P-3

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

Due: November 11, 2019

1 Algorithm Description - Double Ratchet Algorithm

The double ratchet algorithm is creating a very 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. 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 other keys making this extremely secure.

---

procedure INITONE($state, SK, two_dh_public_key$)
    state.DHs $\leftarrow GENERATE_DH()$
    state.DHr $\leftarrow two_dh_public_key$
    state.RK, state.CKs $\leftarrow KDF_RK(SK, DH(state.DHs, state.DHr))$
    state.CKr $\leftarrow$ None
    state.Ns $\leftarrow 0$
    state.Nr $\leftarrow 0$
    state.PN $\leftarrow 0$
    state.MKSKIPPED $\leftarrow \{\}$
end procedure
procedure INITTWO($state, SK, two_dh_public_key$)
    state.DHs $\leftarrow bob_dh_key_pair$
    state.DHr $\leftarrow$ None
    state.RK $\leftarrow$ SK
    state.CKs $\leftarrow$ None
    state.CKr $\leftarrow$ None
    state.Ns $\leftarrow 0$
    state.Nr $\leftarrow 0$
    state.PN $\leftarrow 0$
    state.MKSKIPPED $\leftarrow \{\}$
end procedure

---

procedure RATCHETENCRYPT($state, plaintext, AD$)
    state.CKs, mk $\leftarrow KDF\_CK(state.CKs)$
    header $\leftarrow HEADER(state.DHs, state.PN, state.Ns)$
    state.Ns $\leftarrow 1$
    return header, $ENCRYPT(mk, plaintextmCONCAT(AD, header))$
end procedure
procedure RATCHETDECRYPT($state, header, ciphertext, AD$)
    plaintext $\leftarrow TrySkippedMessageKeys(state, header, ciphertext, AD)$
    if $plaintext! = None$ **then**
        return plaintext
    end if
    if $header.dh! = state.DHr$ **then**
        $SkipMessageKeys(state, header.pn)$
        $DHRatchet(state, header)$
    end if
    $SkipMessageKeys(state, header.n)$
    $state.CKr, mk = KDF\_CK(state.CKr)$
    $state.Nr+ = 1$
    $returnDECRYPT(mk, ciphertext, CONCAT(AD, header))$
end procedure
procedure TRYSKIPPEDMESSAGEKEYS($state, header, ciphertext, AD$)
    if $(header.dh, header.n)instate.MKSKIPPED$ **then**
        $mk = state.MKSKIPPED[header.dh, header.n]$
        $delstate.MKSKIPPED[header.dh, header.n]$
        $returnDECRYPT(mk, ciphertext, CONCAT(AD, header))$
    else
        return None
    end if
end procedure
procedure SKIPMESSAGEKEYS($state, until$)
    if $state.Nr + MAX_SKIP < until$ **then**
        $raiseError()$
    end if
    if $state.CKr! = None$ **then**
        while $state.Nr < until$ **do**
            $state.CKr, mk \leftarrow KDF\_CK(state.CKr)$
            $state.MKSKIPPED[state.DHr, state.Nr] \leftarrow mk$
        end while
    end if
end procedure
procedure DHRATCHET($state, header$)
    $state.PN = state.Ns$
    $state.Ns \leftarrow 0$
    $state.Nr \leftarrow 0$
    $state.DHr \leftarrow header.dh$
    $state.RK, state.CKr = KDF\_RK(state.RK, DH(state.DHs, state.DHr))$
    $state.DHs = GENERATE_DH()$
    $state.RK, state.CKs = KDF\_RK(state.RK, DH(state.DHs, state.DHr))$
end procedure