Refactored AC Lib

Shirley Crompton (UKRI-STFC)

shirley.crompton@stfc.ac.uk

22 May, 2019

Refactored AC library Overview (1/2)

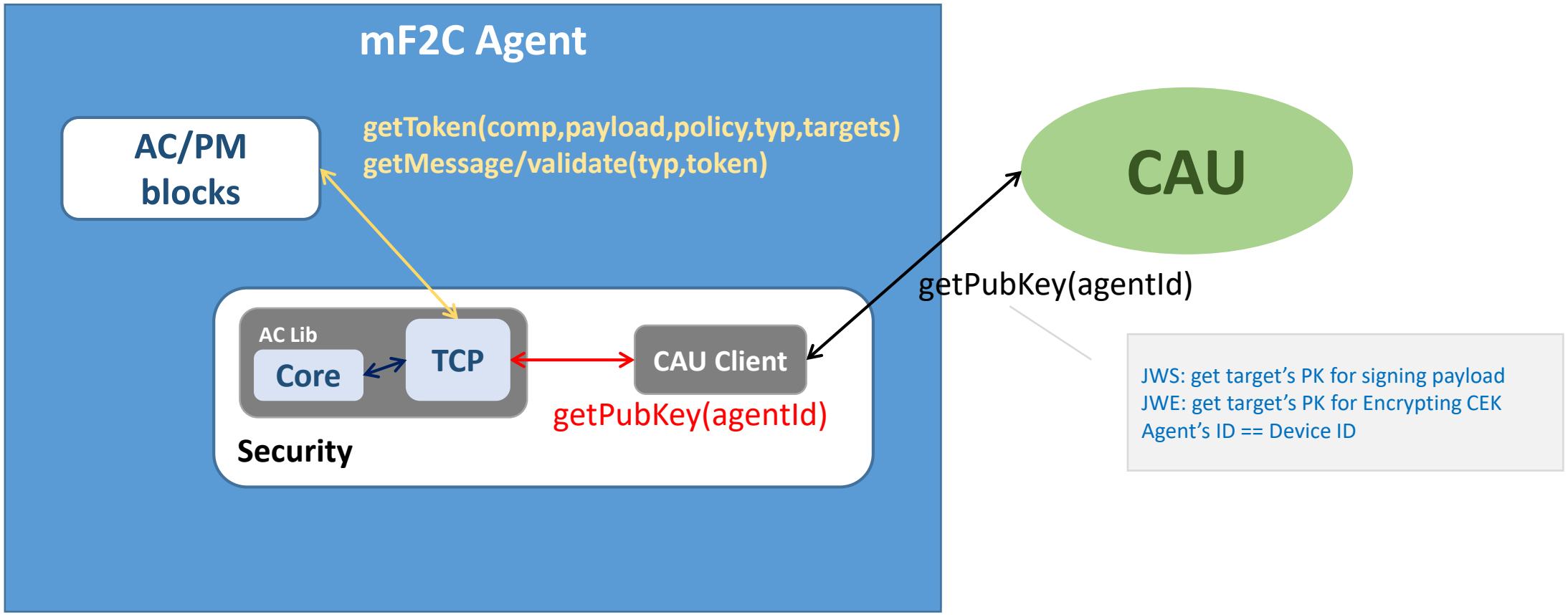
Key features

- Light-weight JAVA implementation, dependent on JRE native security and JOSE4J (NO bouncycastle artefacts)
- Leverage mF2C PKI and CAU security middleware
- Communication with AC Lib is via existing private Docker's network

Objective

- Implement mF2C security policy for message protection (D2.5):
  - public - for data not requiring protection
  - Protected - for data which needs to be integrity protected but is not confidential
  - Private - for data which needs both integrity and confidentiality protection
- Support creation and validation of Agent identity JWT (D3.4) (added in May 2019)

Refactored AC library Overview (2/2)



Input/Output to/from the AC Lib (1/2)

GetToken : Input An UTF8 encoded Json String

“sec”:”pub”,	security policy : public, protected, private (not required if typ=jwt)
“comp”:”t”,	plaintext payload compression flag : true, false (not required if typ=jwt)
“recs”:{“rec1”,”rec2”,”rec3”},	*recipient/s : (comma-separated list of Agent IDs)
“payload”:”payload plaintext”,	payload : plaintext payload (not required if typ=jwt)
“typ”:”jws”	typ : plain, jws, jwe, jwt (mandatoary)
“\n”	!!! Important!!! Terminate input by sending EOF signal
	* Only required for Private messages and JWT.

UTF8({/”sec/”: /”pro/”, /”comp/”:/”f/”, /”recs/”:{[/”rec1/”, /”rec2/”]}, /”payload/”: /”plaintext/”,/”typ/”:/”jws/”})

GetToken : Output a Base64Url encoded String representation of the Token

eyJhbGciOiJub25lIiwidHlwIjoiSk9TRSIsIm1mMmMtc2VjIjoicHVibGljIiwibWYyYy10bXNwIjoiMTU1MzYwMzgyODcyNCIsIm1mMmMtc2VuZGVyIjoiMGY4NDhkOGZiNzhjYmU1NjE1NTA3ZWY1YTE5OGY2NjBhYzg5YTNhZTAzYjk1ZTc5ZDRlYmZiMzQ2NmMyMGQ1NGU5YTVkOWI5YzQxZjg4Yzc4MmQxZjY3YjMyMjMxZDMxYjRmYWRhOGQyZjlkZDMxYTRkODg0NjgxYjc4NGVjNWEiLCJtZjJjLWFjbGliIjoiMS4wIn0.V2VsbCwgYXMgb2YgdGhpcyBtb21lbnQsIHRoZXkncmUgb24gRE9VQkxFIFNFQ1JFVCBQUk9CQVRJT04hICtEbyB5b3UgcmVhbGx5IGNhcmU_ICBJIGRvbid0LiBUaGUgc2lsdmVyIGZveCBxdWlja2x5IGp1bXB0ZWQgb3ZlciB0aGUgZmVuY2UuICBBcyBsb25nIGFzIGl0IGRvZXNuJ3QgZWF0IG15IGNoaWNrZW5zLi4uLkl0IGNhbiBnbyBhcyBpdCBpdCBwbGVhc2VzLiA6RA
.

(See slides on token specifications)

Input/Output to/from the AC Lib (2/2)

GetMessage/Validate : Input An UTF8 encoded Json String

“typ”:”jws”, “token”:”eyJhbGciOiJub….”	typ : plain, jws, jwe, jwt (mandatory) token : a Base64Url encoded String representation of the token (mandatory)
“\n”	!!! Important!!! Terminate input by sending EOF signal

UTF8({/”typ/”: /”jws/”, /”token/”: /”eyJhbGciOiJub…./”})

GetMessage : Output the payload plaintext (on success)

"Well, as of this moment, they're on DOUBLE SECRET PROBATION! Do you really care? I don't. The silver fox quickly jumped over the fence. As long as it doesn't eat my chickens.... It can go as it pleases. :D"

Validate : Output ‘OK’ message (on success)

“OK"

GetMessage/validate : Output error code

“err1”	: integrity error for JWE, signed JWS and JWT
“err2”	: credential error for JWE
“err3”	: client errors, e.g. typ mismatch, missing params, JWT expired etc.
“err4”	: other errors

Token specifications:

Both are formatted as a 3-part JWS Compact Serialization (RFC7515) String

Public Message (Unsigned JWS)

BASE64URL(UTF8(JWS Protected Header)) || ‘.’ ||
BASE64URL(JWS Payload) || ‘.’ ||

Structure

eyJhbGciOiJub25lIiwidHlwIjoiSk9TRSIsIm1mMmMtc2VjIjoicHVibGljIiwibWYyYy10bXNwIjoiMTU1MzYwMzgyODcyNCIsIm1mMmMtc2VuZGVyIjoiMGY4NDhkOGZiNzhjYmU1NjE1NTA3ZWY1YTE5OGY2NjBhYzg5YTNhZTAzYjk1ZTc5ZDRlYmZiMzQ2NmMyMGQ1NGU5YTVkOWI5YzQxZjg4Yzc4MmQxZjY3YjMyMjMxZDMxYjRmYWRhOGQyZjlkZDMxYTRkODg0NjgxYjc4NGVjNWEiLCJtZjJjLWFjbGliIjoiMS4wIn0
.
V2VsbCwgYXMgb2YgdGhpcyBtb21lbnQsIHRoZXkncmUgb24gRE9VQkxFIFNFQ1JFVCBQUk9CQVRJT04hICtEbyB5b3UgcmVhbGx5IGNhcmU_ICBJIGRvbid0LiBUaGUgc2lsdmVyIGZveCBxdWlja2x5IGp1bXB0ZWQgb3ZlciB0aGUgZmVuY2UuICBBcyBsb25nIGFzIGl0IGRvZXNuJ3QgZWF0IG15IGNoaWNrZW5zLi4uLkl0IGNhbiBnbyBhcyBpdCBwbGVhc2VzLiA6RA
.
Unsigned JWS with empty signature part

Example

Protected Message (signed JWS)

BASE64URL(UTF8(JWS Protected Header)) || ‘.’ ||
BASE64URL(JWS Payload) || ‘.’ ||
BASE64URL(JWS Signature)

Structure

eyJhbGciOiJSUzI1NiIsImtpZCI6IjFjNTE2NDY2LTRiZjAtNDRiNC04NDcwLWYzOGYxZjNkZmM3ZiIsInR5cCI6IkpPU0UiLCJtZjJjLXNlYyI6InByb3RlY3RlZCIsIm1mMmMtdG1zcCI6IjE1NTM2MDQyOTUyNDQiLCJtZjJjLXNlbmRlciI6IjBmODQ4ZDhmYjc4Y2JlNTYxNTUwN2VmNWExOThmNjYwYWM4OWEzYWUwM2I5NWU3OWQ0ZWJmYjM0NjZjMjBkNTRlOWE1ZDliOWM0MWY4OGM3ODJkMWY2N2IzMjIzMWQzMWI0ZmFkYThkMmY5ZGQzMWE0ZDg4NDY4MWI3ODRlYzVhIiwibWYyYy1hY2xpYiI6IjEuMCJ9
.
V2VsbCwgYXMgb2YgdGhpcyBtb21lbnQsIHRoZXkncmUgb24gRE9VQkxFIFNFQ1JFVCBQUk9CQVRJT04hICtEbyB5b3UgcmVhbGx5IGNhcmU_ICBJIGRvbid0LiBUaGUgc2lsdmVyIGZveCBxdWlja2x5IGp1bXB0ZWQgb3ZlciB0aGUgZmVuY2UuICBBcyBsb25nIGFzIGl0IGRvZXNuJ3QgZWF0IG15IGNoaWNrZW5zLi4uLkl0IGNhbiBnbyBhcyBpdCBwbGVhc2VzLiA6RA
.
dMg8-IZuk3Fo5R0nC1cOznWdRnapL63opAT-DKmJKixE_Yi_c0aA9JnggZ2o2g5O1sFO1-SlhOsbDD1oMaf2xuthSSm0ASmORxtNkFerGxCzdpbVWQejZnfJtO-vSYsXpoBtMSZlwN0pnUdJ3yJMa7xpGGS8Y4N15OLn-oz2Gs5ZBr0VqkR_Fg0LnrOcK1EY4nEkJtYNEzwpeJx2DsMC3h_KsQqxzFZHu0es3rFsr-e9nMAuN65sNKaTsLN_I_-CbPIB2Q3JYda0cT1QpkhLeW9MCC_E_aJ-vrNRcLk2iO035lNcITFLOkWTng2fNbdKVlYMjPUD8ikr12No7KCmhA.

Example

Example Protected Headers (with compression element set and additional metadata set by AC Lib) :

{"zip":"DEF","alg":"none","typ":"JOSE","mf2c-sec":"public","mf2c-tmsp":"1553603681877","mf2c-sender":"0f848d8fb78cbe5615507ef5a198f660ac89a3ae03b95e79d4ebfb3466c20d54e9a5d9b9c41f88c782d1f67b32231d31b4fada8d2f9dd31a4d884681b784ec5a","mf2c-aclib":"1.0"}

Token specifications:

Single recipient JWE is formatted as a 5-part JWE Compact Serialization (RFC7516) String

Multi-recipient JWE are formatted as a general Json serialization (RFC7516) String

JWE (Compact)

BASE64URL(UTF8(JWE Protected Header)) || ‘.’ ||
BASE64URL(JWE Encrypted key) || ‘.’ ||
BASE64URL(JWE Initialization Vector) || ‘.’ ||
BASE64URL(JWE Ciphertext) || ‘.’ ||
BASE64URL(JWE Authentication tag)

Structure

eyJhbGciOiJSU0ExXzUiLCJlbmMiOiJBMTI4Q0JDLUhTMjU2Iiwia2lkIjoiNGYwNTBlNTQtZmI3Zi00MDNjLTgwNzItNmQzMWEzN2ZhMWFiIiwidHlwIjoiSk9TRSIsIm1mMmMtc2VjIjoicHJpdmF0ZSIsIm1mMmMtdG1zcCI6IjE1NTM2MTEzOTI2MzQiLCJtZjJjLXNlbmRlciI6IjBmODQ4ZDhmYjc4Y2JlNTYxNTUwN2VmNWExOThmNjYwYWM4OWEzYWUwM2I5NWU3OWQ0ZWJmYjM0NjZjMjBkNTRlOWE1ZDliOWM0MWY4OGM3ODJkMWY2N2IzMjIzMWQzMWI0ZmFkYThkMmY5ZGQzMWE0ZDg4NDY4MWI3ODRlYzVhIiwibWYyYy1hY2xpYiI6IjEuMCJ9
.
SNRdIlgBgYmm90uRLf-vxIyJGfGfhAtgIsCJS4oQfAZAEom56pO3rPQWg1zFpAbwSZluq5G4Fqyr-POUKnC1o24aRYAgRPD1xxqtHiOaxkQz-HDI9MuFfBXo74LCstYt2hC1CWRO-rloZR7ionpcQFDgDSwsaUHNl9Lzro28uhqoHPACfyHXRc84lpEZMRPLiVfchA4jJfzsNKrLzx3VLzQZ4GduPJIbrfs9a1eclYp8NLBZ2q06Bd1tYRLydoEz1nzFXxxf8ukvXR8TE2bO1-5c7LCvNARO9u3yKUsykuGccxRQ0FS9YDHdh-ATSDWEB1GhhZYWzQV-rNBtRmaj1A
.
zC93BaSlY1kp02TY2yTg4g
.
cOydbXGdiJDff2t5vldDxhfiXE7qrP3KlW3OonwIhL0GtZr1_E-c3hot7mfxhsCdZxYpkGqUFUR8fTxsVXSkrPYC3Fatl3W1ll85mjt58nYmX4cwX0e5Hj7oK_svruGiE8-wUBXka89bh9Y4-0LWTQMq00R6lhi6qDiEkY1f8GfhLehlUa8aZnvfwDLaJGA0Y8YbkiyNt1saXNhU4VqgkNlTsKtDUBVIxsf_WprBWYN7IODsJBOYuudqw-t8EgHG7nrmUxmNk8zOJuWT2C05J9pLV_ATAqmRpkDeD2V69cM
.
FILX3vcNLaZSKclXApCt7w

Example

Json JWE (General)

{"protected":"eyJtZjJjLWFjbGliIjoiMS4wIiwiYWxnIjoiUlNBMV81IiwiZW5jIjoiQTEyOENCQy1IUzI1NiIsInR5cCI6Impzb24iLCJtZjJjLXNlYyI6InByaXZhdGUiLCJtZjJjLXRtc3AiOiIxNTUzNjA5NjYxMTM2IiwibWYyYy1zZW5kZXIiOiIwZjg0OGQ4ZmI3OGNiZTU2MTU1MDdlZjVhMTk4ZjY2MGFjODlhM2FlMDNiOTVlNzlkNGViZmIzNDY2YzIwZDU0ZTlhNWQ5YjljNDFmODhjNzgyZDFmNjdiMzIyMzFkMzFiNGZhZGE4ZDJmOWRkMzFhNGQ4ODQ2ODFiNzg0ZWM1YSJ9“,
"recipients":[{"header":{"alg":"RSA1_5","kid":"2b82208a-d3ba-49c9-b46a-d99dc0ebb121"},"encrypted_key":"CDARNQMIC0Oq0EdP2BsBQTyjNS5ZtFSVnor-Q6EejWy0Ax6cgaF4cEiQrTa3DJx3nzbkW5GCcAXVRMOBKfianjdrcJ2zMlj-0hREPeOZRtePsdBafGQtcc4ZlfG7O_u8LaFzM78lZMP6eGeDbwRdb-G4fiXKc7oGHYL6ibAnyemHWBipQEf44CwoDWV582gMrGgJTvhzzPt6hJ7gKywjMcFPnhZYmNjLKubZN8kKvG9kqX4f6O2zUL3ijQr5jbbNR0vjam6cGFHI_tSbA0eCVMWOPBS6j6HgPfFej-c0fK0igCvuw8pNnjeOgRrh-aodry-yMMDEkC23p1555eoEoQ"},{"header":{"alg":"RSA1_5","kid":"4cacae66-5357-4d81-b108-de0e17aa7a72"},"encrypted_key":"RH2VyFsQtP2A2LxqB7kLAH4NMfqYOxibXE5maBWUXoRE08nBAjLKsZCH56WnTOg9JY8tkve5w2A1DhQmMQIcSxJCAau2P_O2Hcg8wLXqvb1MuiV5oQVQq5h3yBoMwp5SlvO2xFMi__63XK4BuiftbCpUy7OdujutRnUlO7h7PyeXauDNQ-JAe-R9FMd1E2P31m-qXStdhk2Lm2fGaQ5PLYTzzxbV_m2NblBQ2OHbaufQJ81-Zh1KEHCO7TJTHHHgBcWq_n6kId6Ux11al6oXmwwcXnGc8RsRcF8RlXNRdvU5hdW7-GvlgnTzo4z61ejd_WhU0yLgKFWk7ANiEuoGfg"},{"header":{"alg":"RSA1_5","kid":"885626b6-0bdf-4931-957c-7392abfa4a9f"},"encrypted_key":"UdUmedv1L63tUMfsh-BVVU6n3vYt4AxkPBHAzx5Tal_kDUDHkwJuab053igy7BlVn7qjcwFLcXbVrZGo4u1UdGIxYqY0DieiXLlxWgC9T9Xx6nSsYaj2KsvQIg7IFAenXU23FSQPg_9u8cJLGE6wGHAWLOdWp5awT3djYtU7GAHlpH-vIpNjTKxKiReIi2kbgyWqvh4PGtvnG8Oe5jwXikHR35Q0J8LunszE2nn1iRs_LenzlX4sBDpCDRqWIQuGa3tv7v1JkDM5z3WJ64j3c-XB0gal6qOkoLwqoKmK611TF0tHiuurydBpNudkKbQAfGGk8Y3hBqYb8kpFY0tz8A"}],
"iv":"v2cIQg64zYDdupk8WNpVrQ",
"ciphertext":"FqMiaDCwW0YDS5-y_7RFxVjJLlfJQwwNZ5Etqj1km5p1aE53SbgATw_nhvK2iggKpWLKhNF5QeJMyNbvtpWkEP94BU0lzF1glhrr5TydBHr41_JLbY5QQ8BuvGWGhf2-9gKfWDyCMjTIuTYqIR1WvKGMKO-QnhZ8RKd2yQTgOVNrSgjkvwiHMC9LspkHVuwnmVB--88Rz1LIM_PFnrnRnhvQF92eXxbOAWVtMrK9aQUFY3MarILMW8RgEHWGns9Oig5p7cP2Yg1wAiBavg3jklMEVsvKaN3Dq-ewqYx3cV8",
"tag":"pcytqfh3UcwLSOQZjco79Q"}

Example

26March2019 – AC lib

Token specifications:

An Agent's Identity token is structured as a signed JWS with the payload containing the JWT claims

List of claims (in Payload)

CODE	DESCRIPTION
iss	Issuer ID. This would be the requesting Agent's deviceID. In mF2C, the issuer is the same as the presenter.
aud	Audience (intended recipient/s). This is set to the target agent's device ID.
exp	Expiration time in NumericDate format, set to 10 mins
nbf	Validity start time, set to 2 mins.
iat	Issued at time.
jti	Unique ID of the JWT.

JWT Structure = signed JWS

BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature) Structure

eyJ0eXAiOiJKV1QiLCJraWQiOiJrZXkxIiwiYWxnIjoiUlMyNTYifQ
.
eyJpc3MiOiIwZjg0OGQ4ZmI3OGNiZTU2MTU1MDdlZjVhMTk4ZjY2MGFjODlhM2FlMDNiOTVlNzlkNGViZmIzNDY2YzIwZDU0ZTlhNWQ5YjljNDFmODhjNzgyZDFmNjdiMzIyMzFkMzFiNGZhZGE4ZDJmOWRkMzFhNGQ4ODQ2ODFiNzg0ZWM1YSIsImF1ZCI6IjBmODQ4ZDhmYjc4Y2JlNTYxNTUwN2VmNWExOThmNjYwYWM4OWEzYWUwM2I5NWU3OWQ0ZWJmYjM0NjZjMjBkNTRlOWE1ZDliOWM0MWY4OGM3ODJkMWY2N2IzMjIzMWQzMWI0ZmFkYThkMmY5ZGQzMWE0ZDg4NDY4MWI3ODRlYzVhIiwiZXhwIjoxNTU4NTIzMjk0LCJqdGkiOiI1ejRtWGFGd1dXR21fYU1la3NpZlZ3IiwiaWF0IjoxNTU4NTIyNjk0LCJuYmYiOjE1NTg1MjI1NzR9
.
J8sP7LZ61MirRWyArdKadZ3ksq0mxphVWtHA7cX59M8uHKIR-k87wwKJNskAeJryJrR9EEJhzrwr-TvtUbb7ltQsF0iRe65kk6gs_iTnOPKAZuJQUOnbho3hhlgwRZhfzX7KgB-iJOFmBOxkeyBtGeanbO0oPnW0A5LXFn1y6PhVwV_UUoeZ-TI8ed8ANyIi8NUh70yZ8Sn7wOLU3CQnRFHfdy2-mfEDLQjmX1xhcRfTQ-I79CLobWa8AIVYFiAjS-085oJRbvlGRiA8rvfce0j1cLmFf5p69tPAtNjlQuDUi7fcunmcrvrLCSzoW7x1HC5EfQxq83Mp7R-LgrFliw

Example

24 May 2019 – AC lib

Supported Algorithms:

RSA Json Web Key (2048 bits) :

RSA keypair associated with the Agent’s X509 certificate

Plaintext payload de-/compression :

RFC 1951 DEFLATE

Protected messages :

Signature* : SHA256 with RSA

Private messages :

Key management : RSA with PKCS1Padding

Payload content encryption : AES CBC mode (EtM~)

Integrity protection^ : HMAC SHA2

* Over Protected Headers and Payload parts

^ Over Protected Headers (as AAD), Ciphertext, IV, Tag

~ Encrypt-then-MAC