REST Request Authentication

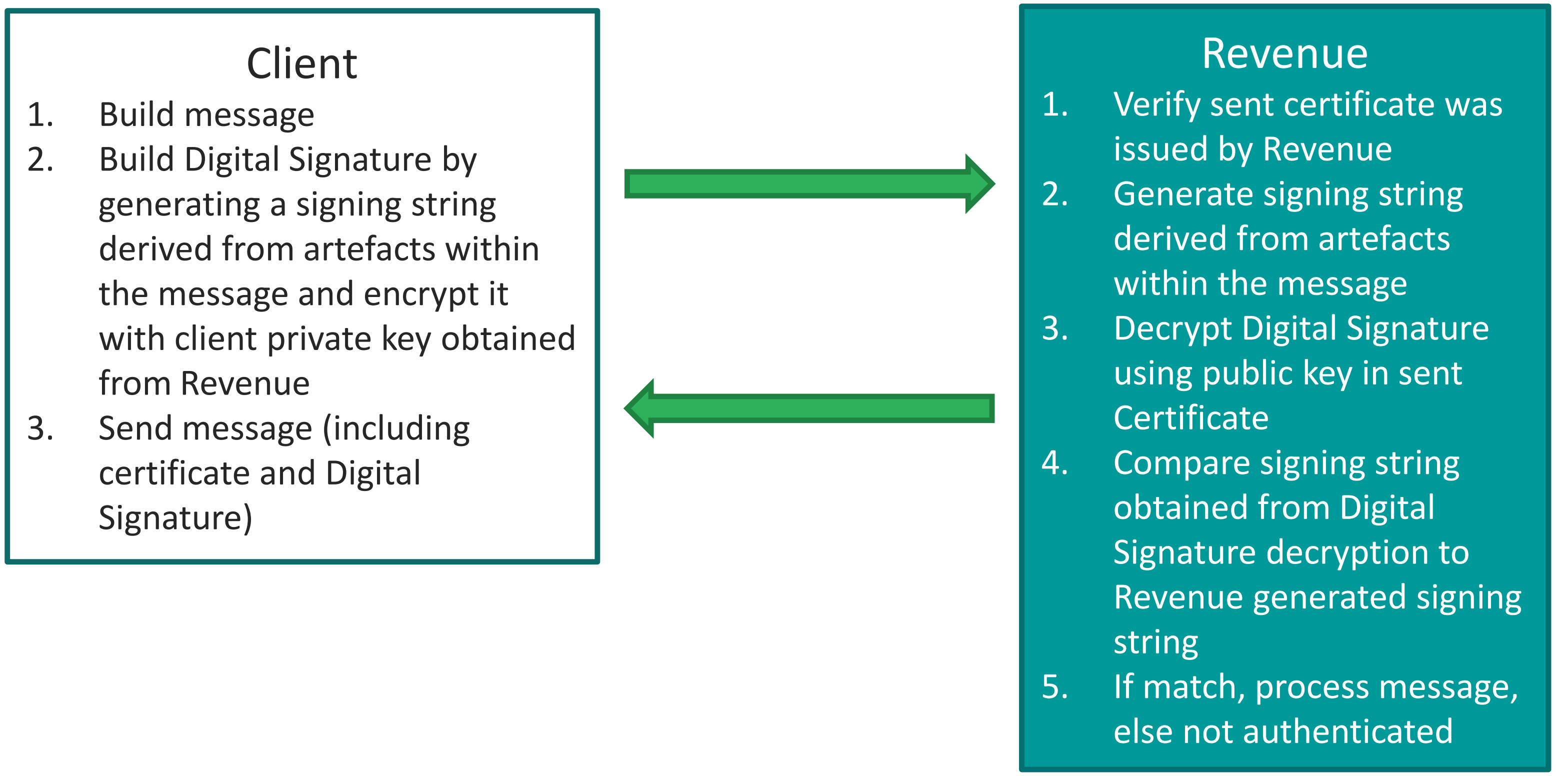
Contents of Technical Workshop

- Use of Digital Signatures
- Implementation of Digital Signature using ‘HTTP Signatures’
- How to build HTTP Signature Header along with sample code snippets
- Questions??

Overview

- Any Revenue web service request that either returns confidential information or accepts submission of information must be digitally signed. This must be done using a digital certificate that has been previously retrieved from Revenue.

- The digital signature ensures the integrity of the document. By signing the document we can ensure that no malicious intruder has altered the document in any way. It is also be used for nonrepudiation purposes.



Client

1. Build message
2. Build Digital Signature by generating a signing string derived from artefacts within the message and encrypt it with client private key obtained from Revenue
3. Send message (including certificate and Digital Signature)

Revenue

1. Verify sent certificate was issued by Revenue
2. Generate signing string derived from artefacts within the message
3. Decrypt Digital Signature using public key in sent Certificate
4. Compare signing string obtained from Digital Signature decryption to Revenue generated signing string
5. If match, process message, else not authenticated

HTTP Signatures

- The HTTP signatures protocol is intended to provide a simple and standard way for clients to sign HTTP requests.
- At a high level, a HTTP Signature is a HTTP header that is added to a HTTP request. It is comprised of a set of components that were used to generate a digital signature and the digital signature itself.
- Below is a sample HTTP Signature Header

```
Signature:  keyId="MIICfzCCAeigAwIBAgIJ... // truncated",
            algorithm="rsa-sha512",
            headers="(request-target) host date digest",
            signature="GdUqDgy94Z8mSYUjr/rL6qrLX/jmudS... // truncated"
```

- keyId – Revenue issued Certificate
- algorithm - Digital signature algorithm to use when generating the signature
- headers – List of headers used when generating the signature
- signature - Digital signature generated from the algorithm and headers field(forms a canonicalized 'String to be signed') above

HTTP Signature header Preparation

- Before we can build the HTTP Signature header, we must add all HTTP headers/ components that will be used to generate the digital signature to the HTTP request. These components will be specified in the 'headers' portion of the HTTP Signature header later.

- Allowable values in the headers field are outlined in the table below

Value	Mandatory
(request-target)	Yes
host	Yes
date	Yes
x-date	Yes, if date header cannot be added.
digest	Yes, if HTTP method is of type POST
content-type	No
content-length	No
x-http-method-override	If HTTP method is of type POST, HTTP header 'X-HTTP-Method-Override' exists and 'Content-Type=application/x-www-form-urlencoded

HTTP Signature header Preparation 2

- It should be noted that the (request-target) Allowable value defined in the table above is built from 2 HTTP headers. It is generated by concatenating the lowercase HTTP method, an ASCII space, and the request path headers. See below for sample

  (request-target): get /v1/rest/rpn/{ employerRegistrationNumber }/{taxYear}

- Unfortunately, the initial implementation of the (request-target) HTTP header differs slightly from that specified in the IETF HTTP Signatures draft as outlined below.

  Example of expected (request-target) as currently implemented by revenue

  (request-target): get /v1/rest/payroll/1234567CH/2019/1/1

  Example of expected (request-target) as outlined by IETF HTTP Signatures draft

  (request-target): get /paye-employers/v1/rest/payroll/1234567CH/2019/1/1?softwareused=xyz&softwareVersion=1.0

- We are working towards aligning our implementation with that specified in the IETF HTTP Signatures draft and hope to release the change to PIT as soon as possible.

Building a HTTP Signature header

- Once the required HTTP headers have been added to the HTTP request, we can begin to build the HTTP Signature header which is simply a concatenation of the following pieces of information
- **keyId** - Get X509 certificate that accompanies the private key as a byte array and Base64 encode. This field is required.
  - Encode the Password used to open the keystore
  - Open the keystore
  - Get the Certificate from the keystore
  - Get Base64 Encoded Certificate As String
- **algorithm**: The ‘algorithm’ parameter is used to specify the digital signature algorithm to use when generating the signature. Revenue expects this to be ‘rsa-sha512’. This field is required.
- **headers**: The 'headers' parameter specifies the list of headers used when generating the signature for the message. The parameter must be a lowercased, quoted list of HTTP header fields, separated by a single space character. The list order is important, and MUST be specified in the order the HTTP header field-value pairs are concatenated together during signing.

Building a HTTP Signature header 2

- **signature**: The signature component is a base 64 encoded digital signature string. The implementer uses the 'algorithm' and 'headers' field to form a canonicalized 'String to be signed'.
  - The 'String to be signed' is signed with the private key that accompanies the X509 certificate with the 'keyId' field and the algorithm corresponding to the 'algorithm' field. The 'signature' field is then base 64 encoded, converted to a String and concatenated with the rest of the fields. The following outlines the steps to be taken to generate the string to be signed

    1. Generate the String to be signed – In order to generate the string to be signed, the implementer MUST use the values of each HTTP header defined in the 'headers' signature field, to build the signature string. Values must be in the order they appear in the 'headers' signature field. If the associated HTTP header does not exist, it should be added to the HTTP request BEFORE attempting to construct this string.
       - The (request-target) header is built from 2 HTTP headers. It is generated by concatenating the lowercase HTTP method, an ASCII space, and the request path headers.
       - All other header field values are created by concatenating the lowercase header field name followed by an ASCII colon ':', an ASCII space ' ', and the header field value. Leading and trailing whitespace in the header field value MUST be omitted. If the header field is not the last value defined in the 'headers' signature field, then append an ASCII newline '\n'. See example below

```
(request-target): get /v1/rest/rpn/3390938BH/2018\n
host: www.ros.ie\n
date: Wed Jun 13 2018 11:37:48 GMT+0100 (GMT Daylight Time
```

Building a HTTP Signature header 3

2. Encode the Password used to open the keystore
3. Open the keystore
4. Get private key From KeyStore
5. Sign string

- Once all 4 parts have been created and concatenated, we will end up with a string similar to below

```
Signature: keyId="MIICfzCCAeigAwIBAgIJ... // truncated",
algorithm="rsa-sha512",
headers="(request-target) host date digest",
signature="GdUqDgy94Z8mSYUjr/rL6qrLX/jmudS... // truncated“
```

Questions

- ???