Keyczar: A Cryptographic Toolkit

Arkajit Dey[1] and Stephen Weis[2]

[1] Stanford University, Stanford, CA, USA 94305
[2] Google Inc., Mountain View, CA, USA 94043

Abstract. Keyczar's goal is to make it easier for application developers to safely use cryptography. Keyczar defaults to safe algorithms, key lengths, and modes, and prevents developers from inadvertently exposing key material. It uses a simple, extensible key versioning system that allows developers to easily rotate and retire keys.

1 Introduction and Philosophy

The motivation for Keyczar grew out of a need to make cryptography easier to use for developers. Developers often can make simple mistakes when using cryptography that can create security vulnerabilities. For instance, developers may use obsolete algorithms, weak key lengths, improper cipher modes, or unsafely compose cryptographic operations. Another common developer mistake is to fail to provision for key rotation or even to hard-code keys in source code.

Keyczar's goal is to address these issues by providing a simple application programming interface (API) for developers that handles basic cryptographic details. Keyczar also provides a simple key versioning and management system based on directories of human-readable flat files, which will be refered to as *keysets*.

2 Using KeyczarTool

All Keyczar keys are generated with the stand-alone `KeyczarTool` utility located in the `com.google.keyczar` Java package.

2.1 KeyczarTool create

KeyczarTool must first create a new keyset using the `create` command. A newly created keyset will initially contain just a metadata file, described in section 5.1.

`KeyczarTool create` requires `location` and `purpose` command-line flags that specify the location of the key set and its purpose. Valid purposes are currently `crypt` and `sign`. The create command may also take an optional `name` flag to give a newly created keyset a name. If the `asymmetric` flag is specified, the newly created set will contain asymmetric keys.

Some example `create` commands:

- Create a symmetric signing (HMAC) keyset:
```
KeyczarTool create --location=/path/to/keyset --purpose=sign
```
- Create a symmetric crypting (AES) keyset named "Test":
```
KeyczarTool create --location=/path/to/keyset --purpose=crypt --name=Test
```
- Create an asymmetric signing (DSA) keyset:
```
KeyczarTool create --location=/path/to/keyset --purpose=sign --asymmetric
```

2.2 KeyczarTool addkey

All Keyczar keys are created using the `addkey` command. This command requites a keyset `location` flag and may optionally have `status` and `size` flags. Section 5.4 describes the meaning of the status values, but briefly they are *primary*, *active*, and *scheduled_for_revocation*. The default status is *active*. User-specified key sizes are supported, although it is recommneded that on default or larger key sizes are used.

The `addkey` command will create a new file in the keyset directory with an integer version number that is one greater than the currently largest version. Version numbers start from 1 and are described in Section 5.3. For example, if the current keyset contains the key file `1`, a new key version will be created in the file `2`. Some example `addkey` commands:

- Create a new primary key:
```
KeyczarTool addkey --location=/path/to/keyset --status=primary
```
- Create a new active key:
```
KeyczarTool addkey --location=/path/to/keyset
```

2.3 KeyczarTool pubkey

Public keys in Keyczar are exported from existing asymmetric key sets. The `pubkey` command requires both an existing `location` flag and a `destination` where public keys will be exported. If the keyset under `location` was not created with an `asymmetric` flag, then a `pubkey` command will fail. An example `pubkey` command works as follows:

```
KeyczarTool pubkey --location=/path/to/keyset --destination=/path/to/dest
```

2.4 KeyczarTool promote, demote, and revoke

The `promote`, `demote`, and `revoke` commands are used to change key status values. Each of these commands require a `location` and `version` flag.

Promoting an *active* key will raise its status to *primary*, and promoting a *scheduled_for_revocation* status will make it *primary*. There can only be a single *primary* key in a given key set.

Similarly, `demote` will lower a *primary* key to *active*, and an *active* key to *scheduled_for_revocation*. The `revoke` command will only work for *scheduled_for_revocation* status values. The `revoke` command will permenantly delete key material, so should be used with caution.

Some example `promote`, `demote`, and `revoke` commands. Suppose that key version 1 initially has an *active* status:

- Promote *active* version 1 to *primary*:
  `KeyczarTool promote --location=/path/to/keyset --version=1`
- Demote *primary* version 1 back to *active*:
  `KeyczarTool demote --location=/path/to/keyset --version=1`
- Demote *active* version 1 to *scheduled_for_revocation*:
  `KeyczarTool demote --location=/path/to/keyset --version=1`
- Revoke the *scheduled_for_revocation* version 1:
- `KeyczarTool revoke --location=/path/to/keyset --version=1`

3 Using Java Keyczar

4 Using Python Keyczar

5 Keys

5.1 Key Metadata

5.2 Identifying Headers

5.3 Versions

5.4 Statuses

5.5 Formats

HMAC

AES

RSA

DSA

6 Output Formats

6.1 Ciphertext

6.2 Signatures

7 Licenses and Dependencies

Keyczar is available under an Apache 2.0 license [1]. Java Keyczar depends on the Google GSON package [2]. It also relies on the Java's `javax.crypto` package, which may not be available in all locales due to local laws and regulations.

Python Keyczar depends on the Python Cryptography Toolkit [4] and simplejson [3].

8 Acknowledgements

Thanks to Ben Laurie, Marius Schilder, and Neil Daswani for their contributions.

References

1. Apache Software Foundation. Apache license, version 2.0. http://apache.org/licenses/LICENSE-2.0.
2. Google Inc. Google GSON project. http://code.google.com/p/google-gson/.
3. Ippolito, B. simplejson. http://pypi.python.org/pypi/simplejson.
4. Kuchling, A. Python crypto toolkit. http://www.amk.ca/python/code/crypto.html.