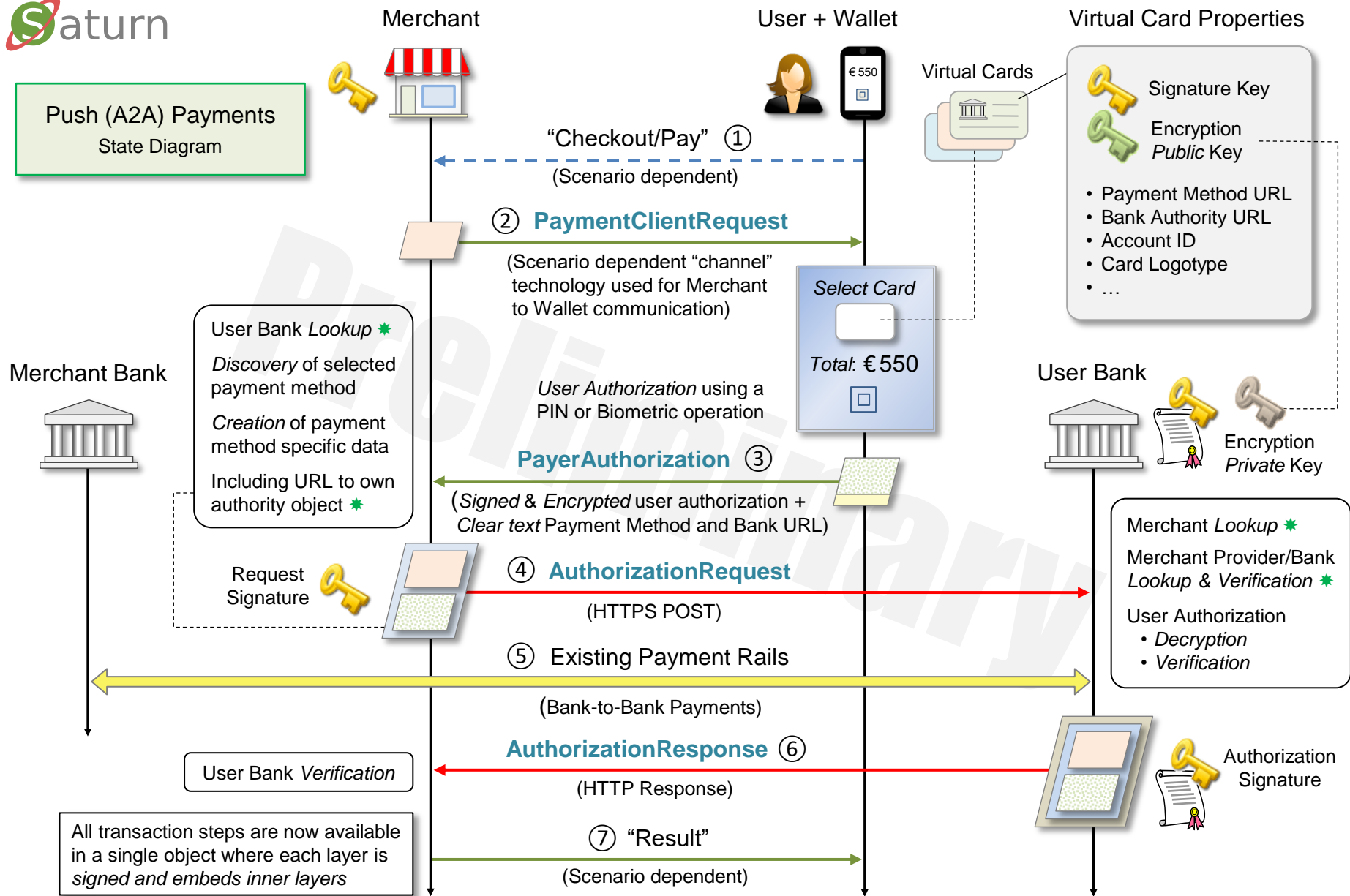# Saturn™

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 account-to-account 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 recurring 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

Push (A2A) Payments
State Diagram

Merchant

User + Wallet

Virtual Card Properties

Virtual Cards

Signature Key

Encryption *Public* Key

- Payment Method URL
- Bank Authority URL
- Account ID
- Card Logotype
- …

"Checkout/Pay" ①
(Scenario dependent)

② **PaymentClientRequest**
(Scenario dependent "channel" technology used for Merchant to Wallet communication)

Select Card

Total: € 550

Merchant Bank

User Bank *Lookup* ✳

Discovery of selected payment method

Creation of payment method specific data

Including URL to own authority object ✳

User Authorization using a PIN or Biometric operation

PayerAuthorization ③
(*Signed* & *Encrypted* user authorization + *Clear text* Payment Method and Bank URL)

User Bank

Encryption *Private* Key

Merchant *Lookup* ✳

Merchant Provider/Bank *Lookup & Verification* ✳

User Authorization
- *Decryption*
- *Verification*

Request Signature

④ **AuthorizationRequest**
(HTTPS POST)

⑤ Existing Payment Rails
(Bank-to-Bank Payments)

AuthorizationResponse ⑥
(HTTP Response)

Authorization Signature

User Bank *Verification*

All transaction steps are now available in a single object where each layer is *signed and embeds inner layers*
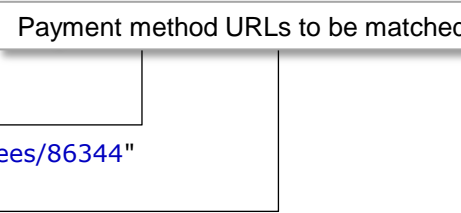
⑦ "Result"
(Scenario dependent)

✳ 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.
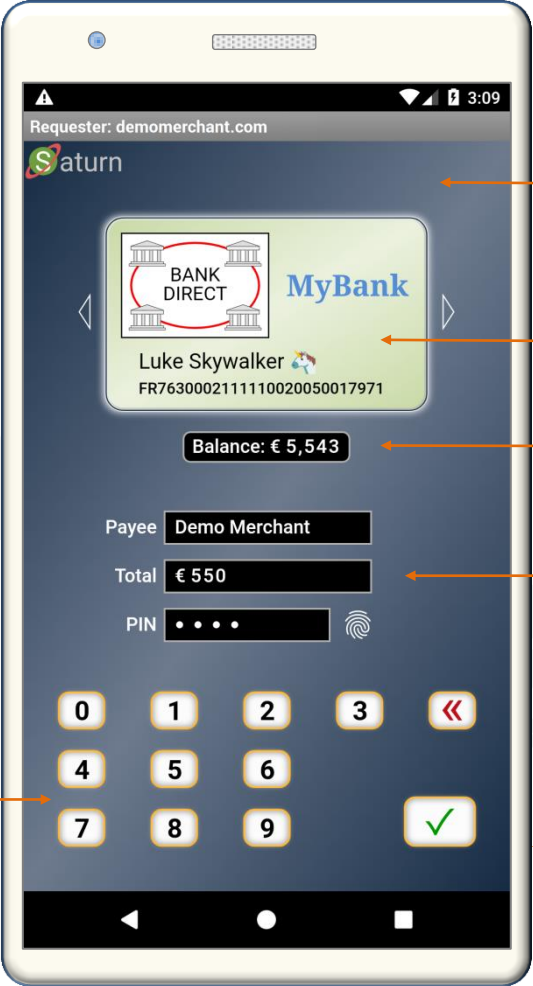
```json
{
  "@context": "https://webpki.github.io/saturn/v3",
  "@qualifier": "PaymentClientRequest",
  "supportedPaymentMethods": [{
    "paymentMethod": "https://bankdirect.net",
    "payeeAuthorityUrl": "https://payments.bigbank.com/payees/86344"
  },{
    "paymentMethod": "https://supercard.com",
    "payeeAuthorityUrl": "https://secure.cardprocessor.com/payees/1077342"
  }],
  "paymentRequest": {
    "commonName": "Demo Merchant",
    "amount": "550.00",
    "currency": "EUR",
    "referenceId": "2020092400000010",
    "timeStamp": "2020-09-24T08:31:44Z",
    "expires": "2020-09-24T09:02:00Z"
  },
  "receiptUrl": "https://demomerchant.com/receipts/2020092400000010jzuJi1bFDbBSM_tuzN_4Nw"
}
```

Payment method URLs to be matched against stored virtual cards

The Merchant (Payee) invokes the Wallet (after a user action) with a list of supported payment methods. Those who are matching the user's virtual cards will be shown in the Wallet UI to select from. The `payeeAuthorityUrl` properties are used to securely bind the Merchant's request signature to a particular payment method / network. The `paymentRequest` object contains the actual request data to be reflected in the wallet UI. The *optional* `receiptUrl` points to a Web address where an associated digital receipt will be published. See TO BE WRITTEN….

Saturn

Adapted to:
- *Your* Language
- *Your* Disability

Virtual Card Logotype & Account Selector (⇐ Swipe ⇨)
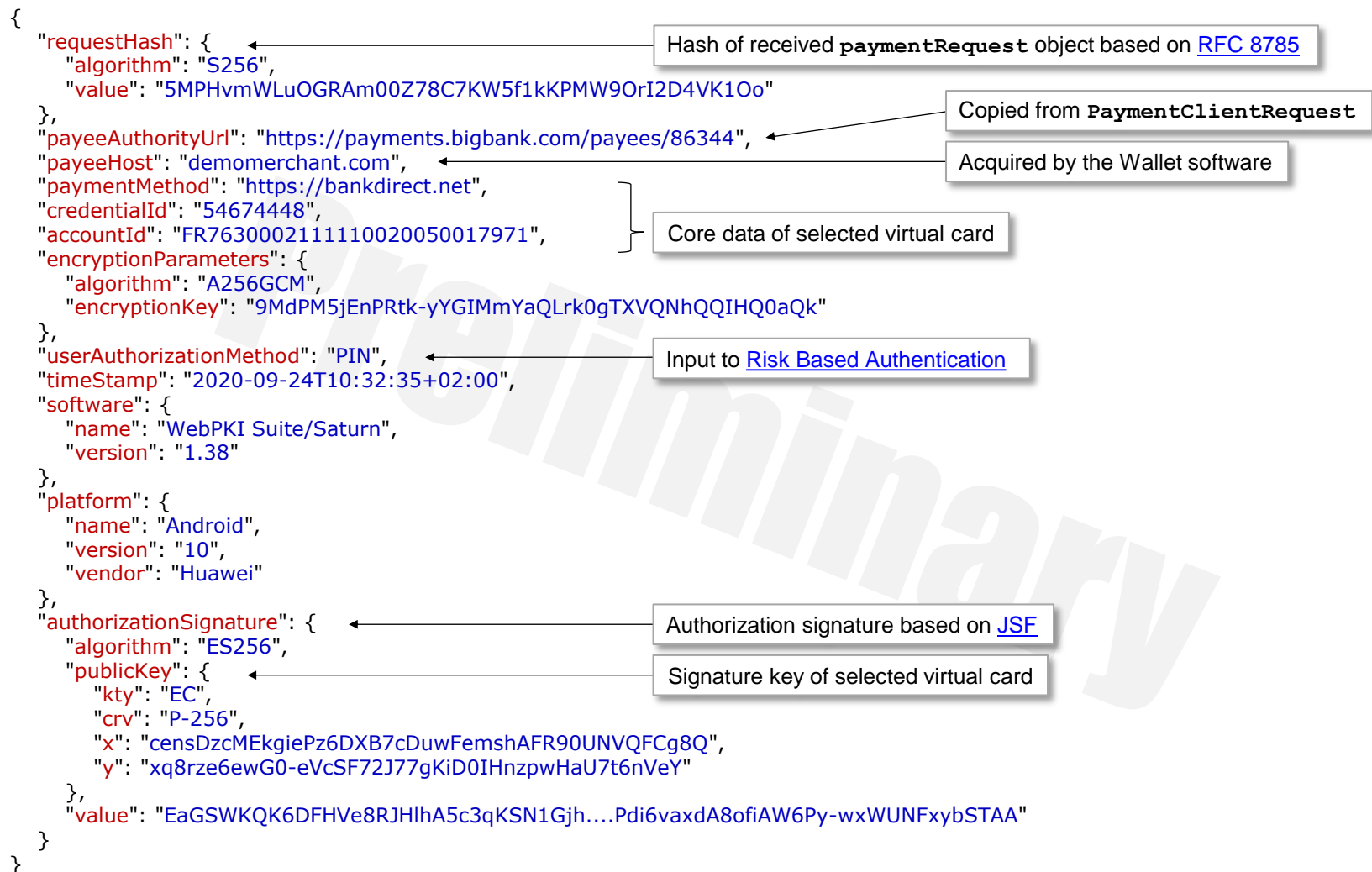
Optional: Real-Time Account Balance

UI Showing:
- Direct Payment
- Booking
- Gas Station
- Subscription
- Etc.

TEE Protected Keys

Requester: demomerchant.com

Saturn

BANK DIRECT — **MyBank**

Luke Skywalker 🏂
FR7630002111110020050017971

Balance: € 5,543

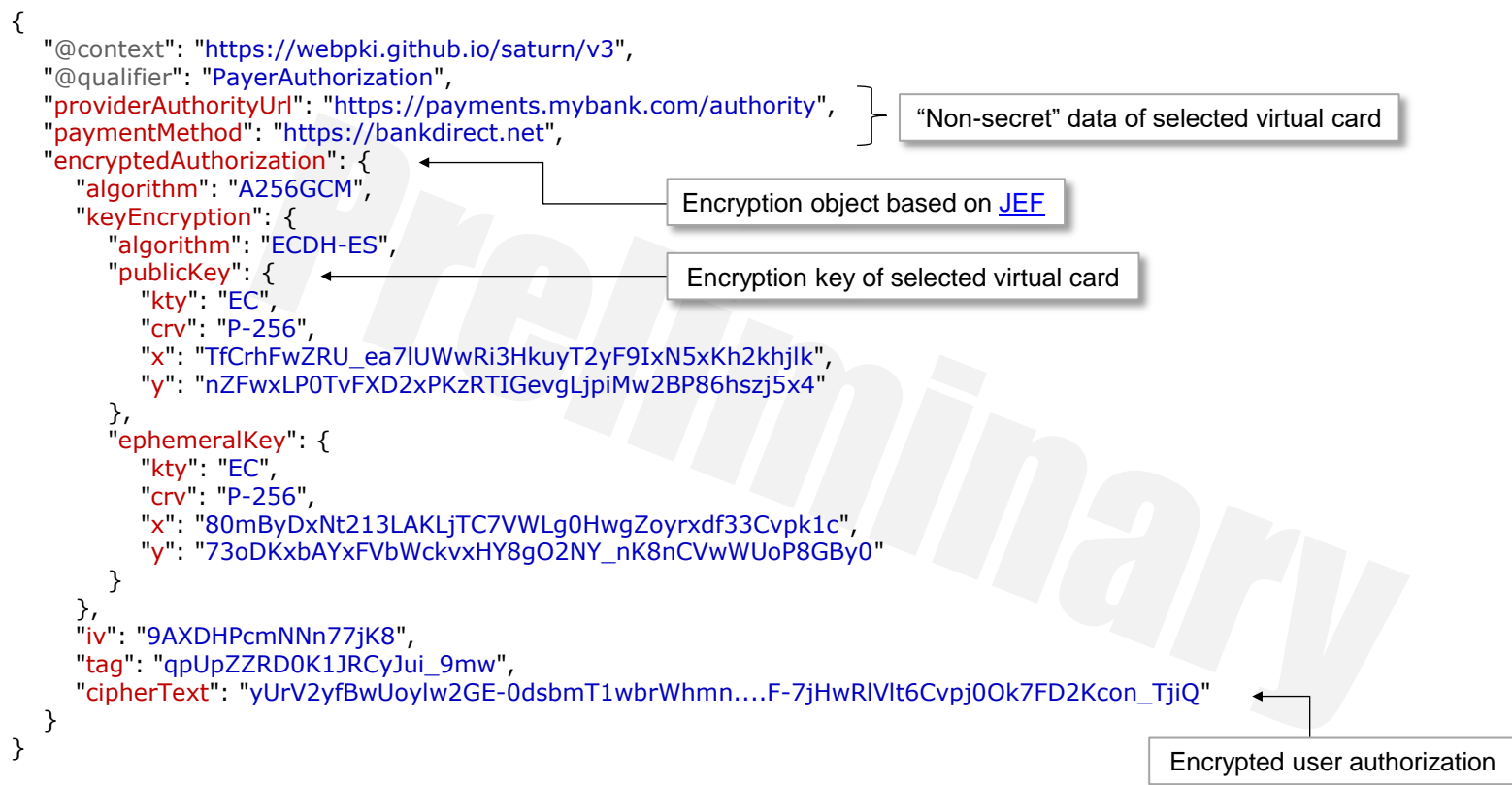| Payee | Demo Merchant |
| Total | € 550 |
| PIN | • • • • |

0 1 2 3 «
4 5 6
7 8 9 ✓

PIN or Biometric for User Authorization
(as defined by the *virtual card issuer*, not the Wallet)

When **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*.

```
{
    "requestHash": {
        "algorithm": "S256",
        "value": "5MPHvmWLuOGRAm00Z78C7KW5f1kKPMW9OrI2D4VK1Oo"
    },
    "payeeAuthorityUrl": "https://payments.bigbank.com/payees/86344",
    "payeeHost": "demomerchant.com",
    "paymentMethod": "https://bankdirect.net",
    "credentialId": "54674448",
    "accountId": "FR7630002111110020050017971",
    "encryptionParameters": {
        "algorithm": "A256GCM",
        "encryptionKey": "9MdPM5jEnPRtk-yYGIMmYaQLrk0gTXVQNhQQIHQ0aQk"
    },
    "userAuthorizationMethod": "PIN",
    "timeStamp": "2020-09-24T10:32:35+02:00",
    "software": {
        "name": "WebPKI Suite/Saturn",
        "version": "1.38"
    },
    "platform": {
        "name": "Android",
        "version": "10",
        "vendor": "Huawei"
    },
    "authorizationSignature": {
        "algorithm": "ES256",
        "publicKey": {
            "kty": "EC",
            "crv": "P-256",
            "x": "censDzcMEkgiePz6DXB7cDuwFemshAFR90UNVQFCg8Q",
            "y": "xq8rze6ewG0-eVcSF72J77gKiD0IHnzpwHaU7t6nVeY"
        },
        "value": "EaGSWKQK6DFHVe8RJHlhA5c3qKSN1Gjh....Pdi6vaxdA8ofiAW6Py-wxWUNFxybSTAA"
    }
}
```

Hash of received **paymentRequest** object based on RFC 8785

Copied from **PaymentClientRequest**

Acquired by the Wallet software

Core data of selected virtual card

Input to Risk Based Authentication

Authorization signature based on JSF

Signature key of selected virtual card

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 PaymentClientRequest slide), while **accountId** holds the actual Account ID of the selected card. For more information about **encryptionParameters**, turn to the slide Risk Based Authentication.

```json
{
  "@context": "https://webpki.github.io/saturn/v3",
  "@qualifier": "PayerAuthorization",
  "providerAuthorityUrl": "https://payments.mybank.com/authority",
  "paymentMethod": "https://bankdirect.net",
  "encryptedAuthorization": {
    "algorithm": "A256GCM",
    "keyEncryption": {
      "algorithm": "ECDH-ES",
      "publicKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "TfCrhFwZRU_ea7lUWwRi3HkuyT2yF9IxN5xKh2khjIk",
        "y": "nZFwxLP0TvFXD2xPKzRTIGevgLjpiMw2BP86hszj5x4"
      },
      "ephemeralKey": {
        "kty": "EC",
        "crv": "P-256",
        "x": "80mByDxNt213LAKLjTC7VWLg0HwgZoyrxdf33Cvpk1c",
        "y": "73oDKxbAYxFVbWckvxHY8gO2NY_nK8nCVwWUoP8GBy0"
      }
    },
    "iv": "9AXDHPcmNNn77jK8",
    "tag": "qpUpZZRD0K1JRCyJui_9mw",
    "cipherText": "yUrV2yfBwUoylw2GE-0dsbmT1wbrWhmn....F-7jHwRlVlt6Cvpj0Ok7FD2Kcon_TjiQ"
  }
}
```

"Non-secret" data of selected virtual card

Encryption object based on JEF

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.

```
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "ProviderAuthority",
    "httpVersions": ["HTTP/1.1", "HTTP/2"],
    "providerAuthorityUrl": "https://payments.mybank.com/authority",
    "commonName": "My Bank",
    "homePage": "https://mybank.com",
    "serviceUrl": "https://payments.mybank.com/service",
    "supportedPaymentMethods": {
        "https://bankdirect.net": ["https://sepa.payments.org/saturn/v3#account"],
        "https://supercard.com": ["https://sepa.payments.org/saturn/v3#account"]
    },
    "extensions": {
        "https://webpki.github.io/saturn/v3/extensions#hybrid": "https://payments.mybank.com/hybridpay",
        "https://webpki.github.io/saturn/v3/extensions#balance": "https://payments.mybank.com/balancereq"
    },
    "signatureProfiles": [
        "https://webpki.github.io/saturn/v3/signatures#ES256.P-256",
        "https://webpki.github.io/saturn/v3/signatures#RS256.2048"
    ],
    "encryptionParameters": [{
        "dataEncryptionAlgorithm": "A128CBC-HS256",
        "keyEncryptionAlgorithm": "ECDH-ES",
        "publicKey": {
            "kty": "EC",
            "crv": "P-256",
            "x": "TfCrhFwZRU_ea7lUWwRi3HkuyT2yF9IxN5xKh2khjlk",
            "y": "nZFwxLP0TvFXD2xPKzRTIGevgLjpiMw2BP86hszj5x4"
        }
    }],
    "timeStamp": "2020-09-24T08:05:46Z",
    "expires": "2020-09-24T09:05:47Z",
    "issuerSignature": {
        "algorithm": "ES256",
        "certificatePath": ["MIIBtTCCAVmgAwIB....3FwxFeOawwmz1bM6", "MIIDcjCCAVqgAwIB....e_-5TddhlTUMNPvw"],
        "value": "ibiFHsP9DizZQ8k6DUjP9zrR3Mkg5v1L....xiL58nHozxZUOWJODQ4MNqBxWWiooMpw"
    }
}
```

Authority object (self) URL of a virtual payment card issuer
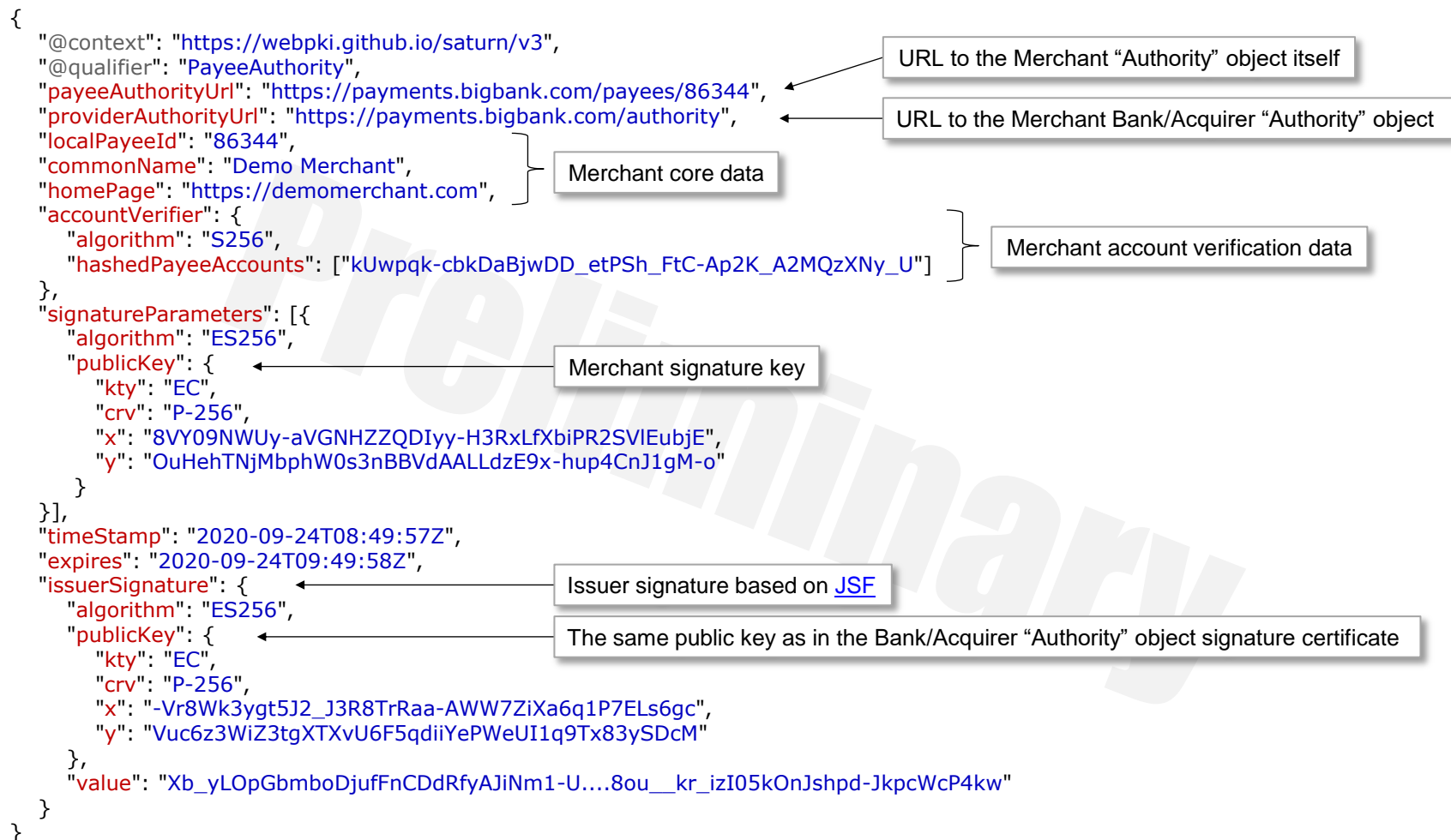
Payment network, [Payment rails, …]

Accepted signature types

Encryption parameters adapted for JEF

Issuer signature based on JSF

"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 `supportedPaymentMethods` object declares the payment methods understood by the Bank. The `encryptionParameters` are used by Issuers for encrypting user account data.

Saturn

```
{
  "@context": "https://webpki.github.io/saturn/v3",
  "@qualifier": "PayeeAuthority",
  "payeeAuthorityUrl": "https://payments.bigbank.com/payees/86344",
  "providerAuthorityUrl": "https://payments.bigbank.com/authority",
  "localPayeeId": "86344",
  "commonName": "Demo Merchant",
  "homePage": "https://demomerchant.com",
  "accountVerifier": {
    "algorithm": "S256",
    "hashedPayeeAccounts": ["kUwpqk-cbkDaBjwDD_etPSh_FtC-Ap2K_A2MQzXNy_U"]
  },
  "signatureParameters": [{
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "8VY09NWUy-aVGNHZZQDIyy-H3RxLfXbiPR2SVlEubjE",
      "y": "OuHehTNjMbphW0s3nBBVdAALLdzE9x-hup4CnJ1gM-o"
    }
  }],
  "timeStamp": "2020-09-24T08:49:57Z",
  "expires": "2020-09-24T09:49:58Z",
  "issuerSignature": {
    "algorithm": "ES256",
    "publicKey": {
      "kty": "EC",
      "crv": "P-256",
      "x": "-Vr8Wk3ygt5J2_J3R8TrRaa-AWW7ZiXa6q1P7ELs6gc",
      "y": "Vuc6z3WiZ3tgXTXvU6F5qdiiYePWeUI1q9Tx83ySDcM"
    },
    "value": "Xb_yLOpGbmboDjufFnCDdRfyAJiNm1-U....8ou__kr_izI05kOnJshpd-JkpcWcP4kw"
  }
}
```

Annotations:
- URL to the Merchant "Authority" object itself
- URL to the Merchant Bank/Acquirer "Authority" object
- Merchant core data
- Merchant account verification data
- Merchant signature key
- Issuer signature based on JSF
- The same public 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 `payeeAuthorityUrl` 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.

Saturn

```json
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "AuthorizationRequest",
    "recipientUrl": "https://payments.mybank.com/service",
    "payeeAuthorityUrl": "https://payments.bigbank.com/payees/86344",
    "paymentMethod": "https://bankdirect.net",
    "paymentRequest": {

        Copy of the paymentRequest object
    },
    "encryptedAuthorization": {

        Copy of the encryptedAuthorization object
    },
    "payeeReceiveAccount": {

    },
    "clientIpAddress": "220.13.198.144",
    "timeStamp": "2020-09-24T08:32:38Z",
    "software": {
        "name": "WebPKI.org - Payee",
        "version": "1.00"
    },
    "requestSignature": {
        "algorithm": "ES256",
        "publicKey": {
            "kty": "EC",
            "crv": "P-256",
            "x": "8VY09NWUy-aVGNHZZQDIyy-H3RxLfXbiPR2SVlEubjE",
            "y": "OuHehTNjMbphW0s3nBBVdAALLdzE9x-hup4CnJ1gM-o"
        },
        "value": "91wNxmoZt-TKUGD1R7prluueL2DSv9iZ....TqYipTRDXSewSlfWgnoxsTkjkw07pJog"
    }
}
```

Where the message is actually sent

URL to Merchant "Authority" object

Payment method (must match user authorization)

Sample data for a SEPA payment method:

"@context": "https://sepa.payments.org/saturn/v3#account",
"iban": "FR7630004003200001019471656",
"nonce": "nZFwxLP0TvFXD2xPKzRTIGevgLjpiMw2BP86hszj5x4"

The **nonce** is adding privacy protection to the account verification hashes published in the **accountVerifier** object of PayeeAuthority.

Request signature based on JSF

Merchant signature key

The **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 **payeeAuthorityUrl** enables the targeted User Bank to verify that the Merchant belongs to a known payment network.

Saturn

```
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "AuthorizationResponse",
    "accountReference": "FR*0504",
    "encryptedAccountData": {

        Parameters removed for brevity…

        "cipherText": "okjRig8y97oHa0kw7buu17XcTZOZAtS1....XG4BoMqDwY0e2fxlGPSHzko5Hs_0UHXz"
    },
    "referenceId": "#0100345648",
    "logData": "CT100006",
    "timeStamp": "2020-09-24T08:32:38Z",
    "software": {
        "name": "WebPKI.org - Bank",
        "version": "1.00"
    },
    "authorizationRequest": {

        Copy of the entire AuthorizationRequest message

    },
    "authorizationSignature": {
        "algorithm": "ES256",
        "certificatePath": ["MIIBtTCCAVmgAwIB....3FwxFeOawwmz1bM6", "MIIDcjCCAVqgAwIB....e_-5TddhlTUMNPvw"],
        "value": "b03W5RPCmoA2ARILtbdvCrlrAj5i0Cr4....hib3XUqun9KxpbL6Ig7i4pA_ko7Gf4yA"
    }
}
```

Optional short form of the user account to be featured in receipts etc.

Encryption object based on JEF

Encrypted user account data

Authorization signature based on JSF

User Bank certificate path

After received the **AuthorizationRequest**, User Bank performs an extensive list of operations to verify the validity of the request, including fetching the Merchant's (Payee) PayeeAuthority and ProviderAuthority objects. If the verification succeeds, User Bank responds with an **AuthorizationResponse** message which in addition to the original **AuthorizationRequest**, also holds the user's account data (ID) encrypted by the Merchant provider's encryption key. This information is used for Card Payments and Refunds.

Saturn

Card Payments
State Diagram

Merchant

User + Wallet

User Bank

① ② ③ ④ Identical to Bank-to-Bank Payments

Acquirer

Acquirer *Lookup* ✳

Card Data *Encryption*

AuthorizationResponse ⑤

User Bank *Verification*

(HTTP Response)

Authorization
Signature

Merchant *Verification*

Card Data *Decryption*

TransactionRequest ⑥

(HTTPS POST)

Request
Signature

Card
Network

⑦ See also Hybrid Mode

Authorization
Signature

⑧ **TransactionResponse**

(HTTP Response)

All transaction steps are now available
in a single object where each layer is
signed and embeds inner layers

⑨ "Result"

(Scenario dependent)

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

Saturn

Hybrid Mode
State Diagram

Merchant

User + Wallet

User Bank

€ 550

① ② ③ ④ Identical to Bank-to-Bank Payments

AuthorizationResponse ⑤

User Bank *Verification*

(HTTP Response)

Authorization Signature

Merchant Bank

Request Signature

⑥ **TransactionRequest**

(HTTPS POST)

Merchant *Verification*

⑦ Existing Payment Rails

(Bank-to-Bank Payments)

All transaction steps are now available in a single object where each layer is *signed and embeds inner layers*

TransactionResponse ⑧

(HTTP Response)

Authorization Signature

⑨ "Result"

(Scenario dependent)

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*.

Private Messaging and Risk Based Authentication

```
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "ProviderUserResponse",
    "encryptedMessage": {
        "algorithm": "A256GCM",
        "iv": "_K4Sgt5y1uKhwiSi",
        "tag": "Xmqyx5XZWmxSFfypag-y_A",
        "cipherText": "qXIsLsZ-zIxVllV920dpxPmTOwGRghU_....fsxbw1LX61Tu6GbsSw1gXEcwkW8S4fOQ"
    }
}
```

Encryption object based on JEF

Encrypted message from User Bank

Decrypted and rendered by the Wallet
(non-normative sample)

Message from *My Bank*

Transaction requests exceeding **€1,000** require
additional user authentication to be performed.
Please enter your mother's maiden name.

[••••••••]

[Submit]

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.

```json
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "RefundRequest",
    "recipientUrl": "https://payments.bigbank.com/refund",
    "amount": "550.00",
    "payeeSourceAccount": {
        "@context": "https://sepa.payments.org/saturn/v3#account",
        "iban": "FR7630004003200001019471656"
    },
    "referenceId": "#1000004",
    "timeStamp": "2020-09-24T22:07:50Z",
    "software": {
        "name": "WebPKI.org - Payee",
        "version": "1.00"
    },
    "authorizationResponse": {

        Copy of the entire AuthorizationResponse message
    },
    "requestSignature": {            ◄——— Request signature based on JSF
        "algorithm": "ES256",
        "publicKey": {               ◄——— Merchant signature key
            "kty": "EC",
            "crv": "P-256",
            "x": "8VY09NWUy-aVGNHZZQDIyy-H3RxLfXbiPR2SVlEubjE",
            "y": "OuHehTNjMbphW0s3nBBVdAALLdzE9x-hup4CnJ1gM-o"
        },
        "value": "rrqbEkm7ZM6uGjnIWg-3c2YHPXsDhzVz....FsMSNotc7QvAsvn2sTFJ-GGdN5Fx6EfQ"
    }
}
```

By including the 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 consists of an embedded **AuthorizationResponse** and an *amount*, signed by the Merchant. Note that the Merchant must send the refund request to *its own bank*. The Merchant's Bank is supposed to respond with (a here not shown) **RefundResponse** object.

Saturn

```json
{
  "@context": "https://webpki.github.io/saturn/v3",
  "@qualifier": "PaymentClientRequest",
  "supportedPaymentMethods": [

      Parameters removed for brevity…

  ],
  "paymentRequest": {
    "commonName": "Planet Gas",
    "amount": "200.00",
    "currency": "EUR",
    "nonDirectPayment": {          ←——— Additional object
      "type": "RESERVATION",
      "subType": "GAS_STATION",
      "fixed": true,
      "expires": "2020-09-25T06:22:00Z"
    },
    "referenceId": "#1000017",
    "timeStamp": "2020-09-24T06:22:59Z",
    "expires": "2020-09-24T06:53:00Z"
  }
}
```

User Interface Implications
(non-normative sample using rotating text)

Payee	Planet Gas
Total	€ 200 *Reserved*, actual payment w

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

Virtual Card Properties

Signature Key

Encryption *Public* Key

Balance Key
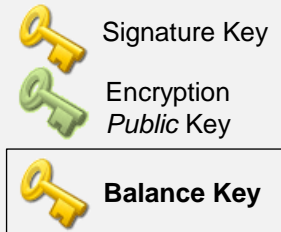
- Payment Method URL
- **Bank Authority URL**
- Account ID
- Card Logotype
- …

Retrieve the ProviderAuthority extensions object

```
{
   "@context": "https://webpki.github.io/saturn/v3",
   "@qualifier": "ProviderAuthority",

            ....

   "extensions": {
      "https://webpki.github.io/saturn/v3/extensions#balance": "https://payments.mybank.com/balance"
   }

                 ....

}
```

The real time account balance option presumes that the virtual card also is equipped with a dedicated *balance key*. This key does unlike the *signature key* not require any user authorization because it can only be used to read the balance of the account associated with the virtual card. The process starts by the Wallet using the virtual card's "Bank Authority URL" to retrieve the ProviderAuthority object using an HTTP GET. In that object there should be an extensions object as shown above. After that the Wallet creates a **BalanceRequest** (as shown on the next slide), signs it with its balance key, and POSTs it to the URL provided by the specific balance extension.

Saturn

```
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "BalanceRequest",
    "recipientUrl": "https://payments.mybank.com/balance",
    "credentialId": "54674448",
    "accountId": "FR7630002111110020050017971",
    "currency": "EUR",
    "timeStamp": "2020-09-24T10:43:05+02:00",
    "requestSignature": {
        "algorithm": "ES256",
        "publicKey": {
            "kty": "EC",
            "crv": "P-256",
            "x": "kiTXwSkkNag5RPjFyPgSNmhPl_97qQPCbPQ2GFmMSp4",
            "y": "g8-4ymBfTg8o14EaJluDE8QmRfkrEy3M0VP61-TsoXg"
        },
        "value": "jstrRDK-2n5FfpiAOr896f1TKuc6wTSU....5zHmAJWMkAIsnA0E679es5KgBiRIH0Ha70XejQUw"
    }
}
```

Core data of selected virtual card

Request signature based on JSF

Balance key of selected virtual card

BalanceResponse Object

```
{
    "@context": "https://webpki.github.io/saturn/v3",
    "@qualifier": "BalanceResponse",
    "accountId": " FR7630002111110020050017971",
    "amount": "5543.00",
    "currency": "EUR",
    "timeStamp": "2020-09-24T10:43:07+02:00"
}
```

The resulting response object holds the current balance (available funds) of the specified account.

Q: Doesn't Saturn's Merchant-to-User Bank AuthorizationRequest introduce security risks?
A: Yes, similar risks as on-line bank applications which effectively are open to requests from *anywhere*. Security features include:
- Small and strict message format
- All messages are signed using industry standard cryptographic algorithms
- AuthorizationRequest is signed by the Merchant and vouched for by the Merchant's Bank/Acquirer through the PayeeAuthority object which also enables *verifiable* Merchant account data
- User signs a hash of `paymentRequest` with a key *which only the User Bank knows about*
- Only *mutually* signed authorizations are considered valid for processing
- Integral support for RBA (Risk Based Authentication)
- Tokenization of payment authorizations makes attacks on Merchant databases useless

Q: Can you trust the Wallet key storage?
A: Saturn depends on hardware backed keys like the AndroidKeystore.

Q: Doesn't Saturn effectively requires new client-side technology to fly?
A: Yes indeed, exactly like Apple Pay did. W3C's https://www.w3.org/TR/payment-request/ is instrumental.

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: Not really, 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: Only the actual payment systems need payment-system specific security, format, names, conventions, and processing. The ability including payment-system specific data in AuthorizationRequest makes Saturn compatible with just about any payment system.

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 a payment is not a resource but rather an event with transactional behavior. In addition, 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, recurring payments, and automated gas station payments without modifications to the underlying payment system