

# SSL logging for proof

Why?

Money hardness scale:

BTC 10 (+!)

Physical cash - 10

Wire transfer over SWIFT – 10 (?)

OK Pay – 10(?)

SEPA, ACH – 5-8?

Paypal, credit cards – 0

So – swap BTC for wire transfers or OK Pay, or maybe “softer” methods like SEPA if small amounts.

Localbitcoins – peer to peer but drawbacks:

Safety/counterfeiting and physical inconvenience if in person for cash

Quasi-centralisation and lack of effective proof if online for wire transfer  
(Quality of dispute arbitration is dubious)

Other “Whys” - there can be a few other ways to use this technology.  
These slides only talk about trading BTC.

# SSL logging for proof

Record ssl session between bank and customer (\*NOT\* MITM)

If there's a dispute, the customer can prove the payment was made by providing decryption key(s) to an arbitrator.

Privacy issues:

It's possible to break the SSL connection to the bank and then rebuild it. Thus, the customer only provides the decryption for 1 or 2 HTML pages containing proof of payment, not the whole banking session. So login information need not be exposed.

# Network trace

Wireshark – open source, extremely rich, long running industry standard (originally “Ethereal”).

Command line version “tshark”.

Recently added facility to decrypt ssl sessions if given master secret

Mozilla NSS library recently added facility to record master secret via an env variable SSLKEYLOGFILE – works with Firefox and maybe Chrome.

Network traces recorded with Wireshark are flaky w.r.t. ssl decryption (issues with e.g. out of order packets).

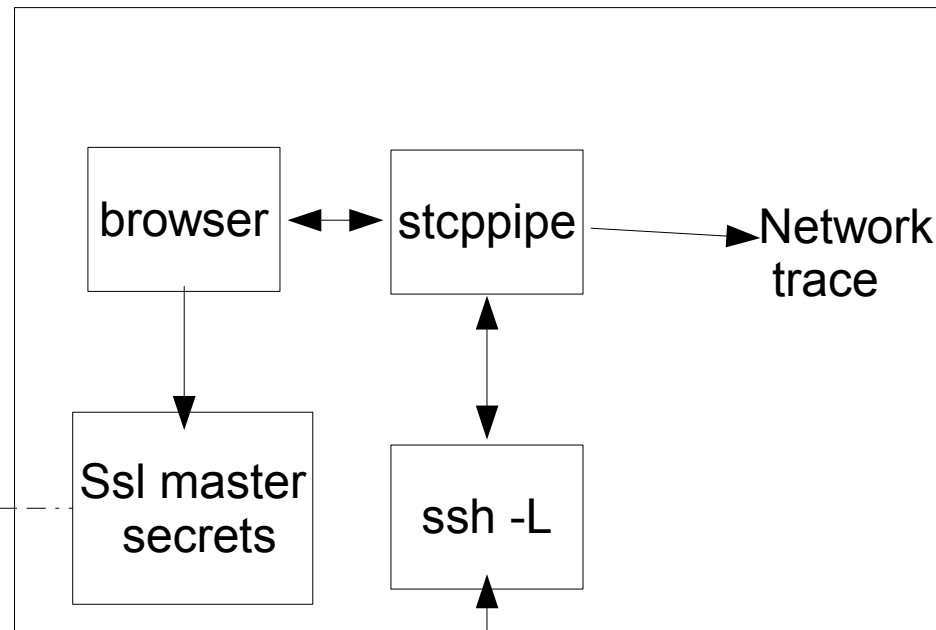
Much better results using traces recorded with stcppipe by Luigi Ariemma - “simple tcp pipe”.

All of this cross platform and open source.

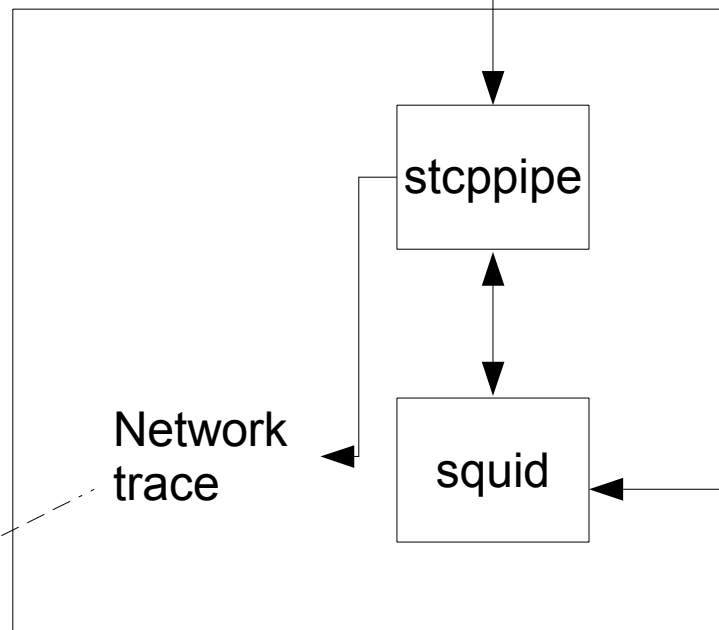
Side note: Wireshark doesn't seem to be able to decrypt Diffie Helman ciphers, although it should be able to in theory (perfect forward secrecy shouldn't affect this).

Camellia ciphers also appear not to work. These can be switched off in FF prefs.

Buyer machine



Oracle machine



Escrow agent reviews in dispute

We have tested this setup with several bank websites and verified it works.



# Wireshark - decryption

merged.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: ssl and http Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Destination Port	Source Port	Info
166	39.476588000	127.0.0.1	127.0.0.1	HTTP	624	33309	49677	GET /online365/spring/resource/org/richfaces/renderkit/
188	39.698601000	127.0.0.1	127.0.0.1	HTTP	535	49677	33309	HTTP/1.1 200 OK (text/css)
189	39.699601000	127.0.0.1	127.0.0.1	HTTP	592	33309	49677	GET /online365/spring/resource/org/richfaces/renderkit/
230	40.160627000	127.0.0.1	127.0.0.1	HTTP	640	33309	49683	GET /online365/spring/resource/org/richfaces/renderkit/
231	40.161627000	127.0.0.1	127.0.0.1	HTTP	576	33309	49684	GET /online365/spring/resource/org/ajax4jsf/javascript.
236	40.208630000	127.0.0.1	127.0.0.1	HTTP	576	33309	49686	GET /online365/spring/resource/org/ajax4jsf/javascript.
239	40.210630000	127.0.0.1	127.0.0.1	HTTP	592	33309	49687	GET /online365/spring/resource/org/richfaces/renderkit/
240	40.210630000	127.0.0.1	127.0.0.1	HTTP	576	33309	49685	GET /online365/spring/resource/org/ajax4jsf/javascript/
241	40.256633000	127.0.0.1	127.0.0.1	HTTP	976	49677	33309	HTTP/1.1 200 OK (text/javascript)
242	40.257633000	127.0.0.1	127.0.0.1	HTTP	592	33309	49677	GET /online365/spring/resource/org/richfaces/renderkit/
248	40.437643000	127.0.0.1	127.0.0.1	HTTP	1499	49685	33309	HTTP/1.1 200 OK (text/javascript)

Frame 63: 1022 bytes on wire (8176 bits), 1022 bytes captured (8176 bits) on interface 0

Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)

Transmission Control Protocol, Src Port: 33309 (33309), Dst Port: 49657 (49657), Seq: 8712, Ack: 2460, Len: 968

Secure Sockets Layer

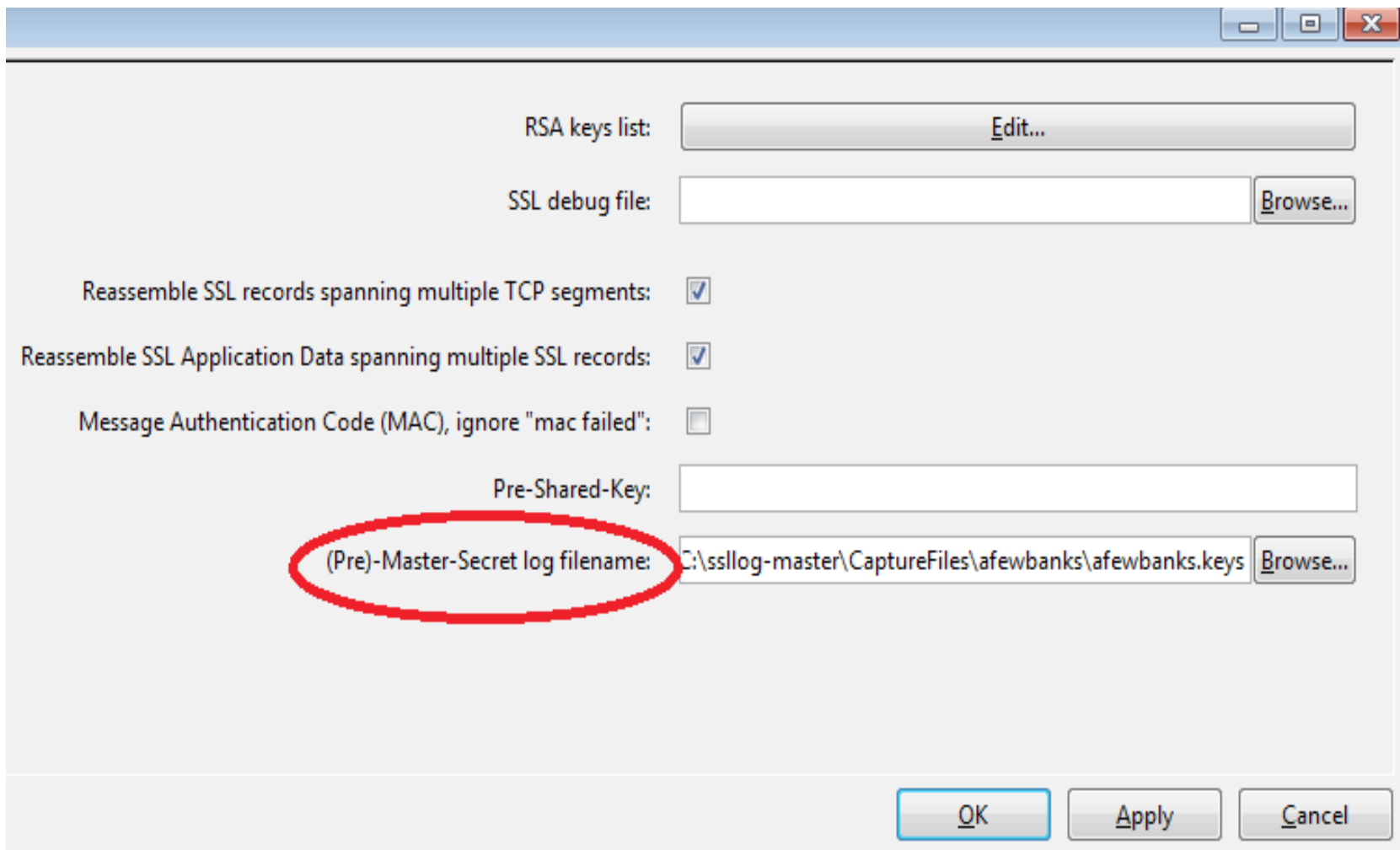
Hypertext Transfer Protocol

Line-based text data: text/html

0000 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK.  
0010 0a 63 6f 6e 74 65 6e 74 2d 6c 65 6e 67 74 68 3a .content-length:  
0020 20 36 37 30 0d 0a 63 6f 6e 74 65 6e 74 2d 74 79 670..content-ty  
0030 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a 64 pe: text /html..d  
0040 61 74 65 3a 20 46 72 69 2c 20 32 30 20 53 65 70 ate: Fri, 20 Sep  
0050 20 32 30 31 33 20 30 39 3a 31 38 3a 30 31 20 47 2013 09 :18:01 G  
0060 4d 54 0d 0a 70 33 70 3a 20 43 50 3d 22 4e 4f 4e MT..p3p: CP="NON  
0070 20 43 55 52 20 4f 54 50 69 20 4f 55 52 20 4e 4f CUR OTP i OUR NO  
0080 52 20 55 4e 49 22 0d 0a 63 61 63 68 65 2d 63 6f R UNI".. cache-co  
0090 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 68 65 0d ntrol: n o-cache.  
00a0 0a 70 72 61 67 6d 61 3a 20 6e 6f 2d 63 61 63 68 .pragma: no-cach  
00b0 65 0d 0a 53 65 74 2d 43 6f 6f 6b 69 65 3a 20 50 e..Set-Cookie: P  
00c0 44 2d 53 2d 53 45 53 53 49 4f 4e 2d 49 44 3d 32 D-S-SESS ION-ID=2  
00d0 5f 46 4e 4b 52 55 53 6e 65 4f 72 76 5a 57 45 78 \_FNKRUSn eorvZWEx  
00e0 36 52 7a 75 70 75 54 70 4f 4b 2b 53 48 49 4c 6a 6RzupuTp OK+SHILj  
00f0 59 34 57 63 68 34 2d 53 75 5a 6b 4c 41 55 71 6a Y4wc iT-U uZkLAUqj

Frame (1022 bytes) Decrypted SSL data (947 bytes)

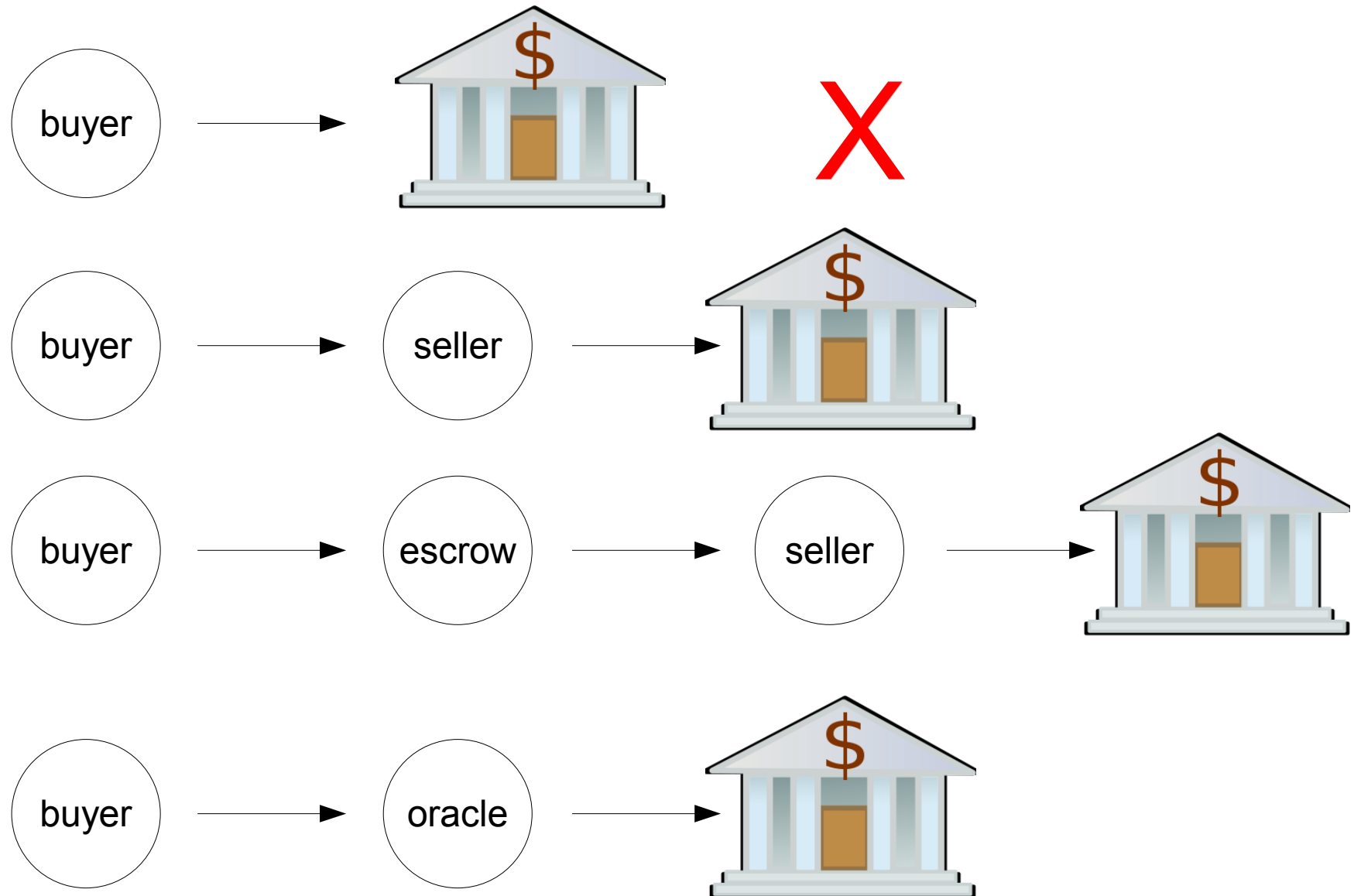
File: "C:\ssllog-master\CaptureFiles\cf5.vb8..." Profile: Default



Example usage:

```
>>tshark -r /path/to/tracefile -Y \  
"ssl and http.content_type contains html" \  
-o ssl.keylog_file:/path/to/keylogfile \  
-T fields -e frame.number
```

# Architectures



# Step by step

- Whichever exact model is chosen, the basic structure would be:
- Buyer and seller negotiate a TX
- 2 of 3 multisig address is created and funded by seller - 3<sup>rd</sup> key belongs to escrow agent
- Seller funds multisig
- Buyer does banking session
- If no dispute, buyer and seller send TX to network
- If dispute, escrow agent requests ssl key for the 1 or 2 HTML pages constituting proof of payment
- Escrow can make a valid decision and sign a transaction either to send to buyer or seller



# Issues

- NAT traversal for buyer-seller direct connection; would need proper P2P network
- In order to arbitrate, the escrow needs to know who is lying if the network trace can't be decrypted; that rules out simple seller-as-proxy model
- Should we run this complex auditing process **during** the bank payment or **after** – only in case of dispute? Not a trivial question!
- Oracle implemented in the cloud so that it's known what code is running on it – so it can be trusted. BUT – we can't have tons of bank session traffic coming just from Amazon AWS IPs (geographical and other restrictions)

# New ideas

- Master secret is partitioned into 4 16 byte blocks – one block each for encryption key and mac key (for message authentication), one pair for client, one for server (TLS RFC 6.3)
- This means we can have the buyer able to decrypt/encrypt but not able to authenticate – we just need to withhold the server mac key and disable the mac check.
- This would provide another party the ability to verify the validity of the buyer's traffic after the event.
- This is a very powerful idea, although it does have security implications