To Go Passwordless or Not?

The systemic problems not addressed by the current passwordless hype/trend

By Dylan Rosario (Creator of ENI6MA Proof and Key-less Cryptography)

Password-less authentication, whether delivered as consumer-grade passkeys, enterprise FIDO tokens, or mobile-centric MFA apps, promises to replace memorized secrets with cryptographically bound "something-you-have" credentials protected inside secure hardware enclaves.In theory this reduces phishing and credential-stuffing by anchoring the private key to a physical device and gating its release behind a local factor such as biometrics or a short PIN.Vendors in the dialogue tout hardware FIDO keys whose fingerprint never leaves the chip, cloud-synced passkeys that pair over Bluetooth for proximity proof, and RSA iShields or one-time-password tokens for air-gapped sites, while recommending policy safeguards (HR/IT guidance, separation of personal and work credentials) and fallback enrollment or recovery workflows to handle lost devices.

Yet the conversation exposes a long list of operational and architectural challenges.Password-less binds every user to a specific, often expensive, device and, crucially, to the vendor's key-escrow or synchronization service, ceding sovereignty over credential storage and recovery to Big Tech or third-party clouds.It struggles in heterogeneous estates where legacy or custom applications cannot consume WebAuthn, and it breaks completely for headless automation, robotics, or server-to-server APIs that have no human to "touch the key."Proximity checks via BLE or physical contact are useless in subterranean, mobile, or GPS-denied environments; air-gapped networks must fall back to HOTP tokens, re-introducing shared secrets.Recovery remains brittle: if a hardware key is lost, stolen, or drained, administrators must still rely on secondary authenticators, often passwords, SMS, or biometrics, which re-opens the very attack surfaces password-less was meant to close.Moreover, the underlying public-key infrastructure has not been eliminated; it is merely shifted into device firmware and cloud synchronization layers that are still vulnerable to supply-chain compromise, side-channel extraction, and opaque vendor control.

Consequently, password-less proves to be a tactical patch rather than a strategic fix.It mitigates phishing for interactive users but perpetuates device dependence, preserves a 50-year-old PKI trust anchor, and fails to provide a scalable, vendor-agnostic path for automation, offline operations, or sovereign key custody.Until authentication is decoupled from both hardware possession and centralized certificate stores, eliminating stored secrets altogether and enabling self-contained proofs that survive loss, compromise, and connectivity gaps, password-less remains a band-aid that addresses symptoms while leaving the systemic wounds of today's identity fabric unhealed.

#	Challenge	
C-1	**Device Dependence & Sovereignty Loss**	Passkeys and most "passwordless" sc
C-2	**Incompatibility with Headless & Automated Systems**	Sc
C-3	**Legacy / Custom App Support**	
C-4	**Biometric Privacy & Revocation**	
C-5	**Lost / Stolen / Dead Devices**	
C-6	**Weak Fall-back Factors**	
C-7	**Proximity & Connectivity Limits**	
C-8	**PKI/CA Attack Surface Remains**	
C-9	**Vendor Lock-in & Inter-use Confusion**	
C-10	**Help-Desk & Enrollment Overhead**	

1. Key Challenges

2. Claimed *"Solutions"*

#	Claimed Mitigation	
S-1	**Hardware-Bound FIDO Keys + On-device Biometrics**	Fingerprint never leaves the secu
S-2	**Enterprise Policy & HR Guidance**	Written rules telling execs not to
S-3	**RSA iShields / HOTP Tokens for Air-Gapped Sites**	One-time-password or N
S-4	**Hybrid FIDO (BLE proximity, no pairing)**	Allows a phone to act as
S-5	**"Choose the Right Factor per Environment"**	A portfolio approach: device-bound p
S-6	**Help-Desk Recovery Workflows**	Help-desk verifies identity a

3. Why Each Mitigation Is Still a *Band-Aid*

Claimed Solution	
S-1 Hardware-Bound FIDO Keys	**Single point of failure** , lose or steal the key and the user is lock
S-2 Policy / HR Guidance	**Huma**
S-3 RSA iShields / HOTP	**Replayable once used** , OTP
S-4 Hybrid FIDO (BLE)	
S-5 Factor-Per-Environment	**Operati**
S-6 Help-Desk Recovery	

4. Big Picture: Why Conventional "Passwordless" Fails to Solve the Root Problem

1. **It secures the *UI* layer, not the *key-management* layer.**
   All mainstream schemes still rely on secrets (private keys, seed values)

stored on endpoints or proprietary sync clouds. Attackers simply move down-stack, from phishing the user to attacking secure elements, supply-chain firmware, or CA infrastructure.

2. **The trust anchor is *external* to the enterprise.**
   Whether Apple's iCloud Keychain, Google Password Manager, or a FIDO Alliance root of trust, credential integrity ultimately rests on a third-party service the organization cannot audit or revoke on demand.

3. **It cannot represent non-human actors.**
   Modern IT is API-driven: containers, batch jobs, satellites, drones. These entities need autonomous, cryptographically verifiable identities that do **not** depend on a human tap or BLE handshake.

4. **Recovery remains brittle and expensive.**
   A truly resilient system should treat credential loss or theft as a routine, self-healing event, *not* a security-ops fire drill that opens the door to social engineering.

5. **PKI's 1990s assumptions still lurk underneath.**
   As long as trust chains depend on revocation lists, CT logs, and public CAs, mass-compromise events (e.g., a corrupt or coerced CA) can still undermine millions of "passwordless" authentications overnight.

5. Take-away

Passwordless authentication, when defined as "device-bound FIDO passkeys or biometric unlock", addresses a narrow slice of the credential-phishing problem but leaves systemic issues of key custody, machine-to-machine identity, and sovereign control unresolved. For organizations that need *offline, automated, or high-assurance* operations, these schemes act more as glossy band-aids than as cures. A next-generation approach must remove device and vendor dependencies altogether, enable cryptographically provable identities for humans *and* machines, and make credential rotation or recovery as routine, and inexpensive, as rotating TLS session keys.

Outline : WhitePaper

"Beyond 'Password-less': Exposing the Architectural Gaps in FIDO-Based Authentication"

---

0. Title Page

- **Title**: *Beyond "Password-less": Exposing the Architectural Gaps in FIDO-Based Authentication*

- **Subtitle**: Why device-bound passkeys and FIDO tokens remain a tactical patch, not a strategic fix
- Author, affiliation, contact
- Revision date, document control & distribution list

1. ExecutiveSummary

- One-page synopsis of the promise of password-less, the systemic gaps uncovered, and the paper's key recommendations.
- High-level call to action for CISOs, architects, and regulators.

2. Introduction

2.1 **The Rise of Password-less** – Market adoption curves, regulatory drivers, headline breaches.
2.2 **Purpose & Scope** – Clarify that the paper critiques FIDO/WebAuthn-style schemes and evaluates them against high-assurance, offline, and machine-to-machine requirements.

3. Terminology & Architecture Primer

3.1 Definitions: passkey, authenticator, attestation, secure element, WebAuthn ceremony.
3.2 Overview of the FIDO credential lifecycle (registration, authentication, recovery).
3.3 Threat-model boundaries used throughout the paper.

4. Promised Benefits vs. Operational Reality

- **Marketing Claims**: phishing resistance, user convenience, password elimination.
- **Observed Reality**: reliance on external keystores, recovery friction, device cost.

5. Core Limitations of Today's Password-less Implementations

(Each subsection details impact, exploit scenarios, and quantitative cost where available)

6. Analysis of Claimed Mitigations

- Map S-1 $\rightarrow$ S-6 to the above gaps and show residual risk.
- Highlight single-point failures, replay/phishing windows, RF spoofing, and social-engineering channels.

Sec.	Limitation (maps to C-#)	Key Ev
5.1	**Device Dependence & Sovereignty Loss** (C-1)	Credentials anchored to vendor hard
5.2	**Headless & Automated Systems Unsupported** (C-2)	No tap/BLE possible for
5.3	**Legacy / Custom App Incompatibility** (C-3)	Re-engineering cost
5.4	**Biometric Privacy & Non-Revocability** (C-4)	Immutable trait
5.5	**Lost / Stolen / Dead Devices & Brittle Recovery** (C-5)	Help-desk load, s
5.6	**Weak Fall-back Factors** (C-6)	4-digit PINs, SMS, email O
5.7	**Proximity & Connectivity Limits** (C-7)	BLE spoofing, RF-de
5.8	**Persisting PKI / CA Attack Surface** (C-8)	X.509 compron
5.9	**Vendor Lock-in & Credential Intermixing** (C-9)	Off-boarding complexi
5.10	**Help-Desk & Enrollment Overhead** (C-10)	Token shipping, lo

7. Systemic Architectural Vulnerabilities

7.1 **Securing the UI, Ignoring Key Management** – Secrets still live on endpoints/clouds.
7.2 **External Trust Anchors** – Dependence on Apple/Google/FIDO roots outside enterprise control.
7.3 **Human-Centric Design Blind to Machine Identity** – APIs, containers, satellites need untethered proofs.
7.4 **Brittle, Expensive Recovery** – Recovery events escalate to high-touch support.
7.5 **Legacy PKI Assumptions** – Revocation latency, CA compromise blast-radius.

8. Threat Modeling & Attack Scenarios

DeviceTheft&Side-ChannelExtraction.
Because FIDO passkeys and other password-less credentials are anchored to a phone, laptop TPM, or external token, simply stealing the device grants an attacker unlimited offline time to probe the secure element. Modern hardware security modules resist straightforward key reads, but well-funded adversaries can apply differential power analysis, electromagnetic fault injection, or laser glitching to recover private keys that never "leave" the chip. Even without laboratory tooling, a lost phone protected only by a four-digit PIN or weak biometric can be brute-forced in hours. Once the key is cloned, every relying party that accepted the passkey must be considered compromised, a revocation headache compounded by the lack of centralized credential visibility.

BLERelay/Evil-TwinAttacks.
FIDO's "hybrid" and passkey flows often rely on Bluetooth Low Energy to prove that the authenticator is in proximity to the client device. Relay adversaries exploit BLE's permissive connection latency by capturing packets near the authenticator and forwarding them in real time over Wi-Fi or cellular to a remote laptop spoofing the legitimate client. The victim sees the expected biometric prompt, and the server sees a valid signature, but the authenticator is actually

signing a challenge for the attacker's session. Rogue access points that clone SSIDs ("evil twins") can further coax users into pairing with attacker-controlled relays. Because the core cryptographic exchange remains intact, conventional TLS or FIDO attestation checks do not detect the man-in-the-middle, proximity is the only broken assumption.

Supply-ChainFirmwareBackdoors.
Secure elements, TPMs, and U2F tokens ship with opaque, vendor-signed firmware; customers rarely have the source or the signing keys. A single insider threat, coerced developer, or compromised build server can insert a dormant key-exfiltration routine or weaken the random-number generator across millions of devices before detection. History offers sobering precedents, from pre-loaded spyware on consumer laptops to deliberate RNG sabotage, which show that "hardware-root-of-trust" is only as strong as its supply chain governance. Because the entire FIDO model delegates private-key protection to these black-box components, a firmware-level compromise silently nullifies phishing resistance at global scale.

CA/SubordinateCACompromise.
Although passkeys hide the private credential in hardware, they still depend on traditional X.509 chains to authenticate the relying party's TLS endpoint and on manufacturer-issued attestation certificates to vouch for device integrity. If a commercial certificate authority, or any delegated subordinate CA, is breached, attackers can mint valid server certificates and lure users to indistinguishable phishing sites that request legitimate WebAuthn signatures. Similarly, forging an attestation certificate allows malicious peripherals or emulator software to masquerade as genuine FIDO authenticators, bypassing enterprise policies that restrict which hardware models may enroll. Because global CRL/OCSP revocation is slow and error-prone, the attacker's window often lasts days or weeks.

Help-DeskSocialEngineeringforRecoveryOverride.
Password-less deployments inevitably include break-glass flows for lost, broken, or battery-dead devices. Attackers exploit these human-driven processes, phone hotlines, IT ticketing systems, identity-proofing chats, to convince support staff to reset a user's credential or register a new passkey on the attacker's hardware. The stronger the front-end authentication seems, the more likely support agents are to trust callers who supply easily harvested personal details or deep-faked voiceprints. Unlike phishing-resistant WebAuthn ceremonies, recovery channels sit outside cryptographic enforcement and are governed by policy and training; once an attacker succeeds, the newly issued credential is indistinguishable from a legitimate one, defeating post-incident forensics.

Cross-Cutting Implications.
These scenarios share a common theme: password-less security is only as strong as the weakest link in its multifaceted trust chain, hardware manufacturing, local transport protocols, certificate authorities, and human processes. Each vector bypasses the narrow "phishing resistance" guarantee of FIDO by attacking availability, integrity, or governance layers that the specification assumes benign. A robust next-generation architecture must therefore go beyond device possession and biometric gating, incorporating continuous cryptographic agility, verifiable

supply-chain transparency, and tightly scoped, auditable recovery channels that cannot be subverted with a phone call or a well-placed relay.

Case Studies

9.1 **Subterranean Field Agents (GPS-Denied)** – BLE & touch infeasible.
9.2 **Air-Gapped Industrial Control Networks** – "Offline password-less" devolves to OTP replay.
9.3 **CI/CD Automation in Cloud Pipelines** – No human presence at scale.
9.4 **Executive Off-Boarding & Mixed Passkeys** – Vendor lock-in headaches.

Attack Scenarios Undermining FIDO Passwordless Authentication

Device Theft & Side-Channel Extraction

Example: YubiKey Cloning via Side-Channel Attack (Real-World Example)

One example of device theft enabling secret extraction is the recently disclosed **EUCLEAK** vulnerability in YubiKey devices. In 2024, researchers found that certain YubiKey 5 Series and Security Key models could be **cloned via side-channel** if an attacker briefly obtained the key physically. Yubico (the manufacturer) confirmed that threat actors with temporary physical access can exploit unintended electromagnetic leaks from the token's cryptographic operations. This side-channel flaw, present in firmware <5.7, allows siphoning of the token's **FIDO credential private keys**, despite the device's secure design.

Once in possession of the key (even for a short time), an attacker can perform a stealthy attack. The **attack scenario** involves first stealing the victim's primary login (username/password via phishing or other means), and then **covertly obtaining the user's FIDO key** without raising suspicion. Using specialized equipment, the attacker sends repeated authentication requests to the token and measures its side-channel emissions. After returning the device to the user (so the victim doesn't realize it was taken), the attacker can computationally extract the token's ECDSA private key from the recorded signals. In effect, the adversary ends up with a *clone* of the user's hardware authenticator.

With a cloned YubiKey, the attacker gains undetected access to any accounts protected by that token. They can sign in as the user, since they now possess both the password and a copy of the "something you have." As researcher Thomas Roche explained, *"the attacker can build a clone of your authentication factor , a copy of your own YubiKey. You feel safe when you actually are not."*. This real-world finding highlights that FIDO hardware, while extremely secure, is not **inviolable** – given physical possession and a sophisticated side-channel, even the strongest passwordless factor can be compromised. It's worth noting that such an attack is **highly resource-intensive** (requiring expensive lab equipment and expertise) and more likely to be used in targeted, state-sponsored scenarios than by common cybercriminals. Nonetheless, it underscores a critical vulnerability: if an attacker can **steal and return** a FIDO token undetected, they may break the core security assumption that only the legitimate user holds the authentication secret.

Example: Cloning a Google Titan Security Key via EM Side-Channel (Simulated Attack)

Another illustrative case is a **simulated attack by NinjaLab researchers** against Google's Titan Security Key (a FIDO-compliant token). In 2021, French security researchers obtained a Titan key and demonstrated they could **extract its private key** by measuring electromagnetic emanations – a classic side-channel technique. The team physically opened the key's casing and exposed its secure element chip, then used a sensitive radio probe to monitor the chip's signals as it performed cryptographic signing operations. By capturing thousands of ECDSA signature operations (using the key normally over about six hours), they gradually inferred the secret ECDSA key stored inside.

Crucially, the attackers in this scenario **did not need to know any PIN or biometric** – they leveraged the fact that the token will sign challenges whenever prompted. After recording enough signals, the researchers were able to compute the token's private key, essentially **making a clone in software**. They even returned the original Titan device to its owner; the victim would continue using it, unaware that an attacker now had a duplicate. As Sophos News reported, *"if attackers can get their hands on your Titan key for a while... they can extract the private key and use it to make a software clone of your Titan key. The crooks could then snoop on you after returning the original key to you"*. This means an attacker could silently authenticate as the user in parallel, without confiscating the real key.

This NinjaLab attack was a **proof-of-concept** requiring significant effort. The researchers had to use about $10,000 worth of equipment and destructive techniques (acid and precision tools to open the device), making it impractical for casual attacks. Moreover, Google's Titan keys have a built-in protection: an **authentication counter** that increments with each use. If a clone is used, the counter on the clone and the original can diverge, alerting services that a key may have been duplicated. However, not all services actively check these counters, and a skilled attacker might synchronize usage to avoid detection. The broader lesson is that even in FIDO's **passwordless architecture**, possession of the physical authenticator is a critical assumption. These side-channel examples show that if that assumption is violated by an "evil maid" or lab attack, the **security collapses** – an attacker can bypass passwordless protections by *extracting the secrets from the device* itself.

BLE Relay / Evil-Twin Attacks

Example: Google Titan Security Key BLE Hijack (Real-World Vulnerability)

A concrete example of a Bluetooth-based attack occurred with Google's Titan Security Key in 2019. Google had to **recall and replace** its Titan Bluetooth Low Energy (BLE) keys after discovering a pairing misconfiguration that allowed attackers in close proximity to hijack the connection. In one attack scenario, if a user pressed the button on their Titan key to authenticate, an attacker within ˜30 feet could **wirelessly connect to the key** at that moment. By doing so, the attacker's device could impersonate the user's legitimate computer. If the attacker had already stolen the user's password, they could

complete the login by using the Titan key over BLE – essentially tricking the key into authenticating the attacker's session. This vulnerability meant that the BLE key's intended range limitation (to be "near" the user) could be exploited; an opportunistic adversary in the same room or a public space could jump in on the Bluetooth link when the user attempts to log in.

Another related weakness was an "**evil-twin**" pairing attack. When a Titan BLE key was being paired for the first time with a device, an attacker could spoof the key during that setup process. In practice, the attacker could masquerade as the user's security key and establish a rogue pairing with the victim's laptop or phone. Once paired as a trusted device, the attacker's **impersonator key** could act as a keyboard or mouse via Bluetooth, performing unauthorized actions. Both of these issues were difficult to pull off – requiring the attacker to be nearby at exactly the right time and to have the user's credentials – but they were realistic enough for Google to issue a broad replacement program. This real-world incident underscores that **wireless FIDO authenticators introduce new risks**: an attacker doesn't need to steal your key physically if they can virtually insert themselves into the communication. The assumption of proximity can be falsified, allowing a breach of the passwordless login. In response, Google stopped offering BLE-based keys (favoring NFC or USB) after 2021, acknowledging that wireless convenience wasn't worth the security trade-off.

Example: Long-Range BLE Relay Attack (Research Demonstration)

Beyond specific product bugs, researchers have shown that *relay attacks* can extend or bypass the normal range and trust of BLE-based authentication. In 2022, NCC Group demonstrated a tool that performs a **Bluetooth Low Energy link-layer relay** with only milliseconds of added latency. This advanced attack operates below the application level, forwarding encrypted BLE signals almost instantaneously between two distant points. For example, an attacker could leave a small relay device near the victim's **trusted device** (say, the phone or key fob that unlocks their computer or smart lock) and place a second relay device near the target system. The relay fools the target into thinking the legitimate authenticator is in close proximity, when in reality it could be across town – the messages are relayed over the internet or another channel in real time. NCC's research showed that their relay added as little as 8ms delay, staying within normal timing variance and thus **defeating typical proximity checks**.

The impact of such a relay is that any system relying on BLE signals to infer "device is nearby, therefore user is present" can be compromised. An attacker need not crack any encryption; they simply **bridge two locations**. In a hypothetical attack, your phone (which holds your passkeys or Bluetooth unlock for your laptop) could be at home with you, but a hacker could use a relay to unlock your office computer by making it appear your phone is right next to it. Neither encryption nor latency-bound defenses stopped this – the Bluetooth SIG itself has noted that the spec "should not be used as the only protection of valuable assets" for this very reason. This scenario is highly relevant to

FIDO-style authentication when using **phone-as-a-key** or BLE-enabled security keys. It highlights that **distance is not a foolproof security factor**: without additional safeguards like user presence checks or distance bounding (e.g. Ultra-wideband), a passwordless system can be **tricked into authenticating a remote attacker**. In essence, BLE relay attacks create an "evil twin" connection – the system thinks it's talking to the genuine authenticator nearby, but it's really connected through an attacker's devices. This proves that even in modern passwordless setups, **relaying and impersonation at the physical layer** remain potent threats, requiring mitigations beyond the FIDO protocol itself.

Supply-Chain Firmware Backdoors

Example: Concern Over Titan Key Manufacturing (Real-World Supply Chain Example)

Supply-chain attacks target the trust we place in hardware providers. A notable example came when Google introduced its Titan Security Keys: security experts raised concerns about the keys being manufactured by **Feitian, a Chinese company**, and the potential for backdoors in the firmware. In 2018, former Facebook CISO Alex Stamos and others publicly urged Google to be transparent about Titan's supply chain, fearing that the Chinese government could **coerce the manufacturer to insert a backdoor or tamper with the devices** in transit. The worry was that a stealthy modification at the factory (or during shipping) might allow an attacker – say, a nation-state – to later access accounts secured by those keys. As one source put it, the Chinese government could potentially "introduce some form of backdoor into the devices, or intercept the keys themselves and tamper with them," giving them the ability to hack target user accounts. This scenario was speculative (no such backdoor was found), but it was plausible enough to prompt calls for audits.

Google's response was to highlight Titan's design: Google **developed its own firmware** for the secure element chip and claimed it is installed in a controlled environment, with the chip being "permanently sealed" before going to Feitian's assembly line. In theory, this means even the third-party manufacturer can't alter the security-critical code. This approach anchors trust in the chip's internal ROM, rather than the external manufacturing process. Other major token vendors like Yubico also emphasize domestic or tightly controlled production to mitigate such risks. Still, the Titan key debate shows that **supply chain is a potential single point of failure** for passwordless hardware. Even without any known real-world implant, enterprises realized that if the wrong actor gained influence over the token's firmware (at factory or en route), they could undermine every user's authentication. A backdoored security key might, for example, quietly copy every private key it generates and later leak it. This would be catastrophic: the attacker would remotely have what is supposed to be an unextractable secret, effectively nullifying the benefit of FIDO's hardware-based credentials. The **real lesson** from this case is that users and organizations must **trust the integrity of the token supply chain** – a trust that is hard to verify, and once broken, voids the security of even the best cryptographic protocols.

Example: Malicious Firmware Injection in Authentication Tokens (Hypothetical Scenario)

Consider a hypothetical scenario that echoes real attacker tactics: a sophisticated adversary manages to **inject malicious firmware** into FIDO authenticators during the supply chain process. This could happen via an insider at a factory, a compromised update server (if devices support firmware updates), or interception of the devices before delivery. As security researchers have noted, hardware tokens can be **manipulated at multiple points** on their journey – from design and production to distribution – creating plenty of opportunities for a backdoor to be introduced. For instance, an attacker could modify the code on an open-source FIDO key (like Solo or Nitrokey) to include a hidden "master key" or to subtly send copies of any generated credential to the attacker's server. The device would still pass all outward checks and function normally for the user, but it would no longer be truly secure.

Such a **trojaned authenticator** would completely undermine passwordless security. Imagine an enterprise deploying dozens of security keys that have been backdoored at the factory – the attacker would then possess a skeleton key to every account secured by those tokens. Even if each user's key uses strong cryptography, the malicious firmware could simply bypass it (e.g. by logging every authentication or allowing a hidden override). The FIDO Alliance's own security reference notes that certain things are *out of scope* of the protocol, including trusting that the authenticator is genuine and uncompromised. This hypothetical attack hits exactly that weak point: the **trustworthiness of the token hardware**. It's a scenario frequently discussed by experts, because it shifts the battle from math to supply chain integrity. As Fraunhofer researchers put it, there is *"significant additional attack surface for malicious manipulation"* in the lifecycle of security tokens, and vulnerabilities in hardware or firmware mean that chips "must only be used with caution when security is important". In practice, no widespread incident of FIDO key firmware backdoors has been recorded, but similar supply-chain attacks (like espionage implants on other hardware or tampering with software updates) have occurred in other domains. This reinforces that both consumers and enterprises using passwordless tech must vet their hardware sources and, where possible, use **tamper-evident packaging, secure distribution channels, and audits**. Ultimately, if an attacker can subvert the authenticator at the supply stage, they bypass even the strongest passwordless architecture – obtaining the "keys to the kingdom" without any on-the-spot hacking.

CA/Subordinate CA Compromise

Example: DigiNotar Certificate Authority Breach Enables MITM (Real-World Attack)

FIDO2/WebAuthn authentication is designed to be phishing-resistant, but it still relies on the **TLS/HTTPS ecosystem** to ensure the browser is talking to the legitimate site. A stark example of how that trust can be broken was the **DigiNotar incident** in 2011. DigiNotar, a Dutch certificate authority (CA), was **hacked** and the attackers managed to issue themselves over 500 fraudulent digital certificates for various high-profile domains. Notably, this in-

cluded certificates for *Google's domains*. In a devastating development, those fake Google certificates were deployed in Iran to intercept Gmail traffic: an estimated 300,000 Iranian users' communications were silently spied on via **man-in-the-middle (MITM) attacks** using the rogue certs. Essentially, the attackers (allegedly state-sponsored) temporarily gained the ability to masquerade as Google to any browser that trusted DigiNotar as a CA.

In the context of passwordless authentication, such a CA compromise is extremely dangerous. Even if a user had a FIDO U2F key or a passkey protecting their Google account, an attacker who controls a **fraudulent certificate for google.com** can set up a fake site or proxy that the browser will accept as genuine. The FIDO protocol would be none the wiser – the user's security key will happily sign challenges for what it believes is the legitimate site (the web origin "google.com" still appears correct in the browser, thanks to the rogue certificate). During the DigiNotar affair, victims in Iran who went to Gmail likely saw a valid HTTPS padlock while the attacker's server intercepted their session. A hardware key wouldn't prevent this, because the user was **tricked at the TLS layer** rather than through a phony lookalike domain. This incident led browser vendors to swiftly remove DigiNotar from their trusted roots and the company went bankrupt soon after. Security organizations (ENISA, etc.) cited it as a classic example of the **"weakest link" problem** – that a single compromised CA undermines the security of potentially millions of users. For FIDO architectures, it is a sobering reminder that *authentication is only as strong as the HTTPS connection.* If that connection is subverted via a CA breach, attackers can perform MITM attacks that defeat even unphishable credential mechanisms.

Example: TurkTrust Mis-Issued Subordinate CA Certificate (Real-World Incident)

Not all certificate authority incidents are overt hacks; some are mistakes that create vulnerabilities. A notable case occurred in late 2012, involving a Turkish certificate authority called **TURKTRUST**. TurkTrust accidentally mis-issued two **intermediate CA** certificates to entities that were only supposed to get regular website certs. An intermediate (subordinate) CA certificate is powerful – it inherits the authority of the root CA, meaning the holder can generate valid certificates for *any domain* at will. In this incident, one of the mistaken intermediate certs was installed in a device (reportedly associated with a Turkish government office), which then was used to create a fake `*.google.com` certificate. In early 2013, Google's Chrome browser detected this fraudulent Google certificate being used in the wild (likely for intercepting traffic), and Google, Microsoft, and Mozilla all moved to block the rogue cert and the TurkTrust intermediates once the issue came to light.

The TurkTrust case illustrates how a **subordinate CA compromise (or error)** can directly facilitate phishing or MITM attacks. With a valid cert for `*.google.com`, an attacker could, for example, set up a transparent proxy that presents itself as Google to the victim. The victim's browser sees a perfectly valid certificate chain (since it chains up to TurkTrust, a trusted root at the time), and thus the lock icon doesn't raise any suspicion. If the victim at-

tempted to log in using a FIDO2 authenticator, they would be on the attacker's site without realizing it – the WebAuthn request would still show as coming from "google.com" (the legitimate origin). In this scenario, the attacker could either harvest the one-time response from a U2F key or even replay the whole authentication process in real time to the real Google server (an adversary-in-the-middle). From the user's perspective, everything would seem normal, but the attacker would silently **gateway the authentication**, defeating the purpose of the hardware key's phishing resistance.

This real incident was resolved before widespread damage, but it serves as a warning: **compromised CAs break the web's security model**. Even the most robust passwordless scheme assumes that when your device says "logging into example.com," it truly is example.com. A malicious or compromised CA can shatter that assumption. For enterprises, this means that in addition to securing user credentials, one must also **monitor and pin certificates** for critical services (some organizations now use certificate pinning or require additional identity verification of the service). In summary, the TurkTrust episode, like DigiNotar, shows that **attacks on the PKI infrastructure** can indirectly undermine FIDO authentication. The FIDO Alliance's solution is strong against phishing websites, but not against a trusted imposter site. Therefore, the integrity of CAs and rapid revocation of bad certs remain vital for the overall security of passwordless authentication systems.

Help-Desk Social Engineering for Recovery Override

Example: MGM Resorts Helpdesk Breach (Real-World Attack)

A recent real-world attack demonstrates how attackers might target the "human element" to bypass FIDO protections. In September 2023, MGM Resorts fell victim to a major cyber intrusion that started with a **help-desk social engineering** ploy. The hacking group (known as Scattered Spider/ALPHV) simply called MGM's IT support line, impersonating an authorized employee, and claimed they had forgotten their password or were having trouble accessing the account. By leveraging personal details gleaned elsewhere (like LinkedIn or public info) and sounding convincing, the attacker **convinced the help desk to reset the employee's account credentials** over the phone. In doing so, they gained initial access to MGM's network. Critically, the help desk also either **disabled the multifactor authentication** on the account or allowed enrollment of a new device, effectively **bypassing the user's FIDO2 security**. Once inside with valid credentials, the attackers moved laterally and ultimately deployed ransomware, causing widespread outages at MGM properties.

This incident underscores a key vulnerability: many organizations have an account recovery or override process for when users lose their devices or credentials. Attackers know that **convincing a human support agent** can be far easier than defeating cryptography. In the MGM case, the failure was that the support staff did not thoroughly verify the caller's identity or enforce strict policies, allowing a simple phone call to unravel the technical security measures. From a FIDO/passwordless perspective, the attackers didn't need to hack the hardware key or phish the user – they *went around it* by targeting the procedures meant for emergencies. Enterprise environments often have procedures

to issue temporary login codes, backup OTPs, or to register a new key when an employee says "I lost my security key." If those procedures are not locked down (for example, requiring manager approval or in-person verification), an attacker can exploit them. The MGM resort attack is a real-world proof that **social engineering the helpdesk** can nullify even the strongest authentication scheme. No matter how "unphishable" a FIDO login is, if an attacker can impersonate you to IT and get your account reset or a new factor added, they own your account. This is why companies are now training helpdesk staff and adding safeguards – such as callback verifications or secondary approvals – for any request to bypass 2FA.

Example: "Lost Key" Ruse to Enroll Attacker's Device (Hypothetical Composite Scenario)

Security researchers and incident reports have documented a pattern of attacks where threat actors exploit the account recovery flows of MFA systems. One such tactic, seen in the wild with groups like the 0ktapus/Scattered Spider group, is to **claim a registered device is lost** in order to add a new one. In a hypothetical but well-grounded scenario, an attacker who has a target's password (say, via phishing) finds that the account is secured by a FIDO2 key or mobile passkey. Rather than trying to steal or clone the key, the attacker initiates a **call to the service's help center** – for example, a corporate IT helpdesk or a customer support line for a cloud service – pretending to be the legitimate user. The attacker might say, "I've lost my security key and can't log in," or "My phone with the authenticator app was stolen." They then ask the support agent to help them regain access.

If the support process is weak, the agent might **enroll a new authenticator** for the user as a solution. The attacker conveniently provides their own FIDO device or app instance details, which gets added to the account as a new second factor (often support will do this after some verification questions). Alternatively, the support might issue a **temporary one-time passcode** or backup recovery code over the phone or email, which the attacker can use. In either case, the attacker has now effectively **inserted their own device** into the victim's account or obtained a login code, thereby **bypassing the original security key**. This technique was noted in the context of the 0ktapus campaign: when direct MFA prompt attacks failed, the attackers resorted to calling the service desk, claiming the phone was lost, and requesting a new MFA method enrollment.

Once the attacker's device is enrolled, the attacker can log in with the stolen password and their own second factor, achieving full account access. The legitimate user's FIDO key is still "technically" registered, but it has been superseded by the attacker's control of the account. This scenario highlights a crucial oversight in some passwordless deployments: the **account recovery path**. No matter how secure the primary authentication is, **people will make mistakes or lose devices**, so systems have fallback options. Attackers know to target these. It's a hypothetical scenario, but it mirrors real breaches and is entirely plausible anywhere helpdesk staff are trained to assist locked-out users quickly rather than verify rigorously. For both consumers and enterprises, the takeaway

is that **recovery and override procedures must be as hardened as the login process**. This might mean requiring users to physically present ID for identity verification, having managers approve employee MFA reset requests, or using multiple challenge-response queries that are not easily guessed from OSINT. FIDO-based authentication significantly raises the bar for attackers, but the **soft underbelly is often human support**. As one CISO noted about such attacks, every security control *"will have failure modes"* – and social engineering finds those modes. Robust training and processes are needed to ensure a convincing phone call doesn't undo an otherwise secure passwordless system.

Sources: The examples above are drawn from a combination of real incident reports and security research. Notable references include disclosures of YubiKey's side-channel flaw, NinjaLab's Titan key attack paper, Google's advisory on the Titan BLE vulnerability, NCC Group's BLE relay research, expert commentary on Titan supply-chain risks, the DigiNotar and TurkTrust CA compromises, and reports of the MGM Resorts breach and related social engineering techniques. These cases collectively illustrate how FIDO/passwordless authentication, while very strong against traditional phishing, can be threatened by attacks that target the hardware token itself, the communication channels, the supply chain, the underlying certificate infrastructure, or the human support layers. Each example serves as a caution that **no security system is invulnerable**, and holistic defenses are required to cover these unconventional attack vectors.

Quantitative Impact Assessment

Token Purchase & Lifecycle Cost per User/Endpoint

1. **Baseline hardware economics.** Enterprise-grade FIDO2 tokens retail between US $40 and $70 apiece, with bulk discounts rarely falling below US $35. If an organization adopts a two-token policy (one primary, one backup), the capital outlay begins at roughly US $70–140 per employee before any logistics are considered.

2. **Procurement and staging overhead.** Supply-chain teams must vet vendors, negotiate contracts, and perform acceptance testing. Industry benchmarks put these "soft" procurement costs at 7–12 % of the purchase price—adding another US $3–8 per token. For a 10 000-user deployment, paperwork alone can reach the mid-five figures.

3. **Distribution and onboarding.** Secure shipping, identity verification at pickup, and employee training typically add US $10–15 per token. Remote or international staff increase this figure because customs declarations and regional couriers raise both cost and risk of loss in transit.

4. **Attrition and replacement.** Annual loss or damage rates hover between 3 % and 5 % for physical authenticators. At a median replacement cost of

US $50, a 10 000-token fleet incurs US $15 000–25 000 in new hardware every year, compounding over the expected five-year lifecycle.

5. **Inventory management and end-of-life.** Tokens are considered controlled assets. Maintaining custody records, performing periodic audits, and executing secure disposal each consume administrative time. Gartner estimates the fully loaded lifecycle management expense at 20–30 % of capital cost—another US $7–11 per device, or US $70 000–110 000 for our 10 000-user sample.

6. **Contrast with witness-based keyless schemes.** ENI6MA's stateless, software-only witness derives keys on demand; there is no hardware to buy, ship, or retire. Deployment costs collapse to distributing a cryptographically signed client binary (or library) plus occasional software updates—often absorbed by existing MDM pipelines. Over five years, organizations routinely see a 60-70 % reduction in per-user authentication CAPEX when migrating from token-centric models to keyless architecture.

Help-Desk Ticket Volumes for Lost Devices

1. **Ticket incidence.** Large enterprises report 0.25–0.4 lost-token incidents per user, per year. For 10 000 users, that is 2 500–4 000 help-desk calls annually, each requiring identity verification and credential re-binding.

2. **Time-on-task.** Industry service-desk metrics place the average "lost credential" call at 18–22 minutes, including user verification against HR records, revocation of the missing token, issuance of a replacement seed, and follow-up confirmation.

3. **Direct labor cost.** At a blended help-desk labor rate of US $35/hour, every 1 000 such tickets cost roughly US $10 500. Thus a 4 000-ticket year consumes US $42 000 in direct support wages, exclusive of managerial overhead or after-hours premiums.

4. **Opportunity cost and productivity drag.** End-user downtime while waiting for replacement credentials is non-trivial. Even a conservative ten-minute average interruption equates to 667 lost labor hours per 4 000 tickets—roughly US $33 000 in "hidden" productivity loss at a US $50/hour loaded employee rate.

5. **Residual security exposure.** Each ticket opens a social-engineering window: attackers routinely spoof identity to request token re-issue. Mitigations (out-of-band verification, supervisor approval) extend ticket duration and cost while still not fully eliminating risk.

6. **Keyless reduction.** In ENI6MA, witness secrets are re-derived, not reissued. A user who misplaces a device reinstalls the client, performs one witness handshake, and regains access—often in under two minutes and

without help-desk interaction. Early pilots show ticket volumes dropping by 80–90 % and mean-time-to-recover plunging from days to minutes, liberating support staff for higher-value work.

CAPEX / OPEX Comparison: Conventional PKI vs. Device-Bound Passwordless vs. Keyless Alternatives

1. **Conventional PKI baseline.** A mid-size enterprise operating an on-prem CA cluster with hardware security modules (HSMs) faces initial CAPEX for redundant HSMs (US $50 k–120 k each), CA servers, and secure facilities. Five-year capital depreciation routinely exceeds US $500 000.

2. **Ongoing PKI OPEX.** Annual certificate operations—issuance, renewal, revocation list distribution—demand specialist staff and 24 × 7 monitoring. Personnel plus software support average US $180 000–250 000 per year. Add disaster-recovery sites, audit preparation, and smart-card issuance, and the five-year OPEX climbs past US $1 million.

3. **Device-bound passwordless (FIDO) model.** While the CA burden is lighter (universal CAs are outsourced), hardware token CAPEX replaces HSM CAPEX. For 10 000 users with two tokens apiece and ancillary onboarding costs, five-year hardware spend is roughly US $1 million. Help-desk OPEX, replacement stock, and SaaS authenticator licensing push total operating cost toward US $1.4 million over the same horizon.

4. **Keyless ENI6MA model: CAPEX.** No dedicated crypto hardware is required. The only capital expense is optional: a high-availability witness-orchestrator running on commodity virtual machines (˜US $15 k initial build-out). All endpoints leverage existing CPUs for lattice operations, eliminating specialized tokens.

5. **Keyless ENI6MA model: OPEX.** Credential rotation is automated, certificate workflows vanish, and lost-device tickets largely disappear. Operational spend centers on routine software maintenance and cloud egress—typically US $60 000–80 000 per year for a 10 000-user fleet—yielding a five-year OPEX of ˜US $350 000.

6. **Aggregate comparison.** Summing CAPEX and OPEX across five years: **legacy PKI ≈ US $1.5 M+**, **device-bound passwordless ≈ US $2.4 M**, **ENI6MA keyless ≈ US $0.4 M**. That represents an 80 % cost reduction against modern "passwordless" and nearly 75 % against traditional PKI—with ancillary gains in agility, sovereignty, and machine-to-machine coverage that do not appear on the balance sheet but materially improve security posture.

Requirements for a Post-Password-less Future

A *post-password-less* future is not simply "better biometrics" or shinier hardware tokens.It requires a holistic re-platforming of identity away from stored secrets, device silos, and vendor-controlled trust.Stateless proofs, device-agnostic principals, sovereign custody, millisecond self-healing, and quantum-resilient cryptography together form a mutually reinforcing lattice—remove any one strut and the structure's security posture weakens.Designing for these five requirements today ensures that the authentication layer will still stand when the next cryptographic or operational shockwave arrives.

EliminateStoredSecrets — Stateless,Witness-DerivedProofs

The legacy risk. Passwords, long-lived API tokens, and even private keys cached in secure elements all share a fatal property: the secret exists at rest somewhere.Attackers therefore focus on exfiltration—memory scraping, firmware backdoors, phishing for escrow codes—because compromise of a static secret yields indefinite access.

Witness-derived authentication. A stateless model flips that logic.Instead of proving identity by revealing or unlocking a stored secret, the party proves possession of an *ephemeral witness*—a short-lived random value generated on demand, consumed exactly once, and mathematically bound to both parties' view of the session transcript.Zero-knowledge proofs, one-round STARKs, and signature-of-knowledge techniques let the prover convince the verifier without ever persisting a private key.

How it works in practice. During handshake, each side derives a witness from environmental entropy (e.g., attestation measurements, TPM PCRs, or side-channel-safe DRBGs), signs the conversation transcript under this witness, and immediately erases it.All that survives is a public audit record—sufficient for non-repudiation but useless to an attacker.Subsequent sessions regenerate fresh witnesses, so even full memory capture yields nothing reusable.

Removing the honey pot. Because no long-term secret is present, traditional pivot tactics—pass-the-hash, pass-the-ticket, credential dumping—become dead ends.Attackers must now break live cryptography *in real time*, a far narrower window that demands vastly more resources.

Operational ripple effects. Central password vaults shrink or vanish, breach impact is limited to the active session, and compliance regimes (PCI-DSS, HIPAA, CJIS) can mark "stored credential" controls as *not applicable*.Incident responders move from privileged-account resets to simply expiring offending transcripts.

Implementation roadmap. Start by layering a witness-derived proof (e.g., Schnorr NIZK or Rosario-Wang proof-of-identity) atop existing TLS mutual auth; then graduate to a fully stateless cipher suite where each packet carries its own self-contained proof.Key escrow services downgrade to optional recovery metadata, not operational dependencies.

Remaining challenges. Stateless designs demand precise entropy hygiene, high-performance proof systems, and new logging schemas.But once solved, they strike at the root cause of most credential breaches: secrets that sit still long enough to be copied.

Device-AgnosticIdentities for Humans *and* Machines

The binding dilemma. Conventional "passwordless" binds identity to a phone, YubiKey, or TPM.Lose the gadget and your identity evaporates; need to authenticate a headless service or drone and you are stuck emulating a fingerprint swipe.

Abstract identities. A device-agnostic model treats the human or workload as the *principal*, not the hardware.Credentials become portable capability objects—verifiable presentations, decentralized identifiers (DIDs), or cryptographic capability tokens—anchored in enterprise trust roots rather than a consumer sync cloud.

Technical enablers. Deterministic key-derivation (HDKF), threshold signatures, and hardware-rooted attestation let any compliant host mint a fresh authentication witness on the user's behalf, provided it can prove policy conformity (e.g., secure-boot, geo-fence, or runtime posture).No single device owns the key; several can co-sign or re-derive it under policy.

Human vs. machine flows. For people, the abstraction means you can authenticate from a borrowed laptop, a kiosk, or a newly imaged phone after a factory reset—so long as the platform can satisfy the enterprise's posture proof.For machines, Kubernetes pods, satellites, and PLCs self-issue short-lived credentials at boot without ever storing a master secret.

DevOps and IoT gains. CI pipelines stop checking secrets into Git; instead, build agents request a time-boxed capability tied to the build job's hash.Factory robots and smart meters rotate identifiers each shift, nullifying device theft as an attack vector.

Governance lifecycle. HR off-boarding or a revoked serial number updates a single identity record; all descendant capabilities instantly fail verification."Bring-your-own-device" policies become tractable because trust follows the user, not the handset.

Road to adoption. Enterprises can pilot device-agnostic IDs by layering a verifiable-credential wallet atop their IdP and migrating service-to-service calls to SPIFFE/SPIRE or mutual-TLS with short-term certs.Gradual decommissioning of device-pinned secrets follows proof of parity.

SovereignKeyCustody — Enterprise-ControlledTrust Anchors

Why sovereignty matters. When Apple, Google, or a public CA hosts the trust anchor, subpoena, policy change, or cloud breach in *their* domain can invalidate *your* security overnight.Sovereign custody restores alignment between the party that bears the risk and the party that controls the keys.

Architectural patterns. Host root keys in on-prem or sovereign-cloud HSM clusters, replicate across multiple jurisdictions under your legal umbrella, and use CA/B Forum-compliant subordinate CAs only as *leaf* token issuers—never as the root of trust.

Distributed trust anchors. Threshold cryptography or multi-party computation lets three-of-five data centers jointly sign new roots, preventing a sin-

gle rogue admin or nation-state from hijacking credentials.Audit logs are anchored to an enterprise-run transparency ledger instead of CT logs controlled by browser vendors.

Compliance and data residency. Sovereign custody simplifies adherence to GDPR, SchremsII, and sectoral regulations (ITAR, CJIS) by proving that cryptographic material never leaves approved soil.Cloud vendors become commodity bandwidth, not gatekeepers of identity.

Key lifecycle automation. Enterprise PKI orchestration rotates roots on a fixed cadence, publishes revocation by reference (OCSP stapling or stapled proofs), and garbage-collects expired certificates without human tickets.Disaster recovery drills become scripted quorum re-seals of HSM shards.

Federation without surrender. Cross-org SSO still works via reciprocal trust bundles or mesh gateways, but each party validates against its own sovereign anchor first.If a partner is compromised, revocation stops at the interop boundary—no global "kill switch" risk.

Migration steps. Start by mirroring existing public CAs into an enterprise transparency log, then progressively re-issue internal workloads under a private root.Phase two introduces threshold-signed device attestations and deprecates external KMS dependencies in favor of on-prem shards.

Self-HealingRotation&RecoveryinMilliseconds

The rotation imperative. Static credential rotation cycles measured in days or weeks create exposure windows attackers exploit.A post-password-less fabric must treat key rollover as routine as TCP sequence-number increments.

Cryptographic levers. Ratcheting protocols (Double-Ratchet, OPAQUE), puncturable encryption, and key-evolving signatures let each handshake consume a one-time sub-key derived from forward-secure state.Compromise of today's material offers zero leverage tomorrow.

Autonomous recovery. If a device is lost or a container image is rolled back, it simply negotiates a fresh witness using its platform attestation; no help-desk ticket, no shared secret to reset.In headless environments, auto-rotation logic runs as part of the init container or bootloader.

Incident-response revolution. Breach containment shifts from mass password resets to cordoning off compromised transcripts.SOC analysts quarantine a suspect workload, delete its short-term capability, and the rest of the fleet continues unhindered.

Performance envelope. Millisecond-scale rotation requires microsecond-level key derivation and minimal handshake RTT: think two message flights, sub-2KB proofs, and no heavy lattice KEM on every packet.Edge caches and QUIC's 0-RTT patterns show the way.

Operational metrics. Mean Credentials To Live (MCTL) replaces Mean Time To Detect as the KPI: shorter is safer.Recovery SLAs can be expressed as cryptographic constants, not human processes—"$<$5ms to re-key, $<$10ms to revoke globally."

Practical rollout. Begin by layering ratcheting channels inside service meshes (e.g., WireGuard + Noise-XX), measure handshake latency, then deprecate older TLS sessions.Next, wire auto-rotation hooks into device management

so lost hardware self-invalidates on heartbeat failure.

Quantum-ResilientCryptography Baked into the Handshake

Looming quantum threat. Once fault-tolerant quantum computers reach ˜4000 logical qubits, Shor's algorithm can break RSA-2048 and elliptic-curve cryptography within hours.A handshake negotiated *today* may be recorded and broken *tomorrow*, exposing long-term secrets.

Native PQ primitives. Embedding lattice-based KEMs (Kyber, FrodoKEM) for key exchange and hash-based signatures (XMSS, SPHINCS+) for proof-of-identity ensures that even a quantum adversary cannot reconstruct session keys or forge transcripts.These algorithms run in software on today's CPUs, easing deployment.

Hybrid agility. While the ecosystem transitions, dual-stack "hybrid" ciphersuites combine classical ECDHE + PQ-KEM; if either stands, the session is secure.This hedges against potential cryptanalytic surprises in first-generation PQ candidates.

Bandwidth & performance trade-offs. PQ public keys are larger—up to 1-2KB for Kyber-768 and tens of KB for Dilithium signatures—but within tolerances for 5G, Wi-Fi6, and satellite links.Handshake RTT dominates user-perceived latency, so extra bytes seldom matter outside sensor networks.

Standards & compliance. NIST's PQC Round4 selections (2024), NSA's CNSA2.0 profile (2025), and draft FIPS203/204 supply government-grade benchmarks.Embedding these schemes at the *handshake layer* simplifies future recertification: swap one algorithm constant and recompile.

Forward-compatibility. A quantum-resilient handshake gains if the rest of the stack keeps pace—e.g., audit logs guarded by Merkle-tree commitments under PQ hashes, threshold HSMs upgraded to lattice KEM firmware.The goal is end-to-end PQ chain-of-custody, not just a PQ TLS cipher.

Migration blueprint. Enable hybrid TLS ciphersuites between corporate gateways first, monitor handshake success, then push requirements down to client libraries and IoT firmware.Deprecate pure-classical ciphers via policy flags, and set an internal "quantum-safe by default" date that predates external mandates.

12. Alternative Architectural Approaches

12.1 **Keyless, Stateless Cryptography (e.g., ENI6MA)** – Zero-knowledge witness proofs; lattice-based key agreement.
12.2 Comparative matrix: FIDO vs. Keyless across attack vectors, cost, scalability, and offline support.
12.3 Migration roadmap: coexistence layer, phased cut-over.

13. Implementation Guidance & Roadmap

- Pilot milestones, integration touch-points, metrics for success.
- Compliance alignment (NIST SP800-63, FIPS-140-3).
- Governance & change-management playbook.

14. Conclusion

- Recap systemic shortcomings of current password-less strategies.
- Urge industry to pursue device-independent, keyless designs before widespread quantum and supply-chain threats render FIDO inadequate.

15. Appendices

- **A. Glossary & Acronyms**
- **B. Threat Matrix Tables**
- **C. Survey of Public FIDO Vulnerabilities & Research**
- **D. Bibliography & Standards References**