Network Working Group
Internet-Draft
Intended status: Informational
Expires: 7 July 2026

O. Keskilammi
Self-published

3 January 2026

Secure Mobile Vault Format
draft-voyager-smv-specification-00

Abstract

The Secure Mobile Vault Format (SMVF) defines a binary container format for storing encrypted password vaults on mobile devices. The format is designed to be offline-first, zero-knowledge, cryptographically robust, and forward-compatible. SMVF specifies strict structural layout, authenticated encryption, and deterministic metadata handling suitable for constrained mobile environments.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/voyager-2021/smv-specification.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 July 2026.

Copyright Notice

Table of Contents

1. Status of This Memo 2
2. Copyright Notice 3
3. Introduction 3
4. Requirements Language 3
5. Design Goals 3
6. File Overview 4
7. File Header 4
8. Section Model 5
9. Section Types 5
10. KDF Parameters Section 5
11. Crypto Parameters Section 6
12. Encrypted Vault Section 7
13. Vault Payload Structure 7
14. Vault Entry Structure 8
15. Authentication and Integrity 8
16. Atomic Update Requirements 9
17. Memory Handling Requirements 9
18. Forward Compatibility Rules 9
19. Explicit Non-Goals 9
20. Threat Model Summary 10
21. IANA Considerations 10
22. Security Considerations 10
23. References 10
23.1. Normative References 10
23.2. Informative References 11
Acknowledgments 11
Authors' Addresses 11

1. Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

2. Copyright Notice

3. Introduction

The Secure Mobile Vault Format (SMVF) defines a binary container format for storing encrypted password vaults on mobile devices. The format is designed to be offline-first, zero-knowledge, cryptographically robust, and forward-compatible.

SMVF follows strict principles including explicit headers, typed sections, cryptographic framing, and strict versioning, while remaining minimal and suitable for mobile environments.

4. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

5. Design Goals

The format is designed to meet the following goals:

1. Provide zero-knowledge encryption where vault contents cannot be recovered without the master password.
2. Support offline-only operation with no network dependencies.
3. Be safe for mobile storage, including crash-safe atomic updates.
4. Use authenticated encryption to ensure confidentiality and integrity.
5. Allow future evolution without breaking existing vaults.
6. Remain auditable and deterministic in layout.

6. File Overview

An SMVF file consists of the following components, in order:

* File Header (fixed-size)
* KDF Parameters Section
* Crypto Parameters Section
* Encrypted Vault Section
* Optional Footer

All multi-octet integer fields are encoded in network byte order (big-endian). The file is tightly packed and contains no padding.

The standard file extension is ".smvf".

7. File Header

The file header is a fixed-size structure of 32 octets.

The header contains the following fields:

* Magic (4 octets) Identifies the file as an SMVF container. The value MUST be the ASCII string "SMVF".
* Major Version (2 octets) Indicates the major format version. Readers MUST reject files with unsupported major versions.
* Minor Version (2 octets) Indicates backward-compatible revisions.
* Header Length (4 octets) Total length in octets of all sections except the Encrypted Vault Section length preceding the encrypted payload.
* Flags (4 octets) Bitmask defining file properties.
* File UUID (16 octets) Randomly generated UUID version 4 identifier as defined in [RFC9562]. This value is non-secret and MUST NOT be reused.

The following header flags are defined:

* Bit 0: Encrypted payload present (MUST be set)
* Bit 1: Footer present

* Bits 2-31: Reserved and MUST be zero

8. Section Model

Sections follow a typed-length-value (TLV) model.

Each section consists of:

* Section Type (2 octets)

* Section Length (4 octets)

* Section Value (variable length)

The section length specifies the length of the section value only.

Unknown section types MUST be skipped using the length field. The section order defined in this specification MUST be preserved.

9. Section Types

The following section types are defined:

* 0x0001: KDF Parameters Section (REQUIRED)

* 0x0002: Crypto Parameters Section (REQUIRED)

* 0x0003: Encrypted Vault Section (REQUIRED)

* 0x0004-0x7FFF: Reserved for future use and MUST NOT be used

No section identifier may be reused for a different purpose.

10. KDF Parameters Section

This section defines how the encryption key is derived from the master password.

The section value contains the following fields:

* KDF Algorithm (1 octet)

* Salt Length (1 octet)

* Salt (variable length)

* Cost Parameter A (4 octets)

* Cost Parameter B (4 octets)

* Cost Parameter C (4 octets)

The following KDF Algorithm identifiers are defined:

* 0x01: Argon2id as defined in [RFC9106]

* 0x02: scrypt as defined in [RFC7914]

* 0x03-0x7F: Reserved for future use and MUST NOT be used

The salt is not secret and MUST be generated randomly per vault.

For Argon2id, the parameters are defined as follows:

* Parameter A: Memory cost in KiB

* Parameter B: Iteration count

* Parameter C: Degree of parallelism

For scrypt, the parameters are defined as follows:

* Parameter A: N (CPU/memory cost)

* Parameter B: r

* Parameter C: p

Implementations SHOULD tune parameters for mobile hardware while maintaining resistance to offline attacks.

11. Crypto Parameters Section

This section defines the encryption algorithm and parameters used to protect the vault payload.

The section value contains the following fields:

* Cipher Algorithm (1 octet)

* Key Length (1 octet)

* Nonce Length (1 octet)

* Authentication Tag Length (1 octet)

* Nonce (variable length)

The following Cipher Algorithm identifiers are defined:

* 0x01: AES-256-GCM as defined in [RFC4106]

* 0x02: ChaCha20-Poly1305 as defined in [RFC8439]

* 0x03-0x7F: Reserved for future use and MUST NOT be used

For AES-256-GCM and ChaCha20-Poly1305, the Key Length MUST be 32 octets.

Implementations MUST ensure that a nonce is never reused with the same derived key. The authentication tag is embedded in the AEAD ciphertext and is not stored separately.

12. Encrypted Vault Section

This section contains the encrypted vault payload.

The payload is produced using an AEAD construction with the following inputs:

* The encryption key is derived from the master password using the KDF Parameters Section.

* The nonce is taken from the Crypto Parameters Section.

* The additional authenticated data consists of the File Header, the KDF Parameters Section, and the Crypto Parameters Section.

All metadata is authenticated but not encrypted. Any modification to these components MUST cause decryption failure.

The AAD input MUST be the exact serialized octet sequence of the File Header followed by the serialized KDF Parameters Section and the serialized Crypto Parameters Section, in file order.

13. Vault Payload Structure

Before encryption, the vault payload is serialized JSON as defined in [RFC8259]. All JSON strings MUST be encoded in UTF-8.

No canonicalization of the JSON payload is required, as integrity is provided by the enclosing AEAD construction.

The top-level object contains the following fields:

* vault_version: Integer indicating the logical vault schema version
* created: [RFC3339] timestamp
* updated: [RFC3339] timestamp
* entries: Array of vault entries
* metadata: Optional application-defined metadata

14. Vault Entry Structure

Each vault entry is a [RFC8259] JSON object containing the following fields:

* id: UUID version 4 string as defined in [RFC9562]
* type: Entry type identifier
* title: Human-readable name
* fields: Key-value mapping of entry fields
* notes: Optional freeform text
* tags: Optional array of strings
* created: [RFC3339] timestamp
* updated: [RFC3339] timestamp

The type field allows future extension to support non-password secrets.

15. Authentication and Integrity

The format relies exclusively on AEAD for security guarantees.

The following properties are provided:

* Confidentiality of vault contents
* Integrity of vault contents
* Integrity and authenticity of metadata

No separate MAC or digital signature is required.

16. Atomic Update Requirements

Implementations MUST perform updates atomically to prevent vault corruption.

A recommended procedure is as follows:

1. Write the updated vault to a temporary file.
2. Flush and synchronize the file.
3. Atomically rename the temporary file over the existing vault.
4. Optionally retain a backup copy.

Mobile operating systems guarantee atomic rename operations within the same filesystem.

17. Memory Handling Requirements

Implementations MUST enforce the following requirements:

* Derived encryption keys MUST NOT be persisted.
* Keys MUST be zeroized on lock or application backgrounding.
* Decrypted vault data MUST reside only in memory and MUST be discarded on lock or suspension.

18. Forward Compatibility Rules

Readers MUST reject files with unsupported major versions.

Readers MUST ignore unknown section types.

Writers MUST preserve section ordering and MUST NOT reuse section identifiers.

19. Explicit Non-Goals

The format intentionally excludes:

* Cloud synchronization
* Compression
* Partial encryption

* Password reuse detection

* Replay protection

20. Threat Model Summary

The format mitigates:

* Offline file theft

* Unauthorized modification

* Metadata tampering

The format does not mitigate:

* Compromised or rooted operating systems

* Active runtime memory attacks

* Screen capture malware

21. IANA Considerations

This document has no IANA actions.

22. Security Considerations

This document specifies a cryptographic container format intended to protect sensitive data at rest. Security properties and assumptions are discussed throughout the document, including key derivation, authenticated encryption, and memory handling requirements.

The format assumes a trusted execution environment and does not protect against compromised operating systems, runtime memory disclosure, or malicious software with sufficient privileges. Implementers are responsible for selecting appropriate cryptographic parameters and ensuring correct use of underlying cryptographic primitives.

23. References

23.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <https://www.rfc-editor.org/rfc/rfc4106>.

[RFC7914] Percival, C. and S. Josefsson, "The scrypt Password-Based Key Derivation Function", RFC 7914, DOI 10.17487/RFC7914, August 2016, <https://www.rfc-editor.org/rfc/rfc7914>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/rfc/rfc8439>.

[RFC9106] Biryukov, A., Dinu, D., Khovratovich, D., and S. Josefsson, "Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications", RFC 9106, DOI 10.17487/RFC9106, September 2021, <https://www.rfc-editor.org/rfc/rfc9106>.

23.2. Informative References

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/rfc/rfc9562>.

Acknowledgments

Authors' Addresses

Ohto Keskilammi Email: voyager-2019@outlook.com