

## Telnet Encryption: CAST-128 64 bit Output Feedback

### Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

### Abstract

This document specifies how to use the CAST-128 encryption algorithm in output feedback mode with the telnet encryption option. Two key sizes are defined: 40 bit and 128 bit.

## 1. Command Names and Codes

### Encryption Type

|                |    |
|----------------|----|
| CAST5_40_OFB64 | 9  |
| CAST128_OFB64  | 11 |

### Suboption Commands

|              |   |
|--------------|---|
| OFB64_IV     | 1 |
| OFB64_IV_OK  | 2 |
| OFB64_IV_BAD | 3 |

## 2. Command Meanings

IAC SB ENCRYPT IS CAST5\_40\_OFB64 OFB64\_IV <initial vector> IAC SE  
IAC SB ENCRYPT IS CAST128\_OFB64 OFB64\_IV <initial vector> IAC SE

The sender of this command generates a random 8 byte initial vector, and sends it to the other side of the connection using the OFB64\_IV command. The initial vector is sent in clear text. Only the side of the connection that is WILL ENCRYPT may send the OFB64\_IV command.

```
IAC SB ENCRYPT REPLY CAST5_40_OFB64 OFB64_IV_OK IAC SE
IAC SB ENCRYPT REPLY CAST128_OFB64 OFB64_IV_OK IAC SE
IAC SB ENCRYPT REPLY CAST5_40_OFB64 OFB64_IV_BAD IAC SE
IAC SB ENCRYPT REPLY CAST128_OFB64 OFB64_IV_BAD IAC SE
```

The sender of these commands either accepts or rejects the initial vector received in a OFB64\_IV command. Only the side of the connection that is DO ENCRYPT may send the OFB64\_IV\_OK and OFB64\_IV\_BAD commands. The OFB64\_IV\_OK command MUST be sent for backwards compatibility with existing implementations; there really isn't any reason why a sender would need to send the OFB64\_IV\_BAD command except in the case of a protocol violation where the IV sent was not of the correct length (i.e., 8 bytes).

### 3. Implementation Rules

Once a OFB64\_IV\_OK command has been received, the WILL ENCRYPT side of the connection should do keyid negotiation using the ENC\_KEYID command. Once the keyid negotiation has successfully identified a common keyid, then START and END commands may be sent by the side of the connection that is WILL ENCRYPT. Data will be encrypted using the CAST128 64 bit Output Feedback algorithm.

If encryption (decryption) is turned off and back on again, and the same keyid is used when re-starting the encryption (decryption), the intervening clear text must not change the state of the encryption (decryption) machine.

If a START command is sent (received) with a different keyid, the encryption (decryption) machine must be re-initialized immediately following the end of the START command with the new key and the initial vector sent (received) in the last OFB64\_IV command.

If a new OFB64\_IV command is sent (received), and encryption (decryption) is enabled, the encryption (decryption) machine must be re-initialized immediately following the end of the OFB64\_IV command with the new initial vector, and the keyid sent (received) in the last START command.

If encryption (decryption) is not enabled when a OFB64\_IV command is sent (received), the encryption (decryption) machine must be re-initialized after the next START command, with the keyid sent (received) in that START command, and the initial vector sent (received) in this OFB64\_IV command.

#### 4. Algorithm

CAST 64 bit Output Feedback

```

key --->+-----+
      +-->| CAST |-->
          | +-----+ |
          +-----+
                      v
INPUT ----->(+ ) -----> DATA

```

Given:

iV: Initial vector, 64 bits (8 bytes) long.

Dn: the nth chunk of 64 bits (8 bytes) of data to encrypt (decrypt).

On: the nth chunk of 64 bits (8 bytes) of encrypted (decrypted) output.

$V_0 = \text{CAST}(iV, \text{key})$

$V(n+1) = \text{CAST}(V_n, \text{key})$

$O_n = D_n \oplus V_n$

#### 5. Integration with the AUTHENTICATION telnet option

As noted in the telnet ENCRYPTION option specifications, a keyid value of zero indicates the default encryption key, as might be derived from the telnet AUTHENTICATION option. If the default encryption key negotiated as a result of the telnet AUTHENTICATION option contains less than 16 (5) bytes, then the CAST128\_OFB64 (CAST5\_40\_OFB64) option must not be offered or used as a valid telnet encryption option.

If there are less than 32 (10) bytes of key data, the first 16 (5) bytes of key data are used as keyid 0 in each direction. If there are at least 32 (10) bytes of key data, the first 16 (5) bytes of key data are used to encrypt the data sent by the telnet client to the telnet server; the second 16 (5) bytes of key data are used to encrypt the data sent by the telnet server to the telnet client.

Any extra key data is used as random data to be sent as an initialization vector.

#### 6. Security Considerations

Encryption using Output Feedback does not ensure data integrity; an active attacker may be able to substitute text, if he can predict the clear-text that was being transmitted.

The tradeoff here is that adding a message authentication code (MAC) will significantly increase the number of bytes needed to send a single character in the telnet protocol, which will impact performance on slow (i.e. dialup) links.

This option was originally drafted back when CPU speeds where not necessarily fast enough to do allow use of CFB. Since then, CPU's have gotten much faster. Given the inherent weaknesses in Output Feedback mode, perhaps it should be deprecated in favor of CFB modes?

Encryption modes using 40-bit keys are not to be considered secure. The 40 bit key mode CAST5\_40\_OFB64 is listed here simply to document the implementations that are already prevalent on the Internet but have never been documented.

## 7. Acknowledgments

This document was based on the "Telnet Encryption: DES 64 bit Output Feedback" document originally written by Dave Borman of Cray Research with the assistance of the IETF Telnet Working Group.

## 8. References

- [1] Adams, C., "The CAST-128 Encryption Algorithm", RFC 2144, May 1997.

## Author's Address

Jeffrey Altman, Editor  
Columbia University  
612 West 115th Street Room 716  
New York NY 10025 USA

Phone: +1 (212) 854-1344  
EMail: [jaltman@columbia.edu](mailto:jaltman@columbia.edu)

## Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

