Implementing Signal

Yi Tang, Yevgeniy Dodis
New York University

August 20, 2019
Generic Signal Protocol: Specification and Implementation

About
The (original) Signal protocol is an open-source secure messaging protocol that provides end-to-end authenticated encryption with forward security, post-compromise security, untraceability and many other appealing security advantages. The protocol is thoroughly described and discussed in the paper "The Double Ratchet: Security Notions, Threats, and Mitigations for the Signal Protocol" by M. J. Bellovin, D. Crispin, Y. Yeh, and V. Shoup. The paper proposes a decomposition of the double ratchet algorithm into multiple generic cryptographic modules. The modularization enables customization of the algorithms using different instances of the modules, which naturally leads to post-quantum variants of the Signal protocol by employing quantum-safe modules instead.

The implementation of the generic Signal protocol follows the modularization and is provided as a C library.

Specification
In [2] it is described how the generic Signal protocol can be decomposed into constituent key agreement (CKA), PREP (HMAC), authenticated encryption with associated data (AEAD) and PRF, as well as how to construct CKA from any key establishment mechanism (KEM) and PRF from HMAC. [2] describes a standard construction of HMAC from KMAC.

[2] mentions the canonical "encrypt-then-MAC" construction of AEAD from KMAC and HMac.

By definition there exists a trivial construction of PRF from PRF, namely \( \text{PRF} \left( \text{PRF} \left( k, x \right), x \right) \) or \( \text{PRF} \left( k, x \right) \) or \( \text{PRF} \left( k, x \right) \).

References

Implementation
The C implementation follows the specification of the modularization.

Refer to the README and code archive for details about the usage as well as the functionality of the library.

Downloads
- Version 0.1.0

Release Notes
- 3.1.0: Added support for the generic Signal protocol as well as the modules and their instances including CKA (implemented with CCA and CCA2) and associated data (AEAD)
- 3.1.0: Added support for the generic Signal protocol as well as the modules and their instances including CKA (implemented with CCA and CCA2) and associated data (AEAD)
- 3.0.0: Added support for the generic Signal protocol as well as the modules and their instances including CKA (implemented with CCA and CCA2) and associated data (AEAD)

Dependencies
- system C++ library of cryptographic primitives
- OpenSSL
- C implementation of FrodoKEM, a post-quantum KEM scheme whose security derives from the learning with error problem

Contributors
- Yi Tang

Figure: Screenshot of project homepage.
# Table of Contents

**Introduction**
- (Original) Signal Protocol
- Generalization of Signal Protocol

**Specification**
- Modularization of Signal Protocol
- Constructions of Misc. Modules

**Implementation**
- Hierarchical Overview of Implementation
- Examples of Module Instances

**Benchmarking**
- Time Benchmarking
- Space Benchmarking

**References**
The *Signal protocol* is an open source secure messaging protocol that provides end-to-end authenticated encryption. The protocol is extensively deployed among popular messaging applications e.g.

- Signal (originally TextSecure),
- WhatsApp,
- Google Allo,
- Facebook Messenger,
- Skype, etc.
Generalization of Signal Protocol

- [ACD19] decomposes the *double ratchet algorithm* (core of Signal) into generic cryptographic modules.
- Customize Signal by using different module instances.
- Create *post-quantum* variants of Signal by employing quantum-safe module instances!
Modularization of Signal Protocol

**Figure:** The Modularization of Signal protocol proposed by [ACD19], involving “CKA”, “P”, “G” and (“Enc”, “Dec”). (Figure 9 in [ACD19].)
Modularization of Signal Protocol (cont.)

The Signal protocol is decomposed into
- “CKA”: Continuous Key Agreement (CKA),
- “P”: PRF-PRNG (PRGF),
- “G”: Pseudo-Random Generator (PRG),
- (“Enc”, “Dec”): Authenticated Encryption with Associated Data (AEAD).
Constructions of Miscellaneous Modules

- CKA from *Key Encapsulation Mechanism* (KEM) ([ACD19]),
- PRGF from Hash-based Key Derivation Function (HKDF) ([ACD19]);
- HKDF from Hash-based Message Authentication Code (HMAC) (e.g. see [KE10]),
- AEAD from Secret/Symmetric Key Encryption (SKE) along with HMAC (e.g. see [McG08]),
- PRG from PRF (trivial according to definitions).
Hierarchical Overview of Implementation

Bottom-up:

- Abundant module instances of CKA, PRGF, PRG, AEAD (and also KEM, HKDF, HMAC, PRF, SKE)
  - where sub-hierarchy exists according to the specification
  - e.g. CKAs can be generically constructed from KEMs
- High-level Signal protocol interface, providing e.g. `send` and `recv` functionalities
- Top-level testing shell, hiding back all Signal-related details

All components are implemented in C.
Examples of Module Instances

Example 1: reconstructing the original Signal Protocol

- CKA: (compressed) Diffie–Hellman with Curve25519
- PRGF: [HKDF-based]
  - HKDF: SHA-256
- PRG: [PRF-based]
  - PRF: SHA-256
- AEAD: AES-128-SIV
Example 2: *post-quantum* variant of the Signal Protocol

- **CKA**: [KEM-based]
  - **KEM**: Frodo640\(^1\)
- **PRGF**: [HKDF-based]
  - **HKDF**: SHA-256
- **PRG**: [PRF-based]
  - **PRF**: SHA-256
- **AEAD**: AES-128-SIV

---

\(^1\)FrodoKEM, a post-quantum KEM scheme whose security derives from the *learning with error* problem
Example 3: *improved* post-quantum variant of the Signal Protocol

- CKA: (compressed) Frodo640
- PRGF: [HKDF-based]
  - HKDF: SHA-256
- PRG: [PRF-based]
  - PRF: SHA-256
- AEAD: AES-128-SIV
Figure: Comparison of time performance of example Signal variants under random asynchronous messaging test.
Figure: Comparison of message-packet size ratio of example Signal variants.
References


I’m just here for the signal