Akelarre: a new Block Cipher Algorithm

Gonzalo Álvarez, Dolores de la Guía, Fausto Montoya y Alberto Peinado

Departamento de Tratamiento de la Información y Codificación.
Instituto de Física Aplicada, Consejo Superior de Investigaciones Científicas.
Serrano 144, 28006 Madrid.
Tel: +34 (1) 5631284, Fax: +34 (1) 5631371, E-mail: fausto@iec.csic.es

Abstract: *This document presents the operation of the high speed encryption algorithm Akelarre. Akelarre is a secret-key iterated block cipher of great flexibility in its security level, allowing the modification via software of such parameters as the number of rounds and the key length. Presumably, it is cryptographically secure, due to the heavy use of data dependent rotations and the mixing of arithmetic operations from different algebraic groups. The encryption and decryption algorithms are identical and of easy implementation in both hardware and software.*

1. Introduction

Akelarre is a secret key fast block cipher, suitable for hardware or software implementations, the algorithm main operation is the data dependent cyclic rotation of prime-length registers, combined with bit-wise XOR and two's complement addition. The architecture is specially designed to ensure with a 100% probability that *all* the plaintext will always determine *at least one* rotation. The use of prime-length registers avoid invariant rotations (output identical to input), which could appear if composite-length registers were used.

These operations belongs to different algebraic groups, so that they are incompatible in the sense that no pair of the three operations satisfies a distributive law or an associative law. The operations are so arranged that the output of an operation of one type is never used as the input to an operation of the same type. In this way, any relationship between the plaintext, the ciphertext and the key is hidden and the relationship statistics is very complicated.

Akelarre is an iterated *r* round block cipher. The output of each iteration, or encryption round, is a complicated function of the output of the previous rounds and several sub-blocks derived from the user's secret key. The user may supply a key of variable length in 64 bits

increments, while a key expansion algorithm generates the various 32 bit sub-keys needed be used in the different encryption rounds.

Akelarre's great flexibility relies upon the possibility of freely choosing such parameters as the key length, *l*, and the number of rounds, *r*, so that the user can modify the trade-off between computation speed and demanded security.

The computational operations are simple, of easy implementation in high level language and available in the assembler instruction set of all microprocessors and DSP's. The hardware design is still easier and more efficient, with low memory requirements. The cyclic rotations can be readily programmed in high level language like a combination of two displacements.

The algorithm is word-oriented, the basic computational operations work on full computer words of data at a time, saving computer time. In this paper we describe the 32 bit word length version, but other versions for different word lengths have been designed.

As a result, it provides high security when appropriate parameters have been chosen.

2. Notation and Akelarre primitive operations

Akelarre uses the following primitive operations over pairs of 32-bit blocks:

- Two's complement addition (and its inverse), denoted by "+". It corresponds to the additive group in $\mathbb{Z}_{2^{32}}$.
- Bit-wise exclusive-OR, denoted by "⊕". It corresponds to the additive group in $\mathbb{Z}_2$.
- Cyclic left rotation: the cyclic rotation of word *x* left by *y* bits is denoted $x \lll y$.

A significant feature in Akelarre is the use of rotations, dependent on both the input data and the key. That is, intermediate result words are rotated an amount determined by other intermediate result words and the key, thus strengthening the cryptographic security of Akelarre, because the bits are rotated to random positions, which are not -and cannot be- predetermined.

Therefore, the strength of Akelarre relies upon the cryptographic properties of data dependent rotations and the mixing of operations from different algebraic groups having the same number of elements.

Rotations as a powerful cryptographic tool were successfully used in the nonlinear generation of pseudo-random sequences [FUS93], structure patented in 1993. Since 1991 a research work has been developed in the C.S.I.C. using rotations as nonlinear structures, as described in the following works: [FUS91], [GUI91], [GUI94].

3. Description of the algorithm

This section describes the design of Akelarre, which consists of two components:

- The encryption/decryption algorithm.
- The key expansion algorithm, intended to generate a long enough key from the secret key entered by the user.

3.1 Encryption algorithm

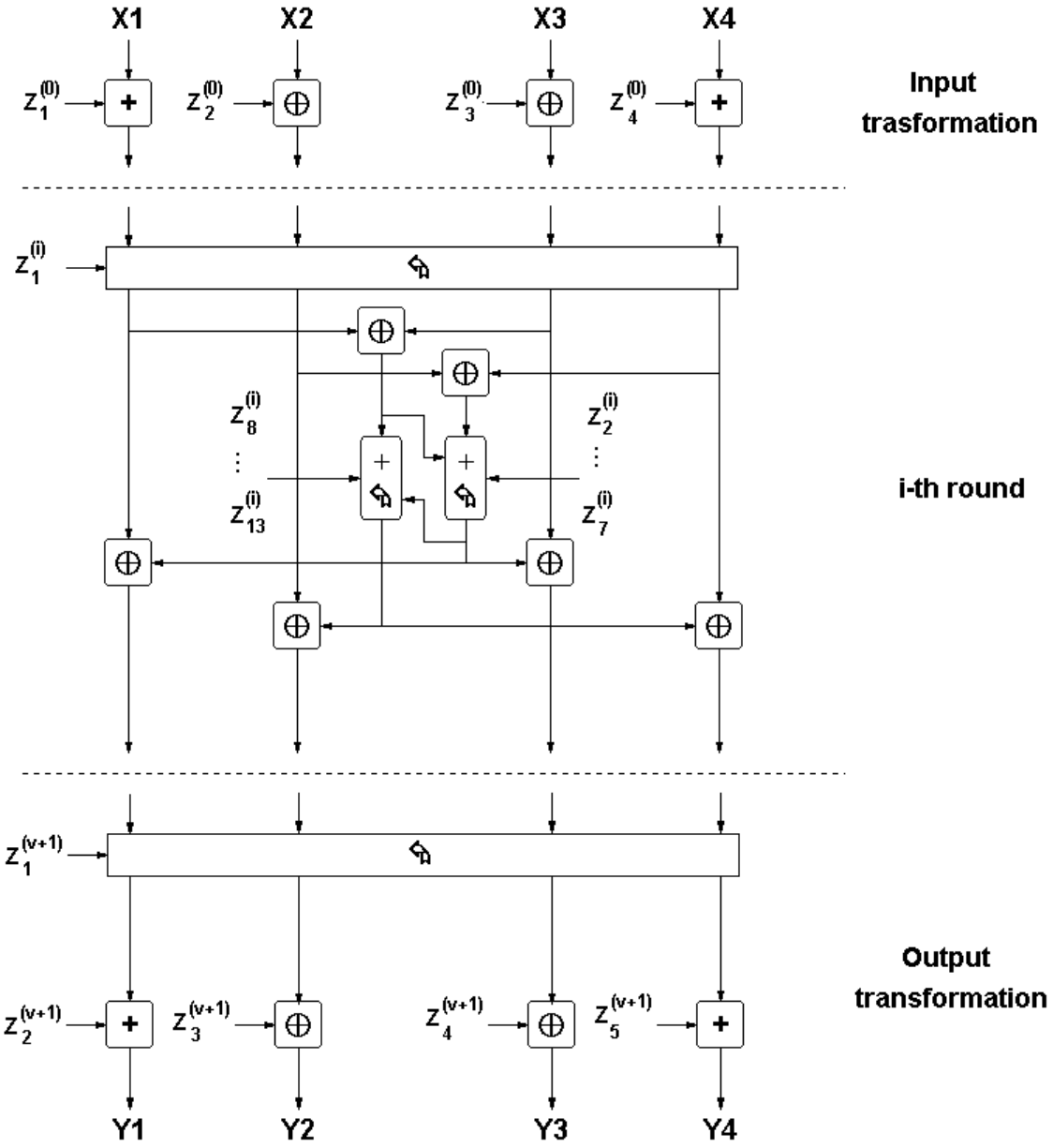
Figure 3.1 is an overview of the Akelarre algorithm. It is composed of three parts: the input transformation, *r* encryption rounds, and the output transformation.

The plaintext is reorganised prior to its entry into the algorithm, being partitioned into 128-bit blocks, denoted by X. These blocks, once fed into the algorithm, are divided into four 32-bit sub-blocks: X1, X2, X3 and X4. This four sub-blocks constitute the initial data to the input transformation.

The input transformation consists of four 32 bit-block operations. The two sub-blocks X1 and X4 are two's complement added with the sub-keys $Z_1^{(0)}$ and $Z_4^{(0)}$, respectively. The other two sub-blocks, X2 and X3, are bit-wise XORed with the sub-keys $Z_2^{(0)}$ and $Z_3^{(0)}$. The goal of this transformation is to prevent the adverse effect of possible input blocks of all-zeroes or all-ones, and to obstruct the differential cryptanalysis.

The first operation of the *i*-th encryption round is a cyclic rotation of the 128 bit block formed by the concatenation of the results of the input transformation or previous encryption round. The rotation is controlled by the seven least significant bits of $Z_1^{(i)}$. Then, the 128 bit resulting block is divided into four 32-bit sub-blocks. The four sub-blocks are combined with

twelve 32-bit sub-keys, represented as $Z_2^{(i)}$,…, $Z_{13}^{(i)}$ according to the addition-rotation structure shown in the figure 3.2.



X_i: 32-bit plain text sub-block
Y_i: 32-bit cipher text sub-block
$Z_i^{(j)}$: 32-bit key sub-block
+: Addition modulo 2^{32} of 32-bit integers
⊕: Bit-wise exclusive-OR of 32-bit sub-blocks
↰: Rotation

Figure 3.1 Akelarre algorithm design.

The output transformation consists of the cyclic rotation of the 128 bit block formed by the concatenation of the results of the *r*-th round, controlled by the seven least significant bits of $Z_1^{(r+1)}$, followed by four 32 bit-block operations in the same way as in the input trasformation, using the sub-keys $Z_2^{(r+1)}$, $Z_3^{(r+1)}$, $Z_4^{(r+1)}$, $Z_5^{(r+1)}$. After the output transformation, the four sub-blocks Y1, Y2, Y3, and Y4 are reattached to produce the output ciphertext block Y.

The goal of the cyclic rotations of the 128 bit blocks at the beginning of each round and at the beginning of the output transformation, is to enhance the diffusion when more than one round is used, although for one round operation has a low effect. Thus avoiding that the input data to the output transformation were simply the output of the input transformation XORed with the outputs Q1 and Q2 of each addition-rotation structure.

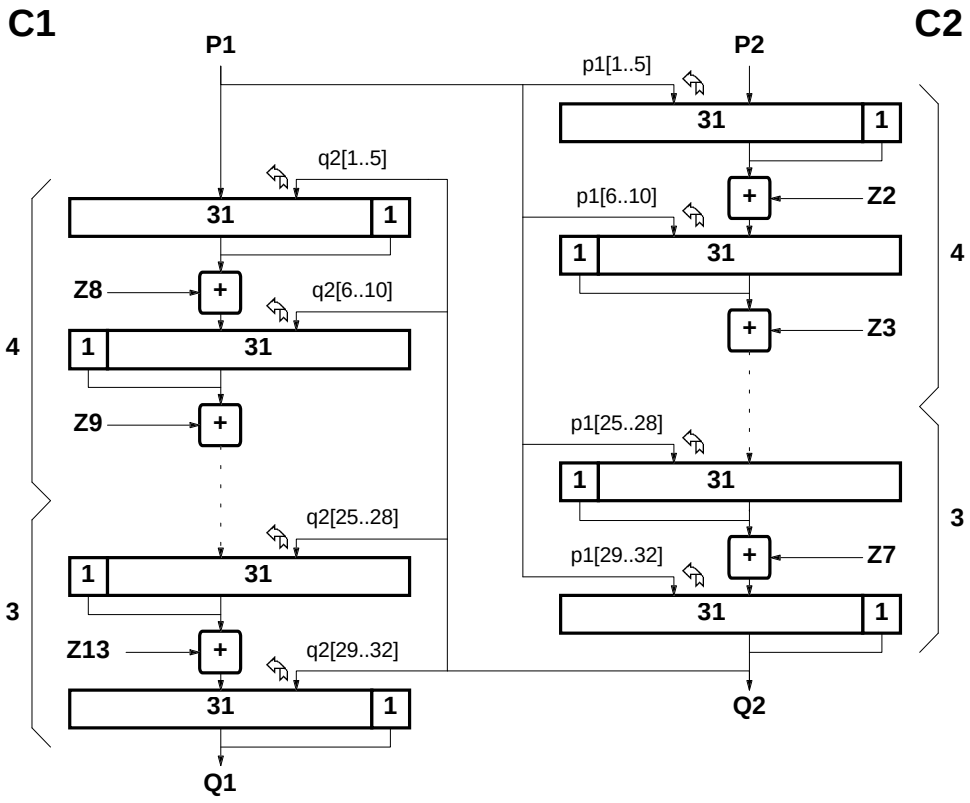


Figure 3.2 Addition-rotation structure

Figure 3.2 shows the addition-rotation structure. It is formed by two columns, each of them with six adders and seven rotators, where the amount rotated is determined by the other

column. This specially designed feature ensures with a 100% probability that *all* the plaintext and key bits will always determine *at least one* rotation. The reason why in each rotation 31 bits instead of the whole word are rotated is that a prime number has no divisors, whereas when a composite number is used, and the data in the rotator have a repetitive pattern, with repetition period equal to divisors of 32, an invariant rotation is produced, whose output result is identical to the input. However, with a prime number, any pattern will lead to a different output for any rotation. The convenience of this operation was studied in [GUI96].

In figure 3.2 it is observed how the 31-bit rotations are performed. In each 32 bit sub-block the least or the most significant bit is left unchanged, alternatively. The remaining 31 bit are rotated to the left an amount determined by the bits of the other column. The first 4 registers of each column are controlled by a set of 5 bits, whereas the 3 last registers are controlled by a set of 4 bits. The column C2 is controlled by the input bits of the column C1, but the column C1 is controlled by the output bits of the column C2. Akelarre is in fact a double iterated cipher, due to it has a variable number of rounds, *r*, but each round can be seen as the iteration of 14 half-rounds of rotation interleaved with 12 additions.

3.2 Decryption algorithm

The decryption algorithm is essentially the same as that for encryption and the computational graph of figure 3.1 is still of application. The only change is that the decryption key sub-blocks are computed from the encryption key sub-blocks as follows:

Round	*Encryption sub-keys*	*Decryption sub-keys*
Input transformation	$Z_1^{(0)}\ Z_2^{(0)}\ Z_3^{(0)}\ Z_4^{(0)}$	$-Z_2^{(r+1)}\ Z_3^{(r+1)}\ Z_4^{(r+1)}\ -Z_5^{(r+1)}$
1	$Z_1^{(1)}\ Z_2^{(1)} \ldots Z_{13}^{(1)}$	$(Z_1^{(r+1)})^{-1}\ Z_2^{(r)}\ Z_3^{(r)} \ldots Z_{13}^{(r)}$
2	$Z_1^{(2)}\ Z_2^{(2)} \ldots Z_{13}^{(2)}$	$(Z_1^{(r)})^{-1}\ Z_2^{(r-1)}\ Z_3^{(r-1)} \ldots Z_{13}^{(r-1)}$
i	$Z_1^{(i)}\ Z_2^{(i)} \ldots Z_{13}^{(i)}$	$(Z_1^{(r+2-i)})^{-1}\ Z_2^{(r+1-i)}\ Z_3^{(r+1-i)} \ldots Z_{13}^{(r+1-i)}$
r	$Z_1^{(r)}\ Z_2^{(r)} \ldots Z_{13}^{(r)}$	$(Z_1^{(2)})^{-1}\ Z_2^{(1)}\ Z_3^{(1)} \ldots Z_{13}^{(1)}$
Output transformation	$Z_1^{(r+1)}\ Z_2^{(r+1)}\ Z_3^{(r+1)}\ Z_4^{(r+1)}\ Z_5^{(r+1)}$	$(Z_1^{(1)})^{-1}\ -Z_1^{(0)}\ Z_2^{(0)}\ Z_3^{(0)}\ -Z_4^{(0)}$

Table 3.1 Encryption and decryption sub-keys.

3.3 Key expansion algorithm

The user secret key length can be freely chosen, in 64 bits increments. But the algorithm needs a much longer key, being the minimum 22 sub-keys of 32 bits (704 bits), needed for a one round algorithm. Therefore, a key expansion routine is required to transform the user secret key k into sub-keys to be used in the different encryption rounds.

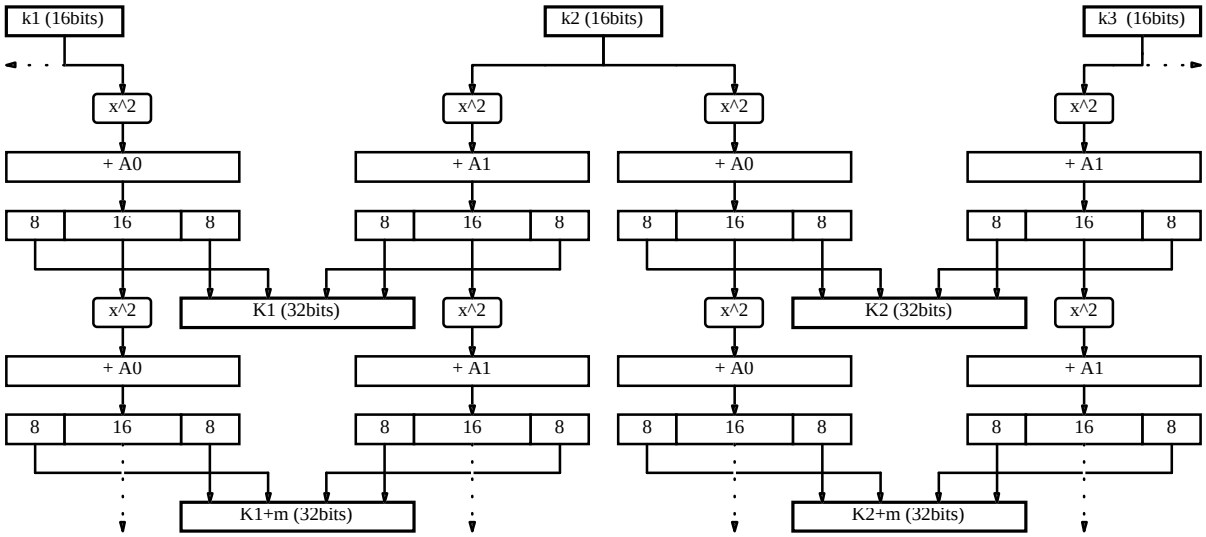


Figure 3.3 Key expansion algorithm.

The design of the expansion algorithm is depicted in figure 3.3. The user-selected key is partitioned into *m*, k_i. Each sub-block is squared and then added a constant A_0 or $A_1 \mod 2^{32}$. The main reason that justifies the use of these constants is to avoid the adverse effect of a possible user secret key of all-zeroes. They must be chosen as A_0, $A_1 \neq 2^{32} - n^2$, with n integer, and with a good statistical distribution of 1's and 0's, to prevent a possible null result of the sum, either because the sub-key is zero or because the result of the squared is 2^{32}, which would be internally represented as 32 zeroes. In our implementation the values are: A_0=A49ED284H and A_1=735203DEH, they have been chosen carefully, to avoid weak keys. In the worst case, with the values k_i=98F3H and k_{i+1}=k_{i+2}=BDC7H and after the first key expansion routine iteration, K_i will have the form XX00XX00H, and the adjacent sub-key K_{i+1} XXXXXX00H, but in the next iteration of the key expansion routine K_{i+8} and K_{i+9} will be of the form XXXXXXXXH.

The 16 inner bits of the resultant addition are used to generate a new 32-bit word and continue iterating, while the 8 most significant bits and the 8 least significant bits of two adjacent columns are used to build up the sub-key K_i, in the way shown in figure 3.3.

The complicated non-linear function used to derive the sub-keys causes a avalanche effect of bit changes: a change of an input bit will induce a change of 8 output bits, in average.

Note that each 16-bit sub-block is used to generate at least two adjacent sub-keys, and usually much more sub-keys. Hence, an important avalanche effect is produced by the key expansion routine. A typical configuration of 4 rounds of encryption with a 128-bit user key, will need 61 sub-keys, to be generated from a set of 8 16-bit sub-blocks, it means that approximately each sub-block is used to generate 16 sub-keys, so a one bit change of the user key will produce, in average, a 128 bits change of the encryption sub-keys.

This key expansion function has been designed to be absolutely unidirectional, that is, the calculation of a certain K_i is of no avail to compute the previous K_{i-1}. Even the knowledge of K_i turns out to be useless to determine K_{i+1}, i.e., it is not only impossible to go backwards but to move forward as well in order to gain insight of the expanded key from a given K_i. This feature is of great importance as long as it prevents a possible cryptanalyst from obtaining the key expansion starting from a known sub-key K_i. The key expansion is pre-computed only once, thus slightly affecting the execution time of the algorithm.

After calculating all the K_i sub-keys, they are read sequentially to fill the $Z_j^{(i)}$ keys of the table 3.1

4. Discussion on performance and security of Akelarre

Akelarre offers the possibility of freely choosing such parameters as the key length, *l*, and the number of rounds, *r*, so that the user can modify the compromise between speed and security. For most applications, we propose that 4 rounds and 128-bit user secret key are used.

Several tests have been performed on a 130 MHz Intel® Pentium™, using a sample program working under Windows®95 compiled with Microsoft® Visual C++™ 4.0 compiler. It was obtained that Akelarre's speed with the proposed conditions, was 3,22 Mbits/second.

Obviously, this speed will increase or decrease as less or more encryption rounds are used.

We have performed some tests, in the same conditions, with other block cipher algorithms, and we have found that the speeds of IDEA and Akelarre are similar for the same number of rounds, although it must be remembered that the authors of IDEA recommend to use it with eight rounds. Otherwise, RC5 with the author's recommended configuration of 12 rounds, and 16-byte secret key is about two times the speed of Akelarre with the above parameters.

The theoretical security evaluation is not yet completed. However, extensive statistical tests have been performed and they show that there is no resemblance between the plaintext and the ciphertext apart from statistical coincidences. Furthermore, that the distribution of 1's and 0's approximates the ideal case of a perfectly random file.

As example, Figure 4.1. shows the graphs of the cross correlation test between plaintext and ciphertext and the runs test, for 512-byte files of all-zeroes plaintext, user key of all-zeroes, one round and key length of 128 bits.

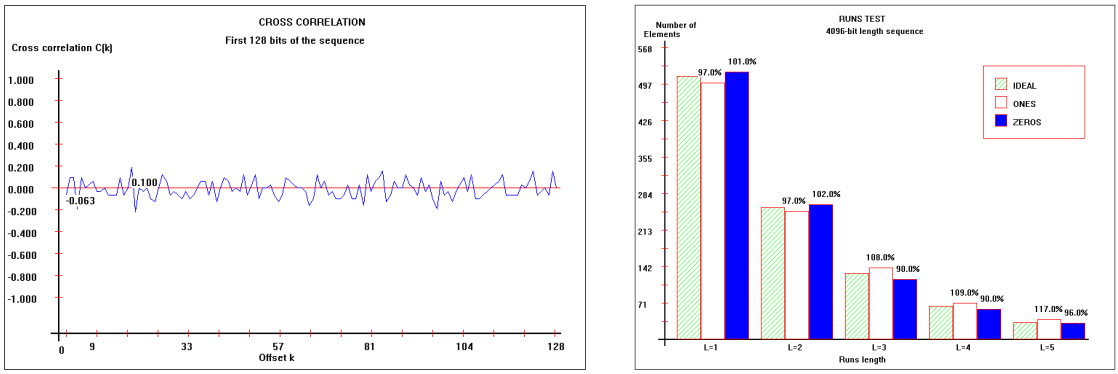


Figure 4.1 Cross correlation test between plaintext and ciphertext files and Runs test of ciphertext file.

In the table 4.1 we can see an example of how diffusion is achieved when a bit change takes place in the key or in the plaintext, with a user key of 64 bits and one round only. It should be highlighted that the change of one input data bit or one key bit is enough to produce a difference of about 50% of encrypted data.

Plaintext	Key	Ciphertext	Difference
0000 0000 0000 0000 0000 0000 0000 0000	0000 0000 0000 0000	6F06 02DF 74B0 117F 8372 49E9 72C2 F0CE	
0000 0000 0000 0000 0000 0000 **1**000 0000	0000 0000 0000 0000	759A D74E 7166 4EA2 9D9F 145A 7698 B79B	61 bits 47.65%
0000 0000 0000 0000 0000 0000 0000 0000	0000 0000 000**1** 0000	B915 06A7 C751 6324 D905 4DB1 2061 A315	66 bits 51.5%

Table 4.1 Comparison between plaintext and ciphertext when one plaintext bit and one key bit is changed with one round only.

Because of the double iterated architecture of Akelarre, differential and linear cryptanalysis seem quite difficult. To simplify both attacks let us ignore the initial transformation and the 128-bit rotations. And then let us operate on the smallest size cipher, that is one round. Then we face up to the addition-rotation structure, which is itself an iteration of a simpler structure, composed of alternated rotations and additions.

This structure presents a notable similarity with RC5 encryption algorithm, and can be analysed in the same way. Kaliski and Yin [Kal 95] have found a way to attack to RC5 with differential analysis focusing on characteristics for which the pair of inputs have the same rotations amount, because "… if a pair of inputs to a half-round have different rotation amounts, then the pair of outputs from the half-round will differ in many bits". They recognise that in the opposite case the differential cryptanalysis will be unfeasible. In our design, the rotation structure guarantees with 100% probability that a change of one bit in the plaintext will produce many rotations. If an input bit to C1 is involved the least rotation number is 1 and the highest is 6, whereas if an input bit to C2 is involved the number of rotations is 8, being the average 5-6 rotations. So we may suppose that Akelarre has a good security against the differential cryptanalysis.

In the same paper, the security of RC5 against linear cryptanalysis is assessed when more than 5 rounds of addition-rotation structure are used. We are using about 6.5 of similar rounds (14 half-rotations and 12 half-additions). Then, we can presume that Akelarre is not unsafe against linear cryptanalysis.

Taking into account the previous arguments, the statistical tests, and the fact that Akelarre will always be used with more than one round, we conclude that, for most applications, 4 rounds and 128-bit user secret key will be suitable. Nevertheless, we are working in a detailed security analysis that we hope to be able to present in next cryptographic meetings.

5. Comparison with other block ciphers

Amongst all block cipher algorithms, the most widespread and probably most secure are DES, IDEA [LAI90], and RC5 [RIV94]. DES, although still secure, is doomed as international standard due to its short key length, of only 56 bits, allowing a brute-force attack to find the key in an average of three hours and a half with a $1 million machine [WIE93]. Furthermore, it has been broken by Shamir and Biham [BIH93] for a reduced number of rounds and for some modified variants and modes of implementation faster than via exhaustive search. RC5 and IDEA are counted amongst the possible candidates to become international data encryption standard. Actually, IDEA has been recently adopted as encryption algorithm by Pretty Good Privacy (PGP).

RC5 relies on data dependent rotations to frustrate differential and linear cryptanalysis. However, in spite of the cryptographic strength of rotations, their use in RC5 turns out to be insufficient, since in each rotation only the $\log_2 w$ least significant bits of the word w commanding the amount rotated are taken into account. Consequently, the other w-$\log_2 w$ have no effect at all on the rotation. In Akelarre, in turn, the 32 bits of each block are used, 5 or 4 bits in each rotation (figures 3.1 and 3.2).

Given that the key expansion process adopted by RC5 allows the cryptanalyst to compute the expanded key table S entry by entry in reverse order, it was possible to recover that table without carrying out an exhaustive search. More specifically, the way in which rotations are used in RC5 helps an attacker considering that information about one bit of a half-input register can be spread by the rotation in the last half-round to give information about every bit of the key $S[2r+1]$ [KAL95].

On the contrary, in Akelarre, due to the design structure and the key expansion method, it is impossible to perform a cryptanalysis in which the key expansion is computed, because the knowledge of a certain K_i does not allow to compute K_{i-1} nor K_{i+1}. Nevertheless, Akelarre equals

the desirable flexibility of RC5, concerning the possibility of choosing between such parameters as the number of encryption rounds and key length.

Although in [LAI90] it is stated that the 3 operations are incompatible in the sense that no pair out of them satisfies a distributive law, multiplication and integer addition satisfy a partial distributive law, stemming from arithmetic modulo $2^{16}+1$, which has been exploited in the cryptanalysis of the first two rounds of IDEA [MEI93], even though it is not suitable for the attack of complete 8-round IDEA. However, Akelarre overcomes this drawback by not using multiplications but data dependent rotations.

Moreover, regarding facility of implementation and speed, Akelarre has the advantage over IDEA of not using multiplications as nonlinear operations, which are time and resource consuming, but rotations, of immediate computation in all microprocessors. However, because of the heavy use of rotations, Akelarre's and IDEA's encryption round speeds are similar, for the same number of rounds. Presumably, hardware implementations of Akelarre would be faster and more feasible, provided that hardware rotation is much easier and much more economic than multiplication.

Another shortcoming in IDEA is its key schedule: the 128-bit key is partitioned into 8 sub-blocks that are directly used as the first eight key sub-blocks. Next, the key is shifted to the left by 25 positions, after which the resulting 128-bit block is again partitioned into eight sub-blocks that are taken as the next eight key sub-blocks, and this procedure is repeated until all 52 key sub-blocks have been generated. Hence, the knowledge of sub-keys gives the attacker information about the original 128-bit user key, which could be completely obtained, yet with Akelarre's expansion method it is not only impossible to know the key sub-blocks previous to a recovered k_i but to compute key sub-blocks posterior to k_i as well.

7. Conclusions

A new block cipher, Akelarre has been proposed. One of its characteristic features is the heavy use of data dependent rotations of prime-length registers, that is, the amount of rotations performed depends on the input data and the key and, therefore, it is not predetermined.

A second distinguishing feature is the design concept of mixing arithmetic operations from

different algebraic groups having the same number of elements.

The complete diffusion requirement is satisfied after a single round, as it can be observed in table 4.1. The influence of individual plaintext or key bits should spread over all the ciphertext bits, so that the change in one plaintext or key bit causes the change of about 50% of the ciphertext bits, uniformly distributed all along the block. Diffusion is provided by the structure called addition-rotation, represented in figure 3.2, which is an invertible transformation and has a complete diffusion effect in the sense that each output sub-block bit depends on each input sub-block and key bit.

The similarity of encryption and decryption means that in order to decrypt Akelarre it is enough to repeat the same transformations performed throughout the encryption process. As we are dealing with an involution, its repeated application with suitable keys, leads to the original data.

The key expansion algorithm allows the generation of as many sub-keys as necessary from a user-defined key of arbitrary length. These sub-keys are pre-computed in a way which prevents the recovery of the initial user key or the computing of the expansion starting from a known sub-key.

It is still soon to resolve about Akelarre's security, but it seems that either data dependent rotations, the mixing of operations from different groups and the key expansion scheme adopted, work together against possible cryptanalysis. After the first tests we recommend the use of Akelarre with four rounds and 128-bit length key for most purposes, but more research on algorithm security is been carried on, and final recommendations may be modified.

Acknowledgement - This paper was supported with C.I.C.Y.T., Spain, under grant TIC95-0080 (project *"Sistema criptográfico de protección de datos para red digital de servicios integrados RDSI"*).

8. Bibliography

[BIH 93] Eli Biham y Adi Shamir: Differential Cryptanalysis of the Data Encription Standard. Springer-Verlag, 1993.

[FUS 91] A. Fúster, D. de la Guía, J. Negrillo, F. Montoya. Diseño e implementación de algoritmos de generación de secuencias binarias. Actas de la I Reunion Española sobre Criptografia. Palma de Mallorca, 1991.

[FUS 93] A. Fúster, D. de la Guía, J. Negrillo, F. Montoya, Estructura no lineal para la generación de secuencias pseudoaleatorias. Patent application 9300561. Spain 1993

[GUI 91] A. Fúster, D. de la Guía, J. Negrillo, F. Montoya. Estructuras no lineales para la generación de secuencias binarias de Aplicación Criptográfica. Actas del IV Simposium Nacional de la Unión Científica Internacional de Radio, 904 -- 908.Cáceres, 1991.

[GUI 94] D. de la Guía Martínez, F. Montoya Vitini, E. Valderrama, L. Porta, ASIC_CRIPTO: un circuito integrado para el modulo de seguridad del PLANBA. Actas de la III Reunión Española sobre Criptografía. Barcelona, 1994.

[GUI 96] D. de la Guía, A. Fúster. Estudio de autómatas celulares para criptografía. To be published in Actas de la IV Reunión Española sobre Criptografía. Valladolid, 1996.

[KAL 95] B. S. Kalisky, Y. L. Yin, On Differential and Linear Cryptanalysis of the RC5 Encryption Algorithm. Advances in Cryptology-CRYPTO'95, 171--184, 1995.

[LAI 90] X. Lai, J. Massey, A Proposal for a New Block Encryption Standard. EUROCRYPT 90, 389-404. Springer Verlag, Berlin 1991

[MEI 93] W. Meier, On the Security of the IDEA Block Cipher. Advances in Cryptology-EUROCRYPT'93, 371--385, 1993.

[RIV 94] R.L. Rivest, The RC5 Encryption Algorithm. In Proceedings of the Workshop on Cryptographic Algorithms, K. U. Leuven, 1994.

[WIE 93] M. J. Wiener, Efficient DES Key Search. CRYPTO '93, 1993.