net2o: WOЯꓘƧ ꟻOЯ MƎ ⟶ Works for you

+ net2o onion routing

reinventing the internet

Bernd Paysan

#wefixthenet, 33c3, Hamburg

Outline

Motivation

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

Works for You

Outlook: Onion Routing

Motivation

3.5 years after Snowden

What happend to change the world:

Politics

Post truth	as excuse for censorship
Crypto Wars 4.0	another “we need to look under every bed to search for monsters”
Legalize it	(dragnet surveillance)
Kill the link	LG Hamburg imposes obligations on link setters

Competition	face Stasi style “Zersetzung” like the Tor project
Solutions	net2o starts to be usable (it woɿkƨ ʇoɿ mɘ)

3.5 years after Snowden

What happend to change the world:

Politics

Post truth as excuse for censorship

Crypto Wars 4.0 another "we need to look under every bed to search for monsters"

Legalize it (dragnet surveillance)

Kill the link LG Hamburg imposes obligations on link setters

Competition face Stasi style "Zersetzung" like the Tor project

Solutions net2o starts to be usable (it woɿꓘƨ ꟻoɿ mɘ)

3.5 years after Snowden

What happend to change the world:

Politics

- Post truth as excuse for censorship
- Crypto Wars 4.0 another “we need to look under every bed to search for monsters”
- Legalize it (dragnet surveillance)
- Kill the link LG Hamburg imposes obligations on link setters

Competition face Stasi style “Zersetzung” like the Tor project

Solutions net2o starts to be usable (it woɹʞƨ ꟻoɹ mɘ)

3.5 years after Snowden

What happend to change the world:

Politics

Post truth as excuse for censorship

Crypto Wars 4.0 another "we need to look under every bed to search for monsters"

Legalize it (dragnet surveillance)

Kill the link LG Hamburg imposes obligations on link setters

Competition face Stasi style "Zersetzung" like the Tor project

Solutions net2o starts to be usable (it woɿʞƨ ʇoɿ mɘ)

Where are the defects?

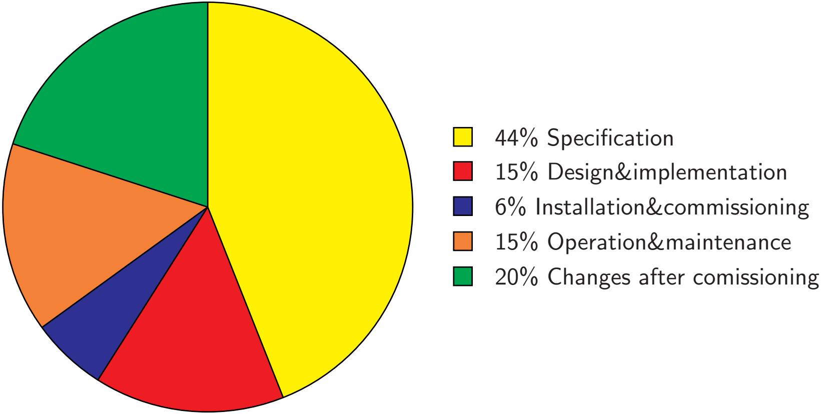


Figure: Bugs by phase [2]

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519,
symmetric authenticated encryption+hash+prng with Keccak,
symmetric block encryption with Threefish
onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519,
   symmetric authenticated encryption+hash+prng with Keccak,
   symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519,
   symmetric authenticated encryption+hash+prng with Keccak,
   symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519,
   symmetric authenticated encryption+hash+prng with Keccak,
   symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

net2o in a nutshell

net2o consists of the following 6 layers (implemented bottom up):

2. Path switched packets with 2^n size writing into shared memory buffers
3. Ephemeral key exchange and signatures with Ed25519, symmetric authenticated encryption+hash+prng with Keccak, symmetric block encryption with Threefish
   onion routing camouflage probably with AES
4. Timing driven delay minimizing flow control
5. Stack–oriented tokenized command language
6. Distributed data (files) and distributed metadata (DHT)
7. Apps in a sandboxed environment for displaying content

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

Objectives

net2o's design objectives are

- lightweight, fast, scalable
- easy to implement
- secure
- media capable
- works as overlay on current networks (UDP/IP), but can replace the entire stack

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI	Create, import, and exchange keys
Permissions	Individual permission bits per key, permission groups
Hashed file copy	Access to big files by hash
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI	Create, import, and exchange keys
Permissions	Individual permission bits per key, permission groups
Hashed file copy	Access to big files by hash
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI Create, import, and exchange keys

Permissions Individual permission bits per key, permission groups

Hashed file copy Access to big files by hash

Vault A container for encrypted data without metadata exposure

DHT Query key/value pairs (keys are pubkeys or hash keys)

Chat Instant messaging 1:1 or in chat groups

Version control system For larger content

Sync to synchronize your computers (RSN)

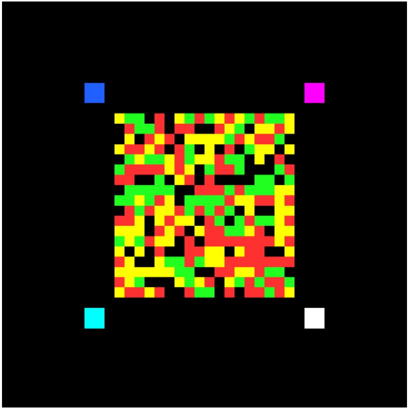
Audio/Video Chat Real time data streaming (RSN)

WOЯꓘƧ ꟻOЯ MƎ: Progress Report

PKI	Create, import, and exchange keys
Permissions	Individual permission bits per key, permission groups
Hashed file copy	Access to big files by hash
Vault	A container for encrypted data without metadata exposure
DHT	Query key/value pairs (keys are pubkeys or hash keys)
Chat	Instant messaging 1:1 or in chat groups
Version control system	For larger content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Color QR

For easy key exchange, scan a color QR image (work in progress)



Get it: Debian and Android

Debian

To use the Debian package, enter as root:

```
cat >/etc/apt/sources.list.d/net2o.list <<EOF
deb [arch=amd64,all] http://net2o.de/debian testing main
EOF
wget -O - https://net2o.de/bernd@net2o.de.gpg.asc | \
apt-key add -
aptitude update; aptitude install net2o
```

Android

Get Gforth from play store or `https://net2o.de/Gforth.apk`

Open/close (back button) Gforth if you like; then open net2o.

Get it: Debian and Android

Debian

To use the Debian package, enter as root:

```
cat >/etc/apt/sources.list.d/net2o.list <<EOF
deb [arch=amd64,all] http://net2o.de/debian testing main
EOF
wget -O - https://net2o.de/bernd@net2o.de.gpg.asc | \
apt-key add -
aptitude update; aptitude install net2o
```

Android

Get Gforth from play store or `https://net2o.de/Gforth.apk`

Open/close (back button) Gforth if you like; then open net2o.

Get it: Windows and macOS

Windows

Get the two current setup.exes for Gforth and net2o, and install them in that order:

`http://www.complang.tuwien.ac.at/forth/gforth/Snapshots/current/gforth64.exe`

`https://net2o.de/windows/net2o64.exe`

MacOS

Once I got around creating a brew tap, it will be easy to install under MacOS, too.

Get it: Windows and macOS

Windows

Get the two current setup.exes for Gforth and net2o, and install them in that order:

`http://www.complang.tuwien.ac.at/forth/gforth/Snapshots/current/gforth64.exe`

`https://net2o.de/windows/net2o64.exe`

MacOS

Once I got around creating a brew tap, it will be easy to install under MacOS, too.

Get it from Source

From Source

for Linux, Mac OS X, Windows (cygwin) you need:

```
git automake autoconf make gcc libtool libltdl7 fossil
you run: mkdir net2o; cd net2o
wget https://fossil.net2o.de/net2o/doc/trunk/do
chmod +x do; ./do
```

This will install some stuff and take some time

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- "circuit" used long enough for correlation attacks
- NSA project, EFF version's primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for "hidden services"
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

State of the Art

Tor Circuit switched onion router with a number of weaknesses:

- centralized directory servers
- “circuit” used long enough for correlation attacks
- NSA project, EFF version’s primary goal apparently to generate cover traffic

I2P Architecture similar to Tor, but

- optimized for “hidden services”
- packet switched instead of circuit switched

Goals for net2o Onion Routing

1. Limited to net2o ⟷ net2o
2. Create a circuit mesh, and then switch quickly, using net2o's fast handover
3. Leverage net2o's inherent capabilities to reduce possible timing attacks
4. Avoid legal problems of especially exit nodes by not exiting
5. If you want content from outside net2o's world, share the imported content as net2o files/dvcs projects

Goals for net2o Onion Routing

1. Limited to net2o ⟷ net2o
2. Create a circuit mesh, and then switch quickly, using net2o's fast handover
3. Leverage net2o's inherent capabilities to reduce possible timing attacks
4. Avoid legal problems of especially exit nodes by not exiting
5. If you want content from outside net2o's world, share the imported content as net2o files/dvcs projects

Goals for net2o Onion Routing

1. Limited to net2o ⟷ net2o
2. Create a circuit mesh, and then switch quickly, using net2o's fast handover
3. Leverage net2o's inherent capabilities to reduce possible timing attacks
4. Avoid legal problems of especially exit nodes by not exiting
5. If you want content from outside net2o's world, share the imported content as net2o files/dvcs projects

Goals for net2o Onion Routing

1. Limited to net2o ⟷ net2o
2. Create a circuit mesh, and then switch quickly, using net2o's fast handover
3. Leverage net2o's inherent capabilities to reduce possible timing attacks
4. Avoid legal problems of especially exit nodes by not exiting
5. If you want content from outside net2o's world, share the imported content as net2o files/dvcs projects

Goals for net2o Onion Routing

1. Limited to net2o ⟷ net2o
2. Create a circuit mesh, and then switch quickly, using net2o's fast handover
3. Leverage net2o's inherent capabilities to reduce possible timing attacks
4. Avoid legal problems of especially exit nodes by not exiting
5. If you want content from outside net2o's world, share the imported content as net2o files/dvcs projects

Implementation plan

- Add a header field for *n* encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged... use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for *n* encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged… use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for n encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged… use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for *n* encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged… use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the “connect node:” connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for *n* encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged… use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for n encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged… use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

Implementation plan

- Add a header field for *n* encrypted paths ($n = 4$ seems to be a good choice)
- Block cipher decryption and encryption can be interchanged... use AES since fast hardware accelerated AES is available
- On arrival, try–decrypt/encrypt the first path with negotiated keys from that source and verify authenication
- Decrypt or encrypt (depending on direction) the rest of the packet with that key
- Shift the path list by one and insert return path (properly encrypted/decrypted)
- At the "connect node:" connect both ends, i.e. flip and remember the incoming path list, and replace it with the outgoing path list.
- While you are connected, the other side tells you several connect nodes, which you can use

For Further Reading I

- Bernd Paysan
  net2o source repository and wiki
  http://fossil.net2o.de/net2o
- Health & Safety Executive HSE – UK
  Out of control, 2nd edition 2003
  http://www.hse.gov.uk/pubns/priced/hsg238.pdf