Stream: Internet Engineering Task Force (IETF)
RFC: 9986
Category: Experimental
Published: June 2026
ISSN: 2070-1721
Authors:
A. DeKok, *InkBridge Networks*
M. Jethanandani, *Arrcus*
S. Agarwal, *Arrcus, Inc.*
A. Mishra, *Aalyria Technologies*
J. Haas, *HPE*

# RFC 9986
# Meticulous Keyed ISAAC for Bidirectional Forwarding Detection (BFD) Optimized Authentication

## Abstract

This document describes a Bidirectional Forwarding Detection (BFD) Optimized Authentication Mode known as Meticulous Keyed ISAAC Authentication. This mode can be used to authenticate some BFD packets with less CPU time cost than using MD5 or SHA-1 with the trade-off of decreased security. This mechanism cannot be used to signal state changes, but it can be used to maintain a session in the Up state.

## Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9986.

## Copyright Notice

## Table of Contents

# 1. Introduction

BFD (Section 6.7 of [RFC5880]) defines a number of authentication mechanisms, including Simple Password and various other methods based on MD5 and SHA-1 hashes. The benefit of using cryptographic hashes is that they are secure. The downside to cryptographic hashes is that they are expensive and time-consuming on resource-constrained hardware.

When BFD packets are unauthenticated, it is possible for an attacker to forge, modify, and/or replay packets on a link. These attacks have a number of side effects. They can cause parties to believe that a link is down, or they can cause parties to believe that the link is up when it is, in fact, down.

[RFC9985] defines procedures that enable better scaling of authentication for BFD by splitting BFD Authentication work between more computationally intensive authentication used for significant changes, and less computationally intensive authentication for packets validating that the session is in the Up state. See [RFC9985] for general performance and security considerations.

This document provides the definition of BFD Optimized Authentication Modes using the existing MD5 (Section 6.7.3 of [RFC5880]) and SHA-1 (Section 6.7.4 of [RFC5880]) Authentication mechanisms for the more computationally intensive work. It also defines methods for using a mechanism, ISAAC [ISAAC], for the less computationally intensive mechanism.

ISAAC requires only a few CPU operations per generated 32-bit number, can take a large secret key as a seed, and it has an extremely long cycle length. These properties make it ideal for use in BFD.

ISAAC+ [ISAAC-Plus] documents some cryptanalysis of the ISAAC mechanism. This analysis addressed an issue with initial seeding, and the method proposed here incorporates recommendations to address that attack.

## 1.1. Meticulous Keying

[RFC5880] uses the term "meticulous keyed" and "meticulous keying" without defining those terms. That meaning of that term is found by examining the definition of the sequence number from [RFC5880] (Section 4.3):

> The sequence number for this packet. For Keyed MD5 Authentication, this value is incremented occasionally. For Meticulous Keyed MD5 Authentication, this value is incremented for each successive packet transmitted for a session. This provides protection against replay attacks.

In this context, the term "meticulous" means that the sequence number is incremented on every new packet that is sent. The term "keyed" means that the packets are authenticated via the use of a secret key or keys that are known to both sender and receiver. Therefore, the term "meticulous keyed" refers to the BFD Authentication type where each subsequently transmitted packet has a sequence number that is one greater than the immediately previous one and can be authenticated.

## 1.2. Requirements Language

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. Experimental Extensions to RFC 5880

This document describes an experimental extension to BFD [RFC5880]. This experiment is intended to provide additional insights into what happens when the authentication method defined in this document is used.

This document is classified as Experimental and is not part of the IETF Standards Track. Implementations based on this document should not be considered as compliant with BFD [RFC5880] and should not assume interoperability with other implementations that conform to this document.

Some of the state variables in Section 6.8.1 of [RFC5880] are related to the authentication type being used for a particular session. However, the definitions given in [RFC5880] are specific to Keyed MD5 or SHA-1 Authentication, which limit their utility for new authentication types. This document presumes a relaxed definition for the following BFD state variables that does not limit them to MD5 and SHA-1 but instead extends them to the mechanism defined herein:

- bfd.RcvAuthSeq
- bfd.XmitAuthSeq
- bfd.AuthSeqKnown

## 3. Architecture of the Auth Type Method

This document specifies two Optimized BFD [RFC9985] Authentication Modes:

- For the more computationally intensive authentication mechanisms, the existing MD5 (Section 6.7.3 of [RFC5880]) and SHA-1 (Section 6.7.4 of [RFC5880]) Authentication mechanisms are leveraged with small PDU changes necessary to carry the Optimization Mode encoding. These changes are documented in Sections 4.2 and 4.3, respectively.
- For the less computationally intensive authentication mode, this document defines the Meticulous Keyed ISAAC Authentication mechanism. The PDU format for this mode is defined in Section 4.1. The procedures for using this format are covered later in this document.

ISAAC is used as a way to generate an infinite stream of pseudorandom numbers, referred to here as "Auth Keys". With Meticulous Keyed ISAAC Authentication, these Auth Keys are used as a signal that the sending party is authentic. That is, only the sending party can generate the correct Auth Keys. Therefore, if the receiving party sees a correct Auth Key in a BFD Control Packet in the Up state, then only the sending party could have generated it.

Note that BFD Control Packets with the less computationally intensive ISAAC Authentication Format type are NOT signed or authenticated. Therefore, this format **MUST NOT** be used to signal BFD state changes.

### 3.1. Rationale for ISAAC and Operational Overview

There are many cryptographically secure pseudorandom number generators (CSPRNGs) available. This section explains why ISAAC was chosen.

The goal for this less computationally intensive authentication was to provide a signal that the session was in the Up state in the form of a 32-bit number, which is difficult for an attacker to guess. The number should be generated from a CSPRNG that produces results based on a Seed composed of both public and private data. Since BFD can have packet loss, the generator should also be "seekable" in that the BFD state machine should be able to query the generator (within a small window) for new numbers.

This last property rules out most CSPRNGs, as they are not seekable by design. That is, most CSPRNGs maintain minimal state and are designed to produce a long sequence of pseudorandom numbers from a few simple calculations. In general, every call to the CSPRNG function modifies the internal state in an irreversible fashion, and then produces a new random number as the result.

It could be possible to use such a generator and manually save many results in a buffer. This buffer could then enable "seeking" within a short window. In contrast, ISAAC produces large sets of numbers by design, making it an integrated solution.

Further, most CSPRNGs are designed to have small seeds. This limitation means that any secret key defined by an administrator is not directly usable as a Seed for the generator. Instead, any secret key (including any per-session data) would have to be hashed before being used to see the generator. For these reasons, ISAAC was chosen. It can accept keys up to 8192 octets in length, which is more than sufficient for BFD.

ISAAC has been subject to cryptanalysis, most notably ISAAC+ [ISAAC-Plus]. There are no known vulnerabilities.

Two instances of ISAAC are created: one for transmission and one for reception. An instance is required for each direction since the inputs for seeding ISAAC require the locally randomly generated Seed value and the current BFD Your Discriminator value for an Up session. These values are distinct on each side of the BFD session.

The process for using ISAAC with BFD for each direction is then as follows:

- The administrator provides a secret key that is used to authenticate each party in the BFD sessions.
- When the session transitions into the Up state, the secret key is combined with per-session data to seed ISAAC.
- The ISAAC process produces a "page" of 256 32-bit random numbers.
- The BFD state machine also records a sequence number that is associated with the first entry of that page. The combination of 256 entries and the sequence number allows the BFD state

machine to "seek" within a 256-packet window with zero cost through simple addition or subtraction of sequence numbers.

- If there is a lost packet, the BFD state machine simply seeks to the entry that is associated with the received packet and checks if the received packet contains the expected number.
- BFD supports packet rates of hundreds of packets per second. Even at those rates, 256 entries per ISAAC page provides for about a second of BFD operation before the next page has to be calculated.
- As the next page calculation is complex, and there is a long period of time available before the next page is needed, this calculation can be done in the background.
- If the next page calculation is started immediately after the current page is fully used, there should be sufficient time to calculate the next page as a background task, no matter what the packet rate.

In summary, the ISAAC Seed depends on both a secret key and per-session data, so it is difficult for an attacker to guess or attack via an offline dictionary attack. The generated numbers are saved in an array, where the BFD fast path can consume them at essentially zero cost.

The only downside to this method is that it does not provide for per-packet integrity checks. This limitation is addressed by mandating that Meticulous Keyed ISAAC Authentication is only used to signal that the session remains in the Up state. The ISAAC numbers then signal that the originator of the packet is authentic, and the BFD state machine verifies that the rest of the packet is well formed and matches the expected state.

The result is an authentication method that satisfies the needs of the BFD state machine and is secure.

# 4. Meticulous Keyed ISAAC Authentication Types

## 4.1. Meticulous Keyed ISAAC Authentication, ISAAC Format

If the Authentication Present (A) bit is set in the header, and the Authentication Type field contains either Optimized MD5 Meticulous Keyed ISAAC Authentication (7) or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8), and the Optimized Authentication Mode field contains 2 (Section 7 of [RFC9985]), the Authentication Section has the following format:

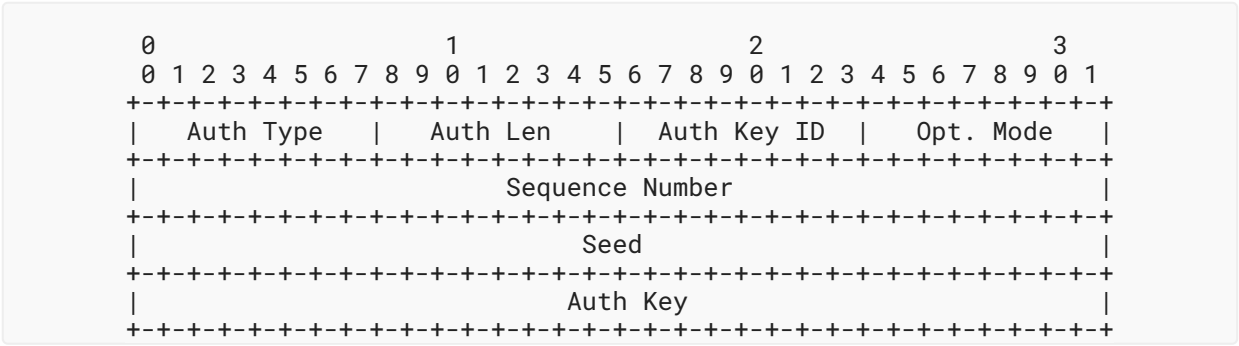


*Figure 1: Meticulous Keyed ISAAC Authentication Format*

Auth Type:
: The current Auth Type. It **MUST** provide for meticulous keying. That is, an authentication type where each packet is authenticated and where the Sequence Number field is incremented by one (1) for every packet that is sent.

Auth Len:
: The length of the Authentication Section in bytes. For Meticulous Keyed ISAAC Authentication Format, the length is 16 bytes.

Auth Key ID:
: The authentication key ID in use for this packet. This allows multiple secret keys to be active simultaneously.

Opt Mode:
: The Optimized Authentication Mode is defined in Section 7 of [RFC9985]. When the Auth Type is either Optimized MD5 Meticulous Keyed ISAAC Authentication (7) or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8), and the format is Meticulous Keyed ISAAC Authentication Format, the Optimized Authentication Mode field will be set to 2.

Sequence Number:
: The sequence number for this packet. For Meticulous Keyed ISAAC Authentication, this value is incremented once for each successive packet transmitted for a session. This provides protection against replay attacks.

Seed:
: A 32-bit (4 octet) Seed that is used in conjunction with the shared key in order to configure and initialize the ISAAC Pseudorandom Number Generator (PRNG). It is used to identify and secure different "streams" of random numbers that are generated by ISAAC.

Auth Key:
: This field carries the 32-bit (4 octet) ISAAC output that is associated with the sequence number. The ISAAC PRNG **MUST** be configured and initialized as given in Section 10.

  Note that the Auth Key here does not include any summary or hash of the BFD Control Packet. The packet itself is completely unauthenticated.

When the receiving party receives a BFD packet with an expected sequence number and the correct corresponding ISAAC output in the Auth Key field, it knows that only the authentic sending party could have sent that message. The sending party is therefore Up, as it is the only one who could have sent the message.

## 4.2. Meticulous Keyed ISAAC Authentication, MD5 Format

If the Authentication Present (A) bit is set in the header, and the Authentication Type field contains Optimized MD5 Meticulous Keyed ISAAC Authentication (7), and the Optimized Authentication Mode field contains 1 (Section 7 of [RFC9985]), the Authentication Section has the following format:

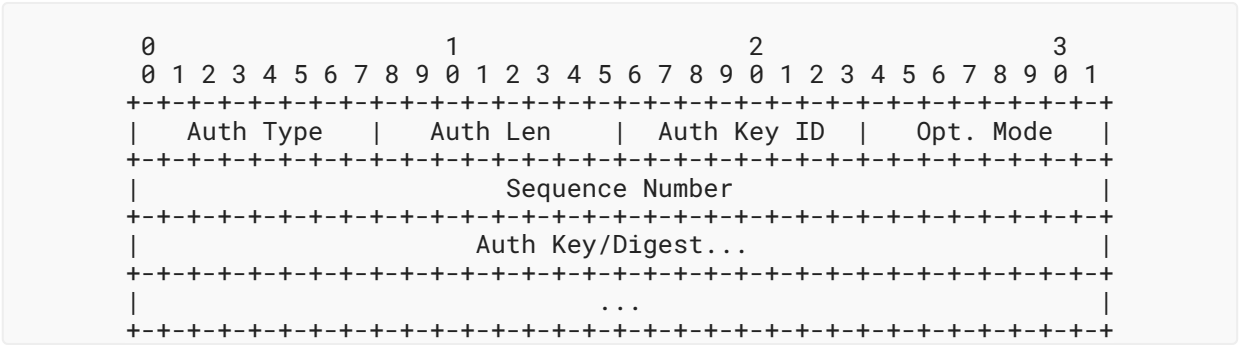


*Figure 2: Meticulous Keyed ISAAC Authentication, MD5 Format*

Auth Type:
: The current Auth Type. It **MUST** provide for meticulous keying. That is, an authentication type where each packet is authenticated and where the Sequence Number field is incremented by one (1) for every packet that is sent.

Auth Len:
: The length of the Authentication Section in bytes. For Meticulous Keyed ISAAC Authentication, MD5 Format, the length is 24 bytes.

Auth Key ID:
: The authentication key ID in use for this packet. This allows multiple secret keys to be active simultaneously.

Opt Mode:
: The Optimized Authentication Mode is defined in Section 7 of [RFC9985]. When the Auth Type is Optimized MD5 Meticulous Keyed ISAAC Authentication (7), and the format is MD5 Authentication Format, the Optimized Authentication Mode field will be set to 1.

Sequence Number:
: The sequence number for this packet. For Meticulous Keyed ISAAC Authentication, this value is incremented once for each successive packet transmitted for a session. This provides protection against replay attacks.

Auth Key/Digest:
: This field carries the 16-byte MD5 digest for the packet. The procedure for calculating this field is documented in Section 6.7.3 of [RFC5880].

## 4.3. Meticulous Keyed ISAAC Authentication, SHA-1 Format

If the Authentication Present (A) bit is set in the header, and the Authentication Type field contains Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8), and the Optimized Authentication Mode field contains 1 (Section 7 of [RFC9985]), the Authentication Section has the following format:

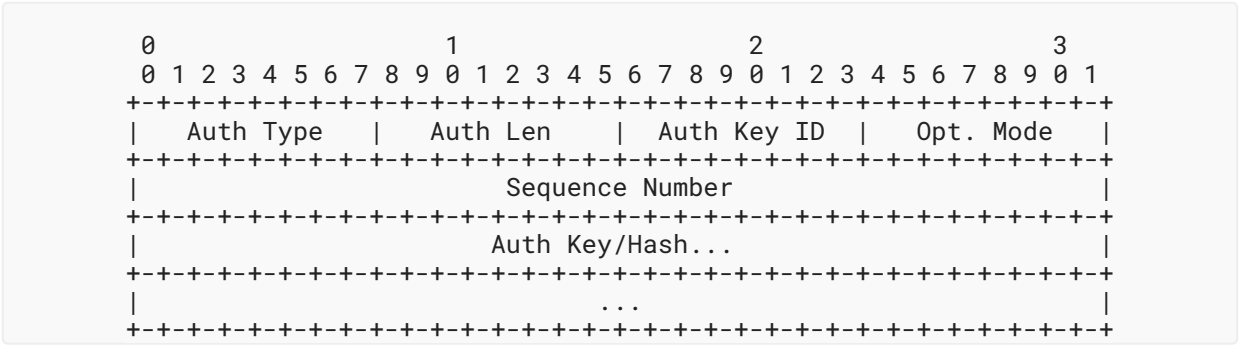


*Figure 3: Meticulous Keyed ISAAC Authentication, SHA-1 Format*

Auth Type:
: The current Auth Type. It **MUST** provide for meticulous keying. That is, an authentication type where each packet is authenticated and where the Sequence Number field is incremented by one (1) for every packet that is sent.

Auth Len:
: The length of the Authentication Section in bytes. For Meticulous Keyed ISAAC Authentication, SHA-1 Format, the length is 28 bytes.

Auth Key ID:
: The authentication key ID in use for this packet. This allows multiple secret keys to be active simultaneously.

Opt Mode:
: The Optimized Authentication Mode is defined in Section 7 of [RFC9985]. When the Auth Type is Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8), and the format is SHA-1 Authentication Format, the Optimized Authentication Mode field will be set to 1.

Sequence Number:
: The sequence number for this packet. For Meticulous Keyed ISAAC Authentication, this value is incremented once for each successive packet transmitted for a session. This provides protection against replay attacks.

Auth Key/Digest:
: This field carries the 16-byte SHA-1 hash for the packet. The procedure for calculating this field is documented in Section 6.7.4 of [RFC5880].

# 5. New State Variables for Meticulous Keyed ISAAC Authentications

This document defines new state variables for use with Meticulous Keyed ISAAC Authentication.

bfd.MetKeyIsaacRcvKeyKnown:
: A boolean value that indicates whether or not the system knows the receive key for the Meticulous Keyed ISAAC Authentication. The initial value is false. This value is changed to "true" when a party verifies that the other party has started to use the Meticulous Keyed ISAAC Authentication with an authenticated Auth Key.

bfd.MetKeyIsaacRcvAuthBase:
: A 32-bit unsigned integer containing a copy of the bfd.RcvAuthSeq number that is associated with the current ISAAC "page" for authenticating received packets.

bfd.MetKeyIsaacRcvAuthIndex:
: An 8-bit number used to index within a particular "page" of pseudorandom numbers.

bfd.MetKeyIsaacRcvAuthSeed:
: A 32-bit unsigned integer containing a copy of the Seed associated with received packets.

bfd.MetKeyIsaacRcvAuthData:
: A data structure that contains the ISAAC data for the received Auth Type method. The format and contents of this structure are implementation specific and hold the internal state of the ISAAC CSPRNG.

bfd.MetKeyIsaacXmitKeyKnown:
: A boolean value that indicates whether or not the system knows the xmit key for Meticulous Keyed ISAAC Authentication. The initial value is false. This value is changed to "true" when a party starts to transmit using Meticulous Keyed ISAAC Authentication.

bfd.MetKeyIsaacXmitAuthBase:
: A 32-bit unsigned integer containing a copy of the bfd.XmitAuthSeq number that is associated with the current ISAAC "page" for authenticating sent packets.

bfd.MetKeyIsaacXmitAuthIndex:
: An 8-bit number used to index within a particular "page" of pseudorandom numbers.

bfd.MetKeyIsaacXmitAuthSeed:
: A 32-bit unsigned integer containing a copy of the Seed associated with sent packets.

bfd.MetKeyIsaacXmitAuthData:
: A data structure that contains the ISAAC data for the sending Auth Type method. The format and contents of this structure are implementation specific and hold the internal state of the ISAAC CSPRNG.

# 6. Procedures for BFD Authentication Using Meticulous Keyed ISAAC, MD5, or SHA-1 Formats

The transmit and receive procedures utilize the additional procedures documented in Section 7.1 of [RFC9985].

The authentication procedure for Meticulous Keyed ISAAC, MD5 Format is described in Section 6.7.3 of [RFC5880] for the Meticulous Keyed MD5 Authentication Mode.

The authentication procedure for Meticulous Keyed ISAAC, SHA-1 Format is described in Section 6.7.4 of [RFC5880] for the Meticulous Keyed SHA-1 Authentication Mode.

# 7. Procedures for BFD Authentication Using Meticulous Keyed ISAAC, ISAAC Format

In this mode of optimized authentication, one or more secret keys (with corresponding key IDs) are configured in each system. One of the keys is used to seed the ISAAC PRNG. The output of ISAAC is used to signal that the sender is authentic. To help avoid replay attacks, a sequence number is also carried in each packet. For Meticulous Keyed ISAAC Authentication, the sequence number **MUST** be incremented by one on every packet.

The receiving system accepts the packet if the key ID matches one of the configured Keys, and the Auth Key derived from the selected Key, Seed, and sequence number matches the Auth Key carried in the packet, and the sequence number is strictly greater than the last sequence number received (modulo wrap at $2^{32}$). If any of these criteria do not match, the packet fails validation and is discarded.

## 7.1. Transmission using Meticulous Keyed ISAAC Authentication, ISAAC Format

The Auth Type field **MUST** be set to one of two values; Optimized MD5 Meticulous Keyed ISAAC Authentication (7) or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8).

The Auth Len field **MUST** be set to 16.

The Auth Key ID field **MUST** be set to the ID of the current authentication key. The Sequence Number field **MUST** be set to bfd.XmitAuthSeq.

The Seed field **MUST** be set to the value of the current Seed used for this session.

The Auth Key field **MUST** be set to the output of ISAAC, which depends on the secret Key, the current Seed, and the sequence number.

The Optimized Authentication Mode field **MUST** be 2, the "less computationally intensive authentication type". See Section 7 of [RFC9985].

For Meticulous Keyed ISAAC Authentication, bfd.XmitAuthSeq **MUST** be incremented by one on each packet in a circular fashion (when treated as an unsigned 32-bit value). The bfd.XmitAuthSeq **MUST NOT** be incremented by more than one per packet.

## 7.2. Receipt using Meticulous Keyed ISAAC Authentication, ISAAC Format

If the received BFD Control Packet does not contain an Authentication Section, or the Auth Type is not correct (either Optimized MD5 Meticulous Keyed ISAAC Authentication (7) or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (8)), then the received packet **MUST** be discarded.

If the Auth Key ID field does not match the ID of a configured authentication key, the received packet **MUST** be discarded.

The Optimized Authentication Mode field **MUST** be 2, the "less computationally intensive authentication type". See Section 7 of [RFC9985].

If the Auth Len field is not equal to 16, the packet **MUST** be discarded.

If bfd.AuthSeqKnown is 1, examine the Sequence Number field. For Meticulous Keyed ISAAC, if the sequence number lies outside of the range of bfd.RcvAuthSeq+1 to bfd.RcvAuthSeq+(3*Detect Mult) inclusive (when treated as an unsigned 32-bit circular number space) the received packet **MUST** be discarded.

If bfd.MetKeyIsaacRcvKeyKnown is "true" and the Seed field does not match the current Seed value, bfd.MetKeyIsaacRcvAuthSeed, the packet **MUST** be discarded.

Calculate the current expected output of ISAAC, which depends on the secret Key, the current Seed, and the sequence number. If the value does not match the Auth Key field, then the packet **MUST** be discarded.

If bfd.MetKeyIsaacRcvKeyKnown is false, the ISAAC related variables are initialized as per Section 10.2 using the contents of the packet.

Note that in some cases, calculating the expected output of ISAAC will result in the creation of a new "page" of 256 numbers. This process will be irreversible and will destroy the current "page". As a result, if the generation of a new output will create a new "page", the receiving party **MUST** save a copy of the entire ISAAC state before proceeding with this calculation. If the outputs match, then the saved copy can be discarded and the new ISAAC state is used. If the outputs do not match, then the saved copy **MUST** be restored and the modified copy discarded or cached for later use.

## 8. Secret Key

The security of the Meticulous Keyed ISAAC Auth Type depends on the secret key. The secret key is mixed with a per-session Seed as discussed below. The result is used to initialize a stream of pseudorandom numbers using the ISAAC random number generator.

Using the same or distinct secret keys for each Optimized Authentication Mode has security and operational impacts. See Section 15.1.2 for discussion on these points.

It is **RECOMMENDED** that implementations permit distinct secret keys to be provisioned for a given Auth Key ID for each Optimized Authentication Mode. The operator's choice to use such distinct secret keys instead of a single secret key is out of scope for this document.

A particular secret key set is identified via the Auth Key ID field. This Auth Key ID is either placed in the packet by the sender or verified by the receiver. Meticulous Keyed ISAAC Authentication permits systems to have multiple Secret Keys configured, but we do not discuss how those keys are managed or used. A session **MUST NOT**, however, change the Auth Key ID for Meticulous Keyed ISAAC Authentication during a session. There is no defined way to resync or reinitialize an ongoing session with a different Auth Key ID and correspondingly different secret key.

For interoperability, the management interface by which the key is configured **MUST** accept ASCII strings and **SHOULD** also allow for the configuration of any arbitrary binary string in hexadecimal form. Other configuration methods **MAY** be supported.

The secret key **MUST** be at least eight (8) octets in length and **SHOULD NOT** be more than 128 octets in length.

## 9. Transition to Using ISAAC

A BFD session that uses Optimized MD5 Meticulous Keyed ISAAC Authentication or Optimized SHA-1 Meticulous Keyed ISAAC Authentication **MUST** begin a session with Auth Type set to the relevant authentication type and the Optimized Authentication Mode field set to 1.

When a BFD session using more computationally intensive authentication transitions to the Up state, the first Up packet **MUST** contain an Optimized Authentication Mode field with value 1. Since state transitions require full packet integrity checks, an Optimized Authentication Mode field with value 2 is not permitted for state changes. Each party **MUST** continue to use the more computationally intensive authentication mode until the other side has confirmed the switch to the Up state with a packet that also uses more computationally intensive authentication.

Once the BFD session has transitioned to the Up state, the sender **MAY** send the subsequent packets for the Up state with the Optimized Authentication Mode field containing value 2 using the ISAAC Format.

When a system first receives a packet containing the Optimized Authentication Mode field with value 2, it initializes the ISAAC PRNG state using the Seed from that packet. A system originating a packet using Meticulous Keyed ISAAC Authentication will generate a Seed and place it into the packet, which is then sent. Further discussion of initialization takes place in Sections 10.1 and 10.2.

The first packet after the transition to the Up state is the only time when the ISAAC random number generator for transmission is initialized. In contrast, a temporary transition away from using Meticulous Keyed ISAAC Authentication, ISAAC Format (Section 12) and back does not cause ISAAC to be rekeyed.

There is no negotiation as to when authentication switches from the original type to using Meticulous Keyed ISAAC Authentication using the ISAAC Format. The sender simply begins sending packets with a relevant Auth-Type and with the Optimized Authentication Mode field set to 1. When the sender switches to using Meticulous Keyed ISAAC Authentication, ISAAC Format, it sets the Optimized Authentication Mode field to 2 and starts performing the ISAAC calculations as described here.

Similarly, a receiving system switches to using this method when it sees that it has received a packet contains the Optimized Authentication Mode field set to 2 when bfd.MetKeyIsaacRcvKeyKnown variable is false. The receiving system then initializes its variables and authenticates the received packet by comparing the Auth Key in the packet with the key it generated itself.

The operation of those state variables **MUST** now satisfy the requirements of the new Optimized Authentication Mode. That is, when changing Optimized Authentication Mode in a session, the current value of the bfd.RcvAuthSeq and bfd.XmitAuthSeq variables is used as the initial value(s) for the new mode.

## 10. Seeding ISAAC

The Seed field is used to identify and secure different "streams" of random numbers that are generated by ISAAC. Each session uses a different Seed, which is used along with the Your Discriminator field (Section 4.1 of [RFC5880]) and the secret key to initialize ISAAC.

The value of the Seed field **MUST** be derived from a CSPRNG source. Exactly how this can be done is outside of the scope of this document.

A new Seed value **MUST** be created every time a BFD session transitions into the Up state. In order to prevent continuous rekeying, once the session is in the Up state, the Seed for a session **MUST NOT** be changed until another state transition occurs.

The ISAAC PRNG is initialized by setting all internal variables and data structures to zero (0). The PRNG is then seeded by using the following structure:

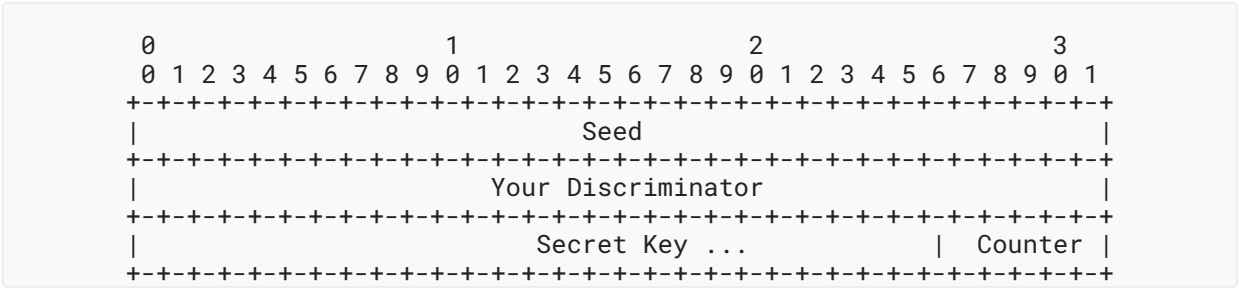


*Figure 4: ISAAC Initialization Structure*

where the Your Discriminator field is taken from the BFD packet defined in [RFC5880], Section 4.1. This field is taken from the respective values used by a sending system. For receiving systems, the field are taken from the received packet. As the size of the buffer used to seed is limited, the length of the secret key **MUST** be no more than 1015 octets. The Counter field is used to ensure the initial seeding of ISAAC avoids the seeding issues discussed in ISAAC+ [ISAAC-Plus].

Whatever the API or other interface used to input the Secret Key, any implementation-specific internal representations of the secret key **MUST NOT** be used when encoding the secret key into the above data structure. That is, there is no length field that indicates how long the secret key is and there is no trailing zero or NUL byte that indicates the end of the secret key. Implementors are reminded that internal representations of data should not affect protocol operation.

The buffer used to initialize ISAAC filled it with repeated copies of the above structure. For each complete copy of the structure, the Counter field is incremented starting from zero (0). The final portion of the initialization buffer holds a partial copy of the structure, which is however much can be accommodated in the remaining portion of the buffer.

Once the ISAAC "page" is initialized, the data is processed through the "randinit()" function of ISAAC [ISAAC]. Pseudo-random numbers are then produced 32 bits at a time by calling the "isaac()" function.

For the sender, this calculation can be done outside of the BFD "fast path" as soon as the Your Discriminator value is known. For the receiver, this calculation can only be done when the Seed is received from the sender, and therefore the initial seeding needs to be done in the BFD "fast path".

The following table gives Seed and Your Discriminator as 32-bit hexadecimal values and the secret key as an eleven-character string. The subsequent table shows the first eight sequence numbers and corresponding Auth Key values that were generated using the above initial values.

| Field | Value(s) |
|---|---|
| Seed | 0x0bfd5eed |
| Y-Disc | 0x4002d15c |
| Secret Key | RFC5880June |

| Field | Value(s) |
|---|---|
| Counter | 0...50 |

*Table 1: Test Inputs for Seeding ISAAC*

| Sequence | Auth Key |
|---|---|
| 0 | 9af65d83 |
| 1 | 44355d56 |
| 2 | 9334074e |
| 3 | b643ef59 |
| 4 | 74d659f1 |
| 5 | 8966dc56 |
| 6 | a1f6f9bc |
| 7 | 21895a46 |

*Table 2: Expected Outputs*

This construct provides 64 bits of entropy, of which 32 bits are controlled by each party in a BFD session. For security, each implementation **SHOULD** randomize their discriminator fields at the start of a session, as discussed in [RFC5880], Section 9.

Note that this construct only uses the Your Discriminator field once to seed ISAAC. It therefore allows the My Discriminator field to change as permitted by BFD [RFC5880] (Section 6.3).

While the Your Discriminator field may change, there is no way to signal or negotiate Seed changes. The Seed is set once by each party after the session transitions into the Up state, and then remains unchanged for the duration of the session. The receiving party **MUST** remember the current Seed value. The Seed value **MUST NOT** change unless sending party has signaled a BFD state change with a packet that is authenticated using a more computationally intensive authentication method. When a system receives a BFD packet containing Meticulous Keyed ISAAC Authentication, it **MUST** check that the received Seed contains the expected value, and if not, it **MUST** discard the packet as inauthentic.

## 10.1. Sender Variable Initialization

A system that sends packets initializes ISAAC as described above. The ISAAC related variables are initialized as follows:

bfd.MetKeyIsaacXmitKeyKnown:
: This variable transitions from false to true when the sender decides to start using ISAAC. The sender also initializes the other variables at the same time.

bfd.MetKeyIsaacXmitAuthBase:
: The sender copies the bfd.XmitAuthSeq number from the current packet to be sent into this variable.

bfd.MetKeyIsaacXmitAuthIndex:
: The sender sets this variable to zero.

bfd.MetKeyIsaacXmitAuthSeed:
: The sender copies the current Seed value into this variable. This variable is then copied into the "Seed" field of each Auth Type packet.

bfd.MetKeyIsaacXmitAuthData:
: The ISAAC state for sending is encapsulated in this variable.

## 10.2. Receiver Variable Initialization

When a system receives packets with Meticulous Keyed ISAAC Authentication and is able to authenticate such a packet the first time, the ISAAC related variables are initialized as follows:

bfd.MetKeyIsaacRcvKeyKnown:
: This variable transitions from false to true when the receiver sees that the sender has started using Meticulous Keyed ISAAC Authentication. The receiver also initializes the other variables at the same time.

bfd.MetKeyIsaacRcvAuthBase:
: The bfd.RcvAuthSeq number from the current packet is copied into this variable.

bfd.MetKeyIsaacRcvAuthIndex:
: The receiver sets this value to zero.

bfd.MetKeyIsaacRcvAuthSeed:
: The receiver copies the Seed value from the received packet into this variable. Note that this copy only occurs when the bfd.MetKeyIsaacXmitKeyKnown variable transitions from false to true.

bfd.MetKeyIsaacRcvAuthData:
: The ISAAC state for receiving is encapsulated in this variable.

As there may be packet loss, the receiver has to take special care to initialize the bfd.MetKeyIsaacRcvAuthBase variable. If there has been no packet loss, the bfd.MetKeyIsaacRcvAuthBase is taken directly from the bfd.RcvAuthSeq variable, and the bfd.MetKeyIsaacRcvAuthIndex is set to zero.

However, if the packet's sequence number differs from the expected value, then the difference "N" indicates how many packets were lost. The receiver can then use this difference to index into the ISAAC page to find the corresponding Auth Key. If the key in the ISAAC page does not match the corresponding Auth Key in the packets, the packet fails validation and is discarded.

If a key found by indexing into this ISAAC page does match the Auth Key in the packet, then the bfd.MetKeyIsaacRcvAuthIndex field is initialized to this value. The bfd.MetKeyIsaacRcvAuthBase field is then initialized to contain the value of bfd.RcvAuthSeq, minus the value of bfd.MetKeyIsaacRcvAuthIndex. This process allows the pseudorandom stream to be resynchronized in the event of lost packets.

That is, the value for bfd.MetKeyIsaacRcvAuthBase is the sequence number for first Auth Key used in this session. This value may be from a lost packet, but can never the less be calculated by the receiver from a later packet.

## 11. Operation

Once the variables have been initialized, ISAAC will be able to produce 256 random numbers to use as Auth Keys at near-zero cost. The AuthIndex field is incremented by one for every new Auth Key generated. Each new value of the Sequence Number field (sent or received) is then calculated by adding the relevant AuthBase and AuthIndex fields.

When all 256 numbers are consumed, the AuthIndex field will wrap to zero. The ISAAC mixing function is then run, which then results in another set of 256 random numbers. The AuthBase variable is then incremented by 256 to indicate that 256 Auth Keys have been consumed. This process then continues until a BFD state change.

ISAAC can be thought of here as producing an infinite stream of numbers, based on a secret key, where the numbers are produced in "pages" of 256 32-bit values. This property of ISAAC allows for essentially zero-cost "seeking" within a page. The expensive operation of mixing is performed only once per 256 packets, which means that most BFD packet exchanges can be fast and efficient.

The receiving party can then look at the sequence number to determine which particular PRNG value is being used in the packet. By subtracting the bfd.MetKeyIsaacAuthBase from the sequence number (with possible wrapping), an expected Index can be derived and a corresponding Auth Key can be found. This process thus permits the two parties to synchronize if/when a packet or packets are lost.

Incrementing the sequence number for every packet also prevents the reuse of any individual pseudorandom number that was derived from ISAAC.

The sequence number can increment without bounds, though it can wrap once it reaches the limit of the 32-bit counter field. ISAAC has a cycle length of $2^{8287}$, so there is no issue with using more than $2^{32}$ values from it.

The result of the above operation is an infinite series of numbers that are unguessable and that can be used to authenticate the sending party.

Each system sending BFD packets chooses its own seed, generates its own sequence of pseudorandom numbers using ISAAC, and places those values into the Auth Key field. Each system receiving BFD packets runs a separate pseudorandom number generator and verifies that the received packets contain the expected Auth Key.

## 11.1. Page Flipping

Once all 256 Auth Keys from the current page have been used, the next page is calculated by calling the isaac() function. This function modifies the current page to create the next page and is inherently destructive. In order to prevent issues, care should be taken to perform this process correctly.

It is **RECOMMENDED** that implementations keep both a current page and a next page associated with the ISAAC state. The next page can be calculated by making a copy of the current page, and then calling the isaac() function.

The system needs to maintain the current page at all times when Meticulous Keyed ISAAC Authentication is used. The next page does not need to be maintained at all times, and can be calculated on demand. However, in order to avoid impacting the fast path, the next page should be calculated in the background in an asynchronous manner.

This process has a number of benefits. First, at 60 packets per second, the system has approximately four (4) seconds of time to calculate the next page. If the calculation is done quickly, the next page is available to the fast path before it is needed.

Second, having the next page available early means that an attacker cannot spoof BFD packets and force the received to spend significant resources calculating a next page on the BFD fast path. Instead, the receiver can simply check the contents of the next page at near-zero cost and discard the spoofed packet.

When the receiver determines that it needs to move to the next page, it can simply swap the current and next pages (updating the BFD variables as appropriate) and then begin an asynchronous calculation of the next page. Such asynchronous calculations are preferable to calculating the next page in the BFD fast path.

This document does not make provisions for dealing with the case of losing more than 512 packets. Implementors **MUST** limit the value of Detect Multi to a small enough number in order to keep the number of lost packets within an acceptable limit.

## 11.2. Multiple Keys

In a keyed algorithm, the key is shared between the two systems. Distribution of this key to all the systems at the same time can be quite a cumbersome task. BFD sessions running a fast rate may require these keys to be refreshed often, which poses a further challenge.

While the Auth Key ID field provides for the provisioning of multiple keys simultaneously, there is no way within the BFD protocol for each party to signal which set of Key IDs are supported. Any such signaling or negotiation needs to be done "out of band" for BFD and usually via manual administrator configuration.

The seeding mechanism for ISAAC, covered in Section 10, is carried out only once for a BFD session. In order to rotate keys, it is **REQUIRED** to administratively disable the BFD session as part of changing the keys. This permits the new session to be seeded as part of bringing up the new session.

## 12. Transition Away from Using ISAAC

There are two ways to transition away from using ISAAC. One way is via state changes: either the link goes down due to a fault or one party signals a state change via a packet signed with a more computationally intensive authentication. The second situation is where one party wishes to temporarily signal via a more computationally intensive method that it is still Up by setting the Optimized Authentication Mode field from value 2 to value 1.

The more computationally intensive authentication type provides for full packet integrity checks, which serves as a stronger indication that the session is Up and that both parties are fully synchronized. This switch can be done at any time during a session.

It is **RECOMMENDED** that implementations periodically switch to the more computationally intensive authentication type for packets that maintain the session in the Up state. The interval between these switches **SHOULD** be long enough that the system still gains significant benefit from using Meticulous Keyed ISAAC Authentication. See [RFC9985] for the appropriate procedure on switching Optimized Authentication Mode.

When switching to the more computationally intensive authentication mode after ISAAC has been seeded, the Authentication Section's Sequence Number field will continue meticulously increasing. In order to permit transition back to ISAAC as the less computationally intensive authentication mechanism, it is necessary for ISAAC to continue to generate pages appropriate for validating the received sequence number.

[RFC9985] describes the procedures that require the switch to the more computationally intensive authentication mode -- particularly BFD Poll Sequences.

## 13. The YANG Module

This YANG module adds two identities defined in this document to the YANG key chain model described in [RFC8177]. One of them uses the Meticulous Keyed MD5 as the more computationally intensive authentication and Meticulous Keyed ISAAC, ISAAC Format as the less computationally intensive authentication. The other uses the Meticulous Keyed SHA-1 as the more computationally intensive authentication and Meticulous Keyed ISAAC, ISAAC Format as the less computationally intensive authentication.

```
<CODE BEGINS> file "ietf-bfd-met-keyed-isaac@2026-06-19.yang"

module ietf-bfd-met-keyed-isaac {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-met-keyed-isaac";
  prefix bfd-mki;

  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains.";
  }

  organization
    "IETF Bidirectional Forwarding Detection (BFD) Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/bfd>
     WG List:  <rtg-bfd@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani@gmail.com)
              Ashesh Mishra (ashesh@aalyria.com)
              Jeffrey Haas (jhaas@juniper.net)
              Alan Dekok (alan.dekok@inkbridge.io)
              Sonal Agarwal (sonal@arrcus.com).";

  description
    "This YANG module provides identities derived from the
     ietf-key-chain model for the experimental BFD Meticulous Keyed
     ISAAC Authentication Mechanism.

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

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Revised BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 9986
     (https://www.rfc-editor.org/info/rfc9986); see the RFC itself
     for full legal notices.";

  revision 2026-06-19 {
    description
      "Initial Version.";
    reference
      "RFC 9986: Meticulous Keyed ISAAC for Bidirectional
       Forwarding Detection (BFD) Optimized Authentication.";
  }

  identity optimized-md5-meticulous-keyed-isaac {
    base key-chain:crypto-algorithm;
    description
      "BFD Optimized Authentication using Meticulous Keyed MD5 as the
```

```
        strong authentication and Meticulous Keyed ISAAC, ISAAC Format
        as the less computationally intensive authentication.";
    reference
      "RFC 9986: Meticulous Keyed ISAAC for Bidirectional
       Forwarding Detection (BFD) Optimized Authentication.";
  }

  identity optimized-sha1-meticulous-keyed-isaac {
    base key-chain:crypto-algorithm;
    description
      "BFD Optimized Authentication using Meticulous Keyed SHA-1 as
       the strong authentication and Meticulous Keyed ISAAC, ISAAC
       Format as the less computationally intensive authentication.";
    reference
      "RFC 9986: Meticulous Keyed ISAAC for Bidirectional
       Forwarding Detection (BFD) Optimized Authentication.";
  }
}
<CODE ENDS>
```

# 14. IANA Considerations

IANA has assigned two BFD Auth Types, one URI, and one YANG module as described in this section.

## 14.1. BFD Auth Types

IANA has added the following two Auth Types to the "BFD Authentication Types" registry and to the accompanying YANG and MIB modules:

- 7: Optimized MD5 Meticulous Keyed ISAAC Authentication
- 8: Optimized SHA-1 Meticulous Keyed ISAAC Authentication

## 14.2. IETF XML Registry

IANA has registered the following URI in the "ns" registry within the "IETF XML Registry" group [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-bfd-met-keyed-isaac
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

## 14.3. The YANG Module Names Registry

IANA has registered the following YANG module in the "YANG Module Names" registry [RFC6020] within the "YANG Parameters" registry group:

Name: ietf-bfd-met-keyed-isaac
Maintained by IANA? N

Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-met-keyed-isaac
Prefix: bfd-mki
Reference: RFC 9986

# 15. Security Considerations

## 15.1. Protocol Security Considerations

All security considerations of [RFC5880] and [RFC9985] apply to this document.

The distribution of secret keys is typically accomplished using provisioning. Secure distribution of these keys for any particular provisioning mechanism is out of scope for this document.

Keys generated and distributed out of band for the purposes described in this specification are generally limited in the security they can provide. It is essential that these keys are selected well and protected when stored.

The security of this proposal depends on the security of the ISAAC algorithm, which has had minimal analysis. While it is believed that the algorithm is secure enough for this use case, no proof is offered. ISAAC was chosen for the reasons discussed in Section 3.1, as no other option was found to be suitable.

The choice of ISAAC was driven in part by the limited functionality of systems that implement this specification. Many of these systems do not have hardware assistance for cryptographic operations, meaning that any CSPRNG based on a block cipher would be unsuitably slow. Where hardware assistance for block ciphers is available, ISAAC offers no advantages over those methods.

As CPUs get faster and hardware acceleration becomes more prevalent, more secure methods become better options. Alternative solutions could be AES with hardware acceleration in Output Feedback Mode (OFB) (see FIPS 197 and SP 800-38A), Chacha in software [RFC8439], or other well-understood techniques.

For these reasons and many others, the ISAAC CSPRNG is, at best, tolerable for use in this specification and is completely unsuitable for use in any other IETF protocol.

The security of this proposal depends strongly on the length of the secret key and on its entropy. It is **RECOMMENDED** that the key be 16 octets in length or more.

The dependency on the secret key for security is mitigated through the use of two 32-bit numbers: the Your Discriminator field from the BFD protocol and the ISAAC Seed. Both numbers are procedurally required to be random. These numbers serve as a nonce that inhibits attackers from performing an offline brute-force dictionary attack to discover the key.

### 15.1.1. Spoofing

Meticulous Keyed ISAAC Authentication protects the session against the spoofing of BFD Up packets and keeps the BFD session Up when it would otherwise be reset.

In the event that Meticulously Keyed ISAAC, which is operating as the less computationally intensive authentication mechanism for Optimized BFD, is subverted, the periodic more computationally reauthentication mechanism will limit the time that the session is kept inappropriately in the Up state (Section 5 of [RFC9985]).

The Meticulous Keyed ISAAC Authentication method allows the BFD endpoints to detect a malicious packet via a number of different methods. Packets that are malformed are discarded. Packets that do not pass the BFD state machine [RFC5880] (Section 6.2) checks are discarded. Packets that do not have the correct sequence number, Seed, and Auth Key are discarded. These discarded packets have no effect on the BFD state machine.

The correlation between the sequence number and the Auth Key ensures that each sequence number has a corresponding Auth Key associated with it. The structure and design of the ISAAC CSPRNG ensures that each Auth Key is unique and is unguessable.

Performing an attack on this authentication method would require all of the following to be true:

- The attacker is on-path and can perform an active attack.
- The attacker has the contents of one or more packets.
- The attacker has deduced the secret key used for ISAAC and is able to correlate the sequence number to the current ISAAC state.

These conditions are unlikely to all be true. If the secret key is long and complex, the search space to guess the secret key is too large to discover via brute-force. The use of the Seed and Your Discriminator fields when seeding ISAAC adds 64 bits of entropy to each session, which further makes offline dictionary attacks impractical.

#### 15.1.2. Reuse of Keys

The cryptographic strength of the Optimized Authentication Mode methods is significantly different between SHA-1 and ISAAC. While ISAAC has had cryptanalysis and has not been shown to be broken, that analysis is limited. The question then is whether or not it is safe to use the same key for both mechanisms (SHA-1 and ISAAC), or should we require different keys for each mechanism?

ISAAC is seeded not only with the secret key, but also 32 bits of random data along with 32 bits of a sequence number. The use of this added randomness increases the difficulty of breaking the secret key.

If we recommend different keys, then it is possible for the two keys to be configured differently on each side of a BFD link. For example, a correctly configured key could allow to the BFD state machine to advance to Up. Then, when the session switches to using to less computationally intensive Optimized Authentication Mode with a different key, that key may not match and the session would immediately drop. Suggesting that the keys be identical instead means that no such misconfiguration is possible.

If it becomes possible to recover the secret key for the Meticulous Keyed ISAAC Auth Type and the same key is utilized as a key for more computationally intensive authentication types, such as the MD5 and SHA-1 types defined in this document, then authentication for those mechanisms would be compromised.

Implementations are therefore free to use the same key or different keys for the Optimized Authentication Modes. The choice to use the a single secret key or distinct secret key per Optimized Authentication Mode must be evaluated by the operator balancing their security and operational requirements.

#### 15.1.3. Random Number Considerations

BFD [RFC5880] and its Authentication mechanisms, including Meticulous Keyed ISAAC Authentication specified in this document, make use of random numbers. Such numbers are used in:

- Per BFD session Local Discriminators (bfd.LocalDiscr - Section 6.8.1 of [RFC5880])
- Initial Authentication sequence number (bfd.XmitAuthSeq - Section 6.8.1 of [RFC5880])
- Meticulous Keyed ISAAC Authentication, ISAAC Format Seed (Section 4.1)

The mechanism defined in this document creates an instance of ISAAC for each BFD session seeded by that session's secret key(s) and two locally generated random numbers: the session's Local Discriminator echoed back in the protocol as Your Discriminator and a locally generated Seed. These random numbers are infrequently generated by comparison to the use case for BFD Optimized Authentication that ISAAC addresses. Thus, stronger random number generators with better guarantees of entropy can be used for these purposes.

It is **RECOMMENDED** that these locally generated random numbers used for the BFD protocol and for initializing ISAAC utilize a non-ISAAC CSPRNG.

Random numbers in BFD **MUST** come from a different source than the ISAAC generator used to create per-BFD session Auth Keys. A different instance of an ISAAC generator **MAY** be used to create random numbers for use elsewhere in BFD. In order avoid inappropriate disclosure of local random number generator state, that instance **MUST** be distinct from the generator used for per-session Auth Keys, and it **MUST NOT** be keyed from any BFD session's secret key.

### 15.2. YANG Security Considerations

This section is modeled after the template described in Section 3.7.1 of [RFC9907].

The "ietf-bfd-met-keyed-isaac" YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as the Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., Secure Shell (SSH) [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and (2) have to use mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are no particularly sensitive writable data nodes.

There are no particularly sensitive readable data nodes.

There are no particularly sensitive RPC or action operations.

The YANG module defines a set of identities. These identities are intended to be reused by other YANG modules. The module by itself does not expose any data nodes that are writable, data nodes that contain read-only state, or RPCs. As such, there are no additional security issues related to the YANG module that need to be considered.

# 16. References

## 16.1. Normative References

**[ISAAC]** Jenkins, R. J., "ISAAC and RC4", 1996, <https://www.burtleburtle.net/bob/rand/isaac.html>.

**[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/info/rfc2119>.

**[RFC3688]** Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

**[RFC5880]** Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

**[RFC6020]** Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

**[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/info/rfc8174>.

**[RFC8177]** Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

**[RFC8341]** Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

**[RFC9985]** Jethanandani, M., Mishra, A., Haas, J., Saxena, A., and M. Bhatia, "Optimizing Bidirectional Forwarding Detection (BFD) Authentication", RFC 9985, DOI 10.17487/RFC9985, June 2026, <https://www.rfc-editor.org/info/rfc9985>.

### 16.2. Informative References

**[ISAAC-Plus]** Aumasson, J-P., "On the pseudo-random generator ISAAC", Cryptology ePrint Archive, Paper 2006/438, 2006, <https://eprint.iacr.org/2006/438.pdf>.

**[RFC4252]** Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

**[RFC6241]** Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

**[RFC8040]** Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

**[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/info/rfc8439>.

**[RFC8446]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

**[RFC9000]** Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

**[RFC9907]** Bierman, A., Boucadair, M., Ed., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 9907, DOI 10.17487/RFC9907, March 2026, <https://www.rfc-editor.org/info/rfc9907>.

## Acknowledgments

The authors want to thank Ketan Talaulikar for his reviews and suggestions that have improved the document.

## Contributors

The authors of this document want to acknowledge Ankur Saxena and Reshad Rahman as contributors to this document.

## Authors' Addresses

**Alan DeKok**
InkBridge Networks
100 Centrepointe Drive #200
Ottawa ON K2G 6B1
Canada
Email: alan.dekok@inkbridge.io

**Mahesh Jethanandani**
Arrcus
Email: mjethanandani@gmail.com

**Sonal Agarwal**
Arrcus, Inc.
170 W. Tasman Drive
San Jose, CA 95070
United States of America
Email: sagarwal12@gmail.com

**Ashesh Mishra**
Aalyria Technologies
Email: ashesh@aalyria.com

**Jeffrey Haas**
HPE
Email: jeffrey.haas@hpe.com