P-4

Group: 7, Members: Peter Gifford, Kyle Brekke, Ren Wall, Madison Hanson

Due: November 18, 2019

1 Algorithm Description - Double Ratchet Algorithm

In communications sometimes it is useful to have that conversation be encrypted, whether it is employees communicating with each other about sensitive information, a whistleblower providing information about something, or people who are concerned about protecting their privacy. The double ratchet algorithm is an algorithm that is used by the encrypted messaging platform Signal and is the focus of our project. The double ratchet algorithm creates a secure way for two people to exchange messages through the use of a shared secret key. With many encryption algorithms once the key is discovered the encryption is compromised because a third party can decrypt all the messages being sent. The double ratchet algorithm solves this problem by consistently generating new keys that cannot be derived from prior keys.

Every message generates a new one of these keys which can be used to decrypt the message meaning that having any single key will only give access to a singular message and cannot be used to retrieve any of the other keys making this more secure than algorithms that just use one secret key throughout the conversation. the algorithm achieves this through the use of KDF chains. These are sets of KDF keys that can be applied to a set of input data to unscramble it. These provide a lot of extra security compared to standard encryption algorithms because at each step there are extra measures taken to ensure that there are no lapses in security. A key part of this is the replacement of keys over time. This is done using a combination of the the Diffie-Hellman ratchet to update the keys with a symmetric-key ratchet. Each message sent contains a header with the sender's current key. On most messages a new key will be generated and sent with the header. Using this new key being sent the receiver can use the DH ratchet step to generate the new private key to decrypt the message. Using these steps new private keys can keep being generated and using the DH step can securely decrypt the messages while also being able to send a public key to a different user. These steps are a big improvement in keeping out an attacker even if they discover the key once which adds extra security to just using the DH step with a standard key to decrypt messages.
The double ratchet is also used to double down on certain aspects of network reliability. Messages could often be received in the incorrect order. Similar to the way that networks assign packets numbers to ensure that they are received in the correct order, the double ratchet attaches an order number to each message to make sure that the re-establishment of a private key is not started when a new public key is received out of order.

Over all the double ratchet is a strong way of keeping an invasive third party from fully compromising a secure message thread. With time people get better and better at breaking through digital security. This means that developers need to create better ways to keep people out. While the double ratchet does not

add security directly related to the access of keys beyond the already established Diffie-Hellman ratchet, the constant change of keys makes any breach reveal very little information because of the constantly changing key.

2 Improvements

3 Psuedocode

---

procedure INITIALIZE_USER($fullKey, sendHeader, receiverHeader$)
    $self.fullKey \leftarrow fullKey$
    $self.sendHeader \leftarrow sendHeader$ ▷ Used to encrypt the headers to add security
    $self.receiverHeader \leftarrow receiverHeader$
end procedure
procedure INITIALIZE_USER_2($startPublicKey, secret, sendHeader, receiverHeader$)
    $self.fullKey \leftarrow getFullKey(startPublicKey, secret)$ ▷ Method exists elsewhere that takes the private key and public key and generates full DH key
    $self.publicKey \leftarrow startPublicKey$
    $self.privateKey \leftarrow secret$
    $self.sendHeader \leftarrow sendHeader$
    $self.receiverHeader \leftarrow receiverHeader$
end procedure
procedure SEND_MESSAGE($recipient, message, publicKey, messageNumber$)
    message ← generateEncryptedMesage(message, publicKey, messageNumber, sendHeader) ▷ Another method assumes to exist that does the actual encryption part of the DH ratcher
    sendMessage(recipient, message)
end procedure
procedure RECEIVE_MESSAGE(messageEncrypted)
    message, header, newKey, messageNumber← extractMessage(receiverHeader, messageEncrypted)
    message ← decryptMessage(message, self.fullKey) ▷ Another method that exists elsewhere to decrypt using the given key
    self.fullKey ← ADJUST_KEYS(newKey)
end procedure
procedure ADJUST_KEYS($newKey$)
    self.fullKey ← generateFullKey(self.publicKey, newKey)
end procedure

---

4 Citation

"The Double Ratchet Algorithm." RSS, https://signal.org/dogs/specifications/doubleratchet/