Abcrypt Encrypted Data Format

Shun Sakai

Version 0.4.0, 2025-05-01

abcrypt is a modern file encryption format with the data authenticity. This document describes the abcrypt encrypted data format.

This document and the abcrypt encrypted data format are distributed under the terms of the CC0 1.0 Universal (CC0 1.0). You can copy, modify, distribute and perform these, even for commercial purposes, all without asking permission.

Table of Contents

Chapter 1. Introduction

abcrypt is a modern file encryption format inspired by the scrypt encrypted data format, which is produced by the scrypt encryption utility. abcrypt uses Argon2 for key derivation, BLAKE2b-512-MAC for header integrity checking and XChaCha20-Poly1305 for encryption.

Chapter 2. Conventions used in this document

Argon2 is the key derivation function from RFC 9106.

BLAKE2b-512-MAC is the keyed hash function based on BLAKE2 standardized in RFC 7693. This uses BLAKE2b and always outputs a 64-byte MAC.

XChaCha20-Poly1305 is the AEAD algorithm from draft-irtf-cfrg-xchacha.

Chapter 3. Format overview

An abcrypt file is composed of two parts: the header containing the required data and the payload encrypted with the derived key.

Table 1. The structure of the abcrypt encrypted data format

Offset	Bytes	Description	Detail
0	7	Magic number ("abcrypt").	Magic number
7	1	Version number.	Version number
8	4	Argon2 type.	Argon2 type
12	4	Argon2 version.	Argon2 version
16	4	Memory size `m` (`memoryCost`).	Argon2 parameters
20	4	Number of iterations `t` (`timeCost`).	Argon2 parameters
24	4	Degree of parallelism `p` (`parallelism`).	Argon2 parameters
28	32	Salt for Argon2.	Salt for Argon2
60	24	Nonce for XChaCha20-Poly1305.	Nonce for XChaCha20-Poly1305
84	64	MAC of the header.	Header MAC
148	n	Ciphertext.	Payload
$148 + n$	16	MAC of the ciphertext.	Payload

All multibyte values are stored in little-endian.

Chapter 4. Key derivation

The derived key for computing the header MAC and the derived key for encryption are produced by Argon2.

The derived key is produced as follows

```
derivedKey = Argon2(
    password = password,
    salt = header[28..60],
    parallelism = header[24..28],
    tagLength = 96,
    memoryCost = header[16..20],
    timeCost = header[20..24],
    version = header[12..16],
    secretKey = [],
    associatedData = [],
    type = header[8..12],
)
```

The size of `secretKey` (pepper) and `associatedData` (associated data) are zero (empty).

The resulting derived key (`derivedKey`) length is 96 bytes. The first 32 bytes of `derivedKey` are the XChaCha20-Poly1305 key (`encryptionKey`) for encryption, and the last 64 bytes are the BLAKE2b-512-MAC key (`headerMacKey`) for computing the header MAC.

The derived key is split as follows

```
encryptionKey = derivedKey[..32]
headerMacKey = derivedKey[32..]
```

`type`, `version`, `memoryCost`, `timeCost`, `parallelism`, and `salt` used when encrypting are stored in the header, and these stored values are used when decrypting.

Chapter 5. Header format

5.1. Magic number

A 7-byte string for identifying the abcrypt encrypted data format. The value is "abcrypt" (`61 62 63 72 79 70 74` in hex).

5.2. Version number

A 1-byte version number of the abcrypt encrypted data format. The current value is 1.

5.3. Argon2 type

Table 2. The following Argon2 types are valid

Value	Description
0	Argon2d.
1	Argon2i.
2	Argon2id.

The Argon2 type is represented as 4 bytes in little-endian.

5.4. Argon2 version

Table 3. The following Argon2 versions are valid

Value	Description
16	Version 0x10 (16 in decimal).
19	Version 0x13 (19 in decimal).

The Argon2 version is represented as 4 bytes in little-endian.

5.5. Argon2 parameters

Table 4. Argon2 has the following parameters that control

Parameter	Minimum value	Maximum value	Description
`memoryCost`	$8 \times p$	$2^{32} - 1$	Memory size in KiB.
`timeCost`	1	$2^{32} - 1$	Number of iterations.
`parallelism`	1	$2^{24} - 1$	Degree of parallelism.

Each parameter is represented as 4 bytes in little-endian.

5.6. Salt for Argon2

A 32-byte salt for Argon2.

The salt should be generated from a CSPRNG.

5.7. Nonce for XChaCha20-Poly1305

A 24-byte nonce for XChaCha20-Poly1305.

The nonce should be generated from a CSPRNG.

5.8. Header MAC

The MAC (authentication tag) of the header. The MAC is computed with BLAKE2b-512-MAC over the whole header up to and including the nonce (first 84 bytes of the header).

The MAC is computed as follows

```
mac = BLAKE2b(
    data = header[..84],
    digestLength = 64,
    key = headerMacKey,
    salt = [],
    personalization = [],
)
```

The size of `salt` and `personalization` (personalization string) are zero (empty).

Chapter 6. Payload

The payload is encrypted with XChaCha20-Poly1305.

The ciphertext is computed as follows

```
ciphertext = XChaCha20-Poly1305(
    plaintext = plaintext,
    aad = [],
    key = encryptionKey,
    nonce = header[60..84],
)
```

The size of `aad` (additional authenticated data) is zero (empty).

`nonce` used when encrypting is stored in the header, and the stored value is used when decrypting.

The abcrypt encrypted data format uses a postfix tag.

Chapter 7. Filename extension

abcrypt files should use the extension `.abcrypt`.

Chapter 8. MIME type

When transferring abcrypt files over the Internet, the appropriate MIME type is `application/x-abcrypt`.

Chapter 9. ABNF definition of the file format

```
abcrypt = header payload

; Header

header = signature version-number argon2-type argon2-version argon2-parameters argon2-
salt xchacha20-poly1305-nonce header-mac

signature                = %s"abcrypt"             ; magic number
version-number           = %x01                    ; current version number
argon2-type              = %x00000001-00000003     ; Argon2 type
argon2-version           = %x00000010 / %x00000013 ; Argon2 version
argon2-salt              = 32OCTET                 ; 32-byte salt for Argon2
xchacha20-poly1305-nonce = 24OCTET                 ; 24-byte nonce for XChaCha20-
Poly1305
header-mac               = 64OCTET                 ; BLAKE2b-512-MAC of the header

; Argon2 parameters

argon2-parameters = memory-cost time-cost parallelism

memory-cost = %x00000008-FFFFFFFF ; memory size in KiB
time-cost   = %x00000001-FFFFFFFF ; number of iterations
parallelism = %x00000001-00FFFFFF ; degree of parallelism

; Payload

payload = ciphertext ciphertext-mac

ciphertext     = *OCTET  ; encrypted with XChaCha20
ciphertext-mac = 16OCTET ; Poly1305 of the ciphertext
```

Chapter 10. Format changelog

Version 1

- Add the Argon2 type field to allow choosing the Argon2 type.
- Add the Argon2 version field to allow choosing the Argon2 version.

Version 0

- Initial release.