Going Passwordless

A Practical Guide to Passkeys in ASP.NET Core

Maarten Balliauw
Bluesky: @maartenballiauw.be

The problem with passwords

Duende.

Passwords...

For users

- **Have to remember password**
  - Not complex enough
  - Re-use
  - Post-its 😅
- **Phishing**

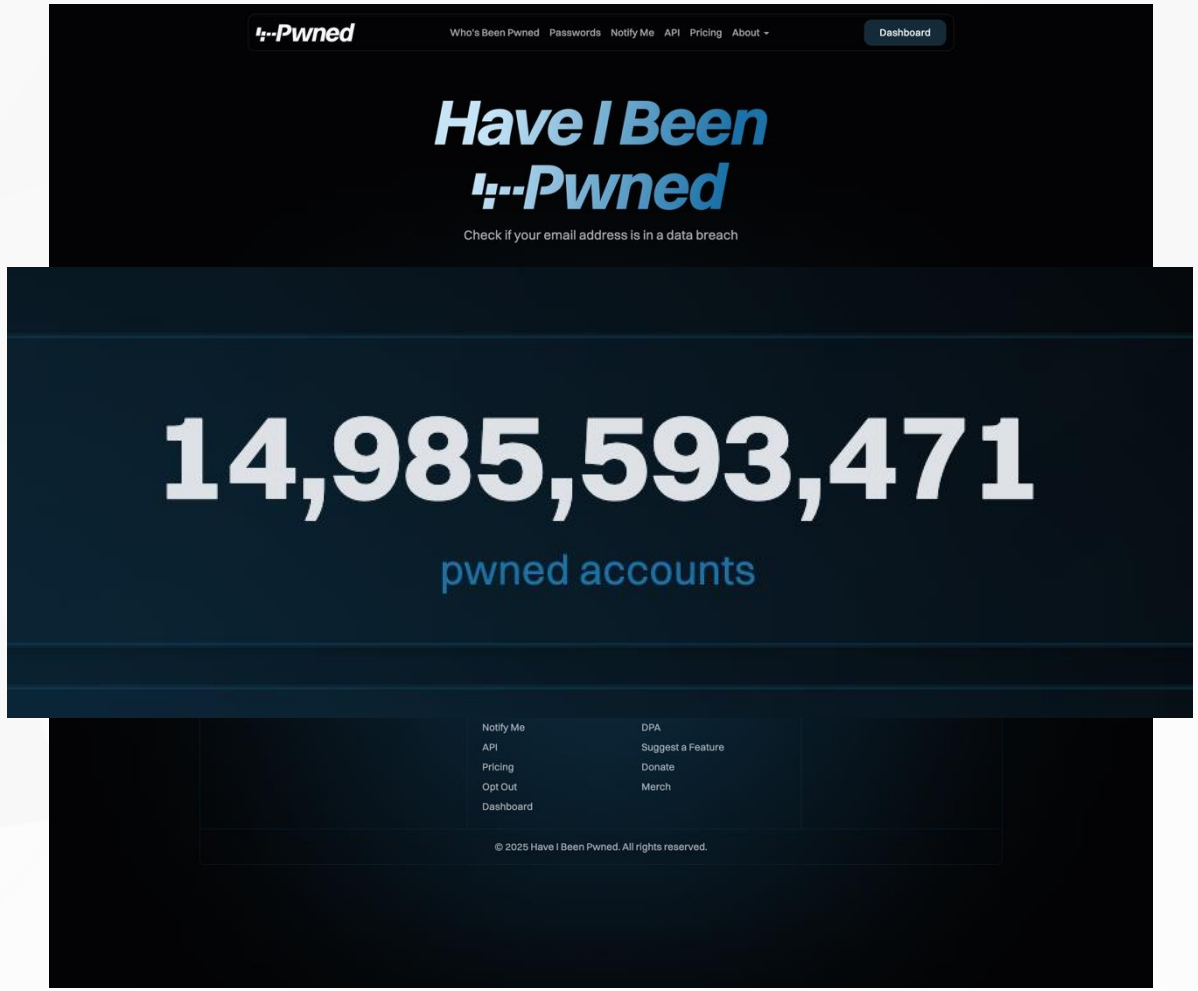
For developers

- **Need to store passwords**
  - Plain text 😭
  - Salted and hashed
- **JavaScript injection and other attack vectors**

Duende.

How to solve?

- **Guidance to non-techies**
  - “Remember a few different ones”
- **Guidance to (somewhat) techies**
  - “Use a password manager that creates random passwords”



https://haveibeenpwned.com/ - July, 2025

Multi-factor authentication!

- Adds another step to the authentication process
- Need to store backup keys
- Text messages are insecure
- Deliverability https://blog.stillgreenmoss.net/sms-2fa-is-not-just-insecure-its-also-hostile-to-mountain-people
- Need another app

What if we could get rid of passwords?

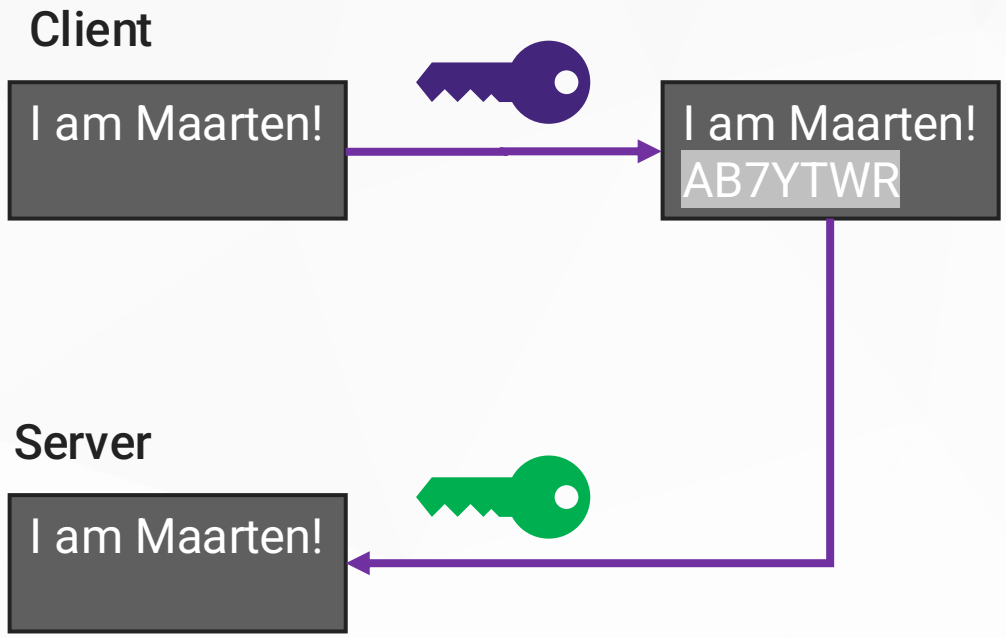
- No need to remember passwords
- No need to store password
- No risk of passwords getting compromised

Let’s fix this – passwordless!

Duende.

Public-key cryptography/signing

- **Client generates private/public key**
  - Client stores private key
  - Server stores public key
- **Login becomes signing and validating**

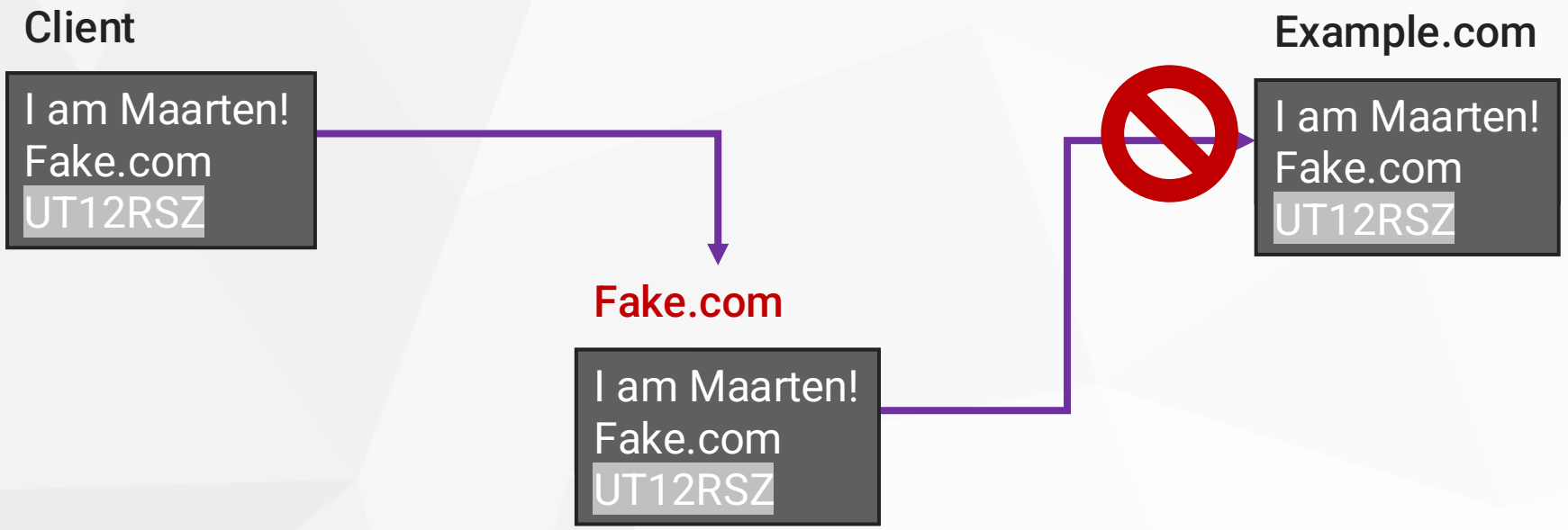


Public-key cryptography/signing

- **3 actions:**
  - Generate key pair and store on client + server
  - Sign a message with private key
  - Verify message with public key
- **Solves breaches**
  - Public key useless by itself
- **Still leaves some other issues…**

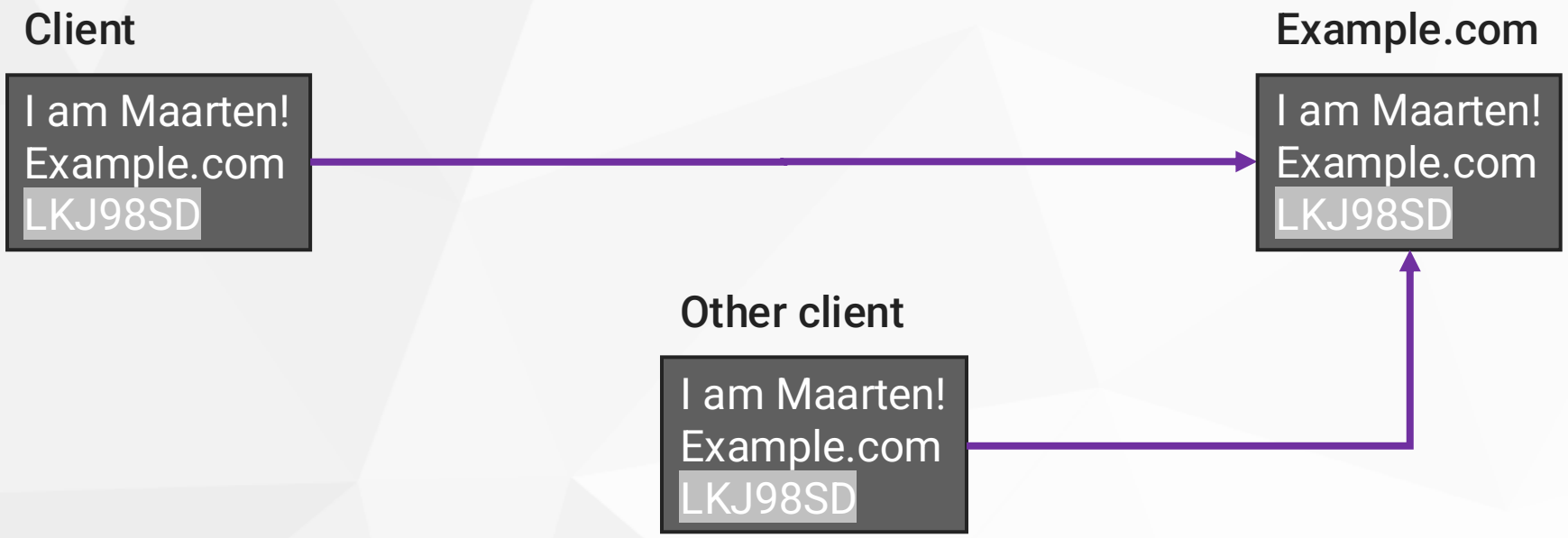
Duende.

Phishing



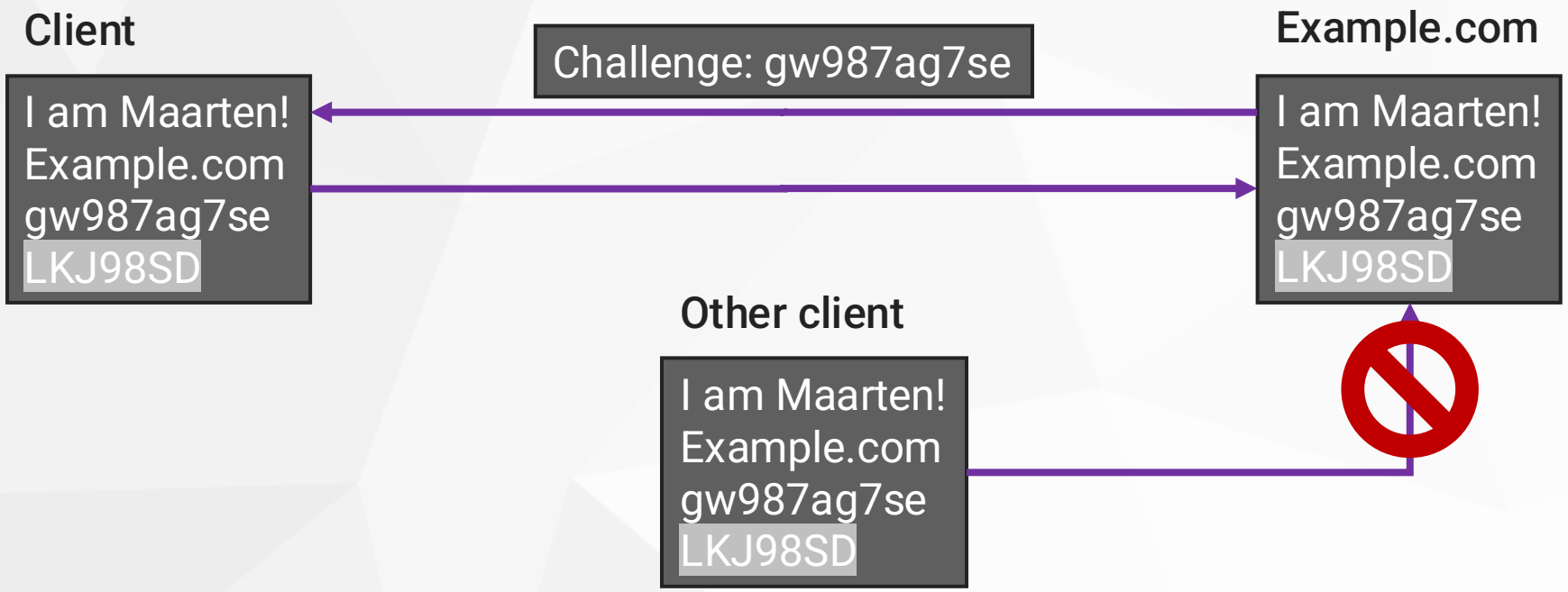
Duende.

Replay



Duende.

Replay



Public-key cryptography/signing

- **Generate, sign, verify**
- **Solves breaches**
  - Public key useless by itself
- **Solves phishing**
  - Include origin in message
- **Solves replay**
  - Include server challenge in message

Duende.

FIDO, U2F, hardware keys

Duende.

FIDO alliance

- “The FIDO Alliance is an open industry association with a focused mission: reduce the world’s reliance on passwords.“
- Consortium of big companies, government organizations, security researchers

Duende.

Universal Second Factor (U2F)

- **Public-key cryptography/signing**
- **Security keys**
  - Usually physical hardware keys (e.g. Yubikey)
- **New concepts!**
  - Attestation to prove type of security key
  - User presence to prove the user is there
  - Signature counter to detect cloned keys
  - Statelessness to derive keys (work around limited storage in hardware)

Image from YUBICO

Statelessness

- **Limited storage in hardware key**
- **Symmetric root certificate in hardware key**
  - Credential ID + public key for origin stored on server
  - Credential ID contains encrypted seed to re-generate private key
- **Some issues…**
  - Credential ID needs to be retrieved from server (discoverability)
  - Ideally needs username/password to do so (verification)
  - Reset root certificate == all credentials lost

Passkeys (and WebAuthn)

Passkeys

- **Public-key cryptography/signing**
- **U2F properties**
  - Attestation, user presence, signature counter
- **Stateful**
  - Helps discoverability, reset single credential, synchronization
- **Options**
  - Require user verification (e.g. PIN/biometrics), platform vs. cross-platform, UI hints, …
- **Authenticators**

Duende.

Authenticators

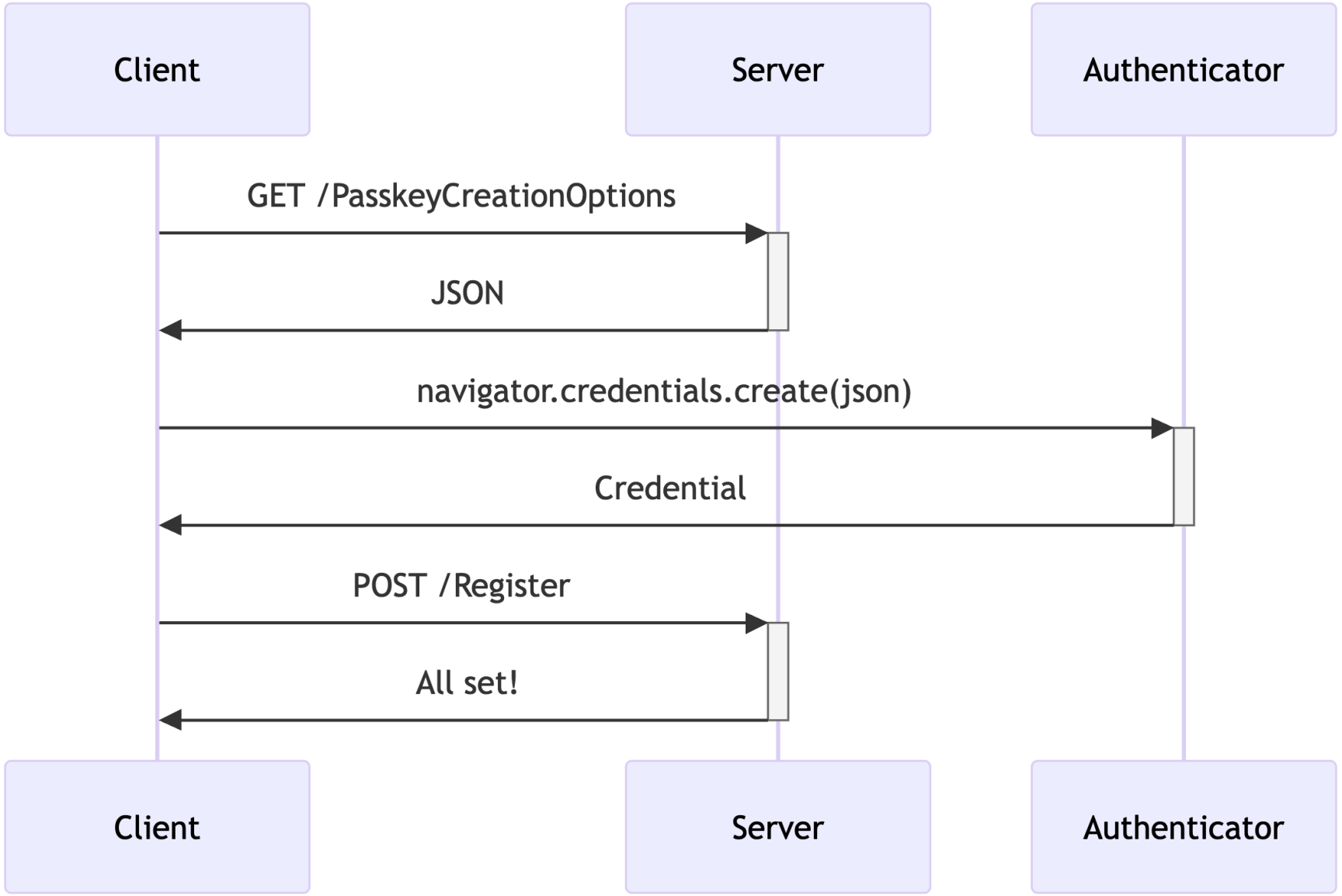
- **Hardware keys**
- **Software keys**
  - Operating system (+ Trusted Platform Module (TPM))
  - Password manager
  - 👀 Synchronization between multiple devices
  - 👀 Cross-device login (access web app on desktop, sign in on mobile)

Duende.

Demo

Logging in with a passkey

Duende.


Client
Server
Authenticator
GET /PasskeyCreationOptions
JSON
navigator.credentials.create(json)
Credential
POST /Register
All set!
Client
Server
Authenticator

Duende.

WebAuthn (client side)

- **Protocol exposing U2F (CTAP1) and FIDO2 (CTAP2)**
- **Browser API**
  - `navigator.credentials.create()`
  - `navigator.credentials.get()`
  - (2 more)
  - https://developer.mozilla.org/en-US/docs/Web/API/Navigator/credentials
- **OS-specific APIs to use WebAuthn with native apps (not covered in this talk)**

WebAuthN (server-side endpoints)

- **Register a key**
  - `PasskeyCreationOptions` generates a value to sign + crypto + options
  - An endpoint to store the credential
- **Use a key**
  - `PasskeyRequestOptions` generates a value to sign + crypto + options
  - An endpoint to validate the credential

Duende

ASP.NET Core and passkeys

Libraries

- **Community**
  - FIDO2 .NET
  - WebAuthn.Net
- **Commercial**
  - RockSolidKnowledge FIDO2 for ASP.NET Core
- **Microsoft**
  - .NET 10

Duende.

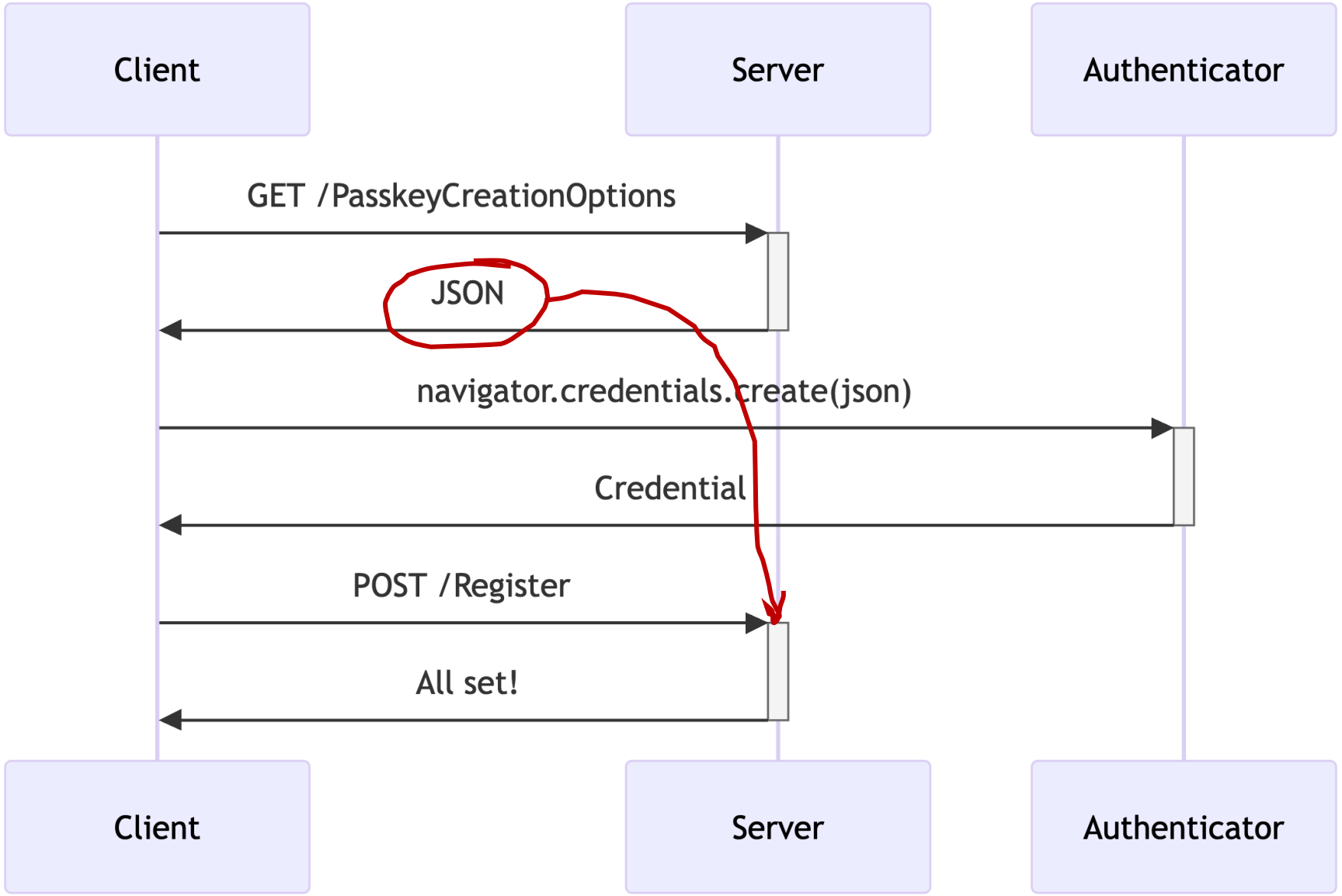
Demo

.NET 10 passkey support

.NET 10 passkey support

- **Part of ASP.NET Identity** (`Microsoft.AspNetCore.Identity`)
- **Included in Blazor Web App with Individual Authentication template**
  - Not (yet?) in Razor Pages / MVC / scaffolding, but it works there
- **Entity Framework Core-based store**
  - `IdentitySchemaVersions.Version3`
  - Adds `AspNetUserPasskeys` table

Duende.


Client
Server
Authenticator
GET /PasskeyCreationOptions
JSON
navigator.credentials.create(json)
Credential
POST /Register
All set!
Client
Server
Authenticator

Duende.

Demo

Storage and session data

Storage and session data

- **Passkey creation / request options stored in `ClaimsPrincipal`**
  - Abstracted through `SignInManager<TUser>`
  - `IdentityConstants.TwoFactorUserIdScheme` auth scheme
  - No server-side state is needed
- **Only `AspNetUserPasskeys` table is needed for persisting passkeys data**

Duende.

Considerations

Passwordless!

- **.NET 10 template UX drives towards passkey in addition to password**
- **`PasskeyInfo.IsBackedUp` will be `true` for many password managers**
  - You can also require backed-up credentials
- **Consider hinting the user to remove password**
  - `_usermanager.RemovePasswordAsync(User)`

Duende.

Implement Passkeys Well-Known URLs

- **Is passkey support available?**
- https://www.w3.org/TR/passkey-endpoints/

```
GET /.well-known/passkey-endpoints

{
  "enroll": "https://example.com/account/manage/passkeys/create",
  "manage": "https://example.com/account/manage/passkeys"
}
```

Recovery

- **What when someone loses their passkey?**
- **Up to you!**
  - Username, e-mail, 2FA, backup code, support call with ID proof, …

Duende

Relying Party ID / Origin

Passkeys contain origin, forever

No way to change URL – it would invalidate all passkeys

Duende

Relying Party ID / Origin

- Passkey creation/signing options include a Relying Party Id
- Browser verifies Relying Party Id against current origin
- Browser signs credential with current origin
- Server verifies signature against list of supported origins

Duende

Relying Party ID / Origin

- **Passkey creation/signing options include a Relying Party Id**
  - Defaults to current origin domain
  - Override in `IdentityPasskeyOptions` options:

```
options.ServerDomain = "example.com";
```

Duende.

Relying Party ID / Origin

- **Passkey creation/signing options include a Relying Party Id**
- **Browser verifies Relying Party Id against current origin**
  - More specific origin with less specific RP ID is supported

Origin	RP ID	WebAuthn
https://example.com	example.com	✅
https://www.example.com	www.example.com	✅
https://www.example.com	example.com	✅
https://example.com	www.example.com	⛔

Duende.

Relying Party ID / Origin

- **Passkey creation/signing options include a Relying Party Id**
- **Browser verifies Relying Party Id against current origin**
- **Browser signs credential with current origin**
- **Server verifies signature against list of supported origins**
  - Defaults to current origin domain
  - Override in options:

```
options.ValidateOrigin = c => ValueTask.FromResult(
    !c.CrossOrigin || c.Origin == "https://example.com" || c.Origin == "...");
```

Duende.

Passkeys on subdomains

- **Set Relying Party ID to be more general?**

```
options.ServerDomain = "example.com";
```

user-content.example.com

Duende.

Passkeys on subdomains

- **Set Relying Party ID to be more general, configure allowed origins?**

```
options.ServerDomain = "example.com";
options.ValidateOrigin = context ⇒ ValueTask.FromResult(
    !context.CrossOrigin ||
    context.Origin == "https://example.com" ||
    context.Origin == "https://www.exam
    context.Origin == "https://login.ex
```

user-content.example.com

Beware of user content…

- **Do not just default to using the most general Relying Party ID**
  - Could allow unintended logins
  - Could allow creating/using credentials and exfiltration
- **Don’t serve user content from the same domain**
- **Use a common authentication endpoint**
  - e.g. Duende IdentityServer at `login.example.com`

Passkeys on multiple domains

- **Support same passkey on `example.com`, `example.org`, `example.net`**
- **Use the [Related Origin Requests (ROR)] extension to FIDO2**

```
GET /.well-known/webauthn

{
    "origins": [
        "https://example.com",
        "https://example.co.uk",
        "https://example.be",
        "https://other-example.com"
    ]
}
```

Duende

Passkeys on multiple domains

- **Not flexible, all origins must stay around forever**
- **Recommendation: use a common authentication endpoint**
  - e.g. Duende IdentityServer at `login.example.com`

Duende.

User
`www.example.com`

Do passkeys here

IdentityServer
`login.example.com`

User
`www.example.be`

User
`www.example.co.uk`

User
`www.example.org`

Duende.

Summary

Summary

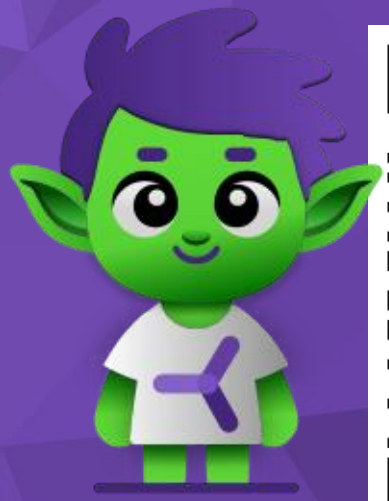
- The problem with passwords
- Going passwordless with public-key cryptography/signing
- U2F, FIDO2, WebAuthn
- .NET 10
- Relying Party ID and Origin

Duende.

Recommended reading

A Tour of WebAuthn by Adam Langley
https://duende.link/tourofwebauthnbook

Duende.



Thank you!

duendesoftware.com

Maarten Balliauw

Bluesky: @maartenballiauw.be