Secure Payment Confirmation

Ideas to enable predictable execution

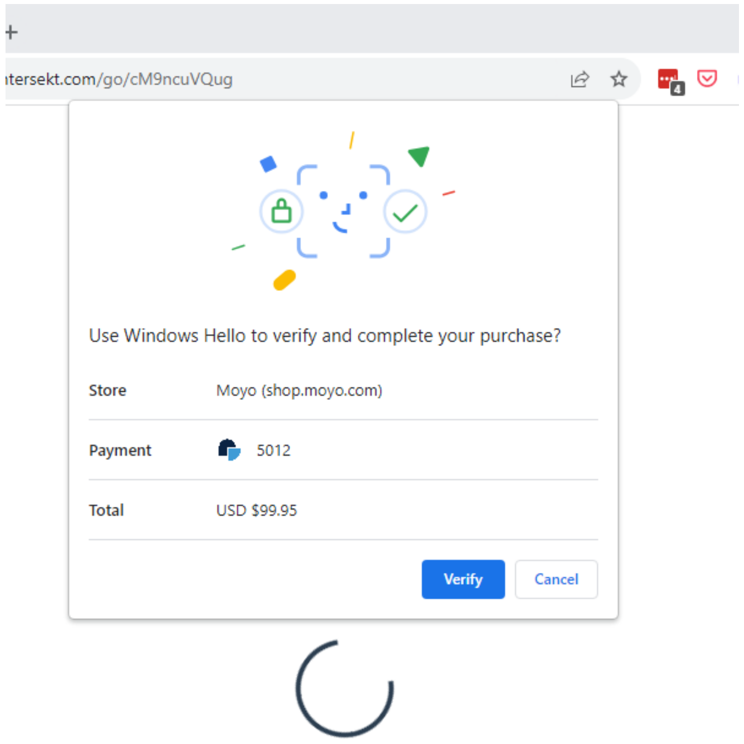
GERHARD OOSTHUIZEN

WITH TONS OF INPUT FROM **IAN JACOBS**

Agenda

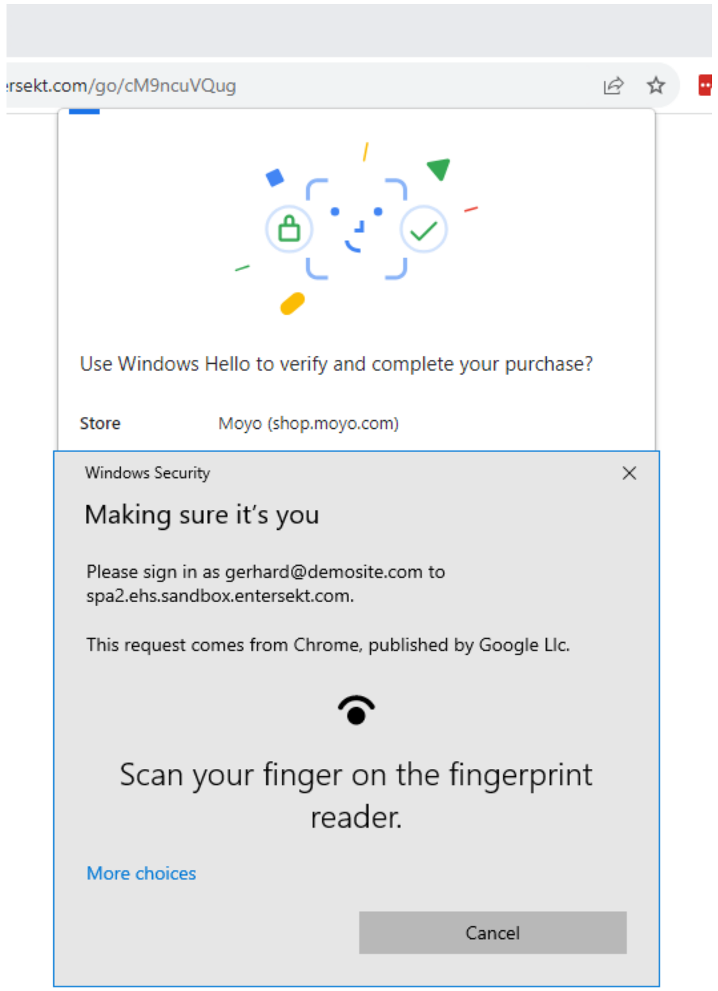
The existing SPC experience

Payment consent screen, followed by WebAuthn SCA, using the OS's Platform Authenticator


ıtersekt.com/go/cM9ncuVQug

Use Windows Hello to verify and complete your purchase?

Store	Moyo (shop.moyo.com)
Payment	5012
Total	USD $99.95


ɪrsekt.com/go/cM9ncuVQug

Use Windows Hello to verify and complete your purchase?

Store Moyo (shop.moyo.com)

Windows Security

Making sure it's you

Please sign in as gerhard@demosite.com to spa2.ehs.sandbox.entersekt.com.

This request comes from Chrome, published by Google Llc.

Scan your finger on the fingerprint reader.

Example on Windows PC with Chrome browser

SPC’s unique benefits

Payment display	Cryptographic proof	Cross-domain control
Payment information shown to user natively in browser	Transaction details signed into cryptogram on the device using hardware backed storage	Merchant or PSP can use Issuer’s credentials for on the merchant page (no re-direct required)

WYSIWYS [What you see is what you sign]

entersekt

What are the pain-points we would want to address

We feel that these are hampering adoption…

A. Predictable user experience (one that that always works)

- Merchant needs to know absolutely what will happen on screen
  (or have a silent way of knowing that it won't before trying).
- Today the SPC screen will not be shown if fido credential is not present – there is a 'error' screen.

B. Know what happened / outcome

- Errors such as cancel, timeout and no-fido-present all presented as the same error.
- The result may be that after a cancel, a user is again asked to approve with a high-friction flow.

C. Support the industry's drive to increase frictionless auth

- Card Associations are mandating ever increasing levels of 'frictionless' approvals. Even in Europe.
- Can we support a non-challenge flow in SPC *('confirmation' is not seen as a challenge)*.

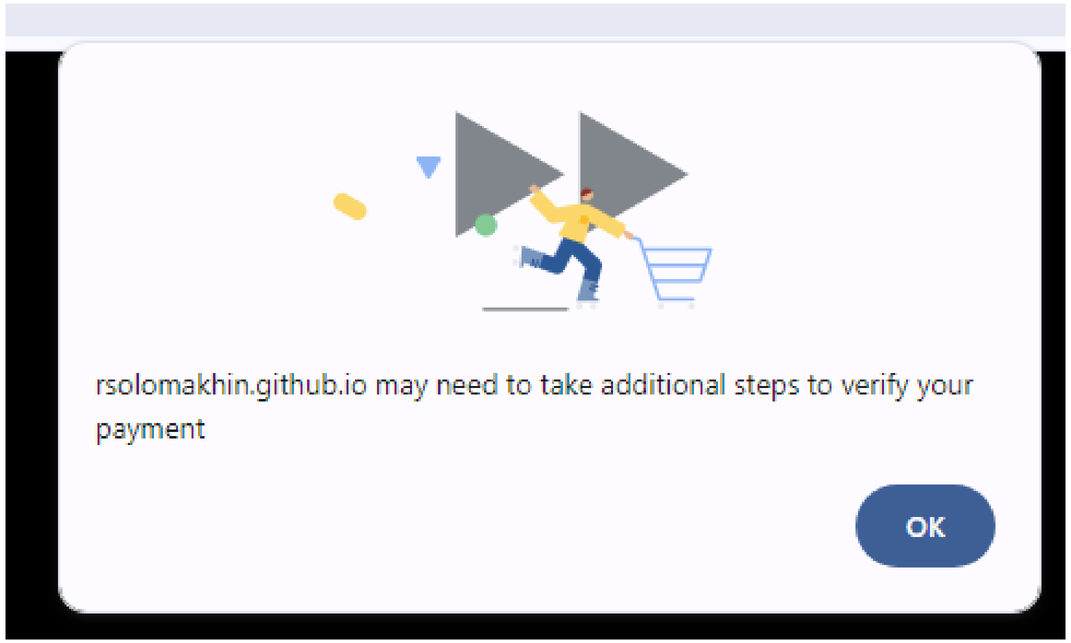
D. Requirement for large scale WebAuthn deployment by Banks/Issuers/Credit Unions

- Issuer must enable WebAuthn and user must have valid Platform Authenticators
- No alternative if this is not in place… which means that most of cardholders cannot leverage SPC

A. Predictable user experience (one that that always works)

There is no silent way to detect if SPC will work without trying it…

If the WebAuthn Credential ID is not on the device, a speed-bump is presented to the customer, breaking the payment flow, and potentially confusing the user.


rsolomakhin.github.io may need to take additional steps to verify your payment

B. Know what happened / outcome

The SPC flow can fail in multiple different ways, due to errors or user-actions

- User cancels during transaction dialog.
- The transaction dialog times out after a period of no action.
- User cancels during the WebAuthn prompt.
- WebAuthn prompt times out
- No suitable WebAuthn credentials are discovered on this device.

All these conditions result in the same generic error (`NotAllowedError`)

- The merchant cannot present an appropriate UX to advise the payor after this error is received.
- *E.g. If a user indicated that they did not want to continue (selected "Cancel" on the transaction dialog), the merchant may think it's because WebAuthn credentials weren't found, and continue to re-challenge the user with an SMS OTP?*

This means that the Merchant is prevented from driving the right user-experience

entersekt

We’ve heard from the industry that SCA causes frustration.

- **Merchants** and **PSP’s** have told us repeatedly that SCA causes abandonment.
- **Card Associations** are mandating an ever-increasing level of frictionless flows.
- **3DS Secure 2** was built to enable allows frictionless flows.
- PSD2 and PSD3 does not require an **SCA** on every transaction.

Payments need a privacy friendly alternative that provides a frictionless flow

- *Changing the 3DS Spec is not an option – rolling out a new version takes 2 to 4 years…*
- *SPC can give a clear signal of a returning customer that’s willing to make this payment*

The SPC dialog is not a challenge – it’s just a confirmation

D. Requirement for large scale WebAuthn deployment by Banks/Issuers/Credit Unions

To use SPC, a bank must have adopted WebAuthn and the user must have registered for it…

To get the benefit of SPC, a cardholder and bank must have a very specific setup

- WebAuthn must be implemented/supported by the Bank (Issuer) for their cardholders
- The Cardholder must have registered for WebAuthn at the Bank (Issuer).
- The Cardholder must have registered a Platform Authenticator and not a roaming authenticator.
- The Cardholder must have registered it on the Chrome Browser that's not on iOS.
- The Cardholder must be performing the payment on that same Chrome Browser.

And even if WebAuthn credentials are provided in the SPC call they still may not work…

Agenda

A collection of possible enhancements

We don't have to approve all of them. Each can stand on their own. Some of them needs to be ideated on…

A. Always show the SPC Payment page, and make WebAuthn conditional

B. Provide a clear error to what happened on SPC page

C. Generate a browser possession signal

D. Require a transaction specific approval to allow signal generation

E. Don't allow Javascript/HTML to access/edit the signature

F. Allow a PaymentRequest caller to decide if WebAuthn is required

G. Enable users to opt out of sharing a returning browser signal

H. Allow registration to be silent

entersekt

These enhancements all drive to resolve these areas of concern…

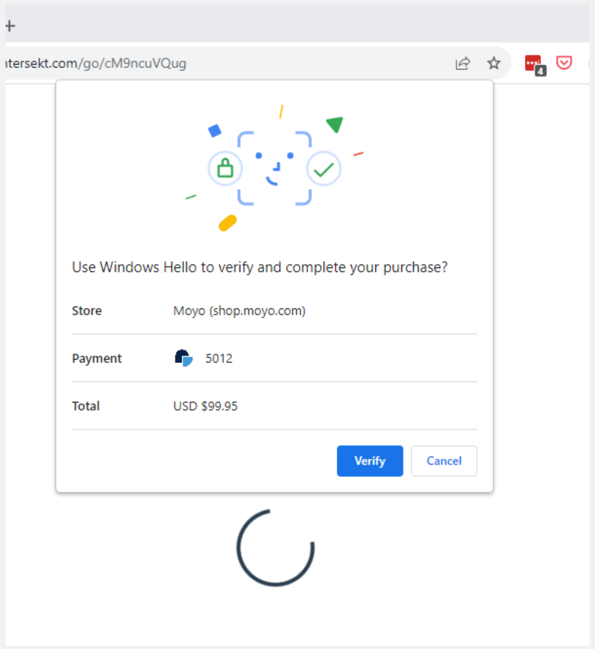
	UX	Outcome	Frictionless	Opt. SCA
A. Always show the SPC Payment page, and make WebAuthn conditional	●		●	●
B. Provide a clear error to what happened on SPC page	●	●		
C. Generate a browser possession signal			●	●
D. Require a transaction specific approval to allow signal generation	●	●	●	
E. Don’t allow Javascript/HTML to access/edit the signature	Security / MITM			
F. Allow a PaymentRequest caller to decide if WebAuthn is required	●	●		●
G. Enable users to opt out of sharing a returning browser signal	Privacy protecting			
H. Allow registration to be silent	●	●		●

entersekt

A. Always show the SPC Payment page, and make WebAuthn conditional

Always show the SPC transaction dialog

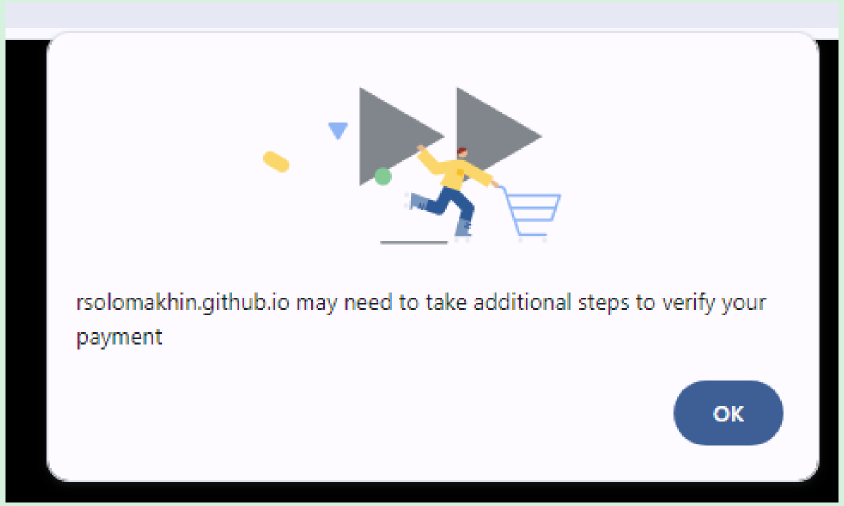
Strong Auth (WebAuthn) becomes a conditional step. If allowed the Payment screen will still be shown even when a provided credential is not available.


ntersekt.com/go/cM9ncuVQug

Use Windows Hello to verify and complete your purchase?

Store	Moyo (shop.moyo.com)
Payment	5012
Total	USD $99.95

1 Users will never experience the 'we tried and failed' flow?


rsolomakhin.github.io may need to take additional steps to verify your payment

2 Merchant and Issuer gets proof of user-consent

- The user wants to continue with this transaction
- The transaction details that was presented is what's been submitted
  (we'll get to the signature soon…)

B. Provide a clear error to what happened on SPC page

Today we do not know what happened that caused SPC to fail (all errors case a NotAllowedError)

- The user may choose to not complete the transaction by cancelling, or the SPC page may timeout.
- This knowledge is important to drive accurate UX. We could **provide more accurate error messages**
  - **For timeout:** `TimeoutError`
  - **For cancellation:** `AbortedError`
- Proposed change to SPC
  - **If** the SPC page is shown, and **the user approves the payment**, then **the Browser Agent should always produce a valid response**.
  - **If WebAuthn** was attempted **and did not conclude successfully** for whatever reason, **the WebAuthn credential will just not be generated**.

C. Generate a browser possession signal

We've had many requests for a 'silent' signal to indicate a returning user (similar to a fingerprint).

- **Use the SPC dialog's payment consent to issue a transaction specific signal (requiring user-action)**

This provides

- UX consistency when seeking user consent.
- Can still create cryptographic evidence of user consent.
- It does not require a Webauthn credential to have been passed down from an issuer and available on the device
- Does not damage privacy, because information is only shared with user consent
- Can provide a valuable device binding signal which value increases with time.
- Can open up SPC to a broader range of use cases (see note below on proposal H)

Add a new `browserSignature` field to the payment response signature

- **The recommendation is that this will be a JSON Web Signature of the `clientDataJSON` field.**

entersekt

C. Generate a browser possession signal

Payment success with Fido…

`Credential #1 payment response:`

```
{
    "requestId": "a860b2ae-6845-43ea-8020-cef349cd5997",
    "methodName": "secure-payment-confirmation",
    "details": {
      "id": "hdyj3v707XhNyoiEwp7Ef6TpHyvqaG767YMK6ZVPJ3w",
      "rawId": "hdyj3v707XhNyoiEwp7Ef6TpHyvqaG767YMK6ZVPJ3w=",
      "response": {
        "authenticatorData": "yFnBU/b2Oli2DTHAX1WdPaZktogLFFJVu9ECZWWXxpMFAAAAAQ==",
        "clientDataJSON":
"{\"type\":\"payment.get\",\"challenge\":\"bmV0d29ya19kYXRh\",\"origin\":\"https://rsolomakhin.github.io\",\"crossOrigin\":false,\"payment\":{\"rpId\":\"rsolomakhin.github.io\",\"topOrigin\":\"https://
rsolomakhin.github.io\",\"payeeOrigin\":\"https://rsolomakhin.github.io\",\"total\":{\"value\":\"0.01\",\"currency\":\"USD\"},\"instrument\":{\"icon\":\"https://rsolomakhin.github.io/pr/spc/troy-alt-
logo.png\",\"displayName\":\"Troy Â·Â·Â·Â· 1234\"}}}",
        "signature":
"YgYiFNa+jBzKnV/gjVkJ7YJOwVPkAxwlqZ56xYGZKENNFcqPHOWcLKdjPAd2/iUlG4zJwhxj0Z35ekiQd/udSuybNmpT+mCiFUZ++gqXP5liYlPsXvrnl8MZcm1zgNIBkWvmOJNqYIYAc/UMregcqalVwpXPV5KI2Eu7Md9q1CGdDIaYhFr0bPDjNwkHv6LOxcH/y3kd
QVaXWGojRbsHa2IN6HHCcmemlVh182QY0VvMSZRB52/8P5ieK3JU0AcnP5ED3T0doZitg4T1Q1ObzIn8rUlULAHB5Ump0cERhxmktPiqsfkkvkEA5NvfgIBdzIeXtD3EZ5TLY8ernfiLdA==",
        “browserSignature": "eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ.dBjftJeZ4CVP-
B92K27uhbUJU1p1r_wW1gFWFOEjXk ",
        "userHandle": "ODg2NTIxNzgxLjUwODU0MTU="
      },
      "type": "public-key"
    }
  }
```

`clientDataJSON` signed in a JSON Web Signature

https://rsolomakhin.github.io/pr/spc-opt-out/

C. Generate a browser possession signal

Payment success without Fido…

Credential #1 payment response:

```
{
    "requestId": "a860b2ae-6845-43ea-8020-cef349cd5997",
    "methodName": "secure-payment-confirmation",
    "details": {
      "id": "hdyj3v707XhNyoiEwp7Ef6TpHyvqaG767YMK6ZVPJ3w",
      "rawId": "hdyj3v707XhNyoiEwp7Ef6TpHyvqaG767YMK6ZVPJ3w=",
      "response": {
        "authenticatorData": "yFnBU/b2Oli2DTHAX1WdPaZktogLFFJVu9ECZWWXxpMFAAAAAQ==",
        "clientDataJSON":
"{\"type\":\"payment.get\",\"challenge\":\"bmV0d29ya19kYXRh\",\"origin\":\"https://rsolomak           crossOrigin\":false,\"payment\":{\"rpId\":\"rsolomakhin.github.io\",\"topOrigin\":\"https://
rsolomakhin.github.io\",\"payeeOrigin\":\"https://rsolomakhin.github.io\",\"total\"           0.01\",\"currency\":\"USD\"},\"instrument\":{\"icon\":\"https://rsolomakhin.github.io/pr/spc/troy-alt-
logo.png\",\"displayName\":\"Troy Â·Â·Â·Â· 1234\"}}}",

        “browserSignature": "eyJ0eXAiOiJKV1QiLA0KICJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJqb2UiLA0KICJleHAiOjEzMDA4MTkzODAsDQogImh0dHA6Ly9leGFtcGxlLmNvbS9pc19yb290Ijp0cnVlfQ.dBjftJeZ4CVP-
B92K27uhbUJU1p1r_wW1gFWFOEjXk ",
        "userHandle": "ODg2NTIxNzgxLjUwODU0MTU="
      },
      "type": "public-key"
    }
  }
```

signature not generated if WebAuthn not completed

clientDataJSON signed in a JSON Web Signature

entersekt

https://rsolomakhin.github.io/pr/spc-opt-out/

C. Generate a browser possession signal

How does the `browserSignature` keypair work?

- The Browser generates a Public/private keypair that is unique across
  - Merchant / top level domain (`payee origin`).
  - Issuer / Relying party (`<rpID>`)

  This aligns to the same privacy principle for cookies in third-party iframes.

- **WYSIWYS**: The signature signs the existing display content that is provided in SPC.
- Signature generation always **requires a positive user gesture**.
- This keypair **may be cleared by the user** when clearing browser data for that domain, or whenever the browser needs storage (similar to Storage Access/Third Party Cookies)
- The `payee origin` (merchant) passes this `browserSignature` on to the `relying party` (issuer)
  - This may be passed up in the 3D Secure AReq of flow and serve as proof that a specific browser was present

How is the `browserSignature` keypair issued/generated?

Four possible options to consider:

1. Provide a dedicated registration/issuing journey
   - This would then occur after the `relying party` (Issuer) has validated the user is present.
   - It could be combined with WebAuthn creation or a special SPC/Payment Hander activity, or leverage some dedicated display such as StorageAccess
   - A dedicated customer flow adds extra complexity for the merchant and if done incorrectly could lead to abandonment.
2. Generate a keypair automatically the if the SPC is used, and no keypair has been issued yet.
   - Trust is then built up over time as the keypair is used legitimately. Attacker would have to be present at point of first use and all subsequent actions
3. Generate the keypair as part of a HTTP Response Header by the Issuer

   (similar to Server-side cookies after a successful challenge.)
   - This could be set after successful 3D Secure HTML challenge (third party domain) or when user successfully logs in on their banking website (top-level domain).
4. Use the same key provisioned by DBSC for `browserSignature`.

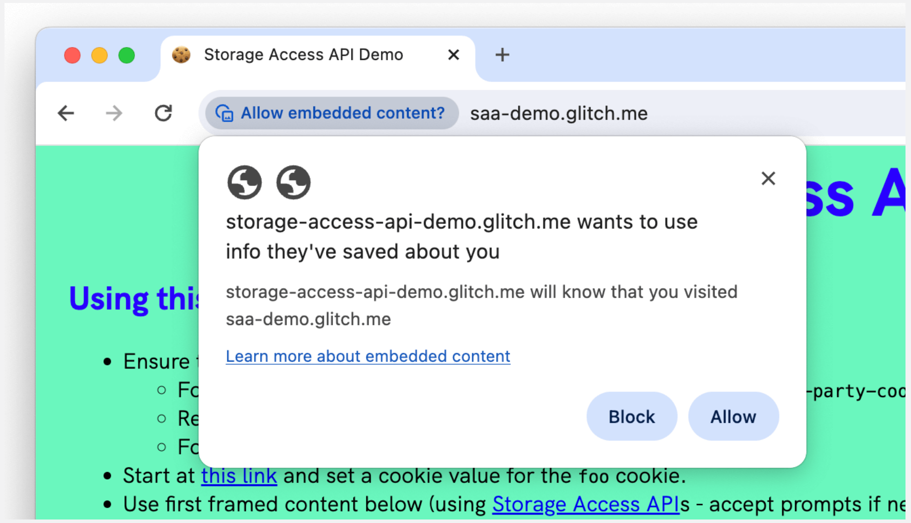
entersekt

D. Require a transaction specific approval to allow signal generation

User consent is required to issue the browser-signal

- Use the SPC payment dialog consent as trigger for this

As comparison, Storage Access enables secrets to be stored in a third party, only has a very generic message, and grants access for a period.


Storage Access API Demo

Allow embedded content? saa-demo.glitch.me

storage-access-api-demo.glitch.me wants to use info they've saved about you

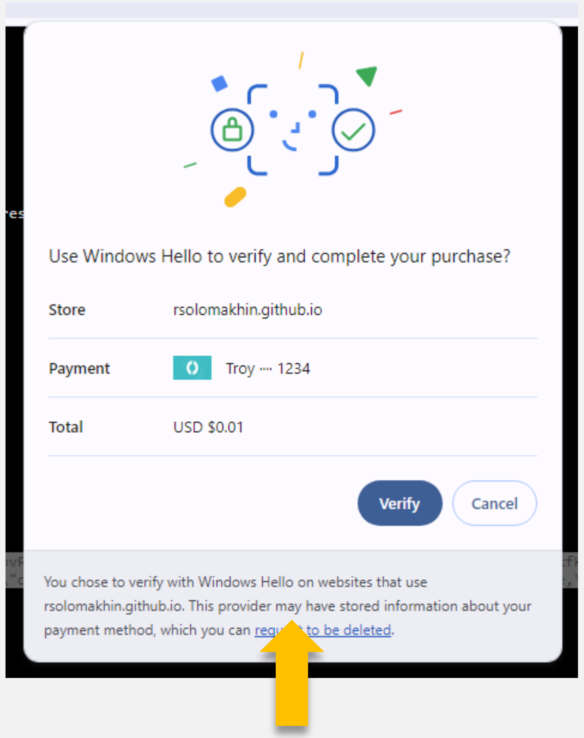
storage-access-api-demo.glitch.me will know that you visited saa-demo.glitch.me

Learn more about embedded content

Block Allow

Using thi

- Ensure
  - Fo
  - Re
  - Fo
- Start at this link and set a cookie value for the foo cookie.
- Use first framed content below (using Storage Access APIs - accept prompts if ne


Use Windows Hello to verify and complete your purchase?

Store	rsolomakhin.github.io
Payment	Troy ···· 1234
Total	USD $0.01

Verify Cancel

You chose to verify with Windows Hello on websites that use rsolomakhin.github.io. This provider may have stored information about your payment method, which you can request to be deleted.

Additionally the SPC dialog can provide specific text can explain the 'signaling' context

E. Don’t allow Javascript/HTML to access/edit the signature

Javascript/HTML can be misused in Man-in-the-browser attacks. We need to mitigate this.

Option 1: Build up trust over time

- First time signature is sent the browser-signature trust will be low (largely untrusted)
- The trust is built up over regular use (so we expect that initial transaction will require a challenge).
- Private key is only available to the User Agent.
- Attacker has to be present at all of the SPC invocations to intercept and adjust.

Other options to consider as we progress, leveraging from Industry patterns

- **Hardware bound keys with Device attestation (WebAuthn)** – sign the browserSignature public key with a hardware key.
- Only send the signature in an **HTTPS header**, and do not allow javascript or HTML to see/edit it (Server side-cookies)
- Issuing the Key as a **request in HTTPS Response** when Issuer has validated the device (DBSC)
- Deliver the signature **via back-channel API** or .wellknown flows (e.g. FedCM)

F. Allow PaymentRequest caller to decide if WebAuthn is required

Put control in the hands of the Merchant, driven by Issuer preference

Indicate the preference for WebAuthn via new parameter in `PaymentRequest`

(Based on level of assurance the issuer will allow)

- Required – fail if not available
- Preferred – show if available; continue if not.
- Discouraged – don't show even if available.

```
strongAuth: preferred | discouraged | required,
    // If Fido not available, still continue
```

OR

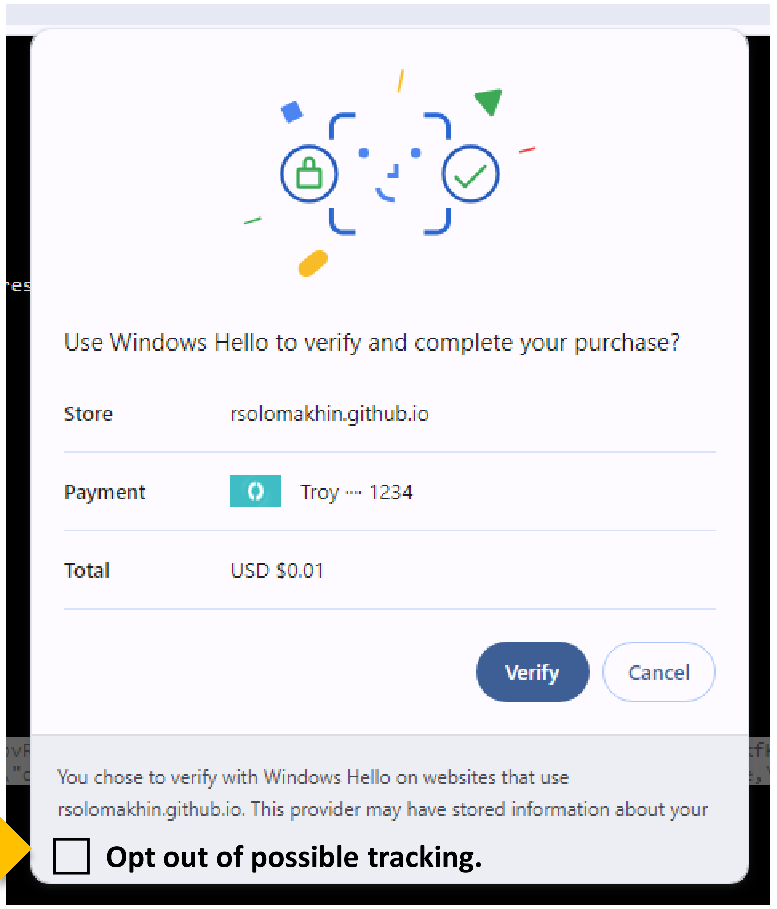
```
browserID: true,
   // Indicate that we request a browser keypair
```

entersekt

G. Enable users to opt out of sharing a returning browser signal

We may need to provide an option for users to opt-out of signal tracking

- This could be presented on the SPC payment dialog
- If User disables it, the `browserSignature` would not be created



Use Windows Hello to verify and complete your purchase?

Store	rsolomakhin.github.io
Payment	Troy ···· 1234
Total	USD $0.01

You chose to verify with Windows Hello on websites that use rsolomakhin.github.io. This provider may have stored information about your

☐ **Opt out of possible tracking.**

Enable users to prevent 'signal' from being created to reduce friction. In this case they will be challenged, but that's their choice.

H. Allow registration to be silent

By enabling SPC to work without WebAuthn, we'll significantly reduce our barrier-to-adoption

- SPC will be available to
  - Issuers that have not implemented WebAuthn for SCA; and
  - Cardholders that have not yet setup WebAuthn

Enabling `browserSignature`, to work without a registration journey, will further reduce this friction.

- No need to issuers or merchants to implement a dedicated registration journey.
- Silent registration/issuing of the `browserSignature` keypair could be achieved in many ways
  - Activation while on the bank site in a first party domain with a javascript call
  - Through an HTTPS header response.
  - Automatically when SPC is used the first time

Once those banks and users eventually adopt WebAuthn, they can seamlessly transition to stronger auth if risk or regulation warrants it, without changing the primary payment consent page.

Addressing the pain-points

Accelerate adoption…

Predictable user experience (one that that always works)

- Consistent experience

✓

Know what happened / outcome

- Clear responses on the SPC page, after user prompt.

✓

Support the industry's drive to increase frictionless auth

- The SPC Page is a confirmation screen; and not authentication.
- Consistently displaying this should not increase/count as friction, since this could be consistently built into the UX.

✓

WebAuthn remains largely beneficial, but is not a prerequisite for adoption

- Initial use-cases could work without being dependent on WebAuthn rollout at Banks.
- A natural upgrade will happen as WebAuthn becomes more widely adopted by Banks.

✓

How will this be utilized in 3D Secure

SPC will immediately become available for all flows outside Europe (possession factor).

Non-WebAuthn SPC will help issuers avoid a full SCA challenge if it's not required.

Issuer: Lower friction challenge-flow (possession only)

- Runs in issuer iFrame during challenge
- Will allow issuer to optimize SMS OTP or OOB App Auth flows that exists outside Europe

Merchant: Implement 3DS specific SPC flow (Two AReq messages)

- The merchant implements the two-step AReq flow for SPC in 3DS 2.3
- Even if user is not setup for WebAuthn at the bank, the signal would allow an issuer to not request a subsequent challenge.

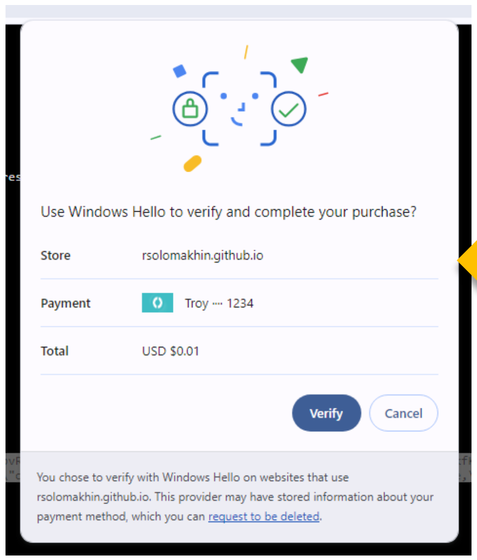
Enable Merchants to submit proof while asking for frictionless flow

- Merchant can submit SPC signal performs in AReq

Simplify various other 3D Secure flows

- E.g. 'Trust the merchant', Delegated auth

EMV 2.3.1: SPC Integration



Use Windows Hello to verify and complete your purchase?

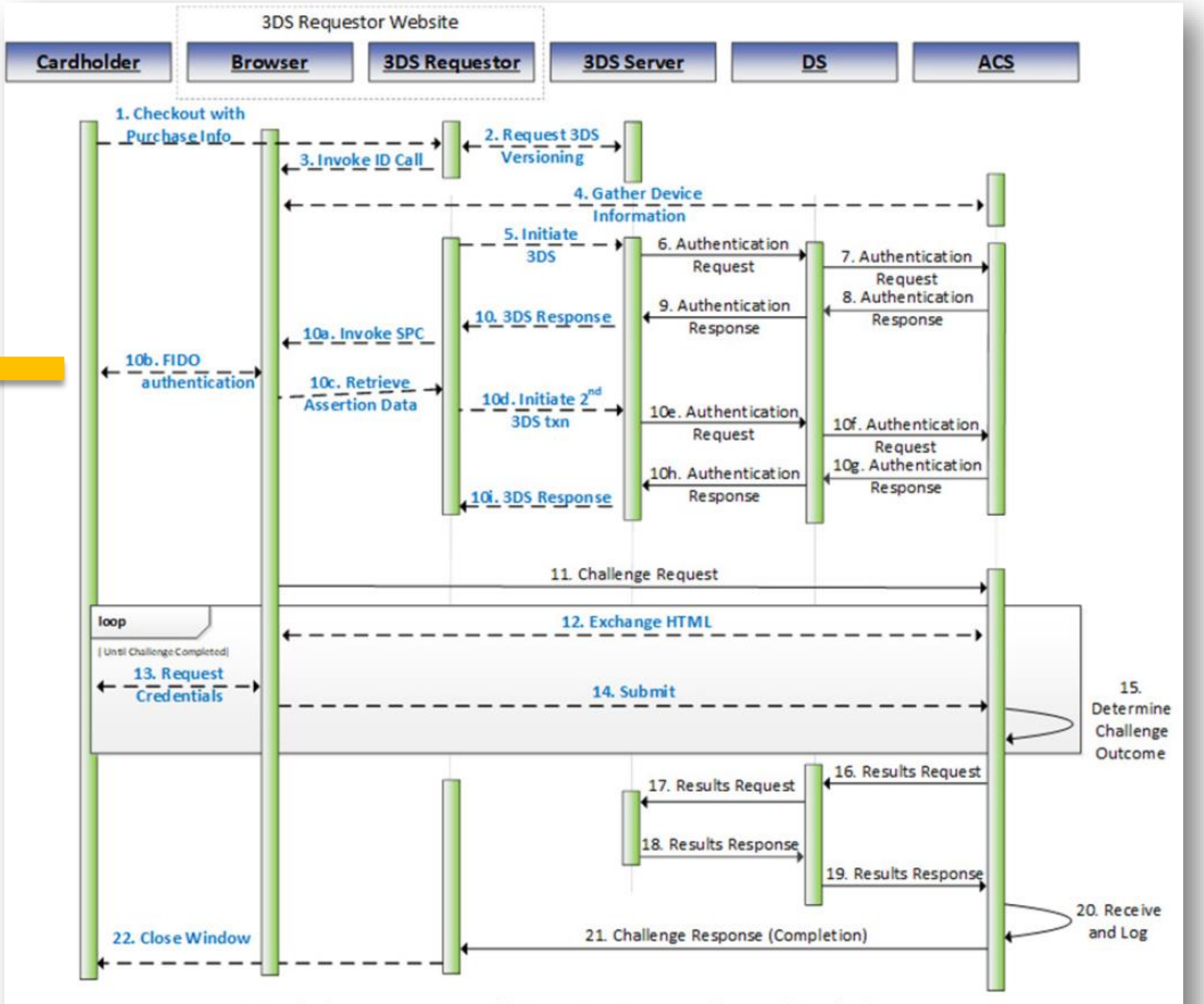
Store	rsolomakhin.github.io
Payment	Troy ···· 1234
Total	USD $0.01

You chose to verify with Windows Hello on websites that use rsolomakhin.github.io. This provider may have stored information about your payment method, which you can request to be deleted.

Bank Indicates willingness to accept the risk signal.

They may include or not include Credential ID’s

Risk signal included in second AReq


3DS Requestor Website
Cardholder
Browser
3DS Requestor
3DS Server
DS
ACS
1. Checkout with Purchase Info
2. Request 3DS Versioning
3. Invoke ID Call
4. Gather Device Information
5. Initiate 3DS
6. Authentication Request
7. Authentication Request
8. Authentication Response
9. Authentication Response
10. 3DS Response
10a. Invoke SPC
10b. FIDO authentication
10c. Retrieve Assertion Data
10d. Initiate 2nd 3DS txn
10e. Authentication Request
10f. Authentication Request
10g. Authentication Response
10h. Authentication Response
10i. 3DS Response
11. Challenge Request
loop
[Until Challenge Completed]
12. Exchange HTML
13. Request Credentials
14. Submit
15. Determine Challenge Outcome
16. Results Request
17. Results Request
18. Results Response
19. Results Response
20. Receive and Log
21. Challenge Response (Completion)
22. Close Window

Note: Dashed arrows are not part of the 3-D Secure 2.0 protocol but are shown for clarity.