Go JOSE

Package jose aims to provide an implementation of the Javascript Object Signing and Encryption set of standards. This includes support for JSON Web Encryption, JSON Web Signature, and JSON Web Token standards.

Disclaimer: This library contains encryption software that is subject to the U.S. Export Administration Regulations. You may not export, re-export, transfer or download this code or any part of it in violation of any United States law, directive or regulation. In particular this software may not be exported or re-exported in any form or on any media to Iran, North Sudan, Syria, Cuba, or North Korea, or to denied persons or entities mentioned on any US maintained blocked list.

Overview

The implementation follows the JSON Web Encryption (RFC 7516), JSON Web Signature (RFC 7515), and JSON Web Token (RFC 7519). Tables of supported algorithms are shown below. The library supports both the compact and full serialization formats, and has optional support for multiple recipients. It also comes with a small command-line utility (`jose-util`) for dealing with JOSE messages in a shell.

Note: We use a forked version of the `encoding/json` package from the Go standard library which uses case-sensitive matching for member names (instead of case-insensitive matching). This is to avoid differences in interpretation of messages between go-jose and libraries in other languages.

Versions

We use gopkg.in for versioning.

Version 2 (branch, doc) is the current version:

```
import "gopkg.in/square/go-jose.v2"
```

The old `v1` branch (go-jose.v1) will still receive backported bug fixes and security fixes, but otherwise development is frozen. All new feature development takes place on the `v2` branch. Version 2 also contains additional sub-packages such as the jwt implementation contributed by @shaxbee.

Supported algorithms

See below for a table of supported algorithms. Algorithm identifiers match the names in the JSON Web Algorithms standard where possible. The Godoc reference has a list of constants.

Key encryption	Algorithm identifier(s)
RSA-PKCS#1v1.5	RSA1_5
RSA-OAEP	RSA-OAEP, RSA-OAEP-256
AES key wrap	A128KW, A192KW, A256KW
AES-GCM key wrap	A128GCMKW, A192GCMKW, A256GCMKW
ECDH-ES + AES key wrap	ECDH-ES+A128KW, ECDH-ES+A192KW, ECDH-ES+A256KW

ECDH-ES (direct)	ECDH-ES[1]
Direct encryption	dir[1]

1. Not supported in multi-recipient mode

Signing / MAC	Algorithm identifier(s)
RSASSA-PKCS#1v1.5	RS256, RS384, RS512
RSASSA-PSS	PS256, PS384, PS512
HMAC	HS256, HS384, HS512
ECDSA	ES256, ES384, ES512
Ed25519	EdDSA[2]

2. Only available in version 2 of the package

Content encryption	Algorithm identifier(s)
AES-CBC+HMAC	A128CBC-HS256, A192CBC-HS384, A256CBC-HS512
AES-GCM	A128GCM, A192GCM, A256GCM

Compression	Algorithm identifiers(s)
DEFLATE (RFC 1951)	DEF

Supported key types

See below for a table of supported key types. These are understood by the library, and can be passed to corresponding functions such as `NewEncrypter` or `NewSigner`. Each of these keys can also be wrapped in a JWK if desired, which allows attaching a key id.

Algorithm(s)	Corresponding types
RSA	*rsa.PublicKey, *rsa.PrivateKey
ECDH, ECDSA	*ecdsa.PublicKey, *ecdsa.PrivateKey
EdDSA[1]	ed25519.PublicKey, ed25519.PrivateKey
AES, HMAC	[]byte

1. Only available in version 2 of the package

Examples

Examples can be found in the Godoc reference for this package. The `jose-util` subdirectory also contains a small command-line utility which might be useful as an example.