MILE, JSON and JOSE

Jim Schaad
Co-Chair JOSE Working Group

Background

- MILE – Currently uses XML
- JSON – The new hot thing
- JOSE – JSON Signature and Encryption

I have had a couple of talks with Kathleen about the fact that there is some possible desire to use JSON rather than XML
JSON is considered to be simpler than XML, but it comes at the cost of a lack of some tools
Will not be looking at questions like – replacing of XML Schema, XPath – These will need to be looked at as well in the future

JOSE was chartered to create a standard for doing Signature, MAC and encryption operations uses JSON, as well as other things, for the input format. Using JSON, as well as URL safe version as output format.

JSON Background

- OpenID initial design
  - Signature only
  - JSON input
  - ASCII String output
- Added URL Encryption
- Added JSON Signature and Encryption as outputs

The JSON work was originally presented to the IETF by the OpenID group for standardization
Initial design was to take JSON as input, but could take arbitrary input
Produce as output a text string that was suitable to be placed in a URL as part of an HTTP request
Needed to be compact, needed to be a string that contained no unsafe characters from URL perspective
Initial design focused only on signatures, not on encryption.

During the WG process, the ability to do encryption was added.
During the WG process, the ability to represent the output in JSON was added.

Difference in capabilities between the URL safe and the JSON version

1- Multiple signers
2- Unsigned statements
3- Multiple encryption recipients
4- Unauthenticated statements

Canonicalization

- XML Signature Canonicalization
- JOSE – "Canonicalization is evil – avoid at all costs"

- JSON Canonicalization problems
  - Minimal Data typing
  - Floating point numbers

Implementers in this group will be familiar with the XML canonicalization problems. Number of different issues involved:

- Inclusion of original document white space [Need to know before the document is parsed if whitespace is preserved]
- Namespace [namespace prefix normalization]
- Default values [present or not – depends on if one is using a validating parser]
- Not a tree [The thing to be signed may be either a tree, or it could be a set of nodes]

Most people have learned the wrong lesson from the XML Canonicalization debacle.

Real lesson is "Send what you sign"
Don't make both sides attempt to re-create the same input
Has not worked for:

- X.509 Certificates [People mess up DER encoding, original project decomposed certificates in to directory and attempted to rebuild and validate signatures – failures occurred]
- XML – Multiple canonicalization methods exist to deal with different environments – need to know what is going on before one starts – lots of problems with people trying to re-construct from different sections of documents as depends on XPATH as well as canonicalization problems

Canonicalization

- JOSE approach is Base64 everything
- Desirable
  - minimize side-channel
  - Simplify comparing outputs
- Robust in absence is a requirement

JOSE decided that it would never do canonicalization. It will always do a send what was signed. However to make sure there are no problems with things being changed, it will always base64 encode the item to be signed – before signing it.

There are reasons why canonicalization is desirable – one minimizes the amount of side channels as things like count of spaces cannot be used
It makes it simpler in many respects to compare two different implementations as well as doing testing against test cases for generation. If we don't produce the same output from the JSON encoding step then we cannot easily generate the same input to the hashing process

The specification needs to be robust in the presence of either lack of canonicalization or failures.

MILE Signature Usages

- RFC 6545 - RID
  - Encapsulated Signature over entire iodef-rid:RID
  - Detached Signature over iodef:RecordItem
- RFC 5070-bis – IODEF
  - HashClass

Based on the documents, I have found three different locations where either Signatures, or something similar is being used in the MILE documents

A detached signature over one or more iodef:RecordItems is created for new reports of data. These are designed to be long term signatures. They are also designed to be passed on to consumers at a later time, quite potentially when combined with other records.

An encapsulated signature over the RID. This is designed to cover the transport from the reporter/requester to the server (or visa versa) and would be a moderate term signature. It might be stored if the entire RID was stored or it might not. It is designed so that the originator of the RID can be identified and the validity of the RID (not necessarily is voracity) can be checked.

In the BIS document, it appear that signatures might be used over external objects, or it could just be hashes of external objects. The text is not clear. It appears to be hashes because of the class name, however in that case items such as KeyInfo objects would not be needed in the structure as well. In any event these are going to be detached items signed over items that are external to the document.

Encapsulated Signatures

- Encapsulated signatures supported by JOSE
- Signed signature statements supported, but few defined
- Base64 expansion on both signed signature statements and payload

The encapsulated signature model is fully supported by JOSE. Providing support for these types of signatures is not an issue. There are however a few downsides that need to be looked at in the event this would be adopted.

It is not clear what, if any, signature statements are being used by the current standard. It currently just references the best practices document for XML signatures. This document recommends the use of a signing time for long term documents, but does not make any other requirements.

The JOSE standard allows for signed statements to be included as part of the signature processing, however the set of possible statements is currently very minimal.
Inner content type is one of few defined – defined using Media Types (i.e. application/mile+jose)

Message size is and issue because when transported everything that was signed is going to be base64 encoded while it is transmitted.

Embedded Signature Example

{"payload": "eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4M
TkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNv
bS9pc19yb290Ijp0cnVlfQ",
"signatures":[
{"protected":"eyJhbGciOiJFUzI1NiJ9",
"header": {"kid":"e9bc097a-ce51-4036-9562-
d2ade882db0d"},
"signature": "DtEhU3ljbEg8L38VWAfUAqOyK
AM6-Xx-F4GawxaepmXFCgfTjDxw5djxLa8IS
lSApmWQxfKTUJqPP3-Kg6NU1Q“
}] }

Detached Signatures

- No true support in the specification
- Requires some type of solution to the canonicalization problem for JSON and other contents
- Or make things strings that cannot change

There is no native support built into the specification for detached signatures. There is an appendix that provides a hint about how to do it. This amounts to removing the payload element from the structure and replacing it before doing the validation

For JSON that is transported as part of the message, some type of canonicalization is going to be required to make sure that the same content can be reproduced. Alternatively, one can convert the things that are signed into strings and then sign that string. This means one ends up sub trees of the document which is no longer a tree but is replaced by a string. This makes the parsing of a document much more complicated than just keeping things simple.

Same issue exists for other external content Need to deal with things like end of line normalizations for text files and e-mail messages. This will need to be done in any event for the bis document as well. The only difference is that there is currently no real way to state a set of canonicalization steps to be applied to detached payload in the XML case as well.

Encryption

- Embedded encryption is used by MILE
- Supported by JOSE
- X.509 Certificate trust model is common to both
- In both cases Base64 encoded encrypted content
- JOSE requires the use of AEAD algorithm

MILE encryptions an various nodes in the XML tree is expected to occur. Where and how the encryption is to occur is not always clear to outside reader.

The same type of encryptions are supported by JOSE so this is not a problem.

MILE is setup to use the X.509 certificate trust model, and to identify certificates in the standard ways: Shared Key Name, Issuer/Serial number, SKI, Subject Name, embedding the certificate in the message, a URL to a certificate.

Of these, the Issuer/Serial number and subject name are not supported in JOSE.

In both cases, since we are looking at text, the payload is going to be base64 encoded when inserted back into the document. JOSE may also have a set of authenticated values that will be base64 encoded and placed in the document as well.

JOSE requires that all encryption methods be Autenticated Encryption with Associated Data algorithms (i.e. AES-GCM).

Summary

- No JSON schema or canonization and none like to come any time soon
- Embedded JSON signatures work
- Detached JSON signatures are iffy if the content is not strictly controlled
  - Rebuilding to a string as hard or harder the XML
- Encryption of JSON works

I would not recommend it at this time.