Storing Cryptographic Keys in Persistent Browser Storage

Presentation at ICMC 2017

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
  - Keys have been stored in smart cards, Infocards, TPMs, files accessed by Java applets, etc.
  - But these solutions are **not generally available to all web users**
- 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 credential 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 desktop 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 desktop and mobile browsers
- Provides **RSA and ECDSA** (with NIST curves P-256, P-384 and P-256)
  - 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

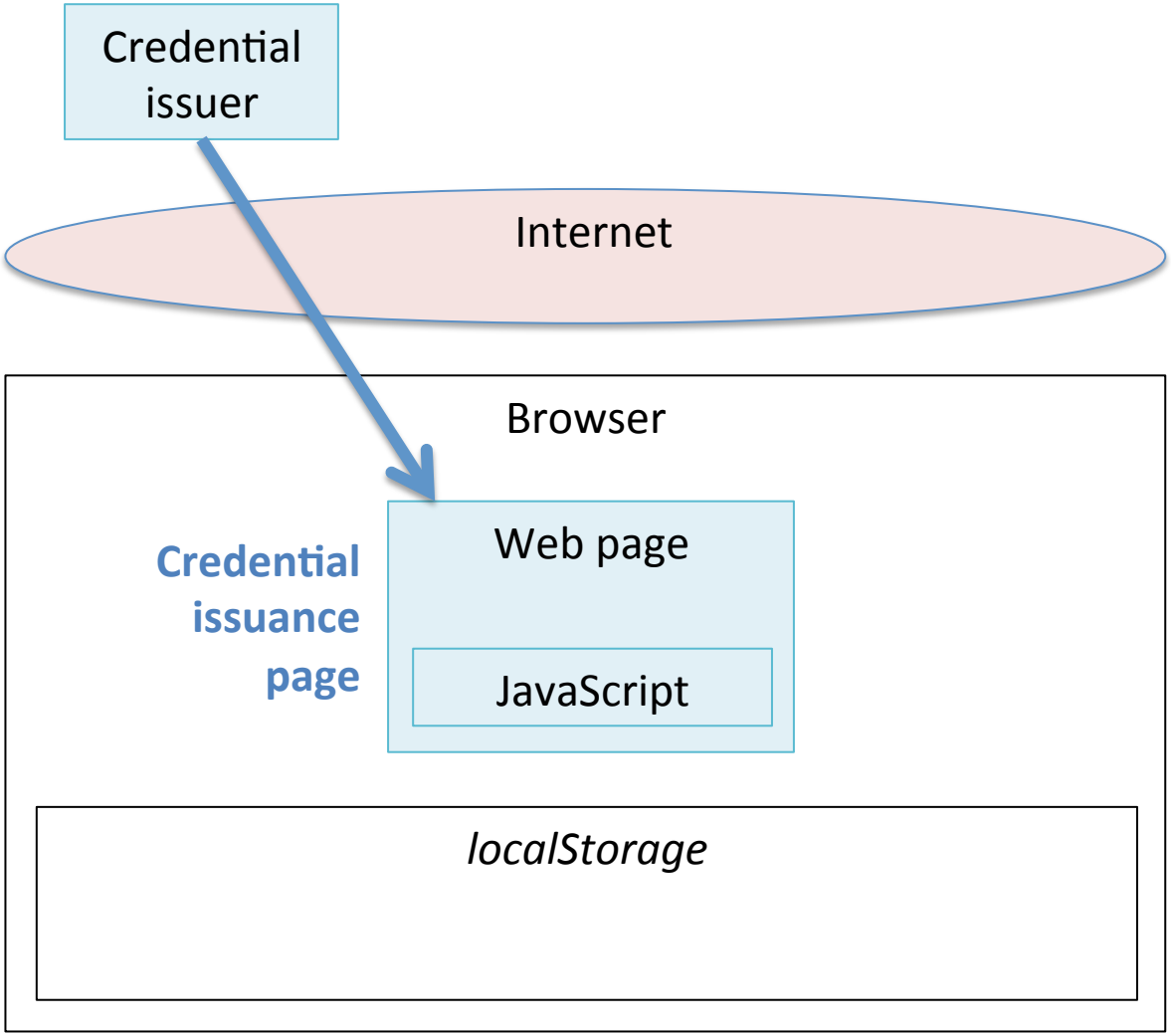
- Allows the front-end of a web app to work "offline" like a native app, without accessing the back-end
- Available in Chrome, Firefox and Opera, under development in Edge, under consideration for Safari
- JS front-end registers a service worker with the browser and configures it to **intercept** certain 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 third party cryptographic credentials in the browser	**Using localStorage** Any cryptographic credential (any certified key pair, anonymous credentials, rich credentials, etc.)	**Using the IndexedDB and Web Cryptography APIs** Credential must be RSA or ECDSA certified key pair; private key not extractable from CryptoKey object
No TTP	**Solution 1**	**Solution 2**
TTP	**Solution 3**	**Solution 4**

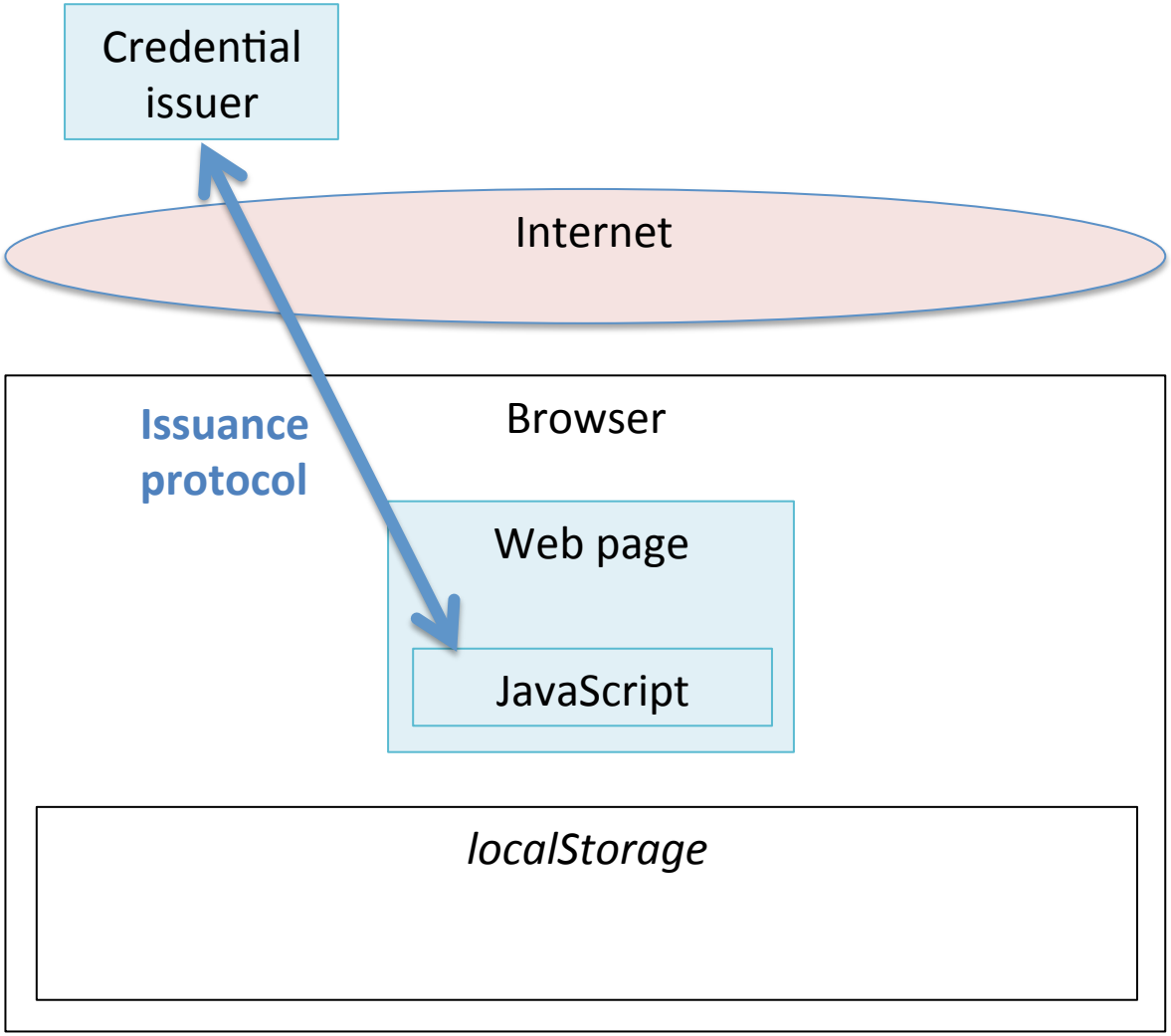
Details in slide 19

TTP = Trusted 3rd party

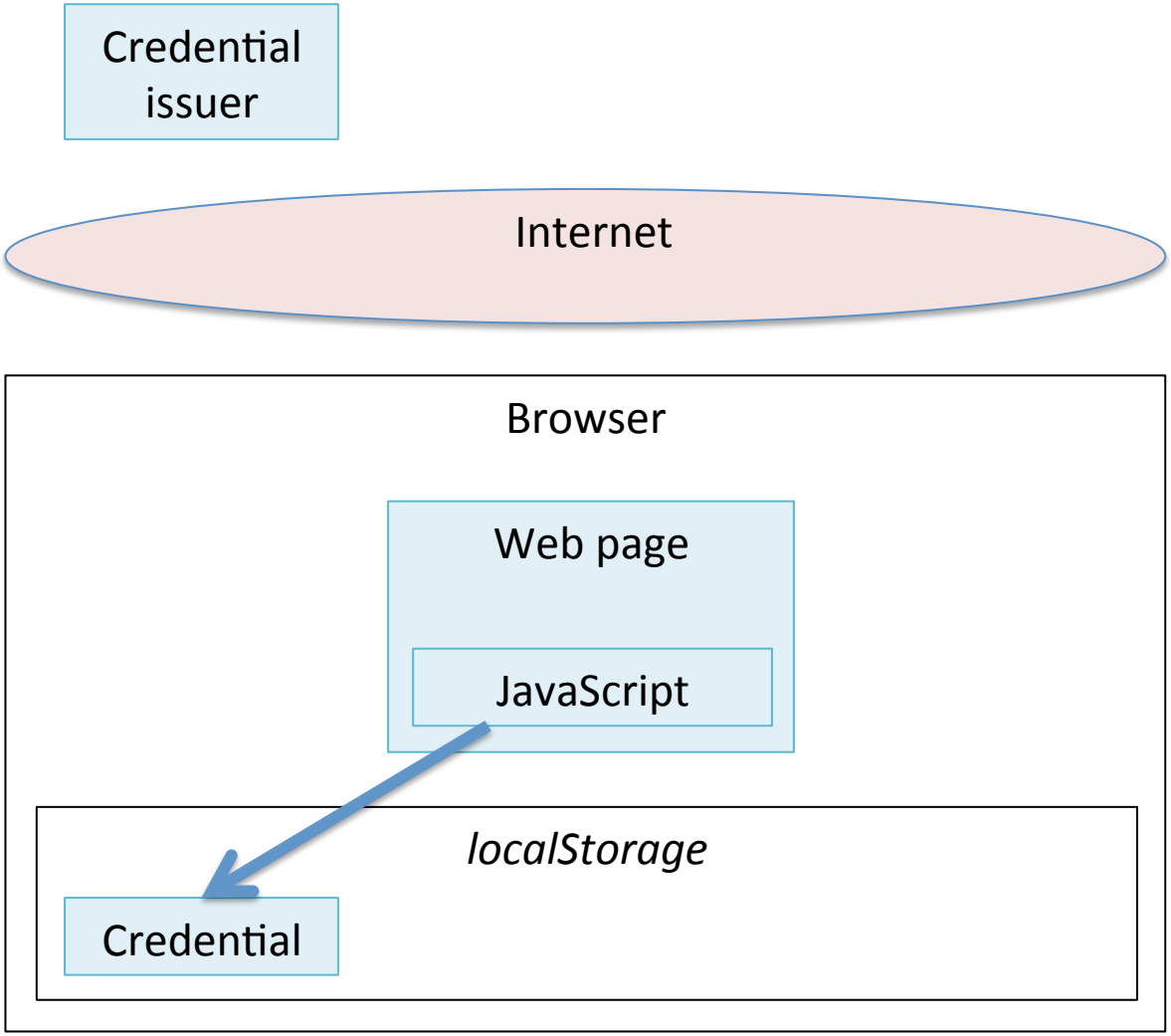
Solution 1 – Issuance



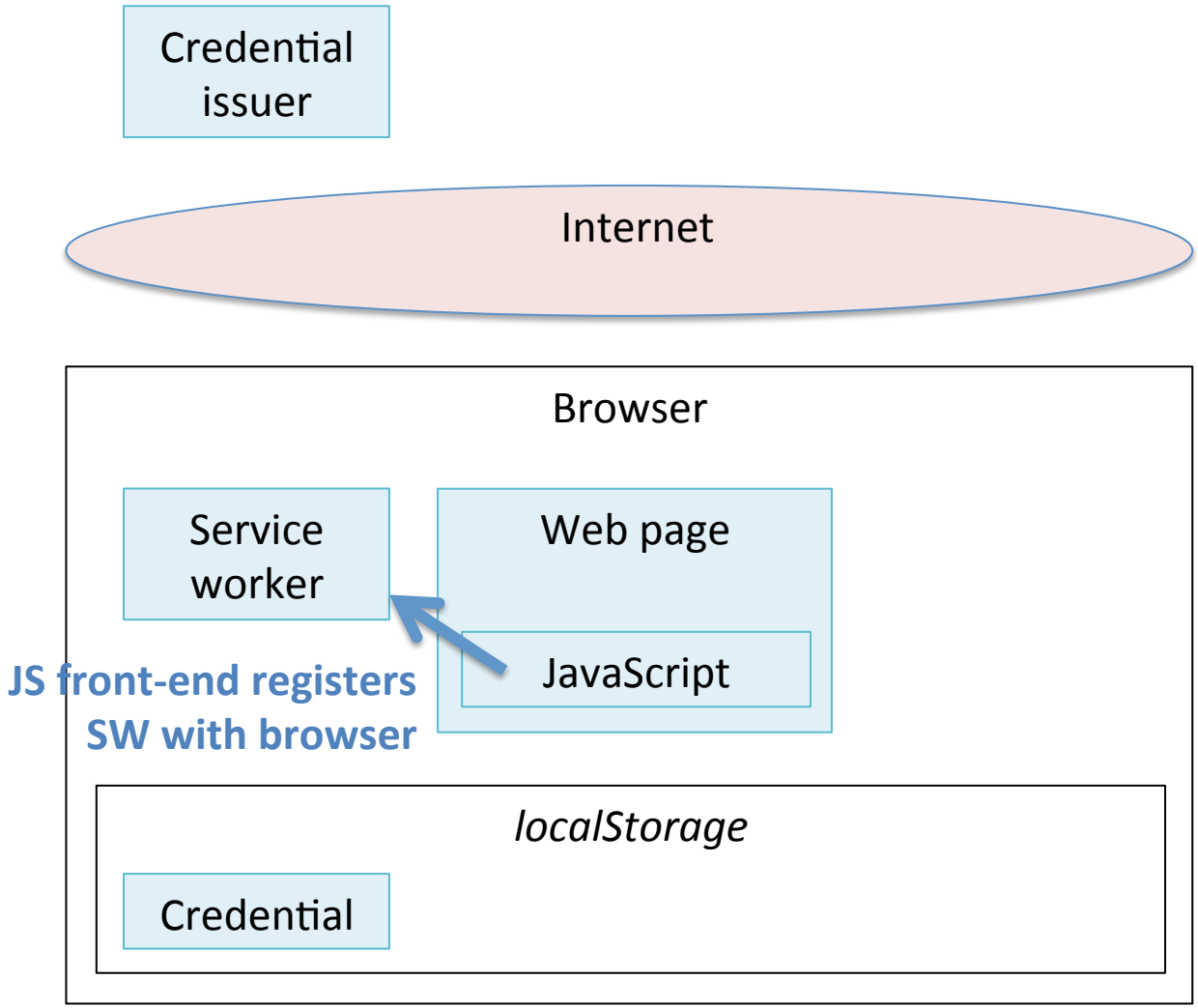
Solution 1 – Issuance



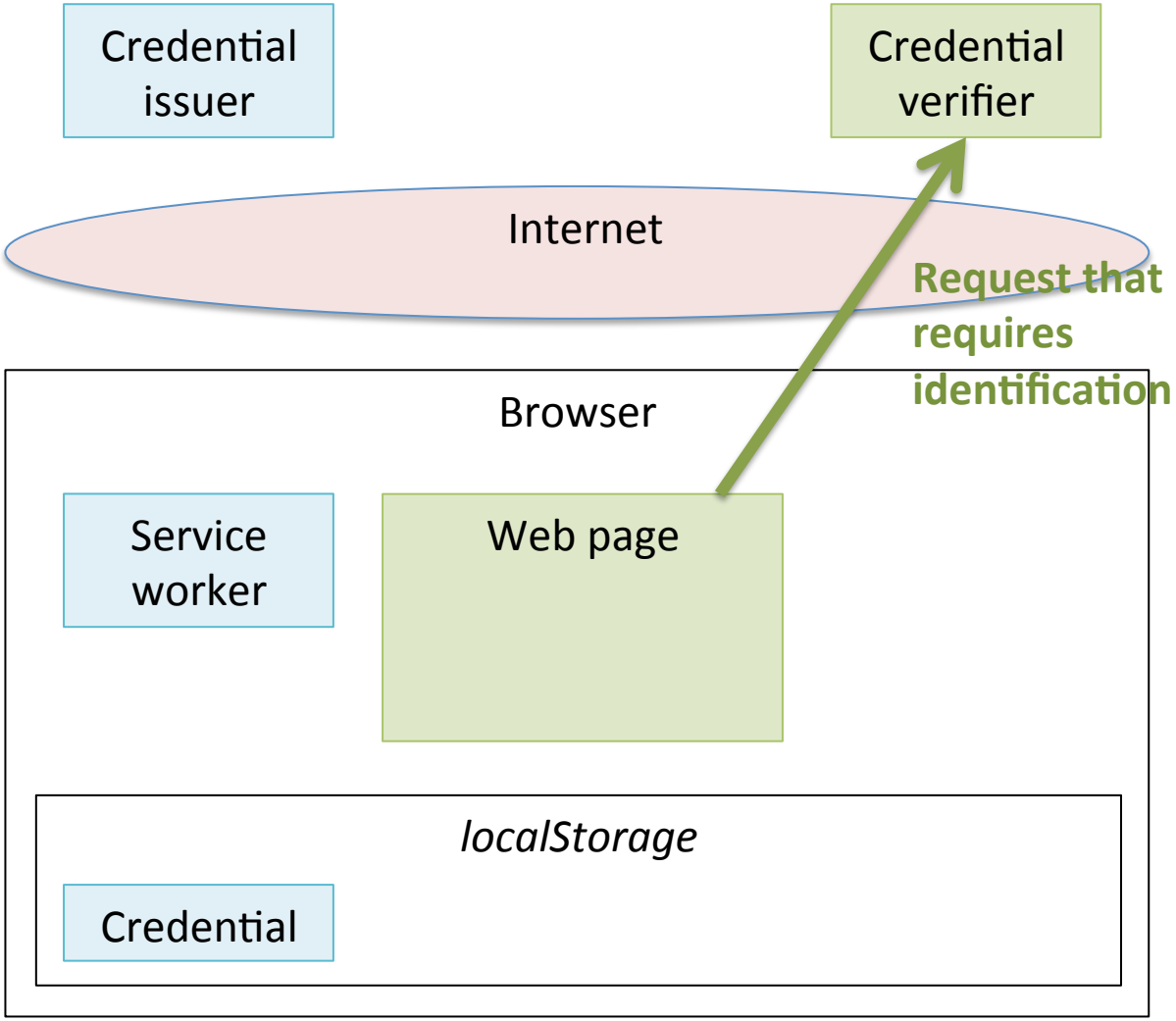
Solution 1 – Issuance



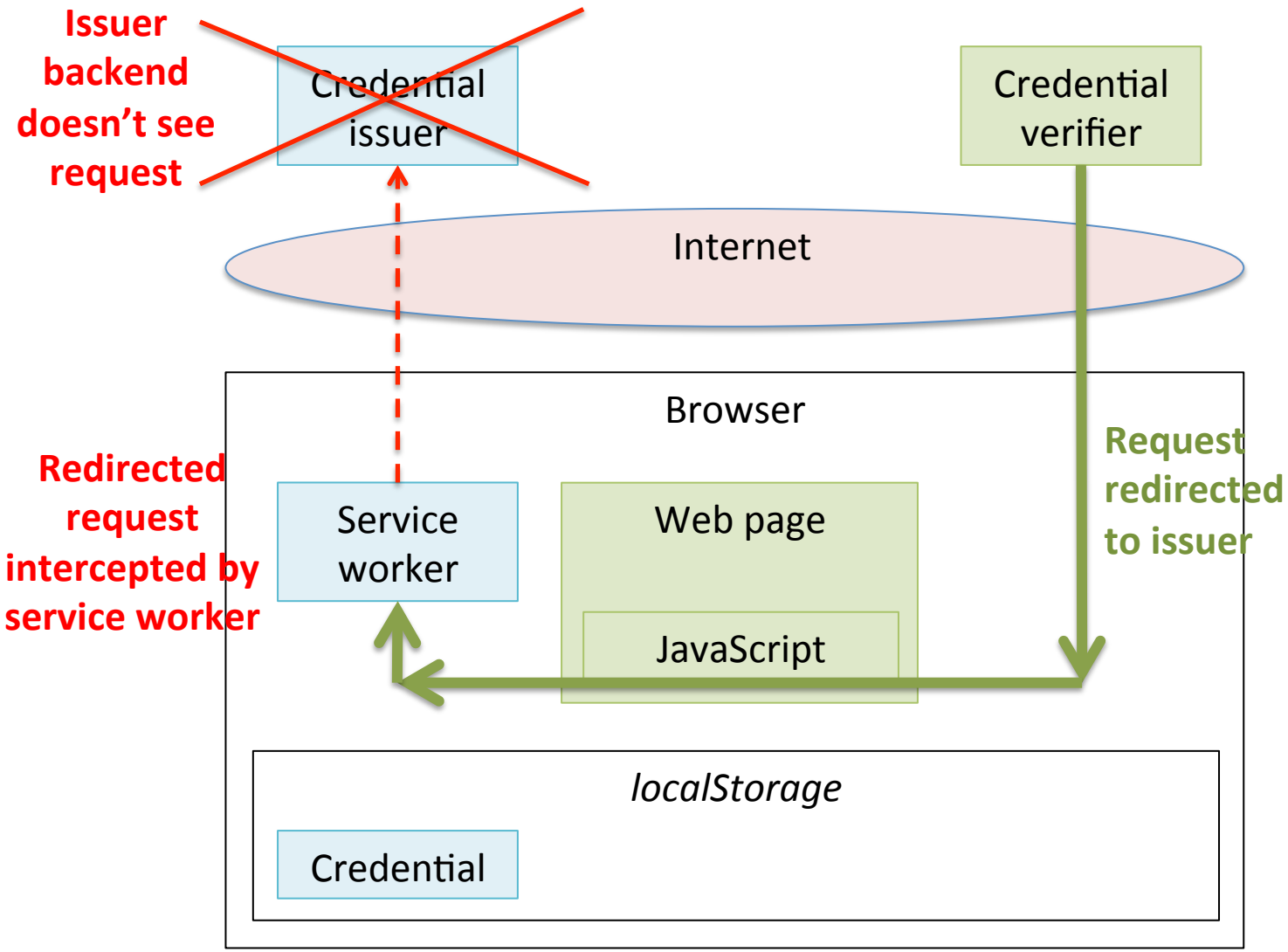
Solution 1 – Issuance



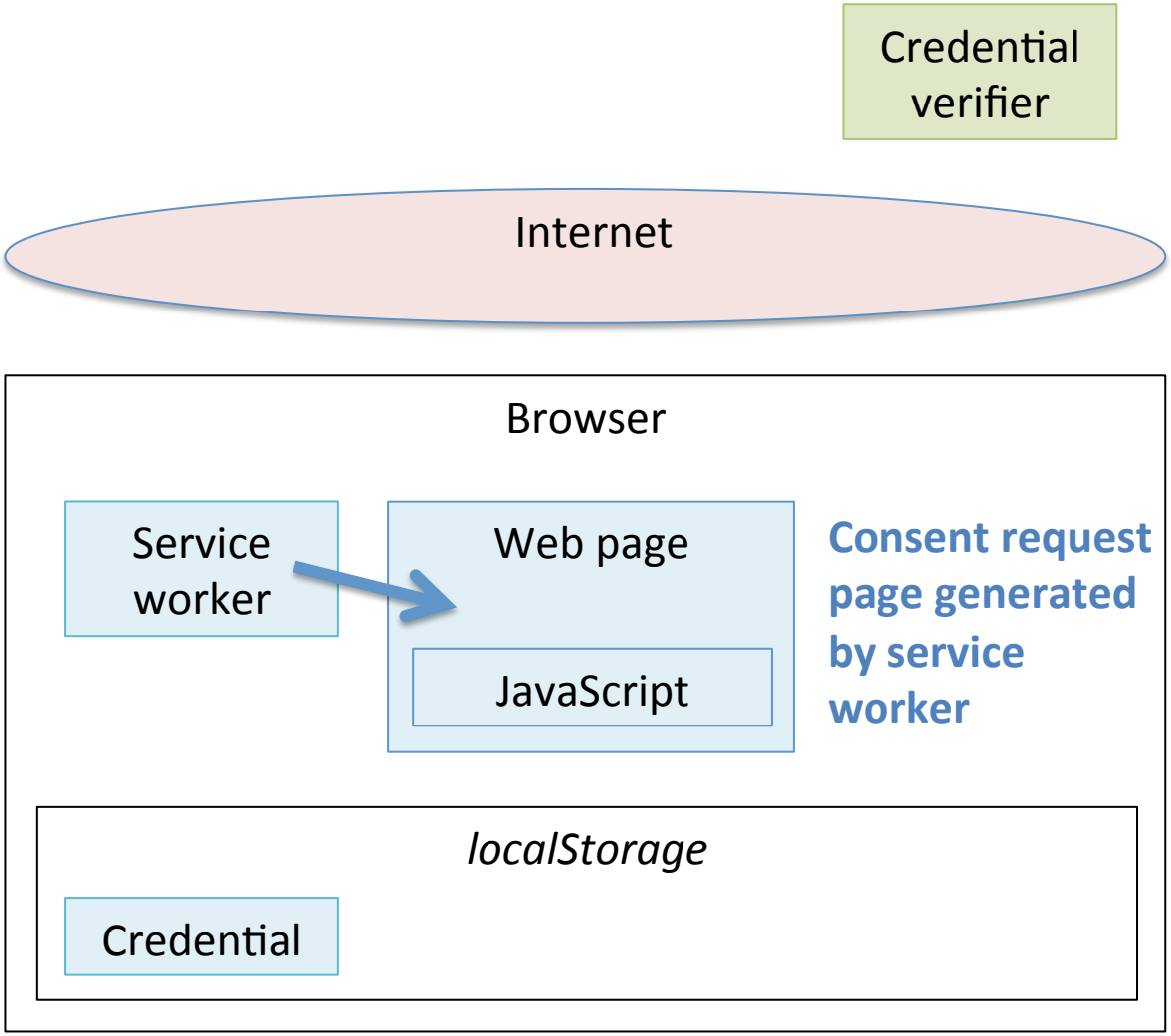
Solution 1 – Presentation



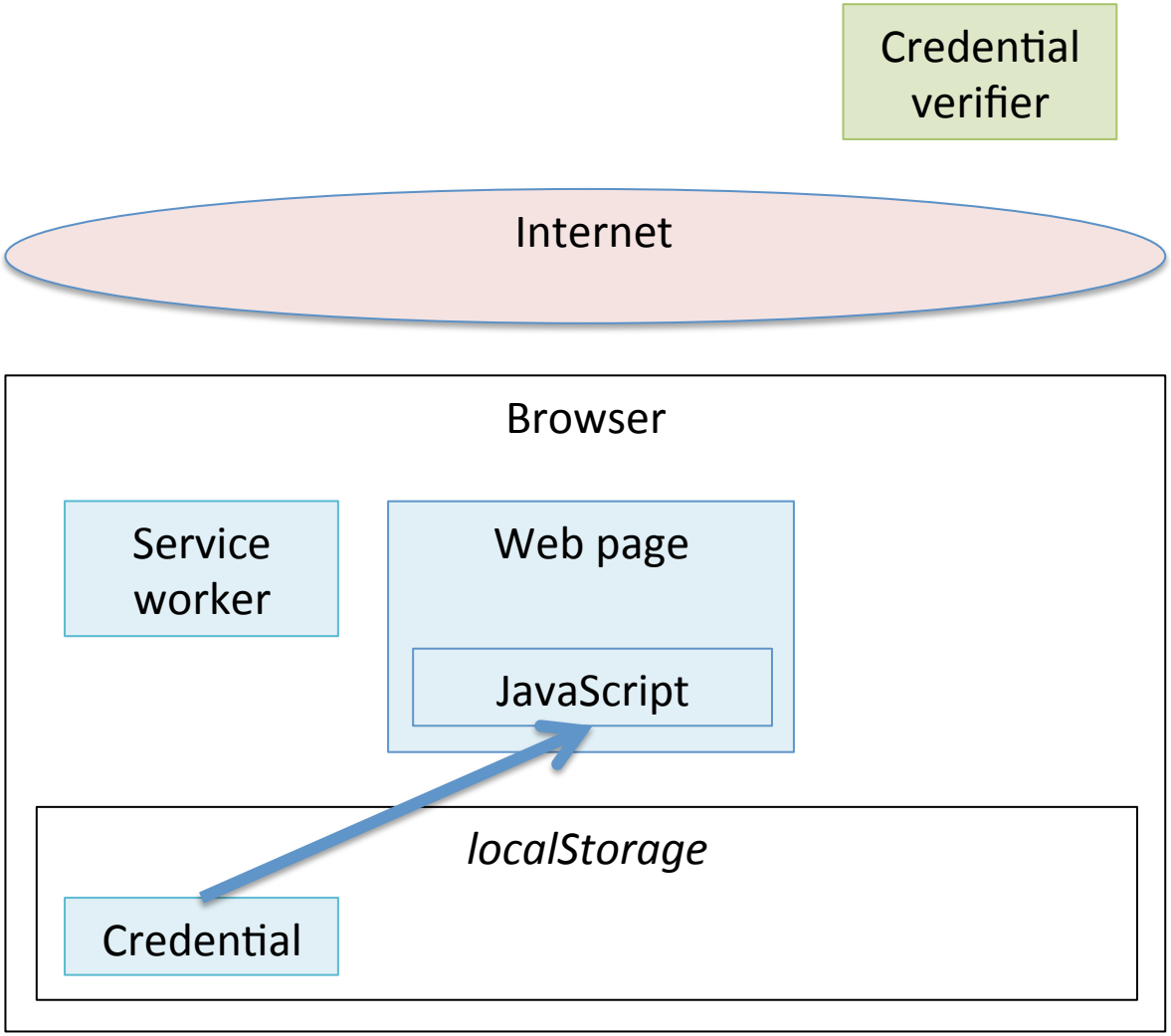
Solution 1 – Presentation



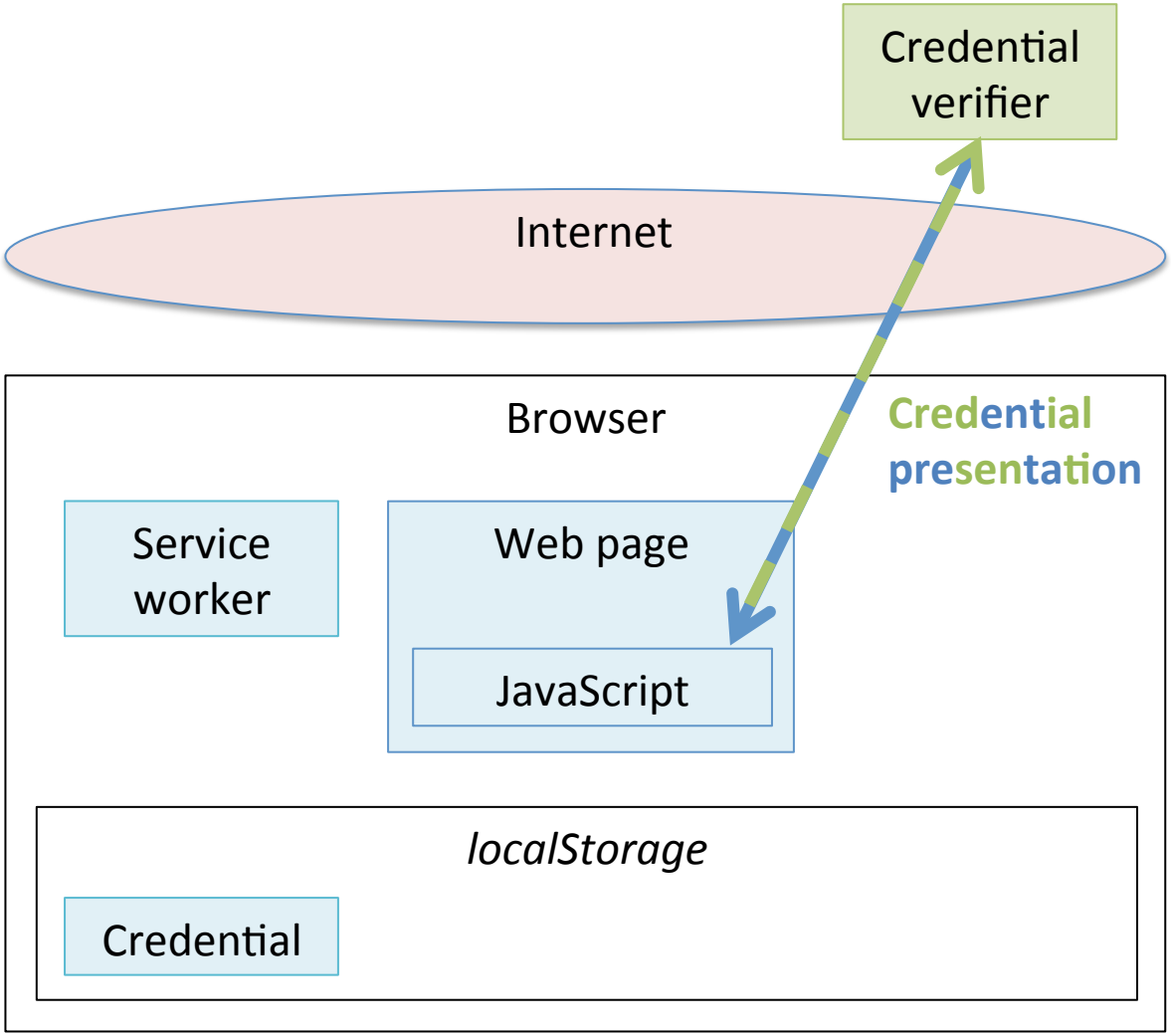
Solution 1 – Presentation



Solution 1 – Presentation



Solution 1 – Presentation



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

TTP = Trusted 3rd party, FE = Front-end, BE = Back-end; SW = service worker

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	Secure				apture
SC/preloaded credential	Capture				ecure
SC/on-card key pair gen	Capture				ecure
SC/TTP 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
TTP = Trusted third party; TP HW = Tamper-proof hardware, e.g. TPM, Secure Element
TEE = Trusted execution environment

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP 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
TTP = Trusted third party; TP HW = Tamper-proof hardware, e.g. TPM, Secure Element
TEE = Trusted execution environment

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP 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) trusted issuer; (ii) no malware on subject's device; and (iii) no physical capture of subject's device

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP 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 by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP 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 by issuer after issuance, e.g. attack by an issuer insider after issuance

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP 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 third party is more secure than smartcard assuming no malware or physical capture, even with on-card key pair generation, if the card is provided by the issuer. Must use TTP firmware on SC to match LS controlled by TTP

SECURITY POSTURES	Attack by issuer at issuance	Attack by issuer after issuance	Malicious JS from 3rd party	Malware	Physical capture
Solution 1: LS	Capture	Capture	Secure	Capture	Capture
Solution 2: IDB/CK	Capture	Use	Secure	Capture	Capture
Solutions 3 and 4: TTP	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/TTP manufacturer	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/

Francisco Corella

fcorella@pomcor.com

Karen Lewison

kplewison@pomcor.com

Any questions?