Stash House

Franklin Diaz
orcid.org/0000-0003-4586-8500

March 19, 2026
DOI: 10.5281/zenodo.19122412

University of Colorado **Denver**

Abstract

As developer environments increase in complexity, local credential sprawl has become a primary vector for authority loss. Project Stash House introduces a multi-layer framework for reconstituting disciplined secret use under conditions of local disorder. By moving beyond passive detection, we propose a model of "Authority Recovery" that bridges local workstation hygiene with decentralized transport layers (Nostr) and federated identity brokers (FOKS). This paper explores the "Materialization Boundary"—the critical transition point where encrypted, stored authority becomes live runtime authority—and validates a hybrid architecture that integrates enterprise standards like Kerberos and LDAP with sovereign, distributed backends.

Contents

0.1 Introduction

Developer workstations routinely accumulate sensitive credentials across shell history files, unencrypted configuration stubs, and source trees. While current tooling emphasizes pre-commit prevention (e.g., secret linting in CI/CD), practical engineering lacks standard patterns for reclaiming authority once a workstation is already disordered.

Project `stash-house` provides a reproducible framework for local secret hygiene, encrypted stewardship, and controlled runtime execution. Rather than acting as a replacement for enterprise vault services, it establishes a discipline for discovering workstation sprawl, migrating credentials into local encrypted storage, and materializing secrets strictly on demand.

0.2 System Architecture

The system defines two functional tiers separated by a critical execution boundary:

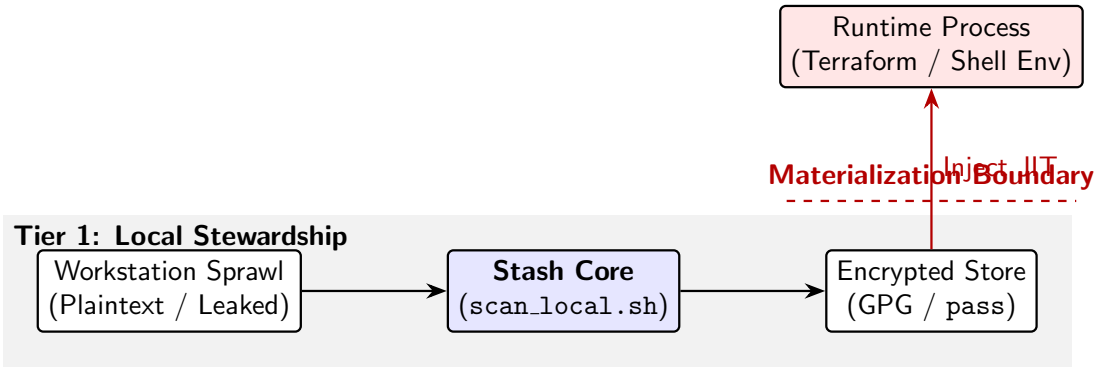

Figure 1: Stash House Core Architecture: Local Stewardship and Materialization Boundary. Federated transports are documented separately in Section 0.5.

0.2.1 Local Stewardship (Tier 1)

The local tier discovers exposed credentials and relocates them into standard encrypted stores managed via GPG and the UNIX `pass` utility. Secrets are tracked within a version-controlled repository utilizing `.gitattributes` filters to ensure ciphertext remains encrypted at rest.

0.2.2 The Materialization Boundary

The materialization boundary marks the exact interface where encrypted credentials become live runtime state (e.g., shell environment variables for Terraform execution). `stash-house` uses Just-In-Time (JIT) subshell injection so that secrets remain in process memory only for the duration of command execution, leaving zero plaintext residue in local disk artifacts or interactive shell history.

0.3 Implementation Mechanics

The workflow operates across three discrete phases:

1. **Discovery (`scan_local.sh`):** Scans standard developer paths (`~/.aws`, `~/.config/gcloud`, shell history) for unencrypted credentials, API keys, and service account tokens.
2. **Encapsulation:** Automatically ingests detected tokens into the GPG-backed `pass` hierarchy and purges the plaintext source files.
3. **Execution Wrappers:** Replaces static credentials with ephemeral evaluation wrappers (e.g., `pass show terraform/aws | eval`), ensuring secrets only exist in ephemeral child process environments.

0.4 Related Work

Secret Detection vs. Recovery. Static tools such as Gitleaks and GitGuardian inspect repositories during pre-commit or CI/CD stages. `stash-house` addresses post-sprawl workstation remediation, transitioning unencrypted secrets into active management [1].

Centralized Vaults vs. Local Stewardship. HashiCorp Vault and enterprise secret managers impose central server requirements and heavy coordination overhead. `stash-house` operates local-first, treating decentralized transports as opt-in synchronization channels rather than operational dependencies.

Materialization Control. Existing CLI tooling (e.g., 1Password CLI, `direnv`) injects environment variables but lacks integrated workstation remediation. Our model pairs automated scanning with scoped, ephemeral process execution to limit live-memory exposure windows.

0.5 Experimental Extensions: Distributed Transport and Federated Quorum

The following mechanisms extend the stable local hygiene toolchain beyond a single workstation. These are research prototypes and are not required for core `stash-house` operation.

> **Boundary:** The Tier 1 stewardship pipeline (Discovery → Encapsulation → Execution Wrappers) functions independently of any federated transport. The mechanisms below operate as optional synchronization and identity-brokerage channels for multi-environment mobility.

0.5.1 Nostr-NIP-78 Transport

Ciphertext can be synchronized across decentralized relays by encoding encrypted blobs as replaceable application events (`kind:30078`). NIP-78 storage decouples persistence from vendor-locked storage backends, enabling resilient replication without introducing a central coordinator [5]. NIP-44 encryption [4] provides the transport-level confidentiality layer for direct-message channels.

0.5.2 FOKS Key Brokerage

The Federated Open Key Service [3] can be integrated to govern decryption via hardware security keys, enabling multi-site quorum requirements for credential access.

0.5.3 Enterprise Identity Bridge

Site-to-site federation with existing directory services (LDAP/Kerberos) allows the local repository to act as a trusted source of truth that interoperates with organizational identity infrastructure [2], [6]. These bridges are opt-in and do not modify the core encryption or stewardship mechanics.

0.6 Conclusion

Project `stash-house` formalizes workstation credential hygiene by pairing automated local exposure scanning with GPG-backed secret stewardship. By isolating the runtime materialization boundary and treating distributed identity mechanisms as modular extensions, the framework restores security discipline to developer workstations without requiring disruptive changes to engineering workflows.

Revision History

Revision	Date	Author(s)	Description
0.1.0	2025-11-15	FD	Initial draft
0.3.6	March 19, 2026	FD	Simplified architecture: removed speculative LLM node, stripped external component diagram, merged NIP-78 into federated extensions, added implementation mechanics section, compressed related work.
0.3.7	March 19, 2026	FD	Drew explicit core vs. experimental boundary; carved Section 0.5 for distributed transports and federated quorum prototypes; removed federated infrastructure references from System Architecture section.

Bibliography

[1] Franklin Diaz. Stash-house: Local credential hygiene and encrypted stewardship, 2026.

[2] Jason Garman. *Kerberos: The Definitive Guide*. O'Reilly Media, 2003.

[3] Maxwell Krohn. The federated open key service (FOKS), 2026.

[4] Nostr Protocol Authors. NIP-44: Encrypted direct messages (version 2), 2023.

[5] Nostr Protocol Authors. NIP-78: Application-specific data, 2023.

[6] The OpenLDAP Project. OpenLDAP software administrator's guide, 2025.