# Storing Cryptographic Keys in Persistent Browser Storage

**Revised after Presentation at ICMC 2017**

**Please see the companion blog post at**

**https://pomcor.com/blog/keys-in-browser/**

Francisco Corella

fcorella@pomcor.com

Karen Lewison

kplewison@pomcor.com

# Key storage in web clients

- Use of cryptography in web apps has been hindered by the problem of **where to store cryptographic keys** on the client side
  - Cryptographic keys can be stored today in smart cards and TPMs, and have been stored in the past in Infocards and files accessed by Java applets
  - TLS client certificates can be imported into web browsers
  - But these solutions are **not generally available today to all users of web apps**
- New web technologies may enable generally available key storage solutions
- In this talk we focus as an example on **keys in cryptographic credentials** used for authentication or identification

# New web technologies

- These new web technologies are available to JavaScript (JS) code embedded in web pages through APIs:
  - Web Storage API
    - Provides "HTML5 *localStorage*"
  - IndexedDB API
  - Web Cryptography API
  - Service Worker API
  - Web Authentication API

# Web Authentication API

- Based on FIDO U2F specification, taken over by the W3C
  - Will be available later this year in Chrome, Firefox, Edge
- Allows JS code to store a cryptographic credential in an "authenticator"
  - Cryptographic module in secure storage (e.g. USB dongle, TPM, Secure Element or TEE)
  - Provides a signed attestation of security
- But the cryptographic credential is an uncertified key pair
  - Only usable for two-party authentication
  - No support for credentials issued by a third party
- Very complex

# Web Storage API

- Available in all browsers
- Provides persistent storage for **JS strings** as properties of the *localStorage* object
- **Data protected by the same origin policy of the browser**
- Very simple

# IndexedDB API

- Available in all browsers
- Provides persistent storage of **JS objects** indexed by keys in databases managed by the *indexedDB* object
- **Data protected by the same origin policy of the browser**
- Complex asynchronous interface
  - "IndexedDB API is powerful, but may seem too complicated for simple cases" – MDN

# Web cryptography API

- Available in most browsers
- Provides **RSA and ECDSA** (with NIST curves P-256, P-384 and P-512)
  - Plus ECDH, AES (including AES-GCM), HMAC, SHA (SHA-1, SHA-256, SHA-384 and SHA-512), HKDF, PBKDF2
  - Does not provide DSA
- Key pair generation produces two **CryptoKey objects** and private key can be made **non-extractable** from its CryptoKey object
  - CryptoKey object is not persistent by itself
  - It **cannot** be encoded as a string for storage in ***localStorage***
  - But it **can** be stored in ***indexedDB***

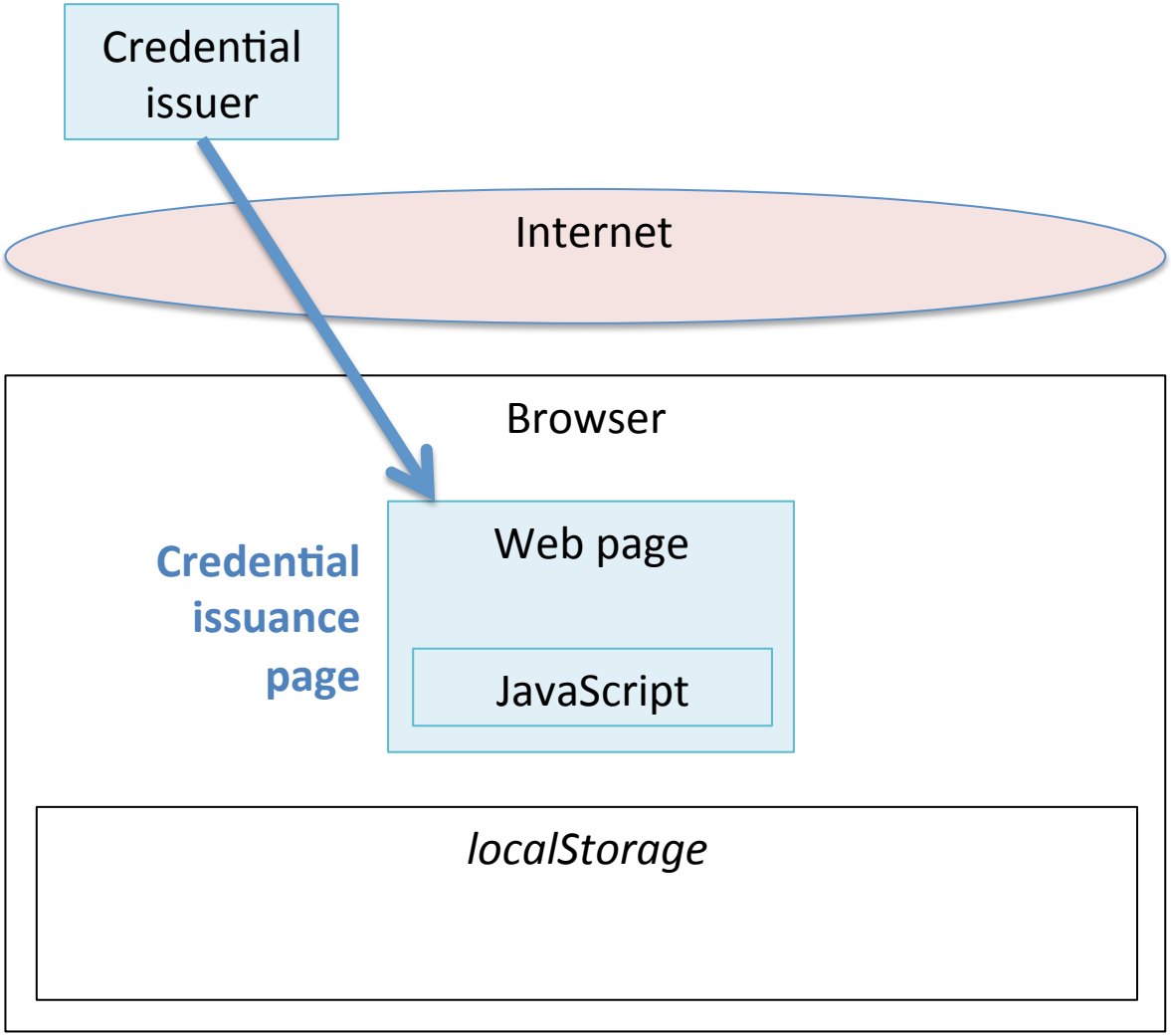
# Service Worker API

- Allows the front-end of a web app to work "offline" like a native app, without accessing the back-end
- Available in Chrome, Firefox and Opera, under development in Edge, under consideration for Safari
- JS front-end registers a service worker with the browser and configures it to **intercept** certain requests to the backend and respond to them by **generating a web page** that is rendered by the browser
- The generated web page may include **JS code**, which can be used to **present a cryptographic credential**

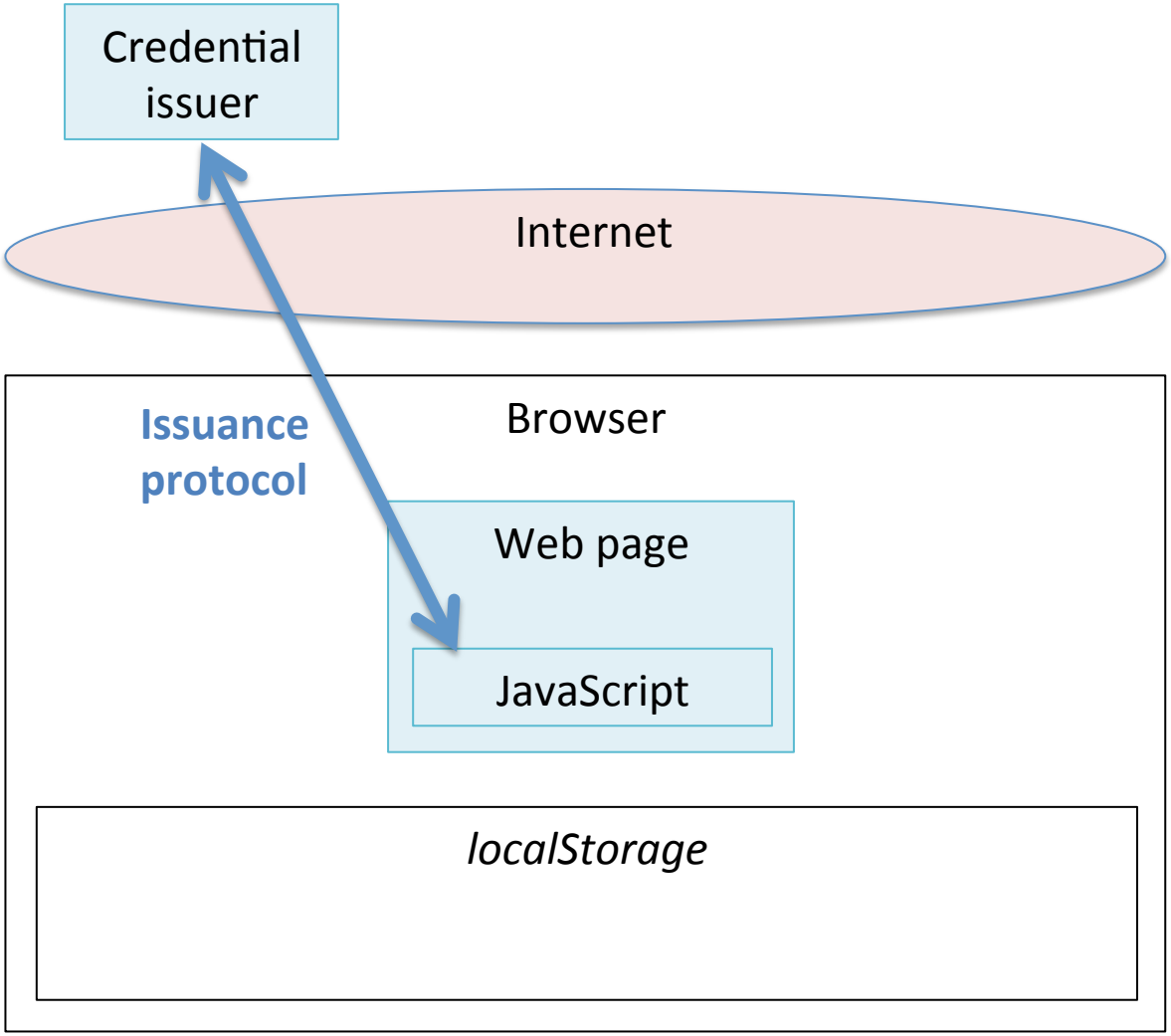
| Four solutions for storing cryptographic credentials in the browser | **Using localStorage**<br><br>Any cryptographic credential (any certified key pair, anonymous credentials, rich credentials, etc.) | **Using the IndexedDB and Web Cryptography APIs**<br><br>Credential must be RSA or ECDSA certified key pair; private key not extractable from CryptoKey object |
|---|---|---|
| **No Trusted Consent Manager** | **Solution 1** | **Solution 2** |
| **With Trusted Consent Manager** | **Solution 3** | **Solution 4** |

Details in slide 19

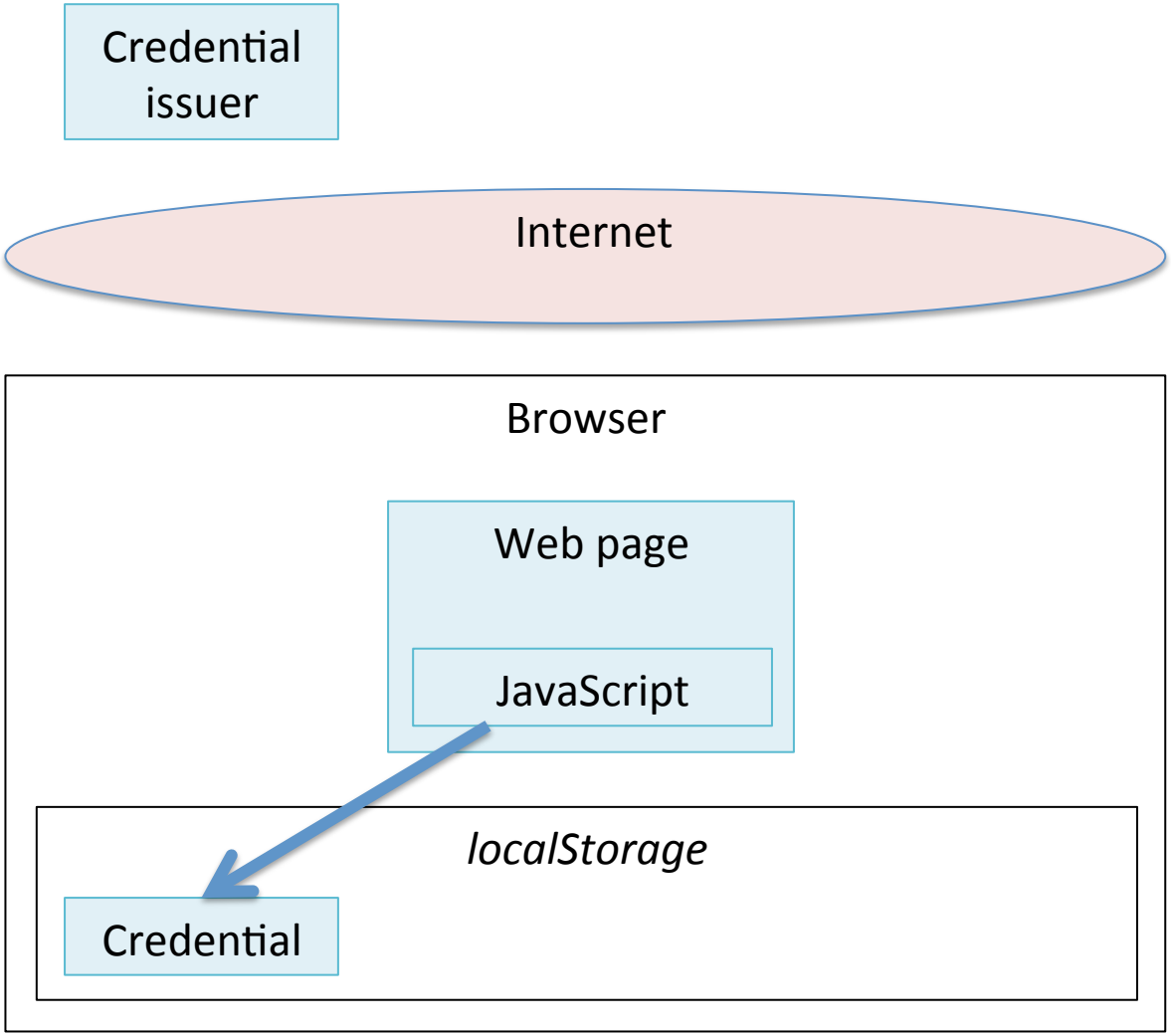
# Solution 1 – Issuance



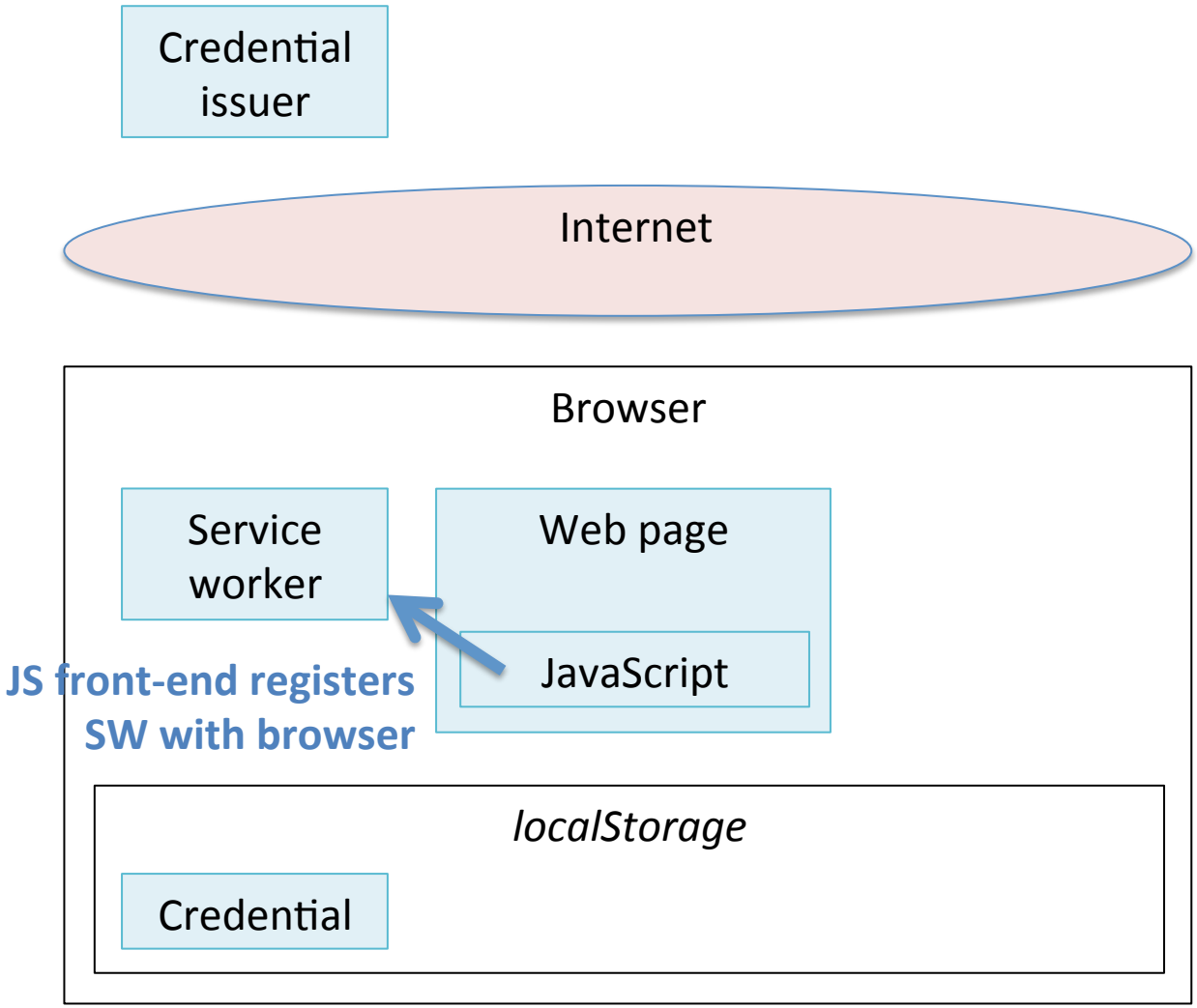
# Solution 1 – Issuance



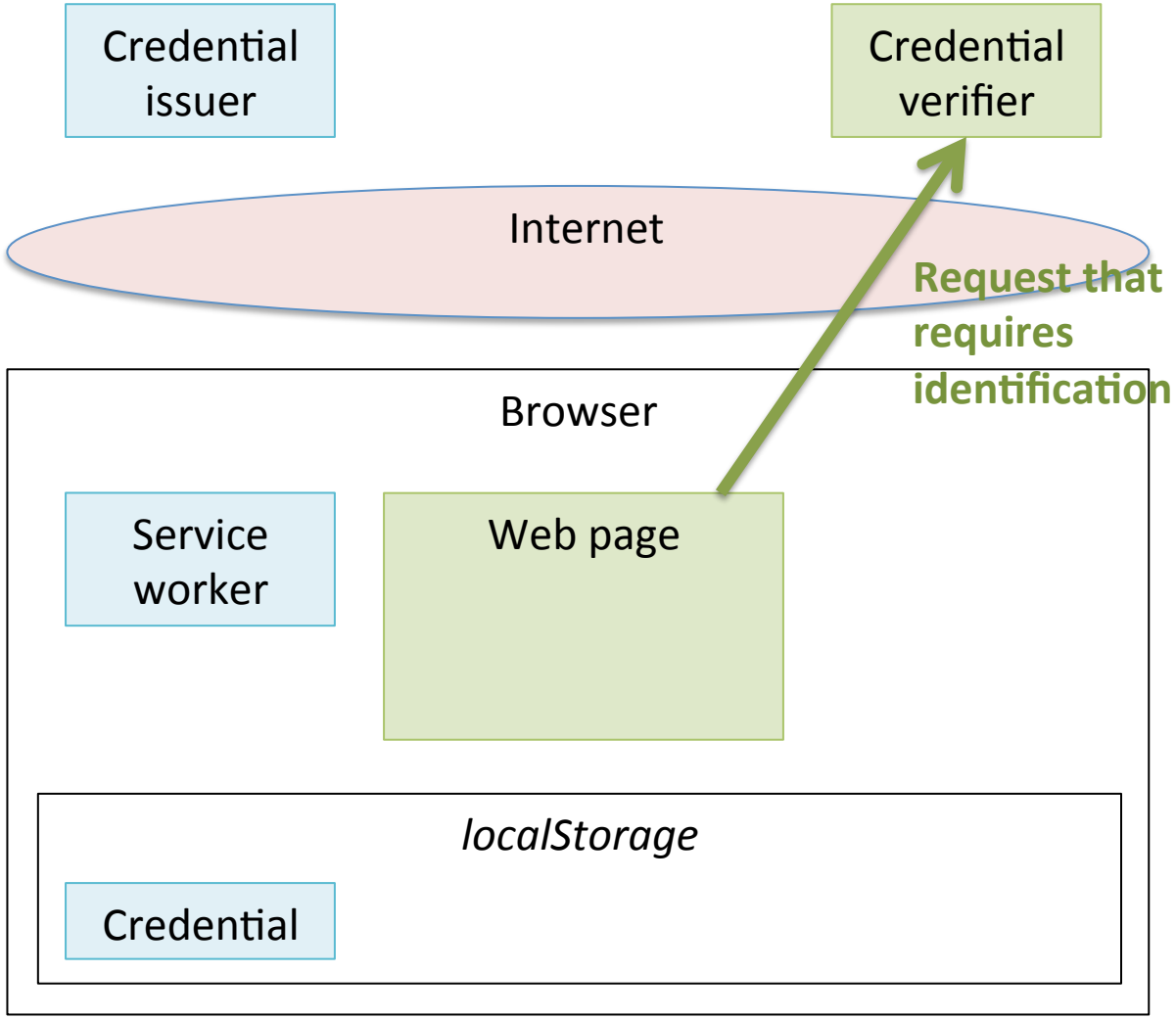
# Solution 1 – Issuance



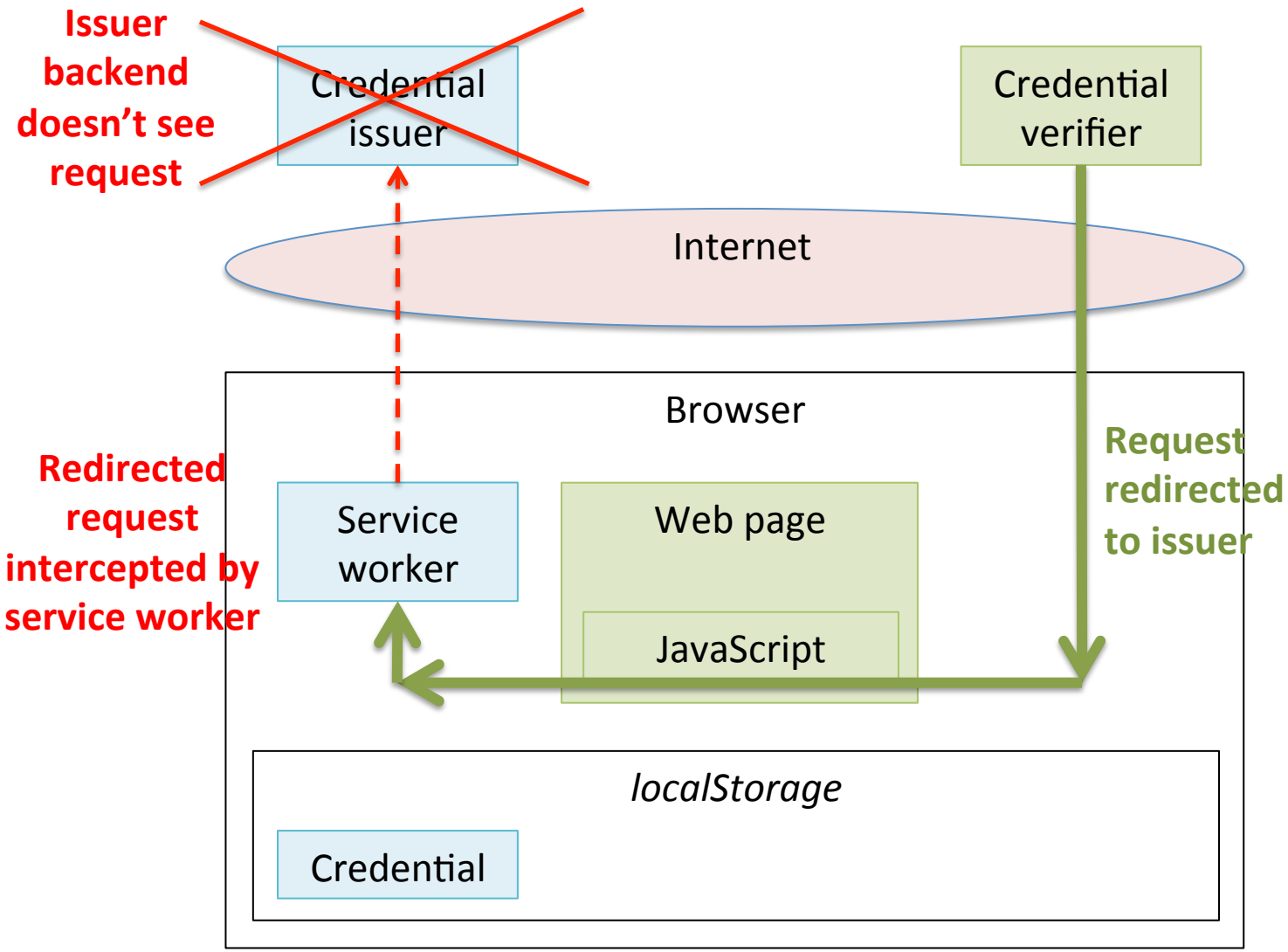
# Solution 1 – Issuance



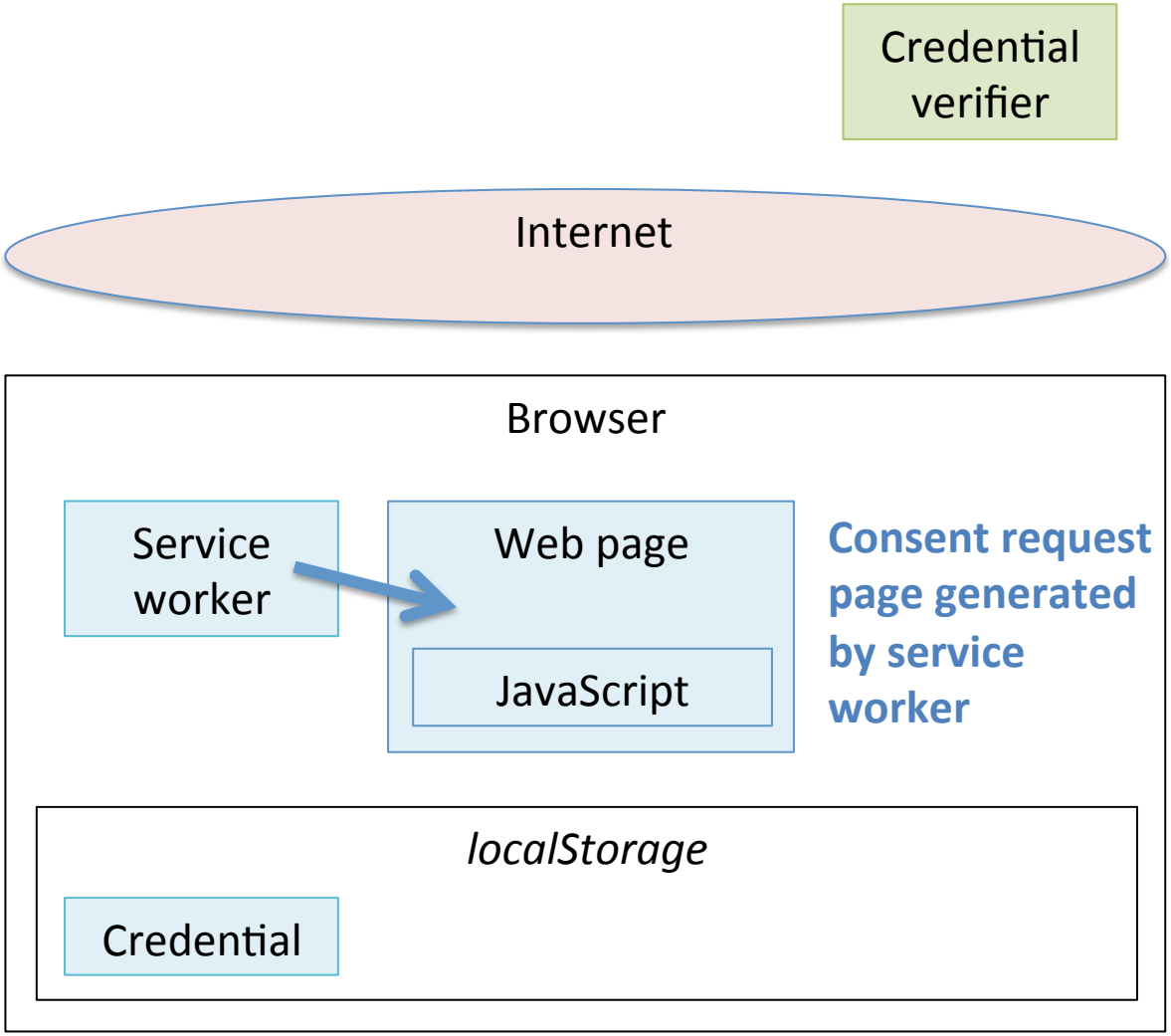
# Solution 1 – Presentation



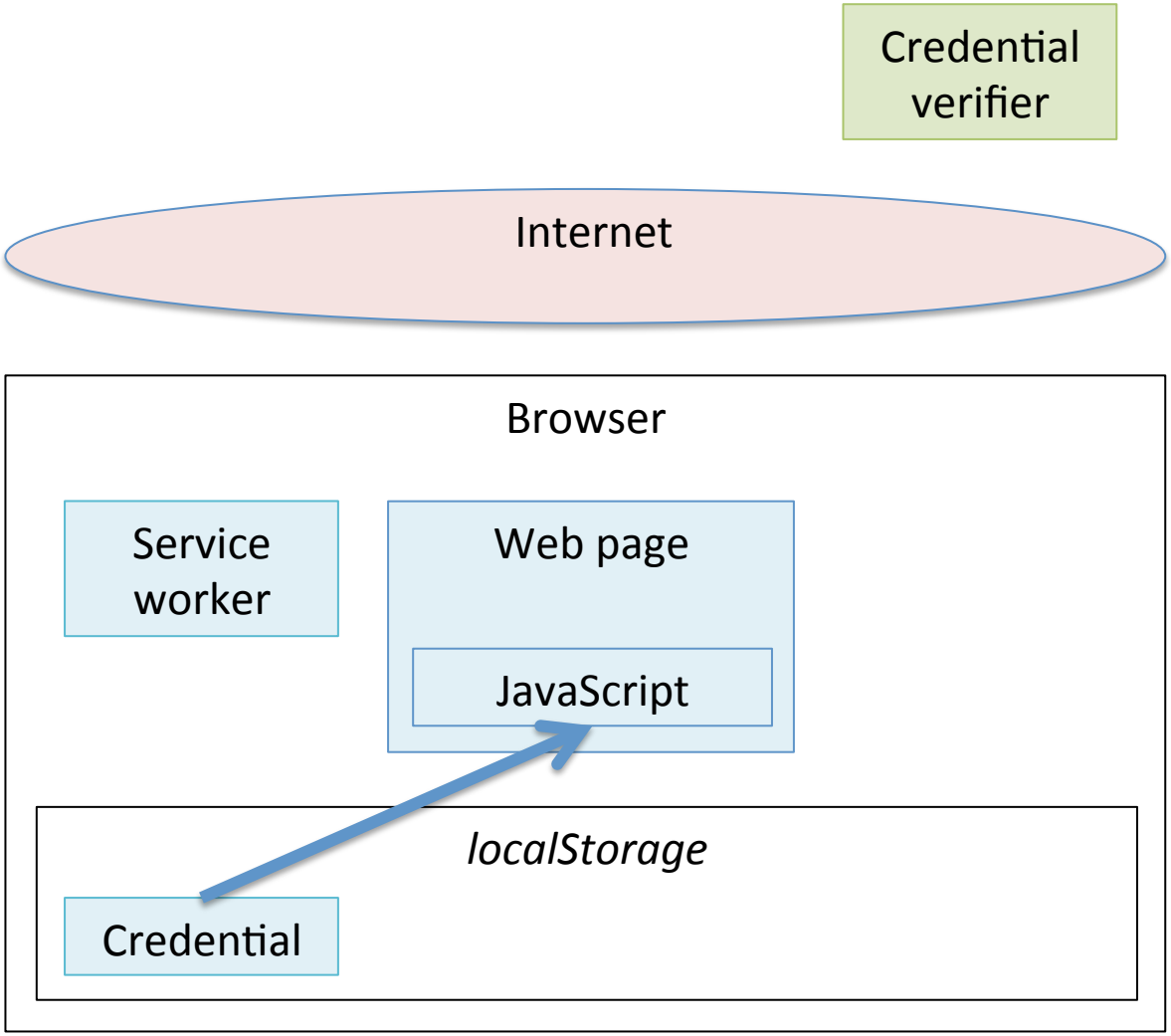
# Solution 1 – Presentation



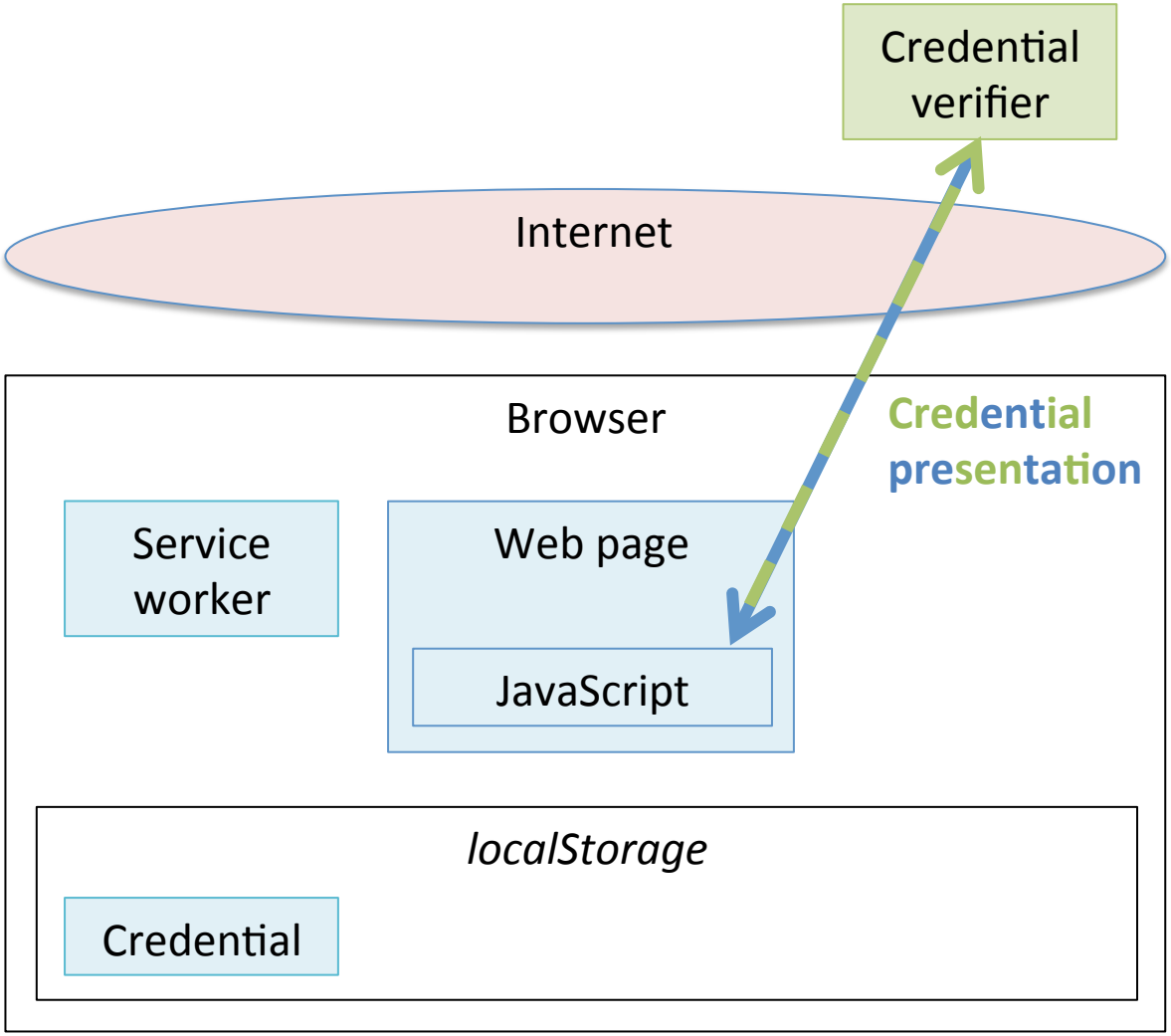
# Solution 1 – Presentation



# Solution 1 – Presentation



# Solution 1 – Presentation



| Four solutions for storing cryptographic credentials in the browser | Using localStorage<br><br>Any cryptographic credential (any certified key pair, rich credentials, anonymous credentials, etc.) | Using the IndexedDB and Web Cryptography APIs<br><br>Credential must be RSA or ECDSA certified key pair; private key not extractable from CryptoKey object |
|---|---|---|
| **No Trusted Consent Manager** | **Solution 1**<br><br>Issuer FE runs issuance protocol with issuer BE<br><br>Issuer SW presents credential | **Solution 2**<br>Issuance protocol: issuer FE generates key pair, issuer BE certifies public key. Issuer SW presents credential but cannot extract private key |
| **With Trusted Consent Manager** | **Solution 3**<br><br>TCM FE runs issuance protocol with issuer BE<br><br>TCM SW presents credential | **Solution 4**<br>Issuance protocol: TCM FE generates key pair, issuer BE certifies public key. TCM SW presents credential but cannot extract private key |

TCM = Trusted Consent Manager, FE = Front-end, BE = Back-end; SW = service worker

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | | | | apture |
| SC/preloaded credential | Capture | | | | ecure |
| SC/on-card key pair gen | Capture | | | | ecure |
| SC/trusted firmware | Secure | | | | ecure |
| TP HW on device | Secure | | | | ecure |
| TEE | Secure | Secure | Secure | Secure | Capture |

**Capture/Use/Secure:**
refers to whether the cryptographic credential can be used by the adversary on the subject's machine, or captured for use elsewhere

JS = Javascript; IDB = *indexedDB*; CK = CryptoKey object; LS = *localStorage*; SC = Smartcard TCM = Trusted consent manager; TP HW = Tamper-proof hardware, e.g. TPM, Secure Element; TEE = Trusted execution environment

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

JS = Javascript; IDB = *indexedDB*; CK = CryptoKey object; LS = *localStorage*; SC = Smartcard
TCM = Trusted consent manager; TP HW = Tamper-proof hardware, e.g. TPM, Secure Element; TEE = Trusted execution environment

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

Storage in browser is secure if: (i) issuer is honest and secure; (ii) no malware on subject's device; and (iii) no physical capture of subject's device

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

Storage in browser not secure against malware or physical capture

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

Storage-in-browser solutions have different security postures w.r.t. attack from issuer after issuance, e.g. attack by an issuer insider after issuance or introduction of an XSS vulnerability

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS from other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

*localStorage* controlled by trusted consent manager is more secure than SC assuming no malware or physical capture, even with on-card key pair generation, if SC is provided by the issuer. Must use trusted firmware on SC to match LS controlled by TCM

| SECURITY POSTURES | Attack by issuer at issuance | Attack from issuer after issuance | Malicious JS frpm other origin | Malware | Physical capture |
|---|---|---|---|---|---|
| Solution 1: LS | Capture | Capture | Secure | Capture | Capture |
| Solution 2: IDB/CK | Capture | Use | Secure | Capture | Capture |
| Solutions 3 and 4: TCM | Secure | Secure | Secure | Capture | Capture |
| SC/preloaded credential | Capture | Secure | Secure | Secure | Secure |
| SC/on-card key pair gen | Capture | Secure | Secure | Secure | Secure |
| SC/trusted firmware | Secure | Secure | Secure | Secure | Secure |
| TP HW on device | Secure | Secure | Secure | Secure | Secure |
| TEE | Secure | Secure | Secure | Secure | Capture |

Tamper-proof hardware on device (TPM, Secure Element) or TEE are good solutions; but web applications cannot use them today

# Potential applications

- Remote identity proofing, recurring authentication and privilege escalation using a cryptographic credential such as:
  - Traditional public key certificate and associated private key
  - Anonymous credential (e.g. Idemix)
  - Rich credential
- End-to-end encryption for web mail
- Cryptographically secured online payments

# Thank you for your attention!

For more information:

pomcor.com

pomcor.com/blog/

**Companion post:** pomcor.com/blog/keys-in-browser/

Francisco Corella

fcorella@pomcor.com

Karen Lewison

kplewison@pomcor.com

# Any questions?