ö-ö.net: einfach sicher chatten (crypto-secured HTML5 p2p-live-chat client)

Andreas Briese

Eine Unternehmung der eduToolbox@Bri-C GmbH, Sarstedt

25. März 2014

Index

einfach
sicher
chatten

http(s)://ö-ö.net

first client connect (Alice)

Alice GET https://xn- - - -0gab.net/chat [1]

ö-ö → ö-ö webApp: ≈150kb html/css/js/img/audio

Alice Math.random() = seedrandom() ← Salsa20/20 PRNG
seeded with 256 bytes from (browser)crypto.random()
upgrade http(s) → websocket ws(s)

ö-ö js to eval: send sha256(webApp script-parts at Alice)

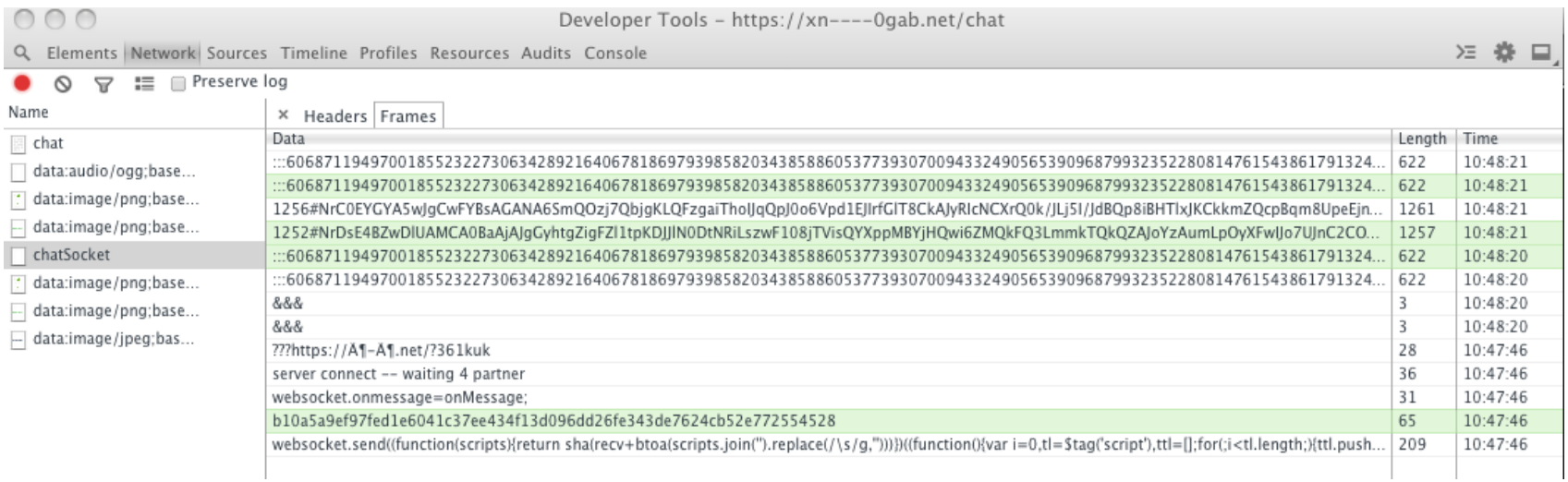
Alice sha256(script-parts at Alice) →

ö-ö ? sha256(delivered scripts) === sha256(script-parts at Alice)
NO → ws(s) close / del from register
YES → enable ws(s)-receive function
YES → "server connect – waiting 4 partner"
YES → URL to connect with Alice

[1](punicode for **ö-ö.net**: xn- - - -0gab.net)

Alice
websocket frames

(opera 20 developer tools)



connect: last five lines in grafik
white: incoming ; green: outgoing

connect of Alice partner (Bob)

Bob GET https://xn- - - -0gab.net/?Alice

ö-ö ? (Alice in websocket register)

NO → "ws is not registered"

YES (Alice ⇔ Bob) ↓ open-register: rm Alice

YES ö-ö webApp → Bob

Bob Math.random() = seedrandom() ← Salsa20/20 PRNG

seeded with 256 bytes from (browser)crypto.random()

upgrade http(s) → websocket ws(s)

ö-ö js to eval: send sha256(webApp script-parts at Bob)

Bob sha256(script-parts at Bob) →

ö-ö ? sha256(delivered scripts) === sha256(script-parts at Bob)

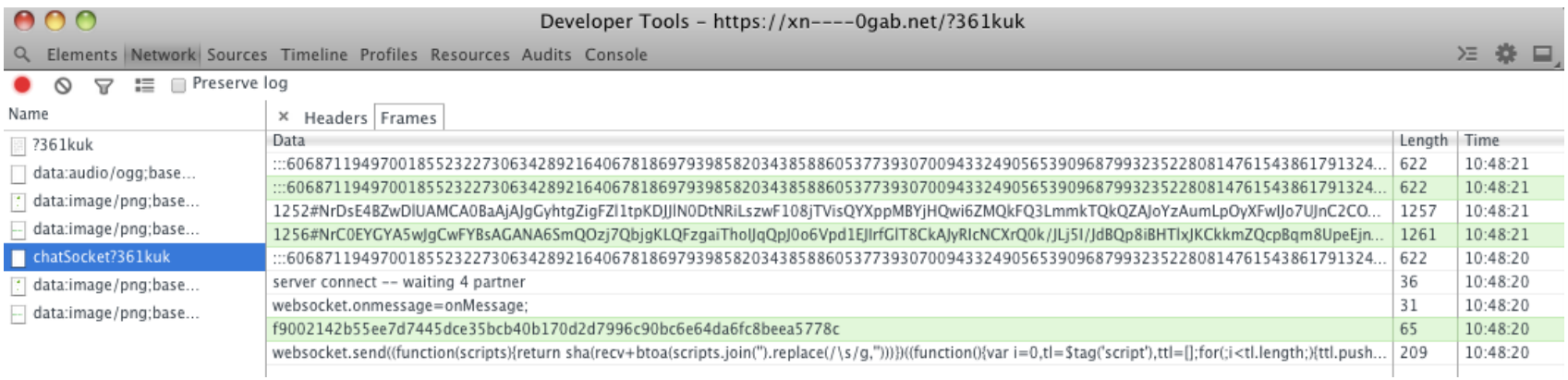
NO → ws(s) close / del from register

YES → enable ws(s)-receive function

YES → "server connect – waiting 4 partner"

Bob
websocket frames

(opera 20 developer tools)



connect: last four lines in grafik
white: incoming ; green: outgoing

... Alice & Bob connected and approved ...

ö-ö $\text{rand}_{1..1000}$ 2056bit save prime (g=2) $\rightarrow$ Alice

Alice 2056bit save prime $\rightarrow$ Bob

Bob ?(prime bitsize===2056) $\rightarrow$ calc Bob publKey

Alice ?(prime bitsize===2056) $\rightarrow$ calc Alice publKey

Bob $\text{RABBIT}_{sha(siteKey)+sha(prime+recv)}$(publKey) $\rightarrow$ Alice

Alice $\text{RABBIT}_{sha(siteKey)+sha(prime+recv)}$(publKey) $\rightarrow$ Bob

Bob ? (publKey bitsize $>$ 370 && publKey $<$ prime-2)

NO $\rightarrow$ kill session

YES $\rightarrow$ calc sessKey/cipherKey

$\Rightarrow$ prime $\rightarrow$ Alice

Alice ? (publKey bitsize $>$ 370 && publKey $<$ prime-2)

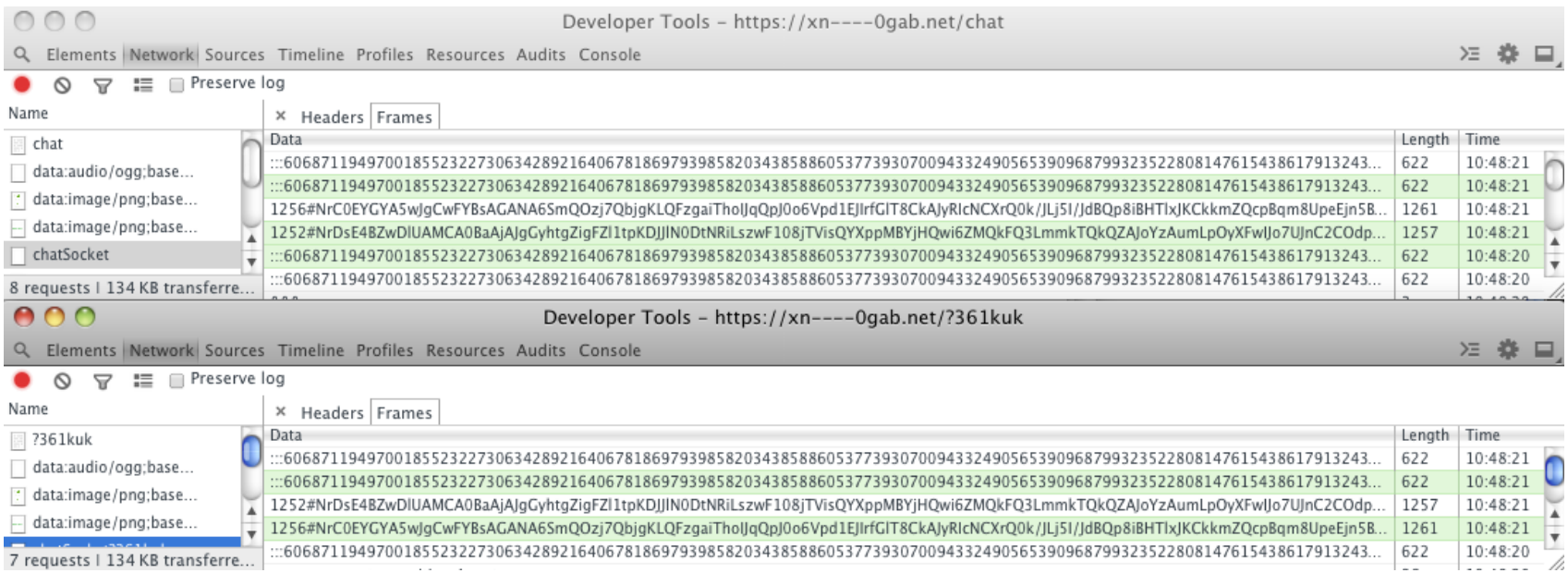
NO $\rightarrow$ kill session

YES $\rightarrow$ calc sessKey/cipherKey

$\Rightarrow$ prime $\rightarrow$ Bob

Alice (top) & Bob
Diffie-Hellman-Handshake websocket frames

(opera 20 developer tools)



white: incoming; green: outgoing

function dhKeys(sessPrime)[2] [3]

(github.com/AndreasBriese/oe-oe.net/oeoe.js)

```
672  function dhKeys(sPrime){
673      if(!!warten()){
674          try{
675              sessPrime = str2bigInt(sPrime, 10, 200);
676              if(bitSize(sessPrime)<2056){
677                  if(websocket.readyState===websocket.OPEN){
678                      websocket.send('!!!DISCONNECT!');
679                      websocket.close();
680                      alert('illegal DH param')
681                  }
682              }
683              sessSecret = int2bigInt(0, 1);
684              for(;(bitSize(sessSecret)<370&&(greater(sessPrime, sub(sessSecret,int2bigInt(2, 1)))));){
685                  sessSecret = randBigInt(primeBits, 0);
686              }
687              var sessGen = powMod(int2bigInt(2, 1), sessSecret, sessPrime);
688              publicKey = "---" + bigInt2str(sessGen, 10);
689              // console.log(sPrime, publicKey);
690              return true;
691          }catch(e){
692              throw e;
693              return false;
694          }
695      }
696  }
```

[2] algorithm from http://tools.ietf.org/html/rfc4253section-8

[3] www.bsi.bund.de "Kryptographische Verfahren: Empfehlungen und Schluessellaengen" BSI TR-02102-1

function dhExchange(publKey)[4] [5]

(github.com/AndreasBriese/oe-oe.net/oeoe.js)

```
698  function dhExchge(pKey){
699      var partnerKey = str2bigInt(pKey, 10, 200);
700      if(greater(int2bigInt(2, 1), partnerKey) || greater(partnerKey, sub(sessPrime, int2bigInt(2, 1)))){
701          if(websocket.readyState===websocket.OPEN){
702              websocket.send('!!!DISCONNECT!');
703              websocket.close()
704          }
705          alert("Unsichere Schluesselsequenz! Sitzung wird abgebrochen!");
706      }
707      cipherKey = powMod(partnerKey, sessSecret, sessPrime);
708      sessKey = sha(bigInt2str(cipherKey, 10));
709      //console.log("ourSecret:", cipherKey);
710      var l = cipherKey.length-2;
711      var ba = new ArrayBuffer(2*l);
712      var ab = new Uint16Array(ba);
713      for(;l;ab[--l]=cipherKey[l]){}
714      cipherKey = new Uint8Array(ba);
715      //console.log("compare:", cipherKey, ab);
716      sessSecret = null;
717      websocket.send(":::"+bigInt2str(sessPrime,10));
718      if(!cllr) nickPopUP();
719      return true;
720  }
```

[4] algorithm from http://tools.ietf.org/html/rfc4253section-8

[5] www.bsi.bund.de "Kryptographische Verfahren: Empfehlungen und Schluessellaengen" BSI TR-02102-1

text-encryption using CryptoJS.rabbit

* CryptoJS v3.1.2
* code.google.com/p/crypto-js
* (c) 2009-2013 by Jeff Mott. All rights reserved.
* code.google.com/p/crypto-js/wiki/License

usage	text-en/decryption with encryption-secret ← sKey/IKey
sKey	sKey = sha256(last incoming message number + salt from last incoming textmessage) + sessKey
IKey	IKey = sha256(outgoing message number + salt from outgoing textmessage) + sesskey

textmessage
message format

length	separator	encrypted payload (compressed & base64 encoded string)
1260	#	NrCME4CYGYBYFZKQGywDSoOy..

Salsa20/20 stream-encryption

* Salsa 20
* unified from http://code.ohloh.net/file?fid=v-0xBPS2xdNtbDPzjFZXCY89n0Acid=vPv6IIgR6co
* Contribution to Cryptocat by Dmitry Chestnykh 21-01-2013

* Snippet had been analysed and tested against D.J.Bernsteins original C-Implementation
* at http://cr.yp.to/snuffle.html (http://cr.yp.to/snuffle/salsa20/ref/salsa20.c)
* ABriese 2014/20/02

Salsa20/20 stream-encryption (ff.)

send-protokoll:

1. calc cipherKey start index: $\text{rand}_{0\ldots256}$
2. seed salsa20 $\leftarrow$ 32 bytes / 8 bytes (256 bits) from the cipherKey (30 bytes of the cipherKey reserved and never used)
3. file $\rightarrow$ blob
4. XOR each byte against 8bit from salsa20 PRNG
5. compose sendfile
6. sendfile $\rightarrow$ POST https://ö-ö.net/upload

ö-ö incoming bytes $\Rightarrow$ receiver-websocket

file transmission
sendfile message format

→ POST https://ö-ö.net/upload

65 bytes plain text	1 byte num	120 bytes	70 bytes	file blob data
		Salsa20/20 encrypted payload		
addressee	start index	file name	file format	blob
vh7jcv	212	encrypted payload		

server checks
messenger websocket responding (ping)

The server checks responsiveness of open (but unconnected) websockets by sending: "&&&" (3 Bytes)