



Rosenpass

Securing & Deploying Post-Quantum WireGuard

Karolin Varner, with Benjamin Lipp, Wanja Zaeske, Lisa Schmidt

January 5, 2024

RWPQC23 | <https://rosenpass.eu/whitepaper.pdf>

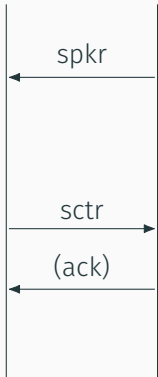
Structure of the talk

- Post-quantum WireGuard¹: How to build an interactive key exchange from KEMs
- Contribution: State Disruption Attacks & cookies as a defense
- Contribution: Symbolic analysis of the Rosenpass protocol
- Contribution: Noise-like specification
- Contribution: New hashing & domain separation scheme
- Contribution: Reference implementation – Securing WireGuard in practice

¹Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R. Zimmermann. “Post-quantum WireGuard”. In: 42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021. Full version: <https://eprint.iacr.org/2020/379>

Post-quantum WireGuard: Three encapsulations

Initiator Responder



Responder Auth

Initiator Responder



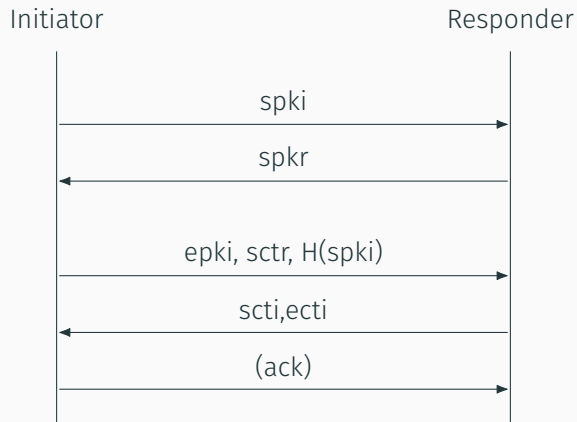
Initiator Auth

Initiator Responder



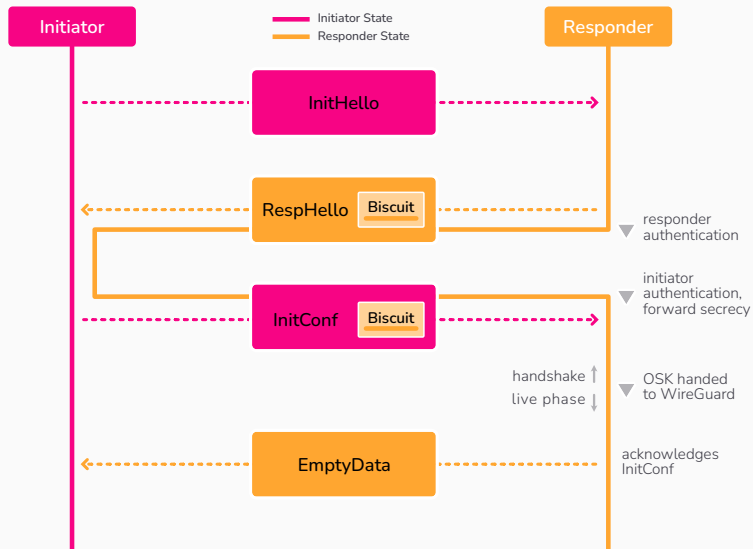
Forward secrecy

Combining the three encapsulations in one protocol



Note that the initiator is not authenticated until they send “(ack)”.

The Rosenpass protocol



CVE-2021-46873 – DOS against WireGuard through NTP

- The replay protection in classic WireGuard assumes a monotonic counter
- But the system time is attacker controlled because NTP is insecure
- This generates a kill packet that abuses replay protection and renders the initiator's key-pair useless
- Attack is possible in the real world!
- Similar attack in post-quantum WireGuard is worse since InitHello is unauthenticated
- Solution: Biscuits

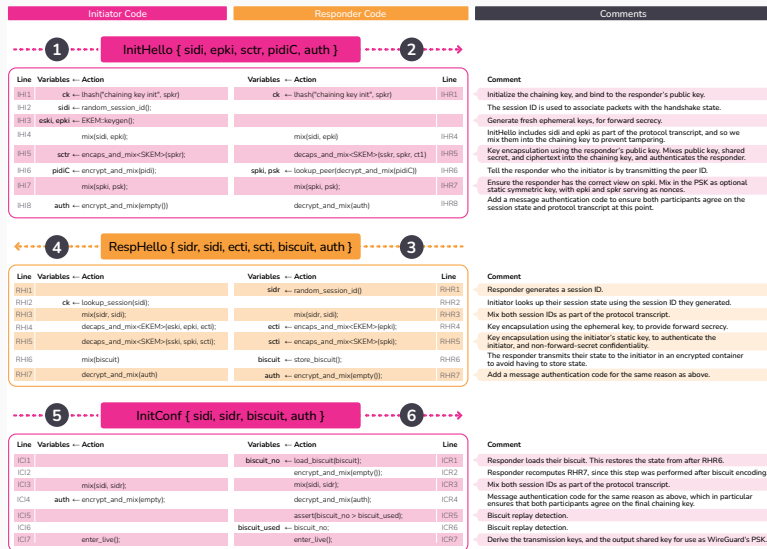
Security analysis of rosenpass

- CryptoVerif in progress
- Symbolic analysis using ProVerif
- Code is part of the software repository & build system
- Symbolic analysis is fast (about five minutes), runs in parallel and is

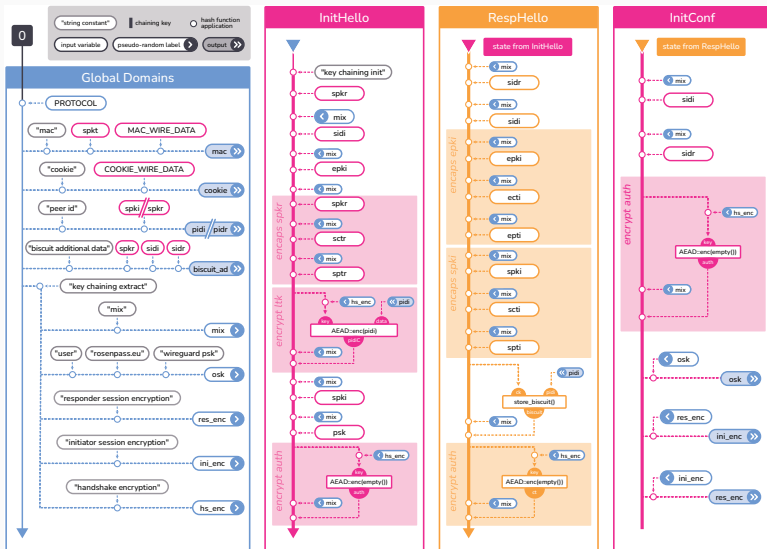
Proverif in technicolor

```
~/p/rosenpass ➤ dev/karo/rwpqc-slides ? ➤ nix build .#packages.x86_64-linux.proof-proverif --print-build-logs [17/17]
rosenpass-proverif-proof> unpacking sources
rosenpass-proverif-proof> unpacking source archive /nix/store/cznyv4ibwlbh257v6lzx8r8al4cb0v0-source
rosenpass-proverif-proof> source root is source
rosenpass-proverif-proof> patching sources
rosenpass-proverif-proof> configuring
rosenpass-proverif-proof> no configure script, doing nothing
rosenpass-proverif-proof> building
rosenpass-proverif-proof> no Makefile, doing nothing
rosenpass-proverif-proof> installing
rosenpass-proverif-proof> $ metaverif analysis/01_secrecy.entry.mpv -color -html /nix/store/gidm68r04lkpanvkgz48527qf6nym6dv
-rosenpass-proverif-proof
rosenpass-proverif-proof> $ metaverif analysis/02_availability.entry.mpv -color -html /nix/store/gidm68r04lkpanvkgz48527qf6n
ym6dv-rosenpass-proverif-proof
rosenpass-proverif-proof> $ wait -f 34
rosenpass-proverif-proof> $ cpp -P -I/build/source/analysis analysis/01_secrecy.entry.mpv -o target/proverif/01_secrecy.ent
ry.i.pv
rosenpass-proverif-proof> $ cpp -P -I/build/source/analysis analysis/02_availability.entry.mpv -o target/proverif/02_availab
ility.entry.i.pv
rosenpass-proverif-proof> $ awk -f marzipan/marzipan.awk target/proverif/01_secrecy.entry.i.pv
rosenpass-proverif-proof> $ awk -f marzipan/marzipan.awk target/proverif/02_availability.entry.i.pv
rosenpass-proverif-proof> 4s ✓ state coherence, initiator: Initiator accepting a RespHello message implies they also generat
ed the associated InitHello message
rosenpass-proverif-proof> 35s ✓ state coherence, responder: Responder accepting an InitConf message implies they also genera
ted the associated RespHello message
rosenpass-proverif-proof> 0s ✓ secrecy: Adv can not learn shared secret key
rosenpass-proverif-proof> 0s ✓ secrecy: There is no way for an attacker to learn a trusted kem secret key
rosenpass-proverif-proof> 0s ✓ secrecy: The adversary can learn a trusted kem pk only by using the reveal oracle
rosenpass-proverif-proof> 0s ✓ secrecy: Attacker knowledge of a shared key implies the key is not trusted
rosenpass-proverif-proof> 31s ✓ secrecy: Attacker knowledge of a kem sk implies the key is not trusted
```

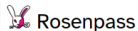

Noise-like specification (easier for engineers)



New Hashing/Domain separation scheme



Reference implementation in rust, deploying post-quantum-secure WireGuard

[Getting started](#)[About](#)[Source Code](#)[Whitepaper](#)[Contributors](#)[Press](#)[Contact](#)

```
rp pubkey server.rosenpass-secret server.rosenpass-public
rp pubkey client.rosenpass-secret client.rosenpass-public
```

Copy the **-public** directories to the other peers and then start the VPN. On the server:

```
sudo rp exchange server.rosenpass-secret dev rosenpass0 listen 192.168.0.1:9999 \
peer client.rosenpass-public allowed-ips fe80::/64
```

On the client:

```
sudo rp exchange client.rosenpass-secret dev rosenpass0 \
peer server.rosenpass-public endpoint 192.168.0.1:9999 allowed-ips fe80::/64
```

Assign IP addresses:

```
sudo ip a add fe80::1/64 dev rosenpass0 # Server
sudo ip a add fe80::2/64 dev rosenpass0 # Client
```

Test the connection by pinging the server on the client machine:

```
ping fe80::1%rosenpass0 # Client
```

You can watch how Rosenpass replaces the WireGuard PSK with the following command:

```
watch -n 0.2 'wg show all; wg show all preshared-keys'
```