black hat USA 2019

AUGUST 3-8, 2019
MANDALAY BAY / LAS VEGAS

cp<r>
CHECK POINT RESEARCH

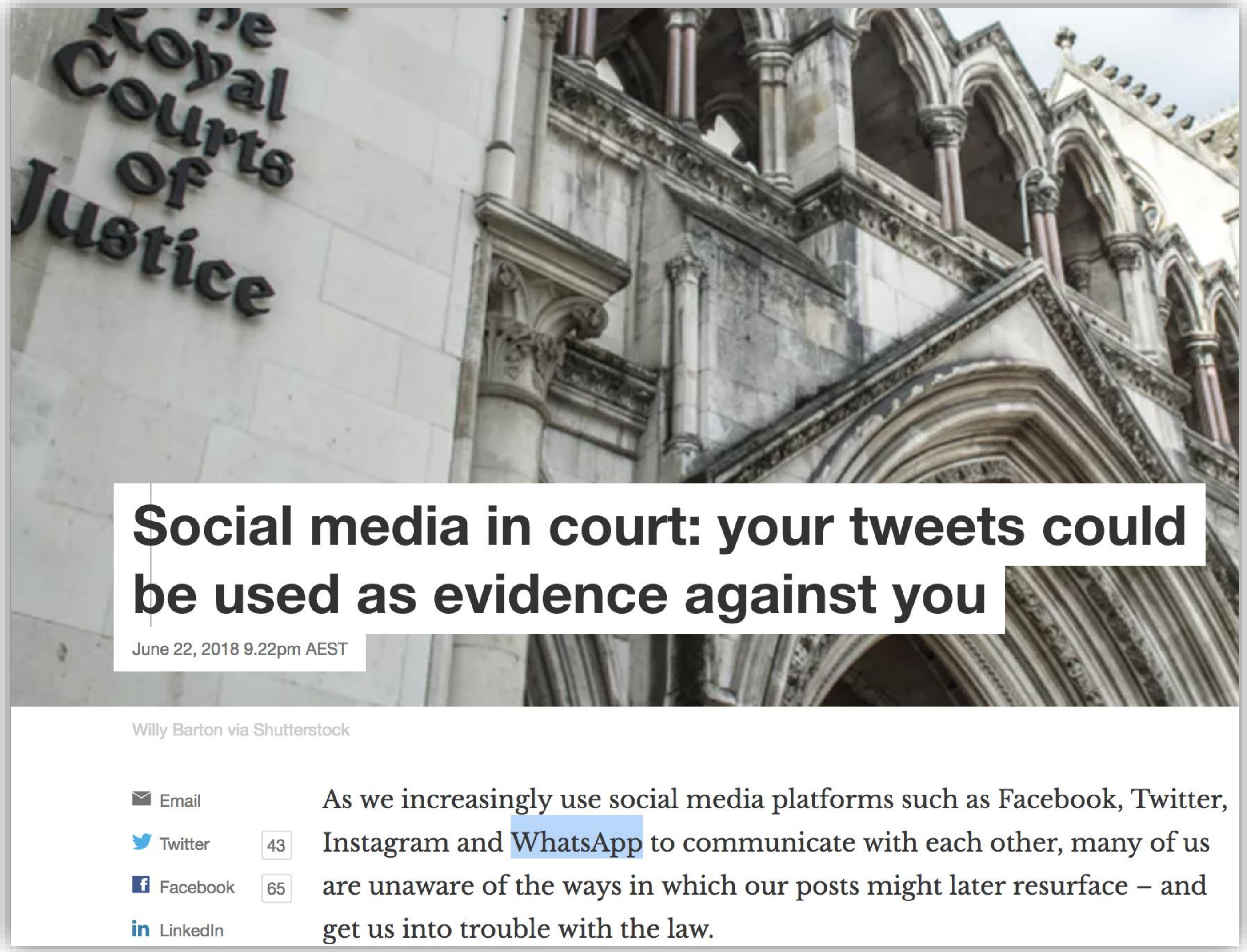
REVERSE ENGINEERING WHATSAPP ENCRYPTION FOR CHAT MANIPULATION AND MORE

Roman Zaikin – Security Researcher

Oded Vanunu – Head of Products and Vulnerability Research

financialexpress.com

WhatsApp as evidence in court

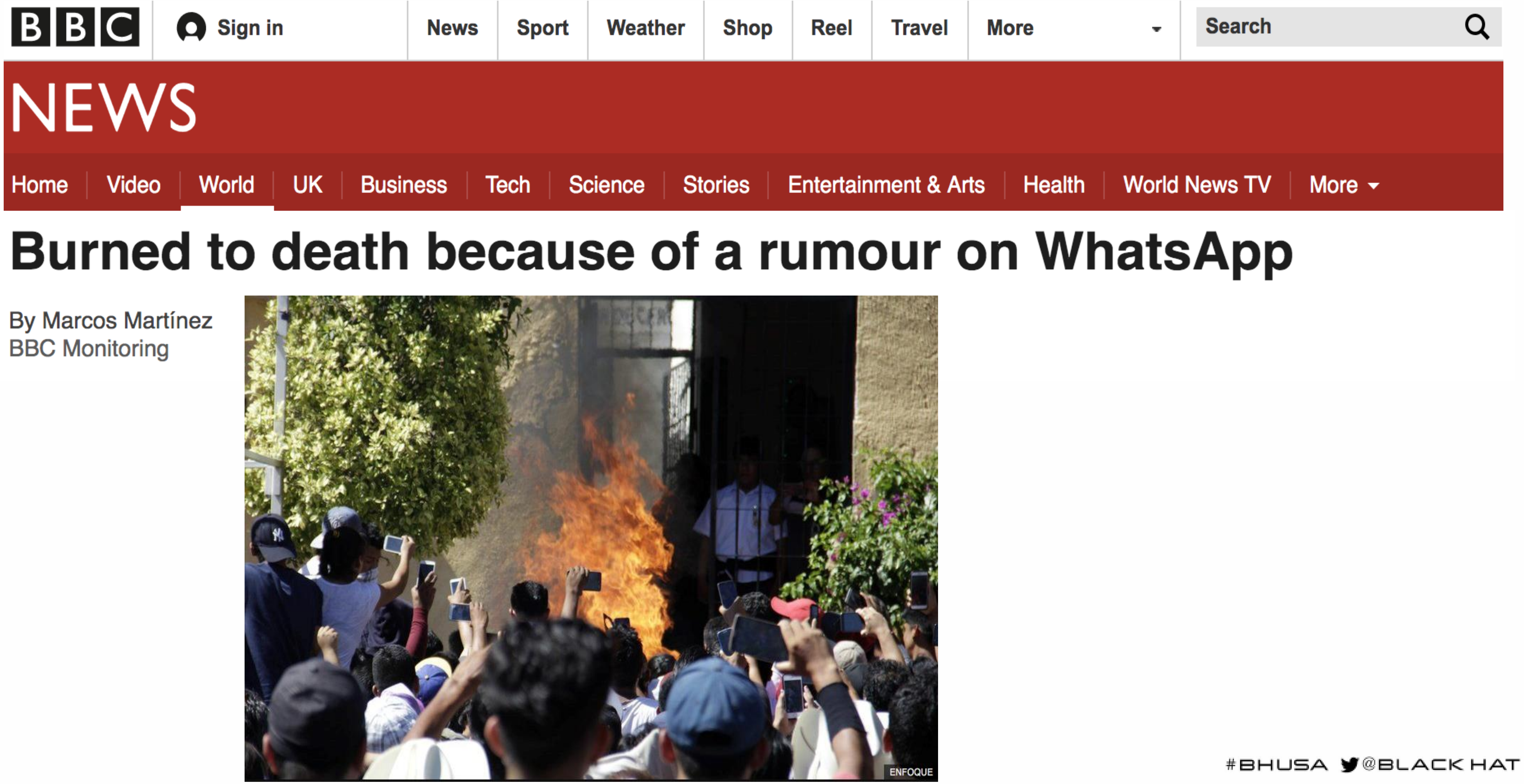


Social media in court: your tweets could be used as evidence against you

June 22, 2018 9.22pm AEST

Willy Barton via Shutterstock

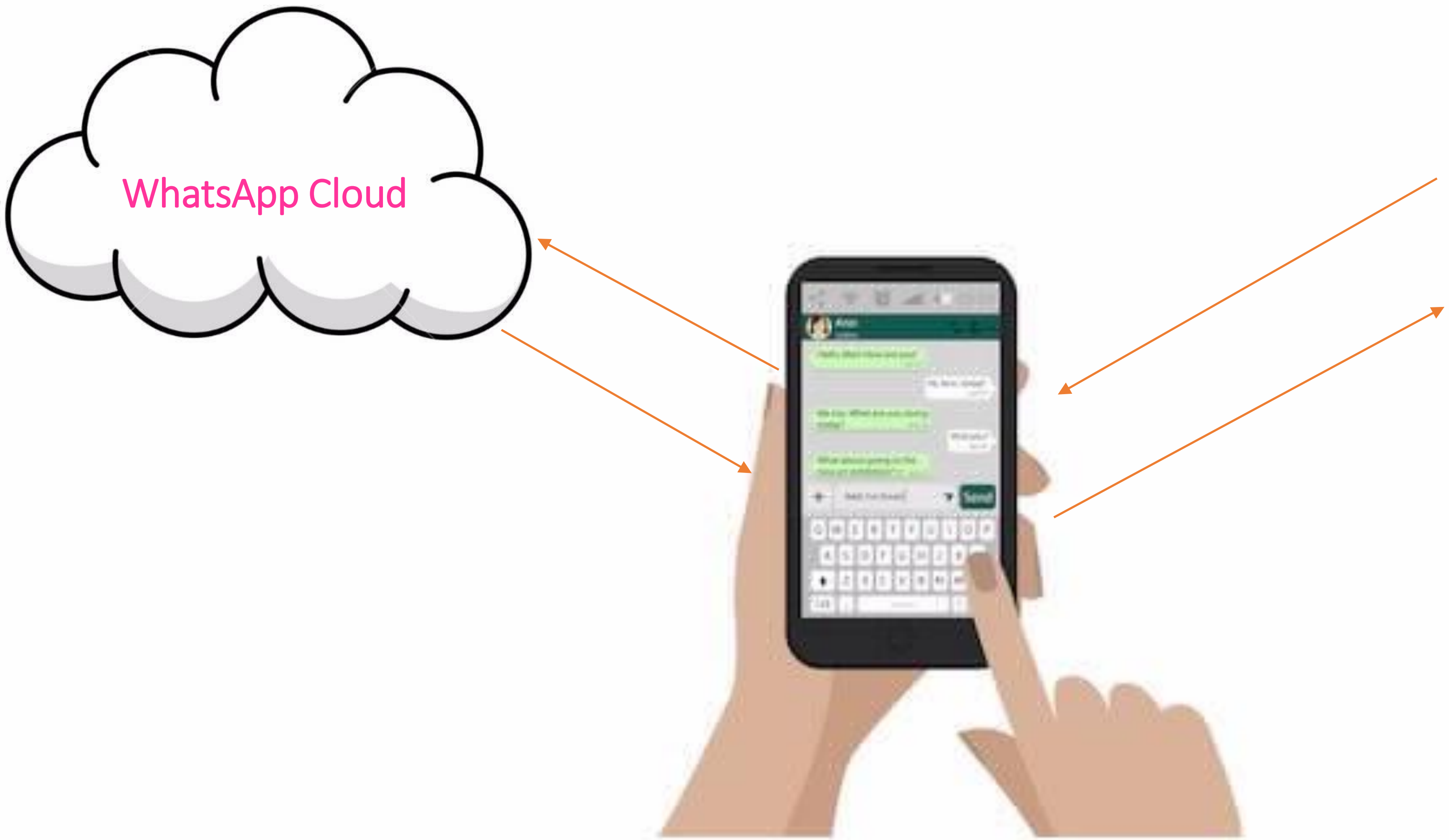
As we increasingly use social media platforms such as Facebook, Twitter, Instagram and WhatsApp to communicate with each other, many of us are unaware of the ways in which our posts might later resurface – and get us into trouble with the law.

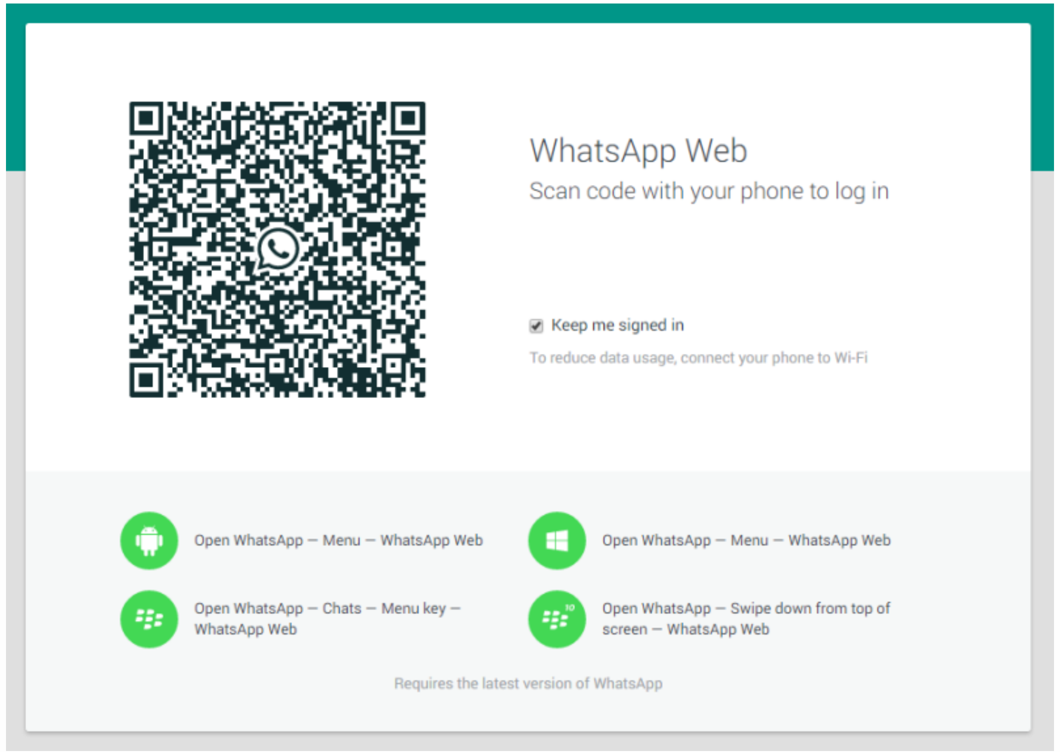


Burned to death because of a rumour on WhatsApp

By Marcos Martínez
BBC Monitoring

The Communication Flow





WhatsApp Web

Scan code with your phone to log in

☑ Keep me signed in

To reduce data usage, connect your phone to Wi-Fi

Open WhatsApp → Menu → WhatsApp Web

Open WhatsApp → Menu → WhatsApp Web

Open WhatsApp → Chats → Menu key → WhatsApp Web

Open WhatsApp → Swipe down from top of screen → WhatsApp Web

Requires the latest version of WhatsApp

WhatsApp Behind the Scenes

ENCRYPTION: Open Whisper System -> Signal -> WhatsApp

COMMUNICATION: WebSocket -> protobuf2 -> JSON

ENCRYPTION: Open Whisper System -> Signal -> WhatsApp

On November 18, 2014, Open Whisper Systems announced a partnership with WhatsApp to provide end-to-end encryption by incorporating the Signal Protocol into each WhatsApp client platform.

On April 5, 2016, WhatsApp and Open Whisper Systems announced that they had finished adding end-to-end encryption to "every form of communication" on WhatsApp, and that users could now verify each other's keys.

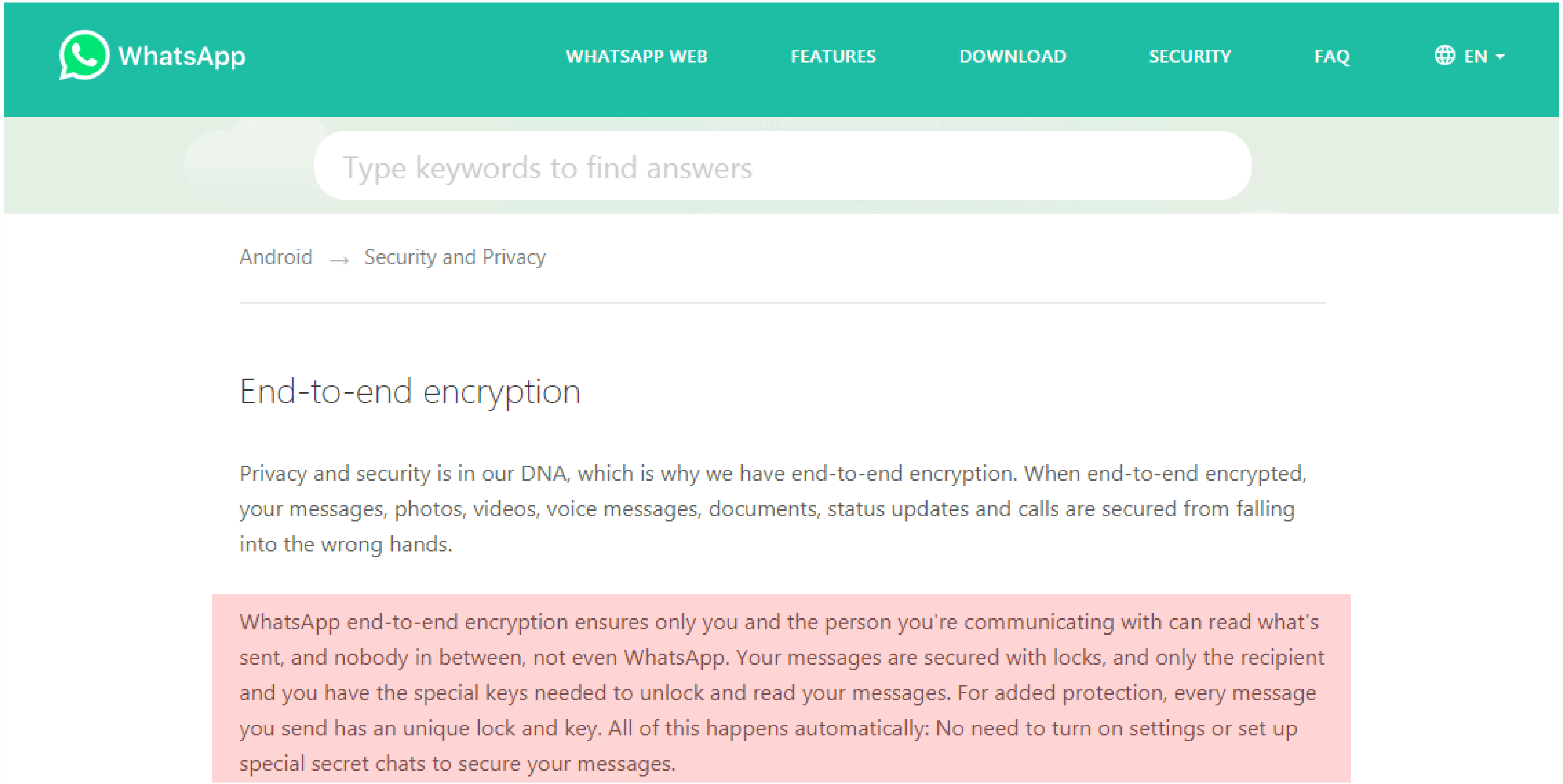
COMMUNICATION: WebSocket -> protobuf2 -> JSON

The **WebSocket API** is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server without having to poll the server for a reply.

The **protobuf** is a method of serializing structured data. It is useful in developing programs to communicate with each other – think XML, but smaller, faster, and simpler.

JSON is a JSON ☺

Is someone can decrypt the traffic?



Android → Security and Privacy

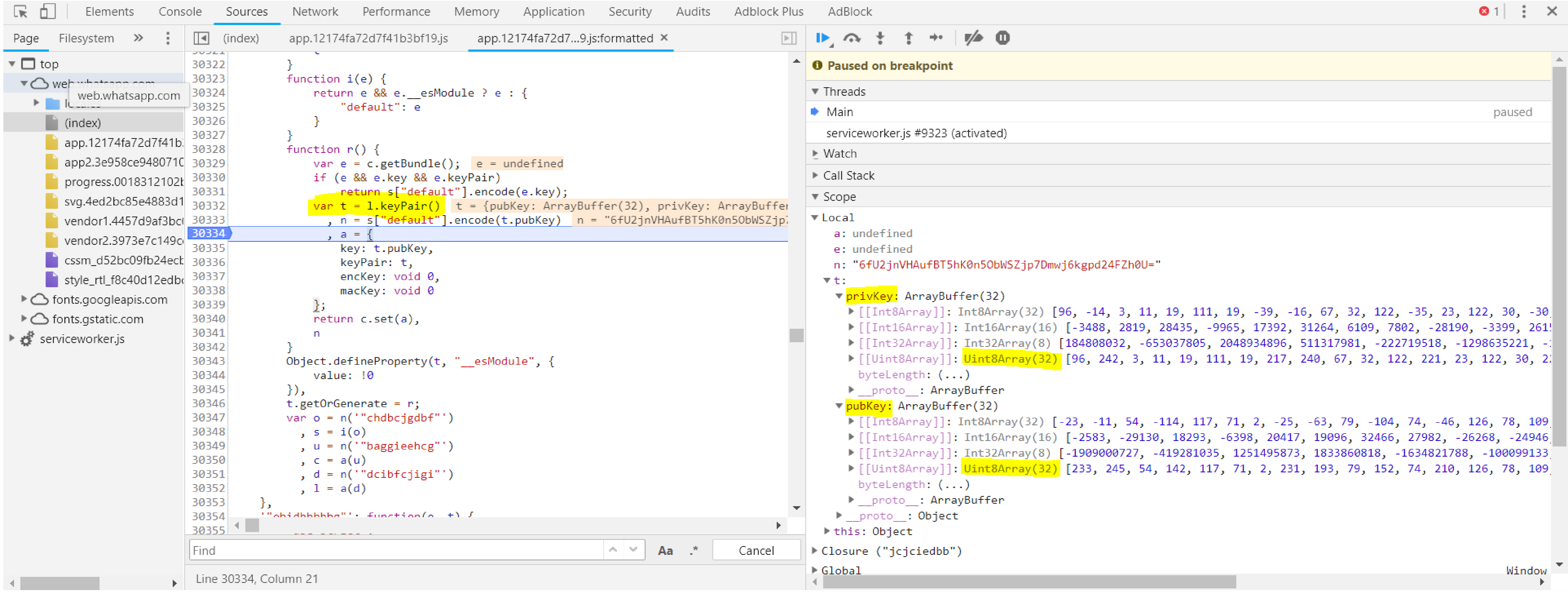
End-to-end encryption

Privacy and security is in our DNA, which is why we have end-to-end encryption. When end-to-end encrypted, your messages, photos, videos, voice messages, documents, status updates and calls are secured from falling into the wrong hands.

WhatsApp end-to-end encryption ensures only you and the person you're communicating with can read what's sent, and nobody in between, not even WhatsApp. Your messages are secured with locks, and only the recipient and you have the special keys needed to unlock and read your messages. For added protection, every message you send has an unique lock and key. All of this happens automatically: No need to turn on settings or set up special secret chats to secure your messages.

WhatsApp Reversing Process

Before generating the QR code, WhatsApp Web generates a Public and Private Key that is used for encryption and decryption Process



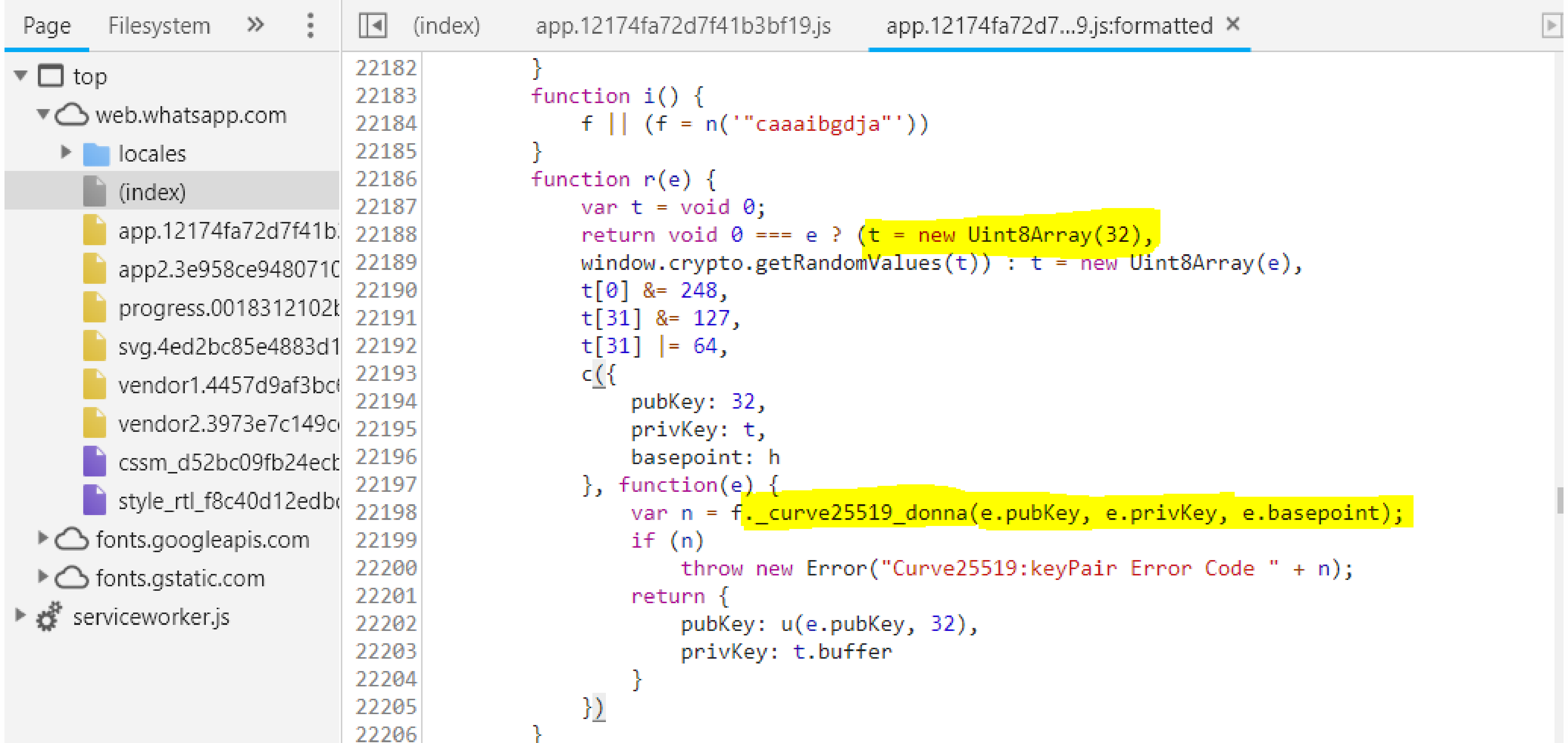
```
        }
        function i(e) {
            return e && e.__esModule ? e : {
                "default": e
            }
        }
        function r() {
            var e = c.getBundle();  e = undefined
            if (e && e.key && e.keyPair)
                return s["default"].encode(e.key);
            var t = l.keyPair()  t = {pubKey: ArrayBuffer(32), privKey: ArrayBuffer
              , n = s["default"].encode(t.pubKey)  n = "6fU2jnVHAufBT5hK0n5ObWSZjp7
              , a = {
                key: t.pubKey,
                keyPair: t,
                encKey: void 0,
                macKey: void 0
            };
            return c.set(a),
            n
        }
        Object.defineProperty(t, "__esModule", {
            value: !0
        }),
        t.getOrGenerate = r;
        var o = n('"chdbcjgdbf"')
          , s = i(o)
          , u = n('"baggieehcg"')
          , c = a(u)
          , d = n('"dcibfcjigi"')
          , l = a(d)
    },
```

```
Local
  a: undefined
  e: undefined
  n: "6fU2jnVHAufBT5hK0n5ObWSZjp7Dmwj6kgpd24FZh0U="
 ▼t:
  ▼privKey: ArrayBuffer(32)
    ▶[[Int8Array]]: Int8Array(32) [96, -14, 3, 11, 19, 111, 19, -39, -16, 67, 32, 122, -35, 23, 122, 30, -30
    ▶[[Int16Array]]: Int16Array(16) [-3488, 2819, 28435, -9965, 17392, 31264, 6109, 7802, -28190, -3399, 261
    ▶[[Int32Array]]: Int32Array(8) [184808032, -653037805, 2048934896, 511317981, -222719518, -1298635221, -
    ▶[[Uint8Array]]: Uint8Array(32) [96, 242, 3, 11, 19, 111, 19, 217, 240, 67, 32, 122, 221, 23, 122, 30, 2
     byteLength: (...)
    ▶__proto__: ArrayBuffer
  ▼pubKey: ArrayBuffer(32)
    ▶[[Int8Array]]: Int8Array(32) [-23, -11, 54, -114, 117, 71, 2, -25, -63, 79, -104, 74, -46, 126, 78, 109
    ▶[[Int16Array]]: Int16Array(16) [-2583, -29130, 18293, -6398, 20417, 19096, 32466, 27982, -26268, -24946
    ▶[[Int32Array]]: Int32Array(8) [-1909000727, -419281035, 1251495873, 1833860818, -1634821788, -100099133
    ▶[[Uint8Array]]: Uint8Array(32) [233, 245, 54, 142, 117, 71, 2, 231, 193, 79, 152, 74, 210, 126, 78, 109
     byteLength: (...)
    ▶__proto__: ArrayBuffer
  ▶__proto__: Object
 ▶this: Object
▶Closure ("jcjciedbb")
▶Global
```

WhatsApp Reversing Process

These keys were created by using curve25519_donna by using random 32 bytes.

In cryptography, **Curve25519** is an elliptic curve offering 128 bits of security and designed for use with the elliptic curve Diffie–Hellman (ECDH) key agreement scheme. It is one of the fastest ECC curves and is not covered by any known patents



```
        }
        function i() {
            f || (f = n('"caaaibgdja"'))
        }
        function r(e) {
            var t = void 0;
            return void 0 === e ? (t = new Uint8Array(32),
            window.crypto.getRandomValues(t)) : t = new Uint8Array(e),
            t[0] &= 248,
            t[31] &= 127,
            t[31] |= 64,
            c({
                pubKey: 32,
                privKey: t,
                basepoint: h
            }, function(e) {
                var n = f._curve25519_donna(e.pubKey, e.privKey, e.basepoint);
                if (n)
                    throw new Error("Curve25519:keyPair Error Code " + n);
                return {
                    pubKey: u(e.pubKey, 32),
                    privKey: t.buffer
                }
            })
        }
```

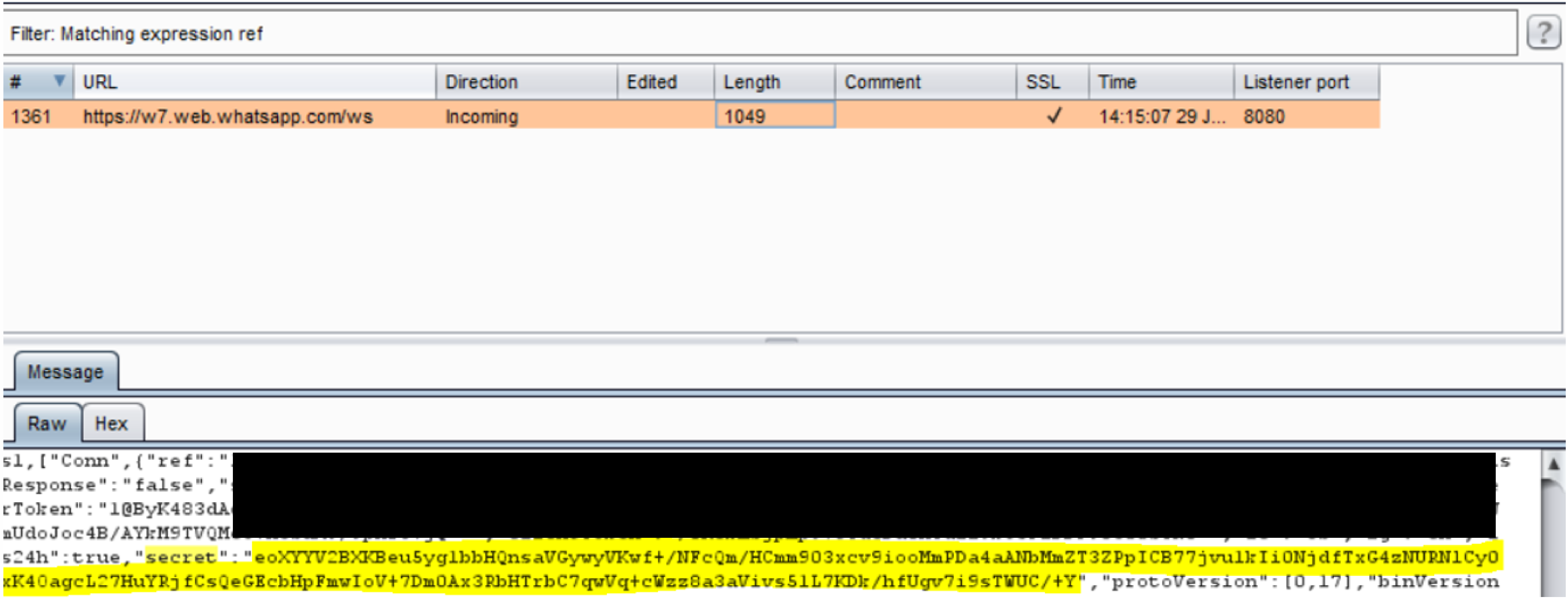
WhatsApp Reversing Process

To decrypt the data we will start to create a decryption code. This will take the private key from WhatsApp Web instead of the random bytes because we need to have the same keys in order to decrypt the data:

```
self.private_key = curve25519.Private(“”.join([chr(x) for x in priv_key_list]))
self.public_key  = self.private_key.get_public()
```

WhatsApp Reversing Process

Then, after scanning the QR code with the phone we have to take the generated secret:

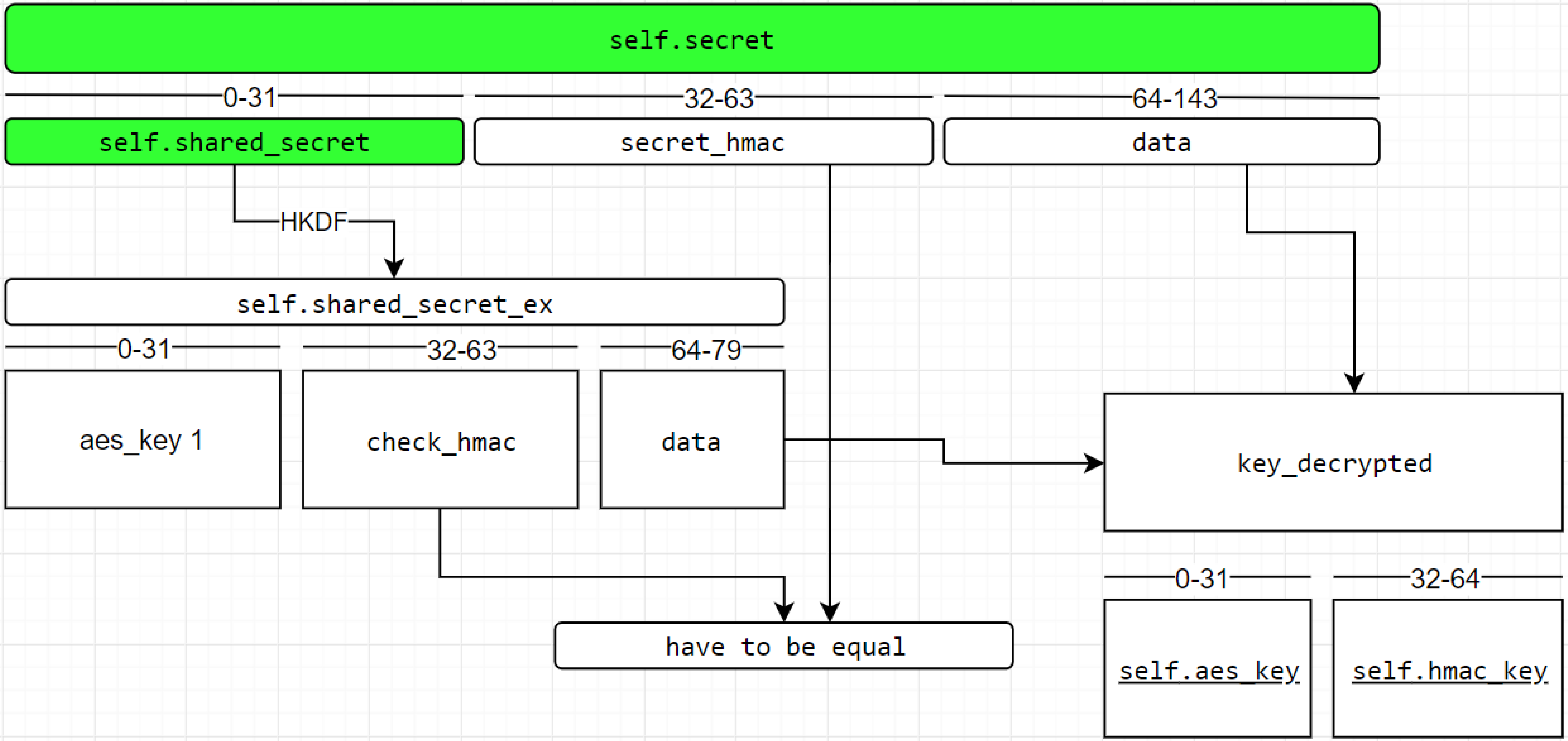


Filter: Matching expression ref

#	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
1361	https://w7.web.whatsapp.com/ws	Incoming		1049		✓	14:15:07 29 J...	8080

```
s1,["Conn",{"ref":"[redacted]
Response":"false","
rToken":"1@ByK483dA
mUdoJoc4B/AYkM9TVQM
s24h":true,"secret":"eoXYYV2BXKBeu5yg1bbHQnsaVGywyVKwf+/NFcQm/HCmm903xcv9iooMmPDa4aANbMmZT3ZPpICB77jvulkIi0NjdfTxG4zNUPN1Cy0
xK40agcL27HuYRjfCsQeGEcbHpFmwIoV+7Dm0Ax3RbHTrbC7qwVq+cWzz8a3aVivs51L7KDk/hfUgv7i9sTWUC/+Y","protoVersion":[0,17],"binVersion
```

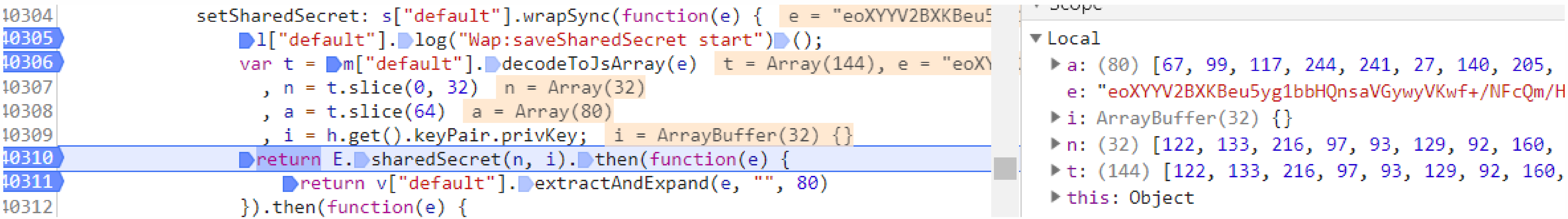
WhatsApp Reversing Process – Shared Secret



WhatsApp Reversing Process – Shared Secret

Then we have 2 interesting functions:

- **setSharedSecret –** This function divides the secret into slices and configure the shared secret.
- **E.SharedSecret –** This function uses two parameters which were the first 32 bytes and the private key from the QR generation:

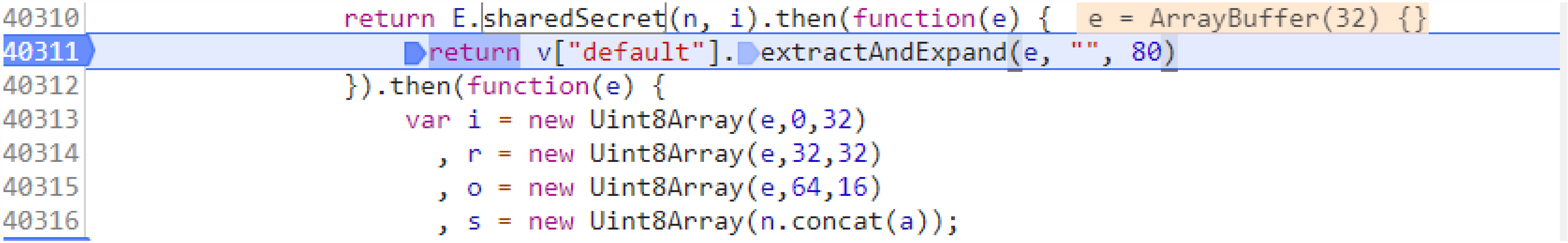


```
        setSharedSecret: s["default"].wrapSync(function(e) {  e = "eoXYYV2BXKBeu5
            l["default"].log("Wap:saveSharedSecret start")();
            var t = m["default"].decodeToJsArray(e)  t = Array(144), e = "eoXY
              , n = t.slice(0, 32)  n = Array(32)
              , a = t.slice(64)  a = Array(80)
              , i = h.get().keyPair.privKey;  i = ArrayBuffer(32) {}
            return E.sharedSecret(n, i).then(function(e) {
                return v["default"].extractAndExpand(e, "", 80)
            }).then(function(e) {
```

```
Scope
▼ Local
  ▶ a: (80) [67, 99, 117, 244, 241, 27, 140, 205,
    e: "eoXYYV2BXKBeu5yg1bbHQnsaVGywyVKwf+/NFcQm/H
  ▶ i: ArrayBuffer(32) {}
  ▶ n: (32) [122, 133, 216, 97, 93, 129, 92, 160,
  ▶ t: (144) [122, 133, 216, 97, 93, 129, 92, 160,
  ▶ this: Object
```

WhatsApp Reversing Process – Shared Secret

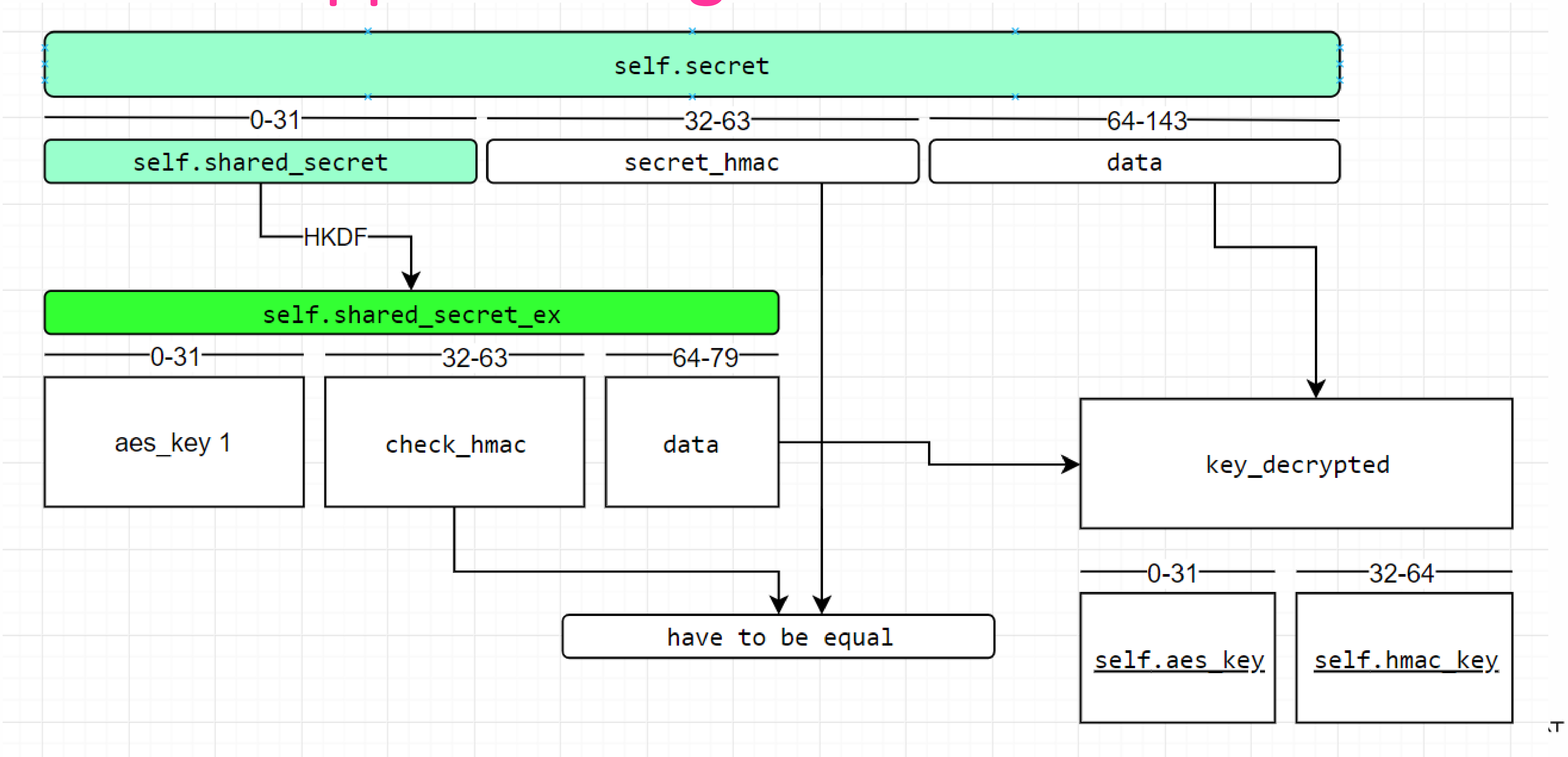
Next we have the expanded shared secret which is 80 bytes:



```
return E.sharedSecret(n, i).then(function(e) {  e = ArrayBuffer(32) {}
    return v["default"].extractAndExpand(e, "", 80)
}).then(function(e) {
    var i = new Uint8Array(e,0,32)
      , r = new Uint8Array(e,32,32)
      , o = new Uint8Array(e,64,16)
      , s = new Uint8Array(n.concat(a));
```

By diving in we can see that the function uses the HKDF, is a simple hmac key derivation function (KDF) function.

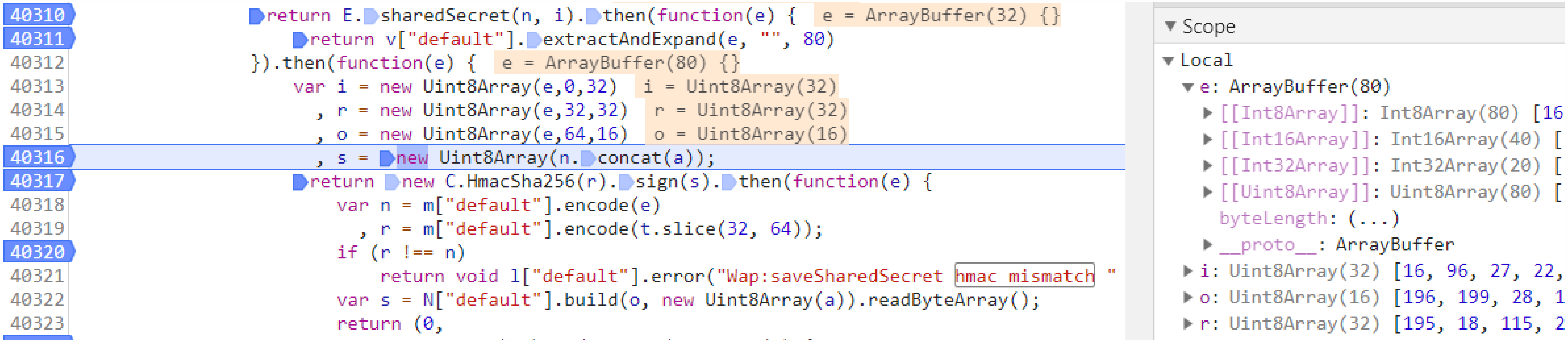
WhatsApp Reversing Process – Shared Secret



WhatsApp Reversing Process – hmac_sha256

We next have the hmac validation function which takes the expanded data as parameter ‘e’ and divides it into 3 parameters:

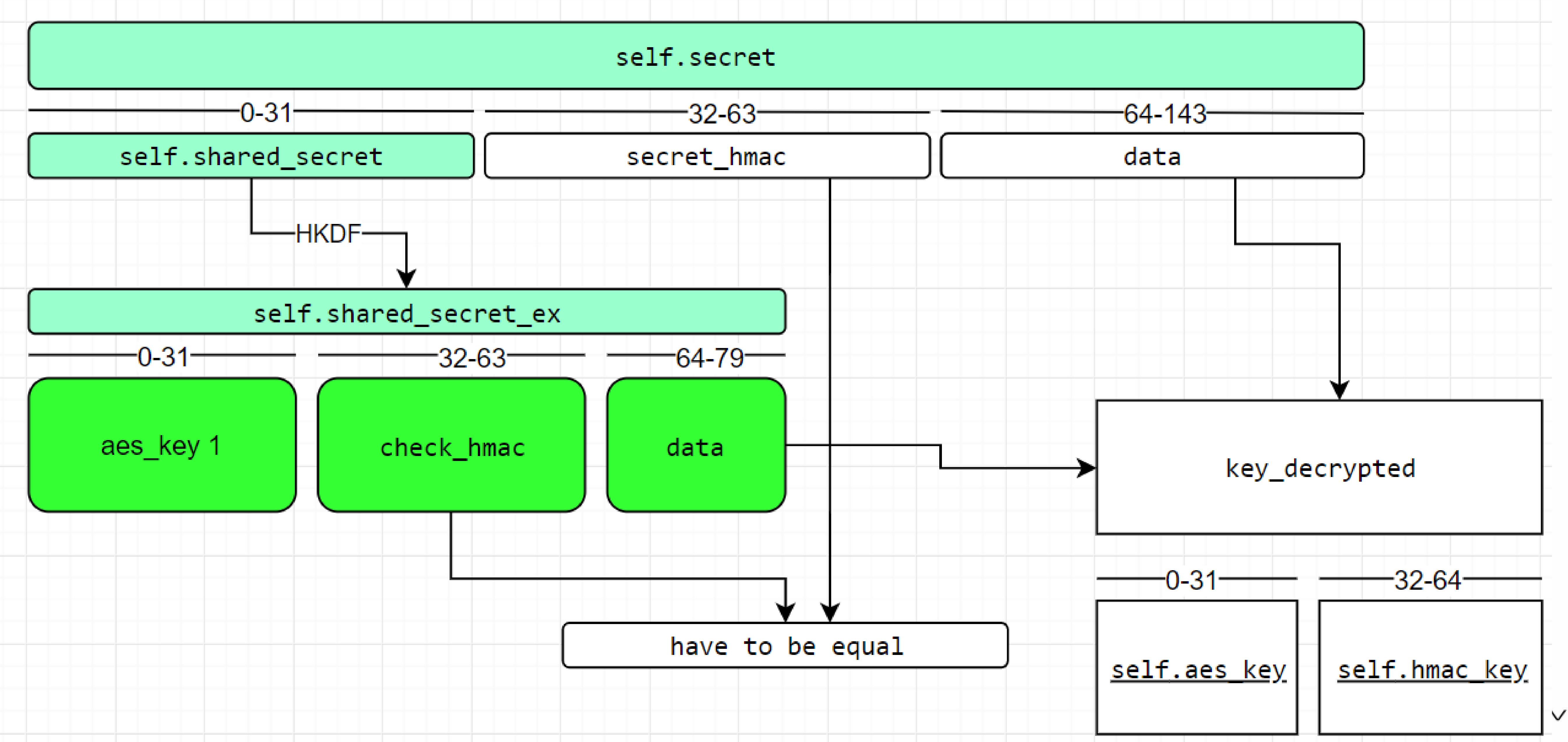
i – The first 32 bytes of shared_expended is the **aes key**
r – The next 32 bytes is the **hmac**
o – The last 16 bytes is the **aes data part**



```
        return E.sharedSecret(n, i).then(function(e) {  e = ArrayBuffer(32) {}
            return v["default"].extractAndExpand(e, "", 80)
        }).then(function(e) {  e = ArrayBuffer(80) {}
            var i = new Uint8Array(e,0,32)  i = Uint8Array(32)
              , r = new Uint8Array(e,32,32)  r = Uint8Array(32)
              , o = new Uint8Array(e,64,16)  o = Uint8Array(16)
              , s = new Uint8Array(n.concat(a));
            return new C.HmacSha256(r).sign(s).then(function(e) {
                var n = m["default"].encode(e)
                  , r = m["default"].encode(t.slice(32, 64));
                if (r !== n)
                    return void l["default"].error("Wap:saveSharedSecret hmac mismatch "
                var s = N["default"].build(o, new Uint8Array(a)).readByteArray();
                return (0,
```

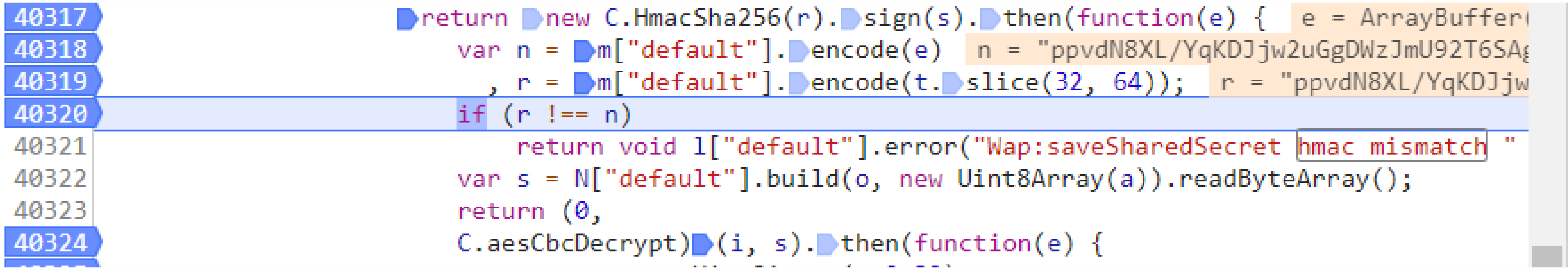
```
▼ Scope
▼ Local
  ▼ e: ArrayBuffer(80)
    ▶ [[Int8Array]]: Int8Array(80) [16
    ▶ [[Int16Array]]: Int16Array(40) [
    ▶ [[Int32Array]]: Int32Array(20) [
    ▶ [[Uint8Array]]: Uint8Array(80) [
      byteLength: (...)
    ▶ __proto__: ArrayBuffer
  ▶ i: Uint8Array(32) [16, 96, 27, 22,
  ▶ o: Uint8Array(16) [196, 199, 28, 1
  ▶ r: Uint8Array(32) [195, 18, 115, 2
```

WhatsApp Reversing Process – hmac_sha256

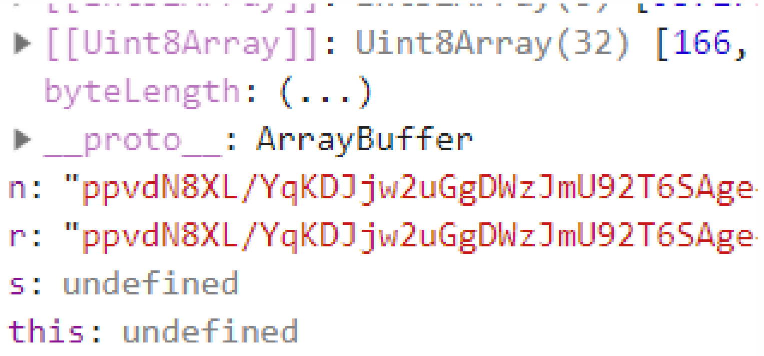


WhatsApp Reversing Process – hmac_sha256

Then the function HmacSha256 will be called with the parameter ‘r’ and it will sign the data with the parameter ‘s’, after that ‘n’ will receive the **hmac** verifier which will be compared to ‘r’(the **hmac** from **extended shared secret)**



```
return new C.HmacSha256(r).sign(s).then(function(e) {   e = ArrayBuffer(
    var n = m["default"].encode(e)   n = "ppvdN8XL/YqKDJjw2uGgDWzJmU92T6SAg
      , r = m["default"].encode(t.slice(32, 64));   r = "ppvdN8XL/YqKDJjw
    if (r !== n)
        return void l["default"].error("Wap:saveSharedSecret hmac mismatch "
    var s = N["default"].build(o, new Uint8Array(a)).readByteArray();
    return (0,
    C.aesCbcDecrypt)(i, s).then(function(e) {
```

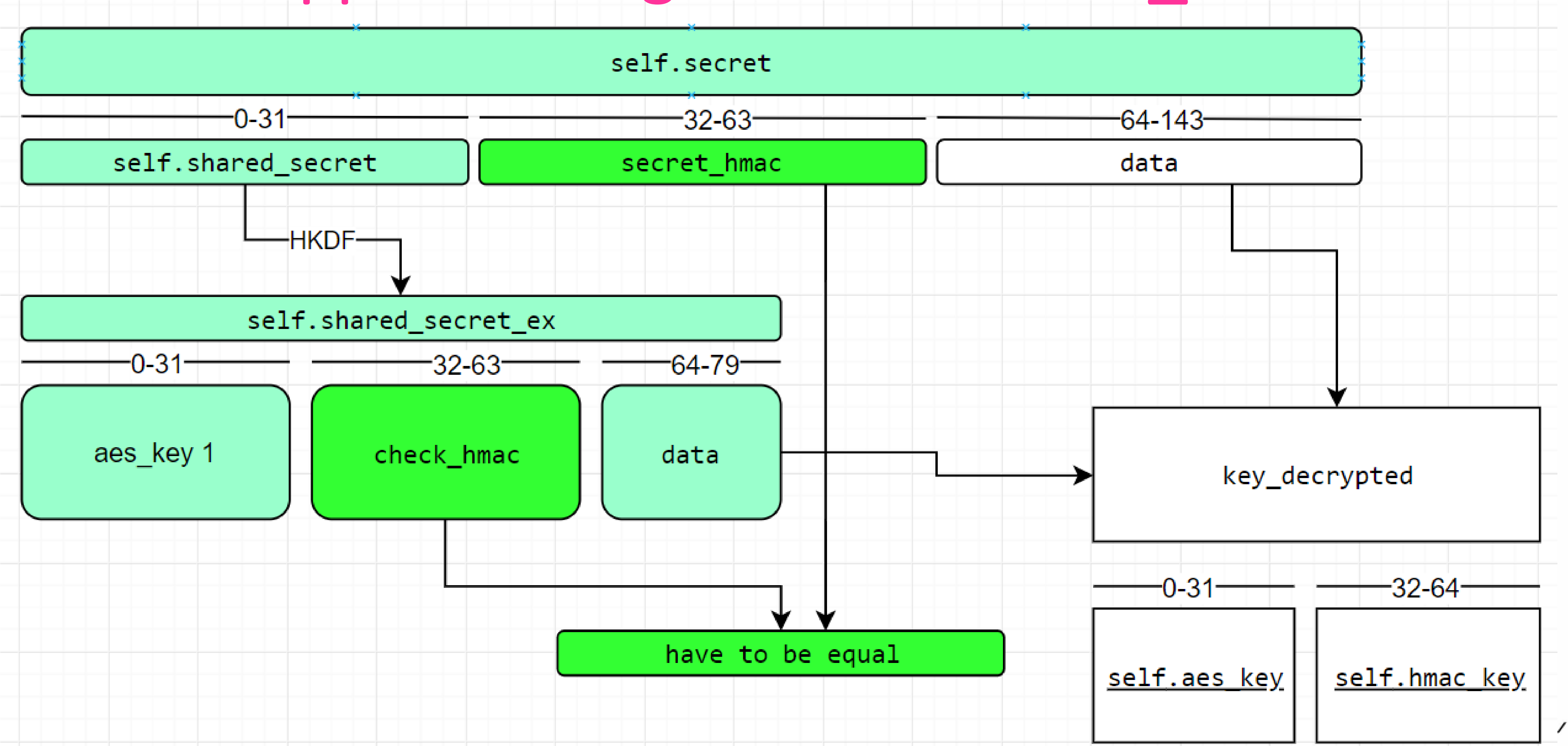


```
▶ [[Uint8Array]]: Uint8Array(32) [166,
  byteLength: (...)
▶ __proto__: ArrayBuffer
n: "ppvdN8XL/YqKDJjw2uGgDWzJmU92T6SAge
r: "ppvdN8XL/YqKDJjw2uGgDWzJmU92T6SAge
s: undefined
this: undefined
```

In python it will look like this:

check_hmac = HmacSha256(**shared_expended**[32:64], self.**secret**[:32] + self.**secret**[64:])
if check_hmac != self.**secret**[32:64]:
        raise ValueError(“**hmac mismatch**”)

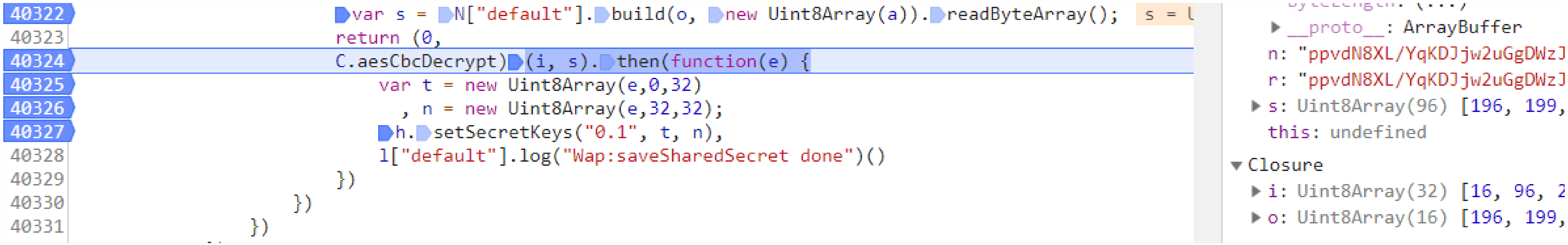
WhatsApp Reversing Process – hmac_sha256



WhatsApp Reversing Process – AES Keys

The last encryption related function in this block is ‘**aesCbcDecrypt**’ which uses two parameters:

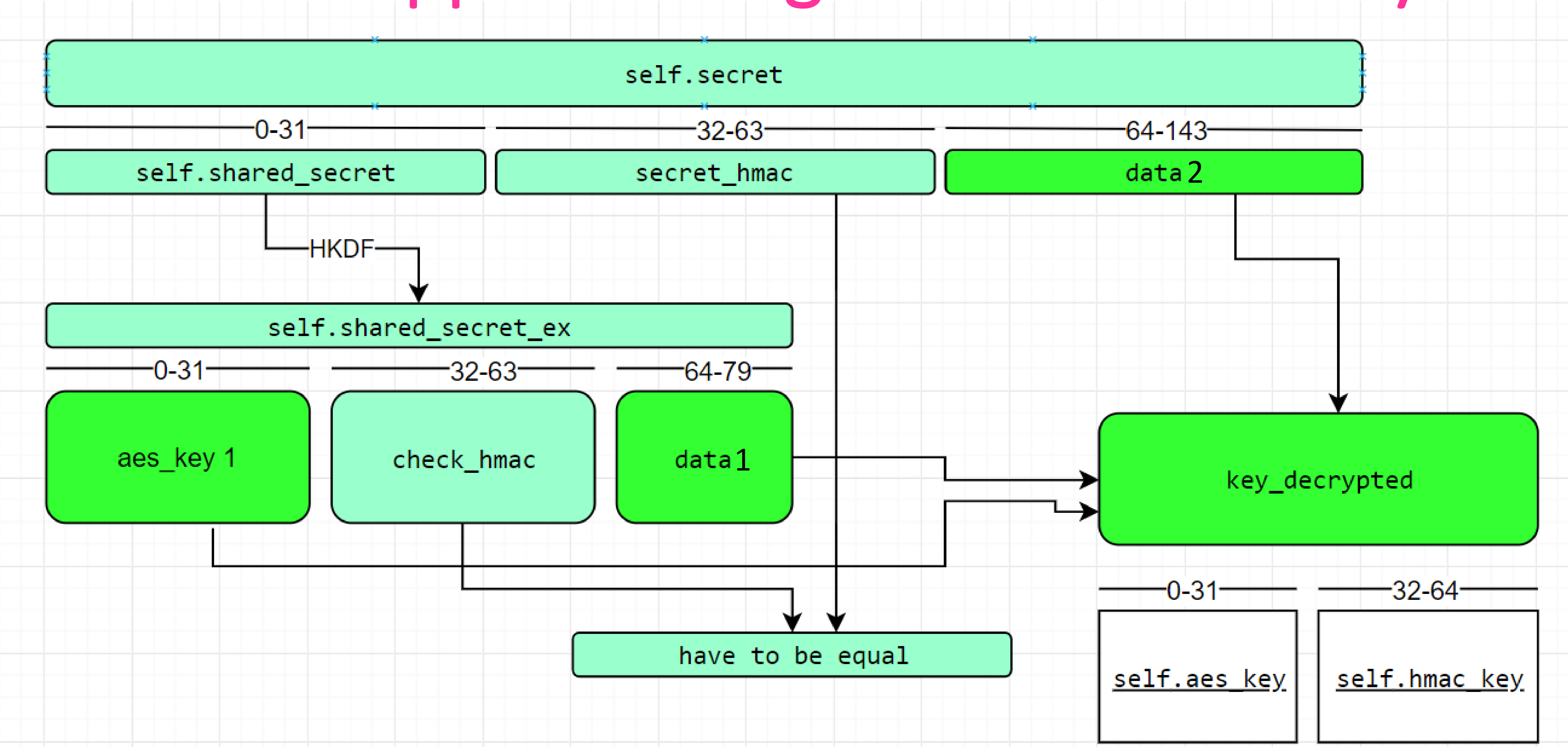
- s – which is a concatenation between the **last 16 bytes of the expanded shared secret** and the **data from byte 64 of the secret**.
- i – which is the **aes key**.



```
        var s = N["default"].build(o, new Uint8Array(a)).readByteArray();   s = U
        return (0,
        C.aesCbcDecrypt)(i, s).then(function(e) {
            var t = new Uint8Array(e,0,32)
              , n = new Uint8Array(e,32,32);
            h.setSecretKeys("0.1", t, n),
            l["default"].log("Wap:saveSharedSecret done")()
        })
    })
})
```

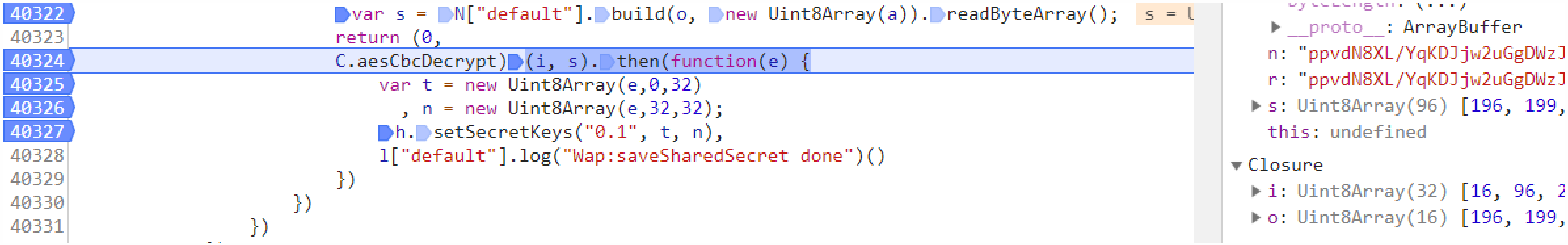
```
    byteLength: (...)
  ▶ __proto__: ArrayBuffer
  n: "ppvdN8XL/YqKDJjw2uGgDWzJ
  r: "ppvdN8XL/YqKDJjw2uGgDWzJ
▶ s: Uint8Array(96) [196, 199,
  this: undefined
▼ Closure
▶ i: Uint8Array(32) [16, 96, 2
▶ o: Uint8Array(16) [196, 199,
```

WhatsApp Reversing Process – AES Keys



WhatsApp Reversing Process – AES Keys

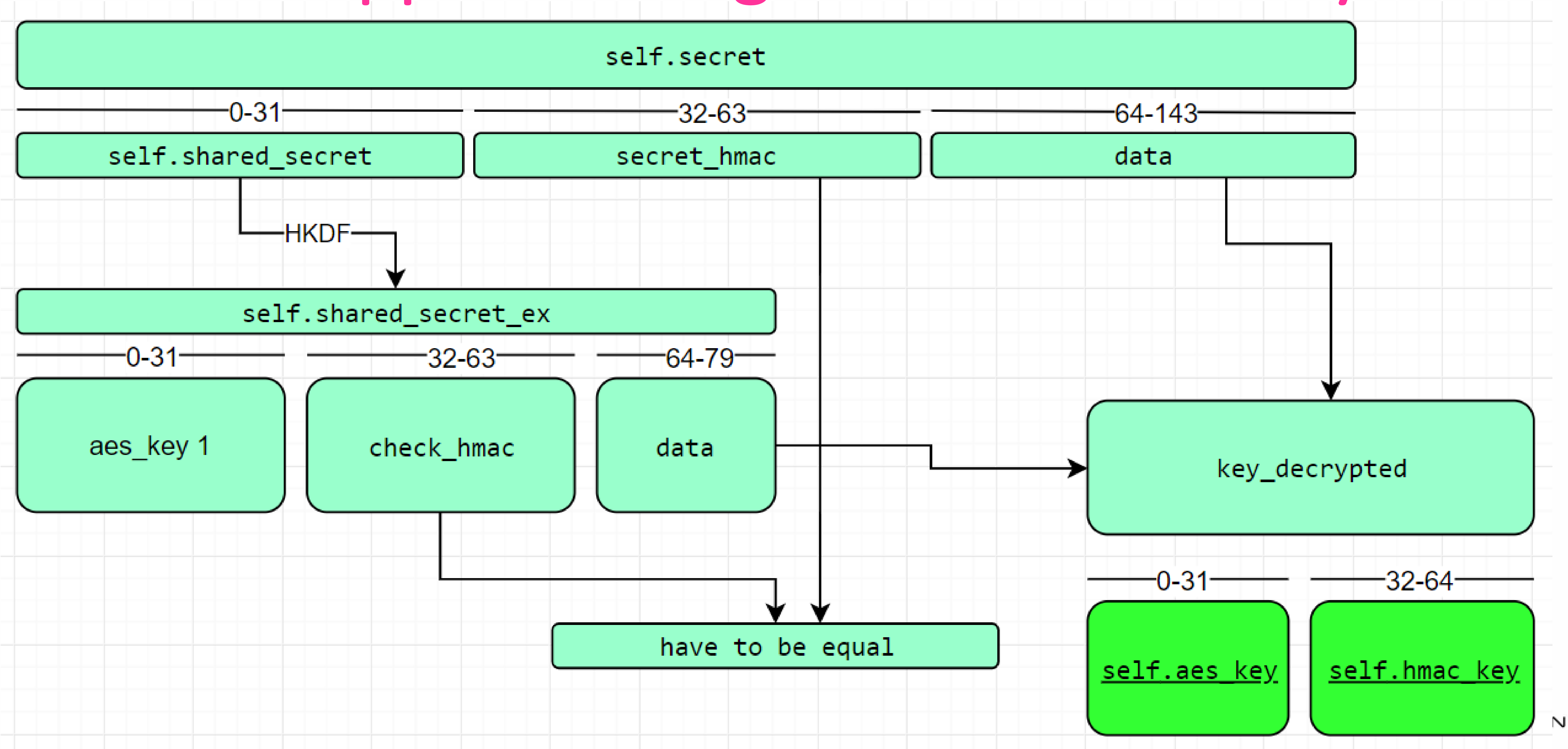
This way we will get the AES Key ‘t’ and HMAC Key ‘n’



```
    var s = N["default"].build(o, new Uint8Array(a)).readByteArray();   s = l
    return (0,
    C.aesCbcDecrypt)(i, s).then(function(e) {
      var t = new Uint8Array(e,0,32)
        , n = new Uint8Array(e,32,32);
      h.setSecretKeys("0.1", t, n),
      l["default"].log("Wap:saveSharedSecret done")()
    })
  })
})
```

```
    byteLength: (...)
  ▶ __proto__: ArrayBuffer
  n: "ppvdN8XL/YqKDJjw2uGgDWzJ
  r: "ppvdN8XL/YqKDJjw2uGgDWzJ
▶ s: Uint8Array(96) [196, 199,
  this: undefined
▼ Closure
▶ i: Uint8Array(32) [16, 96, 2
▶ o: Uint8Array(16) [196, 199,
```

WhatsApp Reversing Process – AES Keys

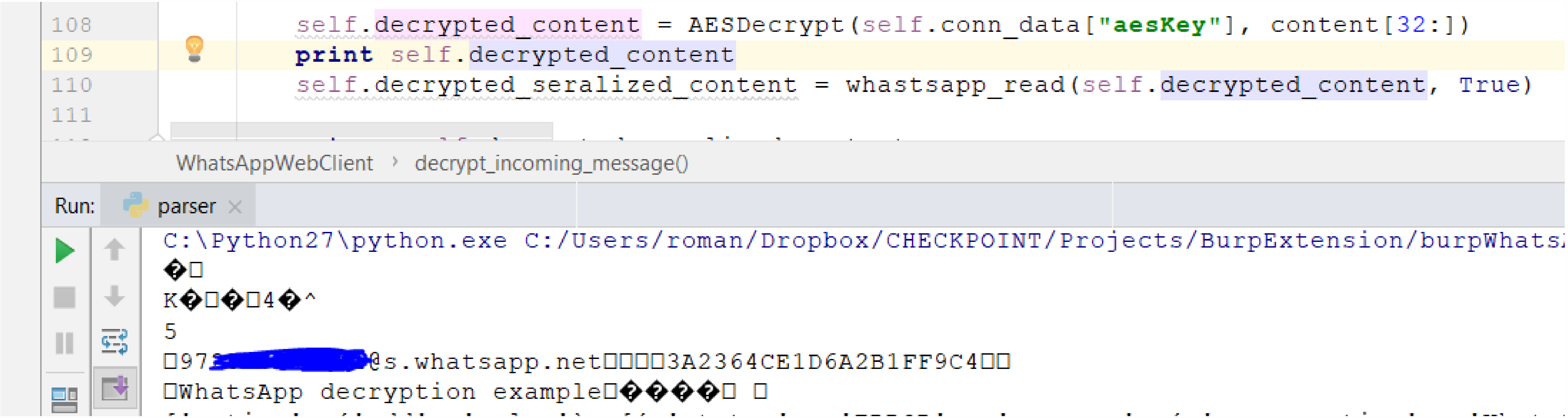


WhatsApp Reversing Process – Code

```
self.secret = None
self.private_key = None
self.public_key = None
self.shared_secret = None
self.shared_secret_ex = None
self.aes_key = None

self.private_key = curve25519.Private("".join([chr(x) for x in priv_key_list]))
self.public_key = self.private_key.get_public()

assert (self.public_key.serialize() == "".join([chr(x) for x in pub_key_list]))

self.secret = base64.b64decode(ref_dict["secret"])
self.shared_secret = self.private_key.get_shared_key(curve25519.Public(self.secret[:32]), lambda key: key)
self.shared_secret_ex = HKDF(self.shared_secret, 80)

check_hmac = hmac_sha256(self.shared_secret_ex[32:64], self.secret[:32] + self.secret[64:])
if check_hmac != self.secret[32:64]:
    raise ValueError("hmac mismatch")

key_decrypted = aes_decrypt(self.shared_secret_ex[:32], self.shared_secret_ex[64:] + self.secret[64:])
self.aes_key = key_decrypted[:32]
self.mac_key = key_decrypted[32:64]
```

WhatsApp Reversing Process – protobuf data

By using the keys we can decrypt any incoming message, the decryption result will be the protobuf message.

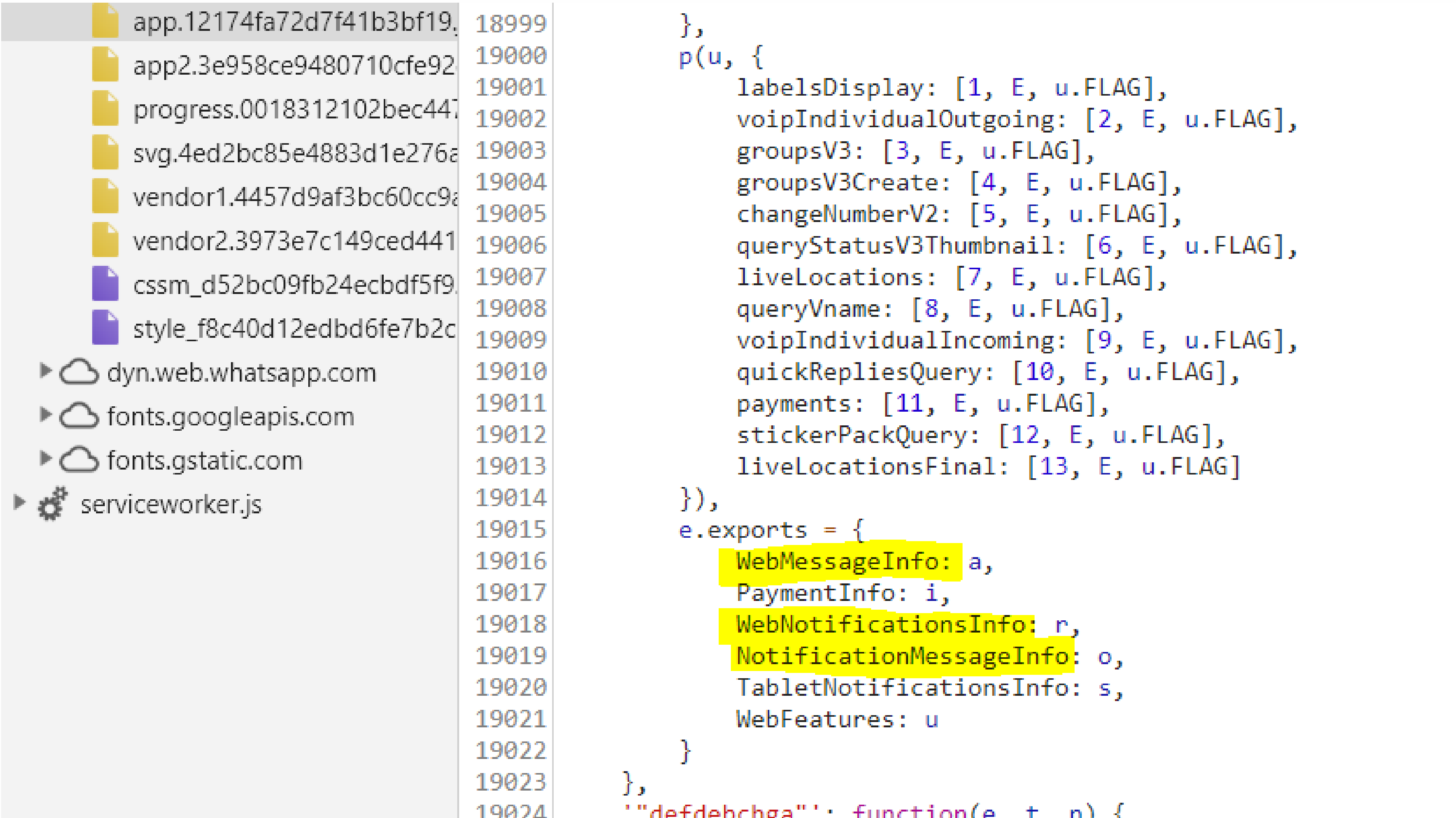


```
        self.decrypted_content = AESDecrypt(self.conn_data["aesKey"], content[32:])
        print self.decrypted_content
        self.decrypted_seralized_content = whastsapp_read(self.decrypted_content, True)
```

```
C:\Python27\python.exe C:/Users/roman/Dropbox/CHECKPOINT/Projects/BurpExtension/burpWhats
�□
K�□�□4�^
5
□97[redacted]@s.whatsapp.net□□□□3A2364CE1D6A2B1FF9C4□□
□WhatsApp decryption example□�����□ □
```

WhatsApp Reversing Process – protobuf data

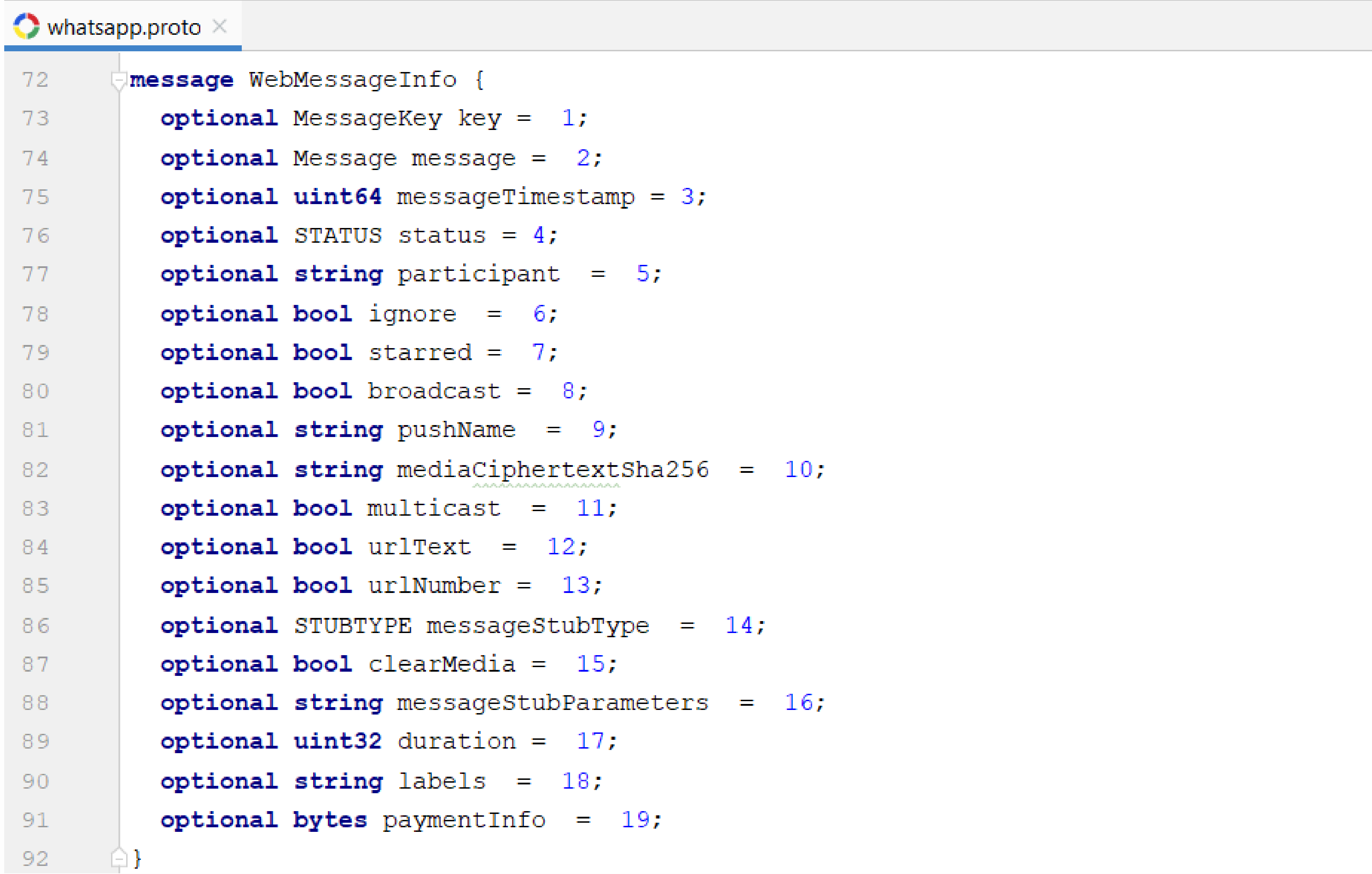
In order to deserialize the protobuf we have to create our mapping, based on whatsapp protobuf that can be found in the file app:



```
        },
        p(u, {
            labelsDisplay: [1, E, u.FLAG],
            voipIndividualOutgoing: [2, E, u.FLAG],
            groupsV3: [3, E, u.FLAG],
            groupsV3Create: [4, E, u.FLAG],
            changeNumberV2: [5, E, u.FLAG],
            queryStatusV3Thumbnail: [6, E, u.FLAG],
            liveLocations: [7, E, u.FLAG],
            queryVname: [8, E, u.FLAG],
            voipIndividualIncoming: [9, E, u.FLAG],
            quickRepliesQuery: [10, E, u.FLAG],
            payments: [11, E, u.FLAG],
            stickerPackQuery: [12, E, u.FLAG],
            liveLocationsFinal: [13, E, u.FLAG]
        }),
        e.exports = {
            WebMessageInfo: a,
            PaymentInfo: i,
            WebNotificationsInfo: r,
            NotificationMessageInfo: o,
            TabletNotificationsInfo: s,
            WebFeatures: u
        }
    },
    '"defdebcbga"': function(e, t, n) {
```

WhatsApp Reversing Process – protobuf data

This is a part of our protobuf file:

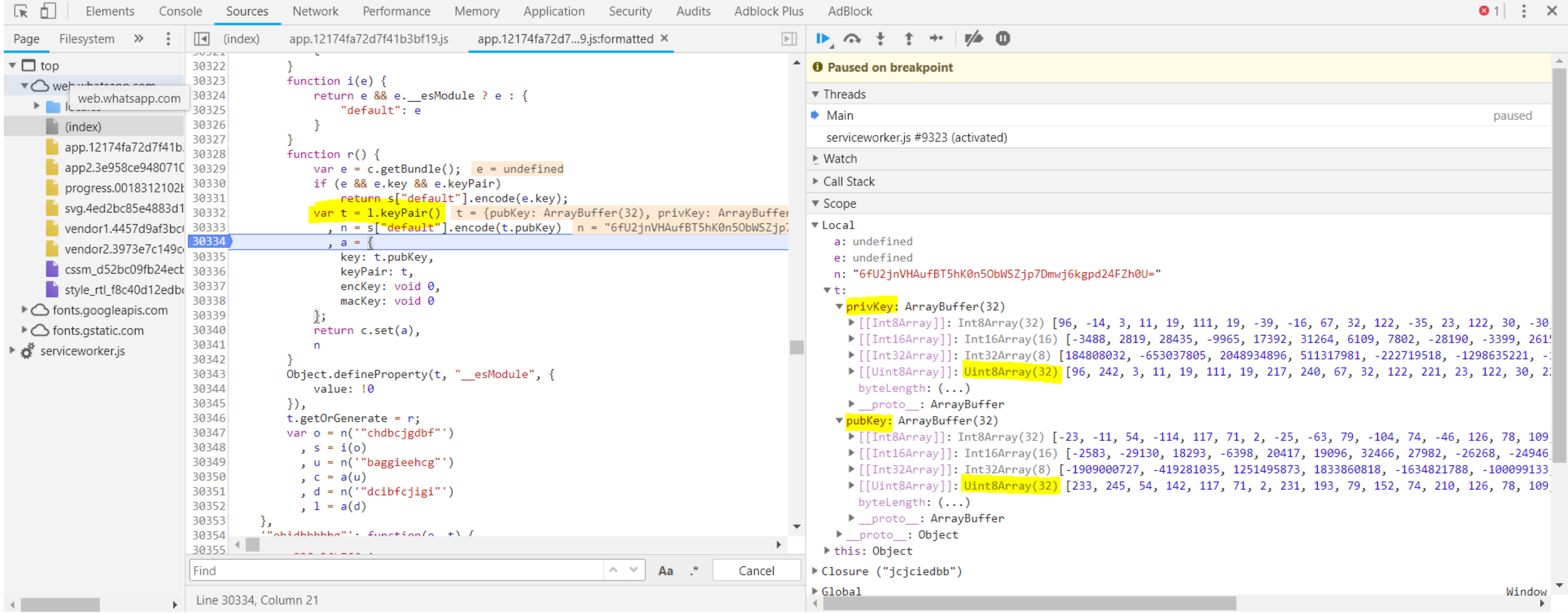


```
message WebMessageInfo {
   optional MessageKey key =  1;
   optional Message message =  2;
   optional uint64 messageTimestamp = 3;
   optional STATUS status = 4;
   optional string participant  =  5;
   optional bool ignore  =  6;
   optional bool starred =  7;
   optional bool broadcast =  8;
   optional string pushName  =  9;
   optional string mediaCiphertextSha256  =  10;
   optional bool multicast  =  11;
   optional bool urlText  =  12;
   optional bool urlNumber =  13;
   optional STUBTYPE messageStubType  =  14;
   optional bool clearMedia =  15;
   optional string messageStubParameters  =  16;
   optional uint32 duration =  17;
   optional string labels  =  18;
   optional bytes paymentInfo  =  19;
}
```

Burp Extension

Accessing the Keys – Burp Extension Keys

Let’s start with WhatsApp Web. Before generating the QR code,
WhatsApp Web generates a Public and Private Key that is used for encryption and decryption

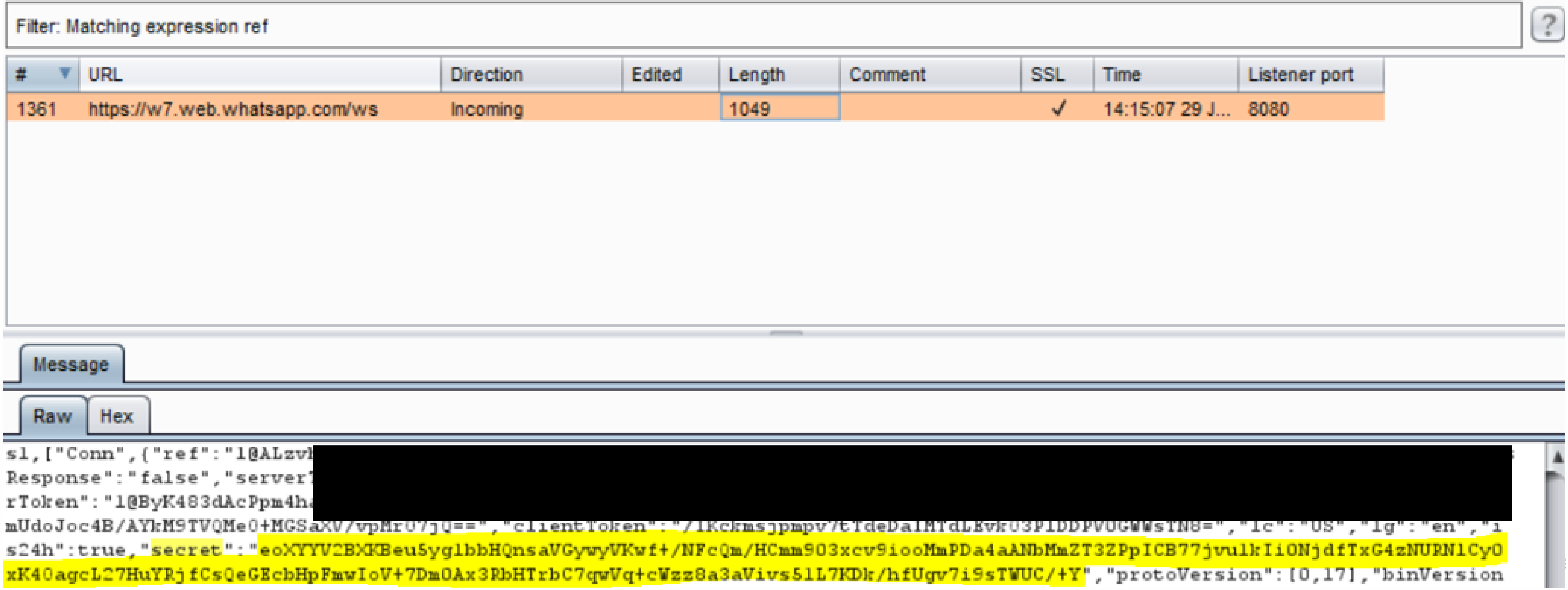


```
        }
        function i(e) {
            return e && e.__esModule ? e : {
                "default": e
            }
        }
        function r() {
            var e = c.getBundle();  e = undefined
            if (e && e.key && e.keyPair)
                return s["default"].encode(e.key);
            var t = l.keyPair()  t = {pubKey: ArrayBuffer(32), privKey: ArrayBuffer
              , n = s["default"].encode(t.pubKey)  n = "6fU2jnVHAufBT5hK0n5ObWSZjp7
              , a = {
                key: t.pubKey,
                keyPair: t,
                encKey: void 0,
                macKey: void 0
            };
            return c.set(a),
            n
        }
        Object.defineProperty(t, "__esModule", {
            value: !0
        }),
        t.getOrGenerate = r;
        var o = n('"chdbcjgdbf"')
          , s = i(o)
          , u = n('"baggieehcg"')
          , c = a(u)
          , d = n('"dcibfcjigi"')
          , l = a(d)
    },
```

```
▼ Local
    a: undefined
    e: undefined
    n: "6fU2jnVHAufBT5hK0n5ObWSZjp7Dmwj6kgpd24FZh0U="
  ▼ t:
    ▼ privKey: ArrayBuffer(32)
      ▶ [[Int8Array]]: Int8Array(32) [96, -14, 3, 11, 19, 111, 19, -39, -16, 67, 32, 122, -35, 23, 122, 30, -30
      ▶ [[Int16Array]]: Int16Array(16) [-3488, 2819, 28435, -9965, 17392, 31264, 6109, 7802, -28190, -3399, 261
      ▶ [[Int32Array]]: Int32Array(8) [184808032, -653037805, 2048934896, 511317981, -222719518, -1298635221, -
      ▶ [[Uint8Array]]: Uint8Array(32) [96, 242, 3, 11, 19, 111, 19, 217, 240, 67, 32, 122, 221, 23, 122, 30, 2
        byteLength: (...)
      ▶ __proto__: ArrayBuffer
    ▼ pubKey: ArrayBuffer(32)
      ▶ [[Int8Array]]: Int8Array(32) [-23, -11, 54, -114, 117, 71, 2, -25, -63, 79, -104, 74, -46, 126, 78, 109
      ▶ [[Int16Array]]: Int16Array(16) [-2583, -29130, 18293, -6398, 20417, 19096, 32466, 27982, -26268, -24946
      ▶ [[Int32Array]]: Int32Array(8) [-1909000727, -419281035, 1251495873, 1833860818, -1634821788, -100099133
      ▶ [[Uint8Array]]: Uint8Array(32) [233, 245, 54, 142, 117, 71, 2, 231, 193, 79, 152, 74, 210, 126, 78, 109
        byteLength: (...)
      ▶ __proto__: ArrayBuffer
    ▶ __proto__: Object
  ▶ this: Object
▶ Closure ("jcjciedbb")
▶ Global                                                        Window
```

Accessing the Keys – Burp Extension Secret

After the QR code is created, after scanning it with a phone
We can send the following information to WhatsApp Web over a WebSocket.



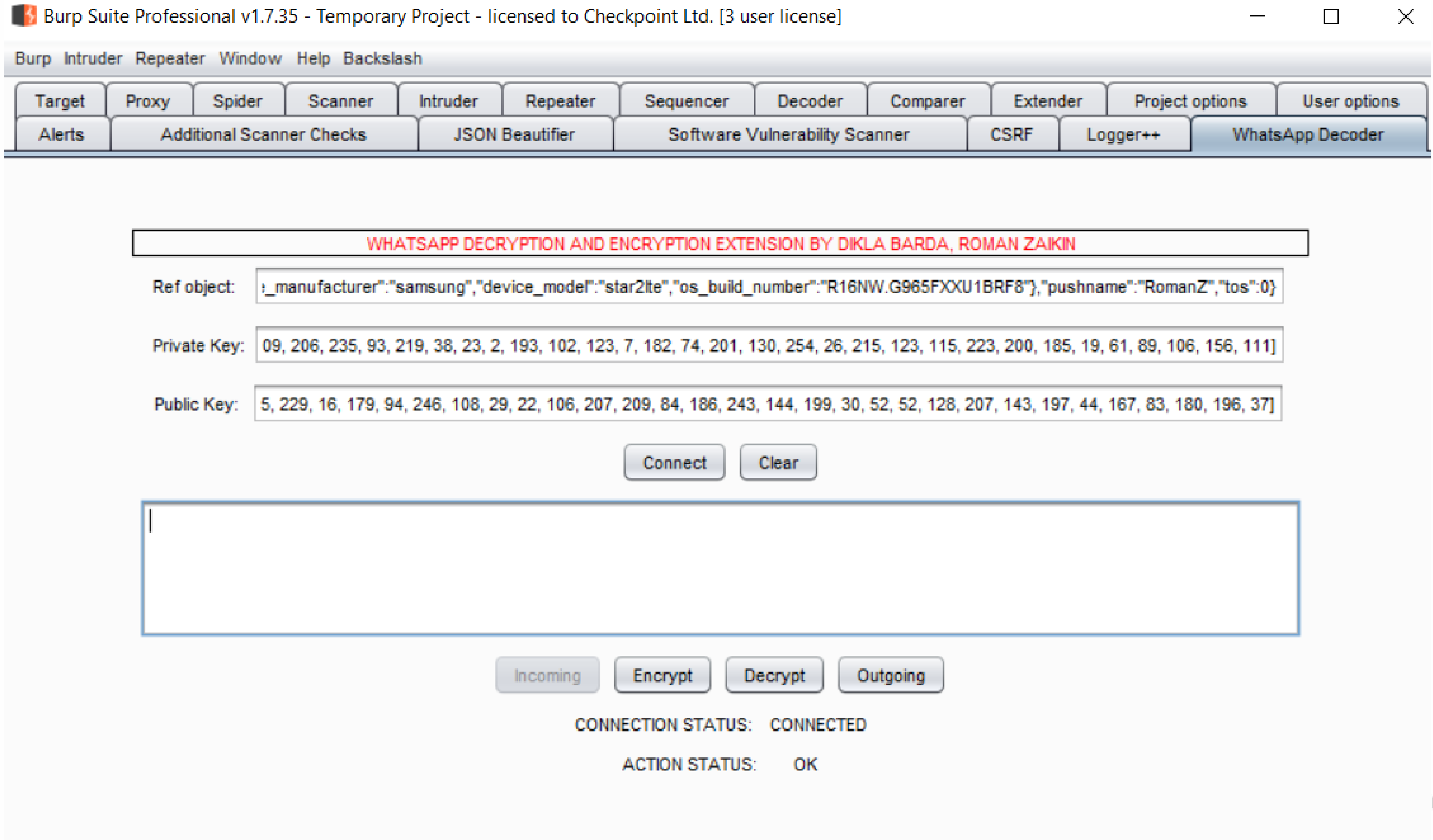
Filter: Matching expression ref

#	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
1361	https://w7.web.whatsapp.com/ws	Incoming		1049		✓	14:15:07 29 J...	8080

```
s1,["Conn",{"ref":"1@ALzv[redacted]
Response":"false","server
rToken":"1@ByK483dAcPpm4h
mUdoJoc4B/AYkM9TVQMe0+MGSaXV/vpMr07jQ==","clientToken":"/1Kckmsjpmpv7tTdeDaIMTdLEvk03PIDDPVUGWWsTN8=","lc":"US","lg":"en","i
s24h":true,"secret":"eoXYYV2BXKBeu5yglbbHQnsaVGywyVKwf+/NFcQm/HCmm903xcv9iooMmPDa4aANbMmZT3ZPpICB77jvulkIi0NjdfTxG4zNURN1CyO
xK40agcL27HuYRjfCsQeGEcbHpFmwIoV+7Dm0Ax3RbHTrbC7qwVq+cWzz8a3aVivs51L7KDk/hfUgv7i9sTWUC/+Y","protoVersion":[0,17],"binVersion
```

NTS

The Extension



Burp Suite Professional v1.7.35 - Temporary Project - licensed to Checkpoint Ltd. [3 user license]
Burp Intruder Repeater Window Help Backslash
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Alerts Additional Scanner Checks JSON Beautifier Software Vulnerability Scanner CSRF Logger++ WhatsApp Decoder
Connect
Clear
Incoming
Encrypt
Decrypt
Outgoing

WHATSAPP DECRYPTION AND ENCRYPTION EXTENSION BY DIKLA BARDA, ROMAN ZAIKIN

Ref object: :_manufacturer":"samsung","device_model":"star2lte","os_build_number":"R16NW.G965FXXU1BRF8"},"pushname":"RomanZ","tos":0}

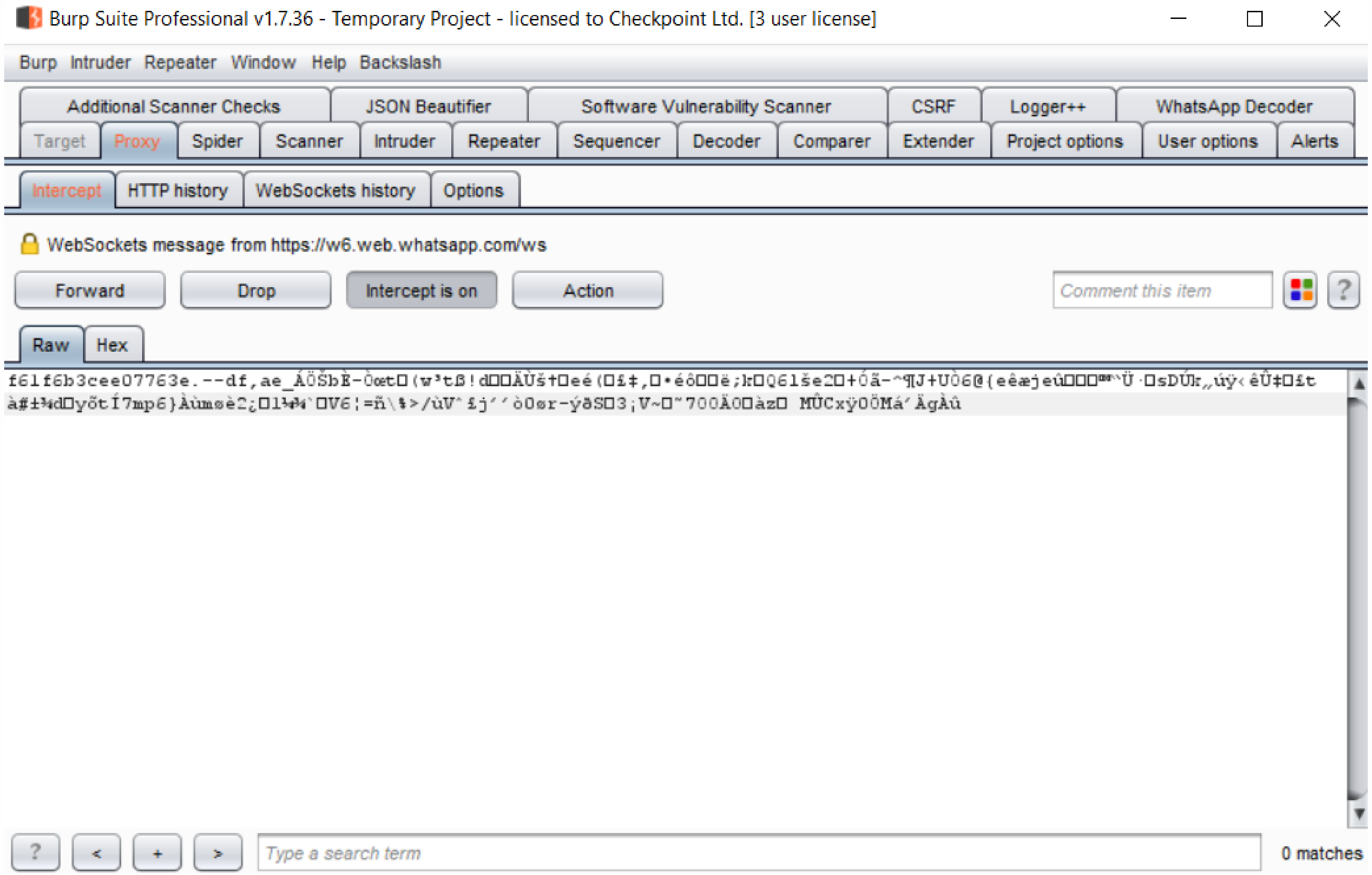
Private Key: 09, 206, 235, 93, 219, 38, 23, 2, 193, 102, 123, 7, 182, 74, 201, 130, 254, 26, 215, 123, 115, 223, 200, 185, 19, 61, 89, 106, 156, 111]

Public Key: 5, 229, 16, 179, 94, 246, 108, 29, 22, 106, 207, 209, 84, 186, 243, 144, 199, 30, 52, 52, 128, 207, 143, 197, 44, 167, 83, 180, 196, 37]

CONNECTION STATUS: CONNECTED

ACTION STATUS: OK

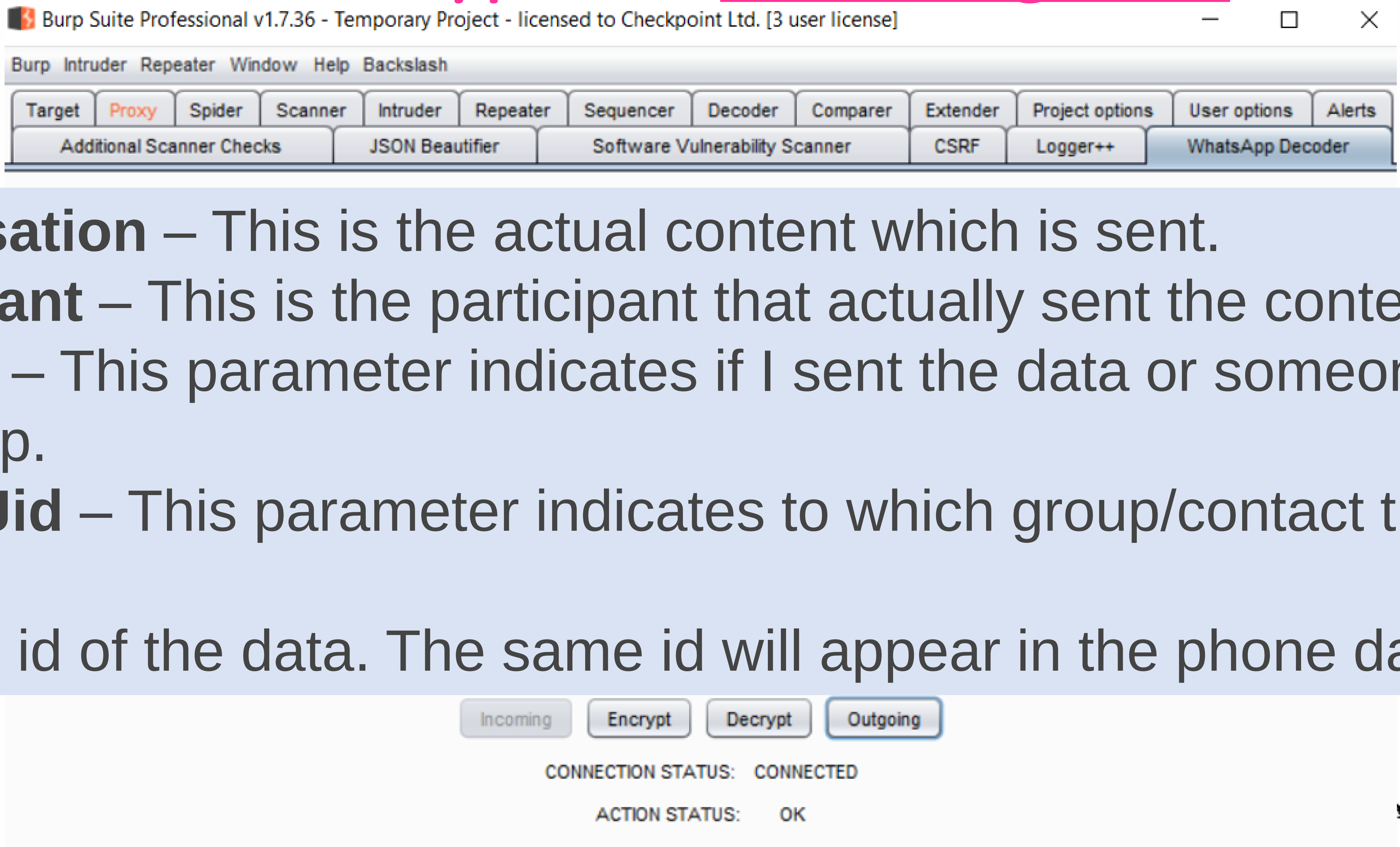
Decrypt the incoming data



WebSockets message from https://w6.web.whatsapp.com/ws

```
f61f6b3cee07763e.--df,ae_ÁÖŠbÈ-Òœt□(w³tß!d□□ÄÙš†□eé(□£‡,□•éô□□ë;k□Q61še2□+Óã-^¶J+UÒ6@(eêæjeû□□□ᵐᵐ`Ü·□sDÚr„úÿ‹êÛ‡□£t
à#±¾d□yõtÍ7mp6}Àùmsè2¿□1¼¼`□V6¦=ñ\‰>/ùV^£j´´ò0sr-ýðS□3;V~□˜700Ä0□àz□ MÛCxÿOÖMá´ÄgÀû
```

Decrypt the incoming data



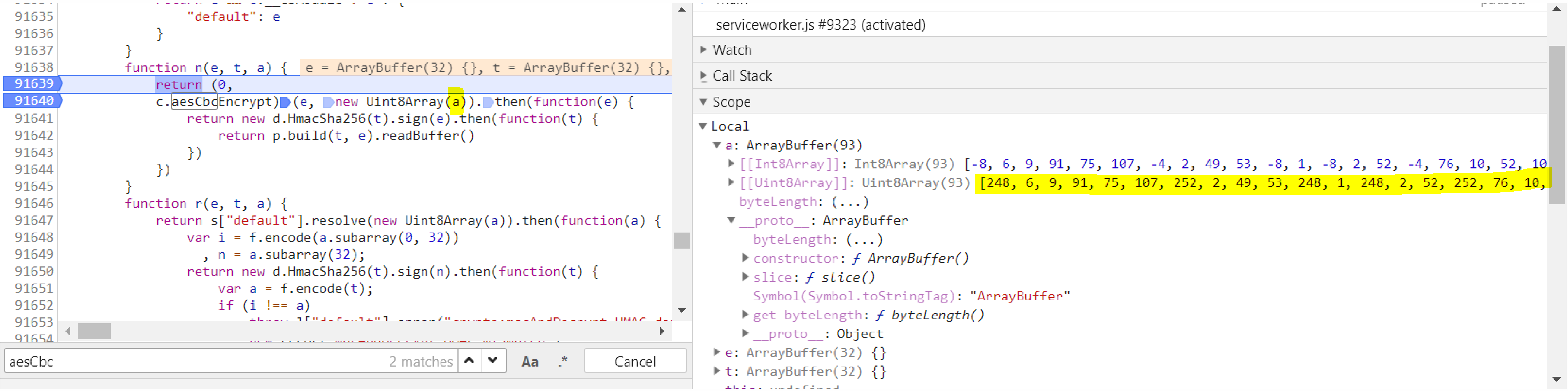
Burp Suite Professional v1.7.36 - Temporary Project - licensed to Checkpoint Ltd. [3 user license]

conversation – This is the actual content which is sent.
participant – This is the participant that actually sent the content.
fromMe – This parameter indicates if I sent the data or someone else in the group.
remoteJid – This parameter indicates to which group/contact the data is sent.
id – The id of the data. The same id will appear in the phone databases.

CONNECTION STATUS: CONNECTED

ACTION STATUS: OK

Decrypt the outgoing data

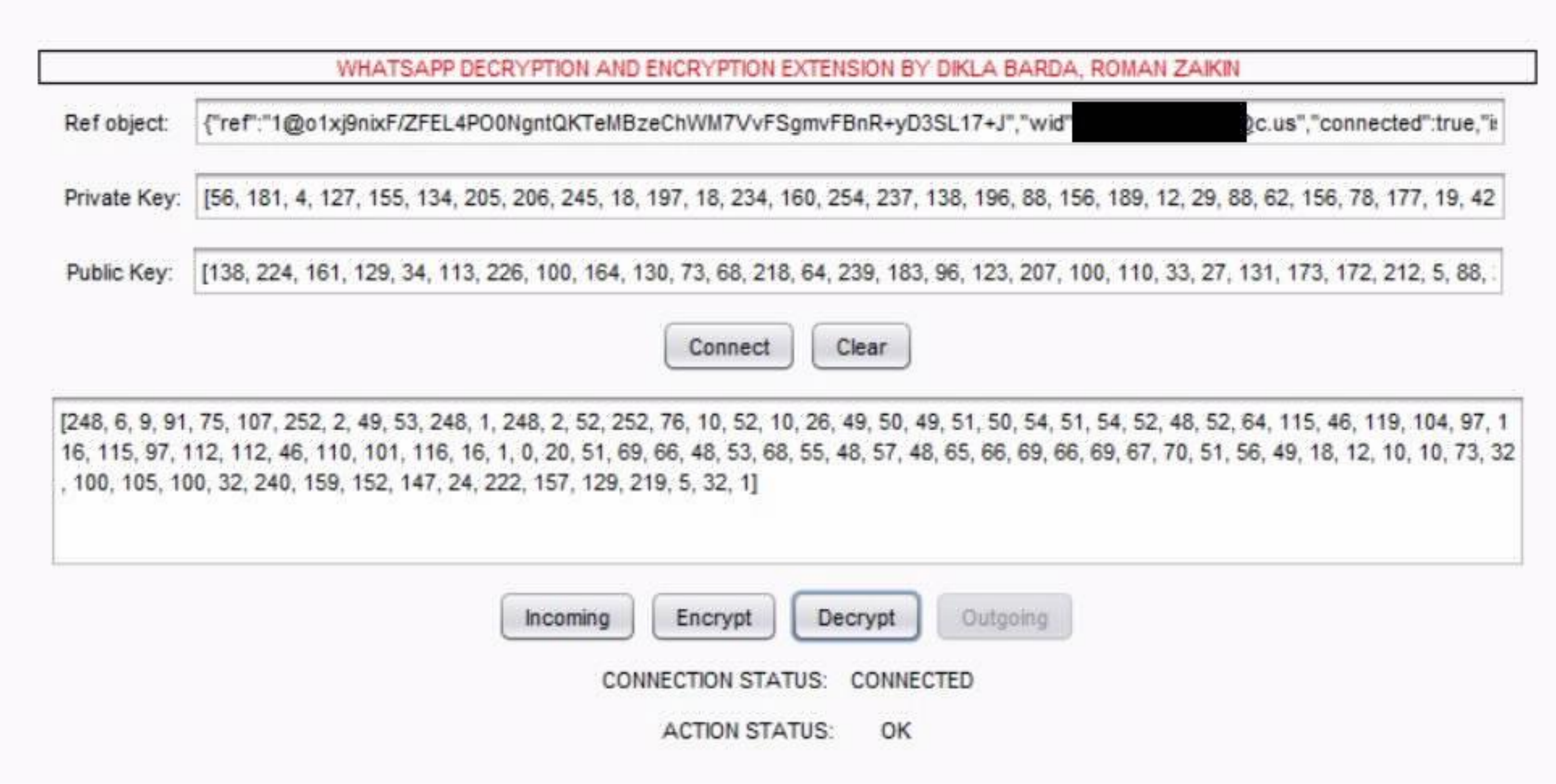


```
                "default": e
            }
        }
        function n(e, t, a) {  e = ArrayBuffer(32) {}, t = ArrayBuffer(32) {},
            return (0,
            c.aesCbcEncrypt)(e, new Uint8Array(a)).then(function(e) {
                return new d.HmacSha256(t).sign(e).then(function(t) {
                    return p.build(t, e).readBuffer()
                })
            })
        }
        function r(e, t, a) {
            return s["default"].resolve(new Uint8Array(a)).then(function(a) {
                var i = f.encode(a.subarray(0, 32))
                  , n = a.subarray(32);
                return new d.HmacSha256(t).sign(n).then(function(t) {
                    var a = f.encode(t);
                    if (i !== a)
```

serviceworker.js #9323 (activated)

```
▼ Local
  ▼ a: ArrayBuffer(93)
    ▶ [[Int8Array]]: Int8Array(93) [-8, 6, 9, 91, 75, 107, -4, 2, 49, 53, -8, 1, -8, 2, 52, -4, 76, 10, 52, 10
    ▶ [[Uint8Array]]: Uint8Array(93) [248, 6, 9, 91, 75, 107, 252, 2, 49, 53, 248, 1, 248, 2, 52, 252, 76, 10,
      byteLength: (...)
    ▼ __proto__: ArrayBuffer
        byteLength: (...)
      ▶ constructor: ƒ ArrayBuffer()
      ▶ slice: ƒ slice()
        Symbol(Symbol.toStringTag): "ArrayBuffer"
      ▶ get byteLength: ƒ byteLength()
      ▶ __proto__: Object
  ▶ e: ArrayBuffer(32) {}
  ▶ t: ArrayBuffer(32) {}
```

Decrypt the outgoing data



WHATSAPP DECRYPTION AND ENCRYPTION EXTENSION BY DIKLA BARDA, ROMAN ZAIKIN

Ref object: {"ref":"1@o1xj9nixF/ZFEL4PO0NgntQKTeMBzeChWM7VvFSgmvFBnR+yD3SL17+J","wid"[redacted]@c.us","connected":true,"i

Private Key: [56, 181, 4, 127, 155, 134, 205, 206, 245, 18, 197, 18, 234, 160, 254, 237, 138, 196, 88, 156, 189, 12, 29, 88, 62, 156, 78, 177, 19, 42

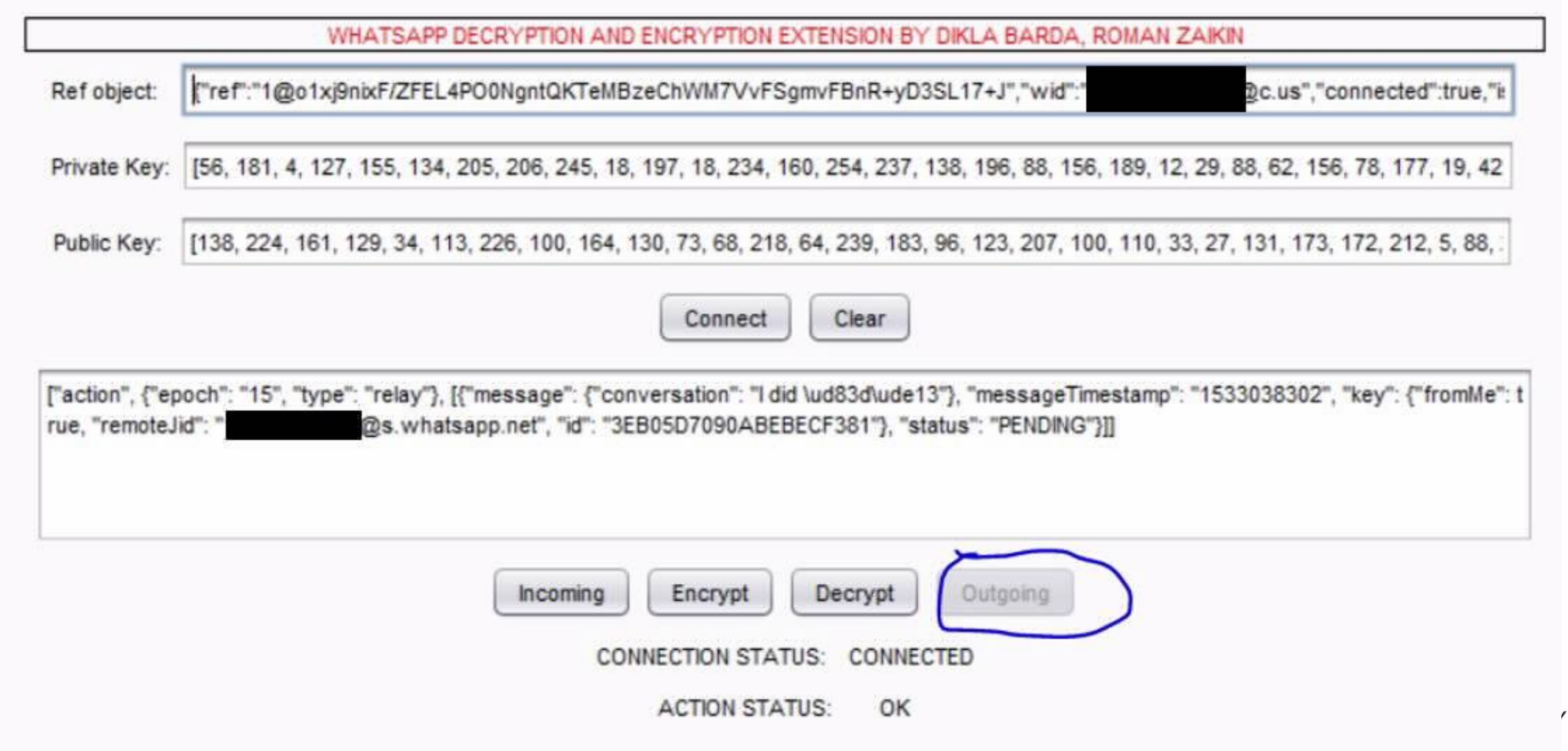
Public Key: [138, 224, 161, 129, 34, 113, 226, 100, 164, 130, 73, 68, 218, 64, 239, 183, 96, 123, 207, 100, 110, 33, 27, 131, 173, 172, 212, 5, 88,

```
[248, 6, 9, 91, 75, 107, 252, 2, 49, 53, 248, 1, 248, 2, 52, 252, 76, 10, 52, 10, 26, 49, 50, 49, 51, 50, 54, 51, 54, 52, 48, 52, 64, 115, 46, 119, 104, 97, 1
16, 115, 97, 112, 112, 46, 110, 101, 116, 16, 1, 0, 20, 51, 69, 66, 48, 53, 68, 55, 48, 57, 48, 65, 66, 69, 66, 69, 67, 70, 51, 56, 49, 18, 12, 10, 10, 73, 32
, 100, 105, 100, 32, 240, 159, 152, 147, 24, 222, 157, 129, 219, 5, 32, 1]
```

CONNECTION STATUS: CONNECTED

ACTION STATUS: OK

Decrypt the outgoing data



WHATSAPP DECRYPTION AND ENCRYPTION EXTENSION BY DIKLA BARDA, ROMAN ZAIKIN

Ref object: {"ref":"1@o1xj9nixF/ZFEL4PO0NgntQKTeMBzeChWM7VvFSgmvFBnR+yD3SL17+J","wid":"[redacted]@c.us","connected":true,"i

Private Key: [56, 181, 4, 127, 155, 134, 205, 206, 245, 18, 197, 18, 234, 160, 254, 237, 138, 196, 88, 156, 189, 12, 29, 88, 62, 156, 78, 177, 19, 42

Public Key: [138, 224, 161, 129, 34, 113, 226, 100, 164, 130, 73, 68, 218, 64, 239, 183, 96, 123, 207, 100, 110, 33, 27, 131, 173, 172, 212, 5, 88,

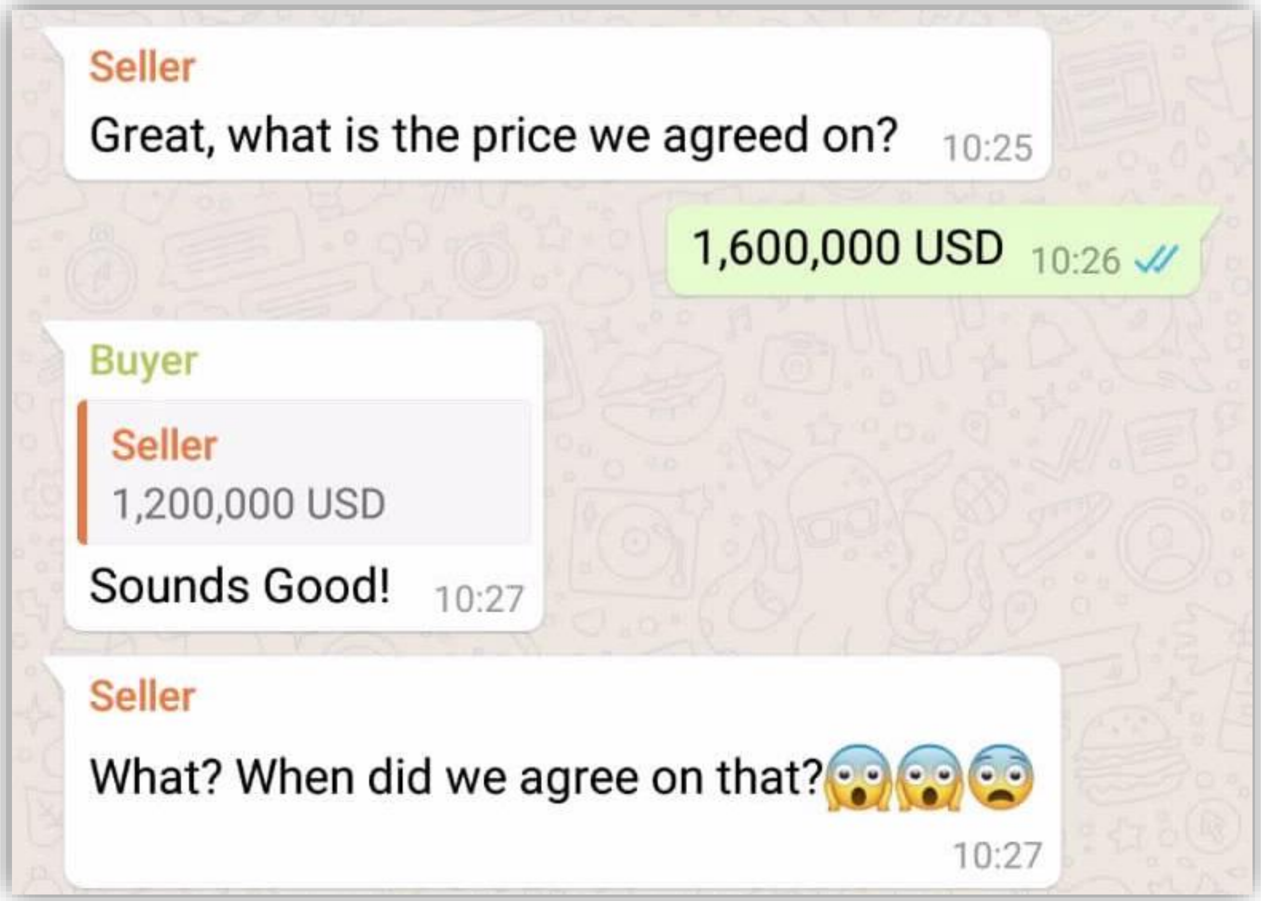
```
["action", {"epoch": "15", "type": "relay"}, [{"message": {"conversation": "I did \ud83d\ude13"}, "messageTimestamp": "1533038302", "key": {"fromMe": t
rue, "remoteJid": "[redacted]@s.whatsapp.net", "id": "3EB05D7090ABEBECF381"}, "status": "PENDING"}]]
```

CONNECTION STATUS: CONNECTED

ACTION STATUS: OK

DEMO

Manipulation #1 – fake reply from someone in the group



Seller
Great, what is the price we agreed on? 10:25

1,600,000 USD 10:26 ✓✓

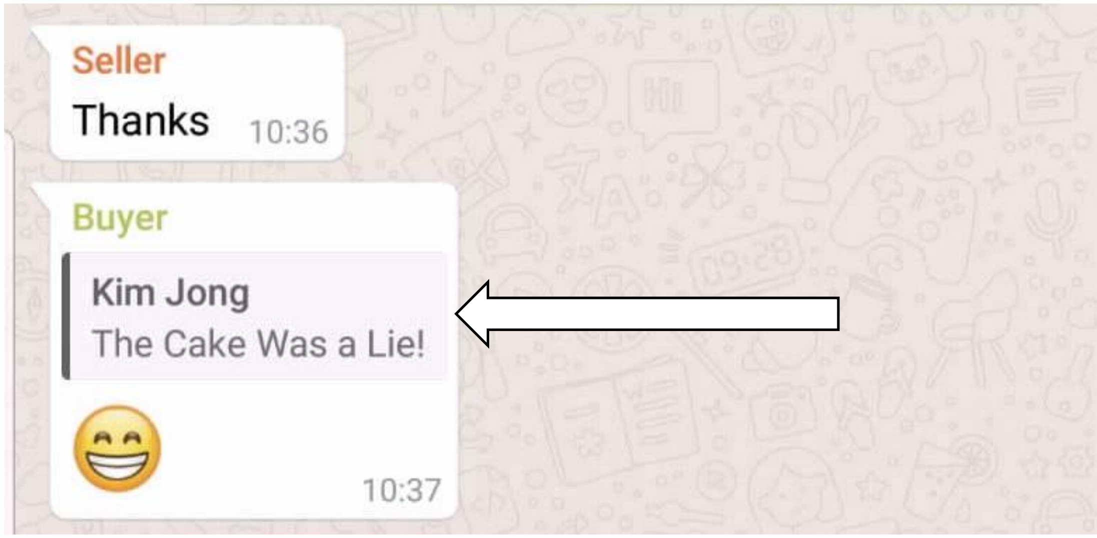
Buyer
> Seller
> 1,200,000 USD

Sounds Good! 10:27

Seller
What? When did we agree on that?😱😱😨 10:27

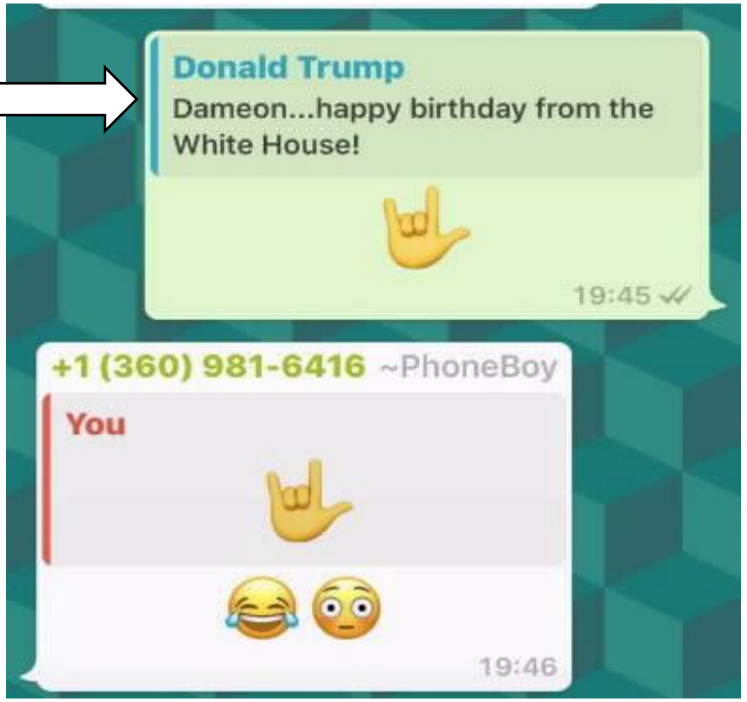
Demo

Manipulation #2 – Fake reply to someone not in the group



Seller
Thanks 10:36

Buyer
Kim Jong
The Cake Was a Lie!
😁 10:37



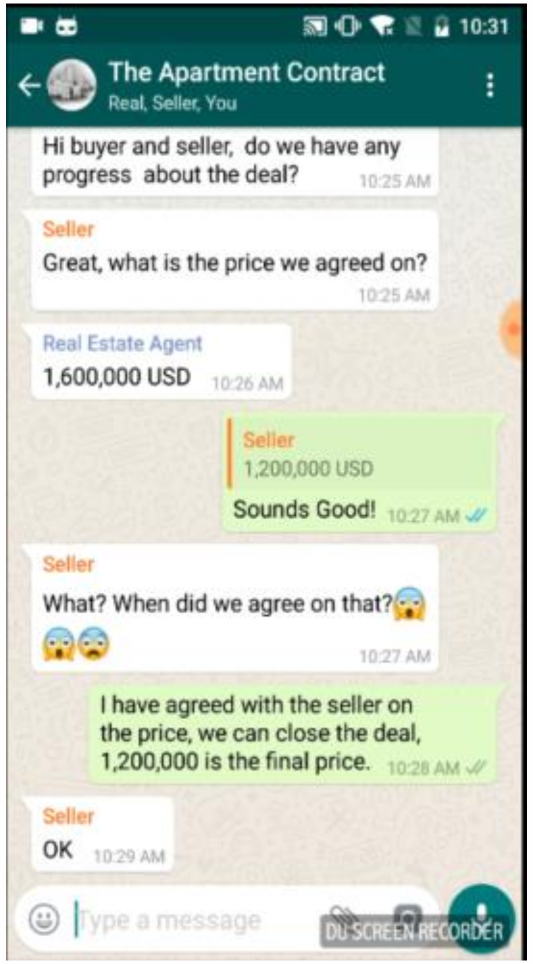
Donald Trump
Dameon...happy birthday from the White House!
🤟 19:45

+1 (360) 981-6416 ~PhoneBoy
You
🤟
😂😳 19:46

Demo

Manipulation #3 – Send a private message in group chat to a specific person

Attacker



10:31

The Apartment Contract
Real, Seller, You

Hi buyer and seller, do we have any progress about the deal? 10:25 AM

Seller
Great, what is the price we agreed on? 10:25 AM

Real Estate Agent
1,600,000 USD 10:26 AM

> Seller
> 1,200,000 USD

Sounds Good! 10:27 AM

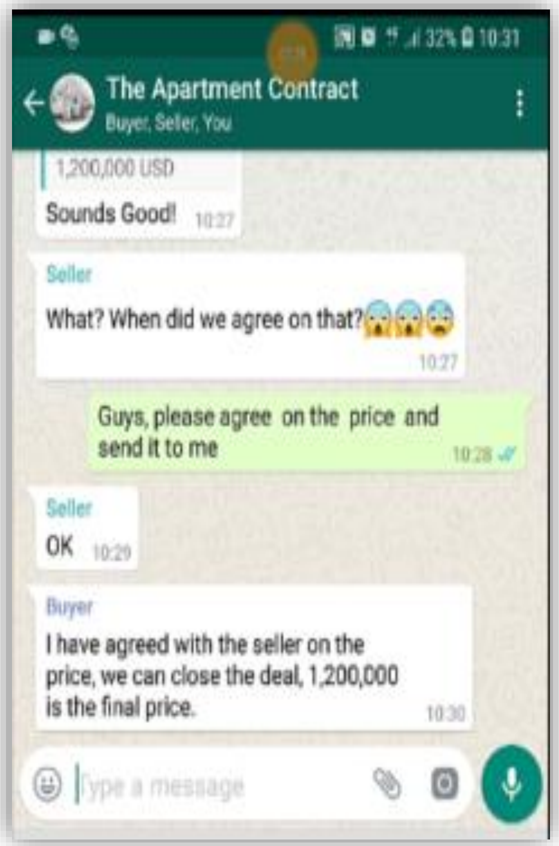
Seller
What? When did we agree on that?😱😱😨 10:27 AM

I have agreed with the seller on the price, we can close the deal, 1,200,000 is the final price. 10:28 AM

Seller
OK 10:29 AM

Type a message

User 1



32% 10:31

The Apartment Contract
Buyer, Seller, You

> 1,200,000 USD

Sounds Good! 10:27

Seller
What? When did we agree on that?😱😱😨 10:27

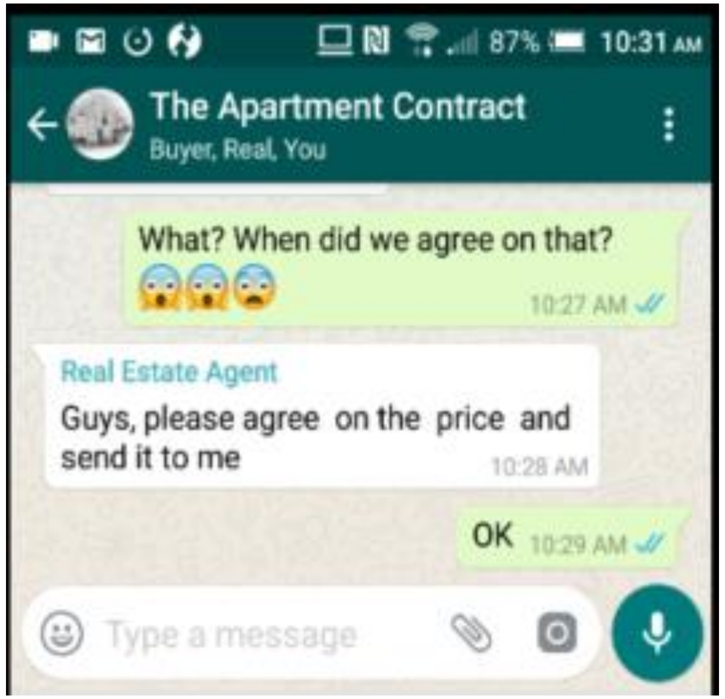
Guys, please agree on the price and send it to me 10:28

Seller
OK 10:29

Buyer
I have agreed with the seller on the price, we can close the deal, 1,200,000 is the final price. 10:30

Type a message

User 2



87% 10:31 AM

The Apartment Contract
Buyer, Real, You

What? When did we agree on that? 😱😱😨 10:27 AM

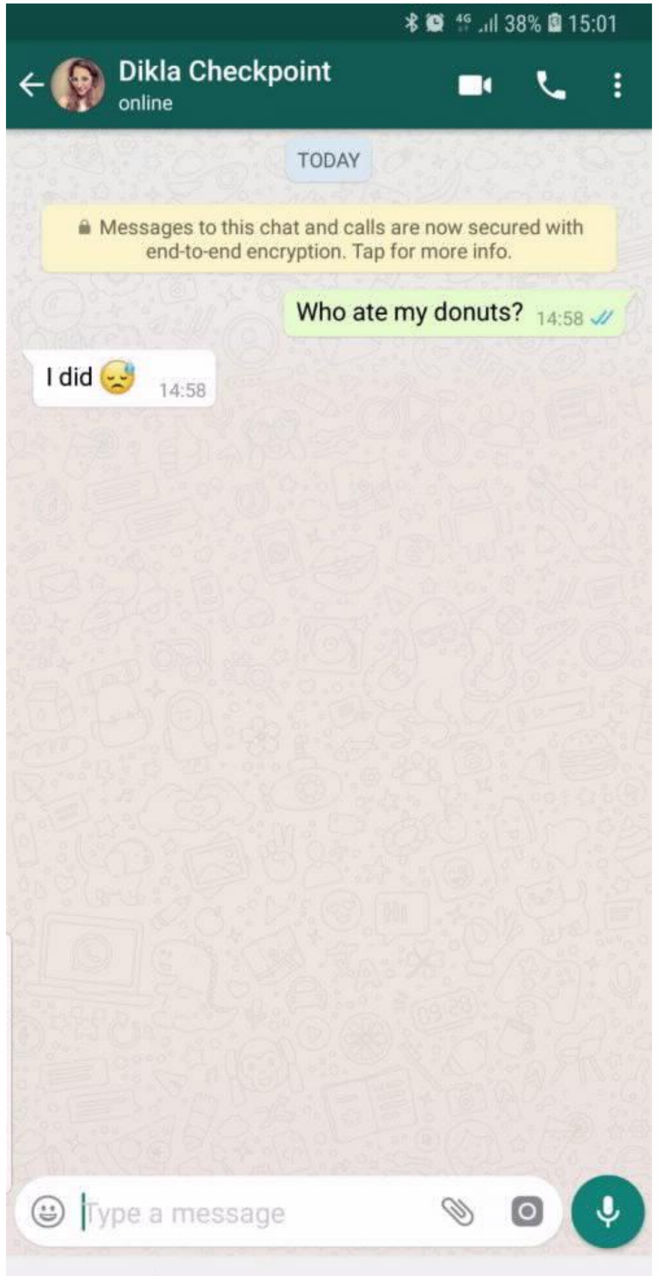
Real Estate Agent
Guys, please agree on the price and send it to me 10:28 AM

OK 10:29 AM

Type a message

[Demo](#)

Manipulation #4: send messages to myself



15:01 38%

Dikla Checkpoint
online

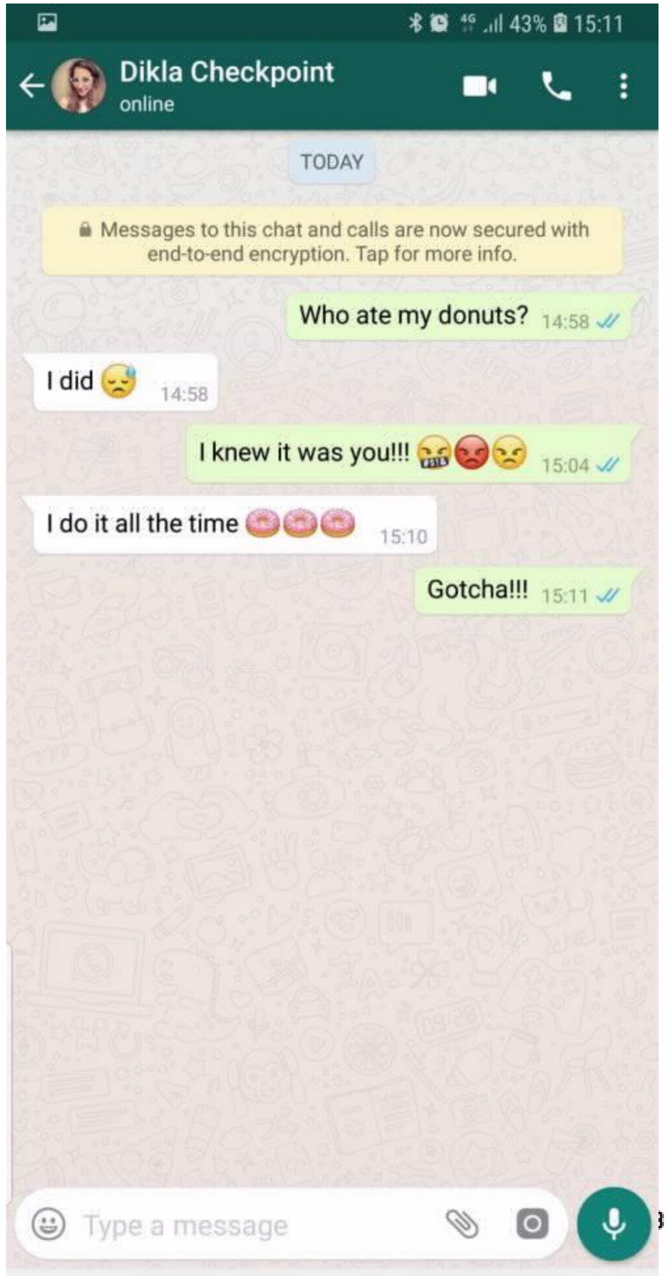
TODAY

Messages to this chat and calls are now secured with end-to-end encryption. Tap for more info.

Who ate my donuts? 14:58

I did 😓 14:58

Type a message



15:11 43%

Dikla Checkpoint
online

TODAY

Messages to this chat and calls are now secured with end-to-end encryption. Tap for more info.

Who ate my donuts? 14:58

I did 😓 14:58

I knew it was you!!! 🤬😡😠 15:04

I do it all the time 🍩🍩🍩 15:10

Gotcha!!! 15:11

Type a message

Demo

THANK YOU!