Keccak Utilities

A Java library providing a dynamic set of cryptographic services based on the Keccak primitive.

Overview

Keccak Utilities provides a range of cryptographic functions:

- SHA3/cSHAKE256/KMACXOF256 hash computation
- Symmetric encryption via KMACXOF256
- EC key generation, asymmetric crypto, and Schnorr signatures

The utilities above are built on top of a NIST compliant Keccak implementation. Compliance testing was done in the style of NIST's Cryptographic Algorithm Validation Program using the suite of test vectors available on the NIST CAVP web page.

Available on Github: https://github.com/NWc0de/KeccakUtils

Use Cases

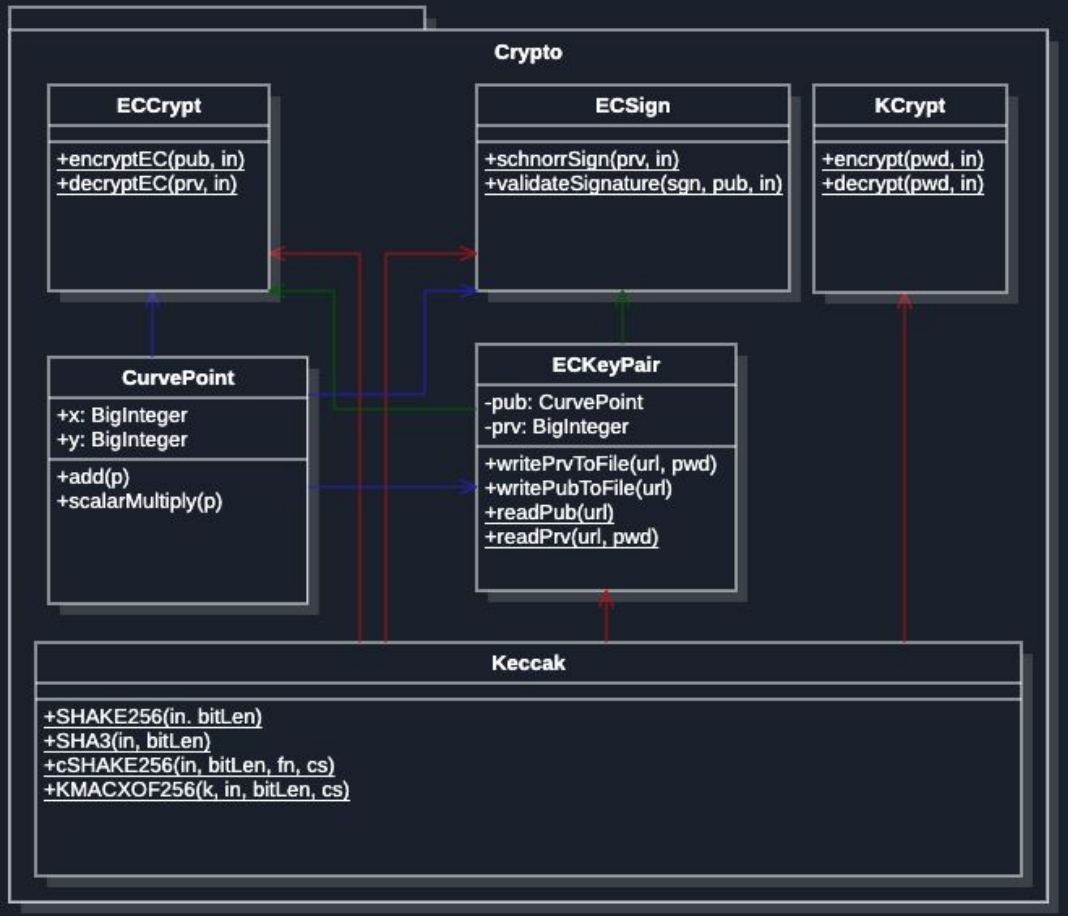
01 Providing a reliable integrated encryption and authentication scheme for Java applications that require symmetric encryption. A prudent choice for projects where longevity is required due to Keccak's innate quantum resistance.

02 Enabling general users who have some knowledge of the command line to verify downloads when a SHA3 hash is provided. Or to encrypt files locally under a user friendly password.

03 Providing asymmetric crypto services (signing code/documents, constructing cryptographic envelopes, etc.) given an appropriate PKI.

Class Structure

- CurvePoint uses BigInteger style arithmetic
- Almost all classes use Keccak methods
- ECKeyPair manages key creation, encryption, storage



NIST Compliance

- NIST provides conformance testing guidelines through CAVP (Cryptographic Algorithm Validation Program)
- True NIST validation requires submitting algorithm to an accredited lab, although the tests they perform are made publicly available
- For the Keccak family of functions SHA3VS is used to verify conformance (can be reached by searching for NIST CAVP then going to the SHA3 section)
- NIST provides suites of test vectors that are designed to verify the functioning of certain aspects of SHA3/SHAKE (extensible output length, proper output, etc.)
- Types of tests: short message, long message, Monte Carlo, variable output (only for SHAKE)
- This library implements SHA3 and SHAKE256 CAPV tests (short message, long message, Monte Carlo, and variable output)

Future Work

- Standardized public/private EC key file format (.pem)
- Support for other curves, to enable interoperability
- Support for standardized protocols (ECDSA)
- Encrypted archival functionality w/ KMACXOF
    - Compress then encrypt or encrypt then compress?
    - Encrypting before makes compression useless (no redundancy in ciphertext for compression)
    - Compressing before can leak data about the plaintext via the size of the file (CRIME, BEAST)
    - Idea is to use KMACXOF256 to encrypt the compressed data, then add some amount of extra psuedorandom bits from the sponge so that detecting the actual size of the encrypted archive is difficult

Thank you!

Project Github:
https://github.com/NWc0de/KeccakUtils