OpenC2 Message Signature

The user SHOULD embed a signature field into the end of the payload that carries all the data required to validate authenticity and integrity of the payload. This should be done as a last step before transfer and only for the purposes of transferring the signature along with the payload. Once the payload is received the receiver should strip off the signature field from the payload, validate the signature, validate the content, and then process the contents. The process in which a particular payload will be signed will be determined by the serialization utilized.

In JSON we can accomplish this by utilizing well know RFC8785 JSON Web Signatures (JWS) and RFC7515 JSON Canonicalizing Scheme (JCS). Although RFC7515 supports a variety of configurations, for this example we will use the RSA256 algorithm and assume that the receiver was previously given the public key during configuration / registration. The following is a generic approach, many libraries in multiple programming languages exist that can alter/simplify this process.

**(Examples below are using a preliminary version of the payload construct proposed by Dave Kemp)**

**OpenC2 Signing Operation (JSON)**

1. Generate the OpenC2 JSON object as described in the OpenC2 Language Specification.

{  
    "content\_type": "application/openc2-cmd+json;version=1.0",  
    "request\_id": "95ad511c-3339-4111-9c47-9156c47d37d3",  
    "created": "1595268027000",  
    "from": "Producer1@example.com",  
    "to": ["consumer1@example.com, consumer2@example.com, consumer3@example.com"],  
    "content": {  
        "request": {  
            "action": "deny",  
            "target": {  
                "file": {  
                    "hashes": {  
                        "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"  
                    }  
                }  
            }  
        }  
    }  
}

1. Canonicalize JSON Data using the process described in RFC8785.

{"content":{"request":{"action":"deny","target":{"file":{"hashes":{"sha256":"22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"}}}}},"content\_type":"application/openc2-cmd+json;version=1.0","created":"1595268027000","from":"Producer1@example.com","request\_id":"95ad511c-3339-4111-9c47-9156c47d37d3","to":["consumer1@example.com, consumer2@example.com, consumer3@example.com"]}

1. Create a JWS using the process described in RFC7515.
   1. Develop a protected header for the type of signature that will be used.

{  
    "alg": "RS256",  
    "kid": "Producer1@example.com"  
}

* 1. Base64 encode the protected header.

eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9

* 1. Base64 encode our canonicalize JSON object from step 2 to create the JWS payload.



* 1. Concatenate the JWS protected header and the JWS payload using with a period character to create our signing input.

eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9.

* 1. Utilize the signing input, RSA256 algorithm, and the senders private key to calculate the signature.

YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA

* 1. Normally at this point we would concatenate all 3 with a period character to create our JWS. However, in order to reduce overhead, we will be using detached version of JWS. To do this we replace the JWS payload portion with an empty string.

eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9..YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA

1. Add the detached JWS back into the original OpenC2 JSON object under the property “signature”.

{  
    "content\_type": "application/openc2-cmd+json;version=1.0",  
    "request\_id": "95ad511c-3339-4111-9c47-9156c47d37d3",  
    "created": "1595268027000",  
    "from": "Producer1@example.com",  
    "to": ["consumer1@example.com, consumer2@example.com, consumer3@example.com"],  
    "content": {  
        "request": {  
            "action": "deny",  
            "target": {  
                "file": {  
                    "hashes": {  
                        "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"  
                    }  
                }  
            }  
        }  
    },  
    "signature": "eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9..YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA"  
}

1. Serialize the signed OpenC2 JSON object and send to recipient(s).

**OpenC2 Signing Validation (JSON)**

1. Parse the received OpenC2 JSON object and separate out the signature. This should yield:
   1. Original OpenC2 JSON object.

{  
    "content\_type": "application/openc2-cmd+json;version=1.0",  
    "request\_id": "95ad511c-3339-4111-9c47-9156c47d37d3",  
    "created": "1595268027000",  
    "from": "Producer1@example.com",  
    "to": ["consumer1@example.com, consumer2@example.com, consumer3@example.com"],  
    "content": {  
        "request": {  
            "action": "deny",  
            "target": {  
                "file": {  
                    "hashes": {  
                        "sha256": "22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"  
                    }  
                }  
            }  
        }  
    }  
}

* 1. Original Detached JWS.

eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9..YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA"

1. Canonicalize JSON Data using the process described in RFC8785.

{"content":{"request":{"action":"deny","target":{"file":{"hashes":{"sha256":"22fe72a34f006ea67d26bb7004e2b6941b5c3953d43ae7ec24d41b1a928a6973"}}}}},"content\_type":"application/openc2-cmd+json;version=1.0","created":"1595268027000","from":"Producer1@example.com","request\_id":"95ad511c-3339-4111-9c47-9156c47d37d3","to":["consumer1@example.com, consumer2@example.com, consumer3@example.com"]}

1. Create a JWS using the process described in RFC7515.
   1. Base64 encode our canonicalize JSON object from step 2 to create the JWS payload



* 1. Overwrite the detached JWS empty string between the first and second period characters with the JWS payload to create a standard, non-detached, JWS.

"eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9..YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA"

1. Follow the JWS validation process described in RFC7515.
   1. Save the JWS signing Input (which is the initial substring of the JWS up until but not including the second period character)

eyJhbGciOiJSUzI1NiIsImtpZCI6IlByb2R1Y2VyMUBleGFtcGxlLmNvbSJ9.”

* 1. Save the JWS signature (Which is the string following but not including the second period character)

“YK6oogmIYMLlyYzOJ6LZkGtgDEWON1IxtWgoX3MAmFyRdXZWZQhIbhdNoPT1SKvYqk4DTuhw1IKUm2TCEHdCSWewLJNOErfQ6hxB4YoBwsBO7796QxnbkWOVeNvhvaQ-Kmr7WfZA6gKPluncEiRKzb4Hyn5gxOJOEFzWDUW57k8QeL0c7NqQIDEpU6RiP2mwngOVn6z8dZ-nkbdDrdj2Jg-caNF5kE-F9MD39ucLYRzT\_hDB1hukFsHJguKH8NIxfdPizOcjdvaXgxrD0W5DJIfjJRd1oOIZBqUDTOdd2iNT5jK\_CvtdGgcJHMEvvdgU3pmp2D099vyGuLrWySJ9QA"

* 1. Pass the public key, the JWS signature, and the JWS signing input to an RSASSA-PKCS1-v1\_5 signature verifier. Expect a Boolean response.

-----BEGIN PUBLIC KEY-----  
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAhFWEXArvaZEpSP5qNX7x4C4Hl28GJQTN  
vnDwkfqiWs63kXbdyPeS06bz6GnY3tfQ/093nGauWsimqKBmGAGMPtsV83Qxw1OIeO4ujbIIb9pe  
ma0qtVqs0MWlHxklZGFkYfAmbuEUFxYDeLDHe0bkkXbSlB7/t8pCSvc8HLgHjEQjYOlFRwjR0D+u  
Lo+xgsCbpmCtYkB5lcT/zFgpRgY4zJNLSv7GZiz2S4Fc5ArGjd34lL47+L8bozuYjqNOv9sqX0Zg  
ll5XaJ1ndvr7UqZu1xQFgm38reoM3IarBP/SkEFbt/v9iak602VO3k28fQhMaocP7JWR2YLT3kZM  
0+WTFwIDAQAB  
-----END PUBLIC KEY-----