Workgroup:	Network Working Group
Internet-Draft:	draft-vasanko-chacha20-blake3-00
Published:	13 October 2026
Intended Status:	Informational
Expires:	16 April 2027
Author:	L. Vasanko

ChaCha20-BLAKE3 Authenticated Encryption

Abstract

This document specifies ChaCha20-BLAKE3, an authenticated encryption construction combining the ChaCha20 stream cipher defined in RFC 8439 with the BLAKE3 keyed hash function. The construction follows ChaCha20-Poly1305, replacing Poly1305 with BLAKE3 while preserving the established key, nonce, and authentication conventions.

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 16 April 2027.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 2
2. Construction 3
2.1. Parameters 3
2.2. Encryption 3
2.3. Authentication 3
2.4. Verification 4
3. Message Representation 4
4. Applications 4
4.1. Key Commitment 4
4.2. Signed Authentication Tags 5
5. Security Considerations 5
6. IANA Considerations 5
7. References 6
7.1. Normative References 6
7.2. Informative References 6
Appendix A. Test Vectors 6
A.1. Authentication Key Derivation 6
A.2. Authenticated Encryption 7
A.3. Empty Message 7
Author's Address 8

1. Introduction

ChaCha20-BLAKE3 combines ChaCha20 encryption with BLAKE3 authentication, supporting applications requiring key commitment or digital signatures over message authentication tags.

The abbreviated name `chab3` **MAY** be used in software identifiers, and other contexts where the full algorithm name is inconvenient.

An independently developed ChaCha20-BLAKE3 construction was published by Kerkour [KERKOUR]. Unlike that construction, this specification closely follows the established ChaCha20-Poly1305 design of RFC 8439, retaining its key derivation, nonce conventions, and encryption process while replacing Poly1305 with BLAKE3. The constructions also differ in authentication input formatting and are not interoperable.

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Construction

2.1. Parameters

Parameter	Definition
KEY	32-byte encryption key
NONCE	12-byte nonce
MACLEN	Authentication tag length, default 16

Table 1

All lengths are expressed in bytes unless otherwise stated.

2.2. Encryption

The construction uses the IETF ChaCha20 variant defined in [RFC8439].

As in Section 2.6 of [RFC8439], the first 32 bytes of ChaCha20 block 0 form the one-time authentication key. The remaining 32 bytes are discarded.

Encryption and decryption use ChaCha20 beginning with block counter 1.

2.3. Authentication

The authenticated input follows Section 2.8 of [RFC8439], replacing 16-byte padding with 64-byte padding:

```
AAD || pad64(AAD) ||
ciphertext || pad64(ciphertext) ||
len64(AAD) || len64(ciphertext)
```

Here:

- `pad64(x)` consists of the minimum number of zero bytes required to make the length of `x` a multiple of 64.
- `len64(x)` is the length of `x` encoded as an unsigned 64-bit little-endian integer.

The padding aligns the authenticated components to BLAKE3's 64-byte compression blocks, allowing implementations to compress each block immediately without maintaining buffered copies. This also keeps alignment with the encrypted ChaCha20 64-byte blocks.

The authentication tag is the first MACLEN bytes of BLAKE3's keyed, extendable output [BLAKE3], using the one-time authentication key derived in Section 2.2. BLAKE3 computation **MAY** be omitted when MACLEN is zero.

2.4. Verification

The receiver **MUST** verify the authentication tag before releasing plaintext. Authentication failure **MUST NOT** release unauthenticated plaintext.

MACLEN **MUST** be established by the surrounding protocol and **MUST NOT** be inferred from the message. Accepting a tag length derived from attacker-controlled input would permit tag truncation attacks.

3. Message Representation

The nonce and authentication tag **MAY** independently be attached to or detached from the ciphertext.

Attached components appear in the following order:

```
[nonce][ciphertext][tag]
```

Detached components are supplied separately and omitted from the serialized message.

The representation does not affect encryption or authentication.

4. Applications

4.1. Key Commitment

The construction is intended to provide key commitment, making it infeasible for an authenticated ciphertext to verify under multiple encryption keys.

This is useful in multi-recipient encryption, where otherwise a maliciously constructed ciphertext might authenticate under different keys chosen by attacker and decrypt to different plaintexts.

The security properties and bounds are discussed in Section 5.

4.2. Signed Authentication Tags

A digital signature over an authentication tag can bind the signature to the complete authenticated content, including the ciphertext, AAD, and nonce, without requiring a separate hash operation over the message.

The signed tag must resist substitution of any authenticated content, including attempts involving different encryption keys. A MACLEN of 32 is **RECOMMENDED** for this application.

The authentication tag **MAY** be omitted from the transmitted message when a signature is provided instead. The recipient then recomputes the tag using the encryption key and verifies its signature. Plaintext **MUST NOT** be released before successful signature verification.

The signature algorithm and verification protocol are outside the scope of this document.

5. Security Considerations

The nonce **MUST** be unique for every encryption performed under the same key. The nonce reuse considerations of [RFC8439] apply.

The construction relies on the pseudorandomness of ChaCha20 output and the security of BLAKE3 in keyed hashing mode. A 16-byte tag provides up to 128-bit generic forgery resistance.

Key commitment (Section 4.1) and signed-tag integrity are subject to two generic attacks:

1. **Targeted substitution (preimage).** Given an existing tag, find another key-message combination producing the same tag. A tag of 16 bytes or longer provides up to 128-bit resistance.
2. **Chosen-pair collision.** Find two distinct key-message combinations producing the same tag. A 16-byte tag provides 64-bit resistance, while a 32-byte tag provides 128-bit resistance.

These estimates assume ideal cryptographic behavior and do not constitute a formal security proof of the complete construction. The properties of AEAD algorithms are further discussed in [RFC9771].

The maximum plaintext length is 256 GiB minus 64 bytes, inherited from IETF ChaCha20. This limit is not expected to constrain typical authenticated messages, which should generally be small enough to authenticate before releasing plaintext.

Streaming protocols, nonce sequencing, key derivation, and key rotation are outside the scope of this document.

6. IANA Considerations

This document requires no IANA actions.

7. References

7.1. Normative References

[BLAKE3] Aumasson, J., Neves, S., O'Connor, J., and Z. Wilcox-O'Hearn, "BLAKE3: One Function, Fast Everywhere", 2020, <https://github.com/BLAKE3-team/BLAKE3-specs>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[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>.

7.2. Informative References

[KERKOUR] Kerkour, S., "ChaCha20-BLAKE3", October 2025, <https://kerkour.com/chacha20-blake3>.

[RFC9771] Bozhko, A., Ed., "Properties of Authenticated Encryption with Associated Data (AEAD) Algorithms", RFC 9771, DOI 10.17487/RFC9771, May 2025, <https://www.rfc-editor.org/rfc/rfc9771>.

Appendix A. Test Vectors

All inputs are printable ASCII for readability: the key is 32 "k" bytes and the nonce is 12 "n" bytes. Outputs are cryptographically derived and given in hexadecimal. The vectors expose the intermediate values of the construction -- the one-time authentication key and the padded MAC input -- so that implementations can be checked step by step without duplicating the primitive-level tests of [RFC8439] or the BLAKE3 specification [BLAKE3].

A.1. Authentication Key Derivation

```
KEY:      "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk" (32 bytes)
NONCE:    "nnnnnnnnnnnn" (12 bytes)

AUTH KEY:
  86d81b22b20ece4ac05606338c49f553e6dee583abcfc7ceb1a111688cd01c99
```

AUTH KEY is the first 32 bytes of ChaCha20 block 0 under KEY and NONCE.

A.2. Authenticated Encryption

```
KEY:       "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk" (32 bytes 6a)
NONCE:     "nnnnnnnnnnnn" (12 bytes 6e)
AAD:       "aad"
PLAINTEXT: "secret"

CIPHERTEXT: fd9a5d20a7a6
```

Intermediate values:

```
AUTH KEY:
  86d81b22b20ece4ac05606338c49f553e6dee583abcfc7ceb1a111688cd01c99

MAC INPUT (144 bytes, shown as 32-byte blocks):
  6161640000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  fd9a5d20a7a6000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0300000000000000060000000000000
```

The first block holds AAD (3 bytes, 616164) followed by 61 zero bytes of pad64. The ciphertext (6 bytes) occupies the next block, padded with 58 zero bytes. The final 16 bytes are len64(AAD) = 3 and len64(ciphertext) = 6, little-endian; this block is not padded.

The MAC input is hashed with BLAKE3 in keyed mode under AUTH KEY. Tags of selected lengths, taken from the start of the extendable output:

```
TAG (16 bytes): 6b294ae99019282afd2976759e030b83
TAG (32 bytes):
  6b294ae99019282afd2976759e030b8343f80b77a6e9b872dfb9783356186f6d
```

With the nonce and a 16-byte tag attached as specified in Section 3, the complete message is:

```
MESSAGE:
  6e6e6e6e6e6e6e6e6e6e6e6efd9a5d20a7a6
  6b294ae99019282afd2976759e030b83
  ^nonce                  ^ciphertext
  ^tag
```

A.3. Empty Message

Empty plaintext and empty AAD, under the same key and nonce.

```
KEY:       "kkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk" (32 bytes)
NONCE:     "nnnnnnnnnnnn" (12 bytes)
AAD:       (empty)
PLAINTEXT: (empty)

MAC INPUT (16 bytes):
  00000000000000000000000000000000

CIPHERTEXT: (empty)
TAG (16 bytes): 89f9644a194beb851a7ccb325864dbc4
```

Both pad64 paddings are empty; the MAC input consists solely of the two zero length fields. With MACLEN = 0 the complete output under these inputs is the empty string: no ciphertext, no tag, and no attached nonce.

Author's Address

Leo Vasanko
Email: git@vasanko.com