Apache Shiro Cryptography Features

Cryptography is the practice of protecting information from undesired access by hiding it or converting it into nonsense so know one else can read it. Shiro focuses on two core elements of Cryptography: ciphers that encrypt data like email using a public or private key, and hashes (aka message digests) that irreversibly encrypt data like passwords.

Shiro Cryptography's primary goal is take what has traditionally be an extremely complex field and make it easy for the rest of us while providing a robust set of cryptography features.

Simplicity Features

- **Interface-driven, POJO based** - All of Shiro's APIs are interface-based and implemented as POJOs. This allows you to easily configure Shiro Cryptography components with JavaBeans-compatible formats like JSON, YAML, Spring XML and others. You can also override or customize Shiro as you see necessary, leveraging its API to save you time and effort.
- **Simplified wrapper over JCE** - The Java Cryptography Extension (JCE) can be complicated and difficult to use unless you're a cryptography expert. Shiro's Cryptography APIs are much easier to understand and use, and they dramatically simplify JCE concepts. So now even Cryptography novices can find what they need in minutes rather than hours or days. And you won't sacrifice any functionality because you still have access to more complicated JCE options if you need them.
- **“Object Orientifies” cryptography concepts** - The JDK/JCE's Cipher and Message Digest (Hash) classes are abstract classes and quite confusing, requiring you to use obtuse factory methods with type-unsafe string arguments to acquire instances you want to use. Shiro 'Object Orientifies' Ciphers and Hashes, basing them on a clean object hierarchy, and allows you to use them by simple instantiation.
- **Runtime Exceptions** - Like everywhere else in Shiro, all cryptography exceptions are RuntimeExceptions. You can decide whether or not to catch an exception based on your needs.

Cipher Features

- **OO Hierarchy** - Unlike the JCE, Shiro Cipher representations follow an Object-Oriented class hierarchy that match their mathematical concepts: `AbstractSymmetricCipherService`, `DefaultBlockCipherService`, `AesCipherService`, etc. This allows you to easily override existing classes and extend functionality as needed.
- **Just instantiate a class** - Unlike the JCE's confusing factory methods using String token arguments, using Shiro Ciphers are much easier - just instantiate a class, configure it with JavaBeans properties as necessary, and use it as desired. For example, `new AesCipherService()`.
- **More secure default settings** - The JCE Cipher instances assume a 'lowest common denominator' default and do not automatically enable more secure options. Shiro will automatically enable the more secure options to ensure your data is as safe as it can be by default, helping you prevent accidental security holes.

Hash Features

- **Deault interface implementations** - Shiro provides default Hash (aka Message Digests in the JDK) implementations out-of-the-box, such as MD5, SHA1, SHA-256, et al. This provides a type-safe construction method (e.g. `new Md5Hash(data)`) instead of being forced to use type-unsafe string factory methods in the JDK.
- **Built-in Hex and Base64 conversion** - Shiro Hash instances can automatically provide Hex and Base-64 encoding of hashed data via their `toHex()` and `toBase64()` methods. So now you do not need to figure out how to correctly encode the data yourself.
- **Built-in Salt and repeated hashing support** - Salts and repeated hash iterations are very valuable tools when hashing data, especially when it comes to protecting user passwords. Shiro's Hash implementations support salts and multiple hash iterations out of the box so you don't have to repeat this logic anywhere you might need it.

Get Started in 10 Minutes with Shiro

Try out Shiro for yourself with our 10 Minute Tutorial. And if you have any questions about Shiro, please check out our community forum or user mailing list for answers from the community.