

# 7-1: Overview of the cryptographic Rust ecosystem (w/o RustCrypto)

---

Artem Pavlov, TII, Abu Dhabi, 11.05.2024

---

# rage

---

- Rust implementation of the age file encryption tool
- Supports password and public key based encryption
- Supports multiple recipients for one encrypted file
- Can use SSH keys (RSA and Ed25519)
- Repository: <https://github.com/str4d/rage>

# openssl

---

- Bindings to OpenSSL
- Supports static linking against OpenSSL using the **vendored** crate feature
- Detects OpenSSL version at compile time and changes exposed public API based on that
- Docs: <https://docs.rs/openssl>

# ring

---

- Cryptographic library based on BoringSSL
- Hybrid of Rust, C, and assembly code
- Provides only a limited set of cryptographic primitive
- Does not expose low-level primitives
- Docs: <https://docs.rs/ring>

# aws-lc-rs

---

- Binding to the **AWS-LC** cryptographic library written in C
- AWS-LC is FIPS 140-3 certified
- Provides **ring**-compatible API
- Docs: <https://docs.rs/aws-lc-rs>

# sodiumoxide

---

- Binding to [libsodium](#), easy-to-use, opinionated cryptographic library
- Warning: the library development has stopped
- Docs: <https://docs.rs/sodiumoxide>
- Pure Rust alternative:  
<https://github.com/RustCrypto/nacl-compat>

# secp256k1

- Bindings to the **libsecp256k1** library written in C
- Used in the **bitcoin** crate
- Docs: <https://docs.rs/secp256k1>

# evercrypt-rust

---

- Bindings to `evercrypt/HACL*` cryptographic library
- Warning: uses the `aes` crate as a fallback on unsupported targets
- Docs:  
<https://www.franziskuskiefer.de/evercrypt-rust/evercrypt/index.html>



# sequoia

- Rust implementation of OpenPGP
- Provides a simple CLI frontend **sequoia-sq** (**sq**)
- Licensed under LGPLv2
- Repository:  
<https://gitlab.com/sequoia-pgp/sequoia>

- Bindings to the **Nettle** cryptographic library written in C
- Both Rust crate and the C library are licensed under GPLv2 or GPLv3 or LGPLv3
- Docs: <https://docs.rs/nettle>

# Dalek crates

---

- Pure Rust implementation of public-key algorithms based on Ristretto and Curve25519
- Exposes 3 crates:
  - `curve25519-dalek`: group operations on the Edwards and Montgomery forms of Curve25519, and on the prime-order Ristretto group
  - `ed25519`: key generation, signing, and verification
  - `x25519`: elliptic curve Diffie-Hellman key exchange
- Repository: <https://github.com/dalek-cryptography/curve25519-dalek>

# snow

---

- Implementation of the Noise protocol
- By default uses RustCrypto crates
- Optionally can use ring
- Docs: <https://docs.rs/snow>
- Repository: <https://github.com/mcginty/snow>

# native\_tls

---

- An abstraction over platform-specific TLS implementations:
  - SChannel on Windows
  - Secure Transport on OSX
  - OpenSSL on all other platforms
- Docs: <https://docs.rs/native-tls>

# rustls

---

- Implementation of TLSv1.2 and TLSv1.3
- By default uses **aws-lc-rs**, but also supports alternative providers such as **ring**
- RustCrypto provider (in development):  
<https://github.com/RustCrypto/rustls-rustcrypto>
- Performance of **rustls** vs OpenSSL:
  - <https://jbp.io/2019/07/02/rustls-vs-openssl-bulk-performance.html>
  - <https://jbp.io/2019/07/02/rustls-vs-openssl-handshake-performance.html>
  - <https://jbp.io/2019/07/02/rustls-vs-openssl-memory-usage.html>
  - <https://jbp.io/2019/07/02/rustls-vs-openssl-resumption-performance.html>

Questions?