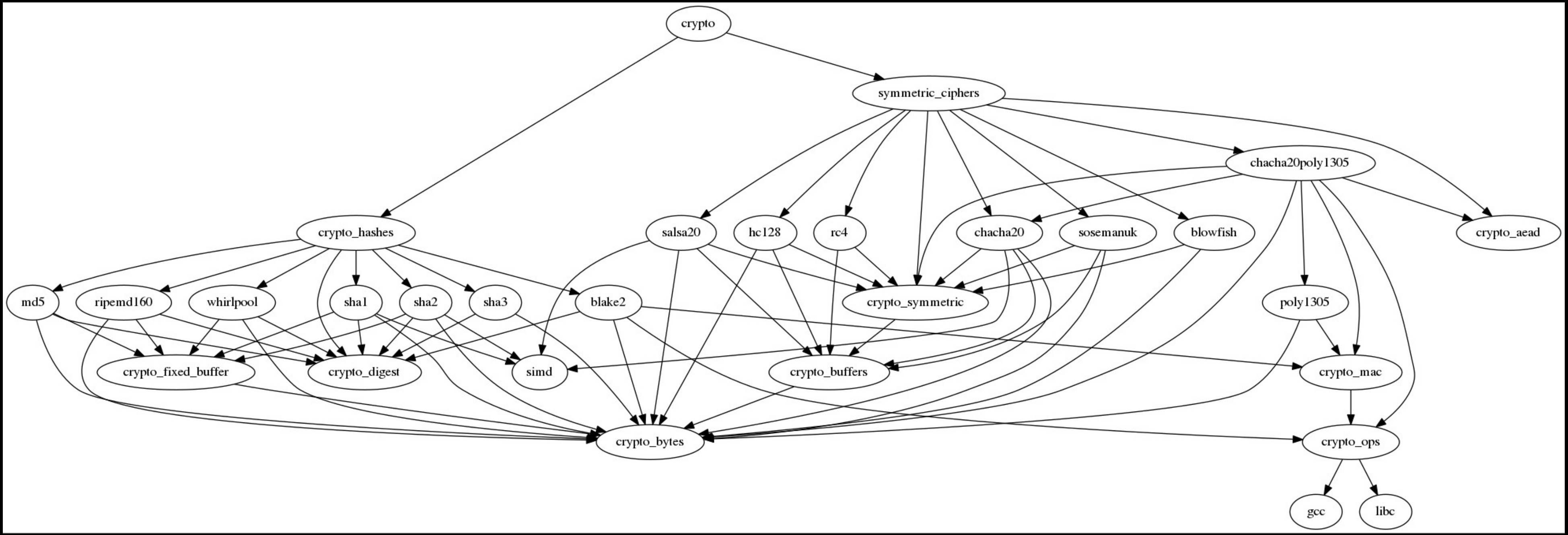
7-3: RustCrypto

Artem Pavlov, TII, Abu Dhabi, 11.05.2024

Origins

- RustCrypto started as a hobby project in 2016
- It was an experiment in modularization of the rust-crypto crate
- rust-crypto was abandoned at the time for more than a year
- After initial development in a private repository, the RustCrypto GitHub org was created
- A number of people have joined the project since, including my co-lead Tony Arcieri

rust-crypto-decoupled



Goals of the RustCrypto project

- Development of a cryptographic ecosystem of crates developed in pure Rust
- All crates are licensed under MIT OR Apache-2.0
- Highly modular architecture (every algorithm lives in its own crate)
- Introduction of cryptographic traits for improving interoperability and genericity of higher-level crates

RustCrypto/traits

- Collection of cryptography-related traits
- Contains a number of crates which primarily define traits and helper types for a class of cryptographic algorithms
- Repository: https://github.com/RustCrypto/traits

typenum

- Type-level numbers evaluated at compile time
- A “stop-gap” solution for working around const generics limitations
- Docs: https://docs.rs/typenum

generic-array

- An alternative to built-in arrays built on top of typenum
- Allow to write generic code over array sizes
- Docs: https://docs.rs/generic-array

digest

- Traits for cryptographic hash functions and message authentication algorithms
- Docs: https://docs.rs/digest
- Implementations:
  - Hash functions: https://github.com/RustCrypto/hashes
  - MACs: https://github.com/RustCrypto/MACs

hkdf

- Generic implementation of HKDF using the hmac crate and traits from digest
- Docs: https://docs.rs/hkdf

cipher

- Traits for block modes, block and stream ciphers
- Docs: https://docs.rs/cipher
- Implementations:
  - Block ciphers: https://github.com/RustCrypto/block-ciphers
  - Block modes: https://github.com/RustCrypto/block-modes
  - Stream ciphers: https://github.com/RustCrypto/stream-ciphers

universal-hash

- Traits for Universal Hash Functions
- Usually used as a building block for MAC (e.g. GMAC)
- Docs: https://docs.rs/universal-hash
- Implementations: https://github.com/RustCrypto/universal-hashes

aead

- Authenticated Encryption with Associated Data (AEAD) traits
- Additionally provides generic implementation of the STREAM construction
- Docs: https://docs.rs/aead
- Implementations: https://github.com/RustCrypto/AEADs

password-hash

- Traits for password hashing algorithms
- Provides support for “Password Hashing Competition (PHC) string” encoding and decoding of password hashing results
- Docs: https://docs.rs/password-hash
- Implementations: https://github.com/RustCrypto/password-hashes

rsa

- Implementation of RSA
- Supports different RSA padding shemese
- Supports PKCS#1 and PKCS#8 key encodings using pkcs1 and pkcs8 crates
- Warning: susceptible to the Marvin attack
- Docs: https://docs.rs/rsa

elliptic-curve

- General purpose Elliptic Curve Cryptography (ECC) support, including types and traits for representing various elliptic curve forms, scalars, points, and public/secret keys composed thereof
- Provides generic ECDH implementation
- Docs: https://docs.rs/elliptic-curve
- Implementations: https://github.com/RustCrypto/elliptic-curves

signature

- Generic, object-safe traits for generating and verifying digital signatures
- Docs: https://docs.rs/signature
- Implementations: https://github.com/RustCrypto/signatures

kem

- Traits for key encapsulation mechanisms, i.e. algorithms for non-interactively establishing secrets between peers
- Docs: https://docs.rs/kem
- Implementations: https://github.com/RustCrypto/KEMs

Questions?