This invention is herby placed in the *Public Domain*

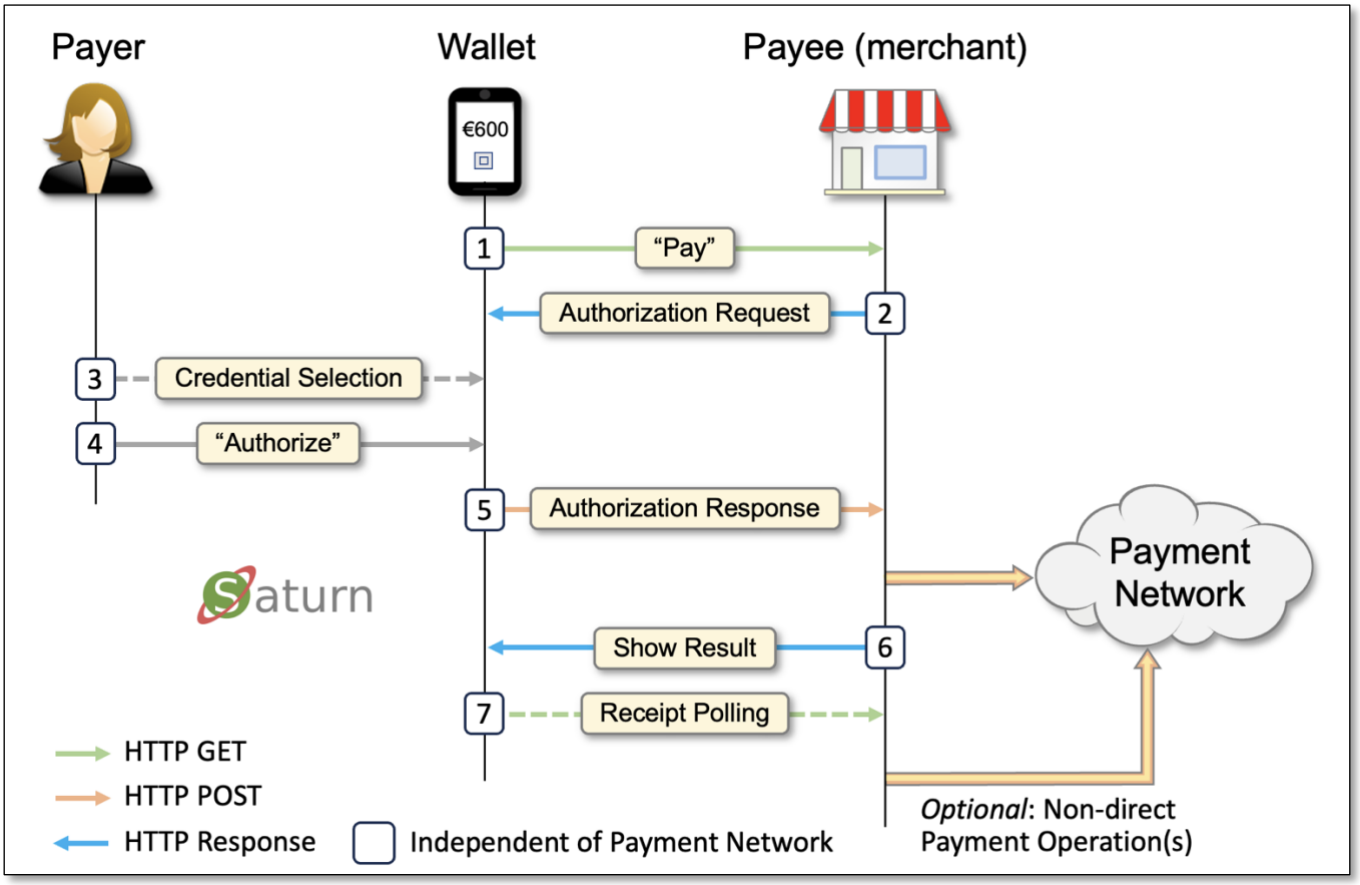
Partial Encryption, Full Signature – Invention Disclosure

ABSTRACT

This document outlines a scheme where a Payee-originated `Authorization Request` is triggering a corresponding `Authorization Response` object, created by a Payment Wallet under the Payer's supervision.
The `Authorization Response` object contains both the core of the original `Authorization Request` object as well as data associated with a by the Payer selected Payment Credential. All data returned is signed by a *single signature* associated with the selected Payment Credential. For *privacy reasons* (with respect the Payee), Personally Identifiable Information (PII) is *encrypted*, while the `Authorization Request` as well as non-personal data related to the Payment Network associated with the selected Payment Credential, is provided in clear to facilitate backend processing. The described scheme builds on CBOR.

1. Sequence Diagram

A Merchant is an entity requesting a payment from a Payer. Merchant entities include on-line (Web) shops, local shops, and automated facilities like gas stations. Payers are equipped with mobile devices hosting digital Wallets containing Payment Credentials issued by Payer Banks. The following sequence diagram was taken from Wallet Core [CORE] but the specification herein is only targeting the `Authorization Request` and `Authorization Response` pair.



Throughout this specification messages are expressed in the human-readable format of CBOR [RFC8949] known as "Diagnostic Notation". In an actual implementation, the normal (binary) format would always be used.

! Note that the described concept depends on that the CBOR serializer and parser supports Deterministic Encoding [DE-CBOR], as well as maintaining proper CBOR representation when adding and removing CBOR map keys.

2. Authorization Request

The following CBOR message represents a possible `Authorization Request` object to be sent to the wallet:

```
{
  "paymentRequest": {
    "id": "20241210.00079",
    "amount": "600.00",
    "currency": "EUR",
    "commonName": "Space Shop"
  },
  "supportedNetworks": ["https://banknet2.org", "https://visa.com"]
}
```

Core request

Note: the `supportedNetworks` attribute is not core to this specification but adds the possibility supporting multiple Payment Networks through a single wallet and authorization concept.

The `PaymentRequest` part of the `Authorization Request` object would then typically be rendered in the wallet like:

After the Payer have authorized the `PaymentRequest` using a PIN or biometric operation, a corresponding `Authorization Response` object is created and sent back to the Payee.

In the sample UI, Payment Credentials (aka virtual payment cards), are selected by swiping.

3. Authorization Response

The `Authorization Response` object is created through a number of discrete steps which constitute the core of this invention disclosure.

3.1 Unsigned Authorization Response

The following CBOR code shows a possible unsigned `Authorization Response` object:

```
{
  "accountId": "FR7630002111110020050014382",  # PII (user account data)
  "unencryptedData": {
    "timeStamp": "2024-12-10T13:28:02-01:00",
    "providerInfo": {
      "networkId": "https://banknet2.org",
      "serviceLocator": "mybank.com"
    },
    "paymentRequest": {
      "amount": "600.00",
      "currency": "EUR",
      "commonName": "Space Shop",
      "referenceId": "20241210.00079"
    }
  }
}
```

Core request

The attributes `networkId` and `serviceLocator` represent payment network/method respectively an indicator used for accessing the Issuer. In the sample, `serviceLocator` is expressed as a host-name meaning that the actual service end-point is to be retrieved through a `/.well-known/` URL [RFC8615].

The `networkId`, `serviceLocator`, and the `accountId` attributes are derived from the selected Payment Credential.

Although "`mybank.com`" provides information about the bank of the Payee, this design considers such information as non-personal.

3.2 Signed Authorization Response

Adding an embedded CBOR signature [CSF], to the unsigned `Authorization Response` object will result in the following CBOR code:

```
{
  "accountId": "FR7630002111110020050014382",  # PII (user account data)
  "unencryptedData": {
    "timeStamp": "2024-12-10T13:28:02-01:00",
    "providerInfo": {
      "networkId": "https://banknet2.org",
      "serviceLocator": "mybank.com"
    },
    "paymentRequest": {
      "amount": "600.00",
      "currency": "EUR",
      "commonName": "Space Shop",
      "referenceId": "20241210.00079"
    }
  },
  "authorizationSignature": {
    1: -50,
    4: {   # PII (static user-specfic key)
      1: 1,
      -1: 6,
      -2: h'fe49acf5b92b6e923594f2e83368f680ac924be93cf533aecaf802e37757f8c9'
    },
    6: h'a4ff37cc4b1db1a760ea930f274c59d71a7e7edef5b2e4a07eff764bb01590a7...'
  }
}
```

Core request

Notes:

The embedded signature encompasses the entire CBOR object except for signature value itself (label 6).

The signature value was truncated for readability reasons.

The sample uses an Ed25519 signature algorithm (1: -50).

The signature key is derived from the selected Payment Credential.

The public part of the signature key is specified by label 4.

Algorithms and key containers are compatible with COSE [RFC9052].

3.3 Encrypted Authorization Response

To build the final `Authorization Response` object, the signed authorization object is first split into two:

- One part holding data that must remain in clear in order to facilitate further processing.
- Another part containing privacy-sensitive data like account numbers and associated public keys. This part must be *encrypted*.

Data to be supplied in clear:

```
{
  "timeStamp": "2024-12-10T13:28:02-01:00",
  "providerInfo": {
    "networkId": "https://banknet2.org",
    "serviceLocator": "mybank.com"
  },
  "paymentRequest": {
    "amount": "600.00",
    "currency": "EUR",
    "commonName": "Space Shop",
    "referenceId": "20241210.00079"
  }
}
```

Core request

By keeping the `paymentRequest` in clear, this information eliminates duplication in other message layers.

Data to be encrypted:

```
{
  "accountId": "FR7630002111110020050014382",  # PII (user account data)
  "authorizationSignature": {
    1: -50,
    4: {   # PII (static user-specific key)
      1: 1,
      -1: 6,
      -2: h'fe49acf5b92b6e923594f2e83368f680ac924be93cf533aecaf802e37757f8c9'
    },
    6: h'a4ff37cc4b1db1a760ea930f274c59d71a7e7edef5b2e4a07eff764bb01590a7...'
  }
}
```

In this specification the `accountId` and the associated credential public key (label 4) define Personally Identifiable Information (PII). Merchants should not have such information; therefore, this part of the `Authorization Response` object is *encrypted*.

Continued…

The encryption uses a CBOR container object called CEF [CEF]. The resulting object represents the completed (final) `Authorization Response` object:

```
{
  0: {
    "timeStamp": "2024-12-10T13:28:02-01:00",
    "providerInfo": {
      "networkId": "https://banknet2.org",
      "serviceLocator": "mybank.com"
    },
    "paymentRequest": {
      "amount": "600.00",
      "currency": "EUR",
      "commonName": "Space Shop",
      "referenceId": "20241210.00079"
    }
  },
  1: 3,
  2: {
    1: -29,
    4: {   # Shared static DH key – Not PII
      1: 2,
      -1: 1,
      -2: h'e812b1a6dcbc708f9ec43cc2921fa0a14e9d5eadcc6dc63471dd4b680c6236b5',
      -3: h'9826dcbd4ce6e388f72edd9be413f2425a10f75b5fd83d95fa0cde53159a51d8'
    },
    7: {   # Ephemeral DH key – Not PII
      1: 2,
      -1: 1,
      -2: h'ce264b040ad50970935438f7588202b633e99587ebf43b8b6f6433cddfd0af79',
      -3: h'8a156afabaccb3143525f83511ca108601c5e12c21d0cec5737e900a7b56f7c1'
    },
    10: h'c25237ed702f9479f247ee95b5cb8465faaa6c096ccb026fb0068d87597661....'
  },
  8: h'33b239edee76332761d9cd041ce82d6f',
  9: h'ab1d2e84b6f48f03cba0ba67',
  10: h'f48bb11ca94bc94e47bfc0d0f31a36de2a1ef2162c3765c576a4d7b6a042d7dc....'
}
```

Core request

Notes:

In a CEF object, label 0 holds data that should pass *unencrypted*.

The encryption uses a key usually hold by the Credential Issuer. To not transcend the encryption public key (label 4) into PII, encryption keys must be *shared* among multiple clients.

The key encryption algorithm (1: -29) is ECDH-ES+A128KW using a P256 key (-1: 1).

The content encryption algorithm (1: 3) is A256GCM.

Additional Authentication Data (AAD) encompasses the entire CBOR object except for the tag (label 8), iv (label 9), and resulting ciphertext (final label 10).

Algorithms and key containers are compatible with COSE [RFC9052].

4. Payment Credentials

Payment credentials are stored in a wallet database. Each entry contains as a minimum the following elements:

- Account identifier. Corresponds to `accountId` in `Authorization Response` objects.
- Payment network Identifier. Corresponds to `networkId` in `Authorization Response` objects.
- Service endpoint identifier. Corresponds to `serviceLocatore` in `Authorization Response` objects.
- Credential private key. Key used for signing `Authorization Response` objects. It is supposed to be unique for each Payer and `accountId`.
- Credential public key. Key matching the credential private key. This key is featured in `Authorization Response` signatures to facilitate signature validation including the association with the `accountId`.
- Encryption public key. Key used for encrypting `Authorization Response` objects.
- Credential card image. Image used in wallet UIs to aid user credential selection.

Payment credentials are created through a (not shown) registration process performed by Issuers and their customers (Payers).

5. Verification and Validation

Verification of `Authorization Response` objects is performed by running the steps described in section 3, but in a reversed mode. That is:

- Decrypting the `Authorization Response` object using a private key associated with the public key provided in the encryption object.
- Assembling the signed `Authorization Response` object using the result of the decryption process.
- Validate the `Authorization Response` signature using the public key provided in the signature object.

Note: full validation also requires credential database lookups as well as validation of input data.

Invention Summary

This invention disclosure describes the following sub inventions:

1. The described scheme combines the security and privacy needed for retail payments.
2. Embedded signatures and deterministic encoding enable signed data to be provided in its original CBOR form.
3. Using a CBOR serializer and parser that permits secure removal and insertion of map elements, signed data can be sliced and later be restored without breaking signatures. This in turn enables selective encryption which is a core part of this invention disclosure.
4. Encryption objects may also include data in clear, making encrypted response objects more versatile. In addition, included clear text data contribute to security and integrity by also becoming a part of Additional Authentication Data (AAD).

References

Name	Description	URL
CORE	Wallet Core	https://github.com/cyberphone/wallet-core
RFC8949	CBOR	https://www.rfc-editor.org/rfc/rfc8949
DE-CBOR	Deterministically Encoded CBOR	https://datatracker.ietf.org/doc/draft-rundgren-cbor-core/
RFC8615	`/.well-known/` URL	https://www.rfc-editor.org/rfc/rfc8615.html
CSF	CBOR Signature Format	https://cyberphone.github.io/javaapi/org/webpki/cbor/doc-files/signatures.html
CEF	CBOR Encryption Format	https://cyberphone.github.io/javaapi/org/webpki/cbor/doc-files/encryption.html
RFC9052	COSE	https://www.rfc-editor.org/rfc/rfc9052.html

Permanent document URL: https://cyberphone.github.io/doc/defensive-publications/partial-encryption-full-signature.pdf

Author: Anders Rundgren (`anders.rundgren.net@gmail.com`)