Formosa Retreat Juli 2023
2023-07-11

Rosenpass

Wanja Zaeske, Stephan Ajuvo, Marei Peischl,
Benjamin Lipp, Lisa Schmidt, Karolin Varner

https://rosenpass.eu

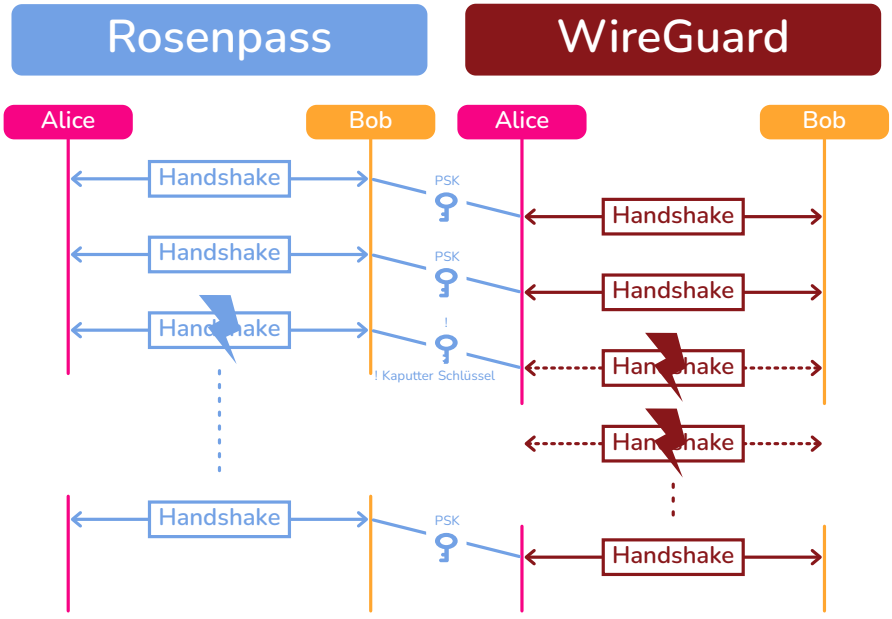
Hello, I am Karolin Varner

- Worked with about every industry tech; incl. Java Web Apps, Microcontrollers, and legacy database system from the 80s
- Did a lot of project management and some people management
- Did a lot of open-source development, privacy- and internet politics activism
- Planning to get involved in the Formosa space

Rosenpass

WireGuard	PQ WireGuard [2]	Rosenpass
✓ Session-key secrecy	✓ **Post-Quantum Security**	✓ **Non-Interruptability** [4]
✓ …	✗ **Hybrid security**	✓ **Hybrid security** [5]
✓ Identity Hiding	✗ **Non-Interruptability** [3]	
✗ **Non-Interruptability** [1]		
✗ **Post-Quantum Security**		

[1] Assuming a trusted system time
[2] Hülsing, Ning, Schwabe, Weber, Zimmermann. “Post-quantum WireGuard”. https://ia.cr/2020/379
[3] Assuming a PSK
[4] Through cookies
[5] Used together with standard WireGuard


Rosenpass
WireGuard
Alice
Bob
Alice
Bob
Handshake
PSK
Handshake
Handshake
PSK
Handshake
!
Handshake
! Kaputter Schlüssel
Handshake
Handshake
Handshake
PSK
Handshake

Rosenpass can be used right now


```
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
```

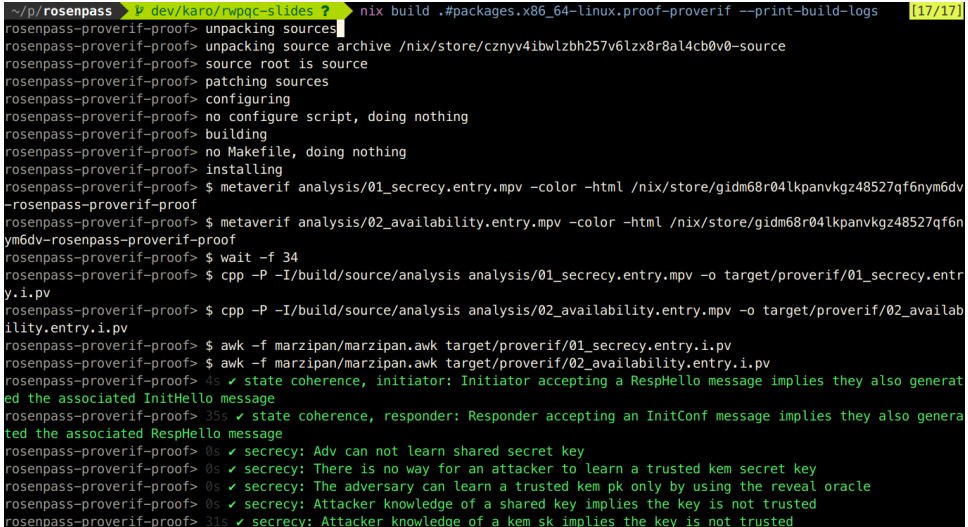
Test the connection by pinging the server on the client machine:

```
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'
```

ProVerif in Technicolor



Having worked in industry has some advantages

- Knowing how to get projects done
- Coordinating teams instead of working on my own
- Product and user focused perspective
- Building tools that I can use to be more productive
- Open-Source approach: How to catch new contributors

The spec makes it easy to implement Rosenpass

Initiator Code	Responder Code	Comments

1 — InitHello { sidi, epki, sctr, pidiC, auth } — **2** →

Line	Variables ← Action	Variables ← Action	Line	Comment
IHI1	**ck** ← lhash("chaining key init", spkr)	**ck** ← lhash("chaining key init", spkr)	IHR1	Initialize the chaining key, and bind to the responder's public key.
IHI2	**sidi** ← random_session_id();			The session ID is used to associate packets with the handshake state.
IHI3	**eski, epki** ← EKEM.keygen();			Generate fresh ephemeral keys, for forward secrecy.
IHI4	mix(sidi, epki);	mix(sidi, epki)	IHR4	InitHello includes sidi and epki as part of the protocol transcript, and so we mix them into the chaining key to prevent tampering.
IHI5	**sctr** ← encaps_and_mix<SKEM>(spkr);	decaps_and_mix<SKEM>(sskr, spkr, ct1)	IHR5	Key encapsulation using the responder's public key. Mixes public key, shared secret, and ciphertext into the chaining key, and authenticates the responder.
IHI6	**pidiC** ← encrypt_and_mix(pidi);	**spki, psk** ← lookup_peer(decrypt_and_mix(pidiC))	IHR6	Tell the responder who the initiator is by transmitting the peer ID.
IHI7	mix(spki, psk);	mix(spki, psk);	IHR7	Ensure the responder has the correct view on spki. Mix in the PSK as optional static symmetric key, with epki and spkr serving as nonces.
IHI8	**auth** ← encrypt_and_mix(empty())	decrypt_and_mix(auth)	IHR8	Add a message authentication code to ensure both participants agree on the session state and protocol transcript at this point.

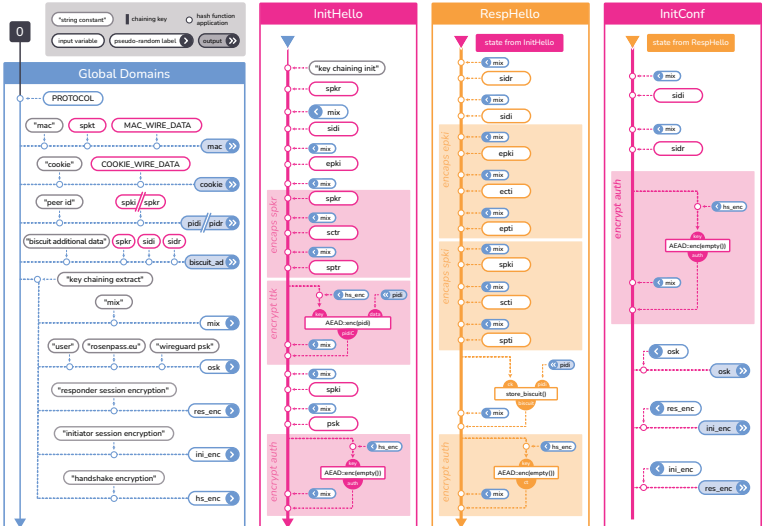
4 ← RespHello { sidr, sidi, ecti, scti, biscuit, auth } — **3**

Line	Variables ← Action	Variables ← Action	Line	Comment
RHI1		**sidr** ← random_session_id()	RHR1	Responder generates a session ID.
RHI2	**ck** ← lookup_session(sidi);		RHR2	Initiator looks up their session state using the session ID they generated.
RHI3	mix(sidr, sidi);	mix(sidr, sidi);	RHR3	Mix both session IDs as part of the protocol transcript.
RHI4	decaps_and_mix<EKEM>(eski, epki, ecti);	**ecti** ← encaps_and_mix<EKEM>(spki);	RHR4	Key encapsulation using the ephemeral key, to provide forward secrecy.
RHI5	decaps_and_mix<SKEM>(sski, spki, scti);	**scti** ← encaps_and_mix<SKEM>(spki);	RHR5	Key encapsulation using the initiator's static key, to authenticate the initiator, and non-forward-secret confidentiality.
RHI6	mix(biscuit)	**biscuit** ← store_biscuit();	RHR6	The responder transmits their state to the initiator in an encrypted container to avoid having to store state.
RHI7	decrypt_and_mix(auth)	**auth** ← encrypt_and_mix(empty());	RHR7	Add a message authentication code for the same reason as above.

5 — InitConf { sidi, sidr, biscuit, auth } — **6** →

Line	Variables ← Action	Variables ← Action	Line	Comment
ICI1		**biscuit_no** ← load_biscuit(biscuit);	ICR1	Responder loads their biscuit. This restores the state from after RHR6.
ICI2		encrypt_and_mix(empty());	ICR2	Responder recomputes RHR7, since this step was performed after biscuit encoding.
ICI3	mix(sidi, sidr);	mix(sidi, sidr);	ICR3	Mix both session IDs as part of the protocol transcript.
ICI4	**auth** ← encrypt_and_mix(empty);	decrypt_and_mix(auth);	ICR4	Message authentication code for the same reason as above, which in particular ensures that both participants agree on the final chaining key.
ICI5		assert(biscuit_no > biscuit_used);	ICR5	Biscuit replay detection.
ICI6		**biscuit_used** ← biscuit_no;	ICR6	Biscuit replay detection.

Professional illustrators create stunning graphics



Creating successful projects by knowing what not to do

- Rosenpass avoids targeting: GUIs, VPN data transport, support for many platforms
- Instead we: Created a core technology; working with companies to integrate Rosenpass (e.g. Open-Source VPN startups)
- **Vitally we** chose to focus on API; making it easy to integrate Rosenpass
- **Vitally we** integrate with the existing ecosystem (i.e. WireGuard) instead of trying to replace it

Starting partnerships...

- with Open-Source VPN companies
- with Kubernetes VPN companies
- with Quantum-Key-Distribution Projects
- to verify the Rosenpass source code
- to apply isolation features to Rosenpass (Micro-VMs)
- with university teaching departments to use the project as a simple example of bleeding-edge modern crypto

Talk to me about...

- using Rosenpass as demonstrator-project to integrate new cryptographic technologies in
- figuring out how to attract independent contributors to Formosa Projects
- applying API-focused techniques to Formosa projects to emphasize interoperability
- **Idea:** Providing XML-representations of proof assistants'[6] inputs and outputs to allow easy integration with external tools
- **Idea:** Python libraries to work with Formosa tools as everybody knows python

[6] EasyCrypt, CryptoVerif, ProVerif, Tamarin