LOKI - A Cryptographic Primitive for Authentication and Secrecy Applications[1]

Lawrence Brown,
Josef Pieprzyk,
Jennifer Seberry,

Centre for Computer Security Research,
Department of Computer Science,
University College, UNSW,
Australian Defence Force Academy,
Canberra ACT 2600. Australia.

Abstract

This paper provides an overview of the LOKI[2] encryption primitive which may be used to encrypt and decrypt a 64-bit block of data using a 64-bit key. The LOKI primitive may be used in any mode of operation currently defined for ISO DEA-1, with which it is interface compatible [AAA83]. Also described are two modes of operation of the LOKI primitive which compute a 64-bit, and 128-bit, Message Authentication Code (or hash value). These modes of operation may be used to provide authentication of a communications session, or of data files.

1. Introduction

This paper provides an overview of the LOKI[2] encryption primitive which may be used to encrypt and decrypt a 64-bit block of data using a 64-bit key. It has been developed as a result of work analysing the existing DEA-1, with the aim of designing a new family of encryption primitives [Bro89], [BrS90a], [BrS90b], [PiF89], [Pie90], [Pie89], [PiS89]. Its overall structure has a broad resemblance to DEA-1 (see Fig. 1), however the detailed structure has been designed to remove operations which impede analysis or hinder efficient implementation, but which do not add to the cryptographic security of the algorithm. The overall structure and the key schedule has been developed from the work done in [BrS90a] and [BrS90b], whilst the design of the S-boxes was based on [Pie89].

The LOKI primitive may be used in any mode of operation currently defined for ISO DEA-1, with which it is interface compatible [AAA83]. Also described are two modes of operation of the LOKI primitive which compute a 64-bit, and 128-bit, Message Authentication Code (or hash value) respectively, from an arbitrary length of message input. The modes of use are modifications of those described in [DaP89], [Win83], and [QuG90].

[1] this paper was presented at Auscrypt90, in Sydney, Australia, January 1990

[2] LOKI - God of mischief and trickery in Scandinavian mythology. "He is handsome and well made, but of a very fickle mood and most evil disposition. He is of the giant race, but forced himself into the company of the gods, and seems to take pleasure in bringing them into difficulties, and in extracting them out of the danger by his cunning, wit and skill" [Bulfinch's Mythology, Avenel Books, NY 1978].

These modes of operation may be used to provide authentication of a communications session, or of data files.

The LOKI encryption primitive, and the above modes of use have been submitted to the European RIPE project for evaluation [VCF90].

2. The LOKI Cryptographic Primitive

2.1. Overview

The LOKI DEA is a family of ciphers designed to encrypt and decrypt blocks of data consisting of 64 bits, under control of a 64-bit key. This Annex defines a common variant of the algorithm for use when compatibility between implementations is required. The same structure, but with alternate substitution functions may be used to build private variants of this algorithm. The same key is used for both encryption and decryption, but with the schedule of addressing the key bits altered so that the decryption process is the reverse of the encryption process. A block to be encrypted is added modulo 2 to the key, is then processed in 16 rounds of a complex key-dependent computation, and finally is added modulo 2 to the key again. The key-dependent computation can be defined in terms of a confusion-diffusion function f, and a key schedule KS. Descriptions of the encryption operation, the decryption operation, and the definition of the function f, are provided in the following sections. The representation of the keys, key values to be avoided and guidelines for the construction of alternate private ciphers, and full results for the tests conducted to date on LOKI, are described in the Appendices.

2.2. Encryption

The encryption computation is illustrated in Fig 1. The 64 bits of the input block to be encrypted are added modulo 2 to the key, processed in 16 rounds of a complex key-dependent computation, and finally added modulo 2 to the key again.

In detail, the 64-bit input block X is partitioned into two 32-bit blocks XL and XR. Similarly, the 64-bit key is partitioned into two 32-bit blocks KL and KR. Corresponding halves are added together modulo 2, to form the initial left and right halves for the following 16 rounds, thus:

$$L_0 = XL \oplus KL_0 \qquad KL_0 = KL \qquad \text{[Eq.1]}$$

$$R_0 = XR \oplus KR_0 \qquad KR_0 = KR$$

The complex key-dependent computation consists (except for a final interchange of blocks) of 16 rounds (iterations) of a set of operations. Each iteration includes the calculation of the encryption function f. This is a concatenation of a modulo 2 addition and three functions E, S, and P. Function f takes as input the 32-bit right data half R_{i-1} and the 32-bit left key half KL_i produced by the key schedule KS (denoted K_i below), and which produces a 32-bit result which is added modulo 2 to the left data half L_{i-1}. The two data halves are then interchanged (except after the last round). Each round may thus be characterised as:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, KL_i) \qquad \text{[Eq.2]}$$

$$f(R_{i-1,}\ K_i) = P(S(E(R_{i-1} \oplus K_i)))$$

The component functions E, S, and P are described later.

The key schedule KS is responsible for deriving the sub-keys K_i, and is defined as follows: the 64-bit key K is partitioned into two 32-bit halves KL and KR. In each round i, the sub-key K_i is the current left half of the key KL_{i-1}. This half is then rotated 12 bits to the left, and the key halves are interchanged. This may be defined thus:

$$K_i = KL_{i-1}$$

$$KL_i = KR_{i-1} \qquad \text{[Eq.3]}$$

$$KR_i = ROL(KL_{i-1,}\ 12)$$

Finally after the 16 rounds, the other key halves are added modulo 2 to the data halves to form two output block halves YL and YR which are then concatenated together to form the output block Y. This is defined as follows (note the swap of data and key halves to undo the final interchange in [Eq.2] and [Eq.3]):

$$YL = R_{16} \oplus KR_{16}$$

$$YR = L_{16} \oplus KL_{16} \qquad \text{[Eq.4]}$$

$$Y = YL \mid YR$$

2.3. Decryption

The decryption computation is identical to that used for encryption, save that the partial keys used as input to the function f in each round are calculated in reverse order, and the initial and final additions of key to data modulo 2 use the opposite halves of the key (interchange KL_0 and KR_0 in [Eq.1] and KL_{16} and KR_{16} in [Eq.3]). The calculation of the partial keys for decryption consists of first exchanging key halves, then rotating the left half 12 bits to the right, and then using the left half as the partial key. This is defined as:

$$KR_i = KL_{i-1}$$

$$KL_i = ROR(KR_{i-1,}\ 12) \qquad \text{[Eq.5]}$$

$$K_i = KL_i$$

2.4. Function f

The encryption function f is a concatenation of a modulo 2 addition and three functions E, S, and P, which takes as input the 32-bit right data half R_{i-1} and the 32-bit left key half KL_i, and produces a 32-bit result which is added modulo 2 to the left data half L_{i-1}. This is shown in Fig 2, and is defined thus:

$$f(R_{i-1,}\ K_i) = P(S(E(R_{i-1} \oplus K_i))) \qquad \text{[Eq.6]}$$

The modulo 2 addition of the key and data halves ensures that the output of f will be a complex function of both of these values.

The expansion function E takes a 32-bit input and produces a 48-bit output block, composed of four 12-bit blocks which form the inputs to four S-boxes in function f. Function E selects consecutive blocks of twelve bits as inputs to S-boxes S(4), S(3), S(2), and S(1) respectively, as follows:

$[b_3\ b_2 \cdots b_0\ b_{31}\ b_{30} \cdots b_{24}]$
$[b_{27}\ b_{26} \cdots b_{16}]$
$[b_{19}\ b_{18} \cdots b_8]$
$[b_{11}\ b_{10} \cdots b_0]$

This is shown in Table 1 in full.

Table 1 - LOKI Expansion Function *E*

3	2	1	0	31	30	29	28	27	26	25	24
27	26	25	24	23	22	21	20	19	18	17	16
19	18	17	16	15	14	13	12	11	10	9	8
11	10	9	8	7	6	5	4	3	2	1	0

The substitution function S provides the confusion component in the LOKI cipher. It takes a 48-bit input and produces a 32-bit output. It is composed of four S-boxes, each of which takes a 12-bit input and produces an 8-bit output, which are concatenated together to form the 32-bit output of S. The 8-bit output from S(4) becomes the most significant byte (bits [31...24]), then the outputs from S(3) (bits[23...16]), S(2) (bits[15...8]), and S(1) (bits [7...0]). In this Annex, the four S-boxes are identical. The form of each S-box is shown in Fig 3. The 12-bit input is partitioned into two segments: a 4-bit row value r formed from bits $[b_{11}\ b_{10}\ b_1\ b_0]$, and an 8-bit column value c formed from bits $[b_9\ b_8 \cdots b_3\ b_2]$. The row value r is used to select one of 16 S-functions Sfn_r, which then take as input the column value c and produce an 8-bit output value. This is defined as:

$$\mathrm{Sfn}_r = (c \oplus r)^{e_r} \bmod \mathrm{gen}_r, \qquad \text{in } GF(2^8) \qquad \text{[Eq.7]}$$

where gen_r is an irreducible polynomial in $GF(2^8)$, and e_r is the exponent used in forming the rth S-box. The generators and exponents to be used in the 16 S-functions Sfn_r in the standard LOKI are specified in Table 2.

The permutation function P provides diffusion of the outputs from the four S-boxes across the inputs of all S-boxes in the next round. It takes the 32-bit concatenated outputs from the S-boxes, and distributes them over all the inputs for the next round via a regular wire crossing which takes bits from the outputs of each S-box in turn, as defined in Table 3.

2.5. Test Data

A single test triplet for the LOKI primitive is listed below.

```
# Single LOKI Certification triplet
# data is saved as (key, plaintext, ciphertext) triplets
#
5b5a57676a56676e 675a69675e5a6b5a 3c61fa7e2e99d048
```

Table 2 - LOKI S-box Irreducible Polynomials and Exponents

Row	gen_r	e_r
0	375	31
1	379	31
2	391	31
3	395	31
4	397	31
5	415	31
6	419	31
7	425	31
8	433	31
9	445	31
10	451	31
11	463	31
12	471	31
13	477	31
14	487	31
15	499	31

Table 3 - LOKI Permutation P

31	23	15	7	30	22	14	6
29	21	13	5	28	20	12	4
27	19	11	3	26	18	10	2
25	17	9	1	24	16	8	0

LOKI Fig 1 here

LOKI Figs 2, 3 here

3. Additional Modes of Use

The LOKI primitive may also be used in any mode of operation currently defined for ISO DEA-1, with which it is interface compatible [AAA83]. In addition, two modes of use are defined using the LOKI primitive for the purpose of providing message authentication. The Single Block Hash (SBH) mode computes a 64-bit Message Authentication Code (MAC or hash value), from an arbitrary length of message input. The Double Block Hash (DBH) mode computes a 128-bit MAC from an arbitrary length of message input.

In the following definitions, the LOKI primitive used for encryption is denoted $Y = EL_K(X)$. That is, Y is a 64-bit block formed by encrypting input X using the LOKI primitive with key K.

3.1. Single Block Hash (SBH) Mode

The SBH mode is defined as follows. Data for which a hash is to be computed is divided into 64-bit blocks, the final block being padded with nulls if required. A 64-bit key is supplied, and is used as the initial hash value IV. For each message block M_i: that block is added modulo 2 to the previous hash value to form a key. That key is used to encrypt the previous hash value. The encrypted value is added modulo 2 to the previous hash value to form the new hash value (see Fig 4). The SBH code is the final hash value formed. This process may be summarised as:

$$H_0 = IV$$

$$H_i = EL_{M_i \oplus H_{i-1}}(H_{i-1}) \oplus H_{i-1}$$

$$SBH = H_n$$

The SBH mode is a variant of the Davies and Meyer hash function described in [DaP89], [Win83]. The major extension is the addition modulo 2 of the previous hash value to the current message block before using it as key input to the LOKI primitive. This was desired to prevent weak keys being supplied to the primitive when the message data was constant.

If the Initialization Value is chosen not to be a weak key, then the chance of generating a weak key from a given message stream should be greatly reduced.

3.2. Double Block Hash (DBH) Mode

The DBH mode is defined as follows. Data for which a hash is to be computed is divided into pairs of 64-bit blocks M_{2i+1}, M_{2i+2}, the final block being padded with nulls if required. A 128-bit key is supplied, composed of two 64-bit blocks, which are used as the initial hash values IV_{-1}, IV_0.

$$H_{-1} = IV_{-1}$$

$$H_0 = IV_0$$

For each pair of message blocks M_{2i+1} M_{2i+2}, the following calculation is performed (see Fig 5):

$$T = EL_{M_{2i+1} \oplus H_{2i-1}}(H_{2i-1} \oplus M_{2i+2}) \oplus M_{2i+2} \oplus H_{2i}$$

$$H_{2i+1} = EL_{M_{2i+2} \oplus H_{2i}}(T \oplus M_{2i+1}) \oplus M_{2i+1} \oplus H_{2i-1} \oplus H_{2i}$$

$$H_{2i+2} = T \oplus H_{2i-1}$$

The DBH block is formed by concatenating the final two hash values as follows:

$$DBH = H_{n-1} \mid H_n$$

The DBH mode is derived from that proposed by Quisquater and Girault [QuG90]. Again it was extended by the addition modulo 2 of the previous hash value to the current message block before using it as key input to the LOKI primitive.

LOKI Figs 4, 5 here

4. Conclusion

The LOKI cryptographic primitive, and its associated modes of use for message authentication have been described. This algorithm is currently undergoing evaluation and testing by several parties.

Acknowledgements

To the members of the Centre for Computer Security Research, and the staff of the Department of Computer Science for their help and suggestions. Thankyou.

Bibliography

[AAA83] "*Information Interchange - Data Encryption Algorithm - Modes of Operation*," American National Standards Institute X3.106-1983, American National Standards Institute, New York, 1983.

[Bro89] L. Brown, "A Proposed Design for an Extended DES," in *Computer Security in the Age of Information*, W. J. Caelli (editor), North-Holland, Amsterdam, 1989.

[BrS90a] L. Brown and J. Seberry, "On the Design of Permutation P in DES Type Cryptosystems," in *Advances in Cryptology - Eurocrypt'89*, Lecture Notes in Computer Science, no. 434, J. J. Quisquater and J. Vanderwalle (editors), pp. 696-705, |Springer Verlag|, Berlin, 1990.

[BrS90b] L. Brown and J. Seberry, "Key Scheduling in DES Type Cryptosystems," in *Advances in Cryptology: Auscrypt '90*, Lecture Notes in Computer Science, no. 453, pp. 221-228, |Springer Verlag|, Berlin, 1990.

[DaP89] D. W. Davies and W. L. Price, *Security for Computer Networks*, John Wiley and Sons, New York, 1989. (2nd edn).

[Mey78] C. H. Meyer, "Ciphertext/plaintext and ciphertext/key dependence vs number of rounds for the data encryption standard," in *AFIPS |Conf.| |Proc.| 47*, pp. 1119-1126, AFIPS Press, Montvale NJ, USA, |June| 1978.

[MeM82] C. H. Meyer and S. M. Matyas, *Cryptography: A New Dimension in Data Security*, |John Wiley & Sons, New York, 1982.

[PiF89] J. Pieprzyk and G. Finkelstein, "Permutations that Maximize Non-Linearity and Their Cryptographic Significance," in *Computer Security in the Age of Information*, W. J. Caelli (editor), North-Holland, Amsterdam, 1989.

[Pie89] J. Pieprzyk, "Error Propagation Property and Application in Cryptography," *IEE Proceedings-E, Computers and Digital Techniques*, vol. 136, no. 4, pp. 262-270, |July| 1989.

[PiS89] J. Pieprzyk and J. Seberry, "*Remarks on Extension of DES - Which Way to Go?*," Tech. Rep. CS89/4, |Dept. of Computer Science, UC UNSW, Australian Defence Force Academy|, Canberra, Australia, |February.| 1989.

[Pie90] J. Pieprzyk, "Non-Linearity of Exponent Permutations," in *Advances in Cryptology - Eurocrypt'89*, Lecture Notes in Computer Science, no. 434, J. J. Quisquater and J. Vanderwalle (editors), pp. 80-92, |Springer Verlag|, Berlin, 1990.

[QuG90] J. Quisquater and M. Girault, "2n-Bit Hash Functions Using n-Bit Symmetric Block Cipher Algorithms," in *Advances in Cryptology - Eurocrypt'89*, Lecture Notes in Computer Science, no. 434, J. J. Quisquater and J. Vanderwalle (editors), pp. 102-109, |Springer Verlag|, Berlin, 1990.

[VCF90] J. Vandewalle, D. Chaum, W. Fumy, C. Janssen, P. Landrock and G. Roelofsen, "A European Call for Cryptographic Algorithms: RIPE RACE Integrity Primitives Evaluation," in *Advances in Cryptology - Eurocrypt'89*, Lecture Notes in Computer Science, no. 434, J. J. Quisquater and J. Vanderwalle (editors), pp. 267-271, |Springer Verlag|, Berlin, 1990.

[Win83] R. S. Winternitz, "Producing a One-Way Hash Function from DES," in *Advances in Cryptology - |Proc.| of Crypto 83*, D. Chaum, R. L. Rivest and A. T. Sherman (editors), pp. 203-207, Plenum Press, New York, |August.| 22-24, 1983.

Appendix 1 - Key Representation and Choice

INTRODUCTION

LOKI keys are 64-bit blocks, numbered as specified in section 2. These keys may be written in hexadecimal thus: hhhhhhhhhhhhhhhh, where h is one hex (4-bit) digit. All 64 bits of the key are used in the LOKI algorithm and contribute to the confusion-diffusion process. There is no concept of parity bits in the key. Valid keys may thus cover the range 0000000000000000_{16} to $\mathrm{ffffffffffffffff}_{16}$.

CHOICE OF KEYS

The cryptographic strength of the LOKI algorithm is greatly reduced if only a small number of internal sub-keys are generated. Thus keys which produce such sub-keys should be avoided.

Weak

are those which result in only a single sub-key being formed on all 16 rounds. These keys thus form their own decryption key, and have the form: $\mathrm{hhhhhhhhhhhhhhhh}_{16}$, $h \; \varepsilon \; [0..f]$. There are 16 such keys.

Semi-Weak

are those which result in two sub-keys being formed on alternate rounds. These keys thus form mutual pairs, each being the decryption key for the other, and have the form: $\mathrm{hhhhhhhhiiiiiiii}_{16}$, $h, i \; \varepsilon \; [0..f]$, $h \neq i$. There are 240 such keys.

Demi-Semi-Weak

are those which result in four sub-keys being formed on successive rounds. It is not known whether these form a security risk, but it is generally accepted that they should be avoided. They have the form: $\mathrm{hihihihijkjkjkjk}_{16}$, $h, i, j, k \; \varepsilon \; [0..f]$, $h \neq i \neq j \neq k$. There are 65280 such keys.

CONCLUSION

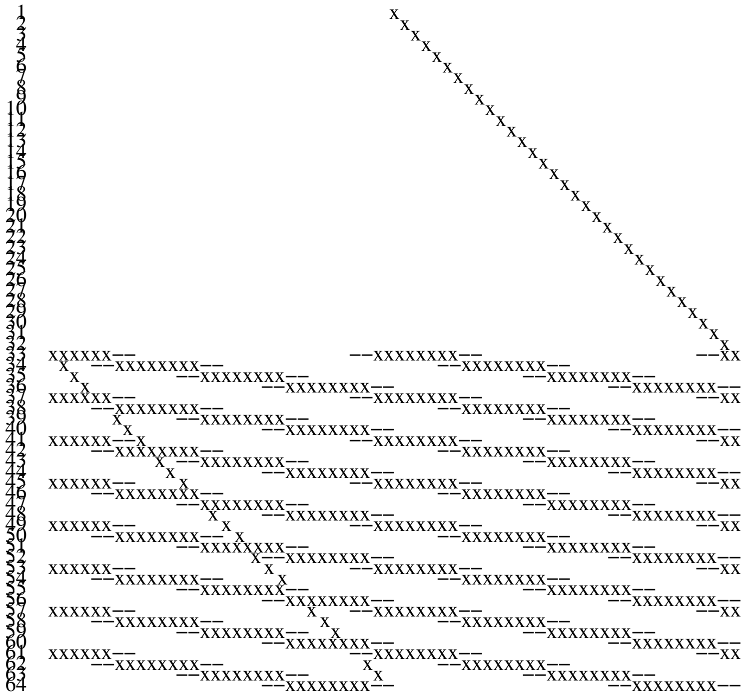
In brief, the keys to be avoided may be described as all keys of the form: $\mathrm{hihihihijkjkjkjk}_{16}$, $h, i, j, k \; \varepsilon \; [0..f]$; that is where both the first four bytes are identical, and the second four bytes are also identical, and thus are very easy to test for and exclude. There are a total of 65536 (ie 2^{16}) keys to be avoided out of a total key space of 2^{64}, a very small fraction of the available number of keys.

Appendix 2 - Dependency Analysis for the LOKI Primitive

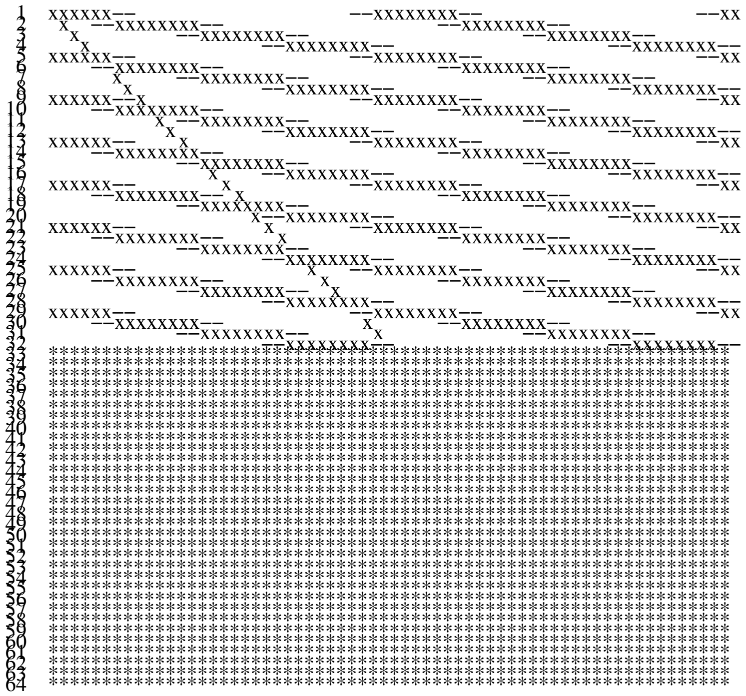
To provide a measure of the effectiveness of the derived permutations, Meyer's analysis [Mey78], [MeM82] of ciphertext dependence on key bits (CKdep) and plaintext bits (CPdep) was used in the design of the overall structure of LOKI. Briefly, following Meyer, this analysis may be described as follows. To provide a measure of the dependency of ciphertext bits on plaintext bits, a 64 * 64 array $G_{a,b}$ is formed. Each element $G_{a,b}(i, j)$ specifies a dependency of output bit $X(j)$ on input bit $X(i)$, between rounds a and b. The number of marked elements in $G_{0,r}$ indicates the degree to which complete dependence was achieved by round r. Similarly, the dependency of ciphertext bits on key bits is measured by forming a 64 * 64 array $F_{a,b}$, each element of which specifies a dependency of output bit $X(j)$ on key bit $K(i)$. Again, the number of marked elements in $F_{0,r}$ will be examined to provide a profile of the degree of dependence achieved by round r. Details of the derivation of these matrices, and the means by which entries are propagated, may be found in [MeM82]. The matrices found for the LOKI primitive are listed below (x specifies message dependency, - specifies autoclave dependency, * specifies dependencies via both message and autoclave inputs):

LOKI CKdep Analysis

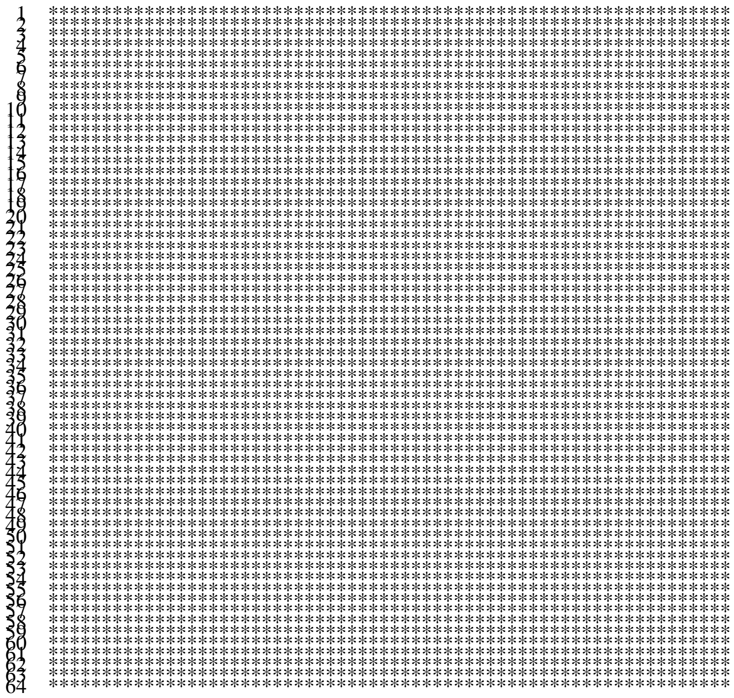
Round 1: None 3276, Msg 564, Autoclave 256, Both 0, Err 0 CKdep: 0.00,20.02



Round 2: None 1260, Msg 532, Autoclave 256, Both 2048, Err 0 CKdep: 50.00,69.24

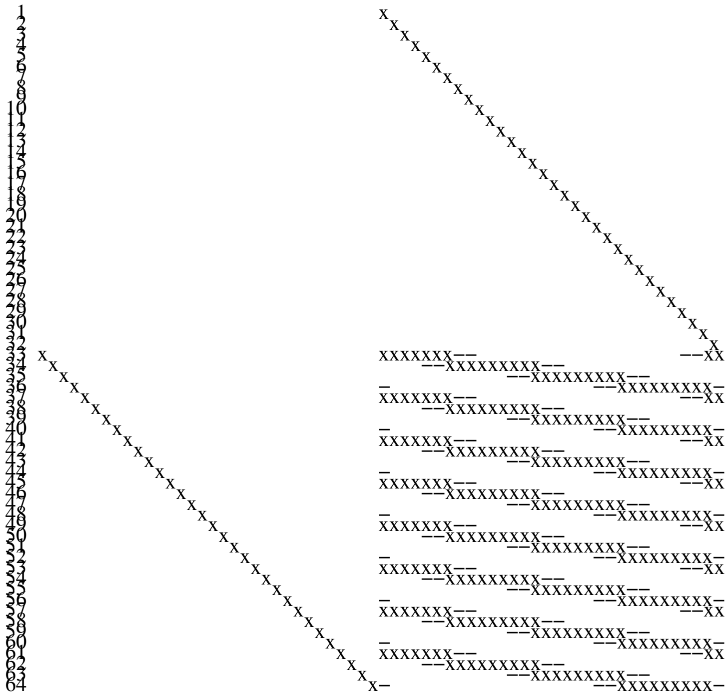


Round 3: None 0, Msg 0, Autoclave 0, Both 4096, Err 0 CKdep: 100.00,100.00

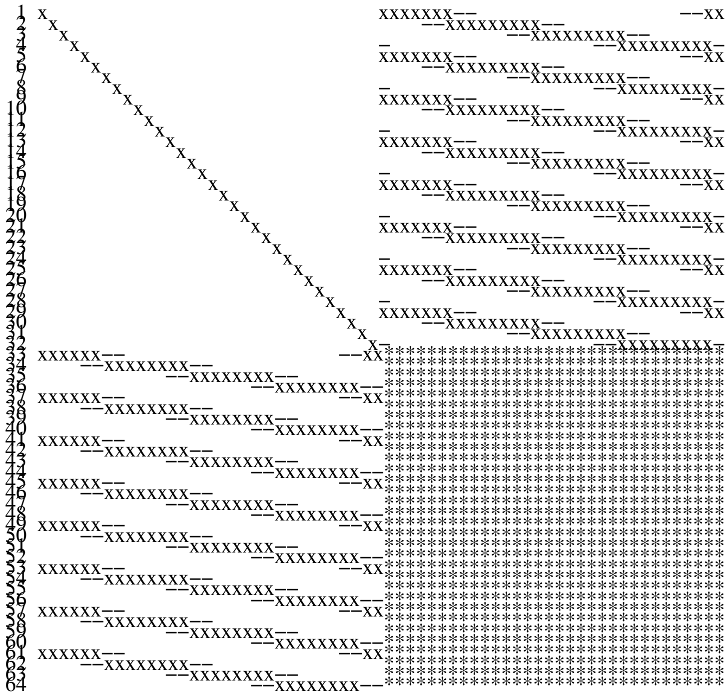


LOKI CPdep Analysis

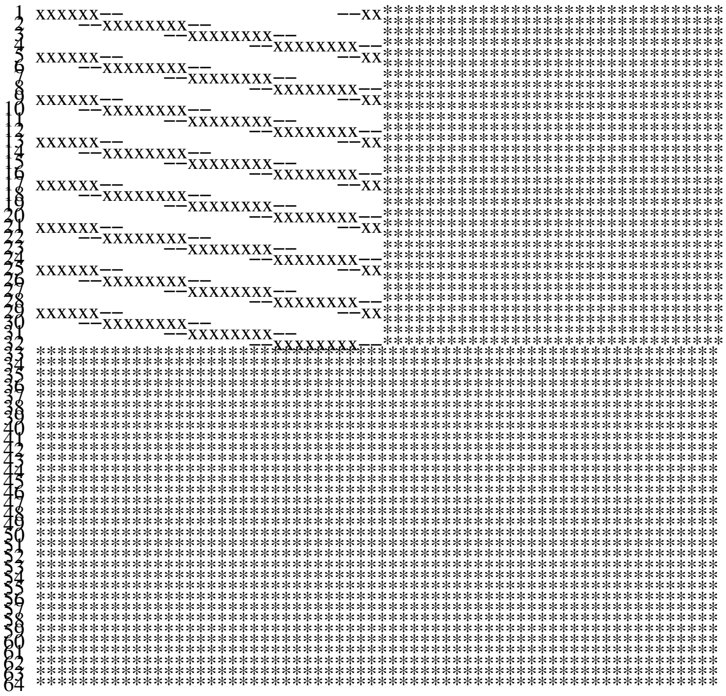
Round 1: None 3616, Msg 352, Autoclave 128, Both 0, Err 0 CPdep: 0.00,11.72



Round 2: None 2240, Msg 576, Autoclave 256, Both 1024, Err 0 CPdep: 25.00,45.31



Round 3: None 640, Msg 256, Autoclave 128, Both 3072, Err 0 CPdep: 75.00,84.38



Round 4: None 0, Msg 0, Autoclave 0, Both 4096, Err 0 CPdep: 100.00,100.00