The Tower

The guardian, the helpers, and the key that changes shape

February 2026

The tower

Much of this content library talks about the vine: how connections grow, how contracts govern relationships, how events flow between vaults. All of that happens outside the tower. This document is about what happens inside.

Your vault is a tower that only you have the key to. It is built from hardware isolation — a physical boundary in the processor that separates your vault's memory from everything else running on the same machine. Not a software boundary that another program could cross. Not a permission setting that an administrator could override. A wall enforced by the silicon itself. The operating system cannot see inside. The cloud provider cannot see inside. The people who built the tower and operate the infrastructure it sits on cannot see inside. Nobody enters without your key.

Inside the tower lives a guardian — the vault manager — and his helpers — the handlers. The guardian receives every request through a narrow slot in the wall. He checks the request against your contracts and policies. If the request passes, he calls on the right helper to do the work: signing a document, authorizing a payment, verifying your identity, encrypting a message. The helper does its job inside the tower, with access to your keys and secrets, and hands the result back to the guardian. The guardian passes the answer out through the slot. The door never opens. Nobody enters. Nobody sees inside. The answer comes out and that is all the outside world ever gets.

> *Your vault is a tower. The hardware is the wall. The guardian checks every request. The helpers do the work. The door never opens. The outside world gets answers, never access.*

The walls: hardware isolation

The tower's walls are not software. Software walls have doors — administrator accounts, debug interfaces, emergency overrides, kernel access. Someone always has the key to a software wall because someone had to build the wall and someone has to maintain it. Hardware isolation is different. The wall is in the chip. No account, no privilege level, no override can cross it.

In VettID's implementation, the tower is an AWS Nitro Enclave: a hardware-isolated virtual machine with three properties that matter.

> **No persistent storage:** The enclave has no disk. Everything in its memory exists only while it is running. When the enclave stops, the memory is gone. There is no leftover data to scrape, no log file to read, no cache to examine. The vault's database is encrypted and stored externally — it can only be decrypted inside the enclave, and only with your PIN.
>
> **No network access:** The enclave has no internet connection, no IP address, no open ports. It communicates with the outside world through a single constrained channel, and everything passing through that channel is encrypted. There is no way to reach into the enclave from the network. There is no way for the enclave to reach out except through the one channel the architecture defines.
>
> **No interactive access:** Nobody can log into the enclave. Not the cloud provider. Not the system administrator. Not the VettID team. There is no SSH. There is no console. There is no debug port. The enclave runs its code and communicates through the constrained channel. That is the entire interface.

These are not policies. They are not promises. They are properties of the hardware. A Nitro Enclave does not have interactive access the way a circle does not have corners. It is not that someone decided to leave them out. The shape does not support them.

How you know the tower is genuine

Hardware isolation means nobody can see inside the tower. But how do you know the right code is running inside? How do you know someone did not replace the guardian with an imposter? This is what attestation solves.

The enclave's hardware — a Nitro Security Module, a physical chip, not software — generates an attestation document that contains cryptographic hashes of the exact code running inside. These hashes are called PCR values. Your app sends a random challenge to the enclave, and the enclave includes that challenge in its attestation response, signed by the hardware. Your app verifies the hardware signature, checks the PCR values against VettID's published values, and confirms the challenge matches. If any check fails, the app refuses to proceed.

This means you can verify, from your phone, that the code running inside the tower is the code VettID published and nothing else. If VettID deploys different code — even a one-byte change — the PCR values change, and every app in the world will reject the connection. VettID cannot silently modify the guardian. Nobody can. The hardware produces the proof. The math verifies it.

And when the code does change legitimately — bug fixes, security patches, new handlers — the new code has different PCR values. Your app is notified. The old enclave and the new enclave run simultaneously. You are not moved to the new version until you approve. You control when the code that handles your secrets changes. Not VettID. You.

> *The walls are hardware. The attestation is hardware. The proof is math. You verify from your phone that the code inside the tower is genuine. If anyone modifies it, every app in the*

> *world rejects the connection. You approve code changes before they take effect. The tower is yours.*

The guardian and his helpers

Inside the tower, the vault manager — the guardian — is the single authority. Every request arrives through the messaging layer, and the guardian processes every one through the same pipeline: decrypt, validate against the contract, determine the right handler, invoke it, encrypt the response, send it back. The guardian never shortcuts the pipeline. The guardian never skips a check. The guardian is a program, not a person, and programs do not get tired or make exceptions.

The helpers — the handlers — are separate executables that live in the tower alongside the guardian. Each one is a specialist: one signs documents, one authorizes payments, one verifies identity, one handles authentication challenges, one manages messaging. The guardian calls on the right specialist for each request. The specialist does its work — reading your keys, signing with your identity, encrypting with your credentials — and hands the result back to the guardian.

The helpers are separate programs, not part of the guardian's own code. Any vault manager can call them. This makes the architecture modular — the dispatcher and the workers are different concerns. But they share the same tower and the same attestation measurement. They are part of one attested image. Change a helper and the tower's measurement changes, which means the entire enclave must be re-attested. The independence is in the code architecture: separate programs, clear responsibilities. The trust boundary is shared: same walls, same proof, same PCR values.

This is also where vault providers differentiate. VettID builds the helpers for authentication, transaction signing, secure messaging, and location tracking — the critical operations that touch every user's secrets. Other providers could build helpers for healthcare consent, financial multi-party authorization, or enterprise policy enforcement. The guardian and the tower are the platform. The helpers are the product. Different providers, different helpers, same walls.

> *The guardian checks every request. The helpers do the specialized work. They are separate programs sharing the same tower, the same attestation, the same trust boundary. The guardian is the platform. The helpers are the product. Different providers build different helpers. The walls protect all of them.*

The key that changes shape

Every security system you have ever used gives you a static key. A password. A private key. A token. You create it once and use it until you change it, which for most people is never. If someone steals it, they have it forever — or at least until you notice and rotate it, which might be weeks or months or never.

Your vault uses a Protean Credential. It is called protean because it changes shape after every single use.

What the credential is

Your Protean Credential is an encrypted package that lives on your phone. Inside it are your identity keys, your private keys, your password hash, and your policies. Everything that defines your digital identity is in this package. But you cannot open it. The package is encrypted with a

Credential Encryption Key — a CEK — that only the guardian inside the tower holds. Your phone has the locked box. The tower has the key.

How you authenticate

Two factors protect the tower, each working differently.

> **Your PIN:** Entered when you open the app. The PIN is sent to the enclave, where it is combined with sealed material from hardware key management to derive the Data Encryption Key that decrypts the vault's database. The PIN is not compared against a stored value. It is verified by whether the math works. Wrong PIN means wrong key, which means the decryption fails cryptographically. There is no hash to steal.
>
> **Your password:** Entered for each sensitive operation — signing a transaction, using a key, authorizing a payment. The password is hashed with Argon2id on your device before it leaves your phone. The hash is then encrypted with a single-use User Transaction Key and sent to the vault. The vault decrypts it with the corresponding Ledger Transaction Key, compares the hash against the one stored inside your credential, and proceeds only if they match. The transaction key pair is single-use: destroyed after one operation. Even if an attacker intercepts the encrypted hash, they cannot replay it.

The protean part

Here is where it gets interesting. After every operation — every time you sign something, authorize something, use a key for anything — the guardian generates a brand new CEK keypair. It re-encrypts your entire credential with the new key. It stores the new private key in the vault's encrypted database. And it sends the re-encrypted credential back to your phone along with the operation result and fresh transaction keys for next time.

Your phone stores the new encrypted blob. The old CEK is dead. The old blob is cryptographic garbage. If anyone intercepted it, copied it, stole it from your phone — they have a package locked with a key that was destroyed the moment the operation completed.

This happens after every single use. Not monthly. Not daily. Every operation. The key generation takes approximately 0.05 milliseconds. There is no performance penalty for changing every lock in the building after every use, because the locks are nearly free to manufacture.

Because the credential changes after every use, there is only one valid copy at any given moment. If an attacker steals your encrypted blob, they have a narrow window: they must present it to the vault and authenticate before you do. The moment either party uses the credential, the CEK rotates and the other copy becomes worthless. It is a race the attacker almost certainly loses, and even if they win once, they get one operation, not permanent access.

> *The credential changes shape after every use. The old key is destroyed. The old blob is garbage. There is no stable target to steal. An attacker who copies your credential has a package locked with a key that no longer exists. That is the protean principle: your identity is a moving target.*

The lockbox under your bed

Phones break. Phones get lost. Phones fall into swimming pools. If your entire digital identity lives in a tower that you access through your phone, losing the phone has to be survivable.

The backup model is deliberately simple. Every time your vault returns the re-encrypted credential to your phone after an operation, you can store a copy at that same moment. Not a separate ceremony. Not a process you have to remember to perform. Not twenty-four words you write on a piece of paper and hide in a drawer. The credential comes back. You save a copy. Done.

What the backup contains

The backup is the same encrypted blob that lives on your phone. It is your Protean Credential — the locked box. It is useless without a genuine enclave and your PIN. An attacker who obtains the backup has an encrypted package they cannot open, locked with a CEK that only exists inside a hardware-isolated enclave they cannot access. The backup is safe to store because the backup cannot be used without the tower.

Cloud and local

You choose where the copy goes based on your comfort level. Store it in VettID's hosted backup. Store it locally on a second device. Store it in both places. The encrypted blob is safe wherever it sits because the encryption is the protection, not the storage location. The backup's security does not depend on where you put it. It depends on the fact that only a genuine enclave with your PIN can open it.

Recovery

Phone lost? Phone broken? You log into the account portal from a new device. You request recovery. There is a deliberate waiting period — an intentional delay that gives you time to cancel the request if it was not you who made it. After the waiting period, you scan a QR code on your new device, enter your PIN, and the backup credential is delivered to the enclave. The vault warms, the credential decrypts, and you are back. No seed phrases. No recovery keys written on paper. No memorizing twelve words. The credential you stored is your way back, and the process is the same process you used to set up in the first place.

This is what self-sovereignty actually means in practice. You are not calling a support line and hoping someone can reset your account. You are not depending on a company to keep a recovery key on your behalf. You stored a copy of your credential. The copy is encrypted. The encryption can only be undone inside the tower with your PIN. You hold everything you need to recover, and nobody else can use what you hold.

> *The backup is simple: when the credential comes back, store a copy. No seed phrases. No ceremony. The copy is the same encrypted blob — useless without the tower and your PIN. Phone lost? Recover with the copy, your PIN, and a genuine enclave. You hold everything you need. Nobody else can use what you hold.*

How they work together

The tower, the guardian, the helpers, the protean key, and the backup are not five separate features. They are one system, and the security comes from how they fit together.

The tower protects the guardian. Hardware isolation means nobody can observe the guardian at work — not the cloud provider, not the vault operator, not an attacker who compromises the host machine. The guardian's decisions happen behind walls that software cannot cross.

The guardian protects your secrets. Every request passes through the guardian's pipeline: validate, check the contract, invoke the right helper. No shortcut. No bypass. The guardian ensures your keys are only used for operations you authorized, under contracts you accepted.

The helpers do the work without opening the door. A payment gets signed. A document gets authenticated. An identity gets verified. All inside the tower. The result goes out. The keys never do. The helpers access your secrets, perform the operation, zero the keys from memory, and return the answer. Your secrets exist in enclave memory for milliseconds.

The protean key makes theft pointless. Even if an attacker steals the encrypted credential from your phone, it is locked with a key that changes after every use. The

> attacker has a snapshot of something that no longer exists. The credential they stole is already worthless.
>
> **The backup makes loss survivable.** Lose your phone and you still have a copy of the credential. The copy is encrypted with the same protections as the original. Recover it into a genuine enclave with your PIN and you are back. No dependency on anyone else. No support ticket. No seed phrases. Just the copy you stored and the PIN you know.

Remove any one of these and the system has a gap. Without the tower, the guardian can be observed. Without the guardian, requests are not validated. Without the helpers, the vault cannot act on your behalf. Without the protean key, a stolen credential is permanently useful. Without the backup, a lost phone is a catastrophe. Together, they form a closed loop: protected execution, enforced governance, specialized capabilities, moving-target credentials, and self-sovereign recovery.

What this means for you

You do not need to understand any of this to use it. You install the app, create a PIN and password, and add the things you want to protect. Behind the scenes, the tower is provisioned, the guardian starts listening, the helpers are ready, and the protean credential is issued. You see: scan code, create PIN, create password, done.

When you sign a transaction, you enter your password and the app shows you the result. You do not see the CEK rotation, the enclave attestation, the handler invocation, the single-use transaction keys. You do not need to. The complexity is real, and it works precisely because you do not have to manage it.

But the complexity is also verifiable. The PCR values are published. The algorithms are named. The attestation chain is checkable from your phone. If you want to verify that the tower is genuine, you can. If you want to trust

that it works and get on with your day, you can do that too. The security does not depend on your understanding of it. It depends on the hardware, the math, and the architecture.

> *A tower you hold the only key to. A guardian who checks every request. Helpers who do the work behind walls nobody can cross. A key that changes shape after every use. A backup you can store with a single action. Together, they make you your own security infrastructure. Not by asking you to be a security expert. By being one for you.*