

<https://rosenpass.eu>

<https://chaos.social/@rosenpass>

MRMCD 2023

2023-09-03



Rosenpass

Sichere Kryptografie trotz Quantencomputern: Projektupdate

Emil Engler, Stephan Ajuvo, Karolin Varner
Marei Peischl, Lisa Schmidt, Steffen Vogel, Alice Bowman, Wanja Zaeske, Sven Friedrich, Benjamin Lipp

Funding: NLNet & Prototype Fund



Was passiert im Talk?

- Was bisher geschah
- Zusammenfassung vom EH20 Talk: Was ist Rosenpass
- Was nach dem Easterhegg passiert ist
- Was wir nun vor haben
 - go-rosenpass
 - NetBird
 - Broker-Architektur & Schnittstellen zum Einbinden



Was bisher geschah

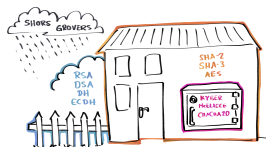
- Seit 2020: Entwicklung der Kryptografie & der Software
- Feb. 2023: Softwarerelease & Whitepaper
- März 2023: NLNet Projekt um Sicherheitsbeweis mit CryptoVerif zu erzeugen
- März 2023: Talk auf dem Real World Post-Quantum Krypto Workshop in Tokyo
- April 2023: Vorstellung/Erklärung auf dem Easterhegg¹
- Aug. 2023: Release Kandidat 0.2.0 mit FreeBSD unterstützung
- Sep. 2023: Beginn des Prototype Fund 14 Projektes für Isolation in Rosenpass

¹ <https://media.ccc.de/search/?q=rosenpass>



Warum sind Quantencomputer (k)eine Bedrohung?

- Grovers Algorithmus **schwächt** symmetrische Kryptografie
 - AES, SHA-2, SHA-3, Chacha20
 - Lösung: größere Keys
- Shors Algorithmus **bricht** asymmetrische Kryptografie
 - RSA, DSA, DH, ECDH
 - Lösung: alternative Kryptografie
- Nur auf großen Quantencomputern
 - Die existieren noch nicht
 - Problem: Store now, decrypt later



Quantencomputer überschatten Kryptografieverfahren.



PQ-sichere VPNs: WireGuard + Rosenpass

- Hybride Sicherheit
 - Bricht nur, wenn Rosenpass **und** WireGuard versagen
- Überall nutzbar, wo WireGuard schon läuft
- Ohne Anpassung vom WireGuard Source Code
 - Shared Secret aus Rosenpass = PSK für WireGuard
- Aber:
 - Ein Prozess mehr
 - Handshake alle 2 Minuten



WireGuard mit Rosenpass.



Rosenpass: Sicherheitseigenschaften

WireGuard

- ✓ Session-key secrecy
- ✓ ...
- ✓ Identity Hiding
- ✗ Non-Interruptability²
- ✗ Post-Quantum Security

PQ WireGuard³

- ✓ Post-Quantum Security
- ✗ Hybrid security
- ✗ Non-Interruptability⁴

Rosenpass

- ✓ Non-Interruptability⁵
- ✓ Hybrid security⁶



Zum Nachbauen... aus dem Whitepaper:

Initiator Code

Responder Code

Comments

1

InitHello { sisi, epki, sctr, pidiC, auth }

2

Line	Variables ← Action	Variables ← Action	Line
IHR1	$ck \leftarrow \text{thash}(\text{"chaining key init"}, \text{spkr})$	$ck \leftarrow \text{thash}(\text{"chaining key init"}, \text{spkr})$	IHR1
IHR2	$\text{sidi} \leftarrow \text{random_session_id}[];$		
IHR3	$\text{eski, epki} \leftarrow \text{EKEM-keygen}[];$		
IHR4	$\text{mix}(\text{sidi}, \text{epki});$	$\text{mix}(\text{sidi}, \text{epki})$	IHR4
IHR5	$\text{sctr} \leftarrow \text{encaps_and_mix}(\text{SKEM} > \{\text{spkr}, \text{ct1}\});$	$\text{decaps_and_mix}(\text{SKEM} > \{\text{sskr}, \text{spkr}, \text{ct1}\})$	IHR5
IHR6	$\text{pidiC} \leftarrow \text{encrypt_and_mix}(\text{pidi});$	$\text{spki, psk} \leftarrow \text{lookup_peer}(\text{decrypt_and_mix}(\text{pidiC}))$	IHR6
IHR7	$\text{mix}(\text{spki}, \text{psk});$	$\text{mix}(\text{spki}, \text{psk});$	IHR7
IHR8	$\text{auth} \leftarrow \text{encrypt_and_mix}(\text{empty}[]);$	$\text{decrypt_and_mix}(\text{auth})$	IHR8

Comment

Initialize the chaining key, and bind to the responder's public key.

The session ID is used to associate packets with the handshake state.

Generate fresh ephemeral keys, for forward secrecy.

InitHello includes sisi and epki as part of the protocol transcript, and so we mix them into the chaining key to prevent tampering.

Key encapsulation using the responder's public key. Mixes public key, shared secret, and ciphertext into the chaining key, and authenticates the responder.

Tell the responder who the initiator is by transmitting the peer ID.

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.

Add a message authentication code to ensure both participants agree on the session state and protocol transcript at this point.

4

RespHello { sidr, sisi, ecti, scti, biscuit, auth }

3

Line	Variables ← Action	Variables ← Action	Line
RHR1		$\text{sidr} \leftarrow \text{random_session_id}[];$	RHR1
RHR2	$ck \leftarrow \text{lookup_session}(\text{sidi});$		RHR2
RHR3	$\text{mix}(\text{sidr}, \text{sidi});$	$\text{mix}(\text{sidr}, \text{sidi});$	RHR3
RHR4	$\text{decaps_and_mix}(\text{EKEM} > \{\text{eski}, \text{epki}, \text{ecti}\});$	$\text{ecti} \leftarrow \text{encaps_and_mix}(\text{EKEM} > \{\text{epki}\});$	RHR4
RHR5	$\text{decaps_and_mix}(\text{SKEM} > \{\text{sski}, \text{spki}, \text{scti}\});$	$\text{scti} \leftarrow \text{encaps_and_mix}(\text{SKEM} > \{\text{spki}\});$	RHR5
RHR6	$\text{mix}(\text{biscuit});$	$\text{biscuit} \leftarrow \text{store_biscuit}[];$	RHR6
RHR7	$\text{decrypt_and_mix}(\text{auth})$	$\text{auth} \leftarrow \text{encrypt_and_mix}(\text{empty}[]);$	RHR7

Comment

Responder generates a session ID.

Initiator looks up their session state using the session ID they generated.

Mix both session IDs as part of the protocol transcript.

Key encapsulation using the ephemeral key, to provide forward secrecy.

The responder transmits their state to the initiator in an encrypted container to avoid having to store state.

Add a message authentication code for the same reason as above.

5

InitConf { sisi, sidr, biscuit, auth }

6

Line	Variables ← Action	Variables ← Action	Line
IC1		$\text{biscuit_no} \leftarrow \text{load_biscuit}(\text{biscuit});$	ICR1
IC2		$\text{encrypt_and_mix}(\text{empty}[]);$	ICR2



Comment



Responder loads their biscuit. This restores the state from after RHR7.

Responder recomputes RHR7, since this step was performed after biscuit encoding.


Zum Nachbauen... go-rosenpass – Steffen Vogel FTW



 cunicu / **go-rosenpass**


 | 





[Code](#) [Issues](#) **12** [Pull requests](#) **1** [Actions](#) [Security](#) [Insights](#)

 **go-rosenpass** [Public](#) [Watch](#) **3** [Fork](#) **1**

[main](#) [Go to file](#) [Add file](#) [Code](#) [About](#)

[Branches](#) [Tags](#)



 **stv0g** Use wg-quick configuration files rather netlink interface to ge... [...](#) [✓](#) yesterday [🕒 146](#)



	<code>.github/workflows</code>	Run CI tests with root permissions	yesterday
	<code>.reuse</code>	Update name of Go module and fix links for new re...	3 weeks ago
	<code>.vscode</code>	Implement integration / interoperability tests	4 months ago
	<code>LICENSES</code>	make repo REUSE compliant	4 months ago









[go](#) [golang](#) [vpn](#) [wireguard](#) [rosenpass](#) [Readme](#) [Apache-2.0](#) [7/15](#) [vity](#)





Zum Nachbauen... go-rosenpass – Steffen Vogel FTW


 rosenpass / **rosenpass**


 | 


 Code  **Issues** 41  Pull requests 1  Discussions  Actions  Projects  Wiki 

Whitepaper proof read #68

 Open  17 tasks **stv0g** opened this issue on May 20 · 26 comments



stv0g commented on May 20 • edited 

Member 

I've started to work on a Golang implementation of the Rosenpass key exchange^[1]. While implementing it, I stumbled over some confusing parts in the whitepaper:

Variables / message fields `mac`, `cookie`, `ini_enc` & `res_enc`

Figure 3 shows the output variables `mac` and `cookie` which apparently should be included in a message envelope (Figure 2).

But I find no mention about use of the `mac` and `cookie` variable elsewhere in the


Assignees

No one—assign

Labels

None yet

8/15

 Recent



Zum Nachbauen... go-rosenpass – Steffen Vogel FTW

Open

17 tasks

Whitepaper proof read #68



stv0g opened this issue on May 20 · 26 comments

Status

- ☐ Describe Payload transmission as a possible protocol extension
- ☐ Highlight relation to WireGuard whitepaper
- ☐ How are the roles of initiator / responder assigned?
- ☐ Endianess of biscuit counter
- ☐ Figure 4: Wrong cipher-text variable
- ☐ Section 2.4.1: Better naming for session/index table
- ☐ Inverted assertions for replay detection
- ☐ Figure 3: ☒ **Wrong PRF labels for chaining key extract/init #67**
- ☐ Figure 3: Wrong PRF labels for session encryption keys
- ☐ Section 2.3: Wrong protocol identifier
- ☐ Section 2.1.1: Wrong hash function?
- ☐ Section 2.1.1: Describe non-standard HMAC-Blake2 variant
- ☐ Figure 3 / Section 2.5: Wrong order of mixing in en/decaps_and_mix()
- ☐ Figure 2: Wrong field ordering in RespHello message



Zum Integrieren... NetBird

 SECURITY.md	Add security policy file (#600)	10 months ago
 go.mod	Routemgr error handling (#1073)	3 weeks ago
 go.sum	Routemgr error handling (#1073)	3 weeks ago

Languages

- Go 97.4%
- Other 0.2%

☰ README.md

👤 **New Release! Self-hosting in under 5 min.** [Learn more](#)



license **BSD-3**  code quality **A**

 slack @netbird

Start using NetBird at netbird.io

See [Documentation](#)

Join our [Slack channel](#)

NetBird combines a configuration-free peer-to-peer private network and a



Zum Integrieren... NetBird

☰ README.md

Together with [CISPA Helmholtz Center for Information Security](#) NetBird brings the security best practices and simplicity to private networking.



CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

Testimonials

We use open-source technologies like [WireGuard®](#), [Pion ICE \(WebRTC\)](#), [Coturn](#), and [Rosenpass](#). We very much appreciate the work these guys are doing and we'd greatly appreciate if you could support them in any way (e.g. giving a star or a contribution).

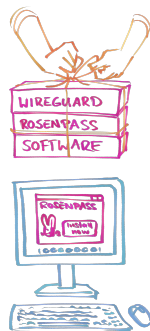
Legal

WireGuard and the *WireGuard* logo are [registered trademarks](#) of Jason A. Donenfeld.



Zum Integrieren... NetBird

- Netbird: Einfaches user interface
- Rosenpass: Hochsichere PQ-Crypto
- go-rozenpass: Um Plattformen zu unterstützen auf denen die Rust Variante schwer zu integrieren ist
 - Android
 - iOS
 - Windows
 - ...





Zum Integrieren... Prototypfund 14 Projekt

- Schnittstelle zwischen Komponenten
- Kommunikation über Unix-Sockets
- Spezielle Serialisierungsbibliothek für Schlüsseldaten⁷
- Broker-Pattern für Rosenpass– Jede Komponente in einem eigenen Prozess
- Mikro-VMs um wirklich hohe sicherheit zu haben
- Minimale Privilegien; Sandboxing

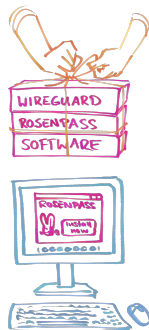


⁷ Externes Memory-Management



Rosenpass Roadmap

- Sicherheitsbeweis
- Formelle Verifikation der Implementierung
- Einfachere Benutzbarkeit
- Kryptografie + Safety Forschung:
 - Kryptografie in der Avionik
 - Decryption Despite Error





Rosenpass Strukturpläne

- Translationsforschung: Schnittstelle zwischen Industrie und Wissenschaft
- Mit mehreren Integratoren arbeiten; Open-Source R&D-Abteilung
- Antihierarchisches Arbeiten
- Karo hätte gerne mal wieder Freizeit





Idee: Kryptografie als Notar Erklären

- Problem: Kryptografie wird als schwarze Magie verstanden
- Problem: Krude vorschläge zur Verwaltungsautomatisierung
- Problem: Und Strafverfolgung
- Problem: Krypto wird auf Verschlüsselung reduziert
- Problem: Kaum jemand weiß was moderne Verfahren tun
 - Elliptic-Curve Pairings
 - Multi-Party Computation
 - Homomorphe Verschlüsselung
 - Datenbanken mit anonymem Zugriff
 - Anonyme Kommunikation



Quelle: White Rabbit aus Alice in Wonderland –
CCO

Häschen sind die
besseren
Menschen.



Idee: Kryptografie als Notar Erklären

- Krypto: Einsatzfähig für viele Prozesse in denen Information Übertragen wird
- Datenschutz: Anonyme, Nutzergesteuerte Prozesse
- Idee: Metapher von Kryptografie als Notar
 - Notare werden Bestraft wenn sie Dinge Zusichern die sie nicht können
 - Oder wenn sie gegen Regeln verstoßen
 - Spezieller schutz vor dem Recht
 - Kryptografie: Ähnlich, nur Mathematisch, statt mit Staatsgewalt



Quelle: White Rabbit aus Alice in Wonderland – CC0

Häßchen sind die
besseren
Menschen.