TECHNICAL PAPER · DRAFT 0.1 · NOVEMBER 2026

VeriGate: two-step codes with nothing to hand over

DRAFT · NOT YET REVIEWED

How the app, the backup and the relay are built, and what each kind of attacker can and cannot get.

Contents

	In one page	2
1	What we protect	3
2	What we never collect	4
3	On the phone	5
4	App passcode and the decoy	6
	The decoy, and its limits	7
5	Two kits and a PIN	8
6	How the keys fit	9
7	The PIN check	10
8	What each attacker gets	11
9	The relay	12
10	Who runs what	13
11	What our server stores	14
12	Signed requests and old versions	15
13	When your backup is opened elsewhere	16
14	New kits, turning off, deleting	17
15	Moving to a new phone	18
16	Coming in and going out	19
17	Damaged files and wrong answers	20
18	If we shut down	21
19	Known limits	22
20	Open source and review	23

In one page

VeriGate makes two-step sign-in codes and keeps the secret keys behind them on your phone, encrypted.

There is no account. We never learn an email, a phone number or a name.

Backup is optional. With it off, the app never goes online.

With it on, your phone seals the backup before it leaves. Opening it always needs your Recovery Kit, plus your PIN or your Emergency Kit. We hold neither.

Backup traffic goes through a relay, so our server doesn't see your IP address.

The apps and their core are open source (AGPL-3.0). The code goes public when VeriGate reaches the stores.

No outside firm has reviewed this design yet. Where something rests on a promise and not on maths, we say so.

1. What we protect

The secret keys in your vault. Anyone with one of those keys can make your codes forever, so the whole design is about keeping them on your side.

Second: who you are. VeriGate holds no email, phone number or name. The backup's ID is made on your phone from your Recovery Kit and can't be traced back to it.

Third: that you can always get your codes back, even if we stop running the service.

In one line. Your phone seals everything before it leaves. Opening a backup always needs your Recovery Kit, plus either your PIN (with our server) or your Emergency Kit (without it).

2. What we never collect

No sign-up. No email, phone number, name, device model or advertising ID.

No analytics, no crash reporting, no ads, and no third-party code in the apps beyond the phone's own.

No contacts, location or photos. The camera is used only to scan QR codes, on the phone.

With backup off, the app makes no network requests at all, not even an update check: the store keeps it up to date.

With backup on, it talks only to our relay and our server, as sections 9 to 13 describe.

3. On the phone

Your vault is one file, compressed and then sealed with XChaCha20-Poly1305 under a random 256-bit data key.

iPhone: the data key is sealed by a Secure Enclave key that works only on this device. **Android:** it is wrapped by a Keystore key (StrongBox where the phone has it) that needs an unlock and stops working when a new fingerprint is added.

The vault leaves memory when the app locks. It locks 30 seconds after you leave it. Copied codes clear from the clipboard after 30 seconds. The vault is left out of phone and cloud backups.

Android blocks screenshots. iPhone can't; there, codes hide while the screen is recorded or mirrored. Private mode hides every code until you tap it.

4. App passcode and the decoy

An optional passcode sits in front of the vault. It lives in one 164-byte file with two slots. Each slot holds a data key, sealed with a key made from the passcode (Argon2id, 64 MiB) and a device secret kept in the phone's secure hardware. The file alone can't be guessed anywhere else.

A second passcode opens the other slot: a decoy with different codes, for the moment someone makes you unlock.

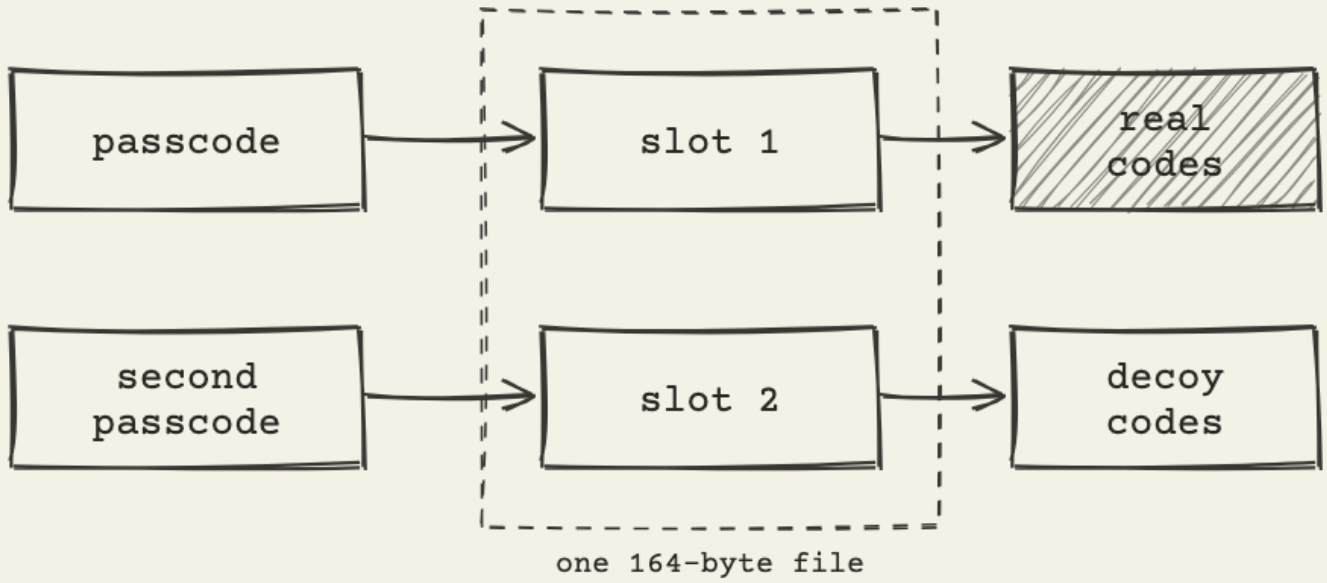


Fig. 1. An unused slot is random bytes and unlocking always tries both.

The decoy, and its limits

Nothing on the phone shows which slot is real, or whether a decoy exists. Vault files are padded to the same size steps, and their dates are kept together.

Inside the decoy, changing the passcode always "works", so it can't be used to test guesses for the real one.

If the new passcode typed in the decoy is the real one and the real vault is backed up, the real codes are wiped from the phone and that passcode opens the decoy. You get them back with your kits.

Limits. This fools a person looking at the phone, not a forensic lab that knows the app. Turn Face ID or fingerprint unlock off for it to mean anything. Your backup is untouched and still needs the kits.

5. Two kits and a PIN

A normal restore: Recovery Kit + PIN, with our server checking the PIN without ever learning it. If we are gone: Recovery Kit + Emergency Kit, with no server at all.

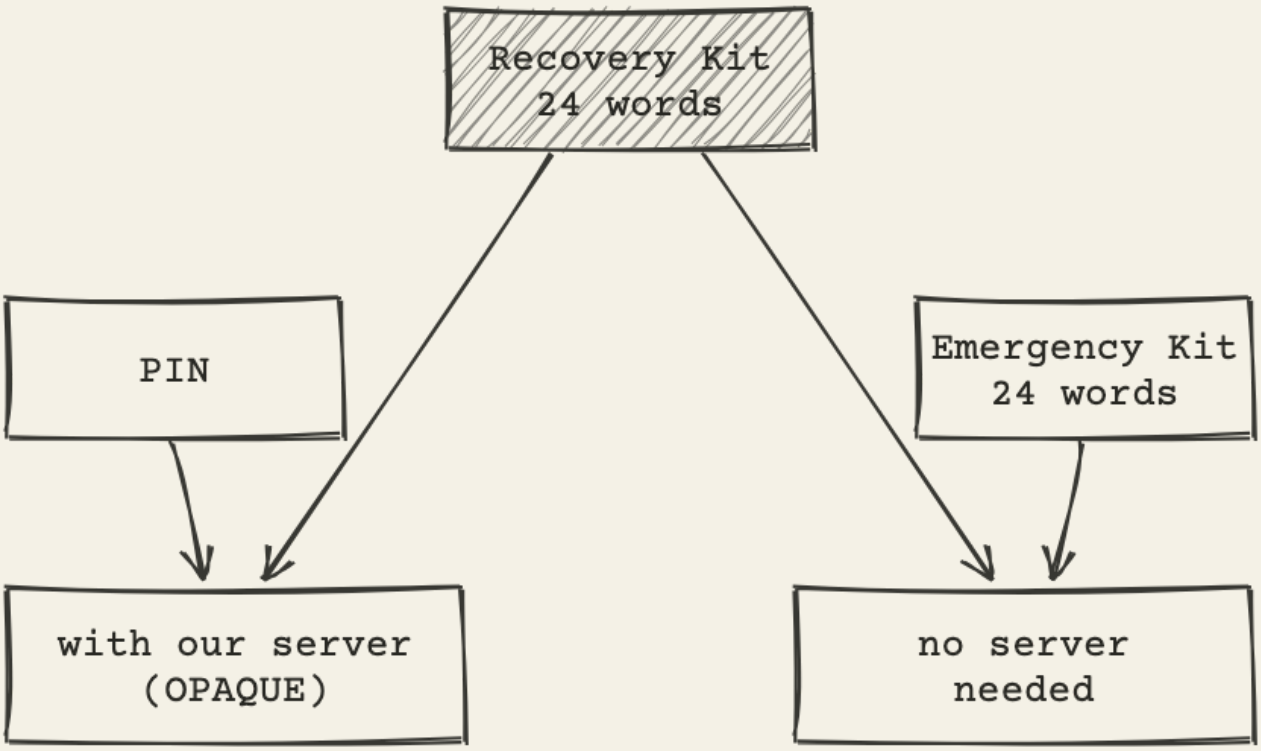


Fig. 2. Each kit is 24 English words (BIP39); the first four letters of each word are enough. The app doesn't keep the kits after setup.

6. How the keys fit

Inside the backup file, the data key is sealed twice. Both seals start from a key made from the Recovery Kit.

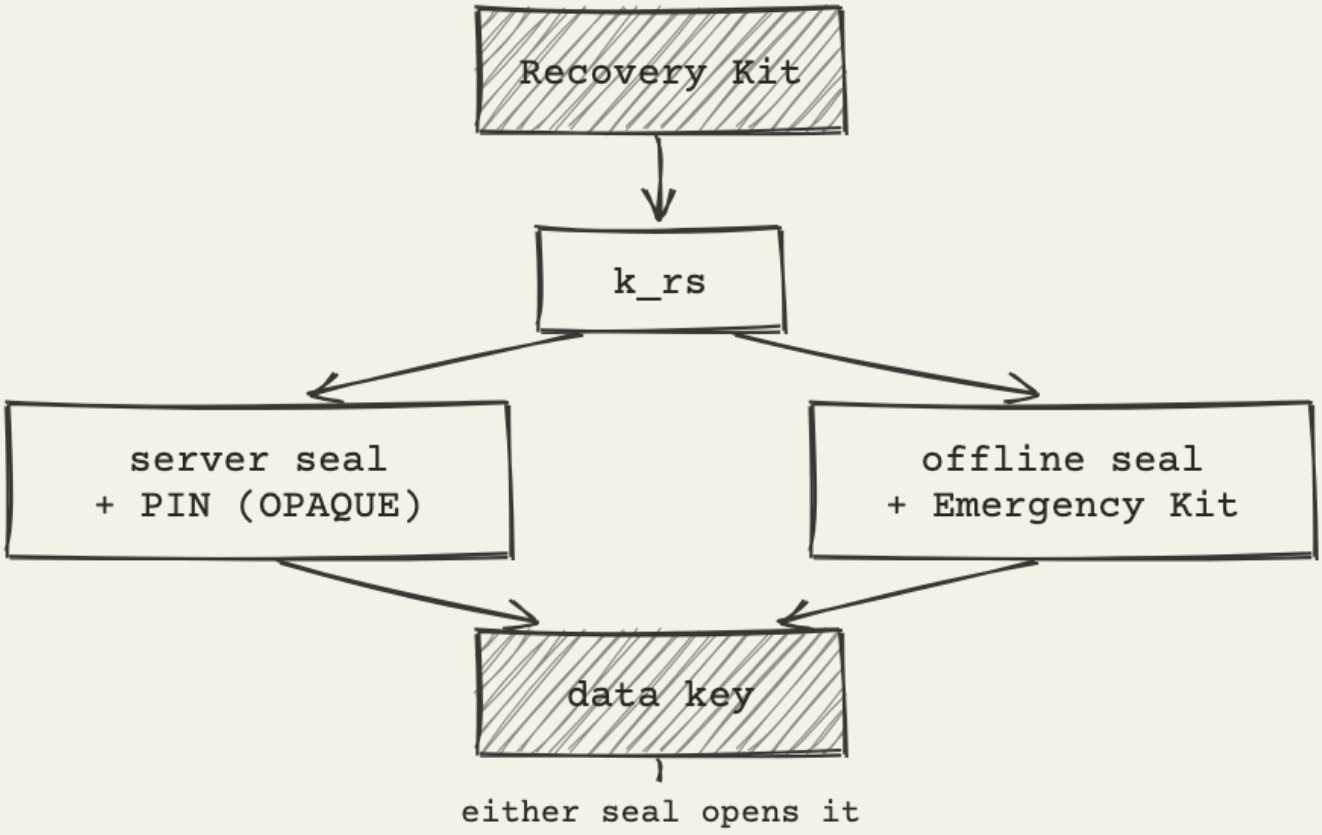


Fig. 3. The backup ID and the key that signs every request also come from the Recovery Kit.

7. The PIN check

The PIN uses OPAQUE: our server keeps a record that lets it check the PIN without learning it, and the PIN never crosses the network. The app knows our server's key and refuses any server with another one.

5 tries an hour. After 10 wrong tries in a row the PIN record is deleted, and only both kits can restore.

Every try must be signed with the Recovery Kit's key, so guessing starts with stealing the kit.

Changing the PIN gives it a fresh server key and shares the same 5-an-hour limit. (Our own review found this route could test guesses; it was fixed before launch.)

Someone who stole everything on our server could try PINs offline. That still leaves them without the Recovery Kit.

8. What each attacker gets

They have	Codes?	Why
The backup file only	No	Sealed with a key only the kits can open
Recovery Kit only	No	Also needs the PIN or the Emergency Kit
Everything on our server	No	Every path also needs the Recovery Kit
Emergency Kit + PIN	No	Both paths start from the Recovery Kit
Recovery Kit + PIN	Yes	The normal restore
Both kits, no server	Yes	The "we are gone" path

Each row is an automated test. We broke the crypto on purpose to check that each test fails when it should.

9. The relay

With backup on, every backup request goes through a relay (Oblivious HTTP). The relay sees your IP address but only sealed data. Our server sees the request but only the relay's address.

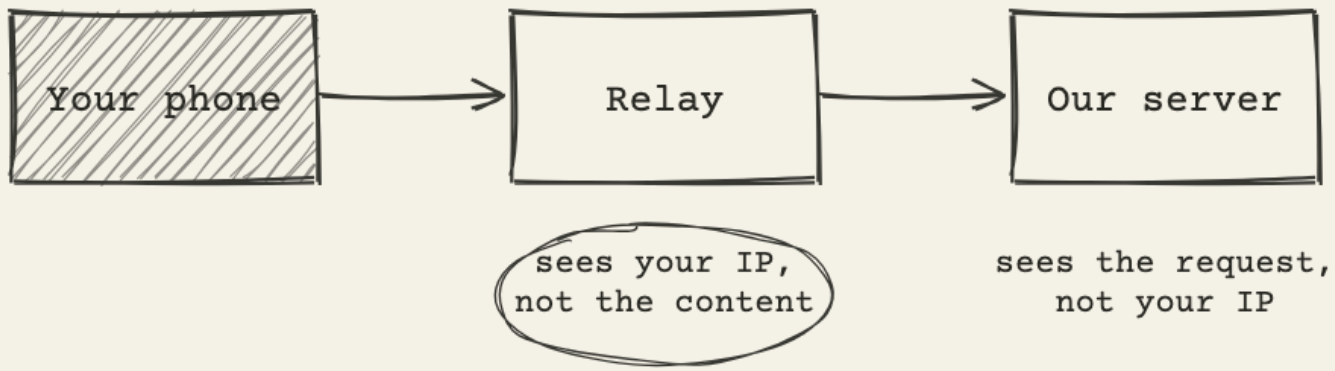


Fig. 4. The relay keeps no logs.

10. Who runs what

Relay: relay.verigate.me, a Cloudflare Worker with logging off.

Server: api.verigate.me on Vercel, Frankfurt.
Database: Neon, Frankfurt.

We run both the relay and the server. Whoever runs both could try to match their traffic by timing. We don't, and the relay keeps nothing to match against, but this is a promise, not maths.

Three small requests go straight to the server and show your IP address to our host: the app's settings (at most once a day), the relay's public key, and a proof-of-work puzzle when you turn backup on. None of them carries your backup ID.

11. What our server stores

What	Why
A random backup ID and a public key	To find your backup; only your phones can change it
Sealed backup: 10 newest versions, plus the newest older than a day and older than a week	Restore, also after a bad change
The PIN check record	Check the PIN without learning it
Wrong-try counts	Stop PIN guessing
Dates made, changed, last used	Delete unused backups
Date and count of the last restore	The "opened elsewhere" notice
One-time request numbers	Stop replays; minutes to 2 days

None of it names you. Each version is at most 256 KB.

12. Signed requests and old versions

Every request is signed with the Recovery Kit's key (Ed25519) and carries a time and a one-time number. Replays, forgeries and stale requests all get the same refusal.

Each backup's version number is sealed together with its content, so a server can't pass an old file off as new. The phone refuses anything older than what it has seen, unless you choose to restore an older version. It also shows the date of the last backup the server confirmed.

The backup ID is checked on open, so nobody can be handed someone else's file.

Keeping the day-old and week-old versions means a thief can't push every good version out in minutes. A proof-of-work puzzle slows mass sign-ups.

13. When your backup is opened elsewhere

The server keeps only the date and the count of the last PIN login or restore. Your other phones check it when you unlock, at most every 6 hours, through the relay.

If it changed, you see: "Your Recovery Kit was used on another phone on (date)". Answer No and the app takes you to new kits. It never shows inside the decoy.

If another phone made new kits, this phone shows "Backup stopped on this phone".

The 6 hours is a setting. A shorter gap means a faster notice, more traffic, and a little more timing data on our side.

14. New kits, turning off, deleting

Backup runs after each change. A backup missed while offline is sent on the next unlock. Nothing is sent while the app is locked: the key only exists while it is open.

New kits make a new data key and re-seal everything. The old backup is retired at once: PIN and uploads stop. Both old kits can still download it for 7 days, then it is deleted, so a stolen phone that makes new kits can't take your backup away.

Turning backup off deletes it after 7 days. Any use from one of your phones cancels that.

A backup unused for 24 months is deleted. Nothing erases every copy at once, on purpose.

15. Moving to a new phone

Screen to screen, with no internet. The new phone shows a one-time key. The old phone seals your accounts to it (HPKE) and shows them as numbered QR parts.

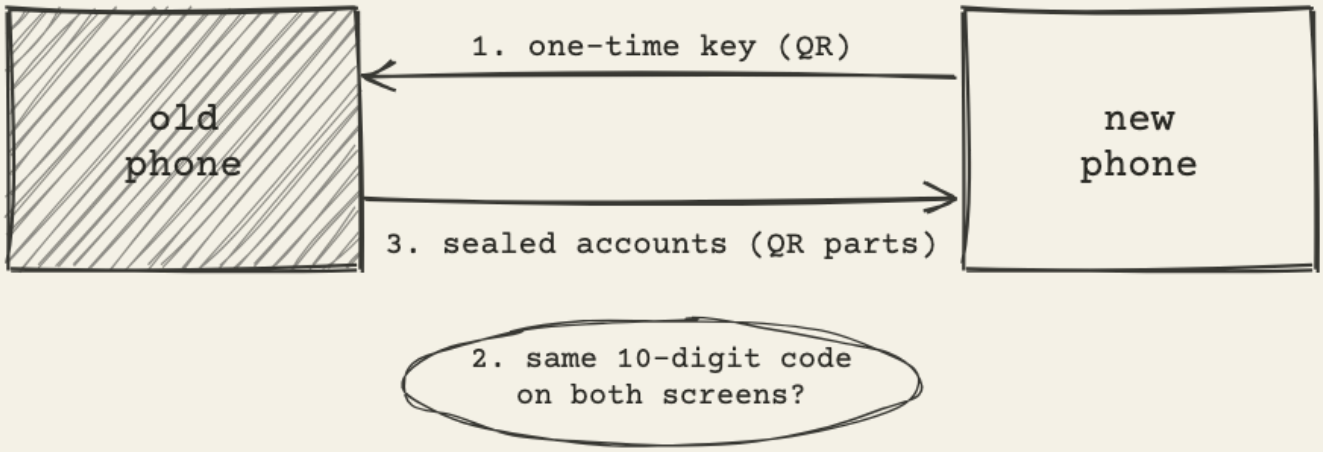


Fig. 5. Both phones show a 10-digit check code; the old one sends only after you confirm they match. Faking a match would take about 10 billion slow hashes in under a minute.

16. Coming in and going out

Import, all on the phone: Google Authenticator export QRs, Proton Authenticator, Aegis, 2FAS and Ente Auth files (plain or password-protected), and any standard otpauth QR. Authy and Microsoft Authenticator block exports. After a plain file import, the app offers to delete the file.

Export: a password-protected file in the Aegis format (scrypt, AES-256-GCM), or Google Authenticator QRs, which list any account they can't carry. One account can also be shown as its own QR.

Badges, groups, sorting and the account check all run on the phone. Nothing is fetched: no brand logos, no lists from the internet.

17. Damaged files and wrong answers

Any single changed byte in a backup is caught. Files are capped at 256 KB, and 4 MB once unpacked. Every file reader is fuzzed: backups, vaults, otpauth links, Google exports, other apps' files, kit words.

An account that fails its checks is set aside and kept, never dropped.

A wrong PIN, wrong kits and a damaged file all give the same error, so the answer teaches an attacker nothing.

The vault keeps fields it doesn't know, so an older app won't damage what a newer one wrote. Deleted accounts leave a bare marker with no secret, for syncing, and go after 90 days.

18. If we shut down

Your codes never need us: the app makes them on the phone, offline, forever.

A saved backup file and both kits restore everything without any server. The file format is documented in this paper's sources and the code is open, so anyone can write a reader.

With backup on, the app checks once a day, through the relay, whether a newer version is needed. A version too old to back up safely says so and offers the store; its codes keep working.

19. Known limits

A phone with malware that can read the screen or has root can read codes. That is out of reach for any app.

With an unlocked Android phone in hand, a forensic lab can guess a short passcode on that phone. Use a longer one for that threat.

iPhone can't block screenshots. The system may keep copies of codes on screen; we wipe our own memory.

Compression can leak size. Exploiting it needs many chosen backups; we accept it for now.

One operator runs the relay and the server. Three setup requests show your IP to our host. Puzzles aren't tied to requests, answers aren't padded, and the relay key has no rotation plan yet.

No outside review yet.

20. Open source and review

The iPhone app, the Android app and the core are open source under the AGPL-3.0: anyone can read, use and change them, and anyone who ships a changed version must publish it.

The code goes public the day VeriGate reaches the stores. The backup server stays closed: it only receives sealed data, this paper lists what it keeps, and the open app decides what it gets.

An outside review of the cryptography is planned. A security contact comes with the code.

Methods used: XChaCha20-Poly1305, Ed25519, OPAQUE, Argon2id, BLAKE2b, HPKE, Oblivious HTTP, BIP39.

END OF PAPER

The code behind every page of this paper goes public when VeriGate reaches the stores.