Rosenpass

Securing & Deploying Post-Quantum WireGuard

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

02 March 2023

https://rosenpass.eu/whitepaper.pdf

Structure of the talk

- Problem statement: Post-quantum WireGuard
- Post-quantum WireGuard[1]: How to build an interactive key exchange from KEMs
- Attack we found: State Disruption Attacks
- Real-World Concerns
- Biscuits as a defense against State Disruption Attacks

[1] Andreas 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

What needs to be done to deploy Post-Quantum WireGuard

- Updating the WireGuard protocol to support post-quantum security
- Updating the (post quantum) WireGuard protocol to be secure against state disruption attacks
- Reference implementation of the Rosenpass protocol in Rust
- A way to create hybrid post-quantum secure WireGuard VPNs
- Stand-alone key exchange app
- A Sci-Comm project teaching people about post-quantum security

WireGuard[3]

- VPN protocol in the linux kernel
- Based on Noise IKpsk1 from the Noise Protocol Framework[2]
- Small, fast, open source crypto

[2]Trevor Perrin. The Noise Protocol Framework. 2016. url: http://noiseprotocol.org/noise.pdf

[3]Jason A. Donenfeld. "WireGuard: Next Generation Kernel Network Tunnel". In: 24th Annual Network and Distributed System Security Symposium, NDSS 2017, San Diego, California, USA, February 26 - March 1, 2017. Whitepaper: https: //www.wireguard.com/papers/wireguard.pdf.

WireGuard/Noise IKpsk security properties

- ✓ Session-key secrecy
- ✓ Forward-secrecy
- ✓ Mutual authentication
- ✓ Session-key Uniqueness
- ✓ Identity Hiding
- ✓ (DoS Mitigation – First packet is authenticated[4])

[4]Based on the unrealistic assumption of a monotonic counter – We found a practical attack

Security of Rosenpass

WireGuard

- ✓ Session-key secrecy
- ✓ Forward-secrecy
- ✓ Mutual authentication
- ✓ Session-key Uniqueness
- ✓ Identity Hiding
- ✓ (DoS Mitigation)

Post-Quantum WireGuard

- ✗ Identity Hiding [a]
- ✗ DoS Mitigation [b]

[a] Based on a Identity Hiding/ANON-CCA security of McEliece; unclear whether that holds.

[b] PQWG provides DoS mitigation under the assumption of a secret PSK, which quite frankly is cheating.

Rosenpass

- ✓ DoS Mitigation
- ✓ Hybrid Post-Quantum security[a]

[a] In deployments using WireGuard + Rosenpass; Rosenpass on its own provides post-quantum security.

Building post-quantum WireGuard: NIKE vs KEM

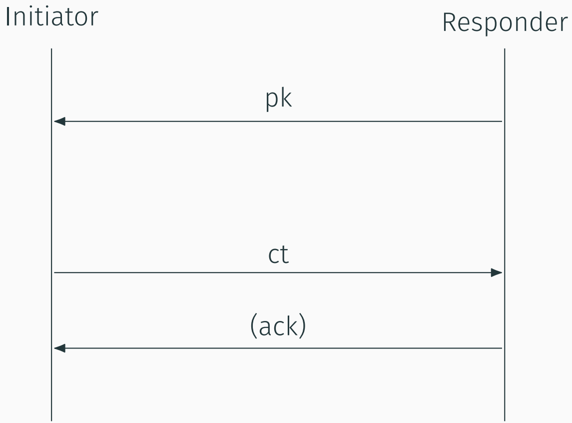
NIKE:

$$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathtt{NIKE.KeyGen}$$
$$(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathtt{NIKE.KeyGen}$$
$$\mathtt{NIKE.SharedKey}(\mathsf{sk}_1, \mathsf{pk}_2) = \mathtt{NIKE.SharedKey}(\mathsf{sk}_2, \mathsf{pk}_1)$$

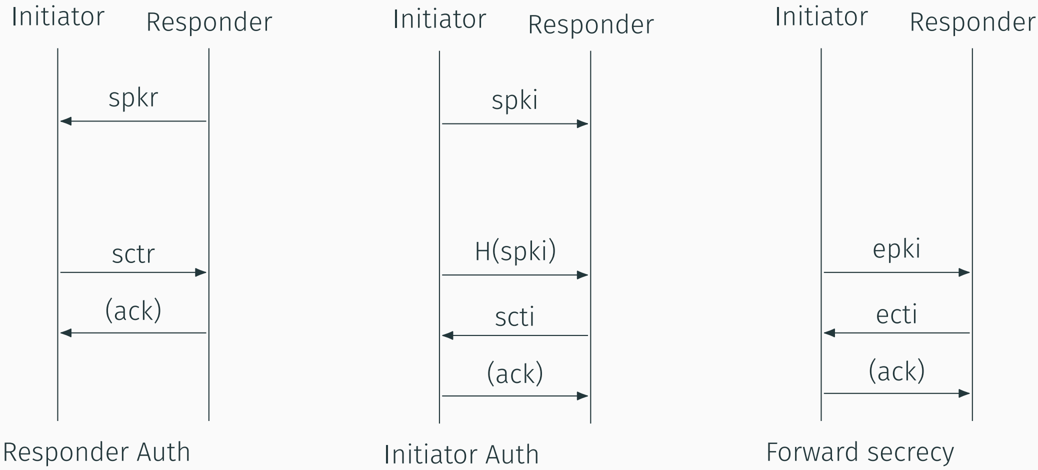
KEM:

$$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{KEM.KeyGen}$$
$$(\mathsf{shk}, \mathtt{ct}) \leftarrow \mathtt{KEM.Encaps}(\mathsf{pk})$$
$$\mathsf{shk} = \mathtt{KEM.Decaps}(\mathsf{sk}, \mathtt{ct})$$

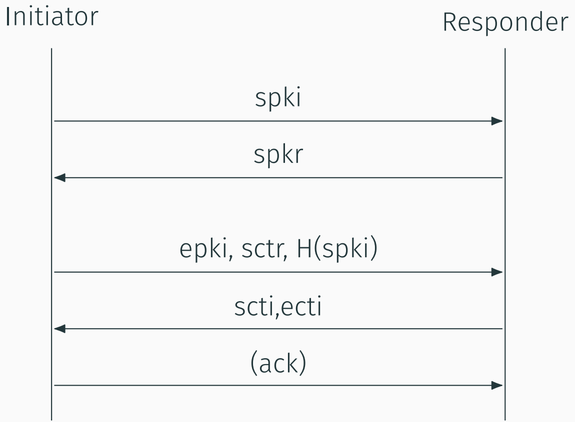
Minimal key exchange using KEMs



Three encapsulations: Achieving mutual authentication & forward secrecy

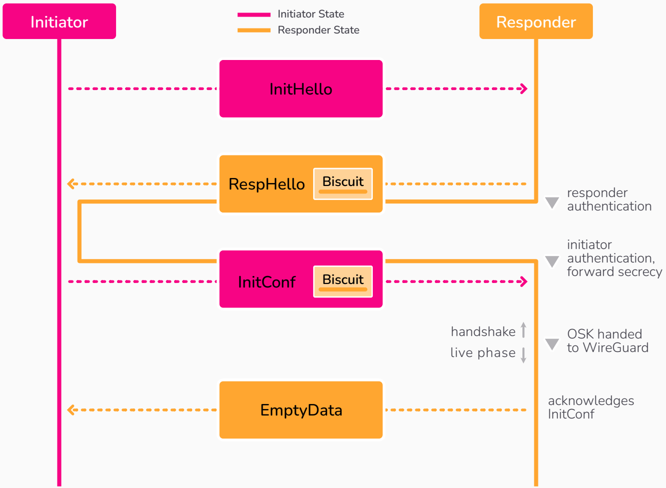


Combining the three encapsulations in one protocol



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

In Rosenpasss specifically



In Rosenpasss specifically

Envelope

	bytes
type	1
reserved	3
payload	n
mac	16
cookie	16
envelope	n + 36

MAC_WIRE_DATA

COOKIE_WIRE_DATA

InitHello
type=0x81

sidi	4
epki	800
sctr	188
pidiC	32 + 16 = 48
auth	16
payload	1056
+ envelope	1092

RespHello
type=0x82

sidr	4
sidi	4
ecti	768
scti	188
biscuit	76 + 24 + 16 = 116
auth	16
payload	1096
+ envelope	1132

InitConf
type=0x83

sidi	4
sidr	4
biscuit	76 + 24 + 16 = 116
auth	16
payload	140
+ envelope	176

EmptyData
type=0x84

sid	4
ctr	8
auth	16
payload	28
+ envelope	64

Data
type=0x85

sid	4
ctr	8
data	variable + 16
payload	variable + 28
+ envelope	variable + 64

CookieReply
type=0x86

type(0x86)	1
reserved	3
sid	4
nonce	24
cookie	16 + 16 = 32
payload	64

biscuit

pidi	32
biscuit_no	12
ck	32
biscuit	76
+ nonce	100
+ auth code	116

data nonce auth code

State Disruption Attacks

- Use the fact that the initiator is not authenticated until their last message
- Send faux initiations, overwriting – and thus erasing – the responder's handshake state
- Erasing the state aborts protocol execution
- PQWG argues: The first package is authenticated using the PSK, therefor sending faux initiations works
- Attacker could replay a legitimate message, but...

State Disruption Attacks on authenticated initial package

- In Classic WireGuard the initial message (InitHello) is authenticated through static-static Diffie-Hellman
- Replay protection uses monotonic counter
- WireGuard stores the time of the last initiator t_i
- When WireGuard receives legitimate initiaton with timestamp t, it stores that time $t_i \leftarrow t$a
- All InitHello messages with a stale timestamp ($t \leq t_i$) get rejected

CVE-2021-46873 – Attacking 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 can be used to render WireGuard keys useless
- Attack is possible in the real world!

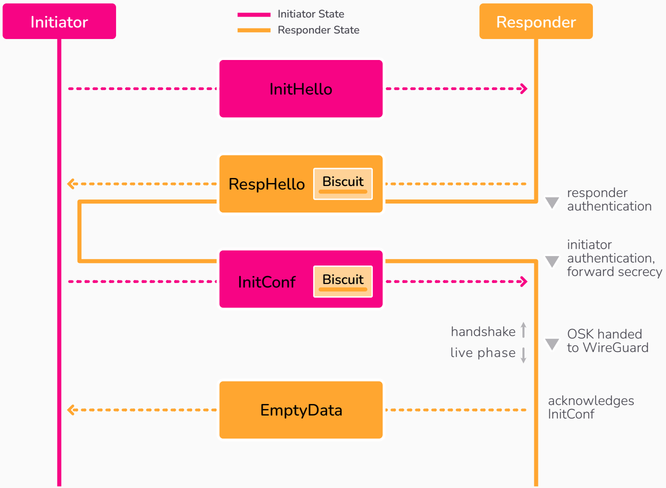
State disruption in Post-Quantum WireGuard

- This mechanism needs an authenticated InitHello message
- Post-Quantum WireGuard relies on the `psk` to provide InitHello authentication
- PQWG sets $\texttt{psk} = H(\texttt{spki} \oplus \texttt{spkr})$ to achieve a secret psk.a
- Relying on private public keys is absurd

$\Rightarrow$ With InitHello effectively unauthenticated, attacker can just generate their own kill packet

Solution: Store the responder state in a biscuit (cookie), so there is no state to override.

Biscuits in the protocol flow



Biscuits in the messages

Envelope

	bytes
type	1
reserved	3
payload	n
mac	16
cookie	16
envelope	n + 36

MAC_WIRE_DATA: type, reserved, payload

COOKIE_WIRE_DATA: type, reserved, payload, mac

InitHello
type=0x81

sidi	4
epki	800
sctr	188
pidiC	32 + 16 = 48
auth	16
payload	1056
+ envelope	1092

RespHello
type=0x82

sidr	4
sidi	4
ecti	768
scti	188
biscuit	76 + 24 + 16 = 116
auth	16
payload	1096
+ envelope	1132

InitConf
type=0x83

sidi	4
sidr	4
biscuit	76 + 24 + 16 = 116
auth	16
payload	140
+ envelope	176

EmptyData
type=0x84

sid	4
ctr	8
auth	16
payload	28
+ envelope	64

Data
type=0x85

sid	4
ctr	8
data	variable + 16
payload	variable + 28
+ envelope	variable + 64

CookieReply
type=0x86

type(0x86)	1
reserved	3
sid	4
nonce	24
cookie	16 + 16 = 32
payload	64

biscuit

pidi	32
biscuit_no	12
ck	32
biscuit	76
+ nonce	100
+ auth code	116

data nonce auth code

Biscuits

- Assumptions such as a monotonic counter are perilous in the real world
- Giving the adversary access to state is dangerous
- In noise protocols the handshake state is very small (32-64 bytes)
- Sending the state to the protocol peer is a viable course of action!
- Formalization of State Disruption Attacks covers many attacks of this style

Security proof of rosenpass

- CryptoVerif in progress (Benjamin Lipp)
- Really fast symbolic analysis using ProVerif

Deployment

- Rust implementation in userspace
- Integrates with WireGuard through the PSK feature to provide Hybrid security

Final statements

- Post-quantum crypto can be deployed now
- There are real complexities in protocol design
- DoS-Resistance needs formalization work
- Availability needs love and attention from cryptographers
- Try it out! https://rosenpass.eu/