the same conversation.

# Primitives

The Signal Protocol makes use of several primitives, a number of which are used within the double ratchet. While the protocol allows for a fair amount of flexibility, this paper will only comment on the recommended defaults used for Open Whisper Systems own implementation.

## The Diffie-Hellman Ratchet

One of the two ratchets used in the Double Ratchet is based on Diffie-Hellman. Specifically it uses X3DH, a modification made of Diffie-Hellman by Open Whisper Systems. Elliptic Curve cryptography is used in the implementation, with Curve25519 being the recommended default curve. Curve25519 offers 128 bits of security, has good security properties for being a Diffie-Hellman curve, and runs particularly quickly. [2]

## Symmetric Key

The symmetric key used for bulk encryption of data is AES. Both Cipher Block Chaining mode and Counter mode are used in the implementation. AES is not used within the Double Ratchet itself, but as part of the SIgnal Protocol. [2]

## Message Authentication Modes

HMAC is used, with the hash function being SHA-2 with 256 bits of output. [2]

# X3DH

## Purpose

X3DH aims to provide an implementation of Diffie-Hellman that allows for a key exchange where one party is not available. The Signal Protocol requires frequent updates to the key in use, so it is important to make key exchange possible in adverse conditions.

To make this possible X3DH brings in a third party, a “server.” The server can be a single entity or split across multiple actual computers. The protocol aims to minimize the threat the server can pose. [2]

## Implementation

If Alice wants to send a message to Bob then Bob must first place his public identity key, along with a signed pre-key onto the server. Alice then collects these from the server. The identity key is used to verify that the pre-key was correctly signed. The pre-key is then used to construct the current session key.

In this setup the pre-key is replaced on a regular basis. The implementation allows for the server to store multiple future keys at once. Though eventually its supply will need to be replenished.

If no pre-keys are available then the protocol has a fallback that allows for a key exchange regardless. However, without a pre-key the protocol becomes vulnerable to a replay attack.

Creating the key involves four Diffie-Hellman operations (three without a pre-key). They are:

$$\begin{aligned} DH1 &= DH(IK_A, SPK_B) \\ DH2 &= DH(EK_A, IK_B) \\ DH3 &= DH(EK_A, SPK_B) \\ DH4 &= DH(EK_A, OPK_B) \end{aligned}$$

With the above values being:

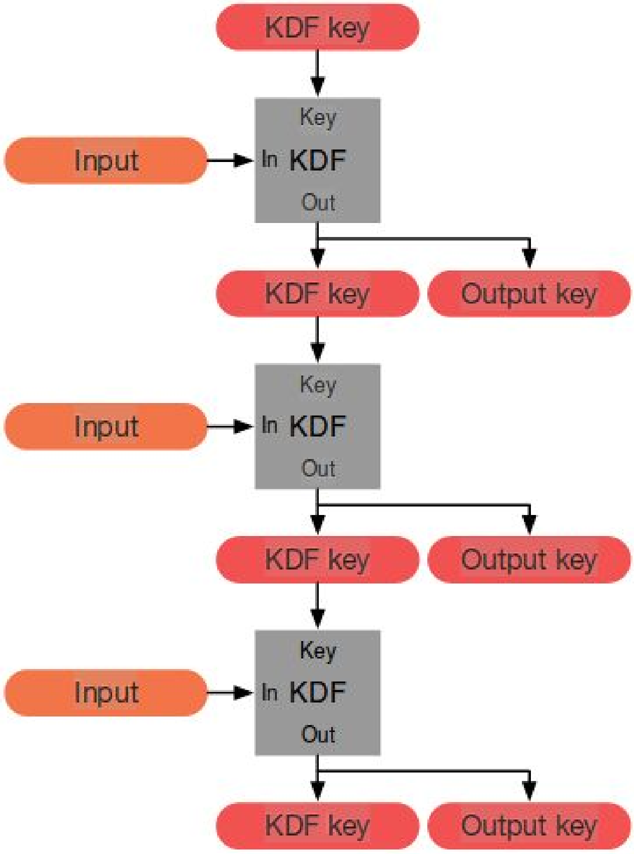
$IK_B$ = Bob's identity key
$SPK_B$ = Bob's signed prekey
$Sig(IK_B, Encode(SPK_B))$ = Bob's pre-key signature
$OPK_B$ = Bob's one-time prekey

The secret key then exists as these four keys concatenated together. DH4 is skipped if the pre-key is not provided. [2]

# The Double Ratchet

## KDF Chains

The ratchets are based around the idea of KDF (Key Derivation Function) chains. The basis for the chain is a hash function, or similar. It needs to be able to accept two inputs and the output must be a fixed length. The outputs are then chained together, along with some input value. So, the previous KDF output is concatenated (or otherwise combined) with an input value. The output is then split into two parts. The first is the KDF key to be used as part of the input for the next round of the chain. The second portion is taken as the output key of that round. The output can then be used as the basis of further cryptographic functions. [1]



An example KDF Chain [1]

The Double Ratchet Algorithm uses three chains: the Root Chain, the Sending Chain, and the Receiving Chain. The Root chain takes the output of a Diffie-Hellman function as its input. The other two chains both use the output of the Root Chain as their inputs. The output is then used to encrypt messages.

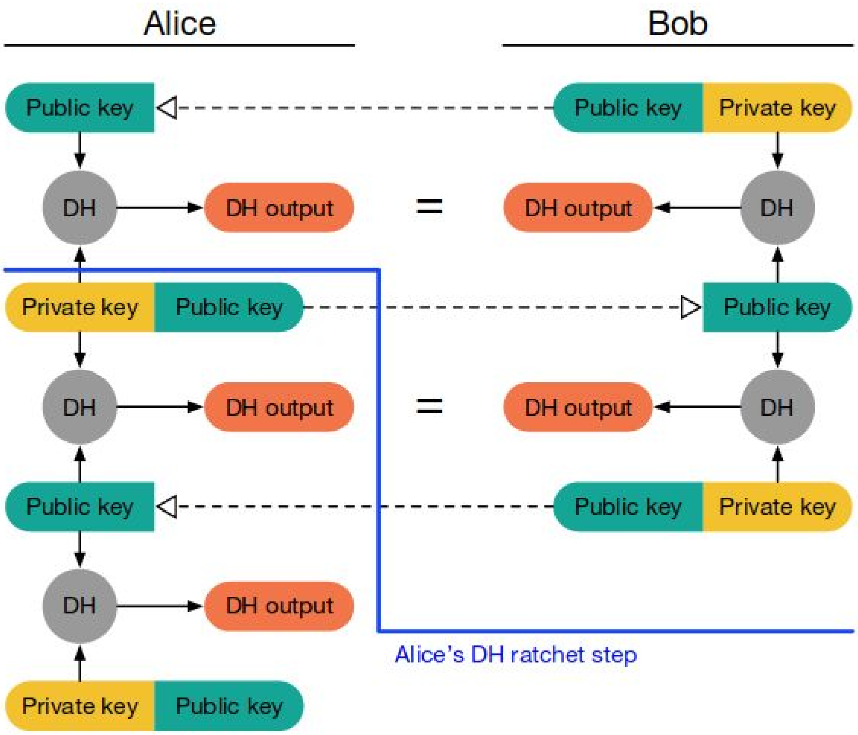
## A Double Ratchet

There are, of course, two ratchets used in the Double Ratchet algorithm. The first is a Diffie-Hellman ratchet and the second is a symmetric ratchet.

### The Diffie-Hellman Ratchet

The outputs of this ratchet are treated as the input of the root chain. The output is created by running Diffie-Hellman on one's own private key and the other party's public key. If both parties do this then they will end up with the same Diffie-Hellman output. This is then fed into the Root Chain. Doing this provides for break in protection.

After a time one party will update the key pair that they are using. They announce to the other party the new public key. The other party then creates a new Diffie-Hellman output by using their current private key and the new public key. Both parties announce what step in the ratchet they are at with every message they send.

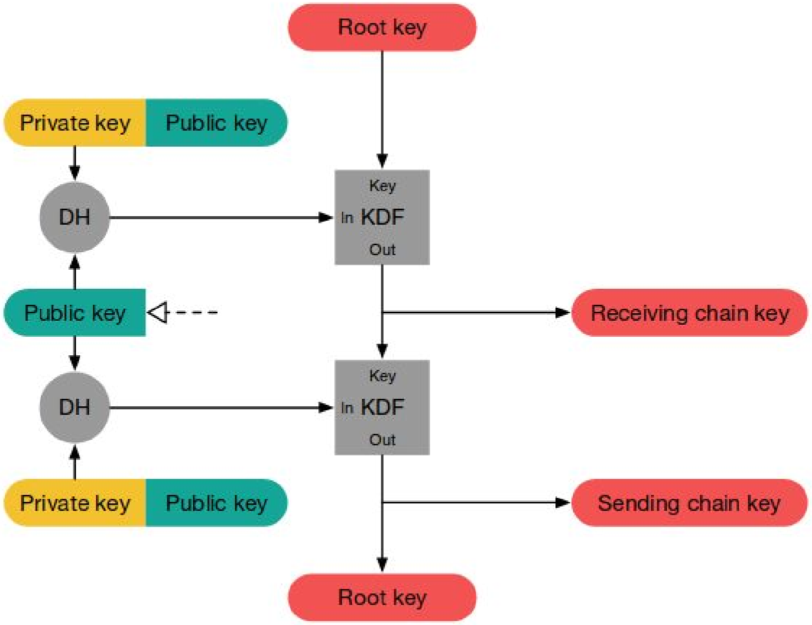


The Diffie-Hellman Ratchet [1]

### The Symmetric Ratchet

At this point the three previously described chains begin to play their part. After each ratchet of the Diffie-Hellman ratched you get a new Diffie-Hellman output. This output is then used as the input of the root chain. So, the root chain updates once every Diffie-Hellman ratchet step. [1]

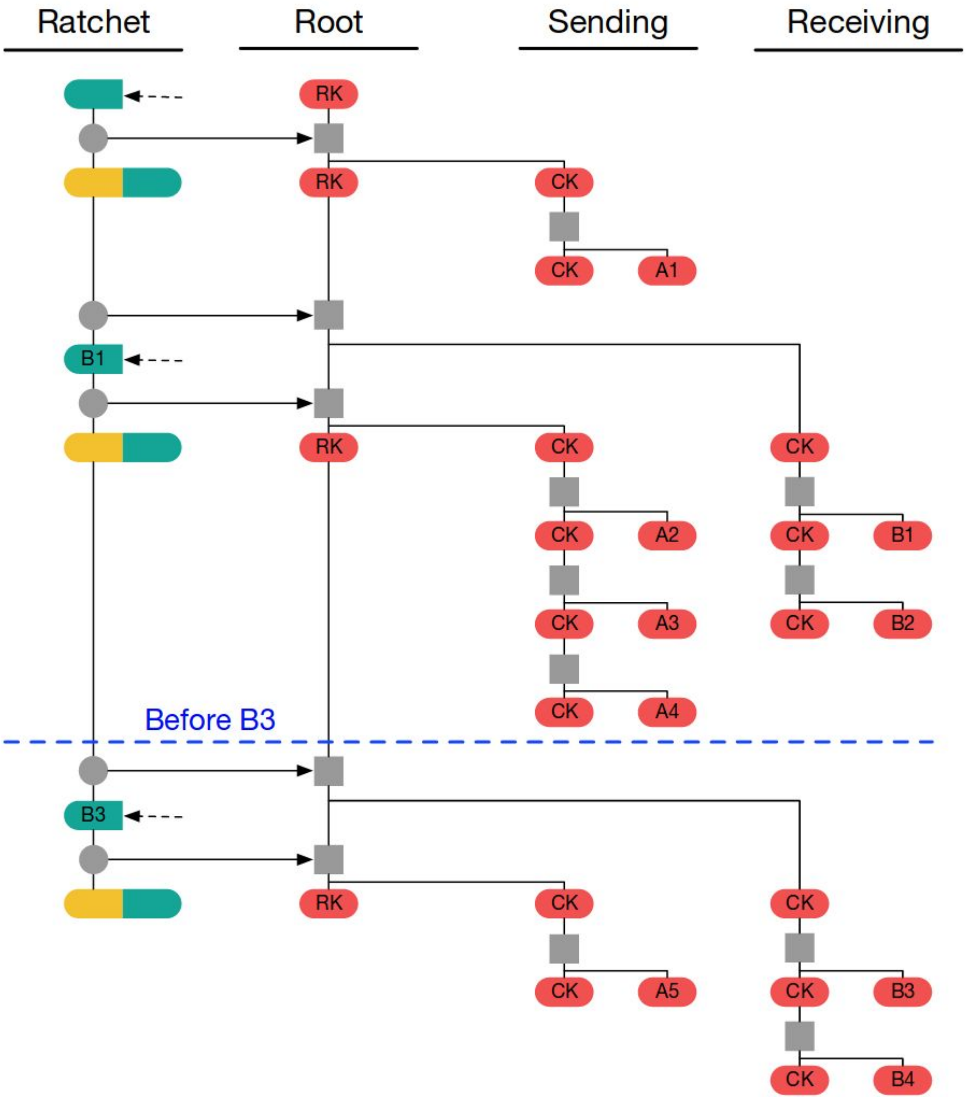
The output of the root chain is then split into two pieces. The first part is used as the key of the next step of the KDF chain. The other part is a secret output which will be used as the input of either the receiving chain or the sending chain. Which chain receives the output is alternated with every step. The receiving chain of one party acts as a complement to the other's sending chain. So, if Alice sends a message encrypted with the output of her sending chain, Bob can decrypt it using the output of his receiving chain.



Incorporating the KDFs into the DH Ratchet [1]

It would, of course, be possible to implement this idea without the root chain. Instead, the Diffie-Hellman output could be directly inputted into each of the chains. However, the use of the root chain aims to provide increased break in protection.

The outputs of the sending and receiving chains are used as the key of a symmetric encryption scheme. In Signal, AES is used. Because the sending and receiving chains of each party complement each other Alice's receiving chain output will exactly match Bob's sending chain output. [1]



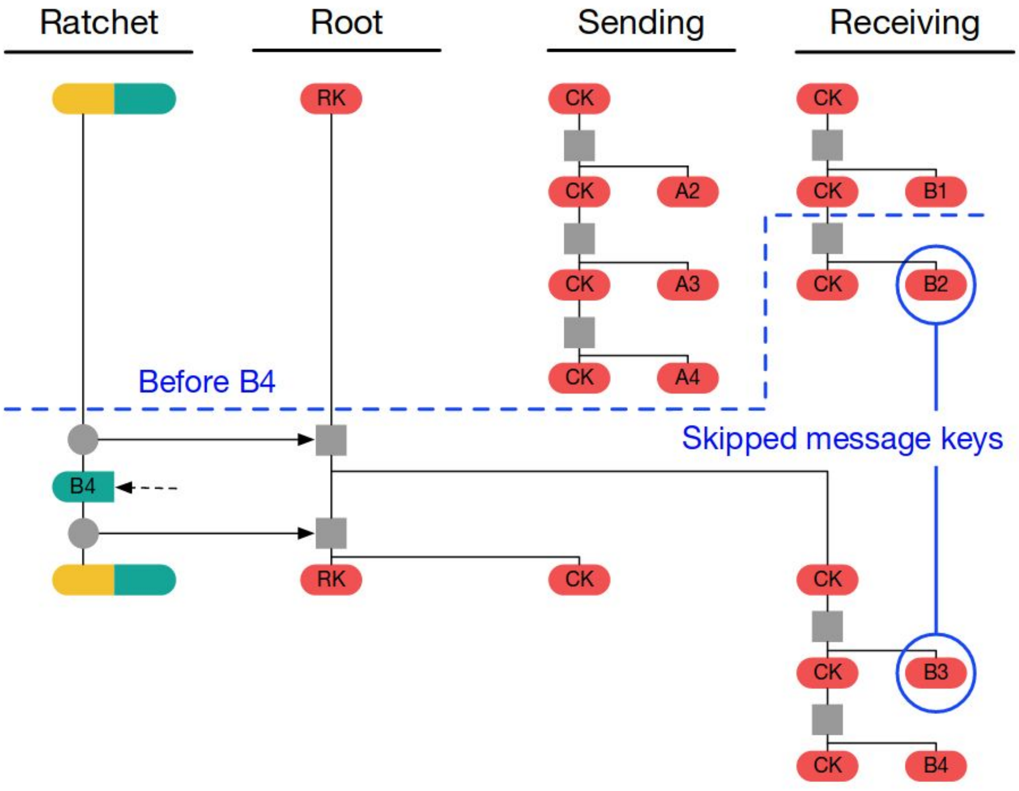
The Full Double Ratchet Setup [1]

## Ratcheting

The sending and receiving chains are updated with every message. So, if Alice sends a message she will immediately ratchet the sending chain. At this point she can delete the key she just used. It is ephemeral and will not be used again. The sent message will include as a header which step of the ratchet was used.

Once Bob has received a message he will check to see which step in the ratchet it was sent at. He will then advance the receiving ratchet to that key and decrypt the message. Once a message is received the key can be deleted.

If Bob receives a message out of order, say he was at step seven in the ratchet and receives a message marked as being from step ten in the ratchet, then he will advance the ratchet to step ten. While he does that he will store the keys that he skipped over. Once he receives those messages he will use the stored key and then delete it. If the message is not received after a reasonable period of time then the key is deleted regardless. [2]



The Double Ratchet with Skipped Messages [1]

# References

[1] Marlinspike, Moxie. “The Double Ratchet Algorithm.” Edited by Trevor Perrin, whispersystems.org/. Accessed 18 Apr. 2017.

[2] Marlinspike, Moxie. "The X3DH Key Agreement Protocol." *Open Whisper Systems* >> *Specifications* >> *The X3DH Key Agreement Protocol*. Ed. Trevor Perrin. Open Whisper Systems, 04 Nov. 2016. Web. 15 Apr 2017.

[3] Marlinspike, Moxie. "Advanced cryptographic ratcheting." *Open Whisper Systems* >> *Home*. Open Whisper Systems, n.d. Web. 17 Apr 2017.

[4] Cohn-Gordon, Katriel, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. "A Formal Security Analysis of the Signal Messaging Protocol." International Association for Cryptologic Research, Oct. 2016. Web. 19 Apr. 2017.