Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 17 February 2026

A. Dekok
InkBridge Networks
M. Jethanandani
Kloud Services
S. Agarwal
Cisco Systems, Inc
A. Mishra
Aalyria Technologies
J. Haas
HPE

16 August 2025

Meticulous Keyed ISAAC for BFD Optimized Authentication
draft-ietf-bfd-secure-sequence-numbers-23

Abstract

This document describes a new BFD Optimized Authentication Mode, Meticulous Keyed ISAAC Authentication. This mode can be used to authenticate BFD packets with less CPU time cost than using MD5 or SHA1, with the tradeoff of decreased security. This mechanism cannot be used to signal state changes, but it can be used to maintain a session in the the Up state.

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 17 February 2026.

Copyright Notice

Table of Contents

1. Introduction

BFD [RFC5880] (Section 6.7) defines a number of authentication mechanisms, including Simple Password, and various other methods based on MD5 and SHA1 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. The goal of this specification is to use a simple method to prevent spoofing of the BFD session being Up. This specification therefore define a Optimized Auth Type that allows parties to securely signal that they are still in the Up state.

This document proposes the use of an Authentication method which provides per-packet authentication using methods similar to that defined in BFD [RFC5880] for Meticulous Keyed MD5 and Meticulous Keyed SHA1. The method defined here has the benefit that it has less impact on resource constrained systems than either MD5 or SHA1. The algorithm chosen is a seeded pseudo-random number generator named ISAAC [ISAAC]. ISAAC has been subject to significant cryptanalysis in the past thirty years, most notably ISAAC+ [ISAAC_]. The only issue found was with initial seeding, and the method proposed here is safe from that attack. 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.

1.1. Meticulous Keying

RFC5880 [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 BFD [RFC5880] (Section 4.2):

Sequence Number
: 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 which is sent. The term "keyed" means that the packets are authenticated via the use of a secret key or keys which are known to both sender and receiver. The term "meticulous keyed" therefore refers to BFD authentication type where each packet is unique, 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this note before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

2025-08-16 with the actual date of the publication of this document.

2. Experimental updates to RFC 5880

This document describes an experimental update 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 the existing document.

Some of the state variables in BFD [RFC5880] (Section 6.8.1), are related to the authentication type being used for a particular session. However, the definitions given in BFD [RFC5880] are specific to Keyed MD5 or SHA1 Authentication, which limit their utility for new authentication types. For the purpose of the experiment, this specification updates the definition of some of the state variables as given below.

These updated definitions are entirely compatible with the definitions given in BFD [RFC5880] (Section 6.8.1), and require no changes to existing configurations or implementations. Instead, the updated definitions clarify that the state variables apply to the current authentication type, no matter what it is.

The text first updates the [RFC5880] definitions, and then define a new authentication type which uses these updated definitions.

These updated definitions also mean that Authentication Sections SHOULD include a Sequence Number field. Where a Sequence Number is not used (as with Simple Password) the variables bfd.RcvAuthSeq and bfd.XmitAuthSeq MUST be set to zero. Where an Authentication Section uses a meticulous keyed authentication type, it MUST include a Sequence Number field.

bfd.AuthType:
: The current authentication type in use for this session, as defined in BFD [RFC5880] (Section 4.1), or zero if no authentication is in use.

  Packets which indicate a state transition MUST use an authentication method which provides for full packet integrity checks. When the bfd.SessionState value is Up, packets MAY use an optimized authentication method (Optimizing BFD Authentication [I-D.ietf-bfd-optimizing-authentication]) such as Meticulous Keyed ISAAC.

bfd.RcvAuthSeq:
: A 32-bit unsigned integer containing the last sequence number for the current Authentication Section that was received. The initial value is unimportant.

bfd.XmitAuthSeq:
: A 32-bit unsigned integer containing the next sequence number for the Authentication Section which will be transmitted. This variable MUST be initialized to a random 32-bit value. This value SHOULD be taken from a cryptographically strong pseudo-random number generator (CSPRNG).

bfd.AuthSeqKnown:
Set to 1 if the next expected Authentication Section has a sequence number which is known, or 0 if it is not known. This variable MUST be initialized to zero.

This variable MUST be set to zero after no packets have been received on this session for at least twice the Detection Time. This ensures that the sequence number can be resynchronized if the remote system restarts.

3. Architecture of the Auth Type Method

When BFD uses authentication, methods using MD5 or SHA1 are CPU intensive, and can negatively impact systems with limited computational power.

However, once the session transitions into the Up state, the packet integrity checks may not be needed. The continued reception of correctly formed packets in the Up state serve as a good indication that the sender is operational. However, removing all security poses issues where an attacker could spoof Up packets, causing a session to erroneously remain in the Up state. Or an attack could spoof a single Down packet, causing the session to erroneously go down.

Instead, an optimized authentication mechanism as described in Optimizing BFD Authentication [I-D.ietf-bfd-optimizing-authentication], which permits BFD to use an authentication method which is less computationally expensive than MD5 or SHA1, but is still strong enough that to prevent an on-path attack. The method defined here does not provide for per-packet integrity checks, and is therefore is not suitable for signaling state changes. Instead, it provides a difficult to forge indication that the session remains in the Up state.

This indication is a 32-bit number taken from a CSPRNG. The number changes for every packet, and has a 1-1 correlation with the Sequence Number. Changing the Sequence Number for every packet means that the receiving party knows the packet is timely, and was not replayed. The associated 32-bit number indicates that the packet was from the correct sender. The combination of the two prevents on-path attackers from forging packets.

ISAAC is used here as a way to generate an infinite stream of pseudo-random numbers, referred to here as "Auth Key"s. 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, then only the sending party could have generated it. The sender is therefore authentic, even if the packet contents have potentially been modified in transit.

Note that with this Auth Type method, the full packet contents are not signed or authenticated. Therefore, Meticulous Keyed ISAAC Authentication MUST NOT be used to signal BFD state changes. For BFD state changes, and a more optimized way to authenticate packets, please refer to BFD Authentication [I-D.ietf-bfd-optimizing-authentication]. Instead, the packets containing Meticulous Keyed ISAAC Authentication are only a signal that the sending party is still alive, and that the sending party is authentic. That is, this Auth Type method MUST only be used when bfd.SessionState=Up, and the State (Sta) field equals 3 (Up).

3.1. Rationale for ISAAC and Operational Overview

There are many CSPRNGs available, so we explain why ISAAC was chosen.

The goal for this optimized authentication was to provide a strong 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 which 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 CSRNGs maintain minimal state, and are designed to produce a long sequence of pseudo-random numbers from a few simple calculations. In general, every call to the CSPRNG function modifies the internal state un an irreversible fashion, and then produces a new random number as the result.

It could be possible to use such a generator, and then to 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 of 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. Again, we choose ISAAC as the answer here. It can accept keys up to 8192 octets in length, which is more than sufficient for BFD.

Finally, ISAAC has been subject to significant cryptanalysis in the past thirty years, most notably ISAAC+ [ISAAC_]. There are no known vulnerabilities. In contrast, other CSPRNGs may be newer, but have generally been subject to less analysis.

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

* The administrator provides a secret key which 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 which 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 which is associated with the received packet, and checks if the received packet contains the expected number.

* BFD supports packet rates of hundred 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 use, 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 off-line dictionary attack. ISAAC has been subject to cryptographic analysis, and the numbers produced by it are secure. 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 which satisfies both 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 (TBD1), or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (TBD2), and the Opt. Mode field contains 2 (Section 7 of [I-D.ietf-bfd-optimizing-authentication]) the Authentication Section has the following format:

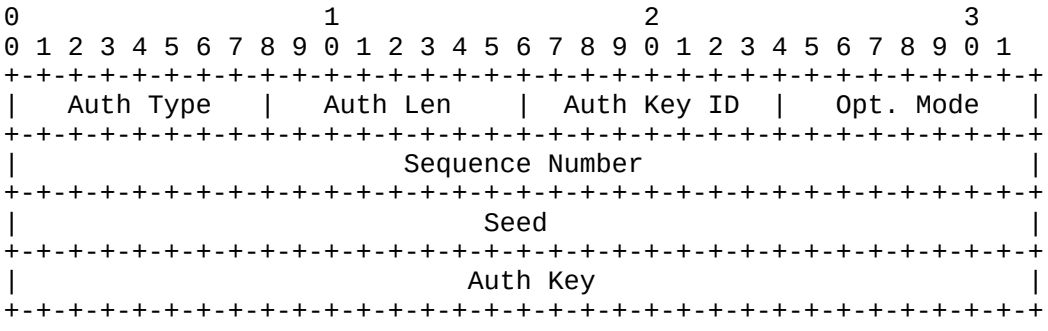

Figure 1: Meticulous Keyed ISAAC Authentication Format

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

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

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 Authentiation Mode is defined in Section 3 of [I-D.ietf-bfd-optimizing-authentication]. When the Auth Type is either Optimized MD5 Meticulous Keyed ISAAC Authentiation (TBD1), or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (TBD2), and the format is Meticulous Keyed ISAAC Authentication Format, the Optimized Authentication Mode field will be set to 2, Optimized.

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 which is used in conjunction with the shared key in order to configure and initialize the ISAAC pseudo-random-number-generator (PRNG). It is used to identify and secure different "streams" of random numbers which are generated by ISAAC.

Auth Key:
This field carries the 32-bit (4 octet) ISAAC output which is associated with the Sequence Number. The ISAAC PRNG MUST be configured and initialized as given in Section 10, below.

Note that the Auth Key here does not include any summary or hash of the 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 (TBD1), and the Opt. Mode field contains 1 (Section 7 of [I-D.ietf-bfd-optimizing-authentication]) the Authentication Section has the following format:

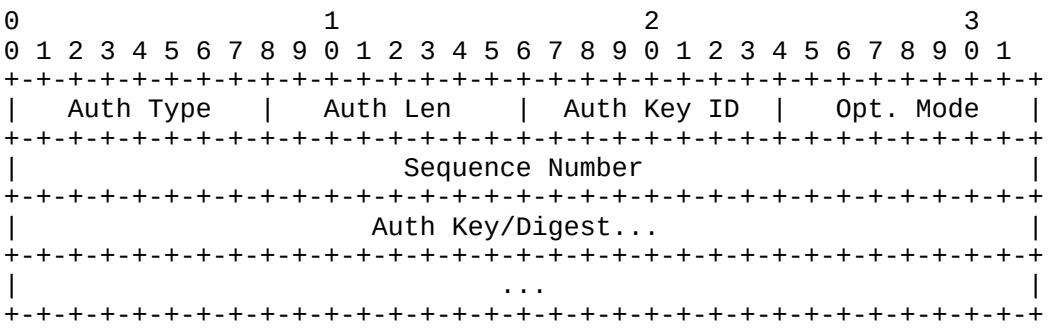

Figure 2: Meticulous Keyed ISAAC Authentication, MD5 Format

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

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

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 Authentiation Mode is defined in Section 3 of [I-D.ietf-bfd-optimizing-authentication]. When the Auth Type is either Optimized MD5 Meticulous Keyed ISAAC Authentiation (TBD1), and the format is MD5 Authentication Format, the Optimized Authentication Mode field will be set to 1, Strong.

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, SHA1 Format

If the Authentication Present (A) bit is set in the header, and the Authentication Type field contains Optimized SHA1 Meticulous Keyed ISAAC Authentication (TBD2), and the Opt. Mode field contains 1 (Section 7 of [I-D.ietf-bfd-optimizing-authentication]) the Authentication Section has the following format:

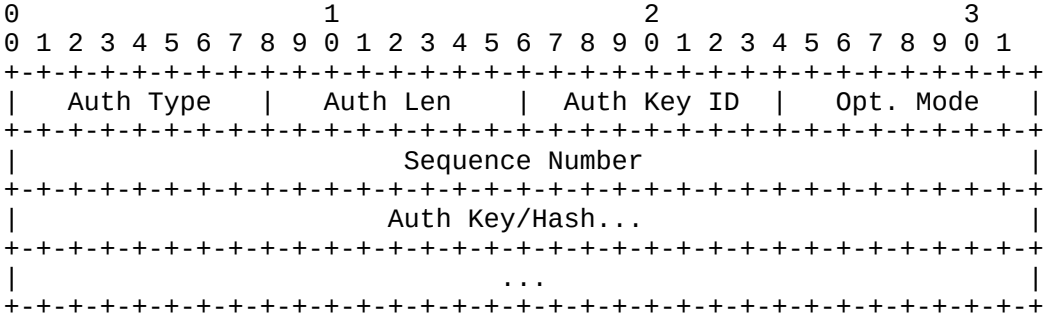

Figure 3: Meticulous Keyed ISAAC Authentication, SHA1 Format

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

Auth Len:
The length of the Authentication Section, in bytes. For Meticulous Keyed ISAAC SHA1 Authentication Format, the length is 28.

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 Authentiation Mode is defined in Section 3 of [I-D.ietf-bfd-optimizing-authentication]. When the Auth Type is either Optimized SHA1 Meticulous Keyed ISAAC Authentiation (TBD1), and the format is SHA1 Authentication Format, the Optimized Authentication Mode field will be set to 1, Strong.

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 SHA1 hash for the packet. The procedure for calculating this field is documented in Section 6.7.4 of [RFC5880].

5. Procedures for BFD Authentication using Meticulous Keyed ISAAC, MD5 or SHA1 Formats

For these modes of authentication, the sending and receiving procedures are identical to those documented in [RFC5880] for the Meticulous MD5 Section 6.7.3 of [RFC5880] or SHA1 Authentication Section 6.7.4 of [RFC5880] with the exceptions that different authentication types are exercising the same procedures in their meticulous modes.

6. 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.

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 (TBD1); or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (TBD2).

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 3.0 of [I-D.ietf-bfd-optimizing-authentication].

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 for a packet.

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 (TBD1) or Optimized SHA-1 Meticulous Keyed ISAAC Authentication (TBD2)), 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 3.0 of [I-D.ietf-bfd-optimizing-authentication].

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 matches 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.

7. 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 which 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 which 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 pseudo-random numbers.

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

bfd.MetKeyIsaacRcvAuthData:
A data structure which 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 which 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 which 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 pseudo-random numbers.

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

bfd.MetKeyIsaacXmitAuthData:
A data structure which 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.

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 pseudo-random numbers using the ISAAC random number generator.

A particular Secret Key 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 re-sync or re-initialize an ongoing session with a different Auth Key ID and correspondingly different Secret Key.

If this Auth Type method was defined as being initialized without a per-session Seed, then an attacker could pre-compute the ISAAC states for many keys, and perform an off-line dictionary attack. The use of the Seed makes these attacks unfeasable.

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 support.

The Secret Key MUST be at least eight (8) octets in length, and SHOULD NOT be more than 128 octets in length.

There are no known issues with using the same secret Key for multiple Auth Type methods. However, it is RECOMMENDED that administrators use different Secret Keys for each Auth Type.

9. Transition to using ISAAC

A BFD session which uses Optimized MD5 Meticulous Keyed ISAAC Authenticationor or Optimized SHA-1 Meticulous Keyed ISAAC Authentication MUST begin a session with Auth Type set to the relevant authentication type, and the the Optimized Authentication Mode field set to 1 (Strong).

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

Once the BFD session has transitioned to the Up state, the sender MAY send the second and subsequent packets for the Up start with the Optimized Authentication Mode field containing value 2 (Optimized).

When a system first receives a packet containing Optimized Authentication Mode field with value 2 (Optimized), it initialize 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 is below in Section 10.1 and Section 10.2.

The first packet after the transition to the Up state is the only time when the ISAAC random number generator 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 re-keyed.

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 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 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.

However, the operation of those 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.

When there is a transition to using ISAAC the first time, the initial ISAAC state has to be seeded. The next section describes this seeding process.

10. Seeding ISAAC

The Seed field is used to identify and secure different "streams" of random numbers which are generated by ISAAC. Each session uses a different Seed, which is used along with the Your Discriminator field, 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 the following structure:

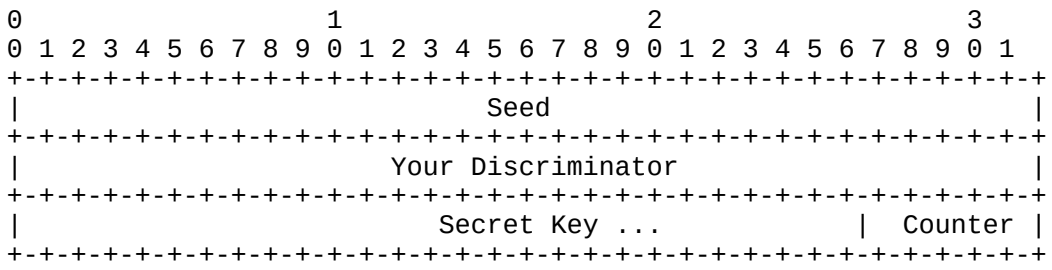

Figure 4: ISAAC Initialization Structure

Where the Your Discriminator field is taken from the BFD packet defined in RFC5880 Section 4.1 [RFC5880]. 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 the initial seeding of ISAAC avoids the seeding issues discussed in ISAAC+ [ISAAC_].

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 which indicates how long the Secret Key is, and there is no trailing zero or NUL byte which indicates the end of the Secret Key. Implementers 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 which were generated using the above initial values.

Field	Value(s)
Seed	0x0bfd5eed
Y-Disc	0x4002d15c
Secret Key	RFC5880June
Counter	0...50

Figure 5: Test Inputs for seeding ISAAC

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

Figure 6: Expected Outputs

This construct provides for 64 bits of entropy, of which 32 bits is controlled by each party in a BFD session. For security, each implemention SHOULD randomize their discrimator fields at the start of a session, as discussed in Section 10 [RFC5880].

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 signalled a BFD state change with a packet that is authenticated using a strong 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 which 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.

If, however, the packet’s Sequence Number differs from the expected value, then the difference "N" indicates how many packets were lost. The receiver then can 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 found key does match the Auth Key in the packet, then the bfd.MetKeyIsaacRcvAuthIndex field is initialized to the 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 pseudo-random stream to be re-synchronized 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 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 re-use of any individual pseudo-random number which 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 which are unguessable, and which can be used to authenticate the sending party.

Each system sending BFD packets chooses its own seed, and generates its own sequence of pseudo-random numbers using ISAAC, and place those values into the Auth Key field. Each system receiving BFD packets runs a separate pseudo-random 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 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. Therefore, it is difficult to change the keys during the operation of a BFD session without affecting the stability of the BFD session. Therefore, it is RECOMMENDED to administratively disable the BFD session before changing the keys.

That is, while the Auth Key ID field provides for the use 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 signalling or negotiation needs to be done "out of band" for BFD, and usually via manual administrator configuration.

12. Transition away from using ISAAC

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

The strong 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 strong authentication type for packets which maintain the session in an Up state. The interval between these switches SHOULD be long enough that the system still gains significant benefit from using Meticulous Keyed ISAAC Authentication. That is, there SHOULD be thousands of packets used in between any switch to the strong authentication type. See BFD Authentication [I-D.ietf-bfd-optimizing-authentication] for appropriate procedure on switching Optimized Authentication Mode.

The nature of Meticulous Keyed ISAAC Authentication means that there is no issue with this switch, so long as it is for a small number of packets. From the point of view of the Meticulous Keyed ISAAC state machine, this switch can be handled similarly to a lost packet. The state machine simply notices that instead of Sequence Number value being one more than the last value used for ISAAC, it is larger by

two. The ISAAC state machine then calculates the index into the current "page", and uses the found number to validate (or send) the Auth Key.

That is, the switch away from, and back to Meticulous Keyed ISAAC Authentication MUST NOT cause the Meticulous Keyed ISAAC Authentication mode to be re-seeded. ISAAC is only re-seeded when the session transitions to the Up state, and the session remains in the Up state during this process. As a result, no re-seeding is performed.

It is RECOMMENDED that the session use the strong authentication type only for a small number of packets, before moving back to using Meticulous Keyed ISAAC Authentication. There are few benefits to continuing the strong authentication type for extended periods of time. If the party would send hundreds of packets with strong authentication authentication, that would negate the benefit of using optimized authentication.

It is therefore RECOMMENDED that the number of packets using the strong authentication mode be no more than the value of bfd.DetectMulti. Implementations MUST also run the ISAAC state machine as normal during any strong authentication, including calculating any next pages. This requirement ensures that the parties can switch back to Meticulous Keyed ISAAC Authentication mode at any time, as the ISAAC state machine remains in synchronization with the Sequence number in the packets.

[I-D.ietf-bfd-optimizing-authentication] describes this switch in more detail, including the suggestion to use Poll sequence to start the strong authentication.

13. The YANG Model

This YANG module adds two identities defined in this document. One of them uses the Meticulous Keyed MD5 as the strong authentication and Meticulous Keyed ISAAC Keyed as the less computationally intensive authentication. The other uses the Meticulous Keyed SHA-1 as the strong authentication and Meticulous Keyed ISAAC Keyed as the less computationally intensive authentication.

```
<CODE BEGINS> file "ietf-bfd-met-keyed-isaac@2025-08-16.yang"
module ietf-bfd-met-keyed-isaac {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-met-keyed-isaac";
  prefix "bfdmia";

  import ietf-key-chain {
```

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

    organization
      "IETF 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 experimental YANG module provides identities derived from
       the ietf-key-chain model for the BFD Meticulous Keyed ISAAC
       authentication mechanism.

       Copyright (c) 2025 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 XXXX
       (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
       for full legal notices.

       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 (RFC 2119) (RFC 8174) when, and only when,
       they appear in all capitals, as shown here.";

    revision "2025-08-16" {
      description
        "Initial Version.";
      reference
```

```
        "RFC XXXX: Meticulous Keyed ISAAC for BFD 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 Keyed as the
         less computationally intensive authentication.";
      reference
        "RFC XXXX: Meticulous Keyed ISAAC for BFD 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 Keyed as
         the less computationally intensive authentication.";
      reference
        "RFC XXXX: Meticulous Keyed ISAAC for BFD Authentication.";
    }
  }
  <CODE ENDS>
```

14. IANA Considerations

This documents requests the assignment of two BFD Auth Types, one URI and one YANG model.

14.1. BFD Auth Types

This document requests an update to the registry titled "BFD Authentication Types". IANA is requested to assign two new BFD AuthType:

* TBD1: Optimized MD5 Meticulous Keyed ISAAC Authentication with a suggested value of 7.

* TBD2: Optimized SHA-1 Meticulous Keyed ISAAC Authentication with a suggested value of 8.

14.2. IETF XML Registry

This document registers one URIs in the "ns" subregistry of the "IETF XML" registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

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

This document registers one YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-bfd-met-keyed-isaac
namespace:    urn:ietf:params:xml:ns:yang:ietf-bfd-met-keyed-isaac
prefix:       bfdmia
reference:    RFC XXXX
```

15. Security Considerations

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 random numbers, with one generated by each party to the BFD session. An attacker cannot simply perform an off-line brute-force dictionary attack to discover the key. Instead, any analysis has to include the particular 64 bits of entropy used for a particular session. As a result, dictionary attacks are more difficult than they would be if the PRNG generator depended on nothing more than the Secret Key.

The security of this proposal depends strongly on ISAAC. This generator has been analyzed for almost three decades, and has not been broken. Research shows that there are few other CSRNGs which are as simple and as fast as ISAAC. For example, many other generators are based on AES, which is infeasibe for resource constrained systems.

15.1. Spoofing

The Meticulous Keyed ISAAC Authentication method allows the BFD end-points to detect a malicious packet via a number of different methods. Packets which are malformed are discarded. Packets which do not pass the BFD state machine [RFC5880] (Section 6.2) checks are discarded. Packets which 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. The ISAAC RNG has been shown to be infeasibe to reverse. 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 makes an off-line dictionary attacks infeasible.

However, the actual attack which we are protecting BFD from is availability. That is, the attacker is trying to shut down then connection when the attacked parties are trying to keep it up. An on-path attacker can simply discard or corrupt any packets at will, which will affect not just BFD, but all traffic on the connection. As a result, the attacks here seem to be irrelevant in practice.

15.2. Re-Use of keys

The strength of the Auth-Type methods is significantly different between the strong one like 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 Auth Type methods (SHA1 and ISAAC), or should we require different keys for each method?

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 weaker Auth Type with a different key, that key may not match, and the session would immediatley drop. Suggesting instead that the keys be identical means that no such misconfiguration is possible.

Implementations are therefore free to use the same key, or different keys. The use of the same key for for both strong and optimized authentication is acceptable, as ISAAC is keyed not only with the authentication key, but also depends on 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 key, and makes off-line dictionary attacks infeasible.

16. Contributors

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

17. Acknowledgements

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

18. References

18.1. Normative References

[I-D.ietf-bfd-optimizing-authentication]
Jethanandani, M., Mishra, A., Saxena, A., Bhatia, M., and J. Haas, "Optimizing BFD Authentication", Work in Progress, Internet-Draft, draft-ietf-bfd-optimizing-authentication-28, 5 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-bfd-optimizing-authentication-28>.

[ISAAC] Jenkins, R. J., "ISAAC", http://www.burtleburtle.net/bob/rand/isaac.html, 1996.

[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>.

18.2. Informative References

[ISAAC_] Aumasson, J-P., "On the pseudo-random generator ISAAC", https://eprint.iacr.org/2006/438.pdf, 2006.

Authors' Addresses

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

Mahesh Jethanandani
Kloud Services
Email: mjethanandani@gmail.com

Sonal Agarwal
Cisco Systems, Inc
170 W. Tasman Drive
San Jose, CA 95070
United States of America
Email: agarwaso@cisco.com
URI: www.cisco.com

Ashesh Mishra
Aalyria Technologies
Email: ashesh@aalyria.com

Jeffrey Haas
HPE
Email: jhaas@juniper.net