Peer-to-Peer Encryption and Authentication from the Perspective of End-Users

jonasschnelli

dev@jonasschnelli.ch

Simple example how difficult it is to be **aware** of end-user issues…

Who of you are using a thin client (**SPV wallet**) on your **smartphone**?

Who of you on a **iOS device**?
Have you **self-compiled** and
self-installed the binary?

There is **no verifiable security-proof** or a link between an app store binary (or an app update) and a specific state of the code (git commit) iOS applications can’t be „**hashed**“ or **verified**

private key storage

mixed with

auto-update

mixed with

app sandboxing

results in

„questionable“ trust and security model

VAULT DOOR
WEIGHT: 22 1/2 Tons
THICKNESS: 22 Inches
STEEL: 11 Layers of Special Cutting and Drill Resistant
LOCKS: 4 Hamilton Watch Movements for Time Locks

The door to the walk-in vault in the Winona Savings Bank in Winona, Minnesota, United States
CC BY-SA 3.0

Lets assume **10’000 users** with a avg. wallet value of **1’000 $**.

This results in an attack-bounty of **10 million $.**

But wait!
We have **code-signing**?

Well,… for what purpose exactly?

Bitcoin scaling focuses mostly on the **core infrastructure**, often leaving out the **end users perspective.**

Running a **full node wallet** is currently **„extremely difficult“** for the novice end-user.

Decentralized SPV wallets are only working on smartphones because we have hundreds of full node operators providing CPU&HDD intensive **free-of-charge services.**

Plus. Almost all „light-clients“ do **leak private data** (due to bloom filtering).

SMTP analogy

- It is relatively complex to run your own Mail Server
- Most „Mail“ users are no longer directly using SMTP
- SMTP has been extended to death and could probably seen as a dying out protocol.
- Most novice end-users are using a centralized hosted mail solution nowadays (resulting in various privacy and security issued)
- Encryption has never been made it to an „industry standard“

Make Bitcoin Great Again!

Amount of economical independent (full) **nodes** is **declining**

Missing option to **securely connect peers** (Connect your thin-client with a trusted full node)

Missing standard for hardware wallets resulting in leaving iOS in the dark.

End users are more and more using **2nd layer applications** not directly connected with the p2p network.

Missing standards or communication channels for **Multisig**.

End users haven't shifted to the „be-your-own-bank" security mindset.

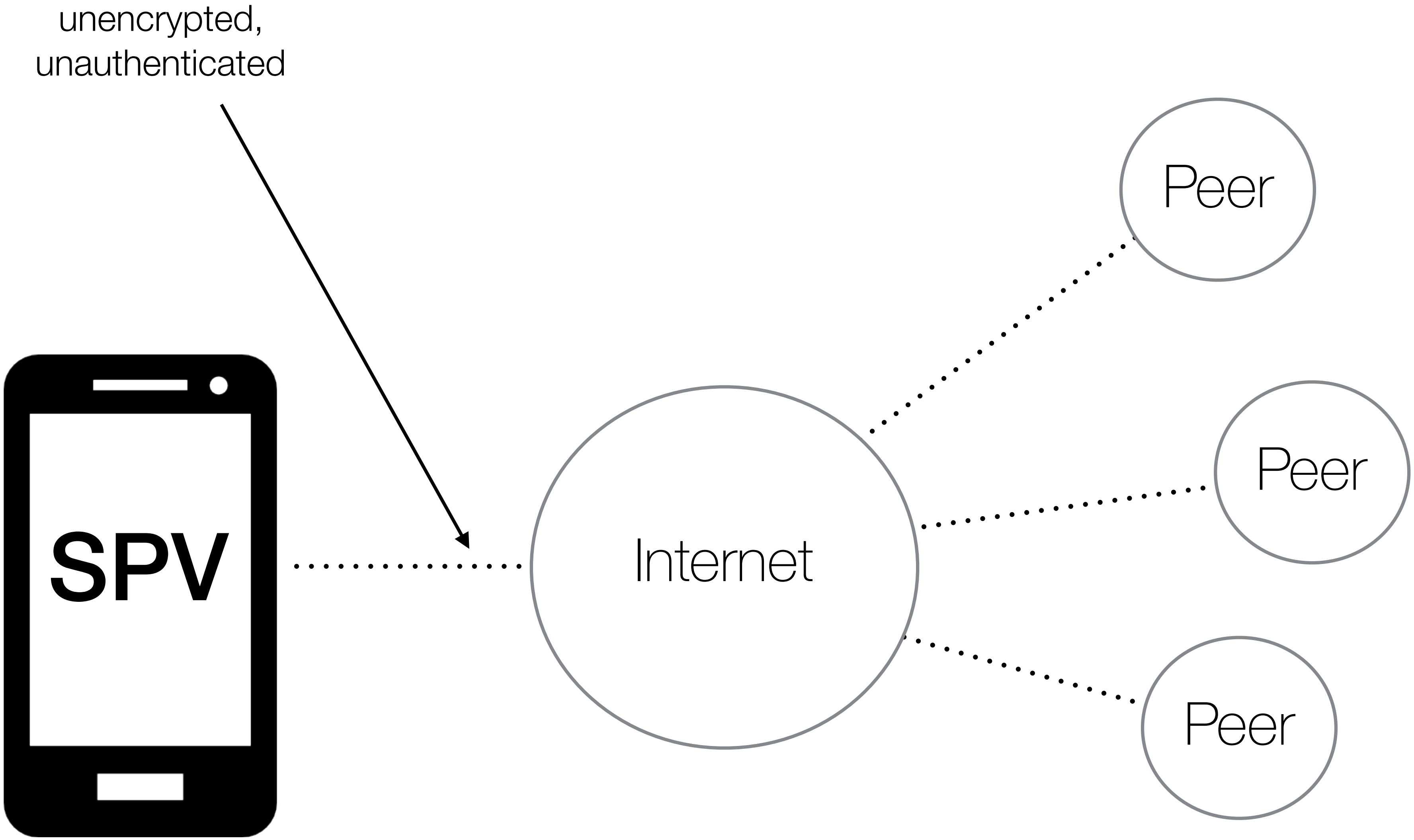
Keeping end-users on the p2p network will help decentralization.

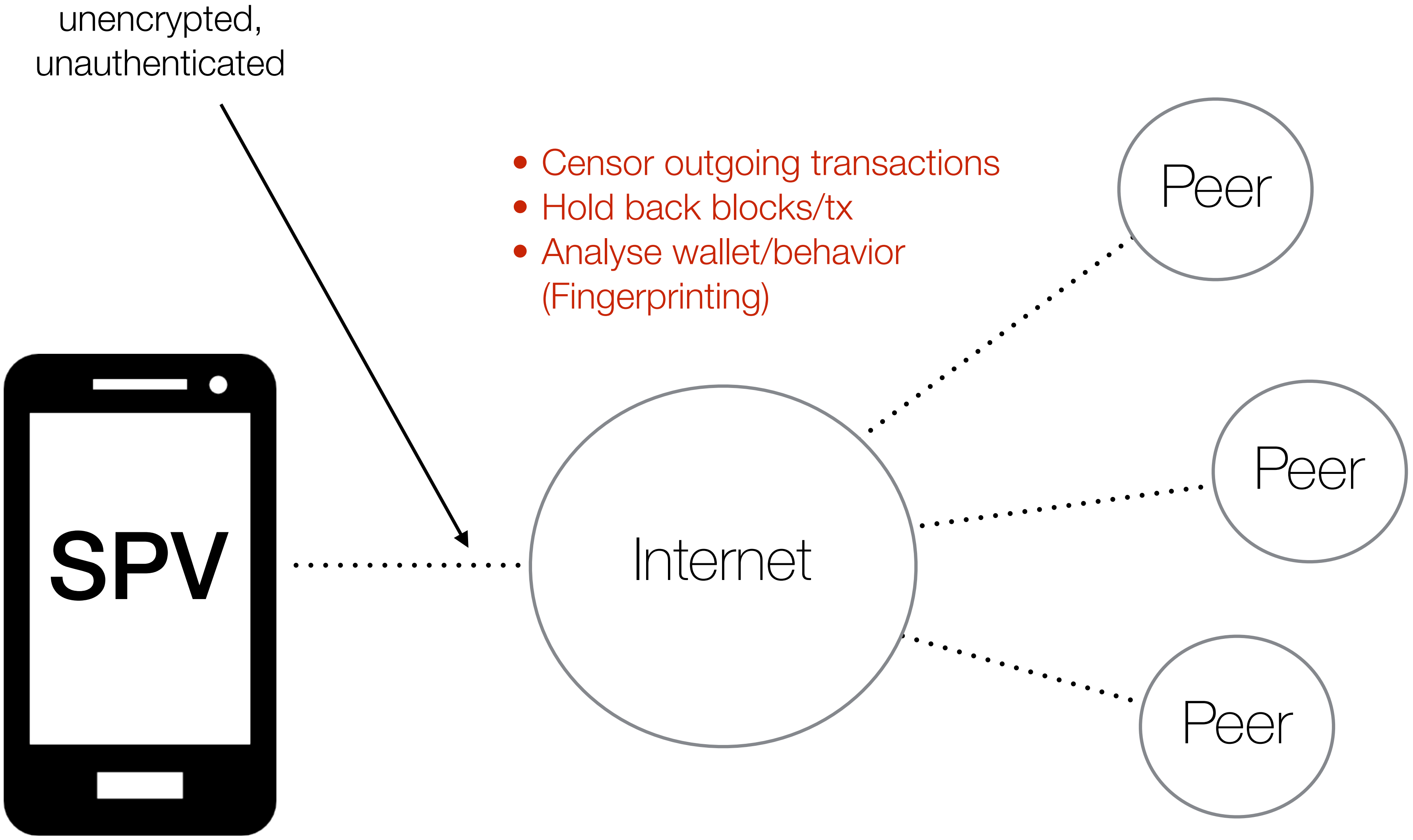
(Or bringing them back)

A Swiss Bank Account in **Your Pocket**?

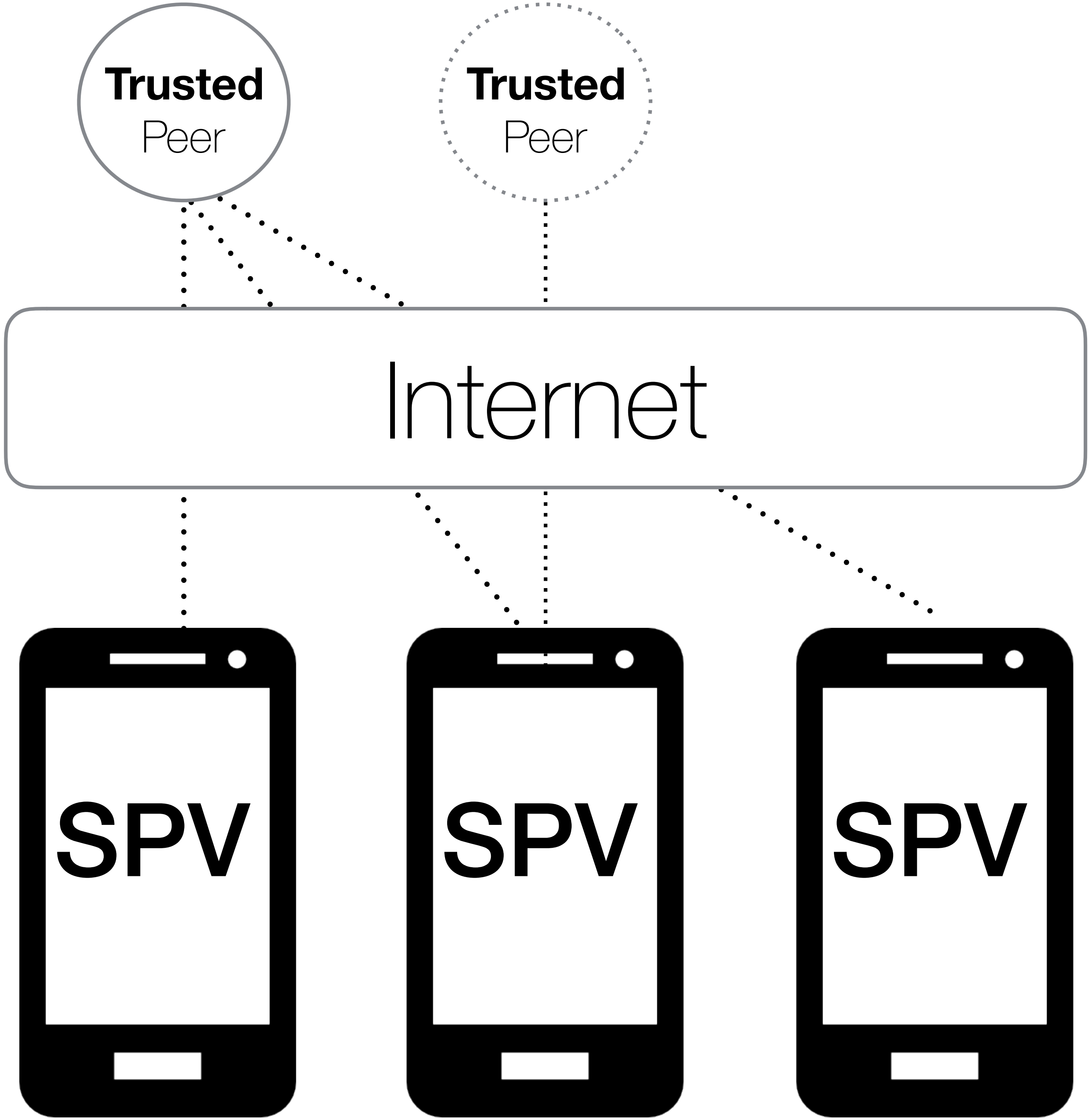
Missing option to **securely connect peers** (Connect your thin client with your full node)

BIP150/151


unencrypted,
unauthenticated
SPV
Internet
Peer
Peer
Peer



- Censor outgoing transactions
- Hold back blocks/tx
- Analyse wallet/behavior (Fingerprinting)


Trusted
Peer
Trusted
Peer
Internet
SPV
SPV
SPV

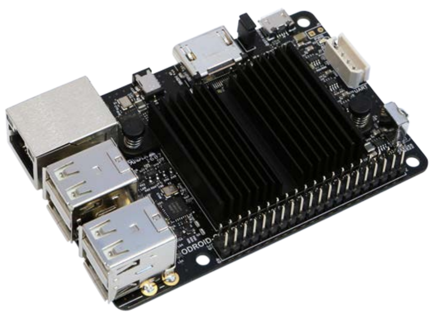
A Swiss Bank Access **at Home**?

BitSeed

~120$

Pine64

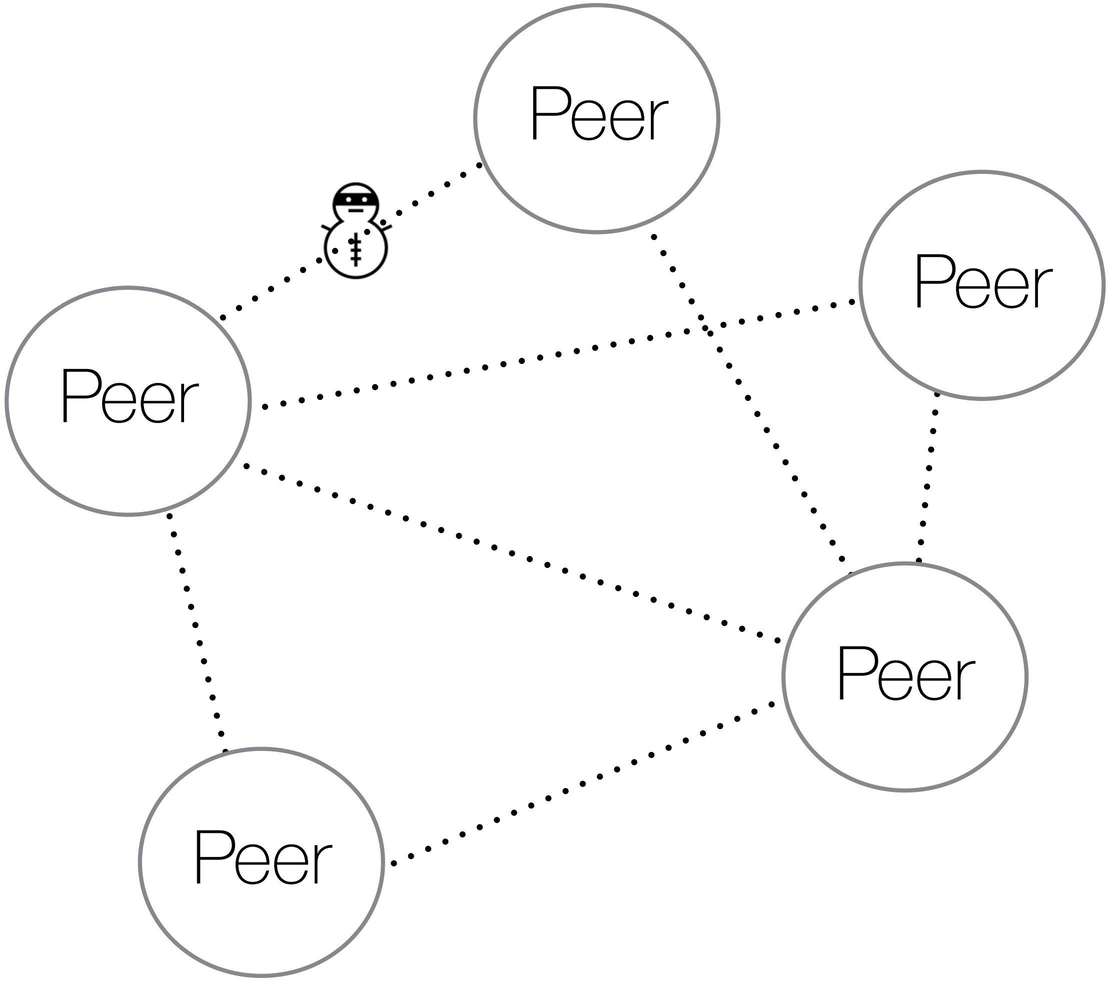
29$



Odroid-C2

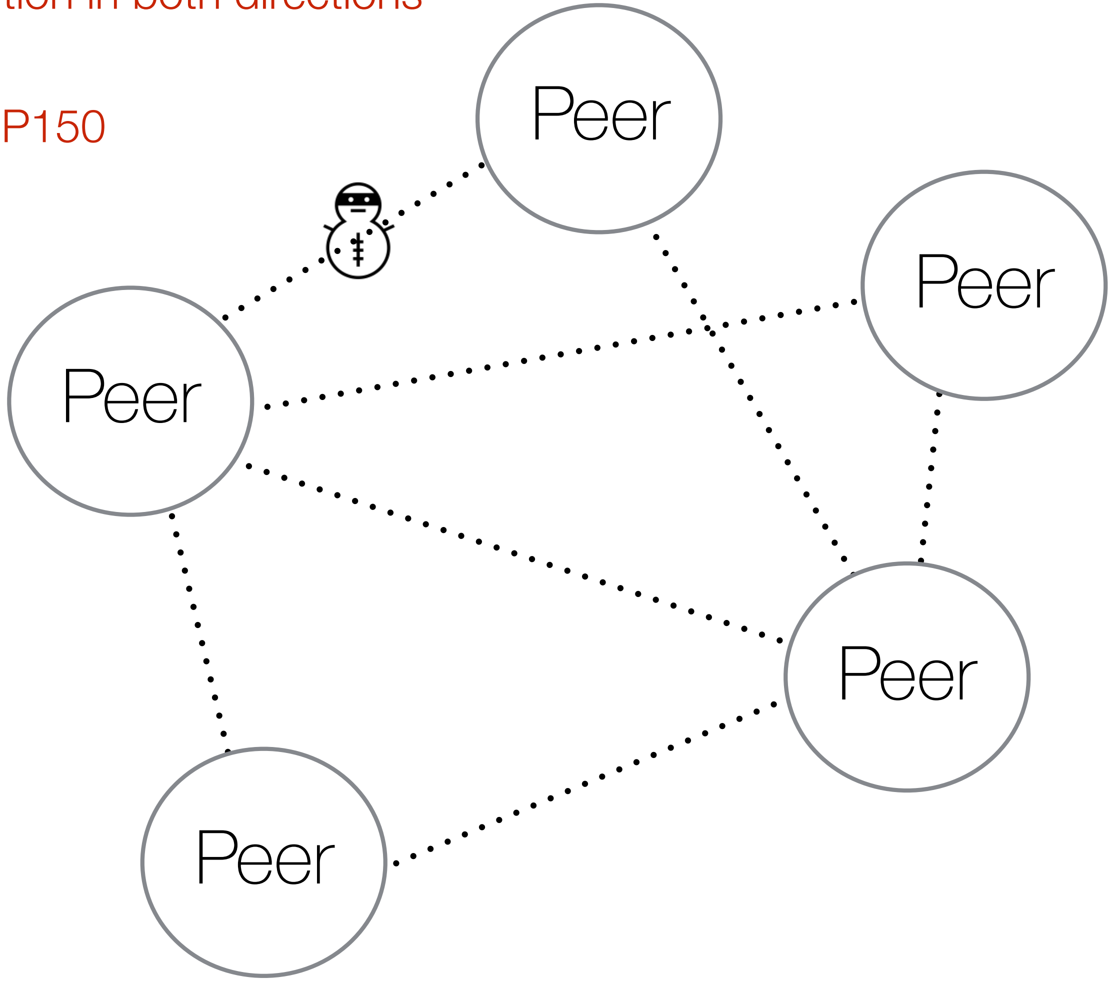
40$

Passiv surveillance



Active surveillance (with BIP151)

- Ephemeral key substitution in both directions
- Risks being detected
- MITM avoidable with BIP150



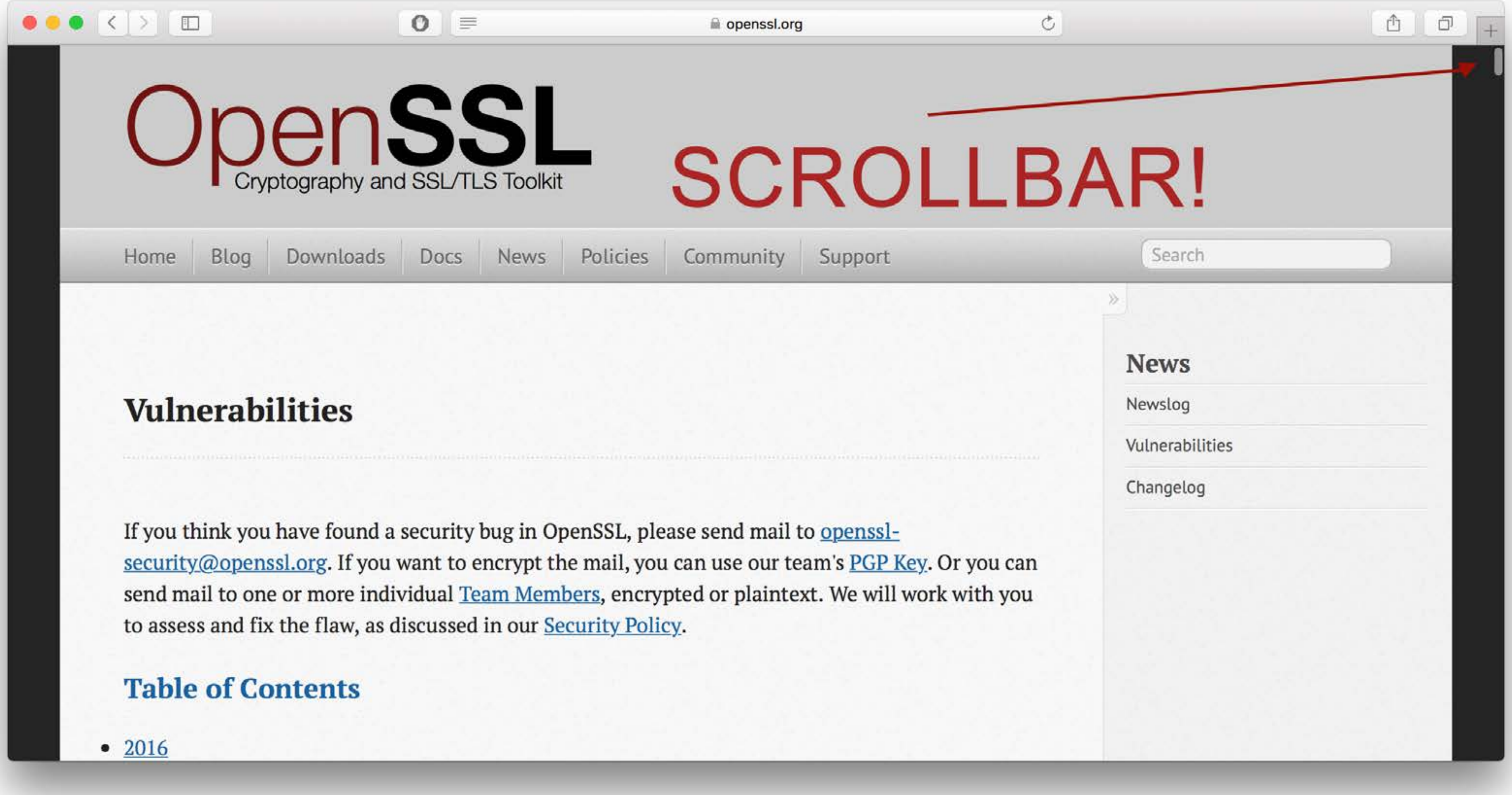
Controllable Stack

ChaCha20-Poly1305

Used by openSSH

Widely used by Google

Why not openSSL?



Vulnerabilities

If you think you have found a security bug in OpenSSL, please send mail to openssl-security@openssl.org. If you want to encrypt the mail, you can use our team's PGP Key. Or you can send mail to one or more individual Team Members, encrypted or plaintext. We will work with you to assess and fix the flaw, as discussed in our Security Policy.

Table of Contents

- 2016

Related solutions?

stunnel

openVPN

Tor

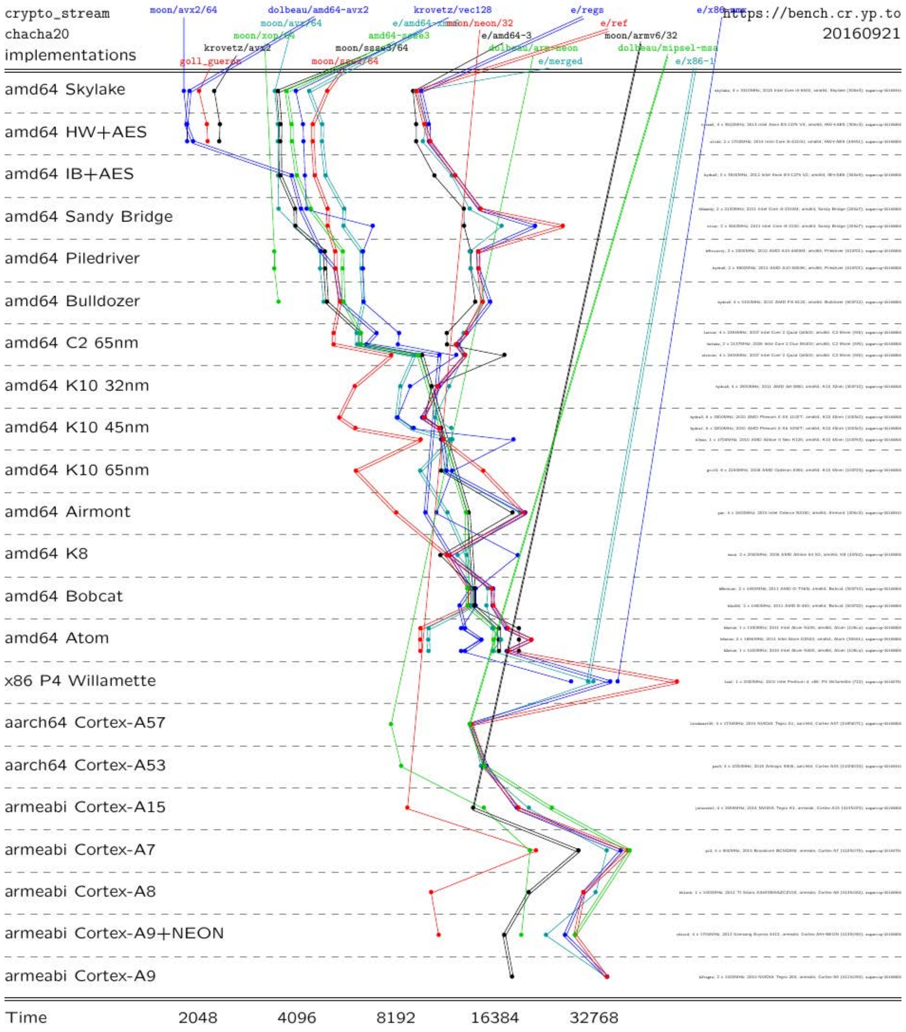
i2p

…

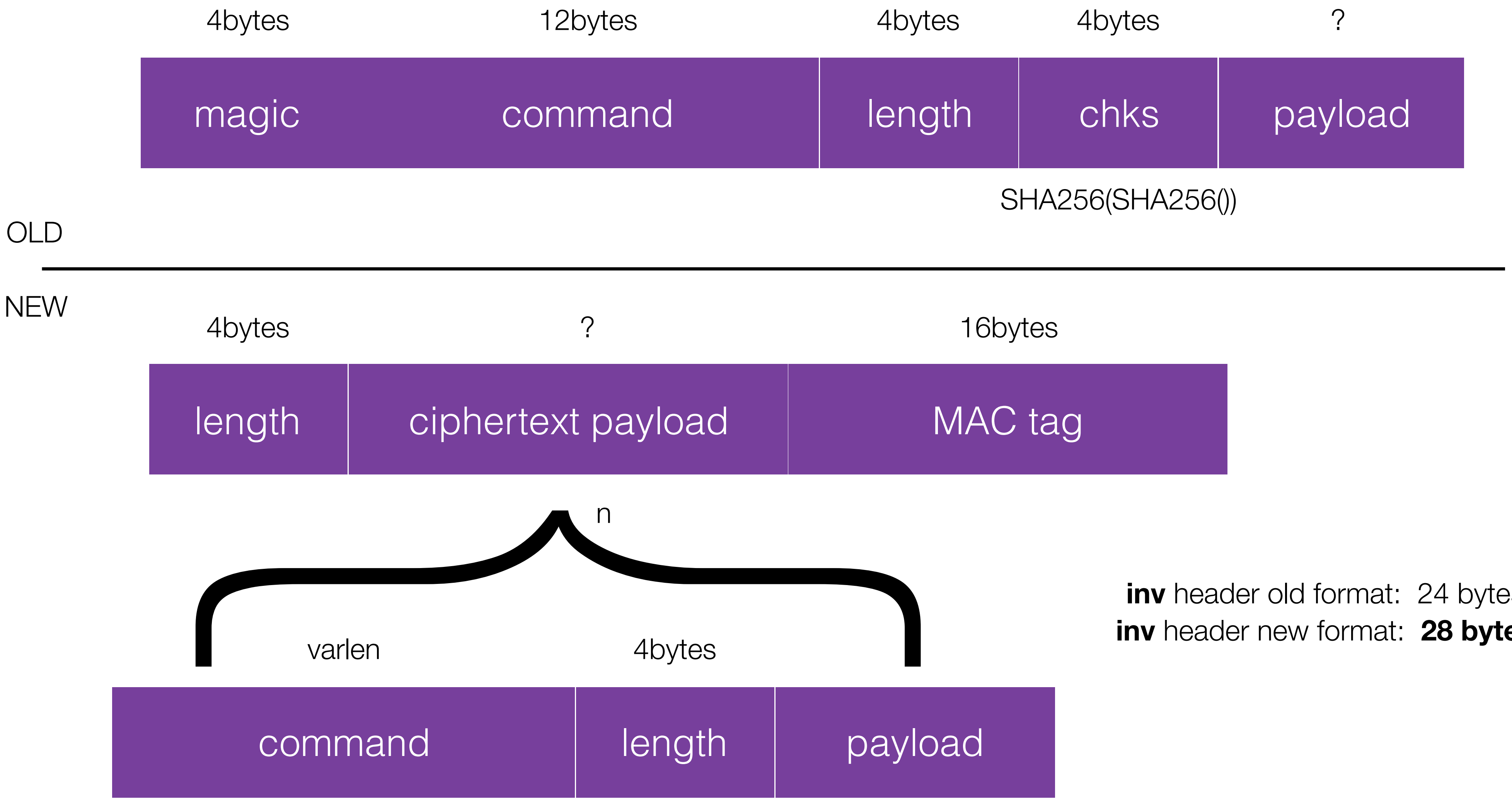
ChaCha20-Poly1305@openssh

- 256bit AEAD (stream cipher)
- ~300 lines of code
- auditable
- fast(er)
- no known security weakness

ChaCha20


crypto_stream
chacha20
implementations
moon/avx2/64
dolbeau/amd64-avx2
krovetz/vec128
e/regs
e/x86-mmx
https://bench.cr.yp.to
20160921
moon/avx/64
e/amd64-xmm6
moon/neon/32
e/ref
moon/xop/64
amd64-ssse3
e/amd64-3
moon/armv6/32
krovetz/avx2
moon/ssse3/64
dolbeau/arm-neon
dolbeau/mipsel-msa
goll_gueron
moon/sse2/64
e/merged
e/x86-1
amd64 Skylake
amd64 HW+AES
amd64 IB+AES
amd64 Sandy Bridge
amd64 Piledriver
amd64 Bulldozer
amd64 C2 65nm
amd64 K10 32nm
amd64 K10 45nm
amd64 K10 65nm
amd64 Airmont
amd64 K8
amd64 Bobcat
amd64 Atom
x86 P4 Willamette
aarch64 Cortex-A57
aarch64 Cortex-A53
armeabi Cortex-A15
armeabi Cortex-A7
armeabi Cortex-A8
armeabi Cortex-A9+NEON
armeabi Cortex-A9
Time
2048
4096
8192
16384
32768

New P2P message structure



OLD

magic	command	length	chks	payload
4bytes	12bytes	4bytes	4bytes	?

SHA256(SHA256())

NEW

length	ciphertext payload	MAC tag
4bytes	?	16bytes

n

command	length	payload
varlen	4bytes	

inv header old format: 24 bytes

inv header new format: **28 bytes**

BIP150

Fingerprinting free peer authentication

AUTHCHALLENGE

→ HASH(session_id || „i“ || remote-peers-expected-identity-pubkey)

AUTHREPLY

← signature(identity_key, session_id)

AUTHPROPOSE

→ HASH(session_id || „p“ || client-identity-pubkey)

AUTHCHALLENGE

← HASH(session_id || „r“ || client-identity-pubkey)

AUTHREPLY

→ signature(identity_key, session_id)

Thanks.

Questions?

🐦 _jonasschnelli_

✉ dev@jonasschnelli.ch