Fast Block Cipher Proposal

Burton S. Kaliski Jr. and M.J.B. Robshaw
RSA Laboratories
100 Marine Parkway
Redwood City, CA 94065

1 Introduction

In what follows we present the basic features of a proposed block cipher that is fast to implement in software. In consideration of the uncharacteristically large block size, it might best be viewed as a presentation of ideas rather than as an immediately applicable cipher.

Whilst the present proposal provides an encryption rate of up to 500 Kbyte/s in software, we feel that provided no inherent weakness in the design philosophy is uncovered, higher encryption rates may well be possible. A particularly nice feature is the possibility of extensive parallelisation, which would make the basic approach very suitable for hardware implementation.

The main motivation, however, is for high-speed performance in software. As a consequence we show little concern for the amount of memory that might be required in an implementation, provided of course that it remains reasonable. We have decided to concentrate on the following three features - increasing the block size, using a modular method of construction and integrating current cryptographic techniques.

Regarding the block size we wish to implement, we have decided to push this to an extreme degree. If the same processing time is required to provide an acceptable level of security for a 64-bit block cipher as for a 128-bit block cipher, perhaps because they both use the same structure, then we have effectively doubled the speed of the block cipher. It is not hard to imagine a 32-bit version of DES [7] that replaces every bit of DES with a 32-bit word, and hence has a larger block size and huge S-boxes.

We note that this was also the motivation behind the DES variant G-DES proposed by Schaumuller-Bichl [12]. Though the security of G-DES has been seriously undermined [1] the basic principle it embodies in attempting to get better performance is not in question.

Additionally, the move to larger block sizes might allow us greater flexibility in the use of computer architecture that is designed to efficiently manipulate 'words' of 32 bits (rather than individual bits themselves); this in itself could lead to an improved level of performance.

Finally, the choice of a particularly large block size is also motivational. We believe it provides an opportunity to move beyond calls for changes to DES-like structures, and to get away from the ideas of iterated block ciphers.

We feel that the proposal we present provides a good structure with some nice features on which to hang some carefully assessed components. For this reason we like to consider the design as a modular one - certain parts can be replaced

as seems fit, others can be extended, and there is a great deal of flexibility in arriving at a final cipher.

2 Proposal

The main approach has been to incorporate techniques already used in hash functions. It is well known that certain hash functions are designed around the use of DES in an attempt to exploit the security offered by DES [9]. In a similar manner, it is possible to modify some of the dedicated hash functions, which are designed to be particularly fast, into block ciphers.

It is well known that the MD5 hash function [11] can be adapted in an obvious way to allow decryption to take place. By removing the feedforward which surrounds each iteration, what remains is reversible. We could then encrypt plaintext blocks of 128 bits using a 512-bit key. Such a cipher would run at $\frac{1}{4}$ the speed of MD5 giving a software performance of around 250 Kbyte/s.

Instead of using the MD5 hash function as a whole, we take parts of the algorithm that provide us with some of the properties we would like a block cipher to possess. In doing this we forego the opportunity to relate the security of the block cipher to the security of the hash function in any obvious manner, but we can perhaps get a block cipher which performs at great speed by using techniques we have seen successfully applied elsewhere.

2.1 Overview

The block cipher we propose operates on unconventionally large plaintext blocks, namely blocks of 256 words, where a word consists of 32 bits. The cipher uses a pseudo-random permutation $\mathcal{P}$ on the numbers $0, \ldots, 255$ and 2048 32-bit words X_i. These should be generated using the secret key K.

There are four rounds and each round uses a different function F_r which takes as input four words of plaintext and some key-dependent information. While it may be potentially confusing at first, we feel the best way to refer to the rounds during a written description is as first, second, third and fourth, but the best way to index them within formulae is by $r = 0, \ldots, 3$.

The question of the best choices for F_r is, of course, open. We have decided to use ones closely based on those used in MD5 because their use is familiar and there is some feel for the security they might offer. The size of the key K is again open to debate, the current fashion seems to be for 80-bit keys which we shall use for the sake of a concrete details in section 2.3. The encryption procedure can be described as follows. $X <<< b$ denotes a left rotation of X by b bits, this rotation might be performed on 8-bit or 32-bit quantities, the context should make this clear.

1. Obtain key K
2. Generate the random permutations $\mathcal{P}$ and words X_i, $0 \leq i \leq 2047$, using the key K

3. Write the plaintext as $P[\cdot] = P[0] \ldots P[255]$ where $P[i]$ is a word of 32-bits. Permute the 256 words of plaintext according to the key dependent permutation $\mathcal{P}$ to give $P'[0] \ldots P'[255]$
4. For round $r = 0$ to 3 do {
   For group $g = 0$ to 63 do {
   Initialize buffer words A, B, C, D as

$$A = P'[(4 * g) <<< 2r] \ , B = P'[(4 * g + 1) <<< 2r],$$
$$C = P'[(4 * g + 2) <<< 2r] \ , D = P'[(4 * g + 3) <<< 2r]$$

   For step $s = 0$ to 7 do {
   $A = A \oplus F_r(B, C, D, X_i)$ for some X_i where $i = 512r + 8g + s$
   $A = D$, $B = (A <<< 5)$, $C = B$ and $D = C$
   }
   Modify the plaintext by

$$P'[(4 * g) <<< 2r] = A \ , P'[(4 * g + 1) <<< 2r] = B,$$
$$P'[(4 * g + 2) <<< 2r] = C \ , P'[(4 * g + 3) <<< 2r] = D$$

   }
}
5. Output $P'[0] \ldots P'[255]$ as ciphertext.

The details of decryption, the reverse operation to encryption, can be easily established.

2.2 The round functions

A function F_r is applied 8 times in the processing of each of 64 independent groups in each round. The round functions we specify were chosen arbitrarily, but are loosely based on the round functions used in the hash function MD5 with some account taken of results on its cryptanalysis [2].

The functions F_r use a Boolean function f_r which differs from round to round. Otherwise the F_r are identical and can be described as follows. Addition is performed modulo 2^{32}.

$$\begin{aligned} A &= A \oplus F_r(B, C, D, X_i) \\ &= A \ominus (B + f_r(B, C, D) + X_i) \end{aligned}$$

The Boolean functions f_r are defined as follows and are those used in the MD5 hash function. The notation should be familiar and the operations are performed bitwise.

$$\begin{aligned} f_0(B, C, D) &= (B \wedge C) \vee (\neg B \wedge D) \\ f_1(B, C, D) &= (B \wedge D) \vee (C \wedge \neg D) \\ f_2(B, C, D) &= B \oplus C \oplus D \\ f_3(B, C, D) &= C \oplus (B \vee \neg D) \end{aligned}$$

2.3 Initialization

The initial permutation $\mathcal{P}$ (represented as the table $S[0], \ldots, S[255]$) could be generated using a key K by variations on techniques presented in Knuth [3]; one variation is presented below. Note that all quantities enclosed in square brackets are to be considered modulo 256.

1. Denote the 80-bit key K by $k[0] \ldots k[9]$
2. Expand the key to 256 bytes $k[0] \ldots k[255]$ by using the following recurrence:
   $k[a+10] = k[a+8] \oplus k[a+4] \oplus k[a+3] \oplus k[a]$
3. For $i = 0$ to 255 do
   $S[i] = i$
4. $m = 0$
5. For $pass = 0$ to 1 do {
   For $i = 256$ to 1 step -1 do {
   $m = (k[256-i] + k[257-i]) \bmod i$
   $k[257-i] = (k[257-i] <<< 3)$
   Swap $S[m]$ and $S[i-1]$
   }
   }

The words X_i, $0 \leq i \leq 2047$, might be derived from a second pseudo-random permutation of $0 \ldots 255$; represented as $T_0 \ldots T_{255}$, generated using the key K in byte-wise reverse order $k[9] \ldots k[0]$. Define the words used in the first round by X_i, $0 \leq i \leq 255$ where X_i is the concatenation of $T_i, \ldots, T_{i+3}$ and X_i, $256 \leq i \leq 511$ is the concatenation of T_i, T_{i+2}, T_{i+4} and T_{i+6}, addition in the subscripts being performed modulo 256. For subsequent rounds define the X_i as follows. For $512 \leq i \leq 1023$, $X_i = (X_{i-512}) <<< 3$; for $1024 \leq i \leq 1535$, $X_i = (X_{i-1024}) <<< 7$; and for $1536 \leq i \leq 2047$, $X_i = (X_{i-1536}) <<< 19$.

The details that we have provided for the initialization process should again be viewed as motivational; there may well be alternative schemes which are both more efficient and offer improved security.

3 Comments and performance estimates

At the beginning of the first round, the plaintext words are moved to 64 groups of four in a pseudo-random manner using the permutation $\mathcal{P}$.

Within each group the words are processed in such a way that all four words at the end of the round are dependent on each of the input words and some key-dependent information. In subsequent rounds we tie the groups together using a method motivated by techniques used in the calculation of the Discrete Fourier Transform [8].

In round one the first group uses words $P'[0]$, $P'[1]$, $P'[2]$ and $P'[3]$. In round two, the first group takes the modified words $P'[0]$, $P'[4]$, $P'[8]$ and $P'[12]$. The first group of the third round takes modified $P'[0]$, $P'[16]$, $P'[32]$ and $P'[48]$ and the final round uses the recently modified $P'[0]$, $P'[64]$, $P'[128]$ and $P'[192]$.

This is generalized to the other groups and at the end of the fourth round a complete diffusive effect has been obtained whereby each output word depends on each of the 256 plaintext words of the input. This effect is easily achieved by using bit rotations of the variable used to index the permuted plaintext words. It can be generalized as follows where $r = 0, \ldots, 3$ denotes the round.

$$A = P'[(4 * g) <<< 2r], \; B = P'[(4 * g + 1) <<< 2r],$$
$$C = P'[(4 * g + 2) <<< 2r], \; D = P'[(4 * g + 3) <<< 2r]$$

Four rounds are sufficient, and are the minimum necessary in this design, to achieve a complete diffusion of 256 words of plaintext information within the resultant ciphertext. Preliminary empirical testing suggests that a very good avalanche effect is obtained using this network together with the proposed round functions. Alternative approaches would include using three rounds with 64 plaintext words or two rounds with 16 plaintext words, each of which would offer an increasing improvement in the speed of the cipher. We have, however, been more conservative and we feel that less than four rounds could lead to a compromise in the security. Clearly, extensive message padding might be required for the encryption of some messages and the details of a suitable padding scheme have yet to be assessed.

In a software implementation, each round consists of processing 64 groups, where the processing requires eight steps. Each step requires about 6 to 8 operations and so we see that an implementation may run very quickly. The indexing of $P'[\cdot]$ is easily optimized. The time to initialize the permutations $\mathcal{P}$ and the words X_i is a fixed overhead and dependent on the methods used. This overhead becomes proportionally less significant the larger the message. It may well be possible to reduce the number of key dependent words X_i without compromising security thus reducing the time required for the pre-computation of the key dependent material.

Each round is effectively 64 applications of $\frac{1}{2}$ of an MD5 hashing round (each round in MD5 has 16 steps). Interestingly, the buffer words in the hash function are replaced by the plaintext and its modifications in the block cipher, whilst the message in the hash function becomes the key-dependent material in the block cipher.

There are 4 rounds in the block cipher, as in MD5, and so we estimate that 256×32 bits are encrypted using the equivalent of $64 \times \frac{1}{2}$ MD5 hashes of 512 bits. This gives a rough estimate for the encryption rate of around 500 Kbyte/s, half the speed of MD5.

We draw attention to the fact that the processing of each of the 64 groups in each round is independent. This provides a great opportunity for parallelisation in hardware, something which would immediately yield a 64-fold improvement in speed.

4 Security and future research

We have attempted to get away from the ideas of iterated ciphers, ciphers for which a weak round function is repeated sufficiently often to resist cryptanalytic attack. Differential [1] and linear [4] cryptanalysis rely on statistical advantages, identified by the analysis of a few rounds, still being of value in the analysis of the complete cipher.

We hinder this approach by using a different complex round function in each of the four rounds. We hope that with a good choice of round functions we can make such attacks infeasible. We take some comfort for this view from the history of cryptanalytic attacks on hash functions. Looking at MD4 [10] as a particularly good example, we see reduced round versions become vulnerable to attack, but it is the combination of rounds, with different round functions, that provides the overall security. We feel that a poor choice of round functions will almost inevitably result in a poor cipher. By this we mean that the structure of the cipher which provides the diffusion adds little complexity to the analysis of the cipher and is unlikely to make up for any deficiency in the round functions. This is perhaps the opposite situation to that of an iterated block cipher where the round functions are known to be weak but the mixing is so thorough during encryption that any weakness cannot easily be exploited.

If the round functions we have chosen prove to be inadequate, or to be vulnerable to certain cryptanalytic attacks, then the modular design allows the whole round function to be replaced. This again marks a shift from iterated ciphers whereby increased security is often obtained by adding more rounds. Instead we keep the structure of the cipher fixed and security can be improved by adjusting the round function itself.

The aim of defining the X_i in the manner we did (section 2.3) is to prevent easily identified partial dependencies building up between rounds. The expansion of the key material from 10 to 256 bytes in the generation of the tables was motivated by techniques used in the Secure Hash Algorithm [6]; some initial statistical testing confirms that the table initialization procedure appears to perform quite well.

Although the choice of functions and structure owes a lot to MD5, it is difficult to equate the security of the block cipher with the security of the motivating hash function. Analysis of the relation between block ciphers and hash functions would be a valuable research effort.

Finally, an alternative way of viewing our proposal is as the concantenation of a series of encryptions using several strong encrypting functions each of which comprises a round of the cipher. It would be particularly useful to equate the security of the overall cipher to the security of the constituent round functions. In this way it might be possible to obtain results analogous to those obtained by Maurer and Massey on encryption using several unrelated ciphers [5].

5 Summary

We have suggested a block cipher design that includes two major features.

The first is a push towards large block sizes. This allows the design of sophisticated functions which take advantage of 32-bit architectures, and perhaps, in the future, even 64-bit architectures. We feel that a great improvement in the speed performance of block ciphers can be gained through such techniques. Additionally, using some of the established features of trusted dedicated hash functions might well be used in the design of faster block ciphers. The commonly trusted dedicated hash functions are often based on 32-bit operations and this provides a great opportunity to fulfil the first aim by using techniques we have seen successfully applied elsewhere.

The second issue we have tried to address is that of a move away from iterated block ciphers. We have tried to obtain comparable security by using what we shall call incompatible round functions. In this way we hope that fewer rounds might be necessary to provide a good cipher, thus giving better performance. Additionally, the use of the proposed network for combining these rounds allows a larger block size and hence a higher bit-rate, together with the opportunity for extensive parallelisation.

The cipher presented here is a first attempt to mesh these ideas together and due to the very limited time available we have been unable to provide a full evaluation of the potential security. The use of a non-iterative cipher with few rounds will undoubtedly open new methods of cryptanalysis, presumably including many that have previously been used on hash functions [9]. It then becomes an open challenge to replace any components of the above proposal that are discovered to be weak with others that might have a better future.

References

1. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.
2. B. den Boer and A. Bosselaers. Collisions for the compression function of MD5. In *Advances in Cryptology — Eurocrypt '93*, 1993. To appear.
3. D.E. Knuth. *The Art of Computer Programming.* Volume 2, Addison-Wesley, Reading, Mass., 2nd edition, 1981.
4. M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology — Eurocrypt '93*, 1993. To appear.
5. U.M. Maurer and J.L. Massey. Cascade ciphers: the importance of being first. In *Proc. IEEE Symposium on Information Theory*, 1990. January 14–19, San Diego, CA.
6. National Institute of Standards and Technology (NIST). *FIPS Publication 180: Secure Hash Standard (SHS).* May 11, 1993.
7. National Institute of Standards and Technology (NIST). *FIPS Publication 46-1: Data Encryption Standard.* January 22, 1988. Originally issued by National Bureau of Standards.
8. A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing.* Prentice-Hall, Inc., New Jersey, 1975.
9. B. Preneel. *Analysis and design of cryptographic hash functions.* PhD thesis, Katholieke Universiteit Leuven, 1993.

10. R.L Rivest. The MD4 message digest algorithm. In *Advances in Cryptology Crypto '90*, pages 303–311, Springer-Verlag, New York, 1991.
11. R.L. Rivest. *RFC 1321: The MD5 Message-Digest Algorithm.* Internet Activities Board, April 1992.
12. I. Schaumuller-Bichl. *On the design and analysis of new cipher systems related to the DES.* Technical Report, Linz University, 1983.