net2o : Using net2o

reinventing the internet

Bernd Paysan

EuroForth 2016, Konstanz/Reichenau

Outline

3 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be increasingly usable

3 years after Snowden

What happend to change the world:

- **Politics** Manhatten project to find “the golden key”?
- **Users** don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
- **Software** NSA backdoors have been refitted by attackers (Juniper)
- **Solutions** net2o starts to be increasingly usable

3 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be increasingly usable

3 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be increasingly usable

3 years after Snowden

What happend to change the world:

Politics	Manhatten project to find “the golden key”?
Users	don’t want their dick picks be watched and use DuckDuckGo and encrypted chat
Software	NSA backdoors have been refitted by attackers (Juniper)
Solutions	net2o starts to be increasingly usable

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 (prefix hash trie)
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 (prefix hash trie)
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 (prefix hash trie)
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 (prefix hash trie)
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 (prefix hash trie)
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 (prefix hash trie)
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 (prefix hash trie)
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

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI Create, import, and exchange keys

Named file copy For testing only

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/structured content

Sync to synchronize your computers (RSN)

Audio/Video Chat Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Basic Frameworks

PKI	Create, import, and exchange keys
Named file copy	For testing only
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/structured content
Sync	to synchronize your computers (RSN)
Audio/Video Chat	Real time data streaming (RSN)

Try it

Debian Use the Debian package, and 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.

Try it

Debian Use the Debian package, and 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.

Try 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 (I will try to improve that).

Try it — Generate a Key

Linux you run:

`n2o cmd`

`keygen` *<nick>*

Enter your passphrase twice.

Android Tap on the little nettie to start the app, it will autodetect that you don't have a key generated. Enter nick and passphrase twice.

Try it — Generate a Key

Linux you run:

`n2o cmd`

`keygen` *<nick>*

Enter your passphrase twice.

Android Tap on the little nettie to start the app, it will autodetect that you don't have a key generated. Enter nick and passphrase twice.

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with `/bye` or Ctrl+D (back on Android)

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with `/bye` or Ctrl+D (back on Android)

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with `/bye` or Ctrl+D (back on Android)

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with `/bye` or Ctrl+D (back on Android)

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with `/bye` or Ctrl+D (back on Android)

Try it — get a key and chat

- To get my key, search for it (32 bit is sufficient)
  `keysearch kQusJ`
- Send me an invitation
  `invite @bernd`
- Try to chat with me
  `chat forth@bernd`
- Aquire more keys by observing a group chat. List your keys with
  `n2o keylist`
  from within the chat.
- Change networks with your Android and watch that the chat still works.
- Leave the chat with **`/bye`** or Ctrl+D (back on Android)

Try it — Vault en/decryption

- Take a file and encrypt it
  `enc test.txt`
- Show it's content
  `cat test.txt.v2o`
- Sign a file with a detached signature
  `sign test.txt`
- Verify the signature
  `verify test.txt`

Try it — Vault en/decryption

- Take a file and encrypt it
  `enc test.txt`
- Show it's content
  `cat test.txt.v2o`
- Sign a file with a detached signature
  `sign test.txt`
- Verify the signature
  `verify test.txt`

Try it — Vault en/decryption

- Take a file and encrypt it
  `enc test.txt`
- Show it's content
  `cat test.txt.v2o`
- Sign a file with a detached signature
  `sign test.txt`
- Verify the signature
  `verify test.txt`

Try it — Vault en/decryption

- Take a file and encrypt it
  `enc test.txt`
- Show it's content
  `cat test.txt.v2o`
- Sign a file with a detached signature
  `sign test.txt`
- Verify the signature
  `verify test.txt`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

Try it — Use the DVCS

- Create a directory and add a few files into it, keep a net2o instance running inside that directory with
  `n2o cmd`
- Initialize the directory
  `init`
- Add the files in the directory
  `add *`
  `ci -m "My checkin message"`
  and check them in
- Change a file and see what has changed
  `diff`
- Check in the changed file
  `ci -m "Second checkin"`
- Show the commit messages
  `log`

For Further Reading I

Bernd Paysan

net2o source repository and wiki

http://fossil.net2o.de/net2o