WebPKI
• O R G

Saturn™

End-to-End Secured Payment Authorization System

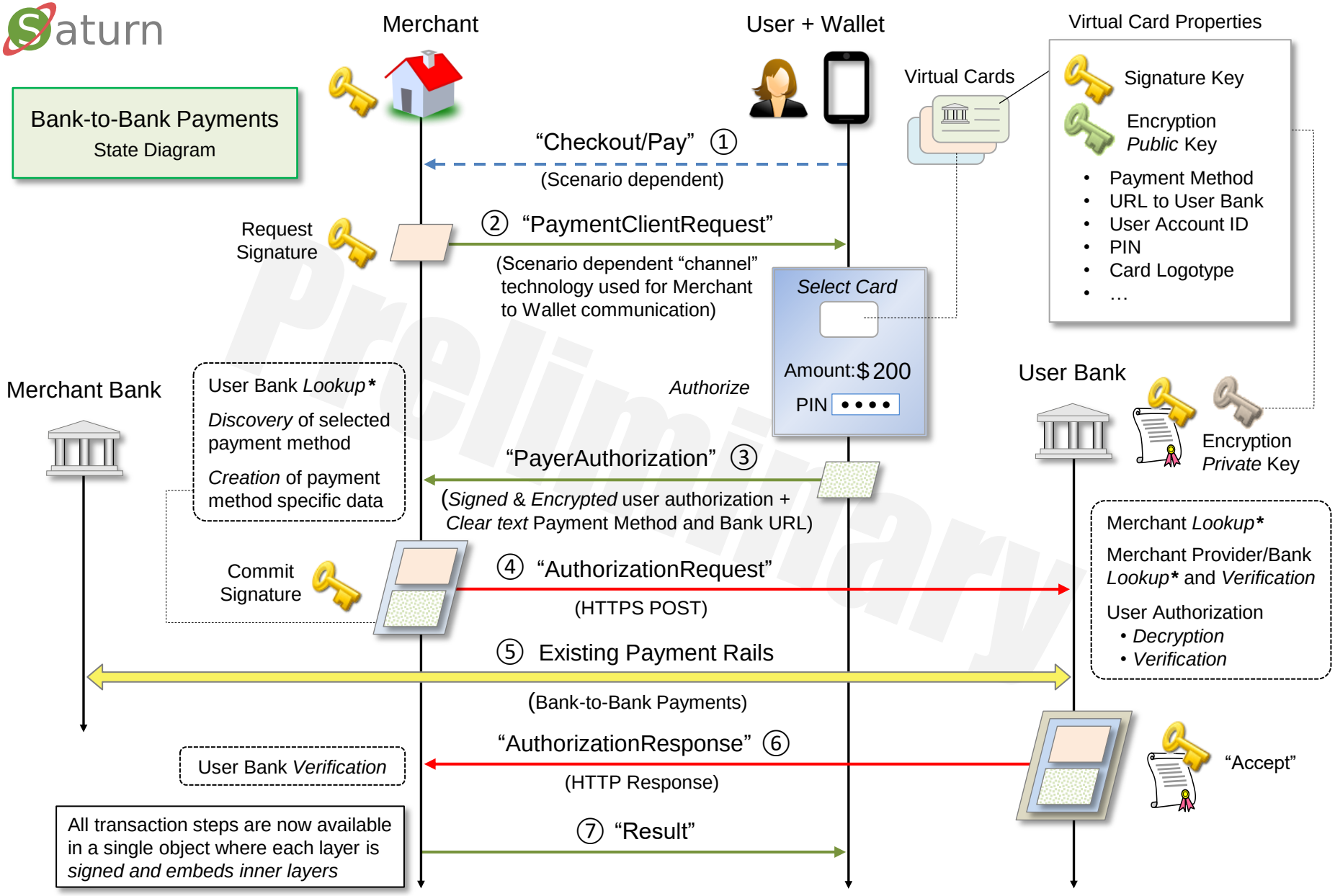
- *Decentralized operation* accomplishes similar goals as 3D Secure and “Tokenization” but *without registries or additional services*
- Facilitates the design of brand/bank independent, “rich UI” wallets, supporting both card- and bank-to-bank payments
- Equally applicable on the mobile Web, locally in a shop, at an automated gas station, or as a “PC companion” on the Web
- *Eliminates* the traditional payment terminal and reduces merchant PCI requirements to a minimum
- Requires a *single* “active” method on the issuer side to function*

** Reservations and reoccurring payments will in non-card-based scenarios need a second method as well*

Disclaimer: This is a system in development and specifications are subject to change without notice

Saturn

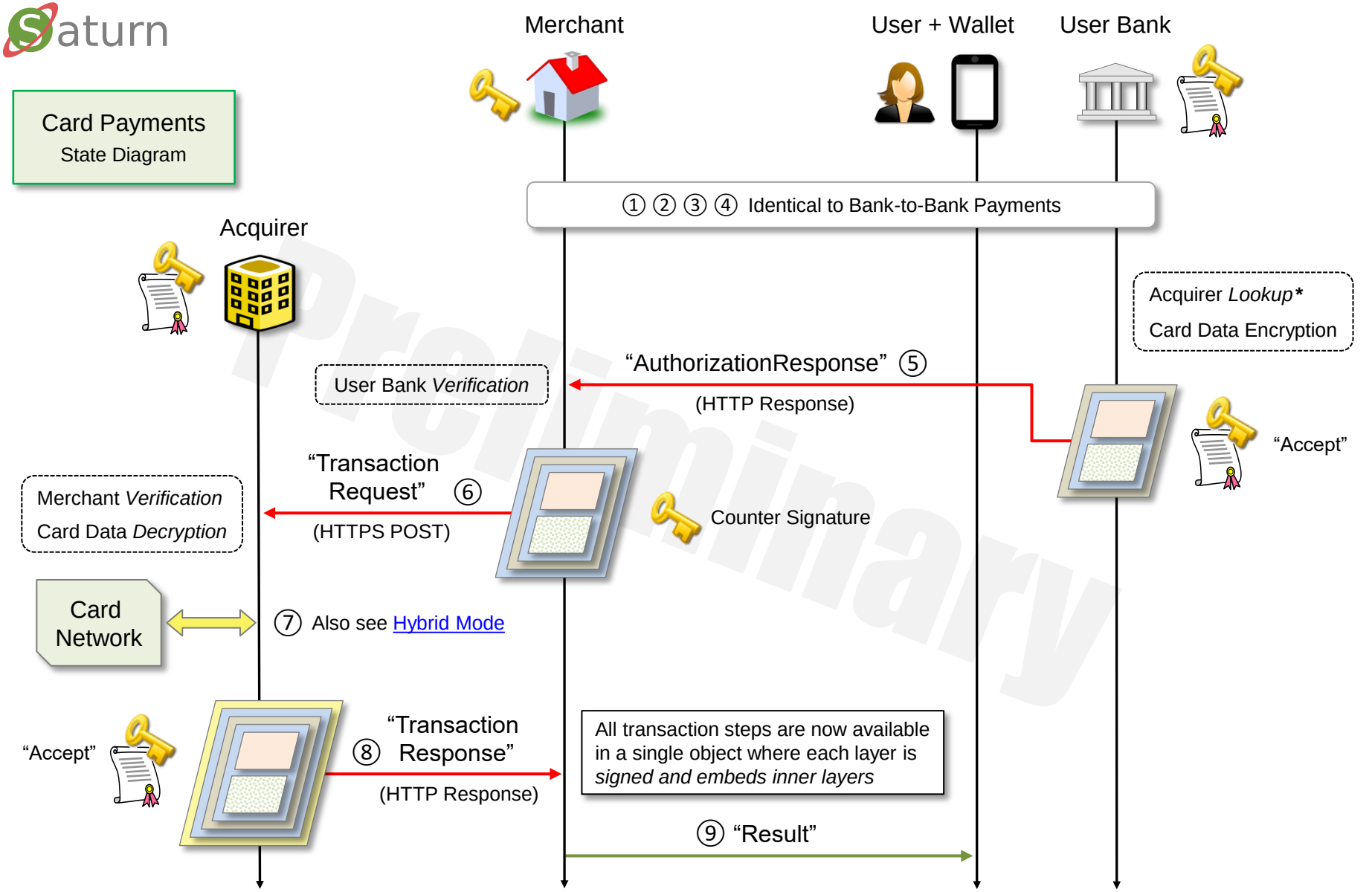
Bank-to-Bank Payments
State Diagram



* See Authority Objects. The rationale for encrypting user authorizations is for enabling such data to pass through Merchants which simplifies the Wallet as described in the Saturn FAQ. *Step #5 does not apply when running under the conditions outlined in* Hybrid Mode.

Saturn

Card Payments
State Diagram



***** See Authority Objects. The flow may stop after step #5 resulting in a *Secure Authorization Object* which *only* can be activated by a *Counter Signature*. This scheme supports hotel bookings, upfront reservations for automated gas stations, as well as reoccurring payments. The card data *Encryption* and *Decryption* processes enable standard card data to securely pass through Merchants from Issuers to Acquirers.

② Merchant Sends – “PaymentClientRequest” JSON Message

Payment method URIs to be matched against those in the virtual cards

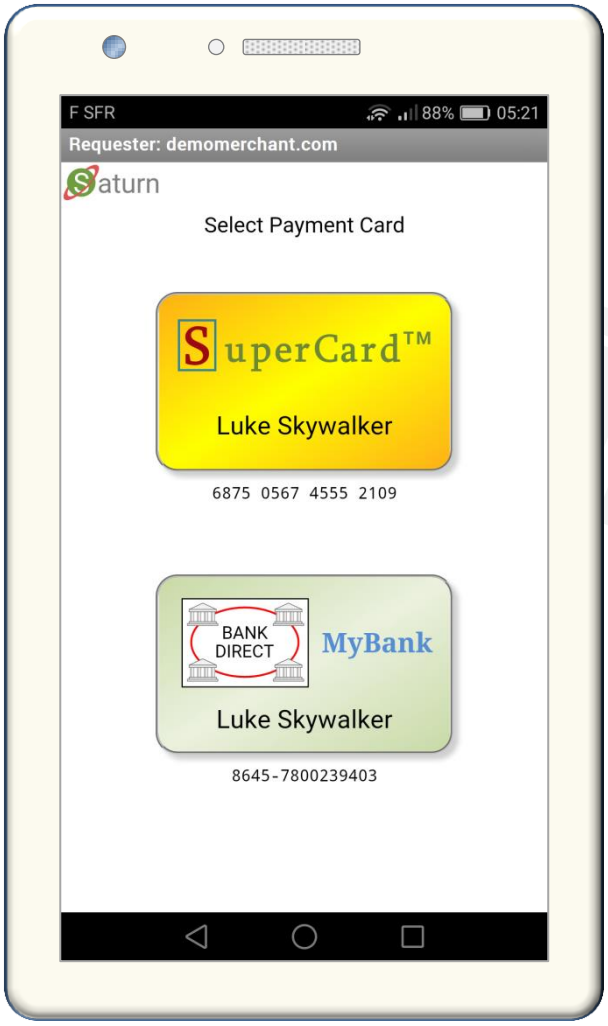
```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "PaymentClientRequest",
  "paymentNetworks": [{
    "paymentMethods": ["https://supercard.com","https://bankdirect.net","https://unusualcard.org"],
    "paymentRequest": {
      "payee": {
        "commonName": "Demo Merchant",
        "id": "86344"
      },
      "amount": "599.00",
      "currency": "EUR",
      "referenceId": "#1000000",
      "timeStamp": "2017-09-09T06:01:36Z",
      "expires": "2017-09-09T06:32:00Z",
      "software": {
        "name": "WebPKI.org - Merchant",
        "version": "1.00"
      },
      "signature": {
        "algorithm": "ES256",
        "publicKey": {
          "kty": "EC",
          "crv": "P-256",
          "x": "rZ344aiTaOATmLBOdfYThvnQu_zyB1aJZrbbbks2P9I",
          "y": "lKOvfJdgN8WqEbXMDYPRSMsPicm0Tk10pmer9LxvxLg"
        },
        "value": "_O4Ta4idtMcAHcRnjyEHkOOkb2 … afRQkUjsnp2LY8wcOn7m4b8OSDA"
      }
    }
  }]
}
```

https://cyberphone.github.io/doc/security/jcs.html

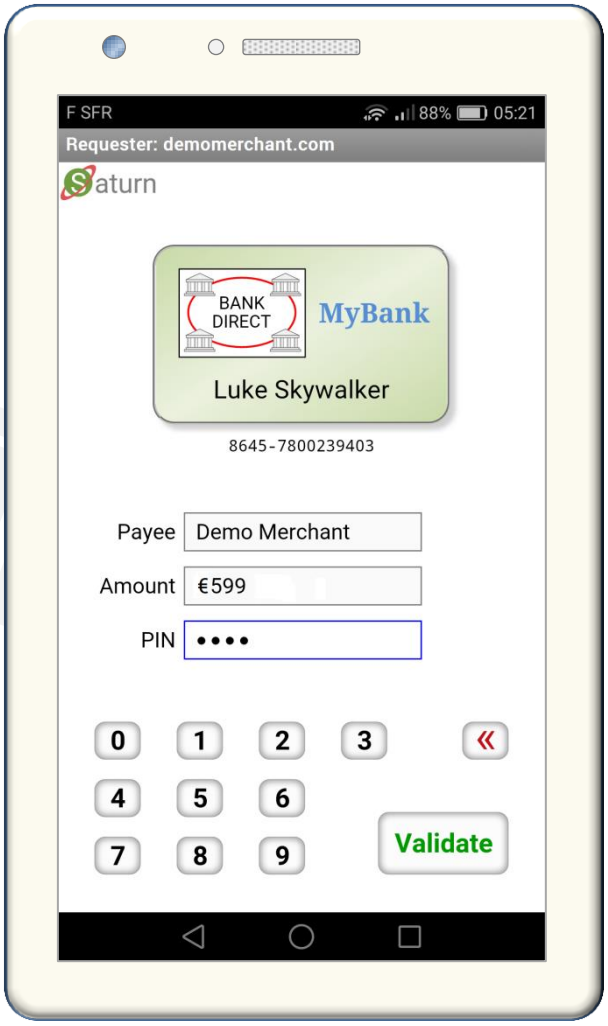
The Merchant (Payee) signs a payment request with its public key. Note that a Merchant may belong to multiple and independent payment networks, where each network typically maintains their own Merchant **`publicKey`** and/or **`id`** data, which is why there is an *array of signed requests* put in a **`PaymentClientRequest`** message. The request array is sent to the client which through a scenario-dependent mechanism (like Web versus NFC), *invokes* the Wallet.

② Wallet Receives the “PaymentClientRequest”

Optional: (more than one matching card)



Authorization UI



When the **`PaymentClientRequest`** has been received by the client, the Wallet user interface is launched. The authorization method may consist of a PIN but could also be a biometric option such as touching a fingerprint reader. The authorization is only used to unlock the Signature Key as described in next slide. Note that the Saturn authorization concept not only emulates payment cards, *but payment terminals as well*.

③ *Internal* Wallet Processing – Creation of *Signed* Authorization Data

```
{
  "requestHash": {
    "algorithm": "S256",
    "value": "eYqbGYkHfAsOUTJiuqfU98Rou_mfn0etWUkvDVOF_Fw"
  },
  "domainName":  "demomerchant.com",
  "paymentMethod": "https://bankdirect.net",
  "accountId": "8645-7800239403",
  "encryptionParameters": {
    "algorithm": "A128CBC-HS256",
    "key": "Ivq5sSrtNNpOvN9t9_pRCfc6dqT3IuVg6H2h9NlHULs"
  },
  "timeStamp": "2017-09-09T08:02:18+02:00",
  "software": {
    "name": "WebPKI.org - Wallet",
    "version": "1.00"
  },
  "signature": {
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "vlYxD4dtFJOp1_8_QUcieWCW-4KrLMmFL2rpkY1bQDs",
      "y": "fxEF70yJenP3SPHM9hv-EnvhG6nXr3_S-fDqoj-F6yM"
    },
    "value": "TDKWQb9idTyPXgpOgIxXeogt … lhC5_dG3uU6MPmqjQLc7jju4f0Q"
  }
}
```

Hash of matching payment request object

Core data of selected virtual card

https://cyberphone.github.io/doc/security/jcs.html

When the user has authorized the transaction the Signature (private) Key associated with the selected card is used to sign a JSON object holding authorization data as follows : **`requestHash`** holds the *hash* of the **`PaymentRequest`** object (see PaymentRequest slide), while **`accountId`** holds the actual Account ID of the selected card. For more information about **`encryptionParameters`**, turn to the slide Risk Based Authentication.

③ Wallet Processing – Creation and Sending of “PayerAuthorization” JSON Message


```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "PayerAuthorization",
  "providerAuthorityUrl": "https://payproc.mybank.com/authority",
  "paymentMethod": "https://bankdirect.net",
  "encryptedAuthorization": {
    "algorithm": "A128CBC-HS256",
    "encryptedKey": {
      "algorithm": "ECDH-ES",
      "publicKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "TfCrhFwZRU_ea7lUWwRi3HkuyT2yF9IxN5xKh2khjlk",
        "y": "nZFwxLP0TvFXD2xPKzRTIGevgLjpiMw2BP86hszj5x4"
      },
      "ephemeralKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "D7zSvy3mbS4WbB2qgKwchLRwQFir5T_p09HpnAi_RqA",
        "y": "gkwNJ2o6BtUASkmp1DO4UvllsQL5zAzvVEHB7t0CqX0"
      }
    },
    "iv": "zyConPq8uA7GFjaTkta-qA",
    "tag": "S8zUQ3tioyYPzbtNBO6Ftw",
    "cipherText": "GLkd4uHnjqL_EX9tssDNLzsZj … s7u2ezBOibNoQN5V3cl2ieB-hLHj4XppJI"
  }
}
```

“Non-secret” data of selected virtual card

https://cyberphone.github.io/doc/security/jef.html

Encryption Key of selected virtual card

Encrypted user authorization

`PayerAuthorization` messages provide the URL to the issuing bank’s “Authority” object and the selected payment method which both are featured in virtual cards. This data is used by Merchants (Payees) for routing payment authorization requests to the applicable Bank. The actual authorization data (see previous slides) is *encrypted* by the Wallet using an *Issuer (not User) specific Encryption Key* (with a *matching private key only known by the issuing Bank*), which also is stored in the virtual card. That is, Merchants do not get any information concerning Users (Payers) except their Bank and associated payment method.

Bank/Acquirer “Authority” JSON Object


```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "ProviderAuthority",
  "httpVersion": "HTTP/1.1",
  "authorityUrl": "https://payproc.mybank.com/authority",
  "homePage": "https://mybank.com",
  "serviceUrl": "https://payproc.mybank.com/service",
  "paymentMethods": ["https://sepa.payments.org","https://ultragiro.se"],
  "signatureProfiles": ["http://webpki.org/saturn/v3/signatures#P-256.ES256"],
  "encryptionParameters": [{
    "dataEncryptionAlgorithm": "A128CBC-HS256",
    "keyEncryptionAlgorithm": "ECDH-ES",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TfCrhFwZRU_ea7lUWwRi3HkuyT2yF9IxN5xKh2khjlk",
      "y": "nZFwxLP0TvFXD2xPKzRTIGevgLjpiMw2BP86hszj5x4"
    }
  }],
  "timeStamp": "2017-09-09T04:34:29Z",
  "expires": "2017-09-09T05:34:31Z",
  "signature": {
    "algorithm": "ES256",
    "signerCertificate": {
      "issuer": "CN=Payment Network Sub CA3,C=EU",
      "serialNumber": "1461174553809",
      "subject": "CN=mybank.com,2.5.4.5=#130434353031,C=FR"
    },
    "certificatePath": ["MIIBtTCCAVmgAwIBAgIGAVQ0 … 9Ly9t7A-jMuGl3FwxFeOawwmz1bM6",
                        "MIIDcjCCAVqgAwIBAgIBAzANB … W9x5ZxVhvpP_We_5TddhlTUMNPvw"],
    "value": "cdRqFlzVEou5Zj-EqWGCCLtxY … JkEBD4fFOqVnU9dstv_P2BoHQ"
  }
}
```


Authority object URL of a virtual payment card issued by this bank

Accepted signature types

https://cyberphone.github.io/doc/security/jef.html

https://cyberphone.github.io/doc/security/jcs.html

“Authority” objects hold Keys, Payment methods, and URLs which are used by Merchants, Banks, and Acquirers as *Secure Distributed Entity Databases* creating the foundation for scalability including *Delegated Trust*. “Authority” objects are *published on the Internet* and accessed by HTTP GET operations. A Bank/Acquirer “Authority” object is signed by the Bank/Acquirer itself. The `paymentMethods` array declares the payment methods understood by the Bank. The `encryptionParameters` are used by Issuers for encrypting card data to Acquirers.

Merchant (Payee) “Authority” JSON Object


```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "PayeeAuthority",
  "authorityUrl": "https://saturn.bigbank.com/payees/86344",
  "providerAuthorityUrl": "https://saturn.bigbank.com/authority",
  "homePage": "https://demomerchant.com",
  "commonName": "Demo Merchant",
  "id": "86344",
  "signatureParameters": [{
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "rZ344aiTaOATmLBOdfYThvnQu_zyB1aJZrbbbks2P9I",
      "y": "lKOvfJdgN8WqEbXMDYPRSMsPicm0Tk10pmer9LxvxLg"
    }
  }],
  "timeStamp": "2017-09-09T16:26:57Z",
  "expires": "2017-09-09T17:26:58Z",
  "signature": {
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "-Vr8Wk3ygt5J2_J3R8TrRaa-AWW7ZiXa6q1P7ELs6gc",
      "y": "Vuc6z3WiZ3tgXTXvU6F5qdiiYePWeUI1q9Tx83ySDcM"
    },
    "value": "CKAfEg7wzCoxGiCvRAMVIug0RKU … 4AOtKZK_RPNoshOGVnxry7vQZeIuIw"
  }
}
```


URL to the Merchant’s “Authority” object

URL to the Merchant’s Bank/Acquirer “Authority” object

Merchant core data

https://cyberphone.github.io/doc/security/jcs.html

The same key as in the Bank/Acquirer “Authority” object signature certificate

A Merchant (Payee) “Authority” object is like a *short-lived*, *automatically updated*, *X.509 certificate not requiring a CA*. Such an object is published on the address **`authorityUrl`** hosted by the party (Bank or Acquirer) which vouches for the Merchant. If a Merchant is to be “revoked”, the object is simply removed. To automate revocation checks, there is an **`expires`** attribute which also is used to clear caching of Merchant “Authority” objects. The **`signatureParameters`** list enable key renewals as well as validation of signatures using old keys.

④ Merchant Creates and Sends an “AuthorizationRequest” JSON Message

```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "AuthorizationRequest",
  "recepientUrl": "https://payproc.mybank.com/service",
  "authorityUrl": "https://saturn.bigbank.com/payees/86344",
  "paymentMethod": "https://bankdirect.net",
  "paymentRequest": {
        As selected by the Wallet/User…
  },
  "encryptedAuthorization": {
        As created by the Wallet…
  },
  "paymentMethodSpecific": {
        Each method defines its own data…
  },
  "referenceId": "#1000005",
  "clientIpAddress": "224.165.21.50",
  "timeStamp": "2017-09-09T06:02:20Z",
  "software": {
    "name": "WebPKI.org - Payee",
    "version": "1.00"
  },
  "signature": {
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "rZ344aiTaOATmLBOdfYThvnQu_zyB1aJZrbbbks2P9I",
      "y": "lKOvfJdgN8WqEbXMDYPRSMsPicm0Tk10pmer9LxvxLg"
    },
    "value": "G4ct46eTx-GgF2qrSnHKRR9f9Ajd … ju85d56gSON2M3I20-u6sfcejw"
  }
}
```

Where the message is actually sent

URL to Merchant “Authority” object

Payment method (must match the encrypted user authorization)

Sample data for a “SEPA-inspired” payment method:

```
"@context": "https://sepa.payments.org/saturn/v3#pms",
"payeeIban": "FR7630004003200001019471656"
```

https://cyberphone.github.io/doc/security/jcs.html

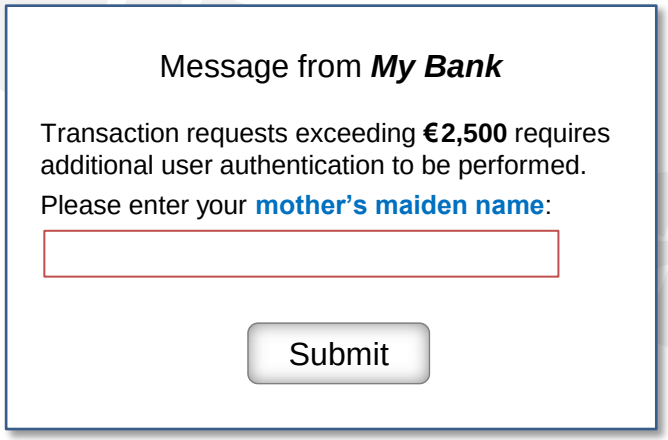
The counter-signed **`AuthorizationRequest`** is sent by a Merchant (Payee) to the **`serviceUrl`** of the “Authority” object given by the user’s choice of payment card (method). See providerAuthorityUrl. The inclusion of **`authorityUrl`** enables the targeted User Bank to verify that the Merchant belongs to a known Bank-to-Bank or Acquirer payment network.

Private Messaging and Risk Based Authentication


```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "ProviderUserResponse",
  "encryptedMessage": {
    "algorithm": "A128CBC-HS256",
    "iv": "cht7SYItQF8LO3QBg2bbbA",
    "tag": "33orw76LP7YibQqKPmKURA",
    "cipherText": "9PyK-rlhb41oBXVSdYa0Ats9 … RBxdxqdGVbLf07Qw8Tr2LblxNYPUEc"
  }
}
```


https://cyberphone.github.io/doc/security/jef.html

When decrypted and rendered by the Wallet (sample)



Occasionally a User Bank needs to inform the user of something related to an AuthorizationRequest like an account overdraft. Another situation requiring an action from the user's side is when the amount requested is unusually high or when "suspicious" user patterns have been identified. In both cases the request is *ignored* and the normal response is replaced by a message which only the (Wallet) user can read while still being delivered through the Merchant's "channel" to the Wallet. Privacy is accomplished by the User Bank encrypting the message contents with the symmetric key supplied in **`encryptionParameters`**
(see Creation of Signed Authorization Data). This key is a random value generated for each Wallet invocation.

A private message like above (requiring an *action*), forces the Wallet adding the response to the user authorization data and then performing a full signed and encrypted User Authorization request again. This process may be repeated if necessary.

Refund Option


```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "AuthorizationResponse",
  "authorizationRequest": {
    "@context": "http://webpki.org/saturn/v3",
    "@qualifier": "AuthorizationRequest",
          Removed for brevity
    }
  },
  "encryptedAccountData": {
    "algorithm": "A128CBC-HS256",
    "encryptedKey": {
      "algorithm": "ECDH-ES",
      "publicKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "DOOwVwUyNdgu4dZ9Ej7pg9j4SDLfGlrzoWso2DIz6ts",
        "y": "WF7ZApRPkbigS4iNoz5-SgPYU-_4891TwHJr-fU4d1w"
      },
      "ephemeralKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "o2Dj9uXUjq2i8mqc0roFEA0ynRC4xZ_4okp9Sr2s50U",
        "y": "Ax8TexrghpeFBt2pZZ0RoIdFUkfkFKeuKSaArkfAnO0"
      }
    },
    "iv": "XHz7Q9AhDk6Y5SaqsR7Lzw",
    "tag": "MXzvGwciTicYxIgUaMKpSg",
    "cipherText": "6Q4ngAYxZJjKmHoo0LPtWXl6BI8sXdsK....LMQA29lLvq8z8Ku4_bYUEfMMyUbrLp-Y"
  },
      Removed for brevity
}
```


https://cyberphone.github.io/doc/security/jef.html

By including the full account ID of the user (but *encrypted* with the Merchant's *payment provider key*), in the **`AuthorizationResponse`** object the Merchant can (aided by their payment provider), transfer money in the opposite direction. A **`RefundRequest`** message (not shown here) in essence consists of an embedded **`AuthorizationResponse`** and an *amount*, signed by the Merchant.

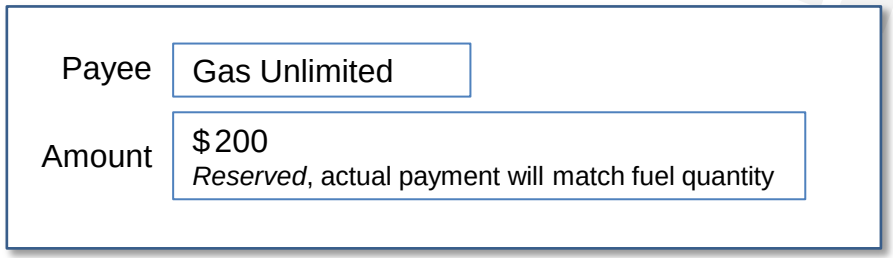
Gas Station Option (Reserve/Capture)

```
{
  "@context": "http://webpki.org/saturn/v3",
  "@qualifier": "PaymentClientRequest",
  "paymentNetworks": [{
    "paymentMethods": ["https://supercard.com","https://bankdirect.net"],
    "paymentRequest": {
      "payee": {
        "commonName": "Gas Unlimited",
        "id": "86344"
      },
      "amount": "200.00",
      "currency": "USD",
      "nonDirectPayment": "GAS_STATION",
          Removed for brevity…
    }
  }]
}
```

For details see PaymentClientRequest

New element

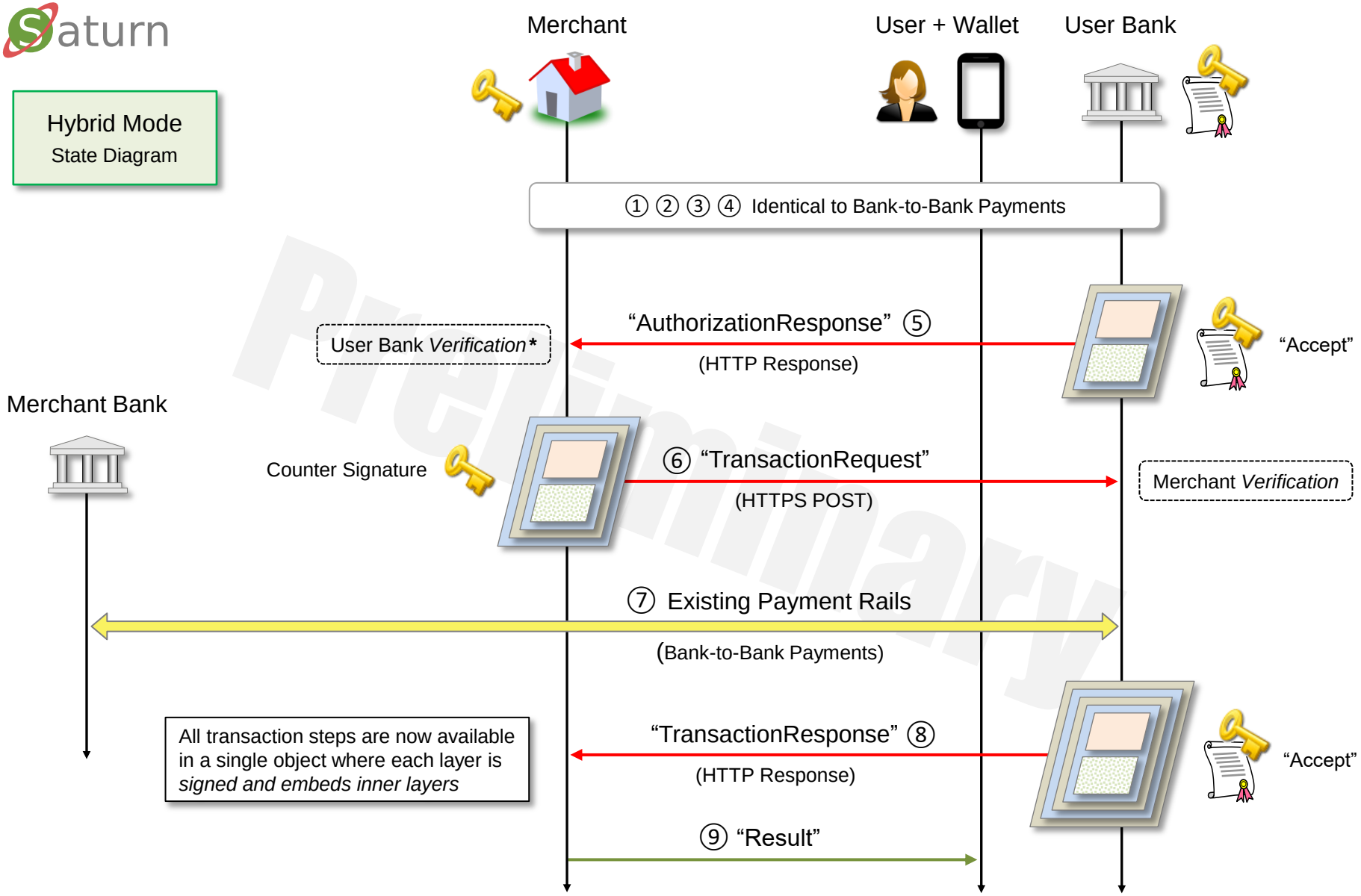
User Interface Implications (Non-normative)



Gas Station payments presume Card Payments or Hybrid Mode to be carried out.

Saturn

Hybrid Mode
State Diagram



***** See Authority Objects. In the Hybrid mode traditional card payment methods are "emulated" including support for hotel bookings, upfront reservations for automated gas stations, as well as reoccurring payments. For three-corner payment schemes like PayPal and Alipay as well as for payments where the User and Merchant have the same bank, *step #7 is not applicable*.

Saturn - FAQ

Q: Doesn't Saturn's Merchant-to-User Bank AuthorizationRequest introduce security risks?
A: Yes, similar risks as requests from a Wallet or Mobile banking App (since there's no way you can see if an App is "hacked").
Security features include:
- Small and strict message format
- Signed by the Merchant which in turn is vouched for by the Merchant's Bank/Acquirer through the PayeeAuthority object
- Signed by the User by a key *which only the User Bank knows about*
- Time stamped by client
- Integral support for RBA (Risk Based Authentication)

Q: Can you trust the Wallet key storage?
A: Apple Pay store keys in a "Secure Enclave". Saturn would need something similar like:
https://cyberphone.github.io/doc/security/sks-api-arch.pdf

Q: Doesn't Saturn effectively requires new client-side technology to fly?
A: Yes indeed, exactly like Apple Pay did.

Q: Wouldn't it be better sending requests from the Wallet directly to the User Bank and then handing over responses to the Merchant?
A: No, see https://cyberphone.github.io/doc/defensive-publications/payment-authorization-scheme.pdf for more details on this matter which also is a *prerequisite for Wallet payment method independence*.

Q: Is Saturn a "Push" or "Pull" payment system?
A: *Saturn is not a payment system*, it is rather a scheme where a User *authorizes* Merchant-initiated *requests** which are transported back to the User Bank via the Merchant. That is, the actual payment system is not a part of the depicted scheme.

Q: How does Saturn relate to ISO 20022, ISO 8583, and SEPA?
A: Not at all since only the actual payment systems need payment-system-specific security, format, names, conventions, and processing. User authorization on the other hand appears to be quite feasible to standardize.

Q: How are Virtual Cards enrolled?
A: Virtual Cards would typically be enrolled from the User Bank's Web site using a secure enrollment protocol like:
https://cyberphone.github.io/doc/security/keygen2.html

Q: Is Saturn a REST API?
A: No, because in REST an operation is also defined by the HTTP verb (GET, POST, etc.) and URL which is in conflict with the message embedding concept featured in Saturn. In Saturn, messages are uniquely defined by their JSON contents making *digitally signing*, *embedding*, *debugging*, and *documenting* straightforward. Wallet communication is based on an *interactive*, *scenario-dependent*, *asynchronous*, *bi-directional* message channel. See: https://cyberphone.github.io/doc/web/yasmin.html

* Enabling Saturn supporting not only direct payments, but bookings, reoccurring payments, and automated gas station payments without modifications to the underlying payment system