WIKEY

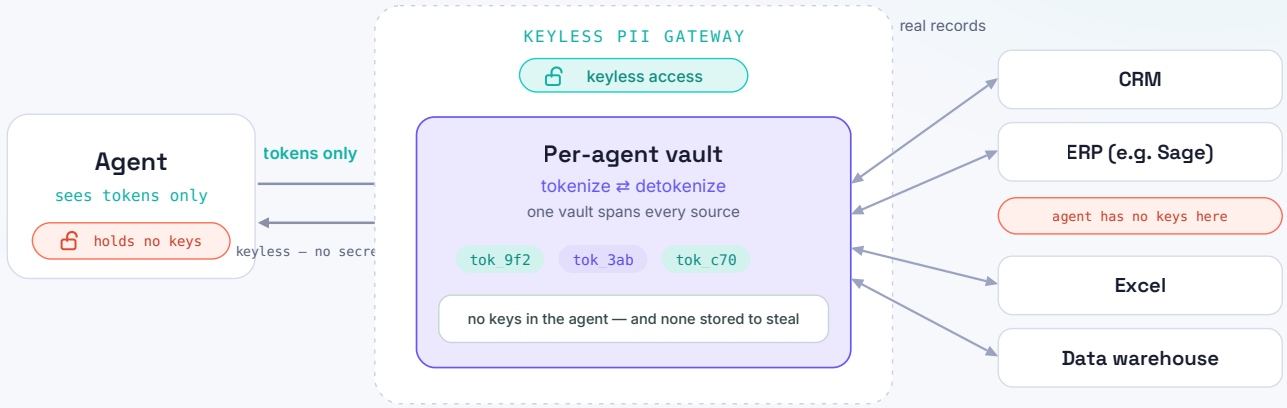
WiKey Data Tokenizer

# Your agent never holds the keys or the data.

The agent holds no keys — not to the tokenizer, not to your systems — and only ever works on tokens.
There's no secret to phish and no real data to leak, even if the agent is hijacked.

THE ARCHITECTURE

## One keyless vault per agent. Real data never reaches the model.




PII detection powered by Microsoft Presidio

### No key to the tokenizer

The agent reaches the gateway keylessly — authentication is hardware-rooted. No secret sits in its context to phish or leak.

### No keys to your systems

The gateway brokers each CRM, ERP, and Excel connection keylessly. No credential sits in the agent — and none is stored to lift.

### Only tokens leave

One vault per agent spans every source. Real values are restored solely on an authorized write — never into the model.

**Personal data stays out of the components agents touch.** By design, the server, vault, and agent work on tokens rather than raw values — so user secrets and PII such as passwords, photo IDs, and phone numbers aren't kept where an agent could reach them.

**Open source, for total security.** Every line of WiKey is open to inspection — so its security is something you can verify for yourself, not just take on trust.

ROOT OF TRUST | **Hardware-bound keys** | Write-only vault, no credentials | Recovery via threshold MPC



No keys · No credentials · Nothing to steal

# Tokenization removes the data risk. The key is still the problem.

The moment an agent holds — or can reach — a secret, that secret becomes the weakest link. Keys get backed up on a server, copied into an agent's context, or reused from the user's own credentials, which means they can be lost, breached, phished, or used by an impersonator. Protecting the data closes half the gap. WiKey closes both halves: there is no agent-held or stored credential to take.

RISK 1 — THE DATA

### Real data reaches the model

The agent or model sees raw PII. Tokenization solves this: values are swapped for tokens, so the model never handles the real thing.

Closed by tokenization — including the competition.

RISK 2 — THE KEY

### The agent holds a usable secret

An API key, a tool or MCP credential, or the user's own keys. It can be phished, impersonated, breached, or backed up — and the key walks out.

Left open by tokenization. Closed only by going keyless.

| Approach | Real data never reaches the model | No key stored server-side | No secret held by the agent | Nothing to phish or impersonate |
|---|---|---|---|---|
| **Data privacy vaults**<br>Skyflow · TokenEx · Basis Theory | ✓ | × | × | × |
| **Vaultless tokenization**<br>VGS · TrueZero · CipherTrust · Rixon | ✓ | ~ | × | × |
| **Secrets managers / KMS**<br>HashiCorp Vault · AWS KMS | — | × | × | × |
| **AI gateways & PII redaction**<br>Protecto · BoxyHQ · cloud guardrails | ✓ | ~ | × | × |
| **WiKey**<br>keyless · hardware-bound · MPC | ✓ | ✓ | ✓ | ✓ |

> **A key always has to live somewhere.** Even a "vaultless" tokenizer needs an API credential to call it; even an MCP server or tool needs a key to configure; even a secrets manager hands the agent a secret at runtime. Any secret that exists can be backed up, breached, phished, or impersonated. WiKey's access is hardware-bound and brokered per request — keyless by construction — so there is nothing to store and nothing to steal.

Comparison by architecture category; individual products vary. ✓ addresses · ~ partial / configuration-dependent · × does not address · — out of scope. The distinguishing question: does a usable, extractable credential exist in the agent or a server-side store?

## Deploy agents on data you can't afford to leak.

Put WiKey in the path and a hijacked agent has nothing to take.

info@wikey.io