Private and Secure by Design

How Rocket Vetting Protects Personal Data Through Hashing

A Rocket Vetting White Paper

Author: David Milburn, Rocket Vetting
Published: October 2026

In brief. Rocket Vetting checks your list of people against the government's list of people who have died, without ever seeing the names or Social Security numbers on your list. Our software runs on your own computers, inside your own network. Before anything leaves your building, it turns each name and Social Security number into a coded fingerprint; without our secret key, no one can work out the name from it. We compare fingerprints, not names, and anything we send back to you is locked and unlocked only on your systems. Every file travels over an encrypted connection into a private folder that no other customer can see. These protections go beyond what government rules require.

Summary

Matching a roster (a list of employees, members or beneficiaries) against the Social Security Administration's Limited Access Death Master File (LADMF), SSA's national file of reported deaths, means handling names, Social Security numbers (SSNs) and dates of birth. Rocket Vetting's "Private and Secure" commitment rests on one principle: the matching engine never needs to read a name to know whether two names are the same.

- **Protection happens inside the customer's firewall** (the security boundary around the customer's own network). A bootstrap, a one-time installer, puts every Rocket Vetting launcher on the customer's own hardware: RVP, RVC, Express, and the RVC transaction launcher for post-onboarding roster changes. A launcher is the program that prepares a roster for upload. Every launcher produces the same 24-column protected layout. Each one normalizes, hashes and tokenizes the data (all defined below) before anything leaves. Plain names, SSNs and roster keys never leave the customer environment in readable form; dates of birth and addresses travel over SFTP (secure file transfer protocol), an encrypted file-transfer tunnel. The customer never sends Rocket Vetting a plaintext (readable) roster, as approaches that collect rosters for vendor-side processing require.
- **Matching runs on hashes and identifiers.** The server turns each name hash into an internal identifier and matches against the LADMF on identifiers and hashes, not plain names.
- **The hash is keyed; returns are tokens.** Without the secret key, no one can reverse a hash by guessing common names. Names, SSN parts and roster keys the customer needs back travel as encrypted "RV1" tokens; other case details travel inside the encrypted, private file transfer.
- **All transfers are encrypted and isolated.** Files move over encrypted SFTP, the in-transit protection the LADMF safeguarding guidelines require, into a private folder only the customer's own account can see.

- **Hashing exceeds the requirements.** No applicable government rule requires hashing. Rocket Vetting adds it on top of the required safeguards.

1. Why keyed hashing

Before hashing, every name is normalized: rewritten into the exact standard form SSA uses, so two copies of the same name match letter for letter (see *Names and Dates of Birth in the Death Master File*). Then it is hashed. A hash is a one-way fingerprint. It turns a value such as JOSE into a fixed-length string of characters that cannot be turned back into the name. The same input always gives the same fingerprint, so two systems can confirm a match without exchanging the name itself, much as two people can compare fingerprints without revealing identities.

Technique	Reversible?	Rocket Vetting use
Plain hash (SHA-256)	No, but common values can be recovered by hashing a list of candidates	Not used
Keyed hash (HMAC-SHA256)	No; guessing requires the secret key	Each name part and each SSN part
Encryption (AES, RV1 token)	Yes, by a key holder	Values returned to the customer, including the roster key

Keyed, not plain. Names are not random; common names repeat across millions of people. Anyone can hash a dictionary of common names and look for a plain hash in the results. That is a dictionary attack. A keyed hash (HMAC, short for hash-based message authentication code) mixes a secret key into every fingerprint. Without the key, the dictionary matches nothing, like trying keys on a lock without knowing its shape.

No per-record salt. Password systems add a salt, a random extra value, so identical passwords produce different hashes. Rocket Vetting deliberately does not. Matching requires the same name to produce the same hash on the roster and in the LADMF reference data. The secret key supplies the protection a salt would otherwise provide.

Tokens for values that come back. A token is a value locked with encryption (here AES, the federal encryption standard). Unlike a hash, a token can be unlocked, but only with the key. Rocket Vetting uses tokens for values the customer needs returned.

2. The vetting pipeline

The bootstrap installs the launchers and reentry (the program that loads results back into the customer's database) on the customer's own hardware. Steps 1–3 and 9 run inside the customer's firewall; steps 4–8 run at Rocket Vetting. Each customer's SFTP folder is chrooted: locked so that account sees only its own folder, like a private mailbox in a shared building.

Step	Where	What happens	Data form
1. Map fields	Customer firewall: customer	Customer maps roster columns to name parts, SSN, date of birth and roster key	Plain, in customer systems

Step	Where	What happens	Data form
2. Normalize	Customer firewall: RV launcher	Each name part normalized per SSA's per-part rules (POMS RM 10205.125)	Plain, in memory
3. Hash and tokenize	Customer firewall: RV launcher	Same 24-column protected layout from every launcher. HMAC-SHA256 (UTF-8, 64-character uppercase hex) of each name part and SSN part, hashed separately; blanks left blank. RV1 token alongside each hash; the roster key is tokenized only. Date of birth, address and benefit date are not hashed	Hashes and tokens; other fields plain inside the file
4. Transfer	Encrypted SFTP to RV	Batch uploaded over encrypted SSH (SFTP) to the inbox folder of the customer's own chrooted SFTP directory, then moved to its processing folder.	Encrypted in transit; per-account directory
5. Look up by hash	RV server	Each name hash looked up in a reference table, created if new, and assigned an internal id	Hashes and ids
6. Identity key	RV server	Name ids stored in a PersonIdentityKey record	Ids
7. Match	RV server	LADMF loader computes the same keyed hash for SSA's names exactly as delivered; roster and LADMF identities compared on ids and hashes	Ids and hashes
8. Return results	RV to customer	Case files written to the outbox folder of the customer's chrooted directory. Decedent names, SSN parts and the roster key are RV1 tokens; the server tokenizes SSA's decedent names with the same shared key. Other case details (addresses, cemetery data including any spouse name, and the customer's own contact details) are not tokenized	Tokens for names, SSN parts and roster key; other fields plain inside the file; encrypted in transit; per-account directory
9. Reentry	Customer firewall: RV reentry	Tokens decrypted and loaded into case tables; roster key restored inside the customer's database (Section 4)	Decrypted, in encrypted database

In testing with a dummy key, the server's hash service and the launcher produced identical fingerprints for all 93,536 test values.

The roster key never leaves the customer in plain form. The roster key is the customer's own record number, such as a voter registration id. The launcher converts it to an RV1 token before upload. The token is deterministic AES encryption, not a hash: the same roster key always yields the same token, and the shared key unlocks it. Rocket Vetting uploads and stores the roster key (RVC) only as this token. Each case record carries the token back, and reentry unlocks it inside the customer's own database. Rocket Vetting never decrypts or stores the plain roster key.

3. Key management

With a keyed hash, the key is the security model. Anyone holding the key can hash candidate values and compare. SSN parts have few possible values: 1,000 for the first three digits (area), 100 for the middle two (group) and 10,000 for the last four (serial).

- **Key file on customer hardware.** The bootstrap places the key file in the secrets directory on the customer's hardware. Protecting that file is the customer's responsibility.
- **Strict format.** Launchers and reentry load the key through one shared module. The key must be valid base64 text (a standard way of writing binary data) that decodes to exactly 32 bytes. Startup fails if a legacy source conflicts.
- **Fingerprints, never keys.** Each component logs only a short key fingerprint, a brief code derived from the key. Operators can confirm all hosts share a key without exposing it. The key is never accepted on the command line.
- **The key is shared with Rocket Vetting.** The same key is configured on Rocket Vetting's servers, which compute hashes of SSA's names for matching and tokenize decedent names and SSN parts for case files. The key is held only on the customer's hardware and on Rocket Vetting's servers.
- **One key, two uses.** A single key is used for name hashing (HMAC) and token encryption (AES). Outputs carry version markers (HASH_VERSION, TOKEN_VERSION); changing the key requires re-hashing the reference data.

What customers do

- **Map fields.** Identify which roster columns hold name parts, SSN, date of birth and roster key.
- **Restrict the secrets directory** to the service account (the system login) that runs the launchers and reentry: for example, a Windows access-control list limited to that account, or chmod 600 on Linux.
- **Treat the key file like any credential:** limit who can read it, and never copy it into email, tickets or shared folders.

4. Reentry

Reentry runs on the customer's hardware. It writes case data and restores the roster key into rv_cms_case tables in the customer's own database, alongside the customer's roster source data. That database is encrypted at rest (while stored). Reentry does no hashing. It unlocks RV1 tokens and writes the results, with three safeguards:

- If any token fails to unlock, processing stops and the file is kept.
- Files are deleted only after the database confirms the save.
- Database errors hide the values involved, so names and SSN parts never appear in error logs.

5. Alignment with government requirements

The rules for LADMF data require encryption in transit, protection at rest, access control and accountability. None requires hashing. Rocket Vetting meets the in-transit requirement with encrypted SFTP. It goes beyond the requirements by hashing names and SSN parts with a secret key before they leave the customer's environment.

Source	What it requires or recommends	Hashing?
15 CFR Part 1110 (LADMF certification)	Defines DMF information as name, SSN, date of birth and date of death. §1110.102(a)(2): safeguards reasonably similar to IRC §6103(p)(4). Certification renewed with NTIS (§1110.105); independent attestation based on an assessment within the prior three years (§1110.101(b)); NTIS audits	Not mentioned
NTIS Publication 100 (safeguarding guidelines)	SC-8: cryptographic protection in transit. SC-28: protection of LADMF information at rest	Not mentioned
NIST SP 800-122 (PII guidance)	Minimization, de-identification, access control, encryption; one-way hashing named "a common de-identification technique"	Optional
GLBA Safeguards Rule, 16 CFR 314.4(c)(3)	Where customers are financial institutions: encrypt customer information in transit and at rest	Not required
State breach laws (e.g., Cal. Civ. Code §1798.82; N.Y. GBL §899-aa)	Notification triggered by unencrypted name plus SSN; encrypted data generally a safe harbor	Not required

Private, per-account SFTP folders support the need-to-know access control these safeguards call for: each customer sees only its own files. Dates of birth and addresses need encryption and access control, not hashing; address is not an LADMF element. An unkeyed hash of values with few possibilities, such as dates of birth or SSNs, can be reversed by trying every value. That is why Rocket Vetting's hashing is keyed. This summary is not legal advice; each organization should confirm its own obligations.

FSC goes through an annual certification process with NTIS and an independent attestation and audit every three years under the LADMF certification program.

6. What hashing does and does not protect

Protects against	Does not protect against
Readable names, SSN parts and roster keys in batch files and matching data, from every launcher	A party holding the key, who can recompute hashes for candidate values
Dictionary guessing by anyone without the key	Frequency patterns: identical names produce identical hashes, by design
One customer seeing another's files: each account has its own chrooted SFTP directory (inbox, processing, outbox)	Fields that are not hashed or tokenized (date of birth, addresses, cemetery data including any spouse name), which rely on encryption in transit and access control
Personal data in logs: only key fingerprints are logged, and database parameters are hidden	Data after reentry decrypts it, which relies on the customer database's encryption at rest and access control

7. Conclusion

Rocket Vetting matches names without reading them. Normalization, hashing, tokenization and reentry run inside the customer's own firewall, so plain names, SSNs and roster keys never leave in readable form. Matching runs on fingerprints and internal identifiers. Names, SSN parts and roster keys return as locked tokens, other case details travel in private, encrypted transfers, and every transfer is encrypted. The design meets the in-transit requirement of the LADMF safeguarding guidelines and adds keyed

hashing that no rule requires. Its strength depends on two things: custody of the key, and protection of data after reentry unlocks it. Being private and secure means being precise about where protection begins and ends.

About Rocket Vetting

Rocket Vetting (rocketvetting.com) is a death-record vetting service, publicly launched on September 22, 2026, that matches customer rosters against the LADMF. Its launchers, installed on the customer's hardware, normalize, hash and tokenize names, SSN parts and roster keys inside the customer's firewall before data leaves. All transfers use encrypted SFTP, and matching runs on keyed hashes and internal identifiers rather than plain names. Under the LADMF certification program (15 CFR Part 1110), FSC goes through an annual certification process with NTIS and an independent attestation and audit every three years.

References

1. NIST. *FIPS 198-1*, *The Keyed-Hash Message Authentication Code (HMAC)*. https://csrc.nist.gov/pubs/fips/198-1/final
2. NIST. *FIPS 180-4*, *Secure Hash Standard*. https://csrc.nist.gov/pubs/fips/180-4/upd1/final
3. NIST. *FIPS 197*, *Advanced Encryption Standard*. https://csrc.nist.gov/pubs/fips/197/final
4. 15 CFR Part 1110, Certification Program for Access to the Death Master File. https://www.ecfr.gov/current/title-15/subtitle-B/chapter-XI/part-1110
5. NTIS. *Publication 100*, *Limited Access Death Master File Safeguarding Guidelines* (June 2016). https://ladmf.ntis.gov/docs/NTIS-DMFsecurityGuidelinesv14.pdf
6. NIST. *SP 800-122*, *Guide to Protecting the Confidentiality of PII*. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf
7. NIST. *SP 800-53 Rev. 5* (controls SC-8, SC-28). https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final
8. 16 CFR 314.4, FTC Safeguards Rule. https://www.ecfr.gov/current/title-16/chapter-I/subchapter-C/part-314/section-314.4
9. California Civil Code §1798.82. https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=CIV&sectionNum=1798.82
10. New York General Business Law §899-aa. https://www.nysenate.gov/legislation/laws/GBS/899-AA
11. Rocket Vetting. *Names and Dates of Birth in the Death Master File: How They Are Formatted, and Why Rocket Vetting Follows Suit*. October 2026.
12. Rocket Vetting. Internal code review and hashing cross-check (dummy key), October 2026.