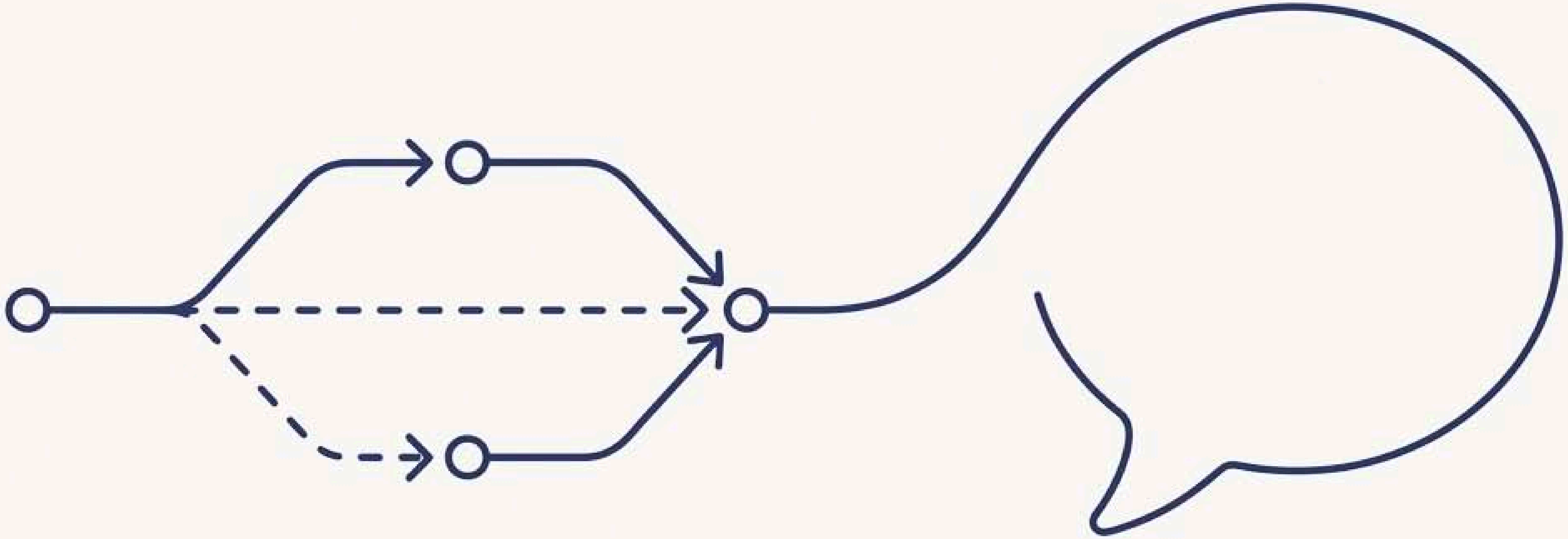


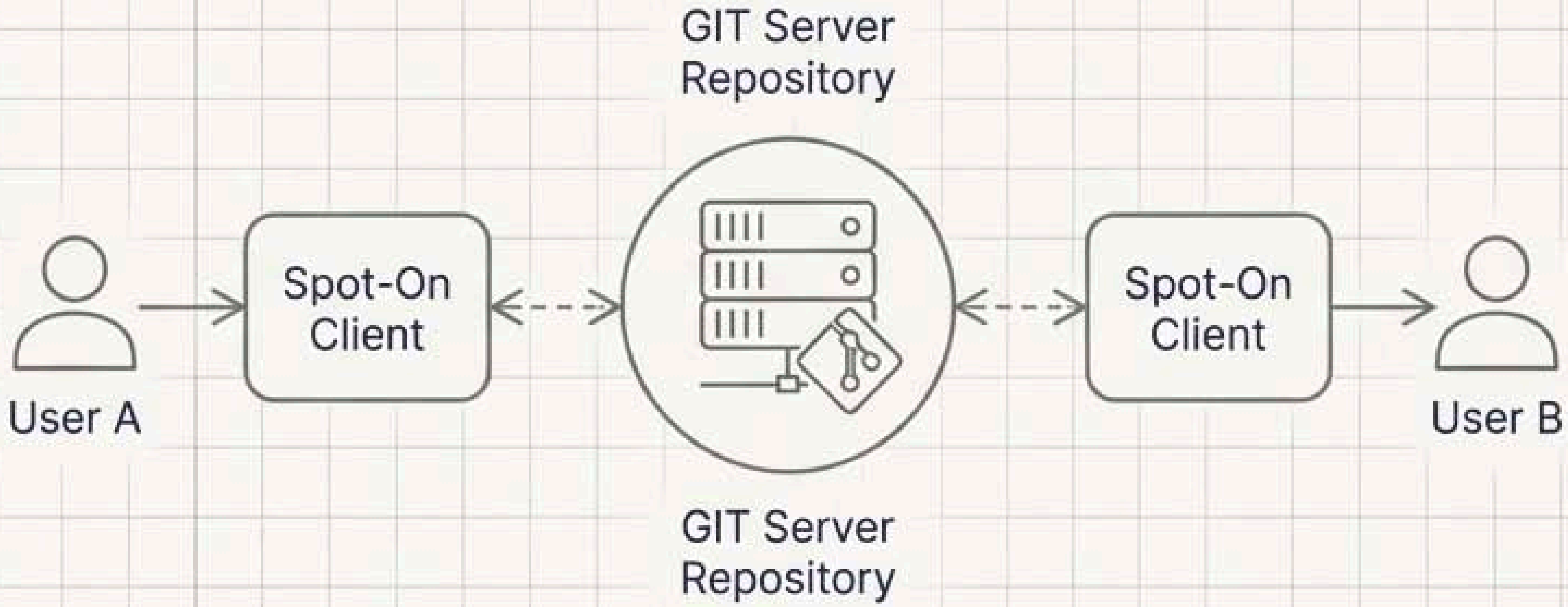
Prison Blues: Encrypted Communications Over GIT

A Deep Dive into Spot-On’s Covert Messaging Channel

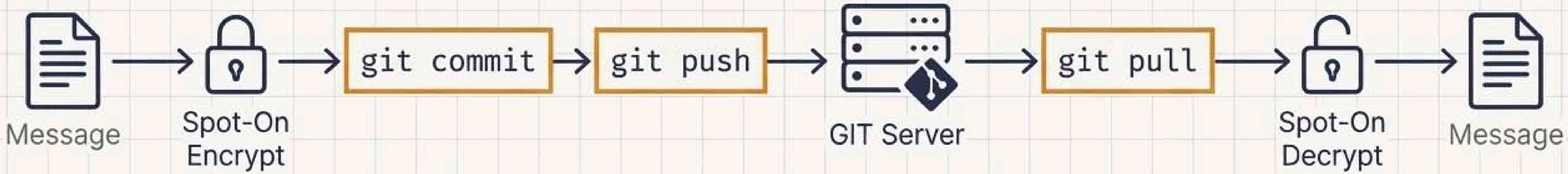
Spot-On Encryption Suite

A Messaging System Disguised as Code Collaboration

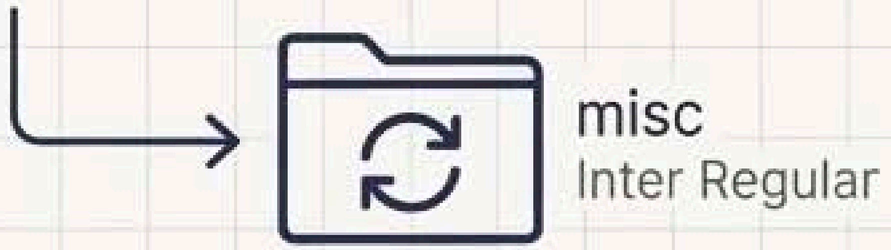
- Prison Blues leverages standard GIT servers as a decentralized, asynchronous message bus.
- Enables encrypted chat and data exchange in environments where only GIT protocols might be permitted.
- Built upon a simple, robust foundation: any system capable of executing Bourne Shell scripts.
- Fundamentally transport-agnostic, using GIT as one of several communication communication methods available within the Spot-On suite.



The GIT Repository as a Message Store



- Spot-On clients `commit` and `push` encrypted messages as file objects to a shared repository.
- Peers `pull` or `fetch` updates from the repository to receive new messages.
- The repository functions as a persistent, auditable, and distributed message queue.
- A "miscellaneous" directory within the repository can function as a file-system-based "Echo," where a Spot-On process can review, process, and purge files.



Two Distinct Modes for Different Missions

Chat

Purpose: Real-time style messaging.

Supported: Standard messages (including bundled messages) and the Socialist Millionaire Protocol (SMP).

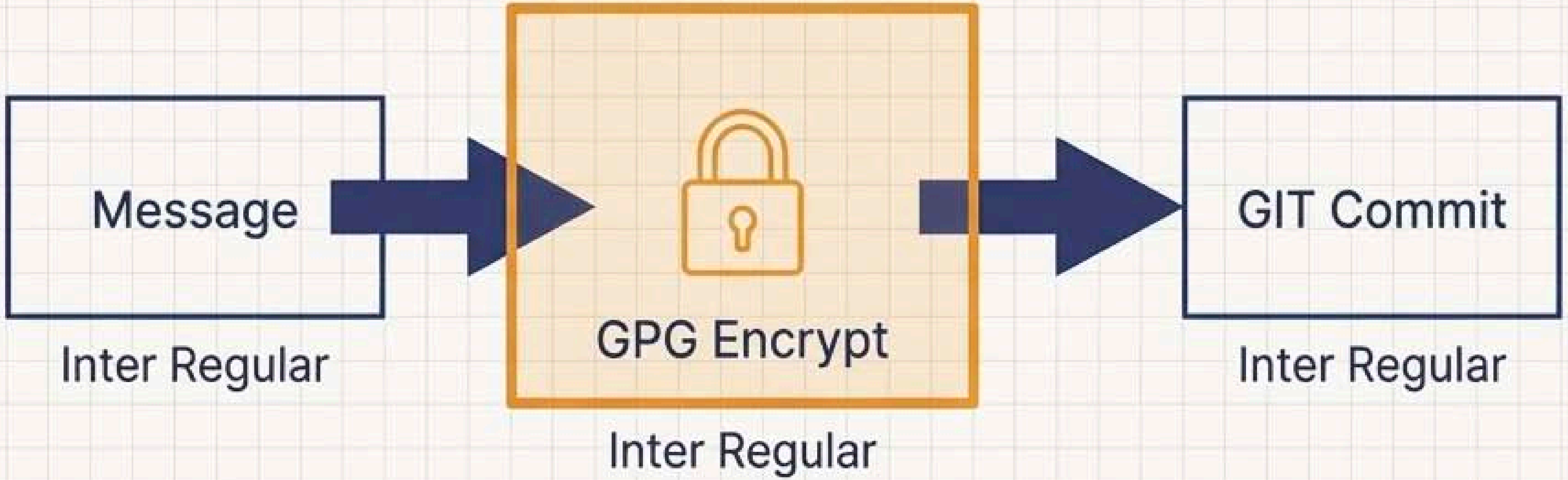
Ignored: Poptastic accounts and Calling messages.

Rosetta

Purpose: Secure data and key exchange.
Supported: GPG messaging, GPG attachments, GPG key bundle sharing, and Status messages (which require valid signatures).

Ignored: Calling messages.

End-to-End Encryption Secured by GPG



- Prison Blues acts as the transport layer; GPG provides the end-to-end message security within the Rosetta module.
- **Key Sharing:** Securely exchange GPG key bundles with peers directly through the GIT repository.
- **Encrypted Messaging:** Send and receive fully GPG-encrypted and signed messages.
- **Secure Attachments:** Transmit encrypted files as GPG attachments.

Laying the Foundation for Secure Communications

Client-Side: A Unix-like environment with a Bourne Shell and the Spot-On client installed.

Server-Side: Any standard GIT server (e.g., GitHub, GitLab, self-hosted).

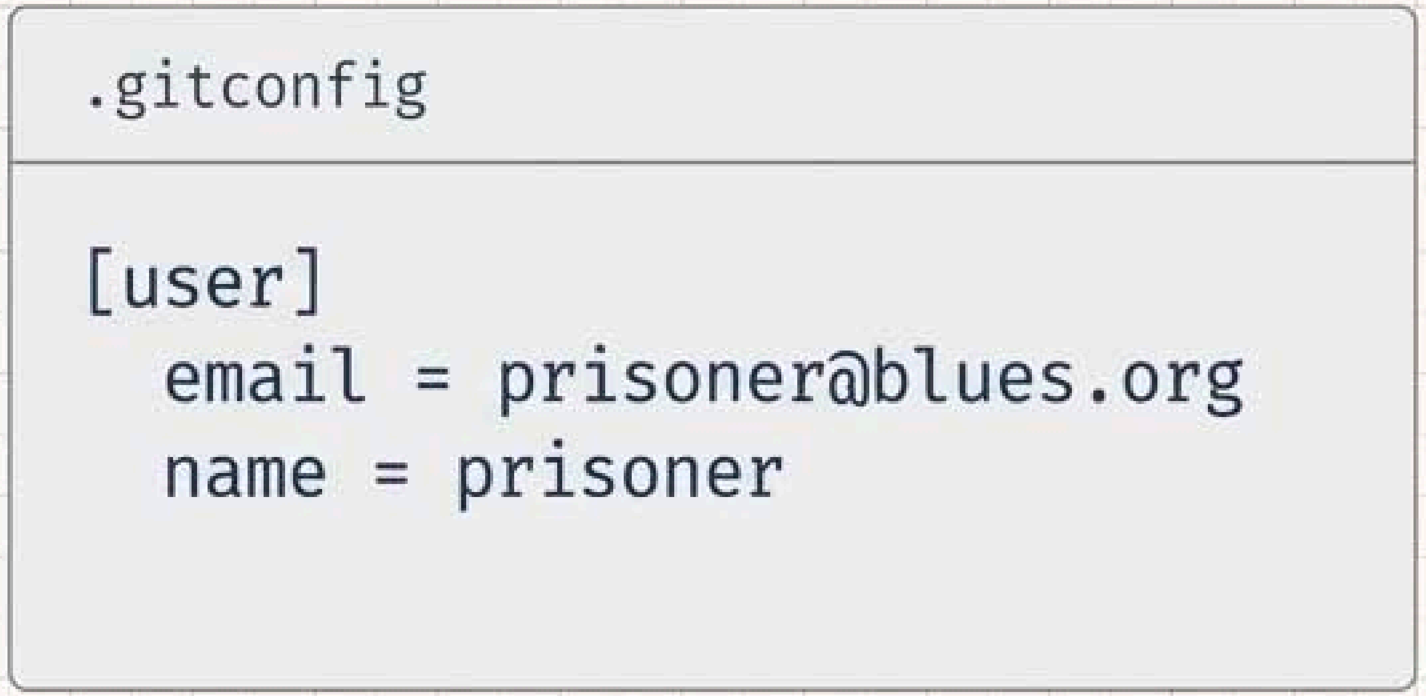
Authentication: User credentials (account/token) must be established outside of Prison Blues. This is a mandatory setup step.

Permissions: GIT accounts require read/write access to the designated repository. Access must be tightly controlled.

Configuring Your Spot-On Client

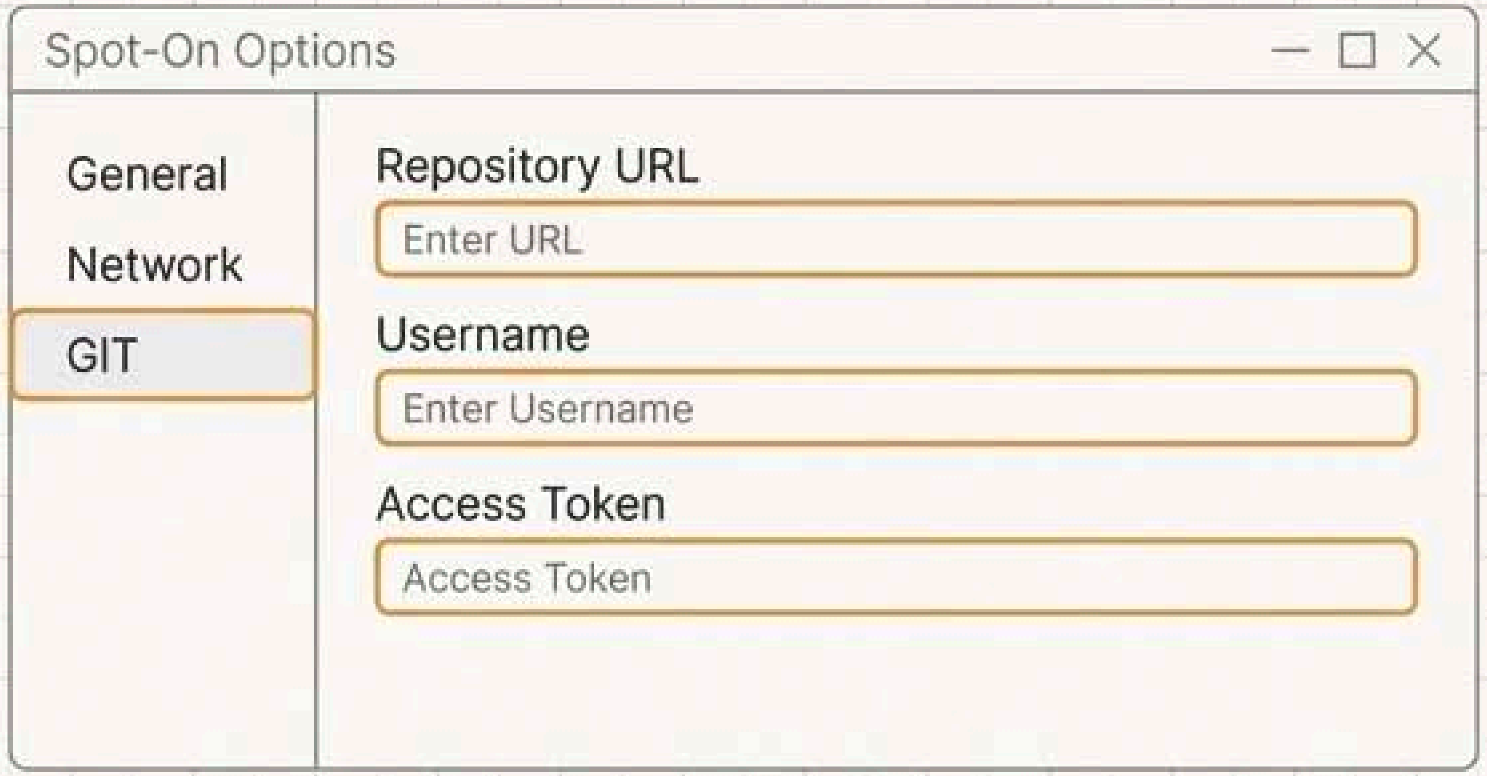
Step 1: Local GIT Identity: Configure your local `.gitconfig` file. The user email and name are used for commits.

Step 2: Spot-On Settings: Enter the repository URL, credentials, and other options in the 'GIT' section of the **Spot-On Options** window.



.gitconfig

```
[user]
  email = prisoner@blues.org
  name = prisoner
```



Repository URL

Enter URL

Username

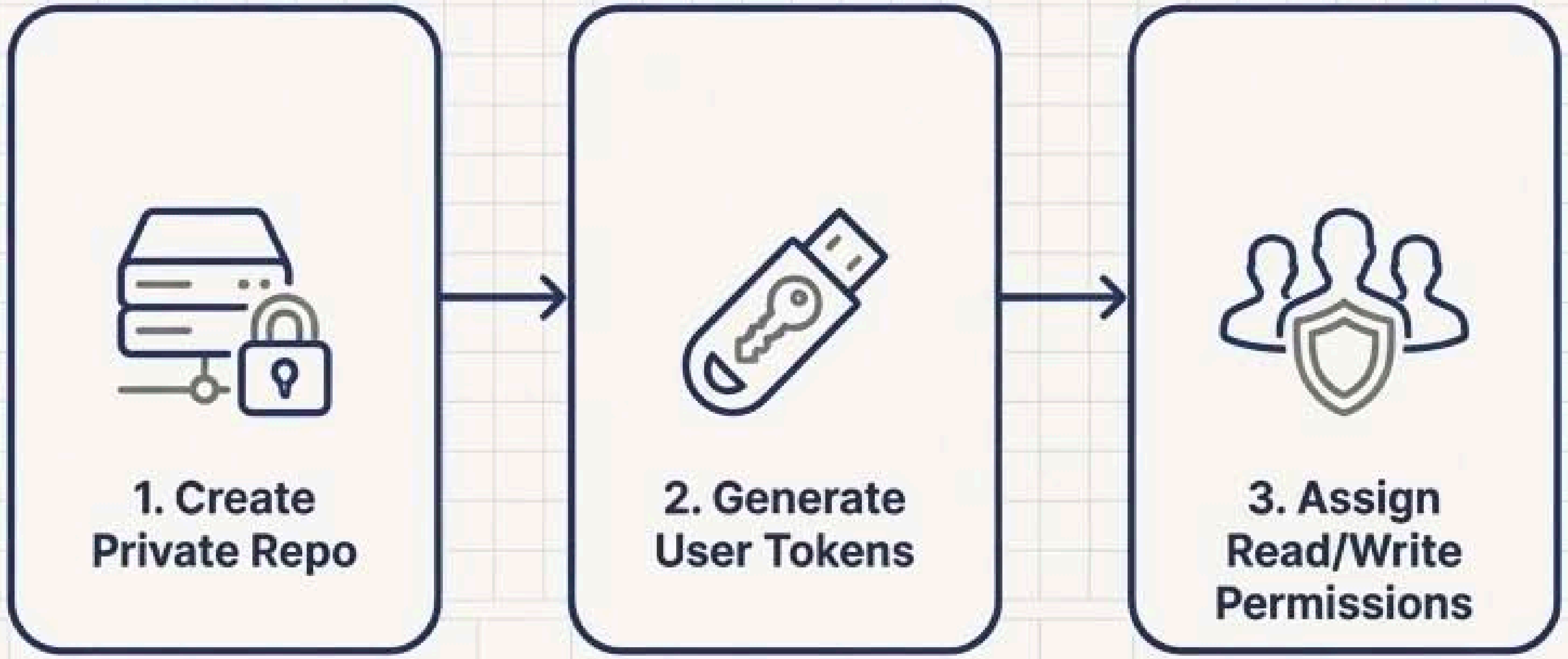
Enter Username

Access Token

Access Token

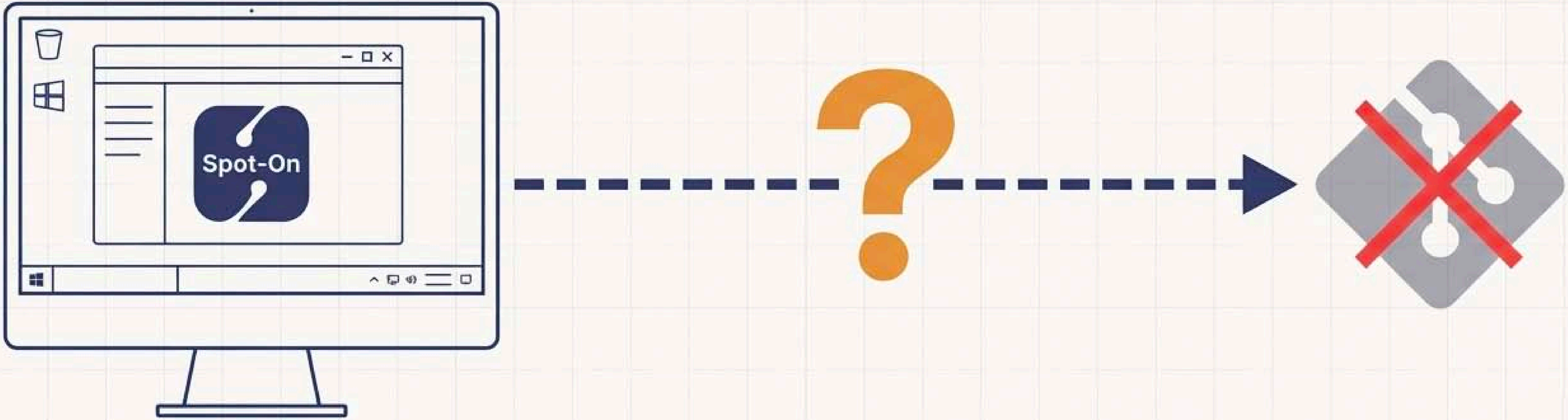
Securing the Repository

- **Create a Dedicated Repository:** Use a private repository specifically for Prison Blues communications to isolate the channel.
- **Generate Access Tokens:** For each client, create fine-grained personal access tokens or SSH keys. Avoid using main account passwords.
- ****Limit Access:** Configure repository permissions to ensure only authorized participants can read and write. This is the primary access control mechanism.



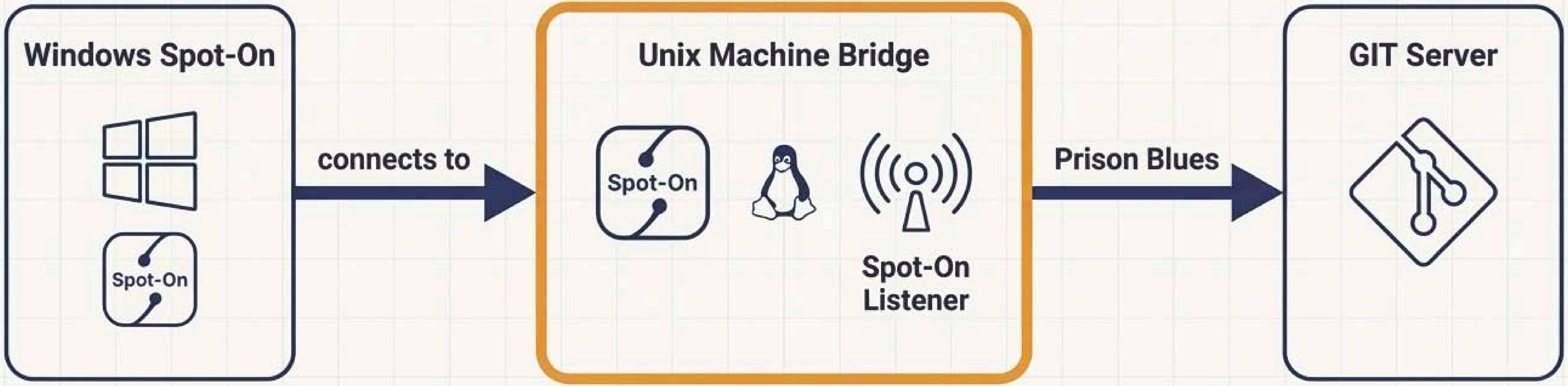
The Problem: Native GIT Isn't Readily Available on Windows

- The core Prison Blues feature relies on a Bourne Shell environment for its GIT operations.
- This presents a challenge for native Windows installations of Spot-On.



The Solution: A Unix-like Machine as a Dedicated Bridge

- **Step 1: The Bridge:** Install and configure Spot-On with Prison Blues on a Unix-like machine (e.g., Raspberry Pi, Linux VM).
- **Step 2: The Listener:** On that Unix machine, prepare a local Spot-On listener interface.
- **Step 3: The Connection:** Connect the Windows Spot-On instance to the listener on the Unix machine.

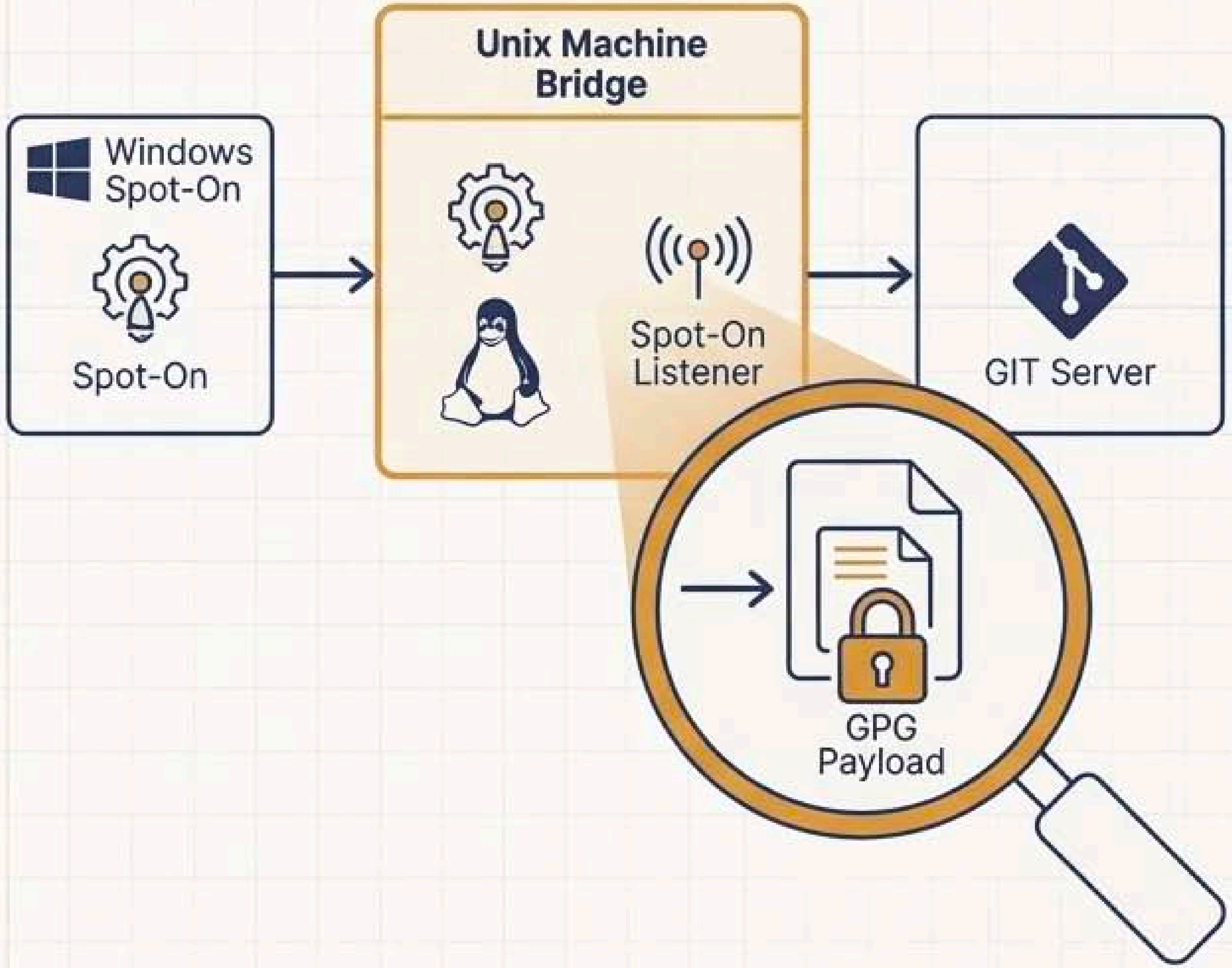


Operational Considerations & Limitations

- **No Calling Messages:** The protocol is not suited for features requiring strange expiration times.
- **No Human Proxies:** This advanced routing feature is not functional over the GIT transport.
- **Asynchronous Nature:** This is not a low-latency, real-time protocol. It is inherently asynchronous, with delivery dependent on `fetch`/`pull` frequency.
- **Metadata Exposure:** While message *content* is encrypted by GPG, GIT commit logs contain metadata (author email/name, **timestamps)**. This activity is visible in the repository history.

Prison Blues: A Resilient and Multi-Layered Covert Channel

- **Transport Layer**: Leverages the ubiquity and resilience of GIT servers.
- **Application Layer:** Offers distinct channels for chat (Chat) and secure data exchange (Rosetta).
- **Security Layer:** Integrates seamlessly with GPG for robust end-to-end encryption of the payload.
- **Platform Adaptability:** Provides a clear architectural pattern for use even on non-native platforms like Windows via a bridge.



Explore the Source

Project Repository:

https://github.com/textbrowser/spot-on

Further Reading:

Spot-On Encryption Suite
- Democratization of Multiple & Exponential Encryption

ISBN: 978-3-7494-3506-7

Spot-On