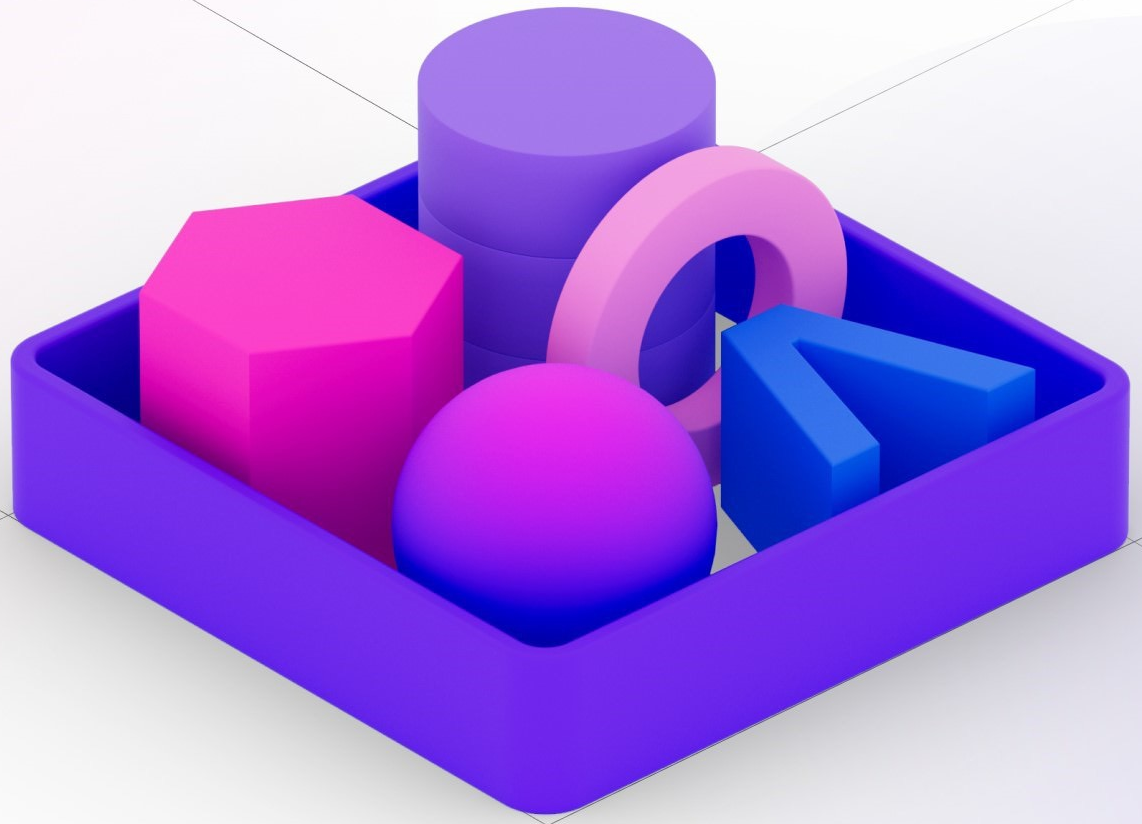
STUDY4
為 學 習 而 生

Build School

.NET Conf
TAIWAN



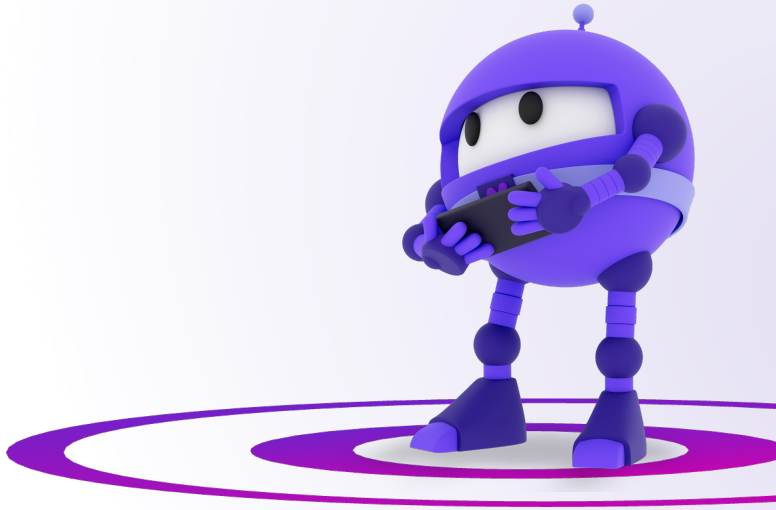
STUDY4 為學習而生 Build School

使用 Passkeys 打造無密碼驗證服務

黃升煌 Mike

Microsoft® MVP Most Valuable Professional

Experts 2023



About Me

- 黃升煌 Mike
  - Angular GDE
  - Microsoft MVP
- Awards
  - 2018 iT 邦幫忙鐵人賽 冠軍
  - 2019 iT 邦幫忙鐵人賽 優選
  - 第 12 屆 iThome 鐵人賽 冠軍

https://fullstackladder.dev

https://github.com/wellwind

https://www.facebook.com/fullstackledder

繁 / 簡

INSIDE 報導 數位轉型 半導體 網路 商業 焦點 會員專屬 登入

趨勢

15 年前發明煩死人的密碼規則，Bill Burr：抱歉浪費大家時間

2023/10/19 ▪ Mia ▪ 密碼 、 規則 、 設定

相信大家都心有戚戚焉，每次碰到這些落落長的要求，心裡就有底，這組密碼防止自己登入的次數將比防止被盜的次數還來得多。

https://www.inside.com.tw/article/10162-the-guy-who-invented-those-annoying-password-rules-now-regrets-wasting-your-time

關於 Passkeys

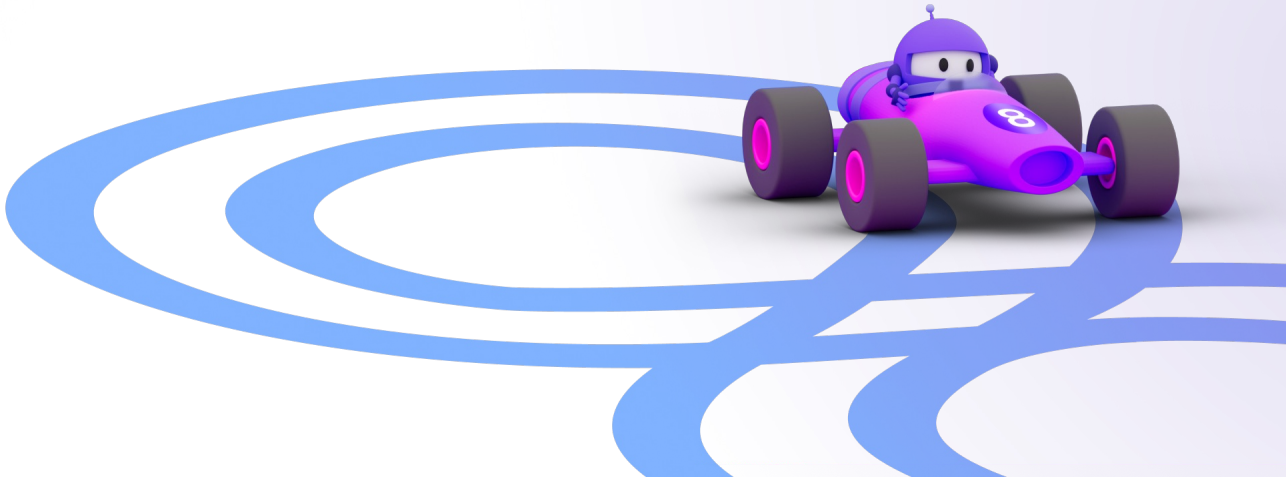
https://fidoalliance.org/passkeys/

- FIDO 聯盟提出的標準
- 一種用來**代替密碼**，以達到**無密碼**的驗證方式
- 提供更快速、更簡便、更安全的登錄方式
- 具有**抵抗釣魚網站**攻擊的能力
- 可以**跨裝置**進行驗證
- 簡化應用程式和網站的帳戶註冊流程

Capability	Android	Chrome OS	iOS/iPad OS	macOS	Ubuntu	Windows
Synced Passkeys	v9+	Planned [1]	v16+	v13+ [2]	Not Supported	Planned [1]
Browser Autofill UI	Chrome Edge Firefox	Planned	Safari Chrome Edge Firefox	Safari Chrome Edge Firefox	Not Supported	Chrome [3] Edge Firefox
Cross-Device Authentication ***Authenticator***	v9+	n/a	v16+	n/a	n/a	n/a
Cross-Device Authentication ***Client***	Planned	v108+	v16+	v13+	Chrome Edge	v23H2+
Third-Party Passkey Providers	v14+	Browser Extensions	v17+	v14+	Browser Extensions	Browser Extensions Native Planned

DEMO

Google Passkeys 註冊/登入



WebAuthn (Web Authentication)

https://webauthn.guide/

- FIDO 聯盟與 W3C 聯合提出的驗證標準
- 利用**非對稱金鑰**的方式進行驗證
  - 伺服器端只會儲存公鑰
  - 私鑰儲存在使用者個人設備上，可透過**生物特徵**進行加密
  - 私鑰會**與網域綁定**，避免釣魚網站攻擊
- Passkeys 透過 WebAuthn，同時加入**跨裝置驗證**的支援
  - 需要作業系統支援
  - 開發上只需要調整 WebAuthn API 的參數

5 分鐘非對稱金鑰小教室

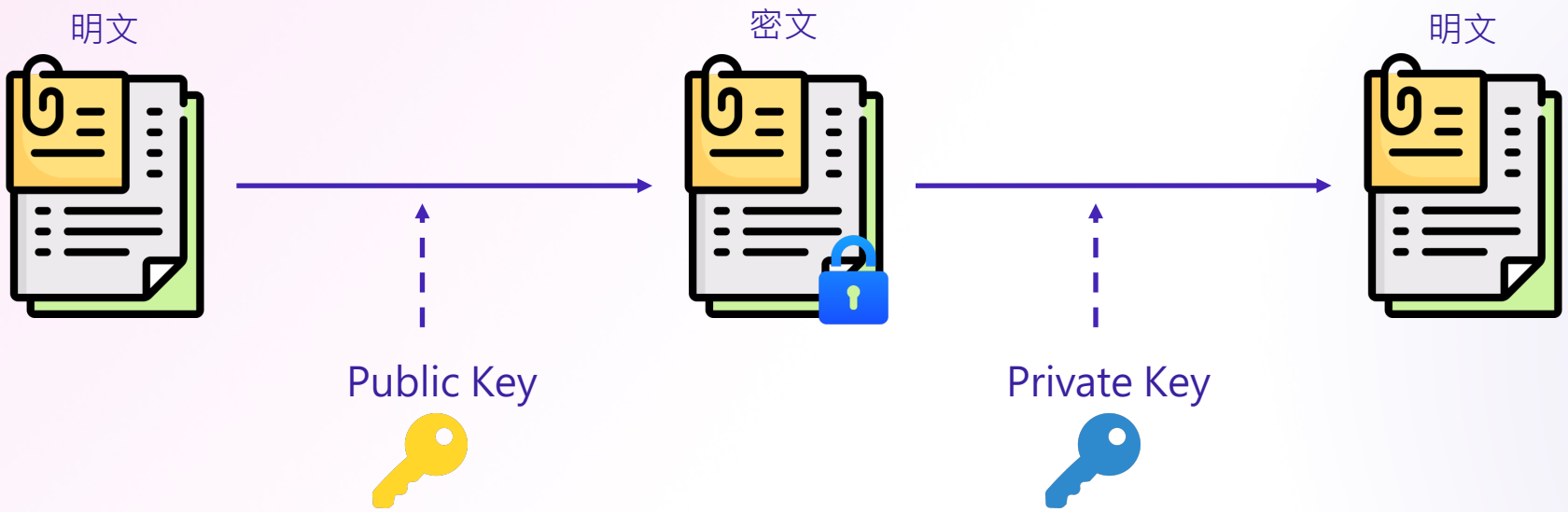
- 使用金鑰對 (分成公鑰與私鑰) 的加解密 & 驗證系統

Public Key

Private Key

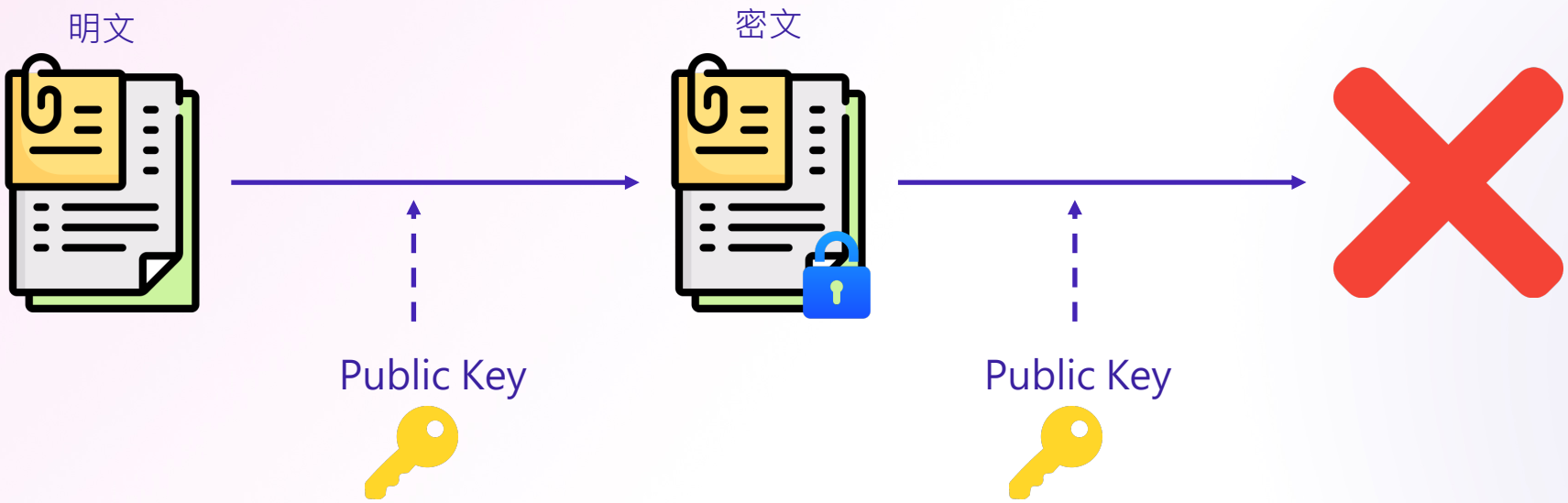
5 分鐘**非對稱金鑰**小教室

- 使用公鑰加密的內容，**可以被私鑰解密**



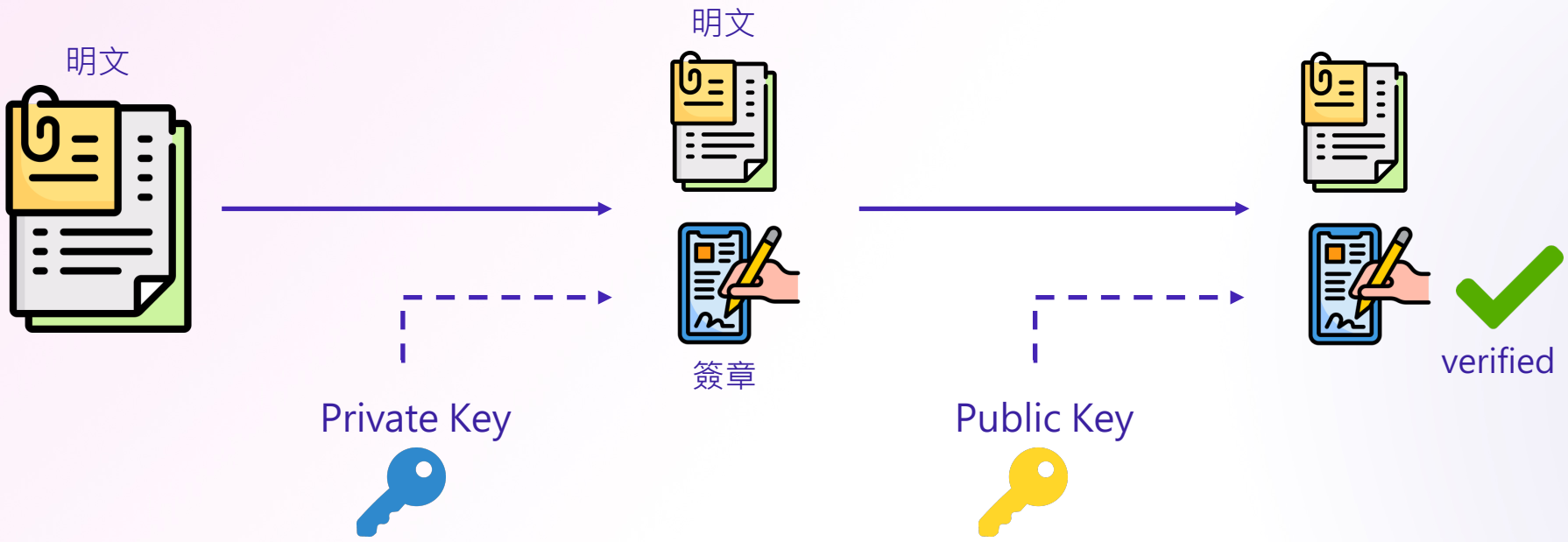
5 分鐘**非對稱金鑰**小教室

- 使用公鑰加密的內容，**無法被原公鑰解密**

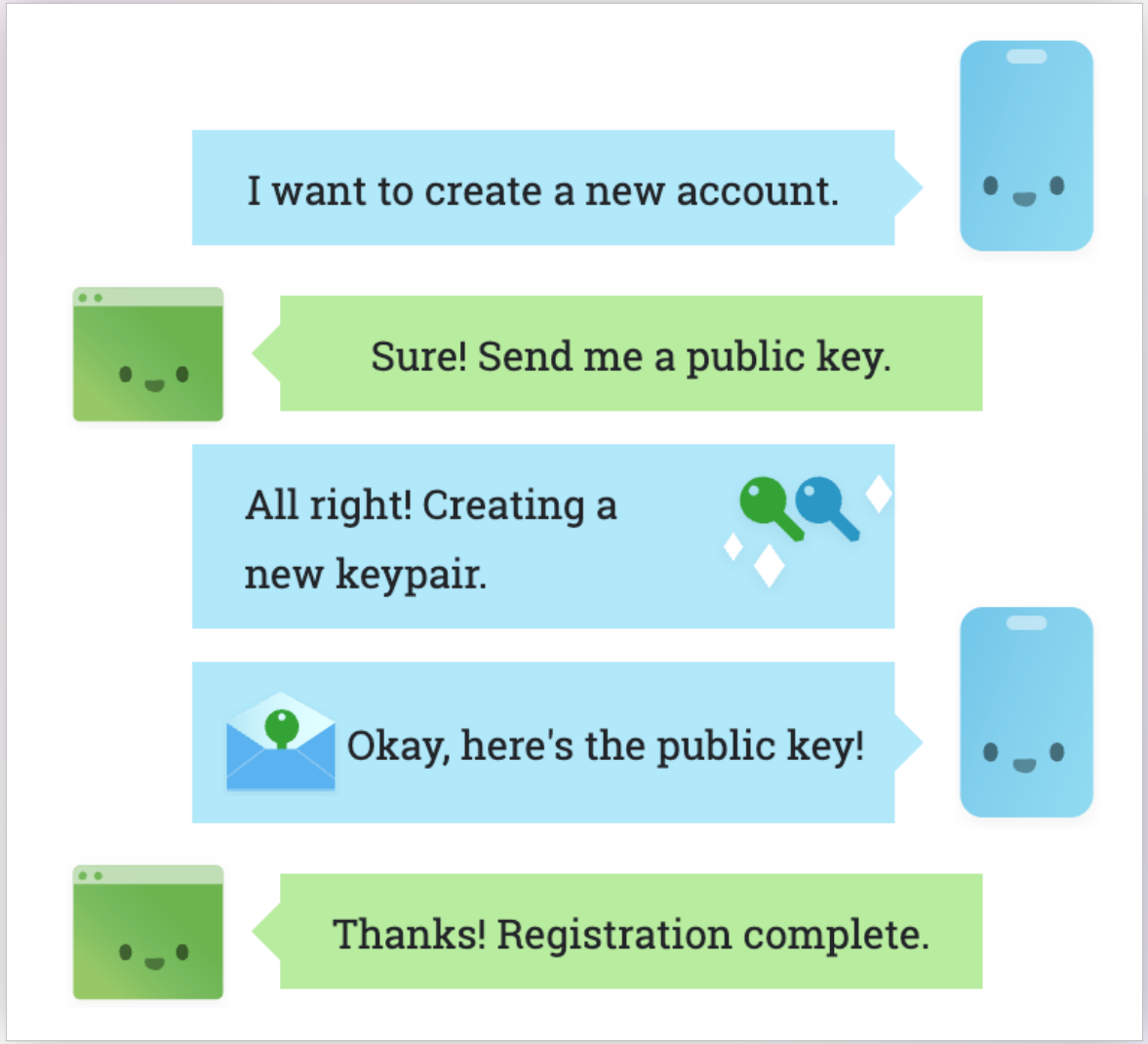


5 分鐘**非對稱金鑰**小教室

- 可以使用私鑰對內容進行簽章，並透過公鑰驗證簽章是否正確



使用 WebAuthn 註冊裝置



使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

從伺服器產生的隨機字串 (challenge)
避免**重播攻擊** (replay attack)
SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

relying party 資訊
通常也就是驗證伺服器的資訊
id 需要與網域名稱相符，避免**釣魚網站攻擊**
SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

目前要註冊裝置的使用者資訊

SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

指定允許使用的簽章演算法
SPEC

建議包含以下幾種演算法已達到最佳的支援

- -8 (Ed25519)
- -7 (ES256)
- -257 (RS256)

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

(非必要) 用來限定可以使用的驗證器來源

SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

註冊流程到期時間 (毫秒)
SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};

const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

是否要回傳驗證器資訊給伺服器

SPEC

使用 WebAuthn 註冊裝置

```
const publicKeyCredentialCreationOptions = {
  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),
  rp: {
    name: "FullStackLadder",
    id: "fullstackladder.dev",
  },
  user: {
    id: Uint8Array.from("User Id", (c) => c.charCodeAt(0)),
    name: "mike@fullstackladder.dev",
    displayName: "Mike Huang",
  },
  pubKeyCredParams: [{ alg: -7, type: "public-key" }],
  authenticatorSelection: {
    authenticatorAttachment: "cross-platform"
  },
  timeout: 60000,
  attestation: "direct",
};
```

```
const credential = await navigator.credentials.create({
  publicKey: publicKeyCredentialCreationOptions,
});
```

產生裝置公鑰及相關認證資訊

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAttestationResponse {
    clientDataJSON: ArrayBuffer(121),
    attestationObject: ArrayBuffer(306),
  },

  type: 'public-key'

}
```

產生的認證資訊

```
PublicKeyCredential {
  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',
  rawId: ArrayBuffer(59),
  response: AuthenticatorAttestationResponse {
    clientDataJSON: ArrayBuffer(121),
    attestationObject: ArrayBuffer(306),
  },
  type: 'public-key'
}
```

產生的認證 ID
SPEC

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAttestationResponse {
    clientDataJSON: ArrayBuffer(121),
    attestationObject: ArrayBuffer(306),
  },

  type: 'public-key'

}
```

也是認證 ID，只是為 binary 格式

SPEC

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAttestationResponse {
    clientDataJSON: ArrayBuffer(121),
    attestationObject: ArrayBuffer(306),
  },

  type: 'public-key'

}
```

瀏覽器與驗證器之間傳遞的資料
SPEC

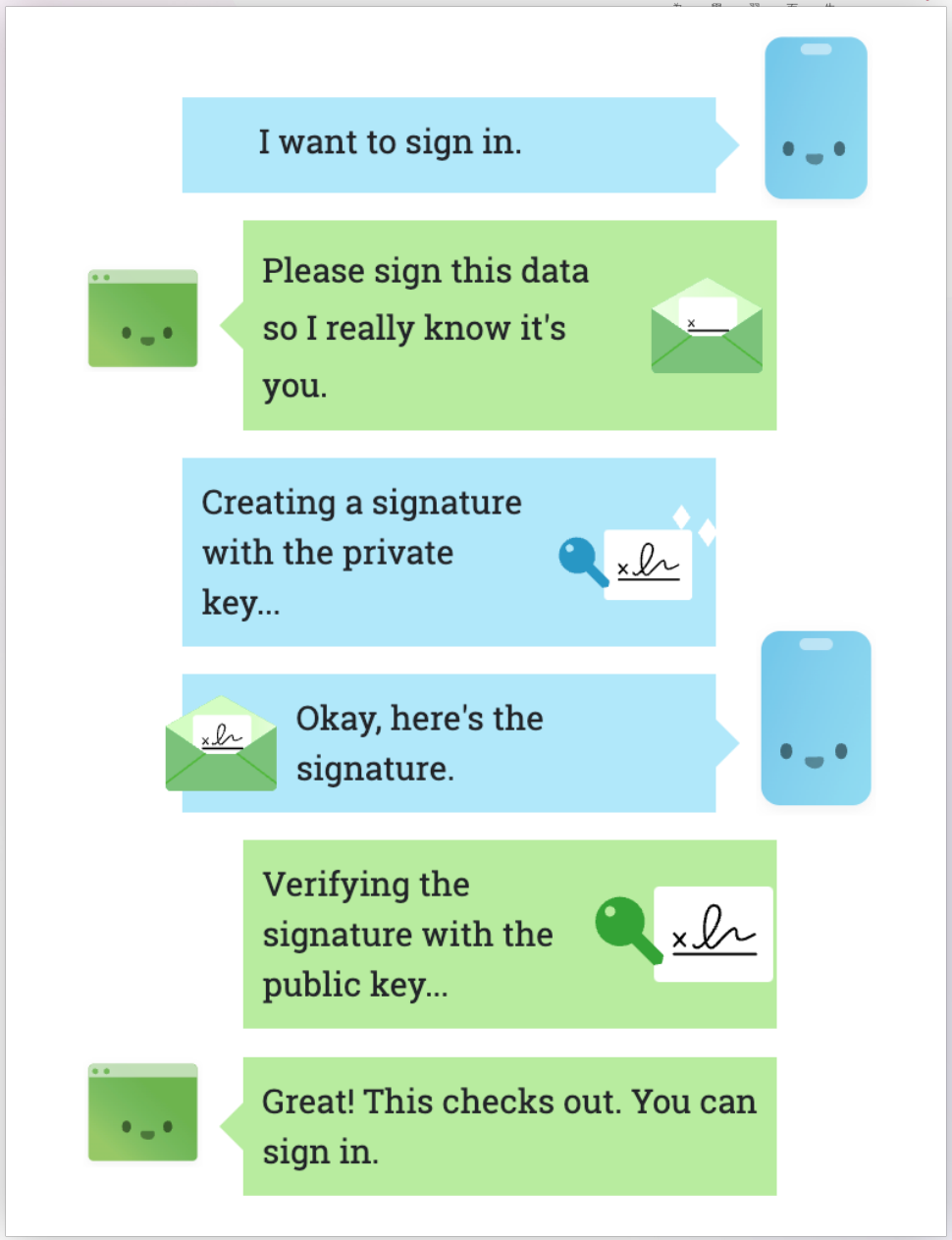
產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAttestationResponse {
    clientDataJSON: ArrayBuffer(121),
    attestationObject: ArrayBuffer(306),
  },

  type: 'public-key'

}
```

驗證器相關資料，包含**公鑰**等資訊

SPEC

使用 WebAuthn 登入



使用 WebAuthn 驗證

```
const publicKeyCredentialRequestOptions = {

  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),

  allowCredentials: [
    {
      id: Uint8Array.from(
        "credential id", (c) => c.charCodeAt(0)),
      type: "public-key",
      transports: ["internal"],
    },
  ],

  timeout: 60000,

};

const assertion = await navigator.credentials.get({
  publicKey: publicKeyCredentialRequestOptions,
});
```

使用 WebAuthn 驗證

```
const publicKeyCredentialRequestOptions = {

  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),

  allowCredentials: [
    {
      id: Uint8Array.from(
        "credential id", (c) => c.charCodeAt(0)),
      type: "public-key",
      transports: ["internal"],
    },
  ],

  timeout: 60000,

};

const assertion = await navigator.credentials.get({
  publicKey: publicKeyCredentialRequestOptions,
});
```

(非必要) 允許使用的驗證資訊
可用來限定登入的裝置
SPEC

使用 WebAuthn 驗證

```
const publicKeyCredentialRequestOptions = {

  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),

  allowCredentials: [
    {
      id: Uint8Array.from(
        "credential id", (c) => c.charCodeAt(0)),
      type: "public-key",
      transports: ["internal"],
    },
  ],

  timeout: 60000,

};

const assertion = await navigator.credentials.get({
  publicKey: publicKeyCredentialRequestOptions,
});
```

登入流程的到期時間 (毫秒)
SPEC

使用 WebAuthn 驗證

```
const publicKeyCredentialRequestOptions = {

  challenge: Uint8Array.from(
    "challenge from server", (c) => c.charCodeAt(0)),

  allowCredentials: [
    {
      id: Uint8Array.from(
        "credential id", (c) => c.charCodeAt(0)),
      type: "public-key",
      transports: ["internal"],
    },
  ],

  timeout: 60000,

};
```

```
const assertion = await navigator.credentials.get({
  publicKey: publicKeyCredentialRequestOptions,
});
```

取得註冊的裝置資訊，以及相關**簽章**結果

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAssertionResponse {
    authenticatorData: ArrayBuffer(191),
    clientDataJSON: ArrayBuffer(118),
    signature: ArrayBuffer(70),
    userHandle: ArrayBuffer(10),
  },

  type: 'public-key'
}
```

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAssertionResponse {
    authenticatorData: ArrayBuffer(191),
    clientDataJSON: ArrayBuffer(118),
    signature: ArrayBuffer(70),
    userHandle: ArrayBuffer(10),
  },

  type: 'public-key'
}
```

用來完成這次驗證的裝置資訊
注意：不會包含公鑰資訊
SPEC

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAssertionResponse {
    authenticatorData: ArrayBuffer(191),
    clientDataJSON: ArrayBuffer(118),
    signature: ArrayBuffer(70),
    userHandle: ArrayBuffer(10),
  },

  type: 'public-key'
}
```

由驗證器私鑰產生的**簽章**資訊
Client 端根據特定規則產生簽章
Server 端使用已註冊的公鑰驗證簽章
SPEC

產生的認證資訊

```
PublicKeyCredential {

  id: 'ADSUllKQmbqdGtpu4sjseh4cg2TxSvrbcHDTBsv4NSSX9...',

  rawId: ArrayBuffer(59),

  response: AuthenticatorAssertionResponse {
    authenticatorData: ArrayBuffer(191),
    clientDataJSON: ArrayBuffer(118),
    signature: ArrayBuffer(70),
    userHandle: ArrayBuffer(10),
  },

  type: 'public-key'
}
```

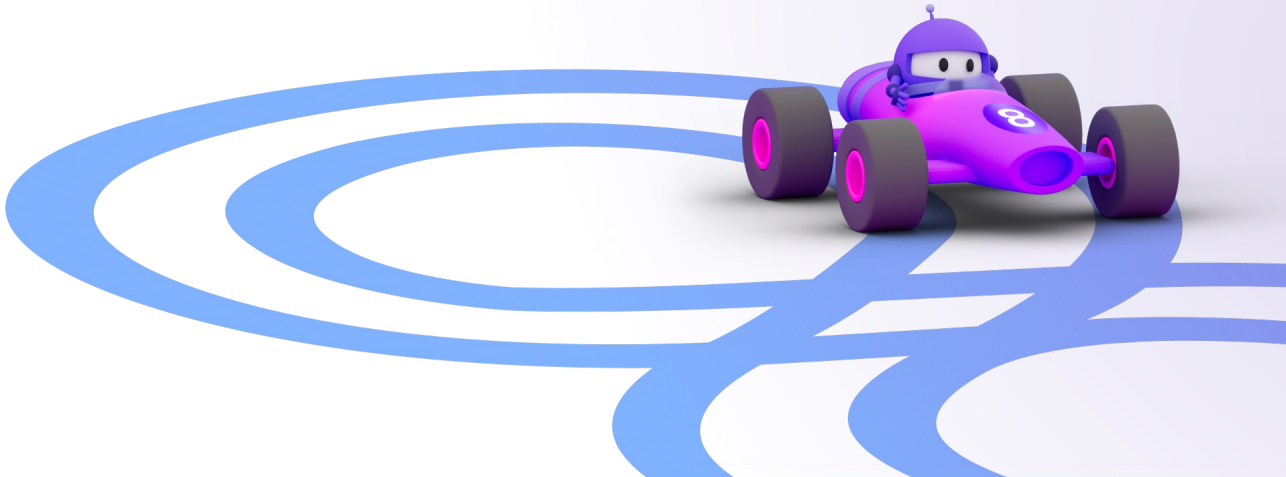
由驗證器提供的額外使用者資訊

SPEC

DEMO

實作 Passkeys 服務

https://github.com/wellwind/dotnetconf-2023-passkeys-demo



Resources

Passkeys (Passkey Authentication)

passkeys.com

WebAuthn vs Passkeys

FIDO2 for .NET

[Google] 使用密碼金鑰進行無密碼登入

一起來了解 Web Authentication

[影片]有 Google 帳號的人注意！這個功能請火速把它打開