

Combined 3DES-CBC, HMAC and Replay Prevention Security Transform
<[draft-ietf-ipsec-esp-3des-md5-00.txt](#)>

Status of this Memo

This document is a submission to the IETF Internet Protocol Security (IPSEC) Working Group. Comments are solicited and should be addressed to the working group mailing list (ipsec@tis.com) or to the editor.

This document is an Internet-Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet-Drafts draft documents are 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."

To learn the current status of any Internet-Draft, please check the "l1d-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on [ftp.is.co.za](ftp://ftp.is.co.za) (Africa), [nic.nordu.net](ftp://nic.nordu.net) (Europe), [munniari.oz.au](ftp://munniari.oz.au) (Pacific Rim), [ds.internic.net](ftp://ds.internic.net) (US East Coast), or [ftp.isi.edu](ftp://ftp.isi.edu) (US West Coast).

Distribution of this memo is unlimited.

Abstract

This draft describes a combination of privacy, authentication, integrity and replay prevention into a single packet format.

This document is the result of significant work by several major contributors and the IPsec working group as a whole. These contributors, cited later in this document, provided many of the key technical details summarized in this document. [[IB93](#)] [[IBK93](#)]

Requirements Terminology

In this document, the words that are used to define the significance of each particular requirement are usually capitalised. These words are:

- MUST

This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

- SHOULD

This word or the adjective "RECOMMENDED" means that there might exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before taking a different course.

- MAY

This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor might choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

For the purpose of this RFC, the terms conformance and compliance are synonymous.

1. Discussion

This draft allows a combination of MD5 and 3DES-CBC. In addition to privacy, the goal of this transform is to ensure that the packet is authentic, can not be modified in transit, or replayed.

The claims of privacy, integrity, authentication, and replay prevention are made in this draft. A good general text describing the methods and algorithm are in [[Schneier95](#)].

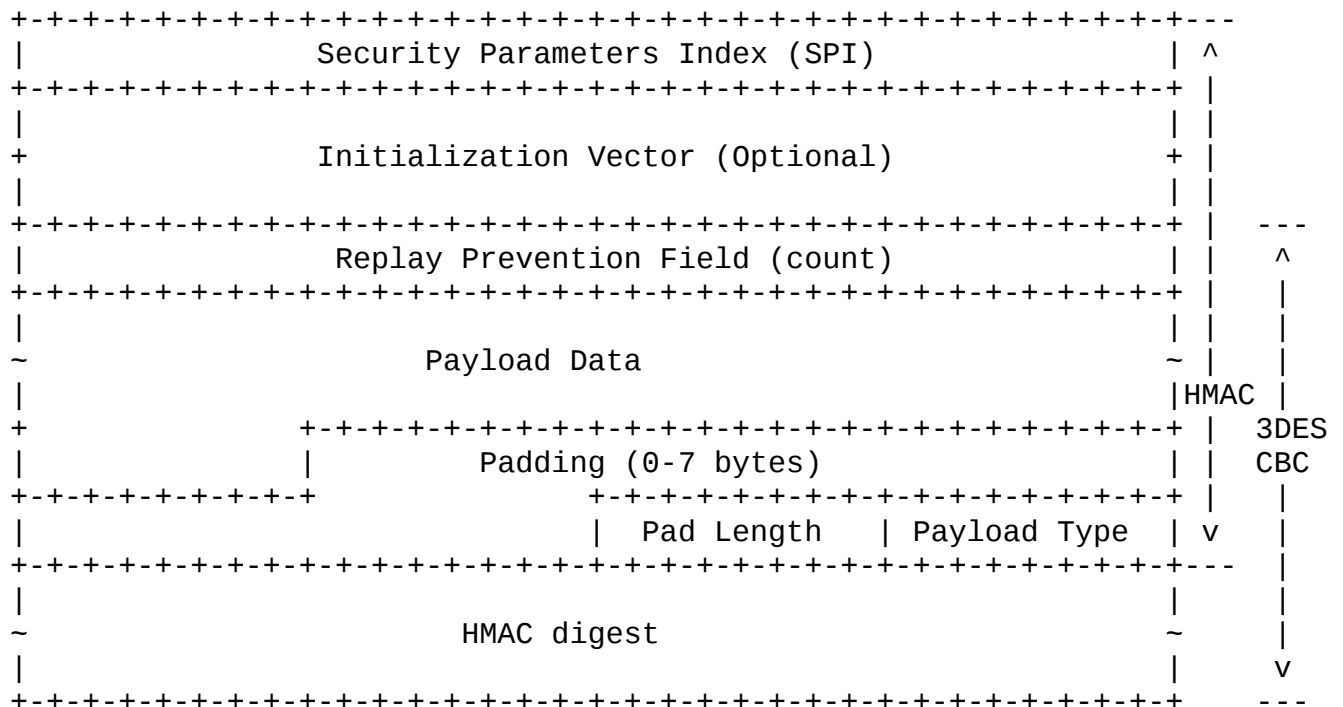
Privacy is provided by 3DES-CBC [[FIPS-46](#)] [[FIPS-46-1](#)] [[FIPS-74](#)] [[FIPS-81](#)].

Integrity is provided by HMAC [[Krawczyk96](#)].

Authentication is provided since only the source and destination know the HMAC key. If the HMAC is correct, it proves that it must have been added by the source.

Replay prevention is provided by the combination of a constantly increasing count, the SPI and the HMAC key. The integrity of the replay field is provided by the HMAC.

2. Packet Format



2.1. Security Parameters Index

This field is negotiated at key setup and MUST not be 0 [[RFC-1825](#)]

2.2. Initialization Vector

The use of an explicit Initialization Vector MAY be negotiated. The purpose of this mode is to support devices that automatically generate IVs and can not operate using a constant IV_key_.

This field is optional and is only used when an explicit IV is negotiated during key exchange. This field contains random data or contains the last cyphertext block of a previous packet sent or received.

For the packet which the explicit IV is received, the explicit IV is used in place of the constant IV_key_ described later in this document.

2.3. Replay Prevention

Replay Prevention is an unsigned 32 bit incrementing counter starting at a mutual agreed upon value (see Key Material) and is enforced to be within a mutually negotiated window size.

The key (K, as described in a later section) MUST be changed frequently enough so that the counter is not allowed to return to the initial value; in other words, the key MUST be changed before 2^{32} packets are transmitted using this key. For a given SPI, counter wrapping MUST be considered to be a replay attack. (While a wrap is a replay attack, there is always the possibility that a packet can get duplicated, so the presence of a single or small number of duplicate packets is not an absolute indication of a replay attack.)

The receiver MUST verify that for a given SPI the packets received have non-repeating (non-duplicate) counter values. This can be implemented as a simple increasing count test or the receiver MAY choose to accept out-of-order packets as long as it is guaranteed that packets can be received only once. For example, an implementation can use a sliding receive window. If such a receive window is supported, the receiver MUST ensure that it will accept packets within the current window only once, and reject any packets it receives with a value that is less than the lower bound of the window.

Negotiated window sizes of 1 and 32 are suggested and larger multiples of 32 are allowed. 1 indicates that only constantly increasing replay numbers are allowed and packets which have replay values less than the highest received are always rejected. 32 indicates that are within 32 of the highest received, and are guaranteed not to have been received before, are allowed.

A window size of 1 MUST be supported. A value of 32 SHOULD be supported.

If a value of 32 is negotiated, then the most recent 32 packets are allowed to arrive out of order. That is, these 32 packets can arrive in any sequence relative to each other except that these packets are guaranteed to arrive only once. [Appendix A](#) has actual code that implement a 32 packet replay window and a test routine. The purpose of this routine is to show how it could be implemented.

2.4. Payload

The payload contains data that is described by the payload type field. This field is an integral number of bytes in length; the following padding and pad length fields will help provide alignment to a four octet boundary.

2.5. Padding

The padding (pad bytes and pad length field) is used to align the following "payload type" field to end on a four octet boundary (when counting from the start of the replay field).

Padding bytes SHOULD be initialized with random data.

At a minimum, the number of pad bytes added MUST be enough to align the payload type field on the next appropriate boundary. However, the sender MAY choose to include additional padding, provided that the alignment is maintained. In total, the sender can add 0-255 bytes of padding.

2.6. Pad Length

The pad length field indicates the number of pad bytes immediately preceding it. The range of valid values is 0-255, where a value of zero indicates that the byte immediately preceding the pad length field is the last byte of the payload.

2.7. Payload Type

Describes what the payload is. The values are described in:

<ftp://ftp.isi.edu/in-notes/iana/assignments/protocol-numbers>

2.8. HMAC Digest

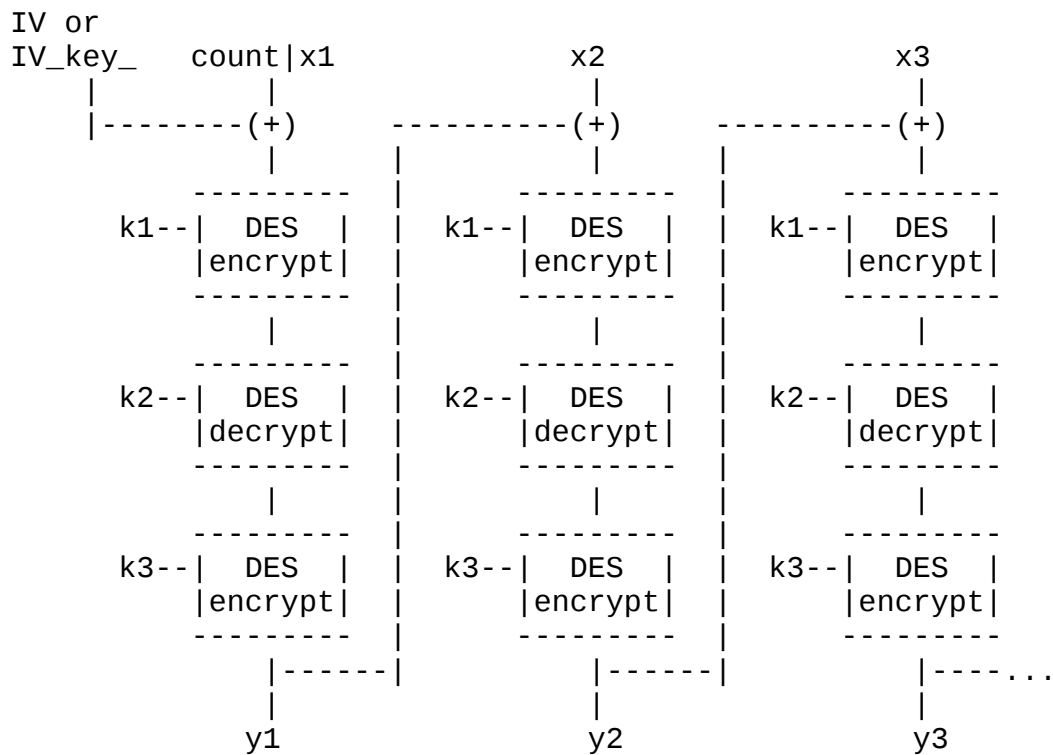
The HMAC digest is a 128 bit residue described in [[Krawczyk96](#)]. This covers the SPI, replay, payload, padding, pad length, payload type.

HMAC is a keyed algorithm, where both directions are keyed separately. The implementation MUST use the HMAC_key_ as described in the section on keys.

3. Encryption Transform Procedure

Triple encryption in Outer-CBC mode with a constant IV_key_ is used (IV_key_I for the initiator -> responder direction and IV_key_R for the responder -> initiator direction). The IV_key_ remains constant for all packets send in this direction.

If an explicit IV is negotiated, 64 bits of random or the last cyphertext block of a previous packet send or receive can be used.



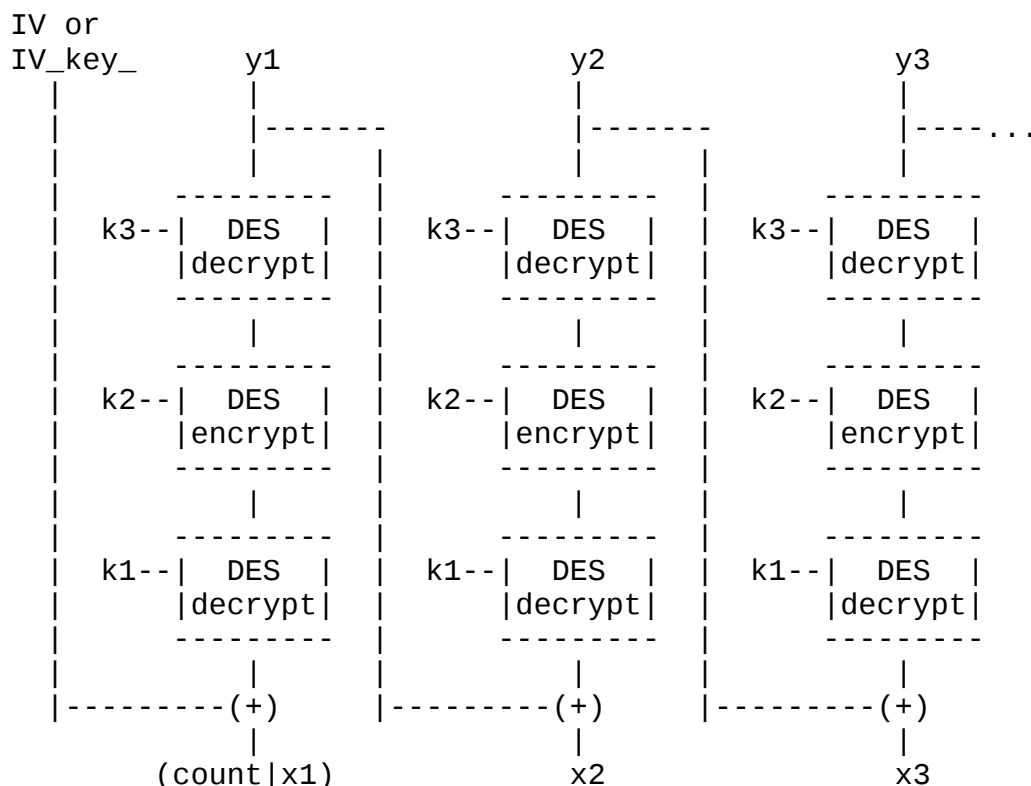
Where count is the Replay counter. x1, x2, x3 are the plaintext (x1 is 32 bits, all others are 64 bits). y1, y2, y3 are the ciphertext. k1, k2, k3 corresponds to DES_KEY_[R|I]1, DES_KEY_[R|I]2, and DES_KEY_[R|I]3.

This transformation is comprised of the following 3 steps.

1. Taking the data and encapsulating it with the SPI, IV (if present), count, pad, pad length, and payload type.
2. Calculating the HMAC using the HMAC_key_ and creating the digest from the SPI, IV (if present), count, data, pad, pad length, and payload type and placing the result into the HMAC digest field.
3. Encrypting the count, data, pad, pad length, payload type, and HMAC digest using 3DES and the appropriate DES_key_ and IV_key_. (Note that the first DES block is a combination of the count and the first word of plaintext.)

4. Decryption Transform Procedure

Triple encryption in Outer-CBC with constant IV_key_ is used. If an IV is present in the packet, then the IV_key_ is not used and is replaced by the IV.



Decryption is comprised of the following 4 steps.

1. (Optional step) Decrypt the first block of data using the appropriate DES_key_ and IV_key_ (or IV) and then do a quick "sanity check" of the count. If the count has decreased below the window or has increased by more than 65k, then it is safe to discard this packet as either a replay, non-authentic or too old. If the count is within 65K, then the probability that the packet is authentic is 65535/65536. (The following replay check and HMAC check are both still required).
2. Decrypt the count (if not already done), data, pad, pad length, and payload type using DES and the appropriate DES_key_ and IV_key_ (or IV).

3. Calculate the HMAC using the HMAC_key_ and create the digest

from the SPI, IV (if present) count, data, pad, pad length, and payload type and checking the result at digest at the end of the packet. If the digest is incorrect, discard the packet.

4. Check the count using the window algorithm discussed above. If the packet is duplicate or too old, discard the packet.

5. Key Material

The key K is provided by the key management layer. This key is used to derive the symmetric keys, they are:

DES_Key_I1, DES_KEY_I2, and DES_KEY_I3 are the keys used for 3DES for traffic from the initiator -> responder. DES_KEY_I1 is the innermost key and DES_KEY_I3 is the outermost key.

DES_Key_R1, DES_KEY_R2, and DES_KEY_R3 are the keys used for 3DES for traffic from the initiator -> responder. DES_KEY_R3 is the innermost key and DES_KEY_R1 is the outermost key.

HMAC_Key_I is the key for the HMAC Algorithm for traffic from the initiator -> responder.

HMAC_Key_R is the key for the HMAC Algorithm for traffic from the responder -> initiator.

IV_key_I is used to stop code book attacks on the first block for traffic from the initiator -> responder.

IV_key_R is used to stop code book attacks on the first block for traffic from the responder -> initiator.

RP_key_I is the initial value and wrap point for the replay prevention field for traffic from the initiator -> responder.

RP_key_R is the initial value and wrap point for the replay prevention field for traffic from the responder -> initiator.

The vertical bar symbol "|" is used to denote concatenation of bit strings.

MD5(x|y) denotes the result of applying the MD5 function to the concatenated bit strings x and y.

Truncate(x,n) denotes the result of truncating x to its first n bits.


```

DES_Key_I2 = Truncate(MD5( 1 | D_PAD_I | K ),64)
DES_Key_I3 = Truncate(MD5( 2 | D_PAD_I | K ),64)
DES_Key_R1 = Truncate(MD5( 0 | D_PAD_R | K ),64)
DES_Key_R2 = Truncate(MD5( 1 | D_PAD_R | K ),64)
DES_Key_R3 = Truncate(MD5( 2 | D_PAD_R | K ),64)
IV_Key_I  = Truncate(MD5( I_Pad_I | K ),64)
IV_Key_R  = Truncate(MD5( I_Pad_R | K ),64)
HMAC_Key_I =          MD5( H_Pad_I | K )
HMAC_Key_R =          MD5( H_Pad_R | K )
RP_Key_I   = Truncate(MD5( R_Pad_I | K ),32)
RP_Key_R   = Truncate(MD5( R_Pad_R | K ),32)

```

where each `_Pad` is 512 bit string.

```

D_Pad_I = 0x5C repeated 63 times.
D_Pad_R = 0x3A repeated 63 times.
I_Pad_I = 0xAC repeated 64 times.
I_Pad_R = 0x55 repeated 64 times.
H_Pad_I = 0x53 repeated 64 times.
H_Pad_R = 0x3C repeated 64 times.
R_Pad_I = 0x35 repeated 64 times.
R_Pad_R = 0xCC repeated 64 times.

```

(Implementation note, The 16 byte intermediate residuals can be pre-calculated from these constants and stored to reduce processing overhead).

6. Security Considerations

The ESP-3DES-HMAC-RP transform described in this draft is immune to the [Bellovin96] attacks. (AH [RFC-1826], in some modes, can also provide immunity to these attack.)

The implications of the size of K can be found in [Blaze96].

7. References

[Bellovin96] Bellovin, S., "Problem Areas for the IP Security Protocols", AT&T Research, <ftp://ftp.research.att.com/dist/smb/badesp.ps>, July, 1996.

[FIPS-46] US National Bureau of Standards, "Data Encryption Standard", Federal Information Processing Standard (FIPS) Publication 46, January 1977.

[FIPS-46-1] US National Bureau of Standards, "Data Encryption Standard", Federal Information Processing Standard (FIPS) Publication 46-1, January 1988.

[FIPS-74] US National Bureau of Standards, "Guidelines for Implementing and Using the Data Encryption Standard", Federal Information Processing Standard (FIPS) Publication 74, April 1981.

[FIPS-81] US National Bureau of Standards, "DES Modes of Operation" Federal Information Processing Standard (FIPS) Publication 81, December 1980.

[Krawczyk96] Krawczyk, H., Bellare, M., Canetti, R., "HMAC-MD5: Keyed-MD5 for Message Authentication", work-in-progress, <http://info.internet.isi.edu:80/in-drafts/files/draft-ietf-ipsec-hmac-md5-00.txt>, March, 1996

[Maughan96] Maughan, D., Schertler, M. Internet Security Association and Key Management Protocol (ISAKMP), work-in-progress, <http://info.internet.isi.edu:80/in-drafts/files/draft-ietf-ipsec-isakmp-04.txt>, February, 1996

[Orman96] Orman, H., "The Oakley Key Determination Protocol", work-in-progress, <http://info.internet.isi.edu:80/in-drafts/files/draft-ietf-ipsec-oakley-00.txt>, February, 1996.

[RFC-1825] Atkinson, R, "Security Architecture for the Internet Protocol", <ftp://ds.internic.net/rfc/rfc1825.txt>, August 1995.

[RFC-1826] Atkinson, R, "IP Authentication Header", <ftp://ds.internic.net/rfc/rfc1826.txt>, August 1995.

[Schneier95] Schneier, B., "Applied Cryptography Second Edition", John Wiley & Sons, New York, NY, 1995. ISBN 0-471-12845-7

[Blaze96] Blaze M., Diffie, W., Rivest, R., Schneier, B., Shimomura, T., Thompson, E., Wiener, M., "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security", <http://theory.lcs.mit.edu/~rivest/bsa-final-report.ascii>, January, 1996

[IB93] John Ioannidis and Matt Blaze, "Architecture and Implementation of Network-layer Security Under Unix", Proceedings of USENIX Security Symposium, Santa Clara, CA, October 1993.

[IBK93] John Ioannidis, Matt Blaze, & Phil Karn, "swIPE: Network-Layer Security for IP", presentation at the Spring 1993 IETF Meeting, Columbus, Ohio.

8. Acknowledgements

This document is based on the document "Combined DES-CBC, HMAC, and Replay Prevention Security Transform," by the IPsec Working Group, J. Naganand, editor [Naganand96]. Much of the text of that document is repeated here, with the details of DES replaced with 3DES. I would like to thank Bob Baldwin, Steve Bellovin, Hugo Krawczyk, Hilarie Orman, and Bill Sommerfeld for their suggestions on key generation for 3DES.

The IPsec working group can be contacted through the chairs:

Ran Atkinson
<rja@cisco.com>
Cisco Systems

Paul Lambert
<PALAMBER@us.oracle.com>
Oracle Corporation

9. Editor's Address

Naganand Doraswamy
<naganand@ftp.com>
Ftp Software, Inc.

Appendix A

This is a routine that implements a 32 packet window. This is intended on being an implementation sample.

```
#include <stdio.h>
#include <stdlib.h>
typedef unsigned long u_long;

enum {
    ReplayWindowSize = 32
};

u_long bitmap = 0;          /* session state - must be 32 bits */
u_long lastSeq = 0;         /* session state */

/* Returns 0 if packet disallowed, 1 if packet permitted */
int ChkReplayWindow(u_long seq);

int ChkReplayWindow(u_long seq) {
    u_long diff;

    if (seq == 0) return 0; /* first == 0 or wrapped */
    if (seq > lastSeq) {    /* new larger sequence number */
        diff = seq - lastSeq;
        if (diff < ReplayWindowSize) { /* In window */
            bitmap = (bitmap << diff) | 1; /* set bit for this packet */
        } else bitmap = 1; /* This packet has a "way larger" */
        lastSeq = seq;
        return 1;          /* larger is good */
    }
    diff = lastSeq - seq;
    if (diff >= ReplayWindowSize) return 0; /* too old or wrapped */
    if (bitmap & (1 << diff)) return 0; /* this packet already seen */
    bitmap |= (1 << diff); /* mark as seen */
    return 1;              /* out of order but good */
}
```

```

char string_buffer[512];
#define STRING_BUFFER_SIZE sizeof(string_buffer)

int main() {
    int result;
    u_long last, current, bits;

    printf("Input initial state (bits in hex, last msgnum):\n");
    if (!fgets(string_buffer, STRING_BUFFER_SIZE, stdin)) exit(0);
    sscanf(string_buffer, "%lx %lu", &bits, &last);
    if (last != 0)
        bits |= 1;
    bitmap = bits;
    lastSeq = last;
    printf("bits:%08lx last:%lu\n", bitmap, lastSeq);
    printf("Input value to test (current):\n");

    while (1) {
        if (!fgets(string_buffer, STRING_BUFFER_SIZE, stdin)) break;
        sscanf(string_buffer, "%lu", &current);
        result = ChkReplayWindow(current);
        printf("%-3s", result ? "OK" : "BAD");
        printf(" bits:%08lx last:%lu\n", bitmap, lastSeq);
    }
    return 0;
}

```