International Journal of Innovative Research in Engineering
Volume 7, Issue 4 (July - August 2026), PP: 46-55.
https://www.doi.org/10.59256/ijire.20260704008
www.theijire.com
ISSN No: 2582-8746

Secure Password Management in Terminal Environments Using Encrypted USB Storage to Mitigate Key logging

Intisar A. Mohamedali[1], Yahia Fadlalla[2]
[1]*PhD Candidate, Faculty of Computer Science, Sudan University of Science and Technology Khartoum, Sudan.*
[2]*Professor and Lead Researcher/Consultant InfoSec Consulting Hamilton, Ontario, Canada.*

How to cite this paper:
Intisar A. Mohamedali[1], Yahia Fadlalla[2*], "Secure Password Management in Terminal Environments Using Encrypted USB Storage to Mitigate Key logging", IJIRE-V7I4-46-55.

Copyright © 2026 by author(s) and Fifth Dimension Research Publication. This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Abstract: *Securing a user's input is vital for protecting their privacy and credentials during authentication and for preventing credential theft or leakage. Key logging attacks represent a persistent and evolving threat to authentication security in terminal environments, where malicious software intercepts keystrokes to capture sensitive credentials. Our research proposes a hardware-dependent trust model that uses encrypted USB mass storage devices as secure and trusted vaults to deliver a user's credentials through a protected channel. By implementing a virtual disk-encrypted volume with AES-256 encryption and a Win USB-based secure communication path, the system separates authentication from the host's standard keyboard input path and reduces exposure to software-based keyloggers operating at both the user and kernel levels. This approach addresses critical vulnerabilities in conventional password managers, including "leaky autofill" and memory-based credential exposure, while maintaining usability through automated, site-specific injection mechanisms.*

Key Words: *keylogging mitigation; encrypted USB storage; hardware security module; Human Interface Device injection; credential injection; Trusted I/O Path; Physical Unclonable Functions; zero-trust authentication.*

I.INTRODUCTION

Authentication constitutes the foundational security process for verifying user identity before granting access to protected resources in terminal environments. However, reliance on manual credential entry creates a persistent vulnerability to malicious keylogging software, which covertly monitors the Human Interface Device path to capture and exfiltrate keystrokes before they are processed by target applications (Elelegwu et al., 2024), (Carr & Shahandashti, 2020). These keyloggers act as user-mode hooks or sophisticated kernel-level drivers, bypassing traditional defensive layers by intercepting raw input events and transforming password entry into a structural risk to authentication credential confidentiality (Singh et al., 2023).

The limitations of software-reliant authentication have prompted a strategic shift toward hardware-anchored security models (Wang et al., 2012). Software-based solutions, such as standard browser password managers, have their own constraints because they operate within the same vulnerable software stack as the operating system and user-mode applications (Ajibi, 2026). As a result, the industry has moved toward models that decouple user identity from host machines by tethering credentials to secure physical hardware tokens that serve as roaming identity vaults (Huseynov, 2020).

This research addresses these challenges by proposing a hardware-isolated architecture that utilises encrypted USB storage to interface with terminals, effectively mandating dual-possession authentication that prevents unauthorised access even during local machine compromise (Deshmukh et al., 2024). By offloading credentials to an isolated hardware medium, the system ensures that sensitive authentication strings are not exposed to host memory during the input process, thereby reducing threats posed by kernel-level and user-space keyloggers.

1.1 Problem Statement

Conventional software-based defences are proving increasingly inadequate against sophisticated attack vectors (Ajibi, 2026). The first point of possible failure is the persistence of manual credential entry, that could provide a path for keyloggers to capture sensitive data (Sharma, 2023). In addition, the emergence of HID-based injection attacks, such as "BadUSB" and the "Rubber Ducky," exploits the "automatic trust" modern operating systems grant to Human Interface Devices (Yadav, 2025). Because these devices masquerade as legitimate keyboards, they can inject malicious keystrokes at speeds far beyond human capability, often bypassing standard antivirus software that scans storage rather than firmware (Bintang et al., 2025). Furthermore, existing software-based password managers demonstrate vulnerabilities such as "leaky autofill," where credentials are inadvertently filled into hidden or deceptive form fields (Fu & Wang, 2024). These defensive systems operate at the same privilege level as the malware they seek to mitigate, allowing attackers with administrative rights to manipulate or terminate them (Ajibi, 2026).

1.2 Research Objectives

The primary aim of this research is to design and validate a hardware-enforced credential submission framework that preserves operational transparency while mitigating the effectiveness of kernel-level monitoring tools (Dhar et al., 2020). To achieve this, the framework establishes a trusted communication channel that protects the confidentiality, integrity, and authenticity of data exchanged between the security-hardened peripheral and the host environment.

Our research defines the primary objectives as follows:

- To design a roaming identity token that decouples credential storage from the host machine's memory (Wang et al., 2012)
- To implement a secure storage procedure using an encrypted virtual disk volume and user-defined password protection (Meng, 2006)
- To rigorously test the efficacy of the proposed architecture against modern HID injection threats such as "BadUSB" (Bintang et al., 2025)

II. RELATED WORK

2.1 Password Security Challenges

Credential management remains a foundational element of digital security, but reliance on manual entry introduces significant risks. Conventional password managers often struggle to balance accessibility with strong protection, creating centralised points of vulnerability (Fu & Wang, 2024).

Empirical analysis has shown that the majority of password manager applications store plaintext passwords in system memory during runtime, with 75% of desktop applications, 100% of browser-integrated managers, and 75% of browser plugin extensions leaking such sensitive information into accessible RAM regions, a practice formally registered as CWE-316 (Chatzoglou et al., 2024a). Similar forensic examinations of widely deployed managers such as KeePass, Password Safe, and RoboForm have consistently recovered unencrypted credentials from clipboard buffers, temporary folders, page files, and recycle bins, demonstrating that conventional vaults fail to sanitise volatile artefacts of credential operations (Chatzoglou et al., 2024c).

2.2 Keylogging and HID-Based Threats

Modern threat models for portable storage extend beyond simple data theft to incorporate adversarial USB devices that leverage HID emulation to exploit the inherent trust operating systems place in peripheral input (Neuner et al., 2018), (Hernandez et al., 2017).

The interception of keystrokes constitutes a critical authentication breach, as HID emulation allows attackers to capture sensitive credentials at the peripheral level, necessitating robust countermeasures, including hardware-based solutions, that decouple input from the host environment (Kang & Saiedian, 2017).

Modern password managers face an expanding matrix of attacker techniques that undermine their advertised security guarantees. A crucial vector is the leakage of saved credentials during autofill operations within WebView controls, where the AutoSpill technique demonstrates that most Android managers are vulnerable even when the host application is not actively participating in the disclosure (Gangwal et al., 2023). This is compounded by permission-based attacks that leverage malicious browser extensions capable of reading clipboard contents, stealing token-bearing cookies and URL parameters, and capturing user keystrokes—capabilities that have been observed across malicious browser extensions (Varshney et al., 2017).

2.3 Hardware-Based Security Approaches

Modern authentication techniques remain susceptible to HID masquerading attacks, wherein malicious devices deceive the host by falsely advertising keyboard capabilities to inject arbitrary payloads (Dumitru et al., 2022). Mitigating these risks requires behaviour-based detection and authentication of devices as legitimate input sources before interacting with the host operating system (Khande, 2023). To protect stored credentials, robust encryption is essential, such as the use of virtual disk volumes, where data is safeguarded by an encrypted header and strong user-defined passwords. This approach ensures that even if the physical storage is compromised, the sensitive credential material remains inaccessible without the correct cryptographic keys.

To defend against memory-based attacks, credential management should be isolated from the host machine's volatile memory through trusted execution environments (Xu et al., 2022). By employing hardware-bound keys and secure storage procedures, authentication material is protected from interception and extraction, ensuring that keys are only utilised in authorised, secure platform contexts (Allen et al., 2020). Traditional defensive software often operates within the same privilege level as the malware it seeks to defend against, making it vulnerable to manipulation or termination (Yamauchi et al., 2020). By offloading sensitive authentication processes to a secure, isolated environment or dedicated processor, the architecture ensures that the authentication path is shielded from compromised kernel environments, effectively neutralising threats that target standard input buffers (Zhou, 2014).

2.4 A Comparison with Industry Protocols

Integrating physical components such as Physical Unclonable Functions or dedicated USB keys provides an additional layer of security by linking authentication to a unique, tamper-evident hardware identity (Sultan & Banday, 2019). Unlike industry-standard frameworks such as FIDO2, which focus on replacing legacy credentials with public-key

infrastructure—a process requiring active, comprehensive support from relying parties (Alqubaisi et al., 2020)—this architecture addresses the persistent dependency on legacy password-based authentication in enterprise environments where FIDO2 implementation remains impractical or unsupported (Koshiy, 2025). Moreover, while FIDO2 offers enhanced security, recent analyses have identified potential vulnerabilities related to kleptographic attacks and evidence-based stability concerns, suggesting that standalone, hardware-isolated credential managers provide a necessary alternative in specialised security contexts where protocol-level dependencies must be minimised (Kutyłowski et al., 2024).

2.5 Detection of HID Masquerading in Endpoint Security

Modern endpoint security solutions increasingly employ behavioural analysis to mitigate the risks posed by Human Interface Device masquerading, where malicious peripherals emulate authorised input devices to facilitate unauthorised command injection (Fakiha, 2024). Unlike traditional signature-based detection, which often fails to identify sophisticated emulated devices, advanced monitoring systems leverage machine learning classifiers and process event graphs to analyse the chronological characteristics of input streams and device-generated event patterns (Huang et al., 2019). By rigorously monitoring these behavioural anomalies—such as improbable typing speeds or irregular HID descriptor data—at the hardware or deep operating-system layer, security frameworks can dynamically distinguish between legitimate user interactions and automated attack vectors, thereby reducing susceptibility to protocol spoofing (Zuyeva et al., 2022). Additionally, deploying Saila-based validation modules enables the real-time inspection of peripheral configuration descriptors against registered device profiles, effectively blocking unauthorised hardware before malicious HID injection can execute (Ghosh et al., 2024).

Unlike Saila, which relies on host-level behavioural analysis and peripheral descriptor validation to identify malicious hardware—a reactive approach that intervenes upon detection (Ghosh et al., 2024)—this architecture establishes a proactive security posture by physically isolating credential decryption and storage within an independent secure element. By offloading cryptographic operations to the dedicated USB controller, the system ensures that master keys and raw sensitive material remain entirely outside the host terminal's volatile memory, effectively mitigating vulnerabilities such as RAM scraping and cold-boot attacks that persist in traditional software-reliant credential management systems (Halderman et al., 2008).

III. SYSTEM DESGIN

3.1 Architecture Overview

This study proposes an architecture that uses a physically isolated, hardware-encrypted USB peripheral as an external, secure vault for credential storage.

This design leverages a dedicated microcontroller to manage cryptographic operations, ensuring that sensitive data is never exposed to the host operating system's kernel, which remains a primary target for malware (Vella et al., 2021). Furthermore, by implementing a mutually authenticated secure channel between the peripheral and the host, this architecture proactively addresses off-path USB injection attacks—where malicious intermediaries, such as compromised USB hubs, manipulate communications by injecting data onto the bus (Dumitru et al., 2022). By enforcing cryptographic binding and verifying the integrity of the data stream at the hardware level, the system ensures that even if an attacker introduces arbitrary packets into the communication channel, the host terminal only accepts commands originating from the verified, hardware-isolated device (Dumitru et al., 2022).

Detection paradigm	Reactive vs. Proactive	Trust-root layer	Source
Signature-based forensic scanning	Reactive	Host OS	(Fakiha, 2024)
Behavioural ML + process event graph	Reactive	Host OS	(Huang et al., 2019)
Peripheral parameter/firmware analysis	Semi-proactive	Host-side firmware inspection	(Zuyeva et al., 2022)
FIDO2 descriptor validation	Reactive (port disconnect on mismatch)	Host boundary	(Ghosh et al., 2024)
Proposed PUF-rooted WinUSB vault	**Proactive enforcement**	**Peripheral**	(Akshaya et al., 2025)

Table 1: Taxonomy of HID-masquerading detection paradigms.

Table 1 shows where existing HID-masquerading defences sit relative to one another on detection philosophy (reactive vs. proactive) compared with the proposed architecture.

This approach delegates the decryption process to the on-board microcontroller of the USB device, ensuring that sensitive master keys are never transferred into the host terminal's volatile memory (Vella et al., 2021). By offloading the decryption logic to a dedicated secure element, the system effectively mitigates the risks associated with RAM scraping and cold-boot attacks that affect traditional software-based vault architectures.

The interface between the peripheral and the terminal uses a protected USB communication path rather than relying solely on standard HID keyboard emulation. This reduces exposure to the operating system's standard input subsystem and mitigates the impact of sophisticated keylogging software. The proposed methodology utilises a Context-Aware Automated Injection architecture anchored on a secured USB device to manage multiple credentials securely, employing a dual-stage verification process that requires both physical token presence and a successful cryptographic challenge-response exchange

before credential delivery (Tritilanunt et al., 2014).

3.2 Virtual Disk Volume Management

A virtual disk encrypted volume is established by writing an encrypted identification mark at the head of the storage device. The virtual disk information header—encapsulating encryption keys, operating strategies, and file structure details—is secured by a user-defined password (Data Encryption Mobile Storage Management Procedure Based on Virtual Disk, 2023). Access is granted only upon successful header verification; otherwise, the device is recognised as an ordinary storage medium.

This virtual disk is mapped to a dedicated, encrypted partition within the USB device's non-volatile flash memory, mitigating risks associated with physical theft or unauthorized hardware access (Cui, 2013). To further protect the credential store, the implementation employs an AES-256-XTS cipher suite (Swierczynski et al., 2016), which mandates a multi-factor authentication sequence before the decryption of the internal key vault (He et al., 2014).

The system architecture secures operations by leveraging an internal hardware controller to execute cryptographic tasks, providing chip-level protection for stored data (Guo-song, 2012). Unlike host-reliant secure I/O channels that depend on software-based enclaves, this design enforces a strict trust boundary by physically offloading cryptographic operations to a dedicated secure element, effectively decoupling sensitive key material from the host terminal's volatile memory and kernel-level vulnerabilities.

In addition, the peripheral integrates a secure key agreement protocol that establishes a protected cryptography session after authentication, preventing unauthorised data exfiltration through protocol interception. By dynamically rotating transient session keys during each communication cycle, the system ensures that even if an attacker captures legitimate traffic via an off-path intermediary—such as a compromised USB hub—the intercepted data remains unreadable to the adversary. This architecture further incorporates hardware-level anti-replay mechanisms that validate the sequence and integrity of each packet (He et al., 2014), forcing the host terminal to discard any intercepted or maliciously injected command streams that lack a valid, cryptographically verified session token, thus effectively neutralising bus-level injection attempts.

3.3 Secure Peripheral Interface

The proposed interface uses a dedicated microcontroller to expose a protected WinUSB/raw bulk endpoint rather than emulating a standard keyboard as its primary credential-delivery path (Dhar et al., 2020). To mitigate protocol masquerading, a custom USB firewall validates all traffic (Tian et al., 2016). Furthermore, the system incorporates Physical Unclonable Functions to establish a unique trust root, complicating reverse engineering (Bhatta et al., 2024). The implementation also utilises TPM 2.0 PCR binding to ensure the cryptographic environment remains uncompromised during the authentication sequence (Ajibi, 2026).

To further harden the interface against man-in-the-middle attacks, the system enforces a policy-based mediation layer that cross-references the device's unique physical signature with expected operational characteristics, preventing unauthorised peripheral interaction (Ibrahim et al., 2020).

Considering ageing processes such as Bias Temperature Instability and environmental variations that can increase Bit Error Rates and jeopardise the reliability of the hardware-derived identity vault over time (Bhatta et al., 2024), the system implements proactive stabilisation mechanisms. The architecture integrates a stability-aware bit-selection algorithm that identifies and masks marginally reliable SRAM cells during initial enrolment, combined with lightweight error-correcting codes to rectify residual bit flips throughout the device's operational lifespan (Xiao et al., 2014).

Additionally, the authentication sequence employs RSA-based digital signatures to guarantee the authenticity of the file structure, ensuring that firmware-level data remains immune to unauthorised modifications during transit (Yadav & Patel, 2026).

IV. IMPLEMENTATION

4.1 Automated Injection Mechanisms

To bypass manual typing and its associated risks, the system implements two primary strategies:

Legacy HID Emulation: A conventional USB password stick may emulate a Human Interface Device to type passwords directly into form fields, but this path remains exposed to the host input subsystem and is therefore treated as a legacy comparison baseline rather than the primary secure delivery mode (Yadav, 2025).

Secure Channel Injection: In web environments, the proposed system uses a protected WinUSB channel between the peripheral and a trusted browser extension. The extension releases the actual credential only after domain verification and request-time validation, reducing the risk that malicious client-side scripts, deceptive forms, or extensions can exfiltrate the real password from the field (Gautam et al., 2024).

Legacy baseline: USB device ⟶ Host OS keyboard buffer ⟶ Browser form
Proposed path: USB enclave ⟶ Raw bulk USB / WinUSB ⟶ Trusted extension ⟶ Verified form injection

Protocol Obfuscation: Where HID-style fallback is required, the injection routine can employ randomised inter-character delays and timing variations to mimic human typing patterns, thereby reducing the likelihood of detection by simple threshold-based behavioural analysis algorithms (Shadman et al., 2025). As illustrated in experimental results, the frequency

distribution of inter-character dwell times for automated input can approximate the stochastic nature of human input, with a variance of 15–20% compared to typical baseline user profiles (Alsultan et al., 2017).

While this intentional jitter can disrupt standard threshold-based detection—which typically flags static, millisecond-precise timing patterns—it is increasingly challenged by modern AI-based behavioural analysis (Condrey, 2026). Consequently, the implementation treats timing variation as a supplementary fallback measure rather than the primary security control.

4.2 Site-Specific Credential Mapping

To prevent credential cross-contamination across various domains, the system is designed to employ a localised hashing algorithm that generates unique identifier keys based on the current Uniform Resource Locator (URL) and an internally stored salt. To support automated filling, the USB device maintains an encrypted database where each entry is mapped to a specific URL or terminal application identifier (Wang et al., 2012). A lightweight secure broker identifies the active domain and queries the USB device, ensuring only the relevant password is retrieved for the current context (Liang et al., 2018). This mapping ensures that the roaming manager provides site-specific credentials without manual user selection.

V. RESULTS AND DISCUSSION

5.1 Throughput and Physical Characterization

We benchmarked the prototype's operational overhead to determine if a hardware-isolated cryptosystem introduces noticeable user delay. Across 100 consecutive authentication cycles, the mean latency measured from the initial user-presence trigger to final credential delivery was 115.4 ms (±12.3 ms).

Rather than relying on traditional static key storage configurations—which remain highly susceptible to non-volatile memory (NVM) or one-time programmable (OTP) physical tampering, such as focused ion beam probing and laser voltage analysis—the architecture derives its root key dynamically using an on-chip SRAM Physical Unclonable Function (PUF).

To verify that these cryptographic profiles are uniquely bound to individual silicon instances, we evaluated the inter-class Hamming Distance (HD) across our batch of 100 test units. The experimental evaluation yielded a mean fractional HD of 50.08%. Because a mathematically ideal, completely random hardware identity across independent chips targets an inter-class HD of exactly 50%, these metrics confirm that the generated keys are structurally unclonable, preventing database cloning even across identical chip manufacturing runs. To validate long-term reliability against silicon aging, a Bit Error Rate test was conducted across a temperature range of -25°C to 70°C and varying supply voltages. In accordance with methodologies established in recent research (Bhatta et al., 2024), these trials confirm that the PUF-derived keys remain stable under diverse environmental operating conditions.

To complement this static isolation, the device integrates an on-chip True Random Number Generator (TRNG) driven by the thermal noise of ring oscillators (Williams et al., 2024). By generating continuous high-entropy cycles for every session token and initialisation vector, the system prevents machine-learning-based side-channel modelling attacks that target algorithmic pattern analysis.

Authentication scheme	Mean latency (s)	Source
Proposed prototype	**0.1154**	Own measurement (100 cycles)
Picosibling (range, low end)	~1.0	(Goldberg et al., 2016)
Picosibling (range, high end)	~3.0	(Goldberg et al., 2016)
Gauth	3.8	(Damopoulos & Kambourakis, 2019)
OIDC + FIDO UAF, 95% CI parallel	~1.5–2.5	(Papadamou et al., 2019)

Table 2: Authentication-latency comparison with related systems.

Table 2 shows the mean authentication latency across hardware-mediated authentication schemes. The proposed prototype (115.4 ms ± 12.3 ms across 100 cycles) is compared with Gauth (3.8 s mean) (Damopoulos & Kambourakis, 2019), Picosibling (1–3 s end-to-end credential reconstruction) (Goldberg et al., 2016), and OIDC / FIDO UAF deployments whose 95% confidence intervals span comparable order-of-magnitude delays under parallel load (Papadamou et al., 2019). The prototype clears the 150 ms instantaneous benchmark because credential decryption and secure-channel delivery occur in a single on-device exchange without server round-trips.

PUF design	Mean inter-class HD	SD	Source
Proposed prototype	**0.5008**	(our measurement)	This work
mrSPUF (memristive, 128-bit)	0.4996	0.0363	(Gao et al., 2015)
S2RAM-PUF (subthreshold)	0.5007	not reported	(Ni & Zhang, 2024)
Theoretical binomial ideal, n = 128	0.5000	0.0456 (theoretical)	—

Table 3a: Underlying data — Mean inter-class Hamming Distance (HD).

PUF design	Temperature range	% stable cells	Source
Conventional SRAM PUF	−15 °C → 75 °C	~87%	(Samanta et al., 2025)
FinFET SRAM PUF (16 nm/14 nm/7 nm)	ambient	Comparable	(Masoumian et al., 2022)
S2RAM-PUF	−40 °C → 120 °C	100% (BER =0)	(Ni & Zhang, 2024)

Table 3b: Underlying data — Fraction of stable power-up cells over temperature.

Table 3: SRAM PUF inter-class Hamming Distance, stability, and Bit Error Rate across operating temperatures. Panels 3a and 3b provide the underlying comparative data.

Table 3 shows that the proposed prototype's 50.08% mean inter-class HD is competitive with dedicated PUF designs in the published literature, while also summarising temperature reliability. Safely routing credentials to a target form on a potentially compromised host operating system requires abandoning standard USB Human Interface Device (HID) keyboard emulation. Standard HID inputs pass directly into the unencrypted host OS keyboard buffer, making them trivial to intercept via kernel-level keyloggers or user-mode accessibility API hooks like SetWindowsHookEx.

To establish a verified Trusted I/O Path, the architecture treats the USB device as a raw vendor-specific bulk endpoint via WinUSB, bypassing the standard OS input subsystem. Rather than relying on host-side keyboard emulation, the cryptographic boundary is shifted to a mutually authenticated channel between the peripheral and a trusted browser extension.

Upon connection, a hardware enclave *on the USB peripheral* executes a persistent handshake with the host-side extension, establishing an out-of-band channel encrypted with AES-256-GCM.

This architecture mitigates off-path injection vectors—such as a compromised kernel dropping, modifying, or injecting malicious bulk packets—by enforcing strict sequence counters and mandatory cryptographic signature checks on every payload before processing.

Unlike traditional Raw HID implementations that remain susceptible to protocol-level spoofing because they still rely on the host's standard HID driver stack (Dumitru et al., 2022), the WinUSB design enforces a custom, secure communication channel. Plaintext credentials never traverse the untrusted host kernel or standard input buffers. Instead, the browser extension receives the encrypted payload, decrypts it within its isolated execution sandbox to mitigate runtime memory scraping, and populates target form fields via direct Document Object Model (DOM) injection. While this design presumes the integrity of the browser's process isolation model, it reduces reliance on the host OS input and peripheral subsystem layers.

To neutralise web-based credential-theft threats—such as leaky autofill vectors where third-party scripts exploit hidden form fields (Fu & Wang, 2024)—the browser extension performs a strict fully qualified domain name verification handshake with the hardware before any payload is released. Furthermore, to prevent background malware from silently monitoring the device (Nicho & Sabry, 2022), the enclave locks execution until the user manually bridges a physical capacitive trace on the device chassis to pass a hardware User Presence Test.

5.2 Comparative Security Analysis

To evaluate the structural advantages of our architecture, Table 4 contrasts the proposed hardware-rooted approach against conventional commercial USB credential injectors.

Target Attack Vector	Standard USB Password Stick (HID Emulation)	Proposed Hardware-Rooted Architecture	Architectural Mitigations
Physical Key Extraction	Vulnerable	Resilient	Keys are dynamically derived via SRAM PUF; zero keys exist at rest
Kernel-Level Keyloggers	Vulnerable	Protected	Bypasses HID subsystems entirely; routes AES-256-GCM ciphertext via raw USB bulk endpoints.
Phishing / Hidden Forms	Vulnerable	Protected	Enforces a cryptographic handshake and strict FQDN domain validation prior to payload delivery.
Silent Malware Spoofing	Vulnerable	Protected	Execution is hard locked by a physical capacitive hardware User Presence Test (UPT).
Side-Channel Modelling	Vulnerable	Protected	Employs an on-chip ring-oscillator TRNG to blind algorithmic execution signatures.

Table 4: Comparative security analysis of the proposed hardware-rooted architecture against standard USB password sticks.

The "Phishing / Hidden Forms" row of Table 4 marks the proposed architecture as **Protected**, in contrast to the empirical probability that an unprotected password manager will autofill credentials into a hidden, deceptive, or attacker-controlled field (Fu & Wang, 2024).

The Protected status of the proposed architecture against phishing and hidden-form attacks is consistent with the empirical scale of the threat. As shown on table 5 below, Fu and Wang reported that **all 30** studied password managers autofilled data into hidden fields in at least one form, with a **65.7%** probability specifically for login forms, and that built-in browser password managers leaked credentials into hidden fields **4.07 times** more often than separately installed password managers (Fu & Wang, 2024). Gangwal et al. demonstrated an analogous autofill leak on Android via WebView-based credential capture (Gangwal et al., 2023), while Silver et al. showed that several popular browser-default autofill policies allow a remote network attacker to extract multiple passwords with **zero user interaction** (Silver et al., 2014). The proposed design reduces this class of leak by enforcing a cryptographic handshake and strict FQDN domain validation before any payload is released; the autofill decision is bound to a verified website identity rather than inferred at the browser autofill layer.

Scenario	Probability / Ratio
All forms, overall (n = 1,032 scenarios studied)	**58.7%**
Login forms specifically	**65.7%**
Built-in browser password managers compared with separately installed password managers	**4.07× more likely**
Password managers leaking credentials in at least one scenario	**100% (all 30 studied)**

Table 5: Password-manager autofill leakage into hidden fields. Source: Fu & Wang (2024).

VI.CONCLUSION

This research demonstrates that an automated, site-specific injection system based on encrypted USB storage provides a robust defence against traditional keylogging and modern HID-based attacks. By prioritising physical isolation and cryptographic authentication, the framework secures authentication workflows against evolving patterns of behavioural-information leakage (Siahaan & Chowanda, 2022).

The proposed architecture successfully addresses critical vulnerabilities in conventional password managers, including "leaky autofill" and memory-based credential exposure, while maintaining usability through automated, site-specific injection mechanisms. However, recent research—such as 'A Deep-Learning-Based Approach to Keystroke-Injection Payload Generation'—indicates that deep learning models can generate effective synthetic forgeries (Gurčinas et al., 2023), suggesting that timing randomisation alone may be insufficient against state-of-the-art liveness detection systems.

The integration of hardware-rooted security primitives, including Physical Unclonable Functions and encrypted virtual disk volumes, establishes a trust anchor that remains resilient even when the host environment is fully compromised.

6.1 Future Work

Future research should prioritise the integration of keystroke dynamics as a behavioural biometric layer to ensure that the physical user is present during the automated injection process. This would allow the system to verify identity through natural typing patterns even when hardware performs the data entry (Ness, 2017). Additionally, future iterations should explore:

- Multi-factor proximity sensors (e.g., Bluetooth or NFC) to ensure the token remains within a secure radius (Shukla et al., 2024)
- Hardware-protected enclaves that mandate firmware integrity verification upon initialisation to counter sophisticated firmware manipulation (Elmaghbub & Hamdaoui, 2024)
- The refinement of detection mechanisms for peripheral enumeration to minimise race conditions (Sen et al., 2024)
- Integration of automated revocation protocols that trigger immediate data-at-rest encryption upon detecting unauthorized host-side peripheral disconnection (Griscioli & Pizzonia, 2021)

Incorporating post-quantum cryptographic primitives into the secure USB enclave will be vital to safeguarding stored credentials against the looming threat of quantum-enabled decryption (Patel, 2026). These advancements will contribute to a more resilient authentication trust framework, addressing both the vulnerabilities of HID-based injection and the broader security challenges posed by promiscuous USB device interactions in untrusted environments (Griscioli & Pizzonia, 2018).

References

1. Ajibi, J. (2026). Hardware-Rooted Tri-Factor Authentication for Kernel-Enforced Ransomware Protection: TPM Attestation, Device Fingerprinting, and Post-Quantum USB Tokens. In Zenodo (CERN European Organization for Nuclear Research). European Organization for Nuclear Research. https://doi.org/10.5281/zenodo.18989541
2. Akshaya, N., Namrutha, M., G, N. R., & Kavitha, C. R. (2025). Secure Offline Password Manager With AES-256-GCM and Honey Encryption for Brute-Force Attack Resilience. https://doi.org/10.1109/icact67549.2025.11351416
3. Allen, S. E., Čapkun, S., Eyal, I., Fanti, G., Ford, B., Grimmelmann, J., Juels, A., Kostiainen, K., Meiklejohn, S., Miller, A., Prasad, E., Wüst, K., & Zhang, F. (2020). Design Choices for Central Bank Digital Currency: Policy and Technical Considerations. In National Bureau of Economic Research. https://doi.org/10.3386/w27634
4. Alqubaisi, F., Wazan, A. S., Ahmad, L., & Chadwick, D. (2020). Should We Rush to Implement Password-less Single Factor FIDO2 based Authentication? 1–6. https://doi.org/10.1109/urc49805.2020.9099190
5. Alsultan, A., Warwick, K., & Wei, H. (2017). Non-conventional keystroke dynamics for user authentication. Pattern Recognition Letters, 89, 53–59. https://doi.org/10.1016/j.patrec.2017.02.010
6. Bhatta, N. P., Singh, H., Ghimire, A., & Amsaad, F. (2024). Analyzing Aging Effects on SRAM PUFs: Implications for Security and

Reliability. Journal of Hardware and Systems Security, 8(3), 174–186. https://doi.org/10.1007/s41635-024-00154-6
7. Bintang, R. M. G. S., Arizal, Purwoko, R., Syahral, M., Zen, S. S., Putra, I. G. M., & Wijayanti, R. A. (2025). SafeUSB: Implementation of Keystroke Monitoring System to Prevent BadUSB HID Injection Attack. 286–291. https://doi.org/10.1109/icocics68032.2025.11383963
8. Carr, M., & Shahandashti, S. F. (2020). Revisiting Security Vulnerabilities in Commercial Password Managers. In arXiv (Cornell University) (pp. 265–279). Cornell University. https://doi.org/10.1007/978-3-030-58201-2_18
9. Chatzoglou, E., Kampourakis, V., Tsiatsikas, Z., Καρόπουλος, Γ., & Kambourakis, G. (2024a). Keep Your Memory Dump Shut: Unveiling Data Leaks in Password Managers. In IFIP advances in information and communication technology (pp. 61–75). Springer Science+Business Media. https://doi.org/10.1007/978-3-031-65175-5_5
10. Chatzoglou, E., Kampourakis, V., Tsiatsikas, Z., Καρόπουλος, Γ., & Kambourakis, G. (2024b). Keep your memory dump shut: Unveiling data leaks in password managers. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2404.00423
11. Chatzoglou, E., Kampourakis, V., Tsiatsikas, Z., Καρόπουλος, Γ., & Kambourakis, G. (2024c). Unmasking the hidden credential leaks in password managers and VPN clients. Computers & Security, 150, 104298–104298. https://doi.org/10.1016/j.cose.2024.104298
12. Condrey, D. (2026). On the Insecurity of Keystroke-Based AI Authorship Detection: Timing-Forgery Attacks Against Motor-Signal Verification. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2601.17280
13. Cui, Q. (2013). Secure Storage Technology Based on Virtual File System. Microelectronics & Computer. http://en.cnki.com.cn/Article_en/CJFDTOTAL-WXYJ201306014.htm
14. Damopoulos, D., & Kambourakis, G. (2019). Hands-Free one-Time and continuous authentication using glass wearable devices. Journal of Information Security and Applications, 46, 138–150. https://doi.org/10.1016/j.jisa.2019.02.002
15. Data Encryption Mobile Storage Management Procedure Based on Virtual Disk. (2023). https://doi.org/10.48047/nq.2020.18.8.nq20241
16. Deshmukh, R., Petewar, P. R., Rathod, S. M., Bijwe, S. D., Pawar, V. D., & Bondre, T. S. (2024). Review on Proposal of a Password Manager, satisfying security and Usability through "Key-Master." International Journal of Innovative Science and Research Technology (IJISRT), 585–590. https://doi.org/10.38124/ijisrt/ijisrt24nov975.
17. Dhar, A., Ulqinaku, E., Kostiainen, K., & Čapkun, S. (2020). ProtectIOn: Root-of-Trust for IO in Compromised Platforms (Vol. 2019, p. 869). https://doi.org/10.14722/ndss.2020.24112
18. Dumitru, R., Genkin, D., Wabnitz, A., & Yarom, Y. (2022). The Impostor Among US(B): Off-Path Injection Attacks on USB Communications. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2211.01109
19. Elelegwu, D., Chen, L., Ji, Y., & Kim, J. (2024). A Novel Approach to Detecting and Mitigating Keyloggers. 1583–1590. https://doi.org/10.1109/southeastcon52093.2024.10500122
20. Elmaghbub, A., & Hamdaoui, B. (2024). Domain-Agnostic Hardware Fingerprinting-Based Device Identifier for Zero-Trust IoT Security. IEEE Wireless Communications, 31(2), 42–48. https://doi.org/10.1109/mwc.001.2300420
21. Fakiha, B. S. (2024). Forensic analysis of bad USB attacks: A methodology for detecting and mitigating malicious USB device activities. Edelweiss Applied Science and Technology, 8(5), 1090–1100. https://doi.org/10.55214/25768484.v8i5.1809
22. Fu, Y., & Wang, D. (2024). Leaky Autofill: An Empirical Study on the Privacy Threat of Password Managers' Autofill Functionality. 288–303. https://doi.org/10.1109/acsac63791.2024.00037
23. Gangwal, A., Singh, S., & Srivastava, A. (2023). AutoSpill: Credential Leakage from Mobile Password Managers. 39–47. https://doi.org/10.1145/3577923.3583658
24. Gao, Y., Ranasinghe, D. C., Al-Sarawi, S. F., Kavehei, O., & Abbott, D. (2015). Memristive crypto primitive for building highly secure physical unclonable functions. Scientific Reports, 5(1), 12785–12785. https://doi.org/10.1038/srep12785
25. Gautam, A., Yadav, T., Seamons, K., & Ruoti, S. (2024). Passwords Are Meant to Be Secret: A Practical Secure Password Entry Channel for Web Browsers. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2402.06159
26. Ghosh, A., Mitra, A., Chakkaravarathy, S. S., Priya, V., Anitha, S., & Babu, R. T. S. (2024). Saila: Human Interface Device (HID) Injection Protection with Smart Phone based Passwordless Security. 122–127. https://doi.org/10.1109/icdcsw63686.2024.00023
27. Goldberg, I., Jenkinson, G., & Stajano, F. (2016). Low-Cost Mitigation Against Cold Boot Attacks for an Authentication Token. In Lecture notes in computer science (pp. 36–57). Springer Science+Business Media. https://doi.org/10.1007/978-3-319-39555-5_3
28. Griscioli, F., & Pizzonia, M. (2018). USBCaptchaIn: Preventing (Un)Conventional Attacks from Promiscuously\n Used USB Devices in Industrial Control Systems. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.1810.05005
29. Griscioli, F., & Pizzonia, M. (2021). USBCaptchaIn: Preventing (un)conventional attacks from promiscuously used USB devices in industrial control systems. Iris (Roma Tre University). https://doi.org/10.3233/jcs-191404
30. Guo-song, F. (2012). A Design of Security USB2.0 Device Controller. Jisuanji Gongcheng. http://en.cnki.com.cn/Article_en/CJFDTOTAL-JSJC201224070.htm
31. Gurčinas, V., Dautartas, J., Janulevičius, J., Goranin, N., & Čenys, A. (2023). A Deep-Learning-Based Approach to Keystroke-Injection Payload Generation. Electronics, 12(13), 2894–2894. https://doi.org/10.3390/electronics12132894.
32. Halderman, J. A., Schoen, S. D., Heninger, N., Clarkson, W., Paúl, W., Calandrino, J. A., Feldman, A. J., Appelbaum, J., & Felten, E. W. (2008). Lest we remember: cold boot attacks on encryption keys. Munich Personal RePEc Archive (Ludwig Maximilian University of Munich), 43(5), 45–60. http://library.tue.nl/csp/dare/LinkToRepository.csp?recordnumber=860730
33. He, D., Kumar, N., Lee, J., & Sherratt, R. S. (2014). Enhanced three-factor security protocol for consumer USB mass storage devices. IEEE Transactions on Consumer Electronics, 60(1), 30–37. https://doi.org/10.1109/tce.2014.6780922
34. Hernandez, G., Fowze, F., Tian, D., Yavuz, T., & Butler, K. (2017). FirmUSB. arXiv (Cornell University), 2245–2262. https://doi.org/10.1145/3133956.3134050
35. Huang, C., Lee, H.-M., Wang, J.-C., & Mao, C.-H. (2019). Identifying HID-based attacks through process event graph using guilt-by-association analysis. 273–278. https://doi.org/10.1145/3309074.3309080
36. Huseynov, E. (2020). Improving user experience with TOTP hardware tokens by implementing QR codes and HID keyboard emulation. 23, 1–5. https://doi.org/10.1109/aict50176.2020.9368574
37. Ibrahim, O. A., Sciancalepore, S., Oligeri, G., & Pietro, R. D. (2020). MAGNETO. ACM Transactions on Embedded Computing Systems, 20(1), 1–26. https://doi.org/10.1145/3422308
38. Kang, M., & Saiedian, H. (2017). USBWall: A novel security mechanism to protect against maliciously reprogrammed USB devices. Information Security Journal A Global Perspective, 26(4), 166–185. https://doi.org/10.1080/19393555.2017.1329461

39. Khande, Ms. S. R. R. (2023). Prevention of code injection from Human Interface Device (HID). Zenodo (CERN European Organization for Nuclear Research). https://doi.org/10.5281/zenodo.7795499
40. Koshiy, P. G. (2025). Human-Centric Passwordless Authentication: Beyond Technology to Workforce Transformation. International Journal of Computational and Experimental Science and Engineering, 11(4). https://doi.org/10.22399/ijcesen.4002
41. Kutyłowski, M., Lauks-Dutka, A., Kubiak, P., & Zawada, M. (2024). FIDO2 Facing Kleptographic Threats By-Design. Applied Sciences, 14(23), 11371–11371. https://doi.org/10.3390/app142311371
42. Liang, S., Yue, Z., Li, B., Guo, X., Jia, C., & Liu, Z. (2018). Secureweb: Protecting sensitive information through the web browser extension with a security token. Tsinghua Science & Technology, 23(5), 526–538. https://doi.org/10.26599/tst.2018.9010015
43. Masoumian, S., Selimis, G., Wang, R., Schrijen, G.-J., Hamdioui, S., & Taouil, M. (2022). A Reliability Analysis of FinFET-Based SRAM PUFs for 16nm, 14nm, and 7nm Technology Nodes. Research Repository (Delft University of Technology), 1189–1192. https://doi.org/10.23919/date54114.2022.9774735
44. Meng, G. (2006). File encryption approach based on virtual disk. Jisuanji Gongcheng Yu Sheji. https://en.cnki.com.cn/Article_en/CJFDTOTAL-SJSJ200615038.htm
45. Ness, J. (2017). Presentation Attack and Detection in Keystroke Dynamics [Vilnius University]. In BIBSYS Brage (BIBSYS (Norway)). http://hdl.handle.net/11250/2448954
46. Neuner, S., Voyiatzis, A. G., Fotopoulos, S., Mulliner, C., & Weippl, E. (2018). USBlock: Blocking USB-Based Keypress Injection Attacks. In Lecture notes in computer science (pp. 278–295). Springer Science+Business Media. https://doi.org/10.1007/978-3-319-95729-6_18
47. Ni, L., & Zhang, J. (2024). S2RAM PUF: An Ultra-low Power Subthreshold SRAM PUF with Zero Bit Error Rate. 1–6. https://doi.org/10.1145/3649329.3658246
48. Nicho, M., & Sabry, I. (2022). Threat and Vulnerability Modelling of Malicious Human Interface Devices. The Eurasia Proceedings of Science Technology Engineering and Mathematics, 21, 241–247. https://doi.org/10.55549/epstem.1225679
49. Papadamou, K., Gevers, S., Xenakis, C., Sirivianos, M., Zannettou, S., Chifor, B., Ţeican, S., Gugulea, G., Caponi, A., Recupero, A., Pisa, C., & Bianchi, G. (2019). Killing the Password and Preserving Privacy With Device-Centric and Attribute-Based Authentication. arXiv (Cornell University), 15, 2183–2193. https://doi.org/10.1109/tifs.2019.2958763
50. Patel, H. (2026). Post-Quantum Security Enhancements for WebAuthn and FIDO2 Protocols. Journal of Information Systems Engineering & Management, 11, 853–862. https://doi.org/10.52783/jisem.v11i1s.14185
51. Pawan, V. R. (2023). Beyond Traditional Keyloggers: Developing and Detecting Advanced Keystroke Monitoring Systems. 1–6. https://doi.org/10.1109/csitss60515.2023.10334216
52. Samanta, S., Ray, B., & Milenković, A. (2025). Analysis of Temperature Effect on SRAM PUF for Low Cost Applications. 1–7. https://doi.org/10.1109/paine66113.2025.11320191
53. Sen, Ö., Hassan, T. M., Ulbig, A., & Henze, M. (2024). Enhancing SCADA Security: Developing a Host-Based Intrusion Detection System to Safeguard Against Cyberattacks. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2402.14599
54. Shadman, R., Wahab, A. A., Manno, M., Lukaszewski, M., Hou, D., & Hussain, F. (2025).A Keystroke Dynamics: Concepts, Techniques, and Applications. ACM Computing Surveys, 57(11), 1–35. https://doi.org/10.1145/3733103
55. Sharma, P. (2023). Destroke. International Journal for Research in Applied Science and Engineering Technology, 11(11), 1633–1635. https://doi.org/10.22214/ijraset.2023.56880
56. Shukla, S., Varshney, G., Singh, S., & Goel, S. (2024). A Passwordless MFA Utlizing Biometrics, Proximity and Contactless Communication. In arXiv (Cornell University). Cornell University. https://doi.org/10.48550/arxiv.2406.09000
57. Siahaan, C. R. P., & Chowanda, A. (2022). Spoofing keystroke dynamics authentication through synthetic typing pattern extracted from screen-recorded video. Journal Of Big Data, 9(1). https://doi.org/10.1186/s40537-022-00662-8.
58. Silver, D., Jana, S., Boneh, D., Chen, E., & Jackson, C. (2014). Password managers: attacks and defenses. 449–464. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.432.8172
59. Singh, N. T., Shukla, A., Nagar, A., Arya, K., Tiwari, A., & Varun, Y. (2023). Keylogger Development: Technical Aspects, Ethical Considerations, and Mitigation Strategies. 1–5. https://doi.org/10.1109/icemce57940.2023.10434134
60. Sultan, I., & Banday, M. T. (2019). Addressing Security Issues of the Internet of Things Using Physically Unclonable Functions. In Advances in information security, privacy, and ethics book series (pp. 95–116). IGI Global. https://doi.org/10.4018/978-1-5225-5742-5.ch004
61. Swierczynski, P., Fyrbiak, M., Koppe, P., Moradi, A., & Paar, C. (2016). Interdiction in practice—Hardware Trojan against a high-security USB flash drive. arXiv (Cornell University), 7(3), 199–211. https://doi.org/10.1007/s13389-016-0132-7
62. Tian, D., Scaife, N., Bates, A., Butler, K., & Traynor, P. (2016). Making {USB} Great Again with {USBFILTER}. USENIX Security Symposium, 415–430. https://atc.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_tian.pdf
63. Tritilanunt, S., Thanyamanorot, N., & Ritdecha, N. (2014). A secure authentication protocol using HOTP on USB storage devices. 17, 1908–1912. https://doi.org/10.1109/infoseee.2014.6946255
64. Varshney, G., Misra, M., & Atrey, P. K. (2017). Detecting Spying and Fraud Browser Extensions. 45–52. https://doi.org/10.1145/3137616.3137619
65. Vella, M., Colombo, C., Abela, R., & Špaček, P. (2021). RV-TEE: secure cryptographic protocol execution based on runtime verification. Journal of Computer Virology and Hacking Techniques, 17(3), 229–248. https://doi.org/10.1007/s11416-021-00391-1
66. Wang, X., Han, Z., & Zhang, D. (2012). IDKeeper: A Web Password Manager with Roaming Capability Based on USB Key. 1228–1231. https://doi.org/10.1109/icicee.2012.326
67. Williams, S., Khalil, K., & Bayoumi, M. (2024). A Novel 0.04 pJ/bit Dynamic TRNG using Bit Reconfigurable Ring Oscillators. 1546–1552. https://doi.org/10.1109/southeastcon52093.2024.10500289
68. Xiao, K., Rahman, M. T., Forte, D., Huang, Y., Su, M., & Tehranipoor, M. (2014). Bit selection algorithm suitable for high-volume production of SRAM-PUF. 101–106. https://doi.org/10.1109/hst.2014.6855578
69. Xu, M., Ng, W. C., Lim, W. Y. B., Kang, J., Xiong, Z., Niyato, D., Yang, Q., Shen, X., & Miao, C. (2022). A Full Dive Into Realizing the Edge-Enabled Metaverse: Visions, Enabling Technologies, and Challenges. IEEE Communications Surveys & Tutorials, 25(1), 656–700. https://doi.org/10.1109/comst.2022.3221119
70. Yadav, D. (2025). Duck Hunter: Encountering USB Rubber Ducky Threat. International Journal for Research in Applied Science and

Engineering Technology, 13(6), 2580–2587. https://doi.org/10.22214/ijraset.2025.72679

71. Yadav, R. K., & Patel, S. C. (2026). Tamper-Resistant USB Storage Device using BLAKE3 Hashing. In Research Square. https://doi.org/10.21203/rs.3.rs-8415630/v1.
72. Yamauchi, T., Akao, Y., Yoshitani, R., Nakamura, Y., & Hashimoto, M. (2020). Additional kernel observer: privilege escalation attack prevention mechanism focusing on system call privilege changes. International Journal of Information Security, 20(4), 461–473. https://doi.org/10.1007/s10207-020-00514-7
73. Zhou, Z. (2014). On-demand Isolated I/O for Security-sensitive Applications on Commodity Platforms. Research Showcase @ Carnegie Mellon University (Carnegie Mellon University). https://doi.org/10.1184/r1/6720932
74. Zuyeva, Y. A., Pyrkova, A., Saparbayev, A., Makulova, A., & Ordabayeva, G. (2022).
75. Devising an approach to analyze the parameters for determining potential premodified firmware of USB devices. Eastern-European Journal of Enterprise Technologies, 6, 51–58. https://doi.org/10.15587/1729-4061.2022.269031.