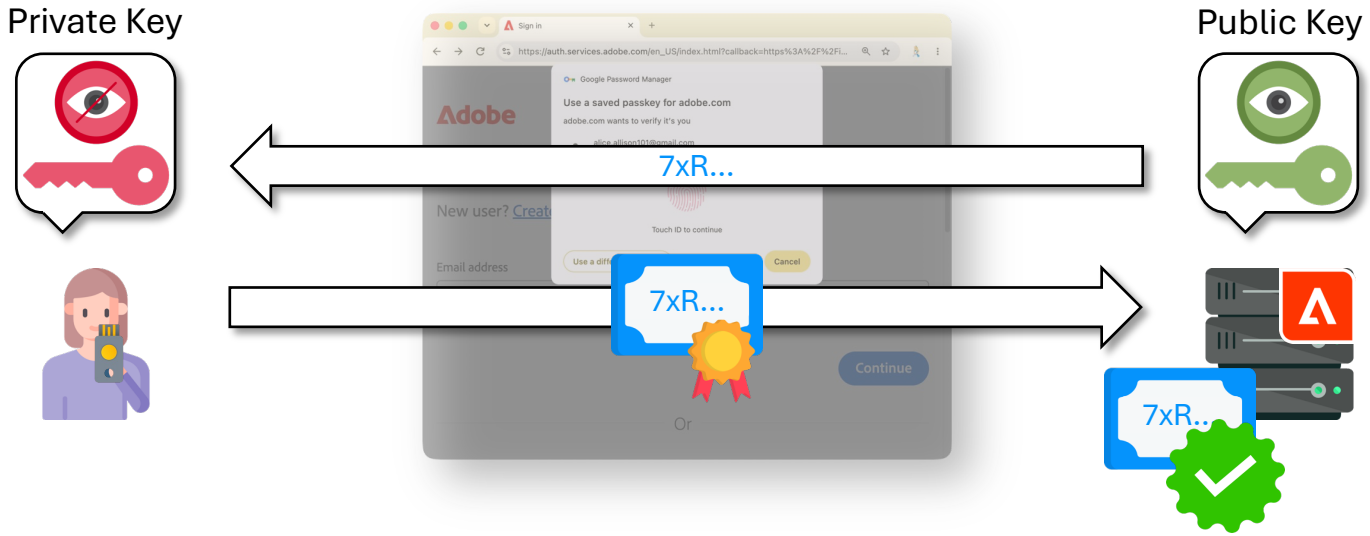


The State of Passkeys: Studying the Adoption and Security of Passkeys on the Web

Louis Jannett[1], Andreas Mayer[2], Maximilian Westers[2],
Vladislav Mladenov[1], Christian Mainka[3], Jörg Schwenk[1]

[1] Ruhr University Bochum, [2] Heilbronn University of Applied Sciences, [3] University of Wuppertal

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

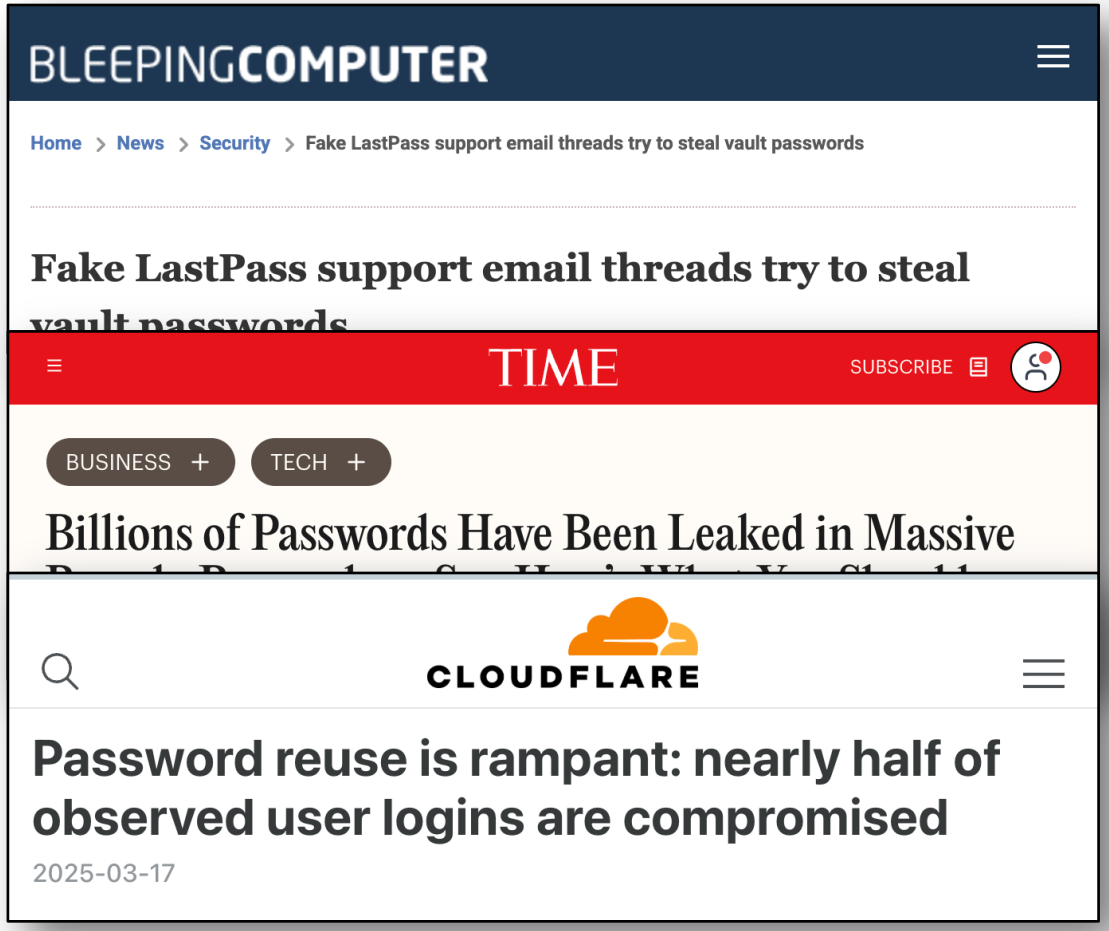
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

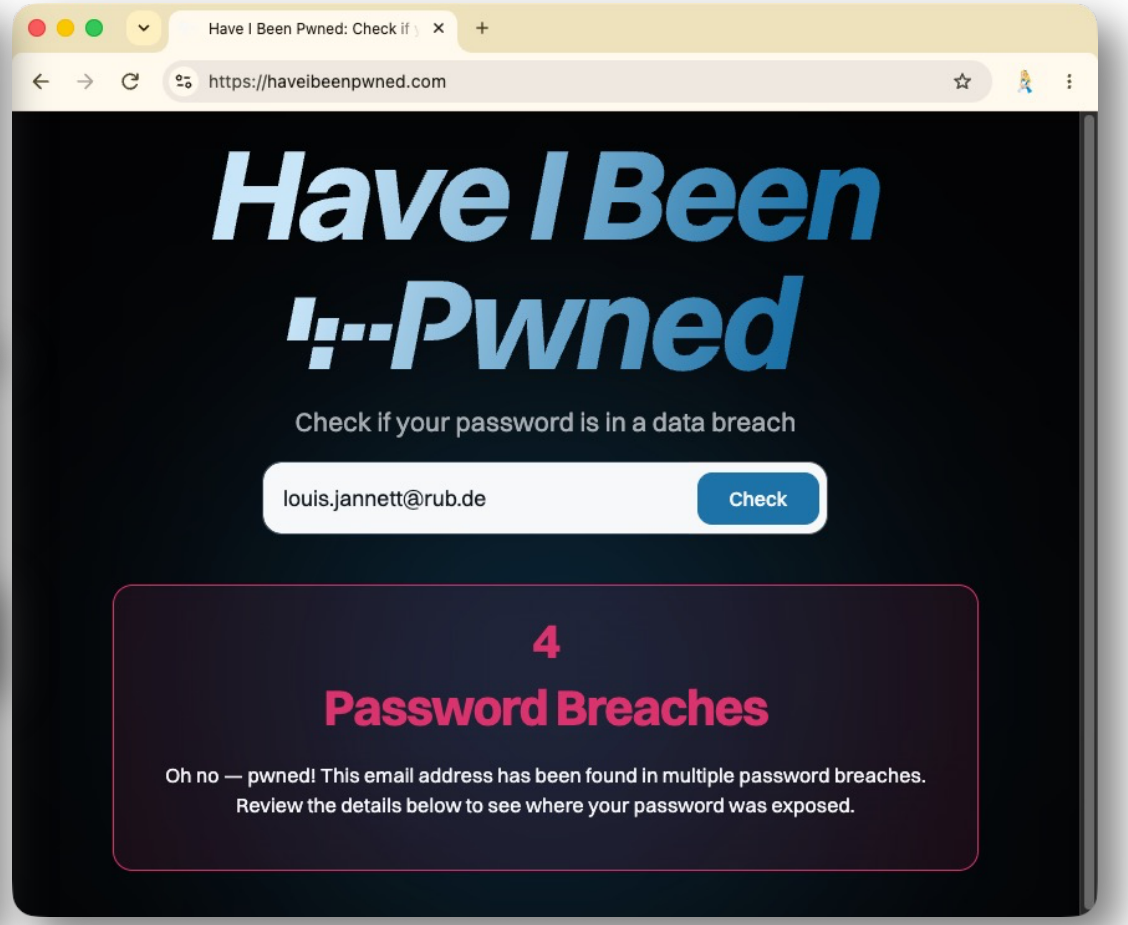
RUHR UNIVERSITÄT BOCHUM RUB

HHN HEILBRONN UNIVERSITY OF APPLIED SCIENCES

BERGISCHE UNIVERSITÄT WUPPERTAL

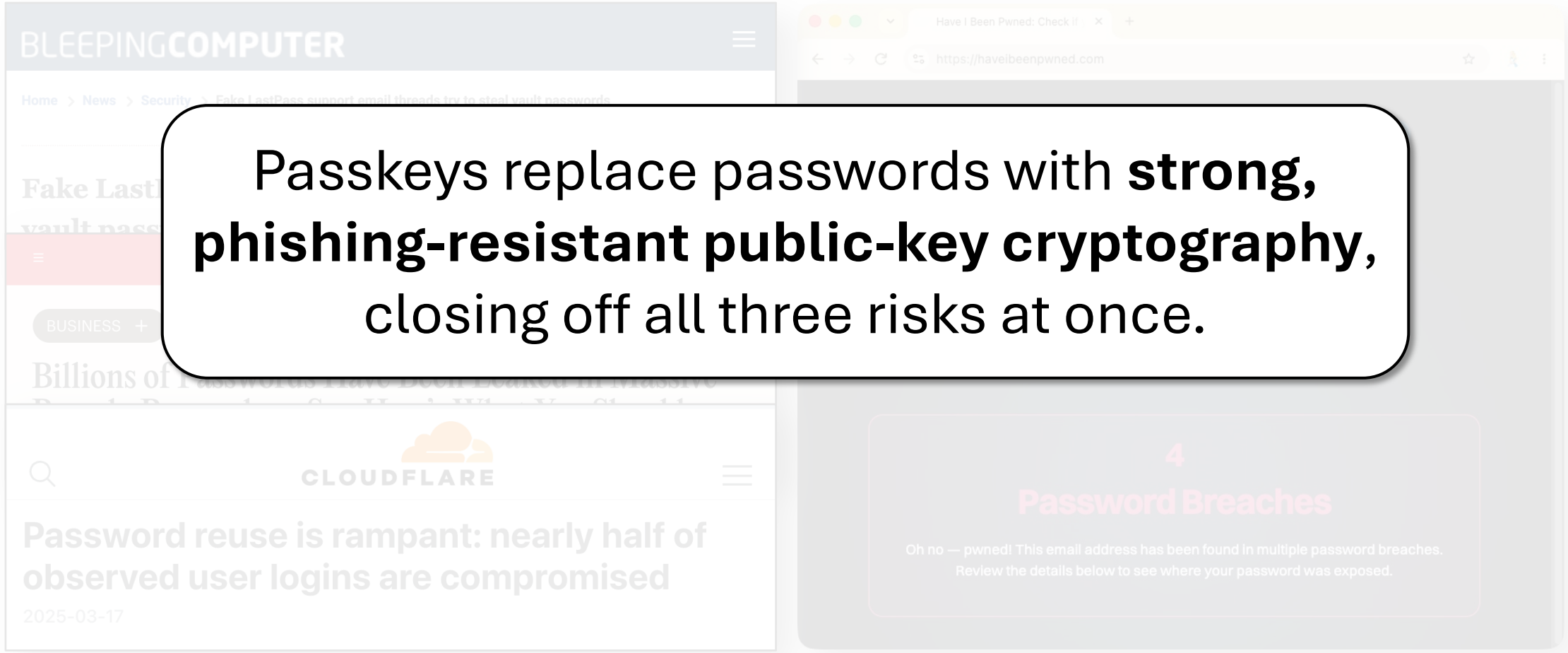
"Hackers don't break in – they sign-in"[1]





[1] https://www.microsoft.com/en-us/security/business/solutions/passwordless-authentication

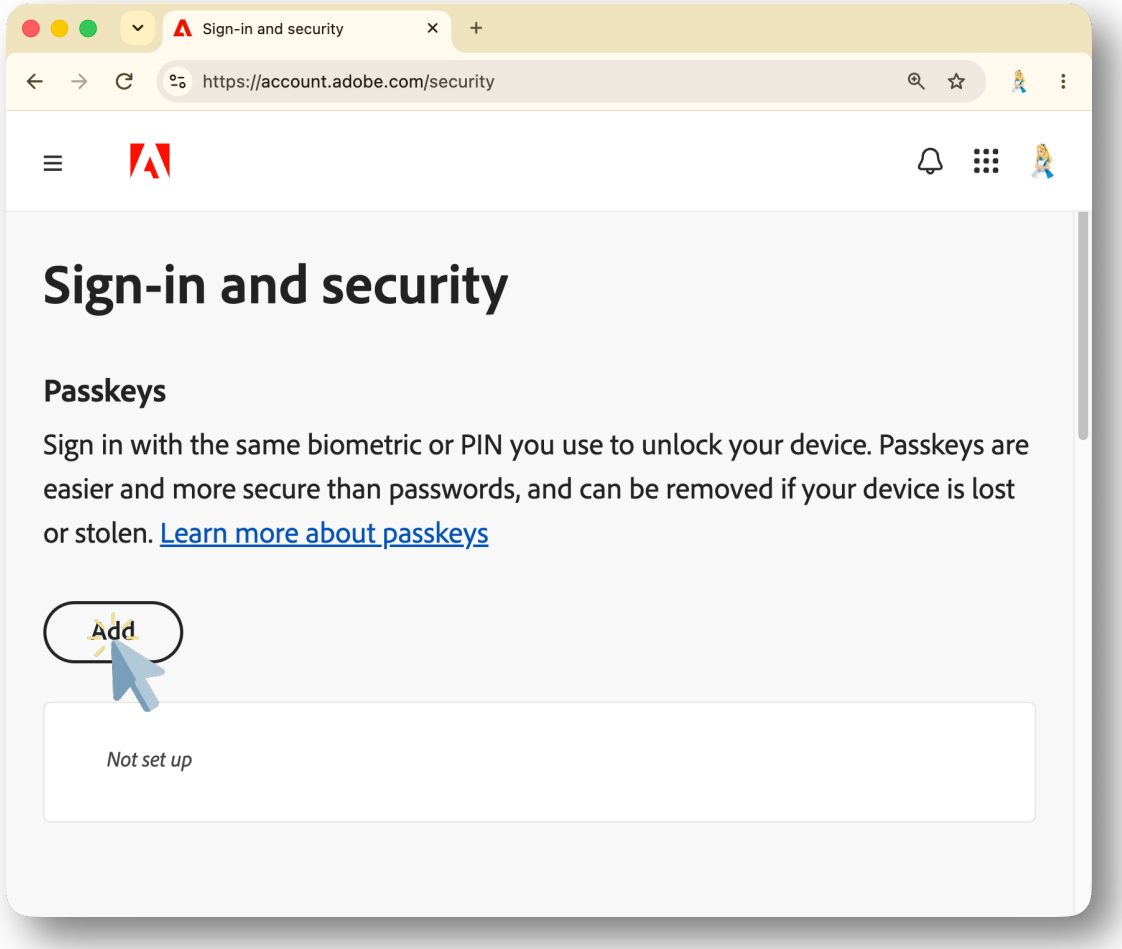
"Hackers don't break in – they sign-in"[1]

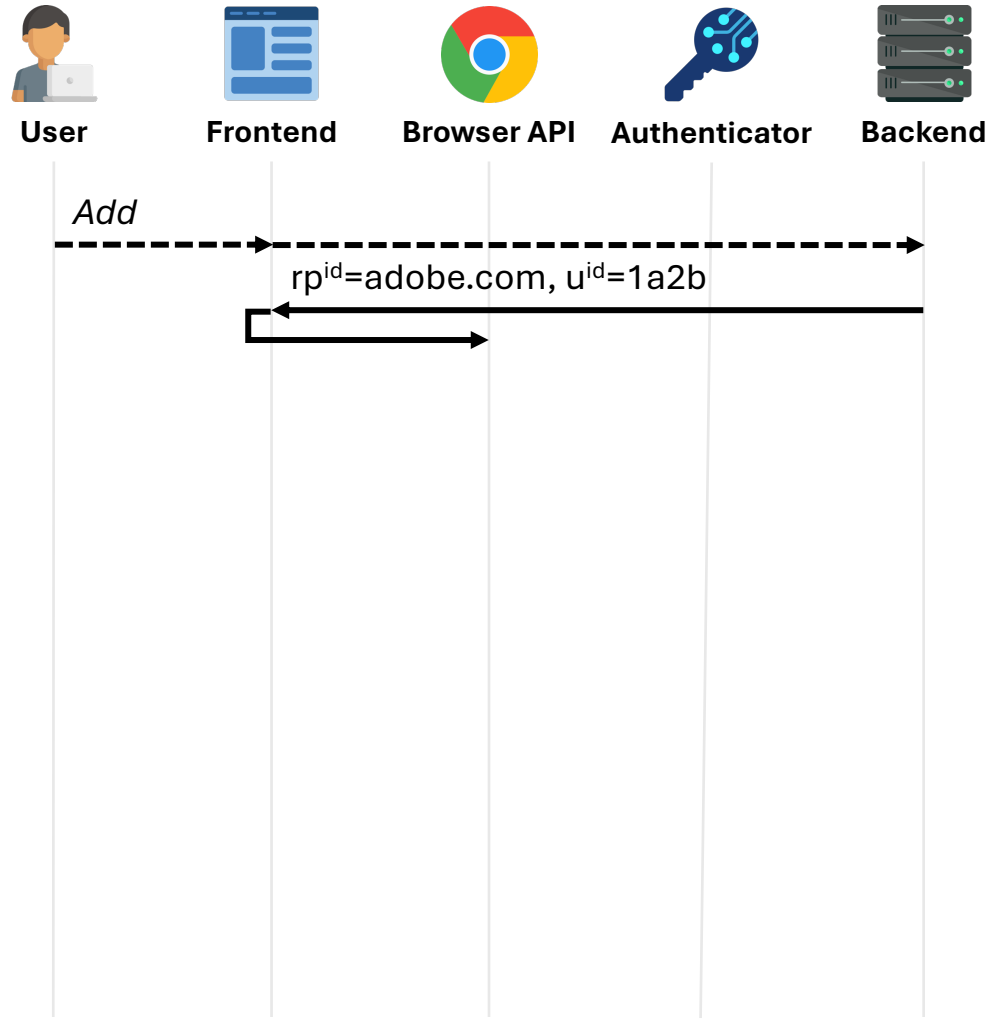


Passkeys replace passwords with **strong, phishing-resistant public-key cryptography**, closing off all three risks at once.

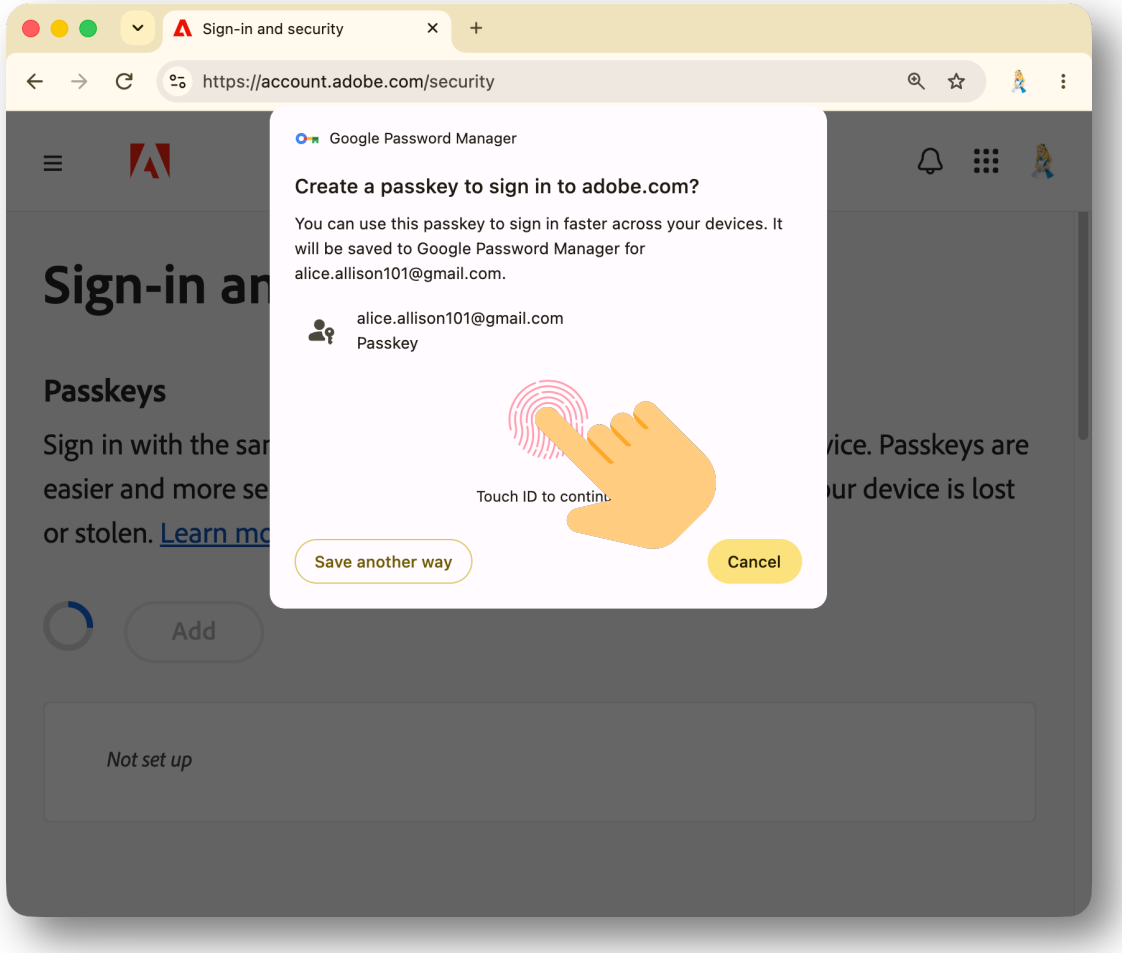
[1] https://www.microsoft.com/en-us/security/business/solutions/passwordless-authentication

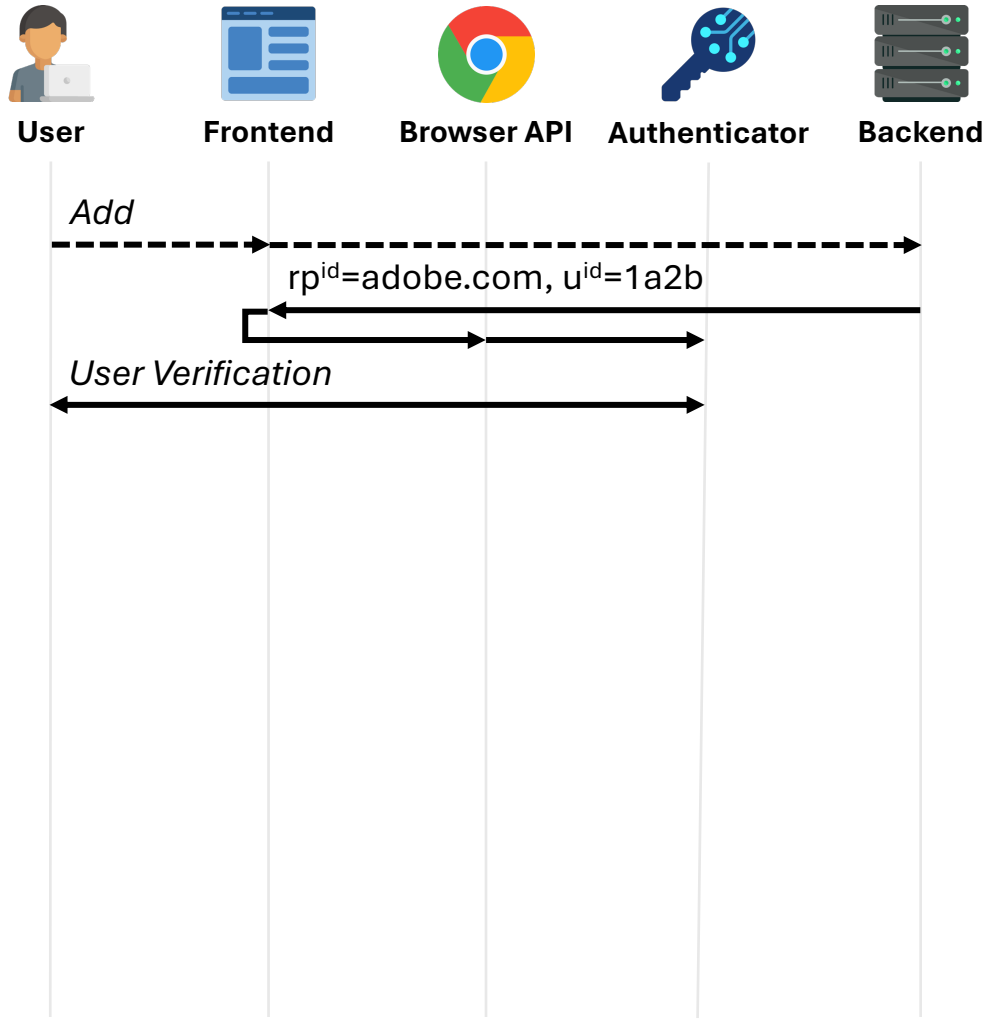
Ceremonies: Registration and Authentication



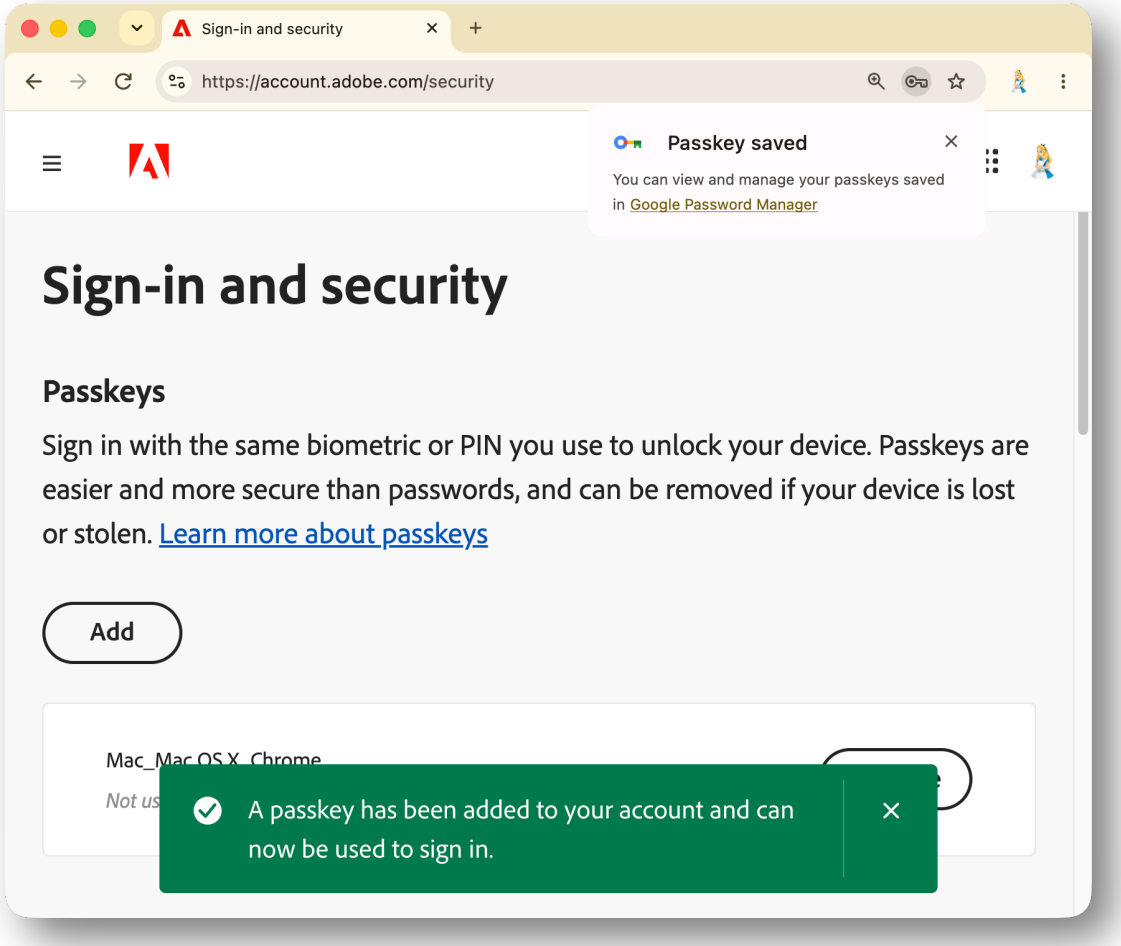


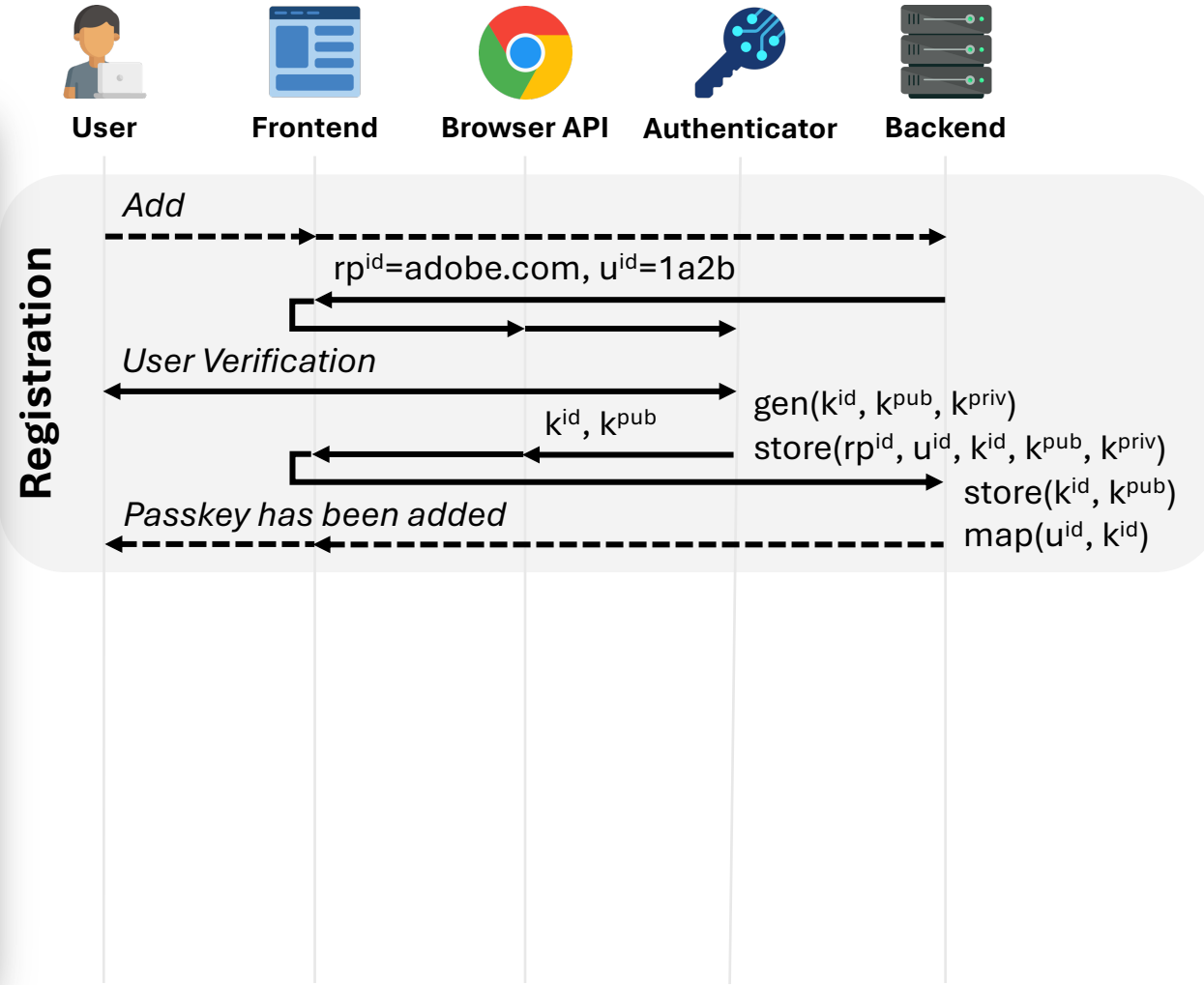
Ceremonies: Registration and Authentication



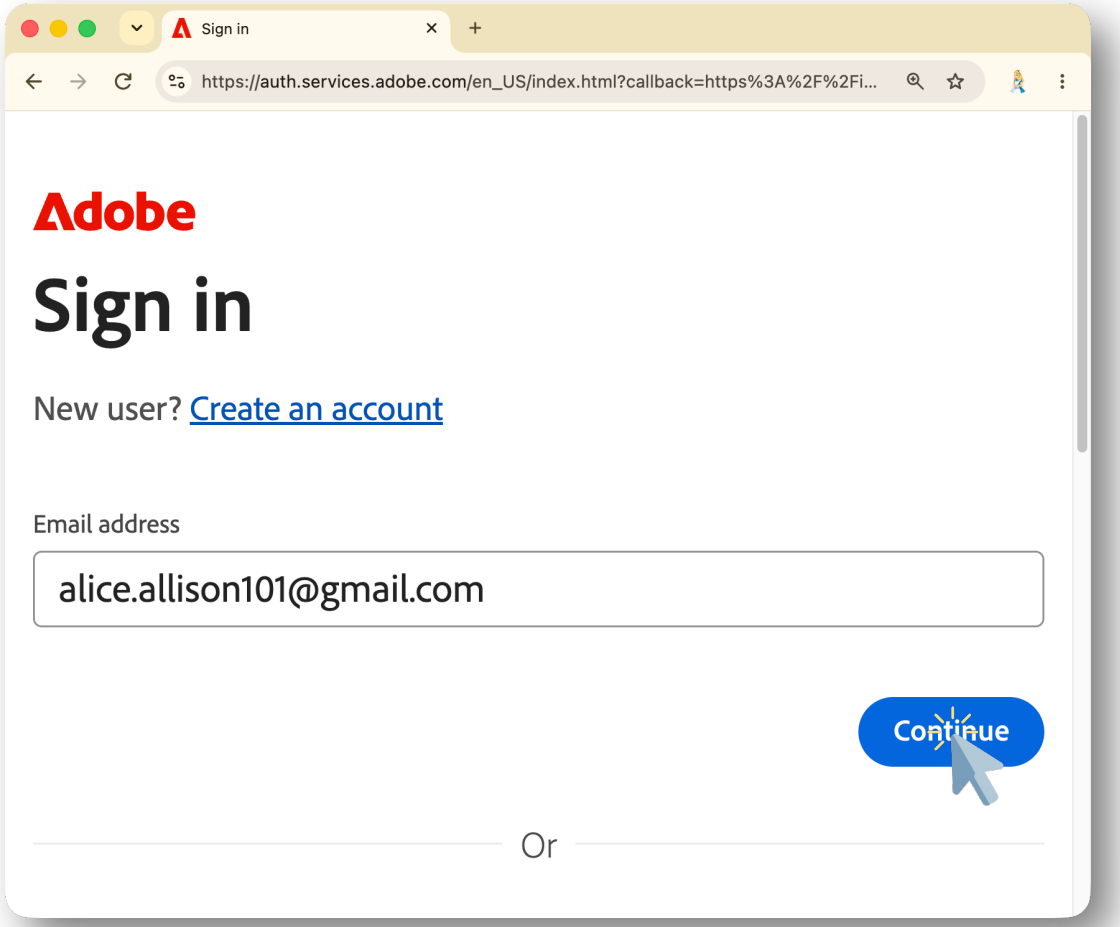


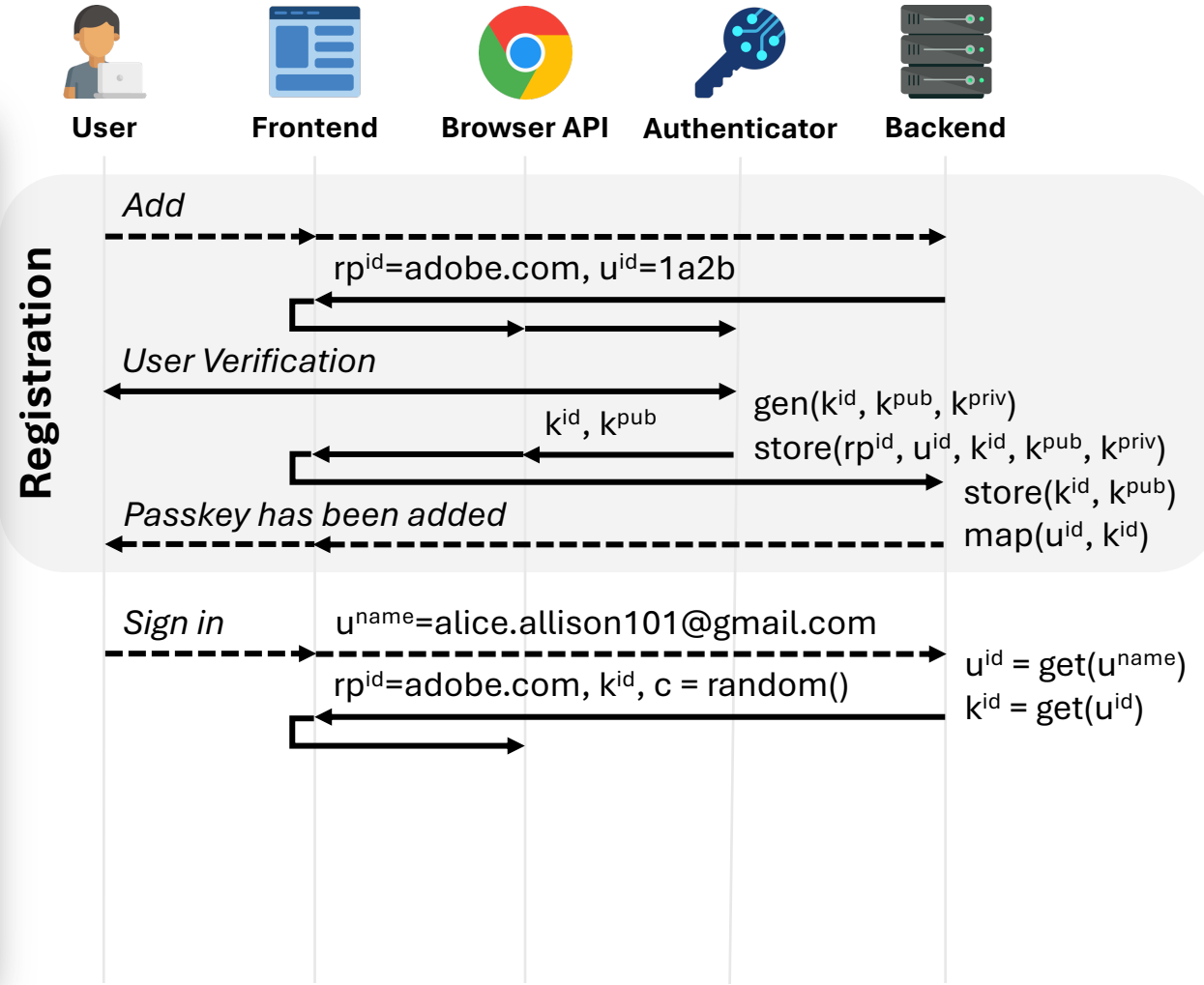
Ceremonies: Registration and Authentication



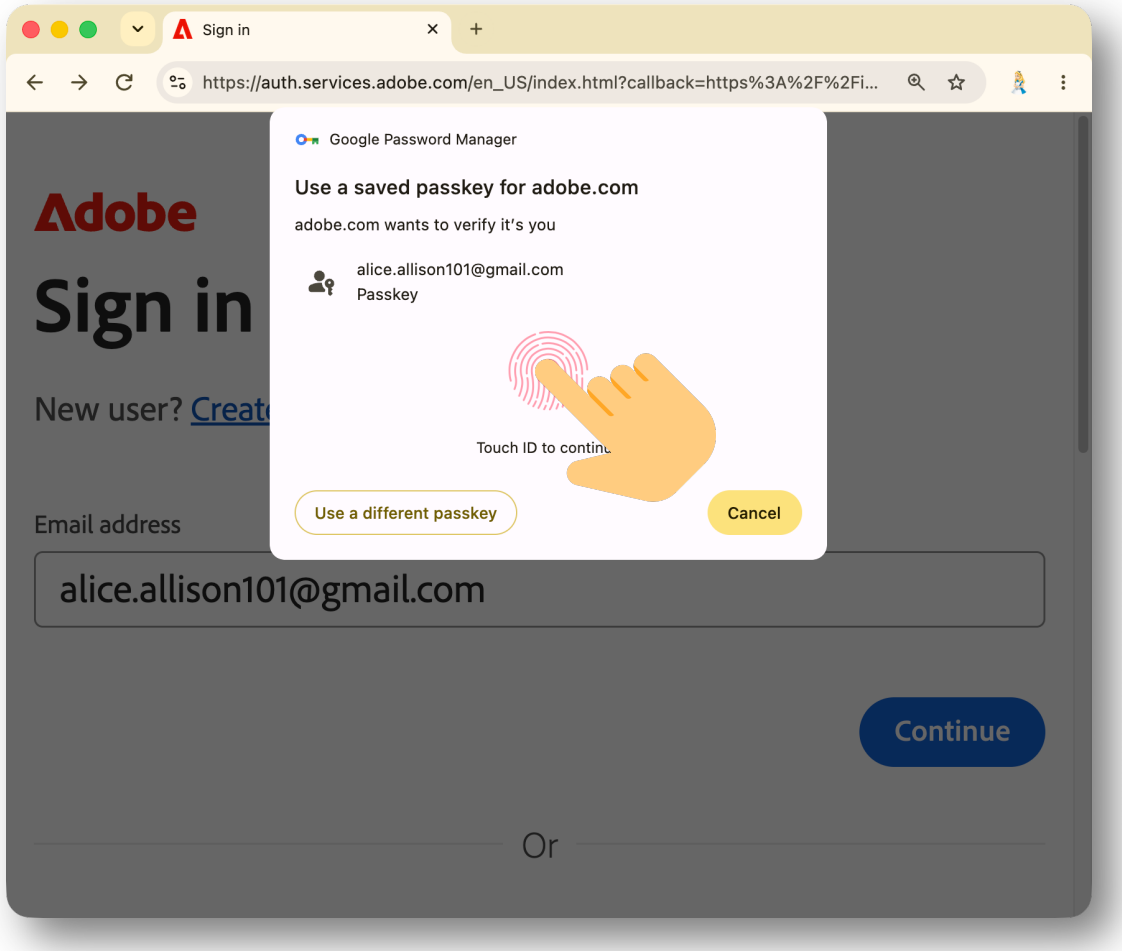


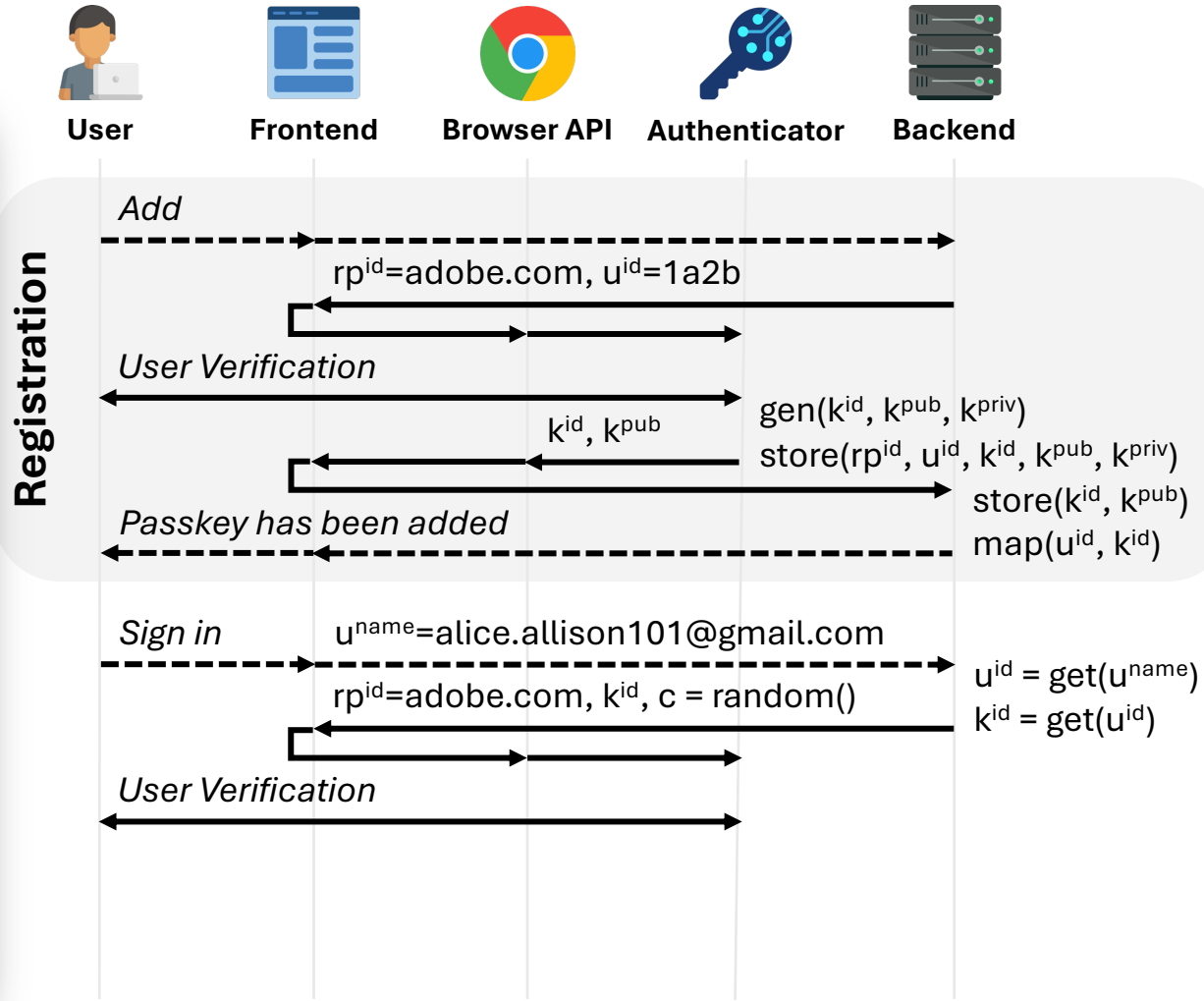
Ceremonies: Registration and Authentication



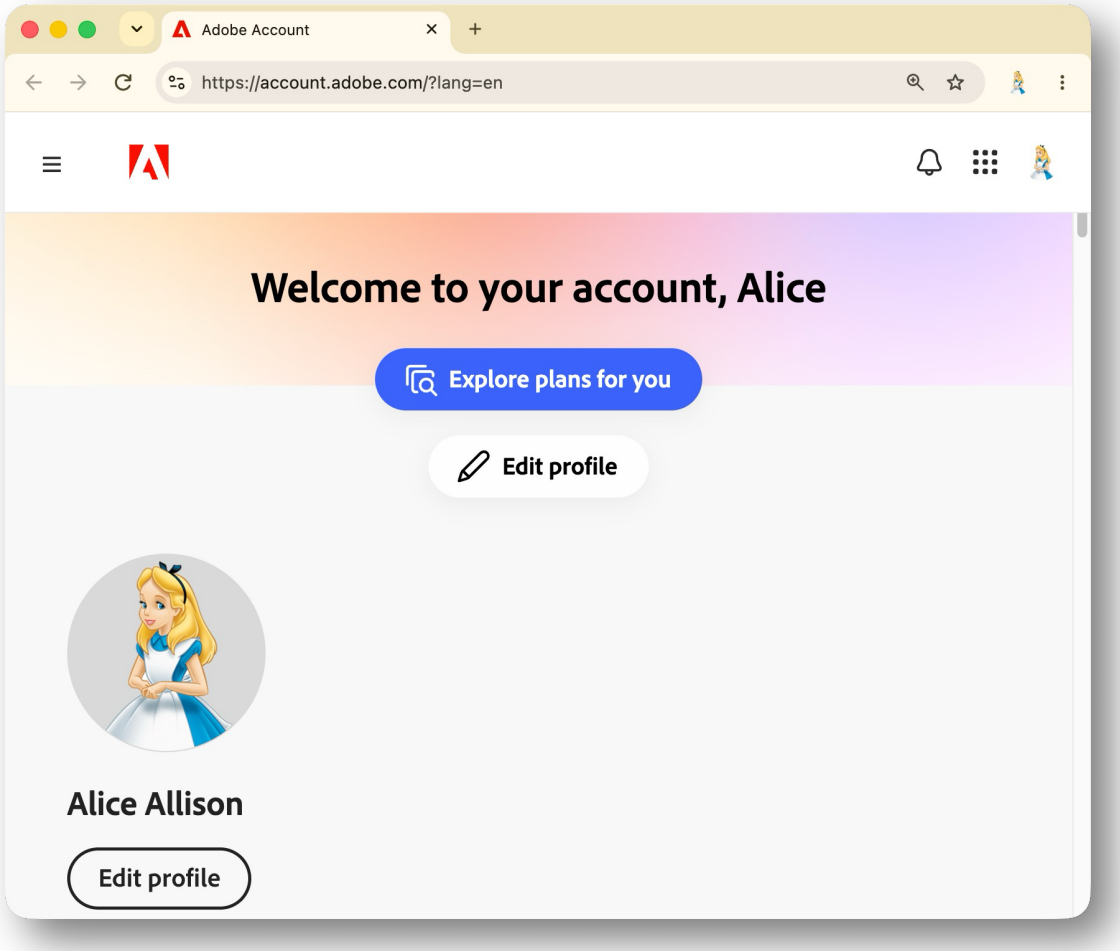


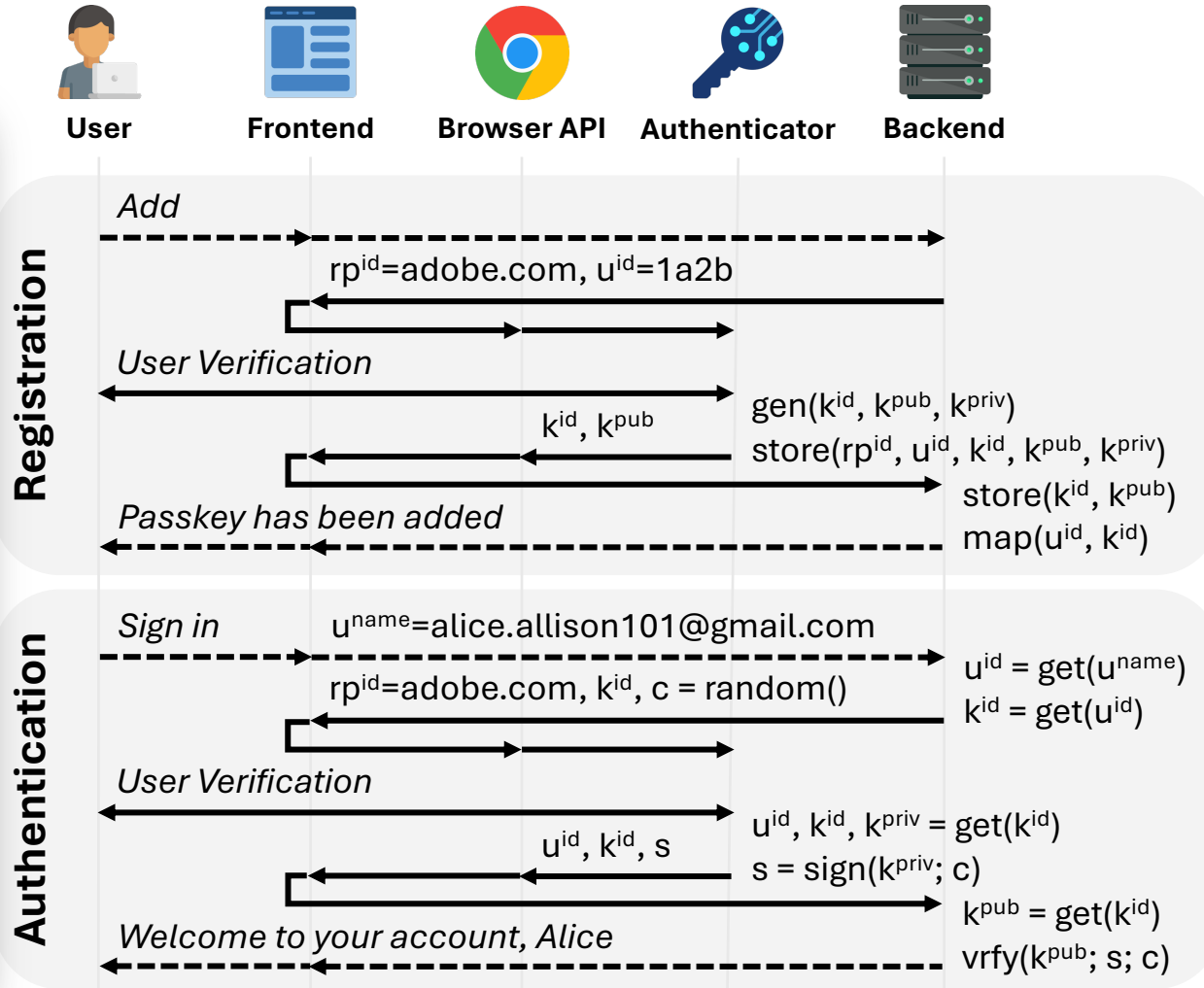
Ceremonies: Registration and Authentication



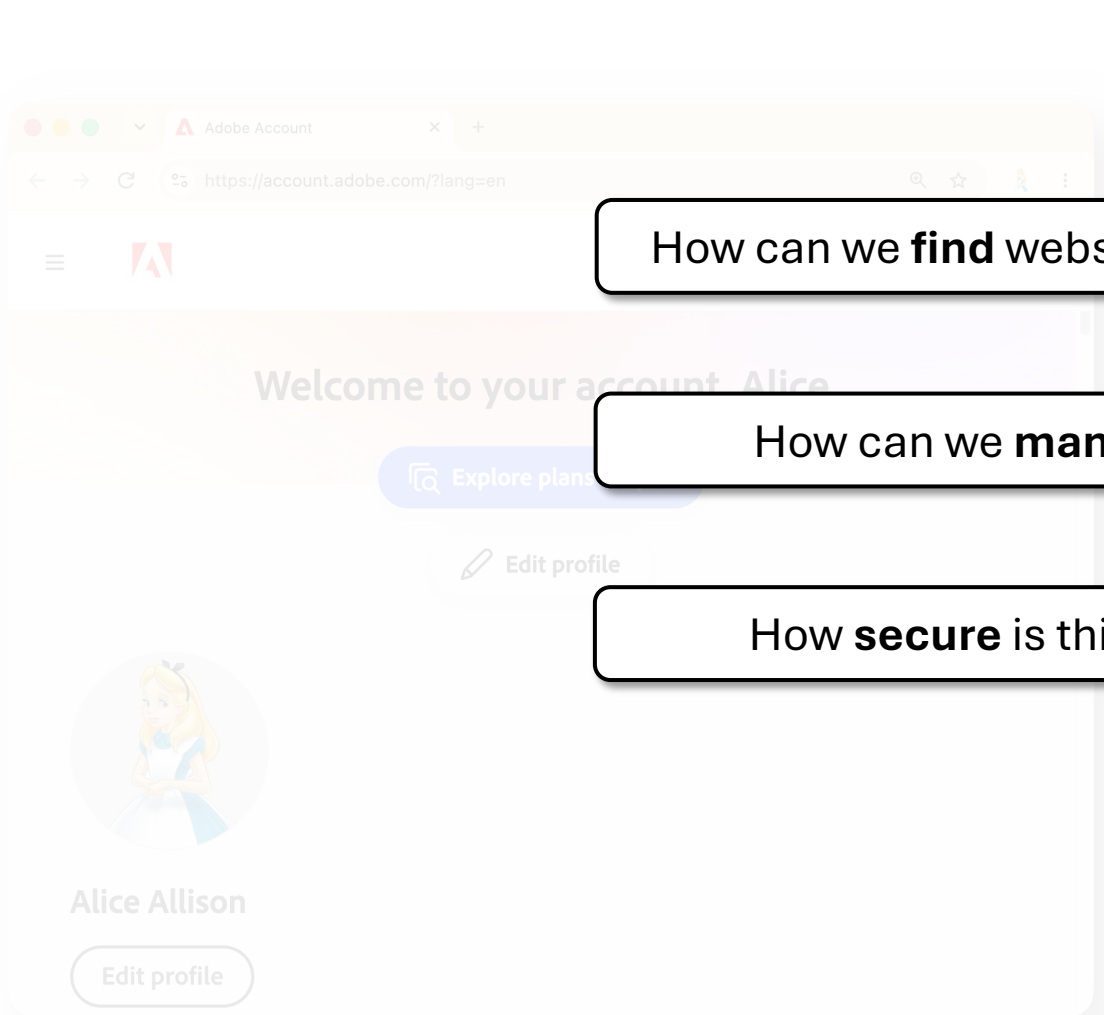


Ceremonies: Registration and Authentication





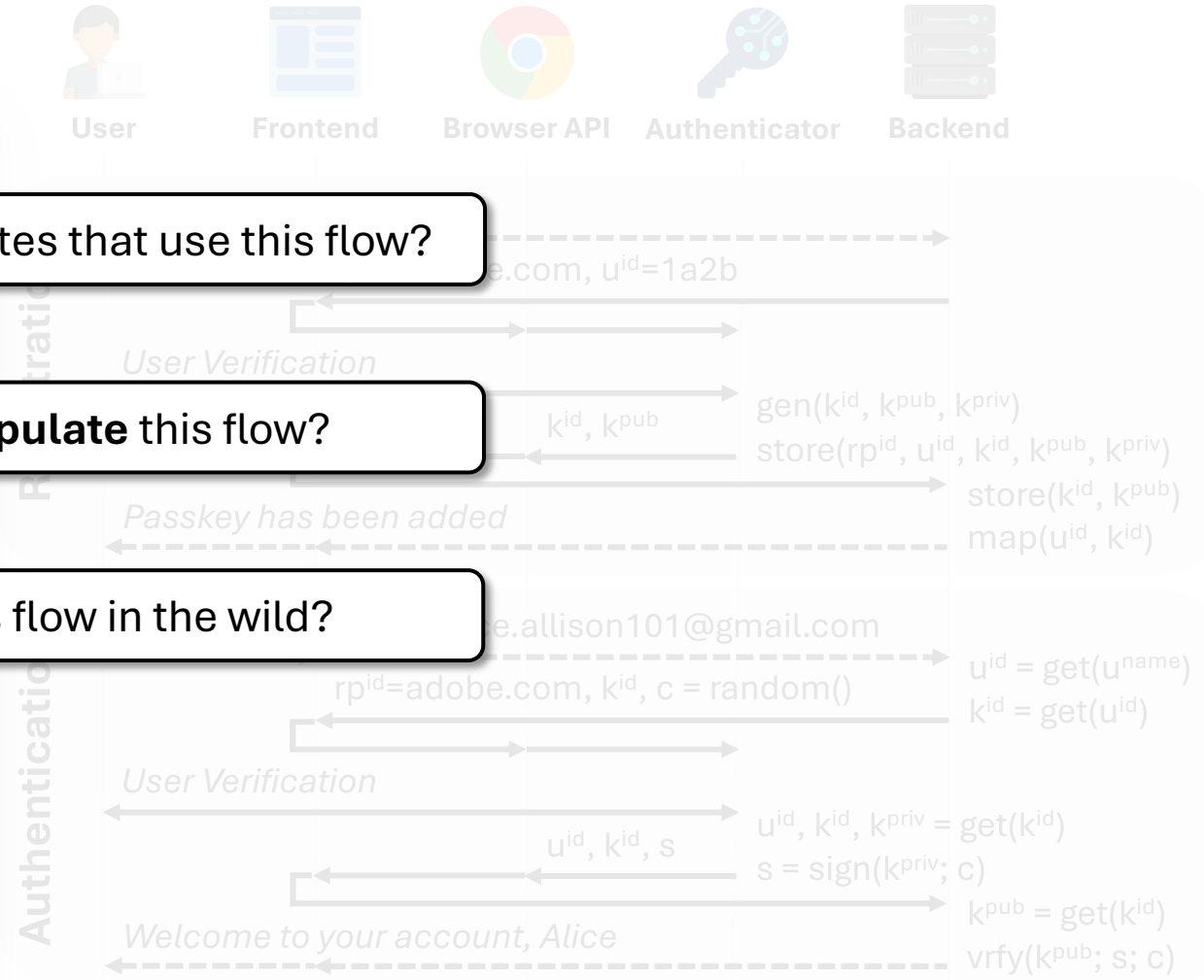
Ceremonies: Registration and Authentication



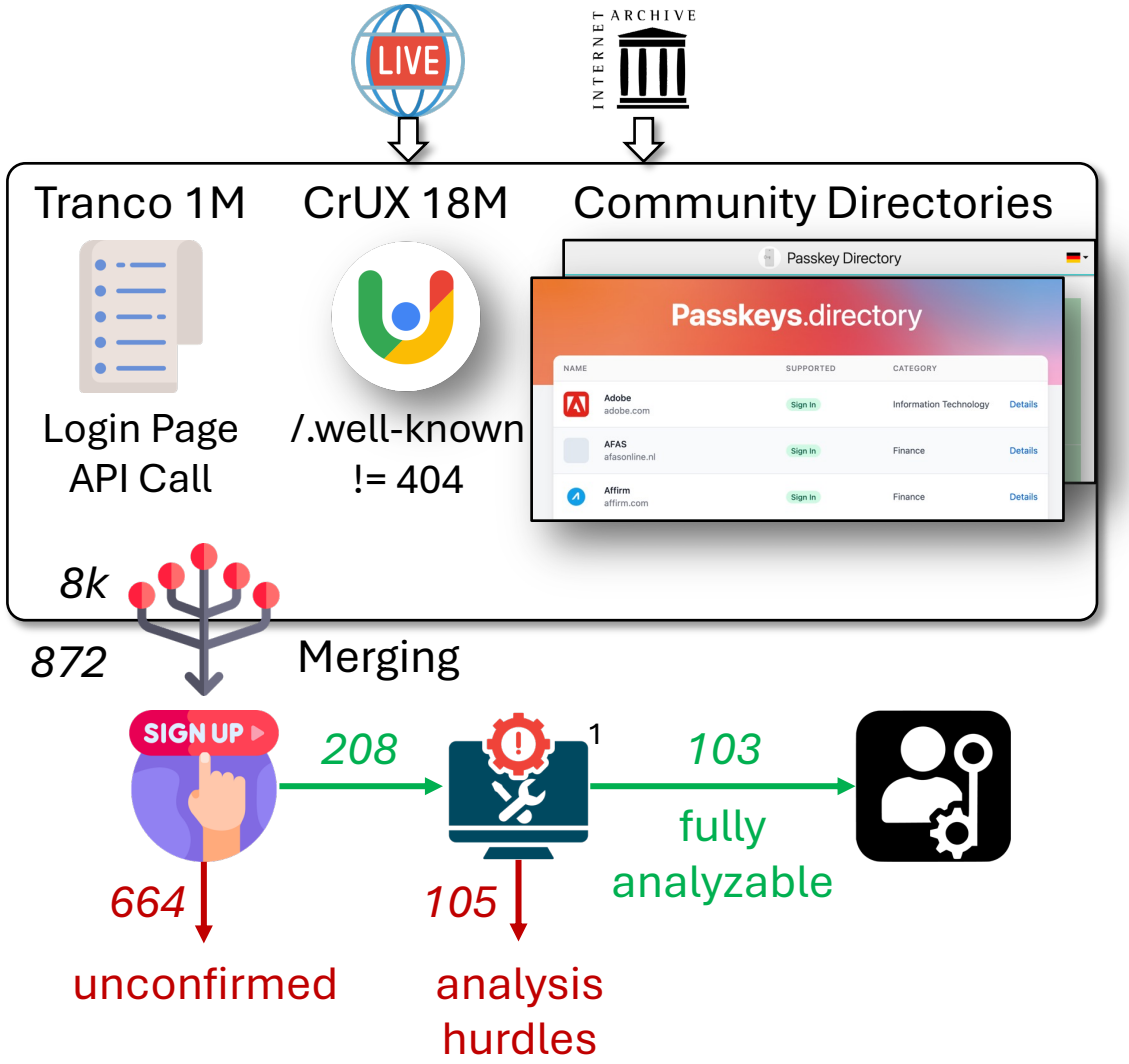
How can we **find** websites that use this flow?

How can we **manipulate** this flow?

How **secure** is this flow in the wild?

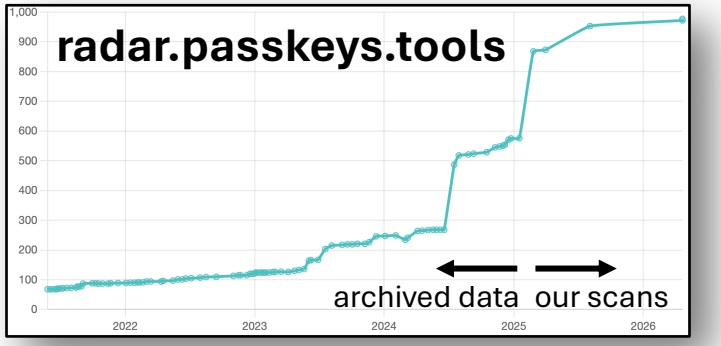


PASSKEYS-RADAR



8k websites before merging

872 websites after merging



664 unconfirmed passkey support

208 confirmed passkey support

105 websites with analysis hurdles

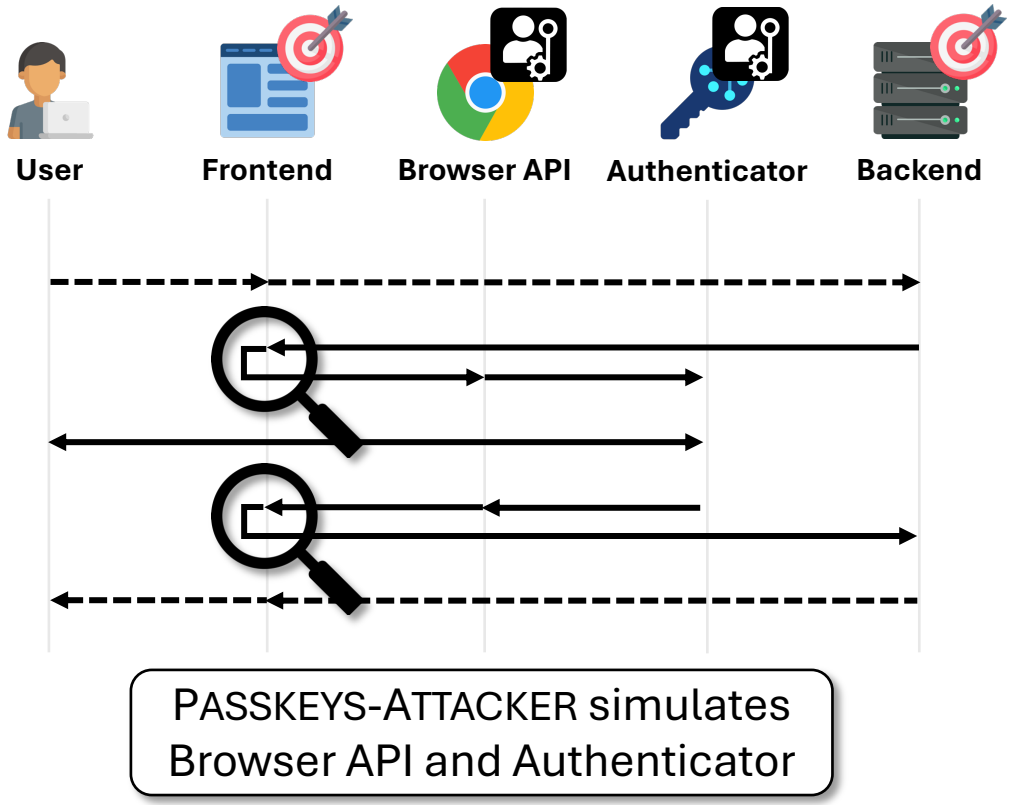
- Confirmations via SMS
- 2FA-only passkey
- No management UI
- No deletion possible
- Single passkey only
- Hardware passkey only

103 websites fully analyzable

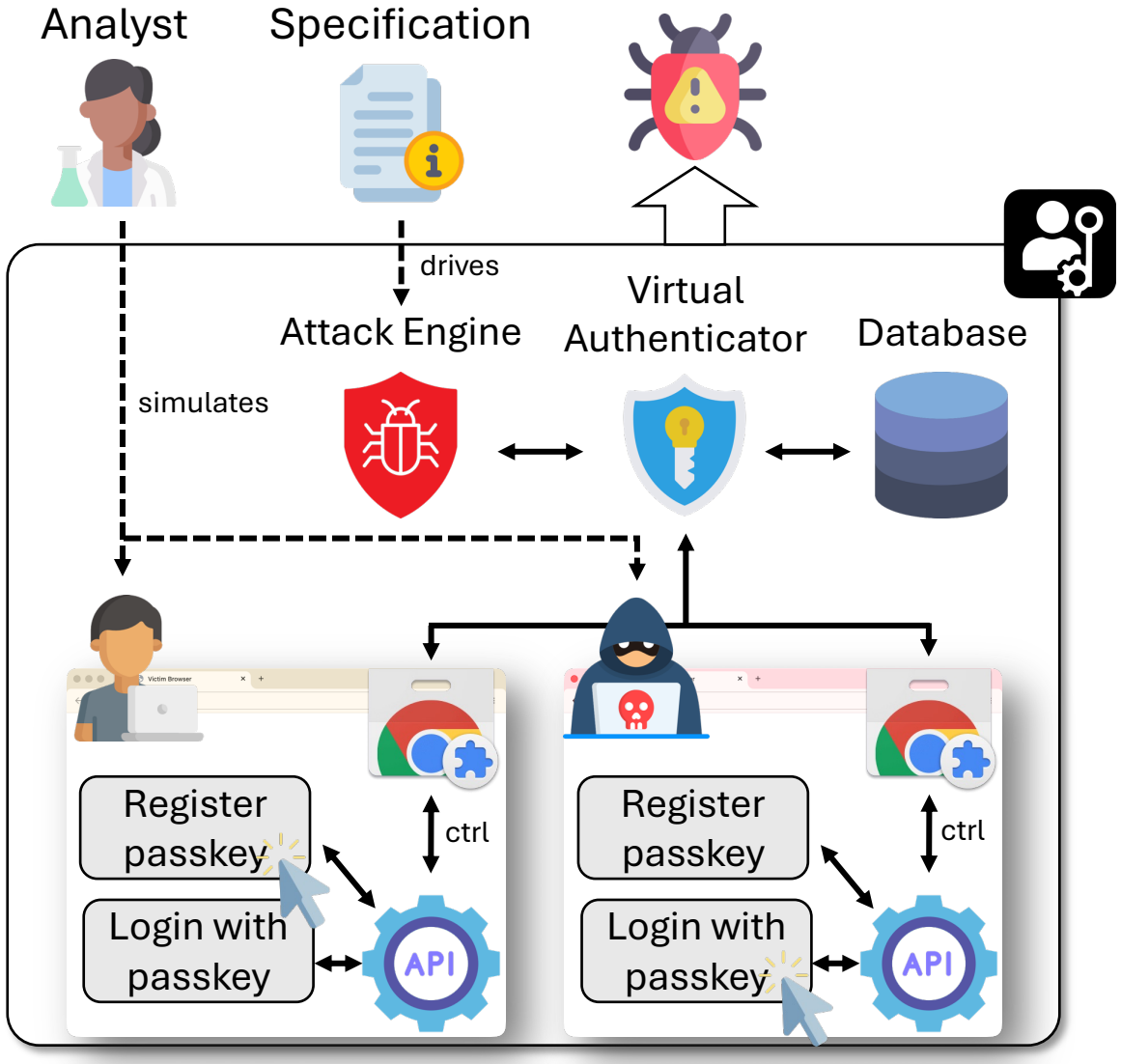
[1] https://web.dev/articles/passkey-management

PASSKEYS-ATTACKER

Focus: Websites (Front + Backend)



Tool: Architecture



Compliance[1] of Passkey Implementations on Websites

Attack Type (AT)	CVSS	CVE	CWE	Requirement	Validation [89]	RW	Detection Method (DM)	# Vuln. RPs Reg.	Auth.	∪
Signature	9.8	–	Improper Verification of Cryptographic Signature (CWE-347)	–	Use $cred^{pubkey}$ to verify signature over authData and clientData (§7.2.21)		**Bit Flip**: Flip the last bit of the signature	–	5	5
Credential Overwrite	9.1	[23]	Authorization Bypass Through User-Controlled Key (CWE-639)	Leaked $cred^{id}$ and/or $cred^{pubkey}$	$cred^{id}$ is not yet registered for any user (§7.1.26)		**Swap $cred^{id}$+$cred^{pubkey}$**: Register passkey with victim's $cred^{id}$ and $cred^{pubkey}$ **Swap $cred^{id}$**: Register passkey with victim's $cred^{id}$ and attacker's $cred^{pubkey}$ **Swap $cred^{pubkey}$**: Register passkey with attacker's $cred^{id}$ and victim's $cred^{pubkey}$	13 12 2	– – –	14
Related Origins	8.8	–	Externally Controlled Reference to a Resource in Another Sphere (CWE-610)	–	Include only active and trusted related origins (§5.11)		**Dangling Domains**: Check if related origins contain registrable domain	–	1	1
Challenge	8.2	[27]	Session Fixation (CWE-384)	Leaked or injected challenge (e.g. via XSS or CSRF)	clientData.challenge is options.challenge (§7.1.8, §7.2.11)	[44]	**Bit Flip**: Flip last bit of challenge **Reuse**: Use challenge that has already been used **Session Binding**: Use valid challenge from another session	1 8 9	1 5 14	22
Origin	8.1	[26]	Origin Validation Error (CWE-346)	Subdomain Takeover [84]	clientData.origin is trusted (§7.1.9, §7.2.12)	[44] [44]	**Cross-Site**: Set clientData.origin to a different site **Subdomain**: Set clientData.origin to a subdomain **Parent Domain**: Set clientData.origin to a parent domain	7 23 31	5 23 23	40
RP ID	8.1	[26]	Origin Validation Error (CWE-346)	Subdomain Takeover [84]	authData.hash(rp^{id}) is hash(options.rp^{id}) (§7.1.14, §7.2.15)	[44]	**Cross-Site**: Set authData.hash(rp^{id}) to the hash of a different site **Subdomain**: Set authData.hash(rp^{id}) to the hash of a subdomain **Parent Domain**: Set authData.hash(rp^{id}) to the hash of a parent domain	8 7 5	4 4 3	9
User Verified	6.8	[21]	Improper Authentication (CWE-287)	Physical access to authenticator	authData.flags.UV is true if options.userVerification is required (§7.1.16, §7.2.17)	[44]	**Unset**: Set authData.flags.UV to false	8	6	10
Context	6.4	–	Improper Verification of Cryptographic Signature (CWE-347)	Access to valid attestation signature (e.g. via XSS)	clientData.type is "webauthn.create" / "webauthn.get" (§7.1.7, §7.2.10)		**Nonsense**: Set context to "abc.def" **Swap**: Swap context create ↔ get	5 11	4 8	12
User Present	5.9	[22]	Improper Authentication (CWE-287)	Malware	authData.flags.UP is true (§7.1.15, §7.2.16)		**Unset**: Set authData.flags.UP to false	21	16	27
Allow Credentials	5.3	[25]	Observable Response Discrepancy (CWE-204)	–	Return imaginary $cred^{ids}$ in requestOptions.allowCredentials for non-existing accounts (§14.6.2, §14.6.3)		**Random**: Identify as random user. requestOptions.allowCredentials empty?	–	68	68
Framing	5.1	–	Improper Restriction of Rendered UI Layers or Frames (CWE-1021)	No generic clickjacking protections in place [17]	If clientData.crossOrigin is true, check if framing by other origins is allowed and if clientData.topOrigin is trusted (§7.1.10+11, §7.2.13+14)		**Cross-Origin**: Set clientData.crossOrigin to true **Cross-Site**: Set clientData.topOrigin to a different site **Subdomain**: Set clientData.topOrigin to a subdomain **Parent Domain**: Set clientData.topOrigin to a parent domain	99 93 94 84	87 92 92 84	102
Signature Counter	4.8	[24]	Improper Authentication (CWE-287)	Physical access to cloned authenticator	authData.sig^{cnt} is greater than previously stored sig^{cnt} (§7.2.22)	[44]	**Reduce**: Set authData.sig^{cnt} to 2. Login. Set authData.sig^{cnt} to 1. Login.	–	57	57
Backup State	0.0	–	–	Authenticator is lost	authData.flags.BS is false if authData.flags.BE is false (§7.1.17, §7.2.18)		**Mutual Exclusion**: Set authData.flags.BE to false and authData.flags.BS to true	58	57	65
Backup Eligible	0.0	–	–	Authenticator is lost	authData.flags.BE is equal in registration and authentication (§7.2.19)		**Swap On**: Register authData.flags.BE=false. Login authData.flags.BE=on. **Swap Off**: Register authData.flags.BE=true. Login authData.flags.BE=false.	– –	55 55	57
Length of $cred^{id}$	0.0	–	–	–	$cred^{id}$.length ≤ 1023 bytes (§7.1.25)		**Length 1024**: Set $cred^{id}$ to 1024 random bytes	48	–	48
							Vulnerable websites with CVSS severity: Critical (18), High (53), Medium (103), None (81), All combined: 103 (of 103 analyzed RPs)			

15 Attack Types

4 CVSS Scores
Critical, High, Medium, None

8 Attack Requirements

15 Validations from Spec.

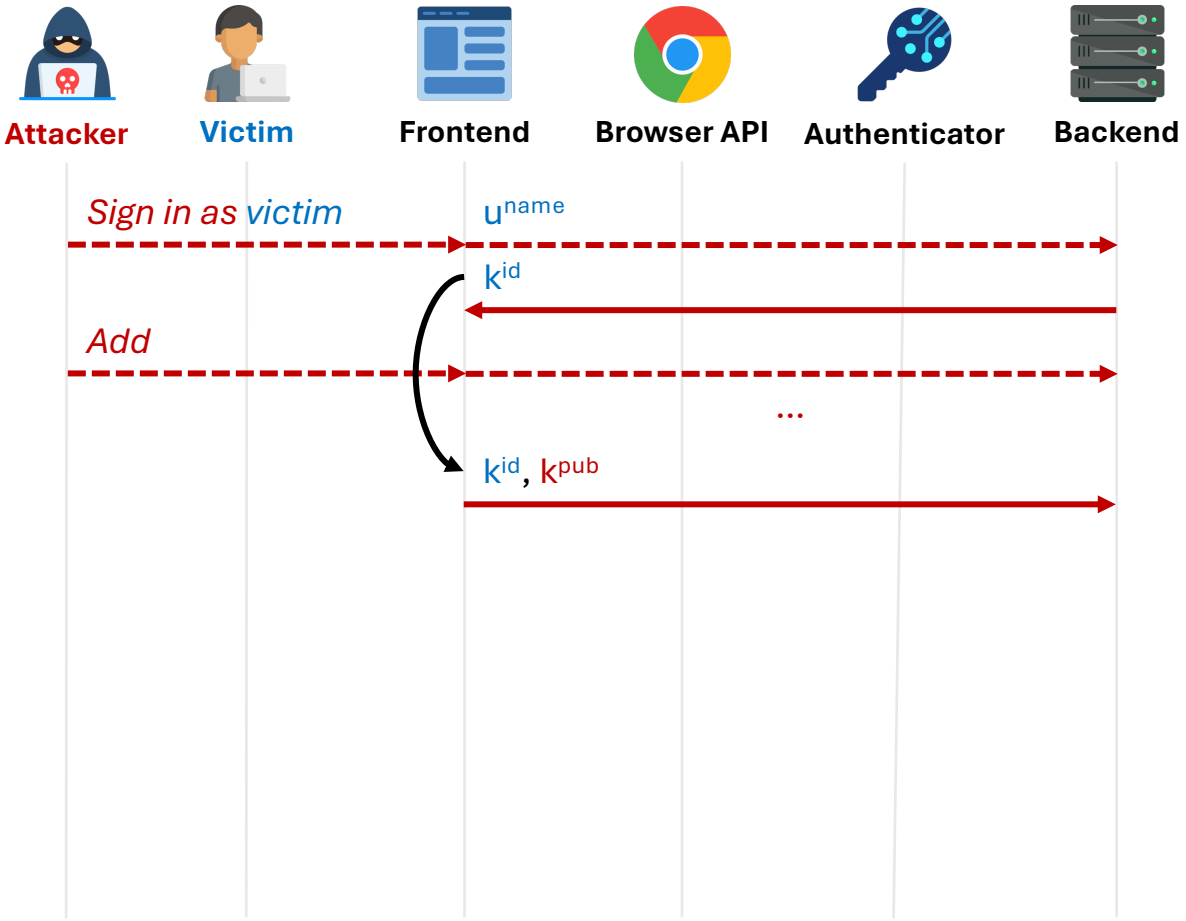
28 Detection Methods

103 of 103 not compl.

18 Critical, 53 High

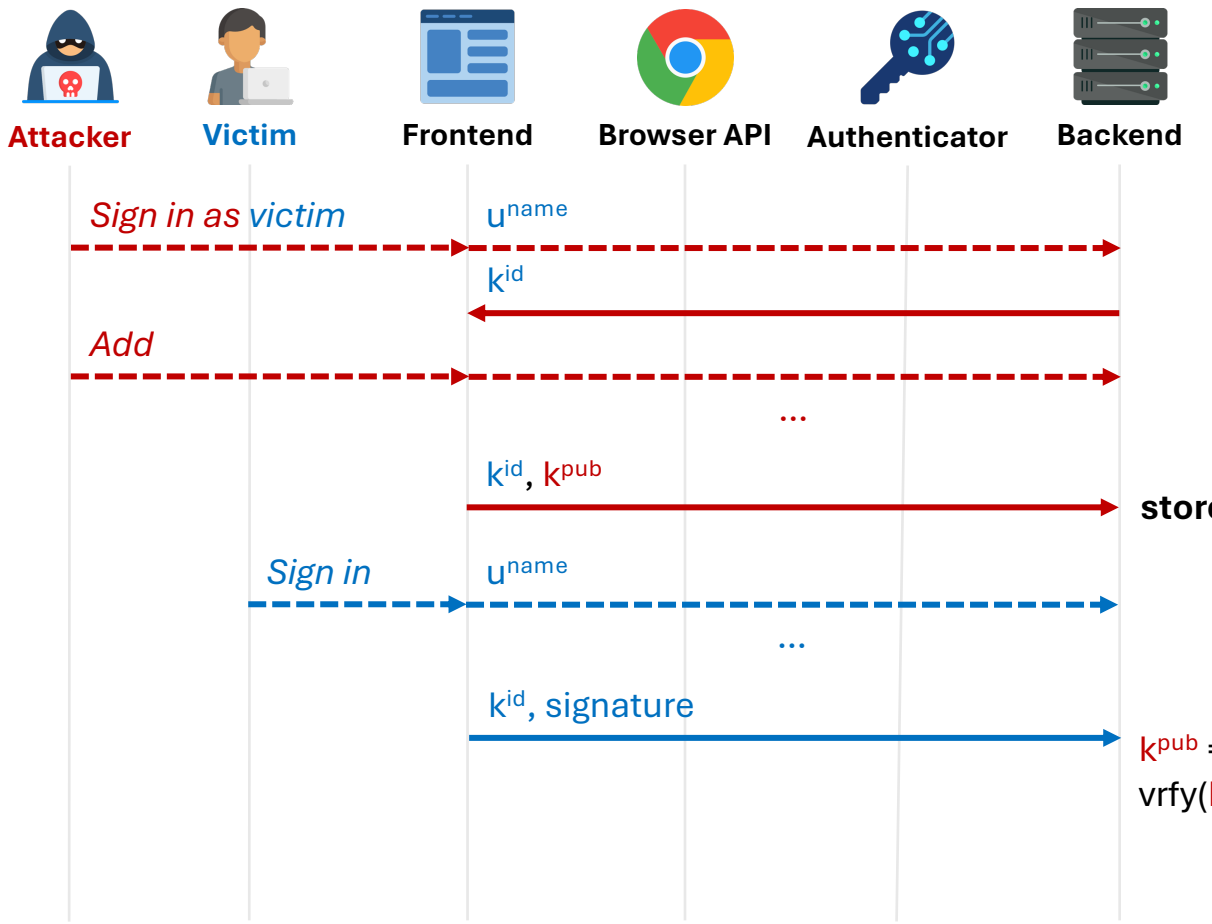
[1] https://www.w3.org/TR/webauthn-3

Passkey Management Failures (1/2)



User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}

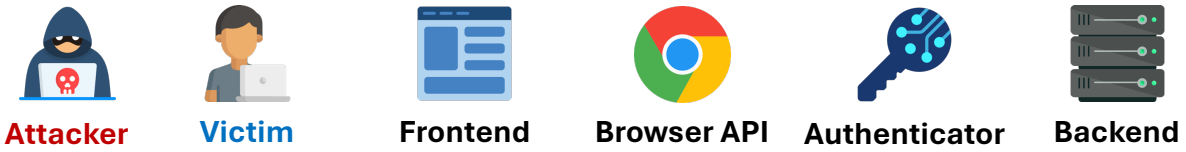
Passkey Management Failures (1/2)



User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}
u^{id}	k^{id}	k^{pub}

Passkey Management Failures (1/2)

User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}
u^{id}	k^{id}	k^{pub}



Sign in as victim — u^{name}

k^{id}

Add

...

k^{id}, k^{pub} — **store_append**(u^{id}, k^{id}, k^{pub})

Sign in — u^{name}

...

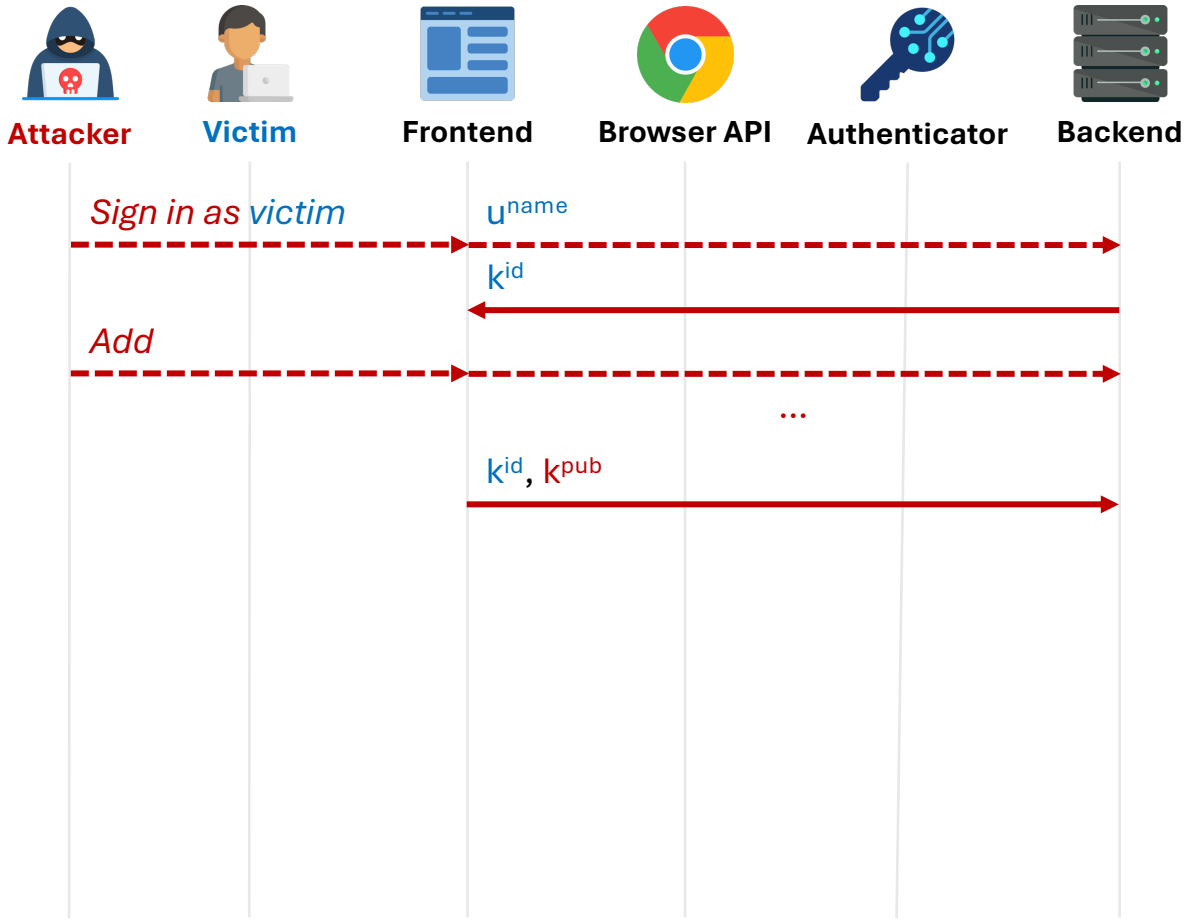
k^{id}, signature

k^{pub} = **get_last**(k^{id})

Error: Signature verification failed! — vrfy(k^{pub}; signature)

Account Lockout

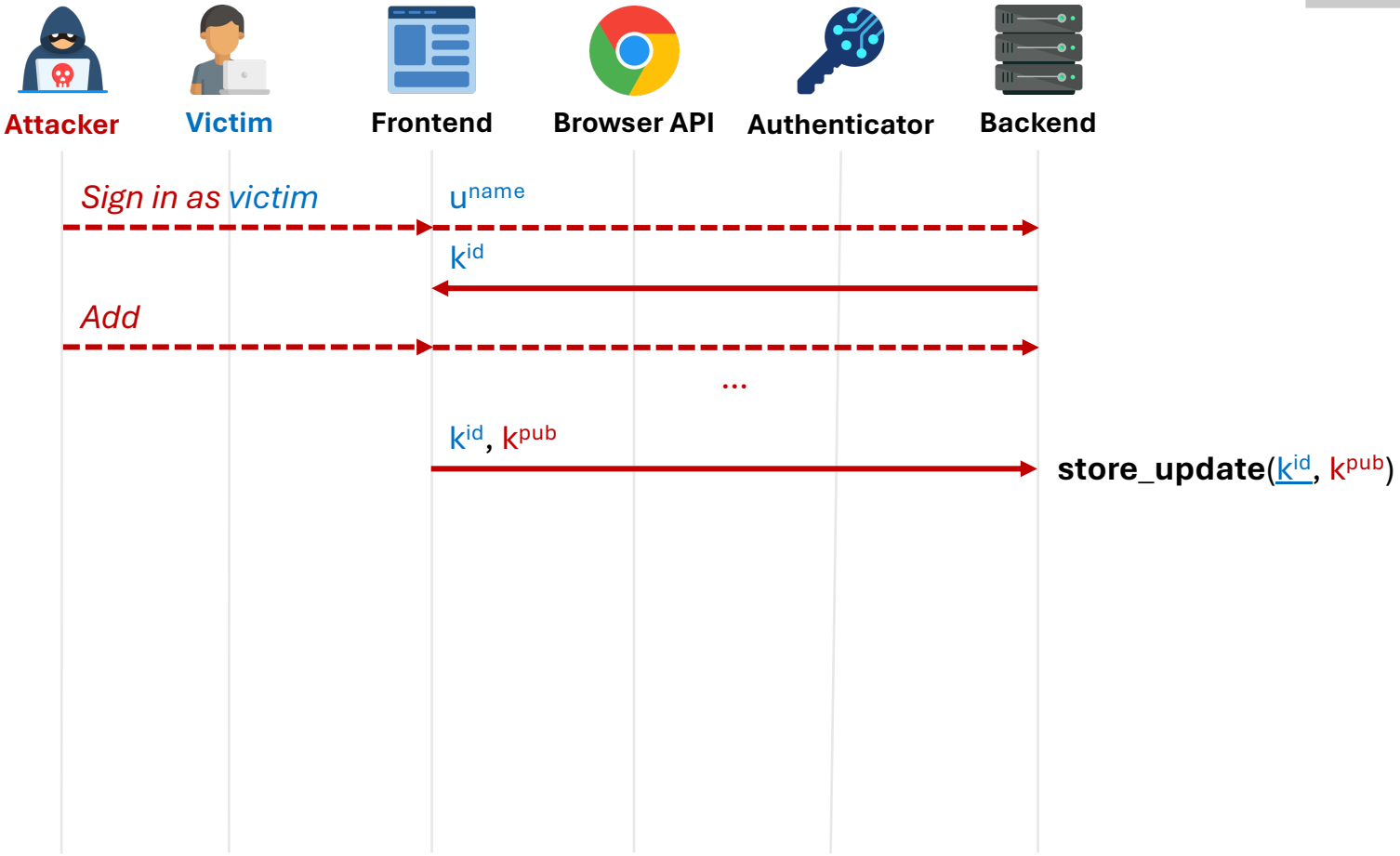
Passkey Management Failures (2/2)



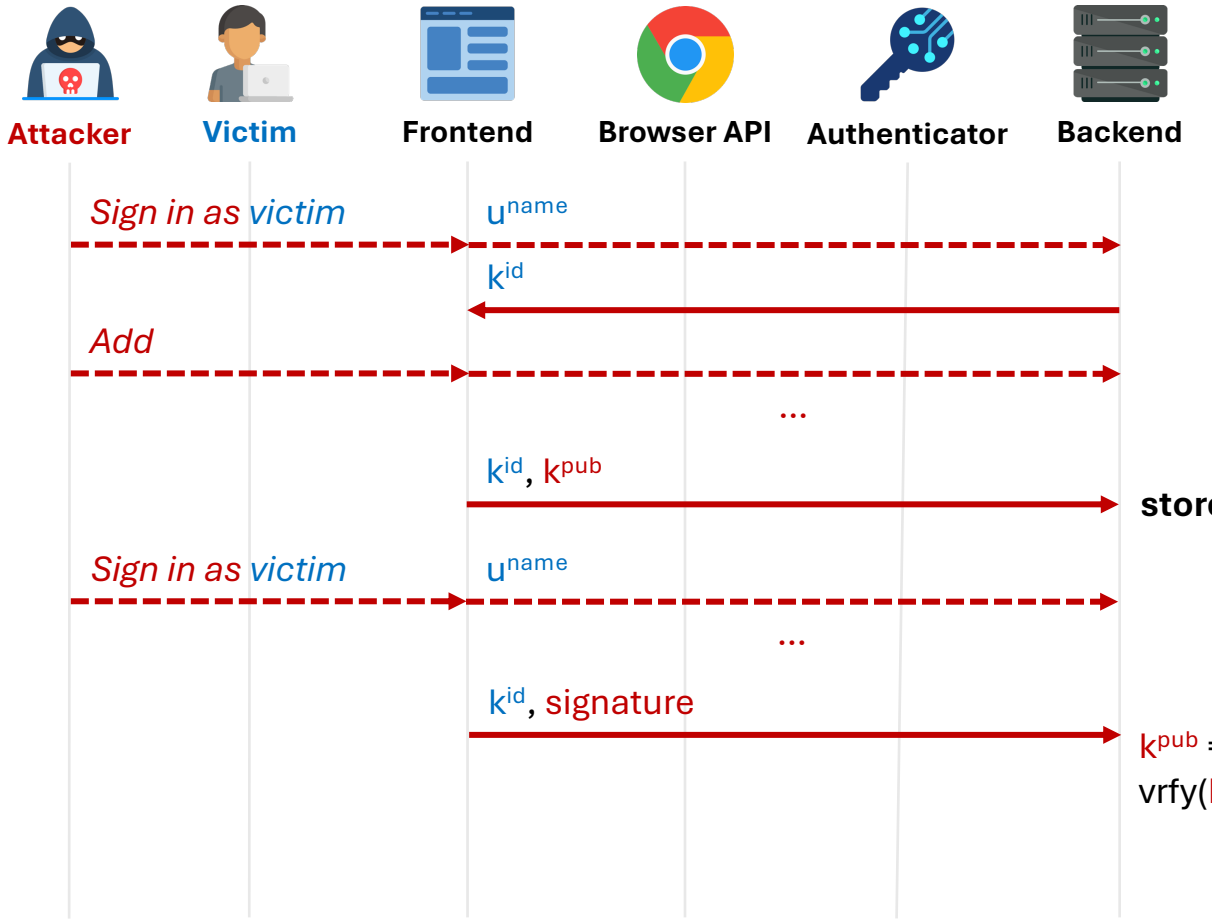
User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}

Passkey Management Failures (2/2)

User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}

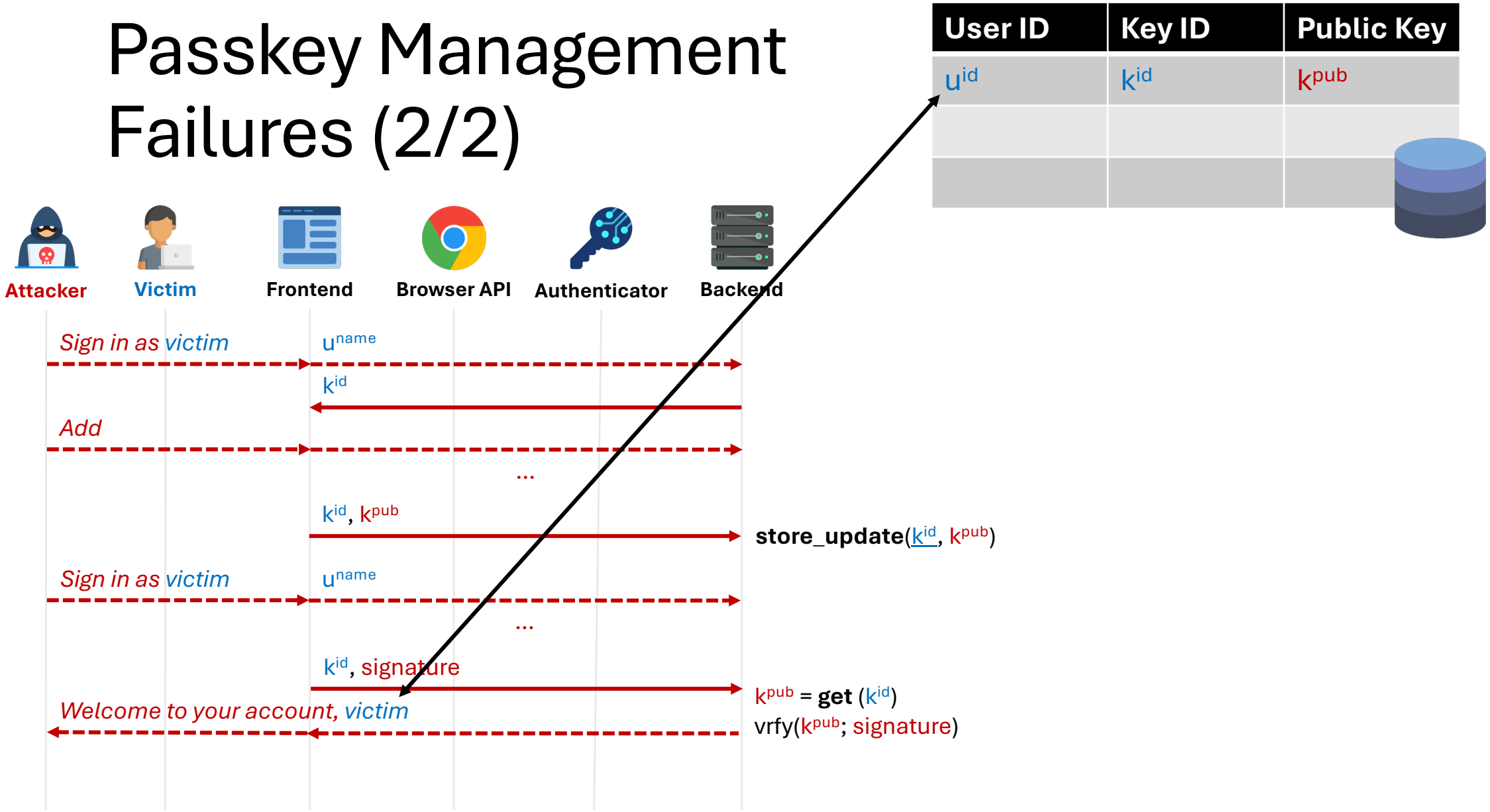


Passkey Management Failures (2/2)



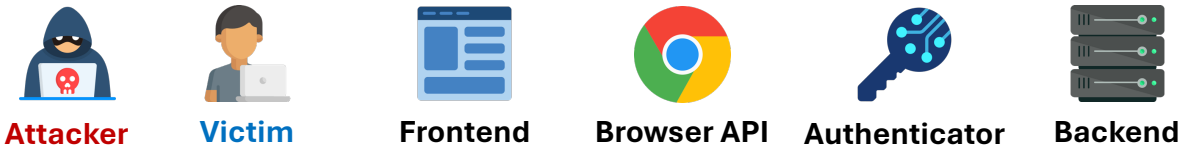
User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}

Passkey Management Failures (2/2)



Passkey Management Failures (2/2)

User ID	Key ID	Public Key
u^{id}	k^{id}	k^{pub}



Sign in as victim — u^{name}

k^{id}

Add

...

k^{id}, k^{pub} — **store_update**(k^{id}, k^{pub})

Sign in as victim — u^{name}

...

k^{id}, signature — k^{pub} = **get** (k^{id})

Welcome to your account, victim — vrfy(k^{pub}; signature)

Account Takeover



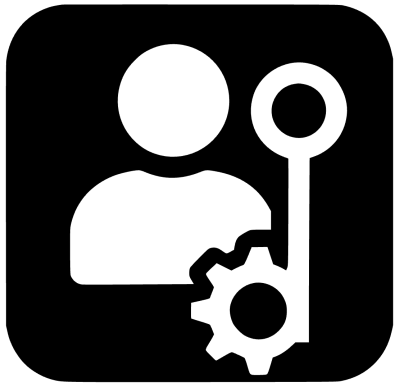




Thanks! Meet me for a **live demo**.

The paper, tools, and all artifacts are available on

https://passkeys.tools







The State of Passkeys: Studying the Adoption and Security of Passkeys on the Web

Louis Jannett [1], Andreas Mayer [2], Maximilian Westers [2], Vladislav Mladenov [1], Christian Mainka [3], and Jörg Schwenk [1]

[1]Ruhr University Bochum
[2]Heilbronn University of Applied Sciences
[3]University of Wuppertal

Abstract

Passkeys provide a secure and phishing-resistant authentication method based on FIDO2 and WebAuthn. They have recently gained popularity, with an increasing number of websites adopting them. Nevertheless, a comprehensive security analysis that evaluates such websites at scale has not been fully addressed. We present PASSKEYS-RADAR, a continuously updated dataset that tracks the deployment of passkeys on the Internet since 2021. To build this dataset, we aggregated diverse sources, including community directories, Tranco 1M, CrUX 18M, and historic Internet archive data. We analyzed the collected data of 872 passkey-enabled websites and shed light on how passkeys are implemented and managed. We identify major differences in how websites allow users to add or delete passkeys and find that websites request authenticators to use deprecated cryptographic algorithms.

To perform a comprehensive security evaluation of passkey-enabled websites, we developed PASSKEYS-ATTACKER. The tool allows for precise manipulation of WebAuthn messages at every step of the protocol and integrates 15 attack types of which 10 were not covered in previous work. Among them, 2 attack types have *critical* CVSS scores. We discovered them on 18 out of 103 evaluated websites. These attacks take over user accounts, delete their passkeys, or lock them out of their accounts. Nearly half of the tested sites (53) were vulnerable to at least one attack with a *high* CVSS score, exposing users to threats such as phishing and session fixation.

1 Introduction

"Hackers don't break in – they sign in" [66]. In May 2025, security researchers found 184 million leaked passwords [96]. Within a few weeks, this estimate was revised to 16 billion passwords, making it the largest known password compromise to date [95]. Such incidents highlight the urgent need to move beyond passwords towards passwordless authentication.

Passkeys. Passkeys are the modern, phishing-resistant, and user-friendly authentication method that replaces passwords with strong public-key cryptography for improved security. They are based on standards developed by the Fast IDentity Online (FIDO) Alliance [36] and W3C [89]. Today, passkeys are supported across all major browsers, including Chrome, Firefox, and Safari, as well as on platforms such as Windows, macOS, iOS, and Android [77]. Large corporations such as Apple, Google, and Microsoft drive the adoption of passkeys. These companies are members of the FIDO Alliance and actively integrate and promote passkeys in their ecosystems.

Passkey Authentication Scheme. Passkeys are built on two core protocols, jointly referred to as FIDO2: (1) Web Authentication API (WebAuthn), which defines how the Relying Party (RP) communicates with the client, and (2) Client-to-Authenticator Protocol (CTAP), which specifies how the client communicates with the authenticator. This paper examines WebAuthn and evaluates the security of RPs, see Figure 1.

(a) First, the RP initiates the authentication process by providing a randomly generated challenge to the client.
(b) The client then asks the authenticator to sign this challenge with the private key that it has stored for that RP.

Figure 1: Scope of this Paper. First, we quantify the adoption of passkeys on real-world websites using PASSKEYS-RADAR (§3) and we study how they are used and managed by these sites (§4). Second, we implement all security tests from the WebAuthn specification into PASSKEYS-ATTACKER (§5), and evaluate how secure websites implement passkeys (§6).

Funded by Ministry of Culture and Science of the State of North Rhine-Westphalia and CASA Cyber Security in the Age of Large-Scale Adversaries