Network Working Group
Internet Draft
expires in six months

William A. Nace(NSA)
James E. Zmuda(SPYRUS)
November 21st, 1997

PPP Fortezza Encryption Encapsulation Protocol
<draft-ietf-pppext-feep-01.txt>

Status of this Memo

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links

PPP also defines an extensible Link Control Protocol, which allows negotiation of an Authentication Protocol for authentication of its peer before allowing Network Layer protocols to transmit over the link.

One of the Authentication protocols that can be negotiated is

the EAP. The EAP can be used in any of a number of variants. When the EAP is used in it's KEA variant, this results in mutual authentication and key generation. This key is available for use during the PPP data transfer phase by an encryption encapsulation. The encryption encapsulation that is described in this memo is one possible encapsulation that can use the keying material generated by the EAP KEA protocol.

1. Introduction

The PPP encryption encapsulation will take a PPP packet including the protocol field, apply the chosen encryption algorithm, place the resulting cipher text (and in this specification, an explicit sequence number) in the information field of another PPP packet. The decryptor will apply the inverse algorithm and interpret the resulting plain text as if it were a PPP packet which had arrived directly on the interface.

1.1. Specification of Requirements

In this document, several words are used to signify the requirements of the specification. These words are often capitalized.

MUST — This word, or the adjective required, means that the definition is an absolute requirement of the specification.

MUST NOT — This phrase means that the definition is an absolute prohibition of the specification.

SHOULD — This word, or the adjective recommended, means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications must be understood and carefully weighed before choosing a different course.

MAY — This word, or the adjective optional, means that this item is one of an allowed set of alternatives. An implementation which does not include this option MUST be prepared to interoperate with another implementation which does include the option.

1.2. Terminology

This document frequently uses the following terms:

Cryptographic Algorithm
: The algorithm used to encrypt data. In the Fortezza Encryption Encapsulation Protocol, the Cryptographic Algorithm is Skipjack.

Cryptographic mode
: Block ciphers may be used in several different manners, known as modes. The basic mode of operation of a block cipher is called Electronic Code Book (ECB). This mode always produces the same block of ciphertext as output for the same block of ciphertext as input. In the interest of increasing the random character of the outputs of encipherment other modes were introduced. One such mode is called Cipher Block Chaining (CBC).

Cipher Block Chaining (CBC)
: The CBC mode of operation for Block Ciphers was introduced to further obscure the relationship between the ciphertext blocks and the actual input data blocks by manipulating the input data block before encipherment. Block ciphers in CBC mode exhibit the property that the result of an encryption or decryption is determined by not only the value of this particular block of data, but also the value of the ciphertext from the previous block. Hence the term "Chaining" in the name.

Cryptographic Synchronization
: For Block ciphers in CBC mode the loss of a block of data will result in not only failing to decrypt that block, but may also result in failure to decrypt the immediately succeeding block. This condition is known as a loss of "cryptographic synchronization." Note that for block mode algorithms in CBC mode, the loss of cryptographic synchronization only persists for one block after the missing block in the case of decryp-

tion. On the other hand, any error during the encryption process will propagate through all subsequent blocks. This is not however, strictly speaking, a loss of cryptographic synchronization. This phenomenon is known as "error-propagation." Therefore, the CBC mode of a block cipher may be said to exhibit "error-propagation" during encryption and be subject to the loss of "cryptographic synchronization" during decrypt.

encapsulation The term used to describe the technique of burying one completely formed Protocol Data Unit (PDU), or packet,

in another PDU. The advantages of using this method for providing encryption services is that it can be provided without impacting the operation of protocols above the encapsulation protocol.

encipherment Synonym for encryption.

Initialization Vector (IV)
For a block cipher algorithm in CBC mode, the IV is input into the encryption of the first block in order to further obscure the relationship between the ciphertext and the actual input data by manipulating the input data block before encipherment. For non-streaming protocols, it is required to provide an IV for each individual packet, since the order of delivery of packets may vary in such protocols, and an attempt to decrypt out of order would result in a "loss of cryptographic synchronization." For streaming protocols (like PPP) it is not necessary to provide an IV for each individual packet, as the in-order nature of deliver in a streaming protocol means that the "IV" needed to perform a decryption can be obtained from the preceding block of ciphertext. (Note: although not necessary, an IV MAY still be provided for each protected PPP packet. In fact, for a number of reasons, this is the approach suggested here.)

Integrity Check Value (ICV)

An Integrity Check Value is a value that is computed by the originator over the entire data value that is to be protected. This ICV value is then sent with the data value being protected. Using the same algorithm an ICV value is computed by the recipient over the received data value. The receiver compares the received ICV with the ICV he has computed. If they match, it is probable that the protected data value has not been modified.

padding — For a block mode algorithm, the data that is encrypted must consist of a whole number of "blocks", where the "block" size is determined by the algorithm (often being 8 bytes in length). If the input data does not contain a whole number of crypto block sized units, it will need to be padded until it reaches that length. Padding can be supplied in two forms: 1) with an explicit length field, or 2) as self-describing

padding. Self-describing padding is the type of padding used in the FEEP encapsulation.

peer — The other end of the point-to-point link.

2. PPP Fortezza Encryption Encapsulation Format

The PPP Fortezza Encryption Encapsulation Protocol packet has the following format:

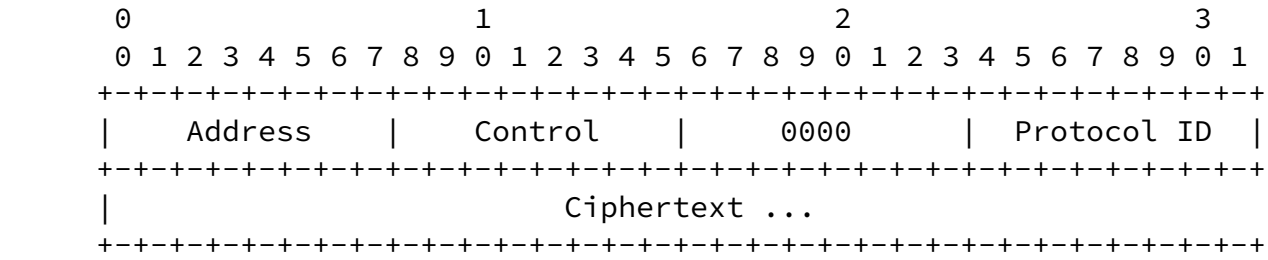


Figure 2.0-1 - The Fortezza Encryption Encapsulation protocol

Address and Control

These fields MUST be present unless the PPP Address and Control Field Compression option (ACFC) has been negotiated.

Protocol ID

The value of this field is 0x53 or 0x55; the latter indicates that ciphertext includes headers for the Multilink Protocol, and REQUIRES that the Individual Link Encryption Control Protocol has reached the Opened state. The leading zero MAY be absent if the PPP Protocol Field Compression option (PFC) has been negotiated.

Ciphertext

The ciphertext is arrived at as described in the following section.

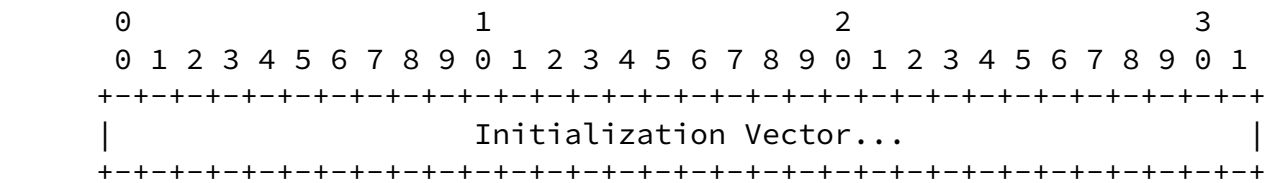
3. PPP Fortezza Encryption Encapsulation Protocol processing

The following sections define the procedures associated with processing the PPP Fortezza Encryption Encapsulation.

These procedures are threefold: First, a pad is appended to the

protocol and information fields of a PPP packet in order to produce a data block that is a multiple of 8 bytes. Second, an Integrity Check Value (ICV) is computed over the result, and this ICV is appended to the packet. Third, and finally, the SKIPJACK Algorithm in CBC mode is applied over this entire packet, including the pad and ICV value.

The following diagram depicts what the ciphertext field in the Fortezza Encryption Encapsulation looks like in detail.



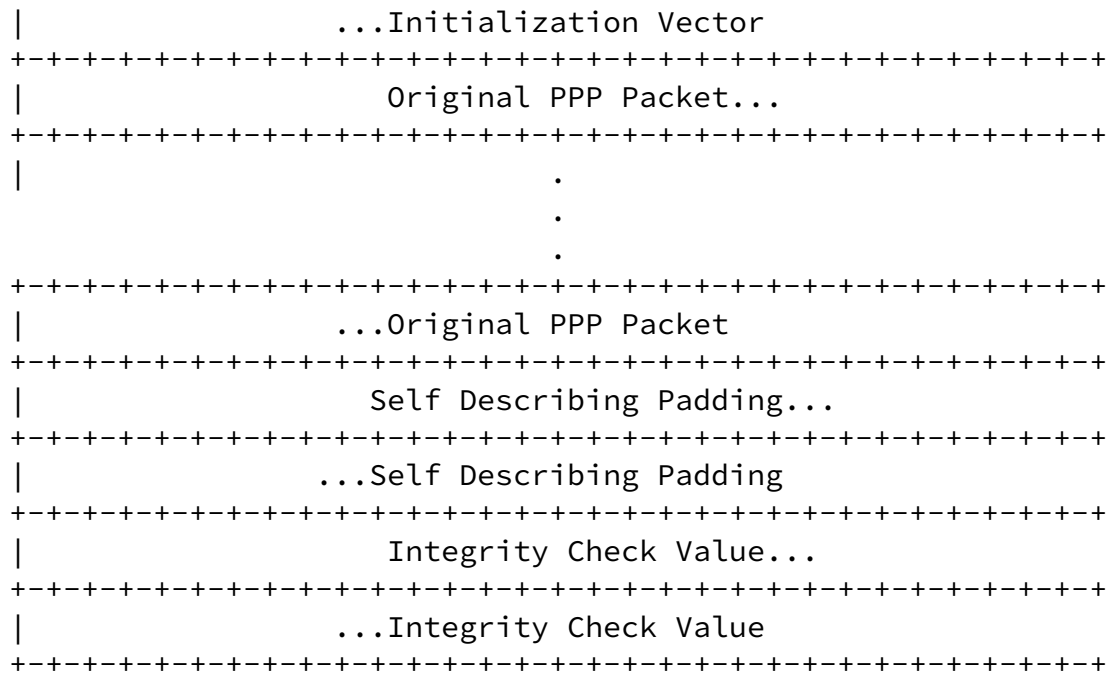


Figure 3.0-1 - The format of the Ciphertext field of the Fortezza Encryption Encapsulation protocol

Initialization Vector

This 8 byte field holds the IV. These are carried on a per PPP packet basis even though PPP is a stream-oriented, or in-order delivery protocol. This is in order to simplify loss-of-sync recovery.

Original PPP packet

This arbitrary sized field holds the original PPP packet.

Self-describing Padding

This field contains the self-describing padding required to produce an 8-byte aligned packet. It is 1 to 8 bytes in length.

Integrity Check Value

This field contains the Integrity Check value computed over everything above. It is 8 bytes in length. The first four bytes are the ICV value itself, as a 4 byte big-endian integer. The second four bytes are all zeroes.

3.1. Padding

Since the SKIPJACK algorithm operates on blocks of 8 octets (and, indeed, so does the ICV algorithm), packets which are of a length which is not a multiple of 8 octets must be padded. This can be injurious to the interpretation of client network-layer protocols being carried over the PPP link, particularly those protocols which do not contain an explicit length field in their protocol headers.

In order to avoid this problem the use of self-describing padding is mandated for use with the PPP encapsulation using SKIPJACK. Self-describing padding is defined in RFC1570. Simply put, self-describing padding works like this:

1. On the generation side, you always place at least one pad byte, and up to eight - in the case where the packet is already block-aligned. Each pad byte placed has a value equal to its index, from 1 to 8. Once padded the PPP packet is integrity-checked and encrypted and placed in a PPP encryption encapsulation frame, as described in the subsequent sections.

2. On the reception side, after the ciphertext is decrypted, and the 8 byte Integrity Check Value field is removed, one looks at the last byte in the remaining data. This is the last byte of the self describing pad field. (The last byte of the self describing pad can be determined by looking for the byte immediately preceding the 8 byte ICV, which in turn is the field immediately preceding the HDLC FCS.) This byte indicates the number of bytes of padding to be discarded.

3.2. Integrity Check Value calculation

The Integrity Check Value is computed in the following manner:

The originator computes the 32-bit 1's complement sum of the data field, after it has been padded out to 64-bit block boundaries. The 1's complement sum is computed by viewing the data as a series of 32-bit big-endian words. The ICV value itself is a 64-bit field. The ones complement sum computed above is placed in the first four bytes and the second four bytes are all zeroes. This ICV value is then appended to the data packet.

On the receiving side, after decryption processing, the recipient computes, using the very same algorithm, the entire 64-bit ICV value and then compares it with what has been received. The two values should match.

3.3. Encryption

The ciphertext is obtained from the plaintext in the following manner:

Generally, the SKIPJACK encryption algorithm, being a block cipher, works on packets composed of a multiple of 8 byte blocks. The above padding and Integrity Check Value calculation steps will have produced a data packet that is a multiple of 8 byte blocks. The Fortezza Encryption Encapsulation thus consists of two steps: first, the pre-pending of a per-packet IV value and secondly, the application of the SKIPJACK algorithm in CBC mode with 64 bit blocks.

On the transmit side, the FEEP encryption encapsulation is performed after the Integrity Check Value processing has appended an ICV and before the HDLC or A-HDLC data link protocol processing. Because FEEP utilizes the PPP HDLC or A-HDLC link protocol it is provided with a sequential and mostly reliable link - e.g. HDLC FCS will detect and eliminate corrupted frames.. For this reason we could utilize chaining across multiple packets. I.E., we could obtain the IV value to be used for encrypting the first block of this packet from the ciphertext resulting from the encryption of the last block of the previous packet.

However, in the interest of simplifying the processing needed to handle a loss of cryptographic synchronization we have elected to include an 8 byte IV value (or "Crypto Synchronization Header" if you will) as the first 8 bytes of the field titled "Ciphertext" in figure 2.0-1 above.

What this means is that with this field each packet will automatically resync itself. Therefore the loss of one packet due to a communications failure will result only in the loss of one packet. Without the Crypto Synchronization Header, the loss of one packet due to communications failure would result in the loss of two packets as the immediately following packet would fail to decrypt properly. The ECP protocol can be utilized to provide crypto resynchronization, but we believe the use of a per packet Crypto Synchronization Header to be the simpler approach.

If the async control character map option has been negotiated on the link, the sender applies mapping after the encryption algorithm has been run.

There are a lot of details concerning what constitutes the Protocol and Information fields, in the presence or non-presence of Multilink, and whether the ACFC and PFC options have been negotiated, and the type of padding to perform.

Regardless of whether ACFC has been negotiated on the link, the sender applies the encryption procedure to only that portion of the packet excluding the address and control field.

The plaintext is obtained from the ciphertext in the following manner:

The information field of the encapsulating packet is obtained and decrypted using SKIPJACK in CBC mode. Because we are assuming the first 8 bytes constitutes an IV value for the purposes of the CBC mode, we load the first eight bytes in the PPP packets data field into the Skipjack algorithm as IV, and the second 8 bytes as the first data packet. The resulting decrypted PPP packet is handed to Integrity Check Value receive processing.

The key to use for these encryption/decryption operations is the one developed during the KEA authentication accomplished during the authentication phase [2].

3.4. MRU considerations and Padding

Because of the padding described above, and the additional protocol field and the two bytes of sequence number, it can be seen that the size of the resulting PPP encryption encapsulated frame may be as much as 27 bytes larger than the original, unencapsulated PPP frame.

Depending upon the MRU size established during the Link establishment

phase, this expansion could result in the need for fragmentation of a single, original PPP packet into two encryption encapsulated frames.

A number of different solutions are available for dealing with this.

The simplest solution is to be aware of this problem during Link establishment phase and simply to adjust the value of the MRU that is negotiated with the peer PPP entity to be 27 greater than that being requested by the original PPP link without encryption. That, coupled with the step of reporting the original MRU value to the upper layers as the MTU.

Another solution is to make use of the PPP Multi-link procedures which add the overhead of another PPP packet header with additional sequence numbers and "Begin" and "End" bits to signal fragment boundaries. Given the fact that we only produce, at most, two fragments...as the extra 27 bytes will surely fit in any MRU value negotiated....the Multilink procedure will not actually be that onerous in terms of buffer overhead.

For the purposes of this initial recommendation, the former procedure is to be utilized.

3.5. Cryptographic synchronization handling

We avoid the problem that the loss of a packet due to communications failure will cause the loss of a second packet due to loss of cryptographic synchronization through the use of an explicit IV, or "Crypto Synchronization Header" at the beginning of each PPP packet. This approach costs 8 bytes per packet for a savings that is dependent on link quality and is thus hard to quantify.

3.6. SKIPJACK mode

As PPP is a packet oriented protocol the cryptographic mode selected

for SKIPJACK operations is Cipher Block Chaining mode on 64 bits of data.

Security Considerations

This memo defines a method for securing the exchanging of data frames over a PPP link.

References:

[1] Simpson, W. A., 'The Point to Point Protocol (PPP)', July 1994, RFC 1661.

[2] Nace, W. A., and Zmuda, J. E. 'PPP EAP KEA Public Key Authentication Protocol', November 21st, 1997, draft-ietf-pppext-eapkea-00.txt, work in progress.

Acknowledgements:

Thanks to Peter Yee and Russ Housley who provided helpful comments on earlier versions of this Memo. Thanks are due to Gerry Meyer for helping me understand his ECP Internet Draft. Thanks also are due to Keith Sklower and Gerry Meyer for their DESE RFC. And thanks to Bill Simpson for the standard PPP spec boilerplate from which I have borrowed heavily.

Chair's Address:

The working group can be contacted via the current chair:

Karl Fox
Ascend Communications, Inc.

Email: karl@ascend.com

Author's Address:

Questions about this memo can also be directed to:

DIRNSA
Attn: X22 (W. Nace)
9800 Savage Road
Fort Meade, MD 20755-6000
USA

Phone: +1 410 859-4464
Email: WANace@missi.ncsc.mil

James E. Zmuda
SPYRUS
2460 N. First Street
Suite 100
San Jose, CA 95131-1023
USA

Phone: +1 408 432-8180
Email: jzmuda@spyrus.com