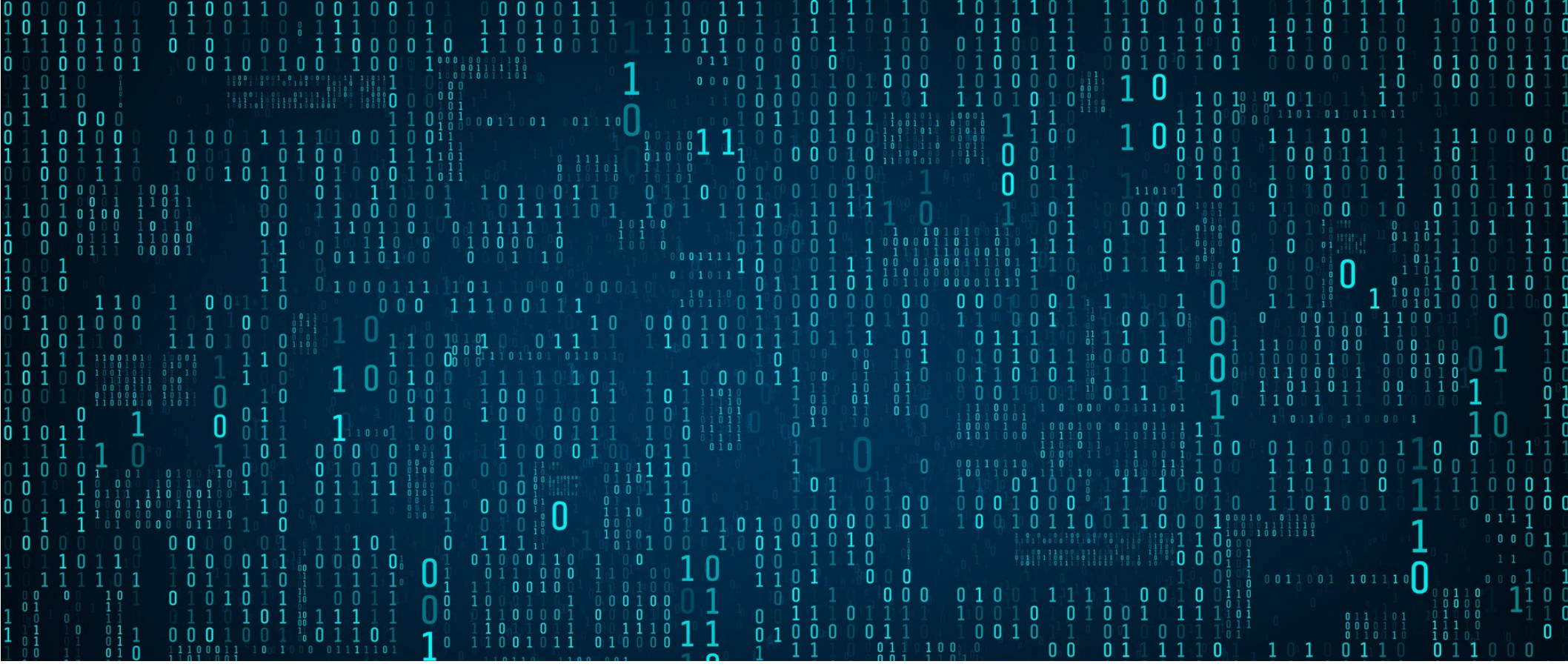


Understanding and Implementing Passkeys

Justin Scott, CISSP

Chief Information Security Officer, Smart Communications

Adobe ColdFusion Summit

September 22, 2025

Hi, I’m Justin Scott

- BBS sysop in the mid 1990’s
- Won a copy of Allaire ColdFusion 4 at SysCon in 1999
- Architect and developer for hundreds of applications
- Network, Systems, and Database admin
- Smart Communications since 2009 as IT Director, VP of Technology, and most recently Chief Information Security Officer
- Patent awarded as a co-inventor on a system for secure mail processing at correctional facilities
- Adobe Certified ColdFusion Professional*
- CISSP

Last Year I Said:

Passkeys are Coming…

They use asymmetric encryption with public and private keys and are far more secure than passwords, but implementation is a mess right now.

What are Passkeys?

Phishing resistant form of authentication based on asymmetric cryptography and tied to device biometrics or a master password.

Sponsored by the FIDO Alliance and major tech companies.

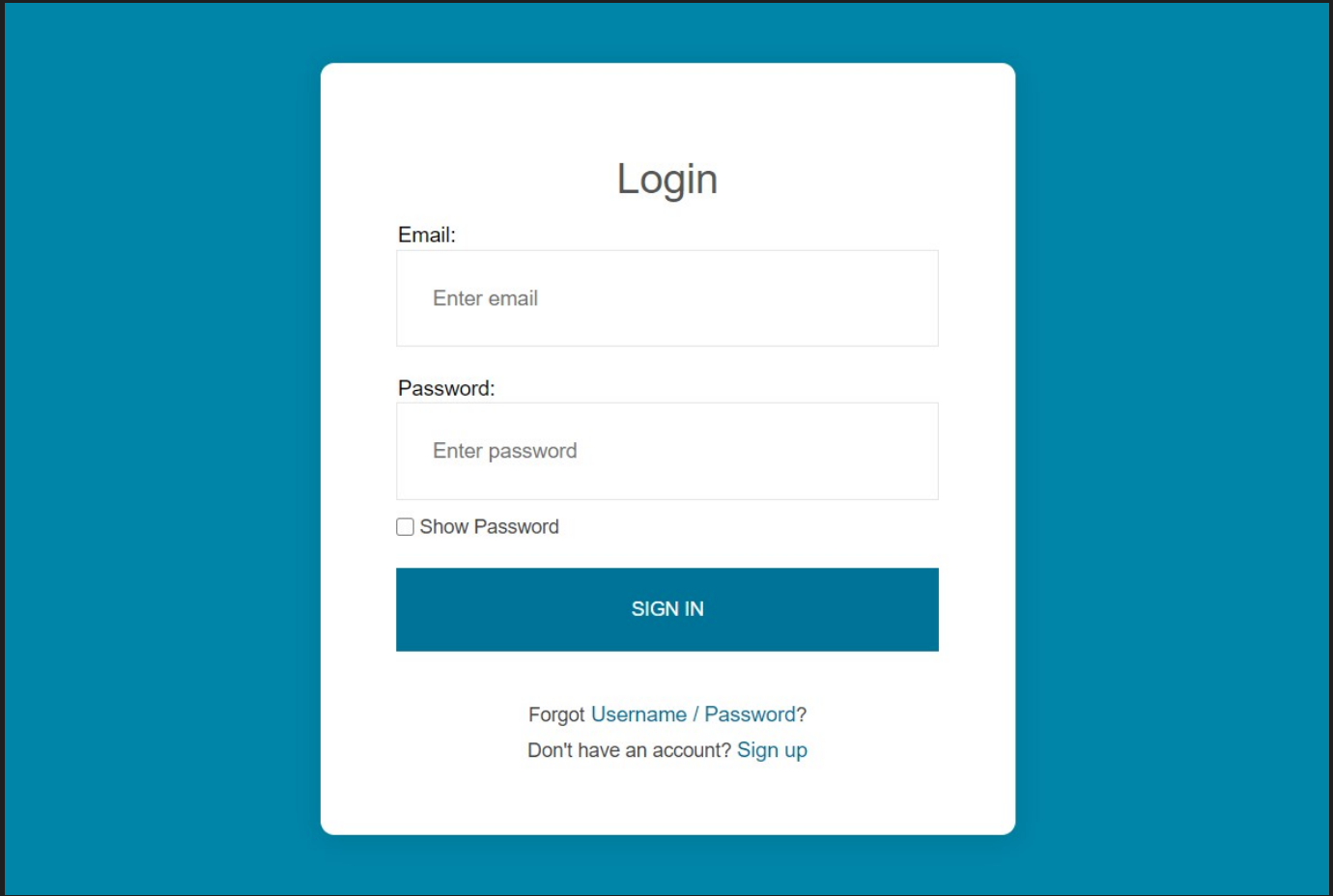
We've all Coded a Login Form

The "old" way:

Identify the user typically with a Username or Email.

Authenticate the user with a password ("something they know").

May be enhanced with an additional factor such as an authenticator (TOTP) code, emailed link, SMS message, etc.

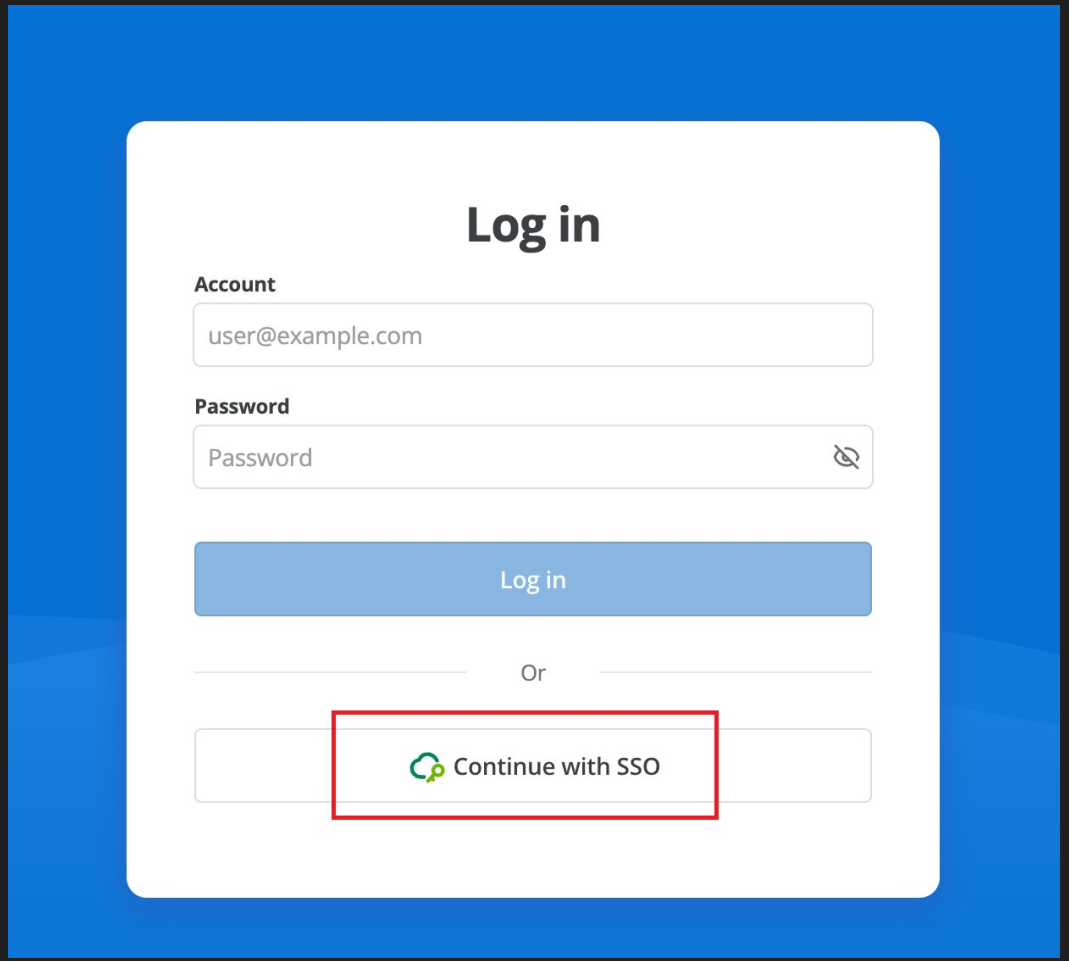


Single Sign-On

We can outsource or supplement authentication to a 3rd-party identity provider such as Microsoft, Google, Apple, Okta, social media platforms, or other SAML 2.0 or OAUTH compatible service.

May still be vulnerable to various attacks depending on the security of the outside platform.

A user's account is now dependent upon their account with the 3rd-party service provider being available and in good standing (e.g. not locked out or disabled).



People Forget Passwords

Memory fades, and people tend to forget their passwords, especially if following advice to use a unique passwords.

Passwords get Reused

People don’t want to remember a lot of passwords, so they reuse the same passwords everywhere. A successful attack on one provider can lead to more breaches.

Passkeys Solve These

Passkeys solve all of these problems with passwords. They cannot be forgotten, reused, phished, or shared, and are not stored on the server to steal or abuse.

Server-Side Storage

A representation of the password must be stored on the server side which can be harder than expected to get right (hash, salt, pepper, iterations, work factors…)

Phishable and Sharable

Passwords, by their nature, can be shared among multiple people or phished by an attacker through social engineering or fake authentication forms.

Nothing to Remember

Passkeys are generated by the user agent as a public-private key pair, and the private key is retained in secure storage or hardware.

Passkeys are Unique

A new passkey gets generated for each registration, so there is no possibility of reusing the same passkey in multiple places.

Still Not Perfect

Implementation is still early, users don't have awareness and experience, and some have concerns about portability, vendor lock-in, and control over private keys.

Asymmetric Cryptography

The server only gets the public key, so we're not storing anything sensitive in the database.

Not Phishable or Sharable

Passkey private keys cannot be exported by design, so an attacker cannot "trick" someone into giving them their Passkey.

Terminology

- RP = Relying Party (i.e. your server/website)
  Expressed as root domain of credential (can include subdomain)
- Browser = Exposes JS interface for Authenticator
- Authenticator = Software handling create/get
- CredentialID = Unique ID of a passkey

Asymmetric Cryptography

- Encryption that uses different keys to encrypt and then decrypt the same data (encrypt with one key, decrypt with the other)
- Known as a “key pair” consisting of a “public” key which can be shared (not secret) and a “private” key which should only ever be known to the user
- Basis of SSL/TLS certificates
- Much slower than symmetric cryptography, but fine for small chunks of data (vs. large documents)

Requirements

- Requires HTTPS – Passkeys only work in a secure context and require secure transport
- Localhost domains are treated as secure for this purpose (ex: sampleenterprises.localhost)
- Localhost IP is NOT considered secure (127.0.0.1)
- A User Agent (browser) that supports WebAuthn

How Do Passkeys Work?

Registration Ceremony

- RP (server) creates registration challenge upon request
- JavaScript calls credentials API: navigator.credentials.create()
- Authenticator generates key pair + returns attestation to server
- Server verifies attestation and stores credential ID + public key

Authentication Ceremony

- RP (server) creates authentication challenge upon request
- JavaScript calls credentials API: navigator.credentials.get()
- Authenticator signs challenge with private key and returns to Server
- Server verifies signature with stored public key; logs in user

Registration Ceremony

Request Registration Challenge

```
// Step 1: Get registration options from server
const optionsResponse = await fetch('passkey-api.cfm', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        action: 'getRegistrationOptions',
        username: '#encodeForJavaScript(userData.username)#',
        displayName: '#encodeForJavaScript(userData.fullName)#'
    })
});

const optionsData = await optionsResponse.json();
```

Server JSON Challenge Response


```
{
  "message": "",
  "success": true,
  "options": {
    "rp": { "name": "Sample Enterprises", "id": "sampleenterprises.localhost" },
    "user": {
        "id": "UZSL85T9AFC",
        "name": "alice",
        "displayName": "Alice Abernathy"
    },
    "challenge": "16999F6BAA7C7084433C935C4E175EA0",
    "pubKeyCredParams": [{ "type": "public-key", "alg": -7 }, { "type": "public-key", "alg": -257}],
    "authenticatorSelection": {
        "authenticatorAttachment": "platform",
        "residentKey": "preferred",
        "userVerification": "preferred"
    },
    "timeout": 60000,
    "attestation": "none"
    }
}
```


ES256 (Elliptic Curve, alg = -7)
Fast, compact, widely supported

RS256 (RSA, alg = -257)
Larger, slower, but broadly compatible

Platform vs Cross-Platform

- “platform” (used in demo)
  - Built-in (FaceID, TouchID, Windows Hello, 1Password, etc.)
  - Bound to device or ecosystem, convenient
- “cross-platform”
  - External keys (YubiKey, security key via USB, NFC, Bluetooth, etc.)
  - Portable across devices
- Best practice: support both for flexibility

Browser Calls Authenticator API

```
// Step 2: Create the credential using WebAuthn API
const credentialCreationOptions = {
    publicKey: {
        ...optionsData.options,
        challenge: base64UrlDecode(optionsData.challenge),
        user: {
            ...optionsData.options.user,
            id: base64UrlDecode(optionsData.options.user.id)
        }
    }
};

const credential = await navigator.credentials.create(credentialCreationOptions);
```

Register Credential With Server

```
// Step 3: Send the credential to the server for registration
const registrationData = {
    action: 'registerPasskey',
    deviceName: document.getElementById('deviceName').value,
    userAgent: navigator.userAgent,
    credential: {
        id: credential.id,
        rawId: base64UrlEncode(credential.rawId),
        response: {
            clientDataJSON: base64UrlEncode(credential.response.clientDataJSON),
            attestationObject: base64UrlEncode(credential.response.attestationObject)
        },
        type: credential.type
    }
};

const registerResponse = await fetch('passkey-api.cfm', {
    method: 'POST', headers: {'Content-Type': 'application/json'},
    body: JSON.stringify(registrationData)
});

const registerData = await registerResponse.json();
```

Credential Registration Response


```
{
    "message": "Passkey registered successfully",
    "passkeyID": 19,
    "success": true
}
```

Authentication Ceremony

Request Authentication Challenge

```
// Get authentication options
const authOptionsResponse = await fetch('passkey-api.cfm', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify({
        action: 'getAuthenticationOptions',
        username: username
    })
});

const authOptionsData = await authOptionsResponse.json();
```

Server JSON Challenge Response


```
{
  "challenge": "16999F6BAA7C7084433C935C4E175EA0",
  "rpId": "sampleenterprises.localhost",
  "allowCredentials": [
    { "id": "credential123", "type": "public-key" }
  ],
  "timeout": 60000,
  "userVerification": "preferred"
}
```

Browser Calls Authenticator API

```
const credentialRequestOptions = {
    publicKey: {
        ...authOptionsData.options,
        challenge: base64UrlDecode(authOptionsData.challenge),
        allowCredentials: authOptionsData.options.allowCredentials?.map(cred => ({
            ...cred,
            id: base64UrlDecode(cred.id)
        })) || []
    }
};

const authCredential = await navigator.credentials.get(credentialRequestOptions);
```

Verify Credential With Server

```
const authData = {
    action: 'authenticatePasskey',
    username: username,
    credential: {
        id: authCredential.id,
        rawId: base64UrlEncode(authCredential.rawId),
        response: {
            authenticatorData: base64UrlEncode(authCredential.response.authenticatorData),
            clientDataJSON: base64UrlEncode(authCredential.response.clientDataJSON),
            signature: base64UrlEncode(authCredential.response.signature)
        },
        type: authCredential.type
    }
};
const authResponse = await fetch('passkey-api.cfm', {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json',
    },
    body: JSON.stringify(authData)
});
const authResult = await authResponse.json();
if (!authResult.success) {
    throw new Error(authResult.message || 'Passkey authentication failed');
}
```

ColdFusion Passkey Implementation

Base64URL Encoding

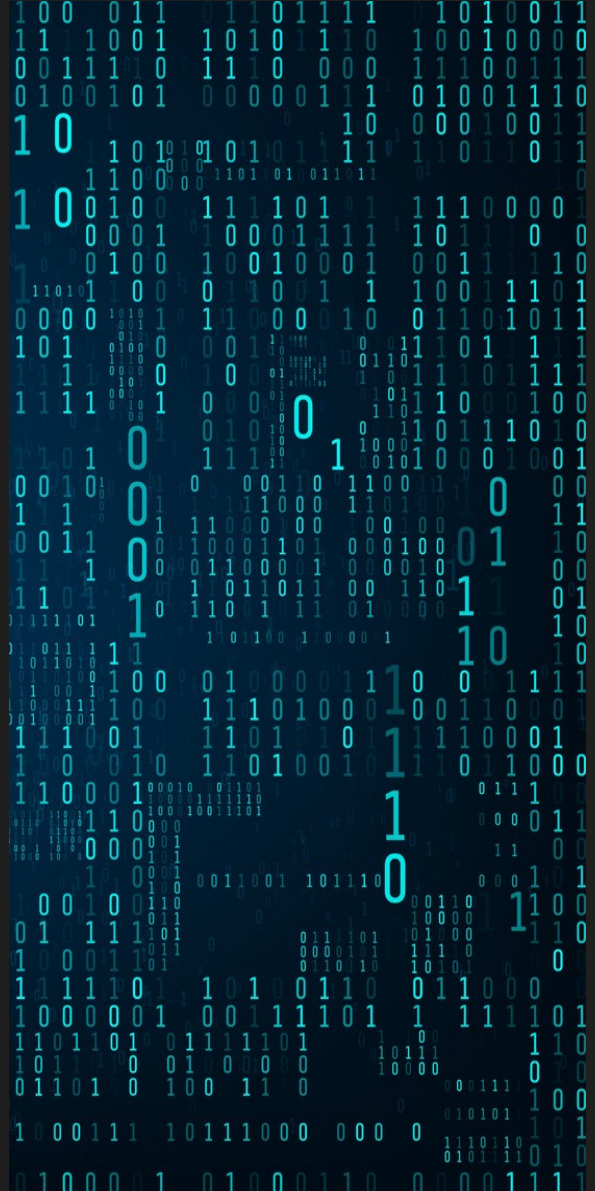
Modified base64 encoding format that is “URL safe”; swaps out certain characters; no built-in function support in ColdFusion, or even JavaScript for that matter.

Asymmetric Encryption

Passkeys use asymmetric cryptography with public and private key pairs. No native support within ColdFusion. Have to drop into Java.

Not a Simple Task

But it can be done.



CBOR/COSE Encoding

Concise Binary Object Representation
CBOR Object Signing and Encryption

Likely new for web developers, mainly used in low-power IoT devices.

External Libraries

Java classes available to help such as the WebAuthn library or Yubico Passkey library.

ColdFusion 2026

Native passkey functions have been announced which will make this a LOT easier!

Demo and Code

Device-Bound Session Credentials

- Traditional: cookies store session; portable, can be stolen
- Device-bound: session token tied to device/browser profile
- Benefits:
  - Stolen tokens useless elsewhere
  - Mitigates cookie harvesting
  - Stronger session integrity
- Still emerging, but promising for future web security

Transitioning to Passkeys

- Start with MFA: passwords + TOTP or passkeys
- Add “Add a passkey” button in settings
- Educate users: “easier and safer login”
- Gradually shift login flow to highlight passkeys
- Offer to register a passkey after login if they don’t have one
- Ultimately force passkey registration as a condition of use

How Many Passkeys?

- Recommend at least two (e.g. desktop + phone)
- No need to limit, but if you want to limit, set it high (10+)
- More passkeys = More resilience against loss
- Do not limit to one; makes recovery more difficult and use across different platforms more difficult or impossible leading to lower adoption

Do We Still Need Passwords?

- Keep support for passwords for now (users still need education)
- Prefer or force passkey registration for new accounts
- Fall back to password if a passkey isn’t available

Recovery Strategy

- Support generation, storage, and use of recovery codes
  - Generate 10 single-use, eight digit codes for the user to keep
  - Codes can be used in place of a passkey or password when needed
  - Invalidate each code after use and trigger an alert to the user
  - Allow code refresh any time as needed through Profile
- Alternate fallback to SMS or email code/link
- Ultimately fall back to support with manual verification and reset

Adoption Roadmap

- Phase 1: Add passkeys alongside passwords
- Phase 2: Passwordless opt-in
- Phase 3: Passkey-first experience
- Phase 4: Optional full passwordless accounts

The Future of Authentication

- Passkeys + device-bound sessions = stronger end-to-end authentication and session protection
- Reduced reliance on cookies & passwords
- Ecosystem improving: browser + OS support is rapidly maturing; password managers now support
- ColdFusion apps can adopt now to stay competitive and secure; CF2026 will include support natively!

AI Demo?

Giveaway!

Q&A

Thanks!

Contact Info:

leviathan@darktech.org
www.darktech.org

www.linkedin.com/in/justinscott