Monte Carlo Scheme: Cryptography Application

R. Flores-Carapia
Instituto politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico En Cómputo
rfloresca@ipn.mx

V. M. Silva-García
Instituto politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico En Cómputo
vsilvag@ipn.mx

C. Rentería-Márquez
Instituto politécnico Nacional
Escuela Superior de Física y Matemáticas
crenteriam@gmail.com

Abstract

It has been previously shown that the DES algorithm is strengthened when it begins with a variable permutation, following the JV Theorem. Said algorithm states that a positive integer number less than $64! \approx 10^{89}$ may be associated to a 64 positions permutation in 63 steps, thus increasing the DES algorithm computational complexity by making each permutation a as a key. However, the effects of the variable initial permutation and the 56 bits key on the computational complexity of the algorithm are yet unexplored. The current work presents a robust evidence — that is, the error can be reduced as much as needed — for the efficiency of the algorithm. The latter is achieved using a Monte Carlo scheme coupled with a Birthday model for probabilities computation, and the numbers $\pi$ and $e$ for random strings generation. In this paper, the error has an upper bound of $1/2^{100}$.

Keywords: Efficient, JV Theorem, Birthday Model, DES, Monte Carlo Decision Model, Variable permutation
1. Introduction

In the analysis presented here, the DES encryption cycle [5] does not consider the inverse permutation at the end: only a variable initial permutation is applied. The latter is denoted as $\chi_{185}$ and $\chi_{301}$, where $\chi_{184}$ is a 64 positions permutation, $\chi_{183}$ is a 56 bits key, and $\chi_{1850}$ is a 64 bits string, also known as plaintext. Also, notice that the 64 bits string 00...0 is denoted as $\chi_0$.

Consider the following statement as valid: “When the $2^{56}$ keys allowed by DES are applied to $\chi_0$ as plaintext, the results are all different”. If the latter is true, then it is not difficult to show that the DES algorithm — as defined above — encrypts in a different manner [13] under any of the following situations:

1. Given $P_1 \neq P_2$ and the same key.

2. Given $P_1 \neq P_2$ and $K^1 \neq K^2$.

The first case is proved in [13]. For the second case, it is enough to show that there exists at least one plaintext, $X_0$ for which $e_{P_1,K^1}(X_0) \neq e_{P_2,K^2}(X_0)$, which proves that the output arrays associated to $P_1$, $K^1$ on one hand, and $P_2$, $K^2$ on the other are different. It follows that it is enough to take $X_0 = 0$, since for such plaintext $X_0 = 0$ we have $e_{P_1,K^1}(X_0) \neq e_{P_2,K^2}(X_0)$, by the assumption introduced above.

Clearly, a particular case of the former reasoning is that of $K^1 \neq K^2$ and the same permutation.

In this sense, the assumption that when the 256 keys allowed by DES are applied to $0$ as plaintext, the results are all different is quite important. In the current work, a robust evidence for such assumption to be true is given, using the Monte Carlo method [6]; that is, the probability of being wrong can be reduced as much as needed. Thus, the modified DES algorithm can be considered to be efficient in the sense that each bit added to the key causes a two-fold increase in the computational complexity of a brute force attack needed to solve it.

Remember that for the proposed DES algorithm, the key is made up by two parts: the 56 bits key on one hand (with 256 possibilities), and the number associated to the variable permutation (which can be up to $10^{89}$) [13]. Given that this number can be represented by a string of approximately 296 bits in length, there are $2^{296}$ possible numbers.

In this context, it may be considered that the proposed DES algorithm has a computational complexity, for a brute force attack attempting to solve it, of approximately ($2^{56}$) ($2^{296}$) = $2^{352}$. That is, all $2^{56}$ possible keys K would have to be tested for each permutation, in order to find the appropriate output array [13].
2. Monte Carlo Scheme

As mentioned before, the Monte Carlo method [6] is used because, on one hand, the encryption needs to be applied $2^{56}$ times to string 0 (once for each of the 56 bits key); and on the other hand, all the elements of the set of 64 bits strings must be compared, since such strings are the results of encrypting 0 with each of the keys. Clearly, these requirements imply a large amount of number crunching, thus validating the use of the Monte Carlo method.

In what follows, let us denote the set of all possible encryptions of plaintext 0, using all $2^{56}$ possible keys, as $E$. Notice that there are at most $2^{56}$ elements of $E$, each of which being a 64 bits long strings. Then, in order to find out whether it is reasonably true that all elements of $E$ are different, it is necessary to know the probability that for a sample $M$ of size $m$ taken from the set $E$, at least two of its elements are equal. In this respect, the Birthday [12] model may be used to compute the latter probability.

Thus, if the first string is obtained from encrypting plaintext 0 with key $K^1$ and the selection of key $K^2$ is considered to be independent to $K^1$, then the probability that the second string — the result of applying $K^2$ to 0 — has a different value from the first string, is that of taking one of the $2^{56} - 1$ values left (assuming there are $2^{56}$ different strings as possible results). Given that at this point, it is unknown how many different elements have set $E$, the probability that the second string takes a different value from the first string is at most $\frac{2^{56}-1}{2^{56}} [12]$.

According to the latter, when a third key $K^3$ — also picked independently to the former two keys — is applied, the probability that the result takes one of the different values left is at most: $\left(\frac{2^{56}-1}{2^{56}}\right)\left(\frac{2^{56}-2}{2^{56}}\right)$.

By following this procedure, the probability that $m$ resulting strings take different values from each other — when all keys have been picked independently from each other — is at most:

$$\left(\frac{2^{56} - 1}{2^{56}}\right)\left(\frac{2^{56} - 2}{2^{56}}\right) \cdots \left(\frac{2^{56} + 1 - m}{2^{56}}\right)$$

In this train of thought, the probability of at least two strings being equal — denoted as $P_m$ — has a minimum value of:

$$P_m = 1 - \left[\left(\frac{2^{56} - 1}{2^{56}}\right)\left(\frac{2^{56} - 2}{2^{56}}\right) \cdots \left(\frac{2^{56} + 1 - m}{2^{56}}\right)\right]$$
In this regard, if $m$ is computed such that \( \left( \frac{2^{56}-1}{2^{56}} \right) \left( \frac{2^{56}-2}{2^{56}} \right) \cdots \left( \frac{2^{56}+1-m}{2^{56}} \right) \approx \frac{1}{2} \), then it may be said that for such particular $m$, probability $P_m$ is at least $\frac{1}{2}$. Calculating that $m$ gives the result as $m = 32000000$.

Now, let us define the following decision problem [4]: taking as keys each element of the set $K = \{K | K \in \{0,1\}^{56}\}$: do at least two equal strings exist when said keys are applied to encrypt the plaintext 0?

This decision problem has only two possible answers: either yes or no. However, if a sample of size $m$ is taken from the set of encryptions of plaintext 0, then for this particular instance of decision there are the following scenarios:

(1) If the answer is yes, it is correct.

(2) If the answer is no, there is a risk of it being false with a probability of at most $\frac{1}{2}$.

Now, if this experiment is repeated, say 100 times (where each experiment is run independently from the others), and for each instance the answer is no, then the probability that such an answer is false would be at most $\frac{1}{2^{100}}$. If the latter is true, then the assumption that applying each key from the set $K$ to plaintext 0 gives different results becomes reasonable, especially for practical situations.

Now, in order to use the Monte Carlo method, pseudo-random samples must be taken. For this purpose, the fractional part of transcendental numbers [14] such as $\pi$ and $e$ are of great use, given that they are considered to fulfill the normality property [6], even though there is no theoretical proof thereof as of yet.

A number is said to be transcendental if it is not a solutions for any expression of the form $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = 0$, where all $a_i$ are integers for $i = 0, \ldots, n$ [14]. Thus, it is not complicated to prove that any number of the form $b * e$ or $b * \pi$ where $b$ is any integer other than 0 also being a transcendental number.

In fact, this proof may be done by *reductio ad absurdum*. As a particular instance, $b$ may be chosen to be any prime number greater than 1 (i.e. $b \geq 3$). Then, 100 prime numbers may be chosen pseudo-randomly, in order to multiply 50 of them by $e$, and the other 50 by $\pi$, thus obtaining the needed transcendental numbers.

For the current work, the decimal fractions of both $\pi$ and $e$ were taken from [9], while reference [1] presents an alternative manner to find the desired decimals from the fraction of both $\pi$ and $e$. 
3. Results and Discussion

Initially, there must be 320000000 strings of 56 bits in length each, which must also be different from each other. In practice, this does not happen. When $\pi$ and $e$ are multiplied by a prime number—which may be of up to 18 digits long—and the fractional part of the product is used to select the needed strings, there may be some repeated strings. Thus, more digits are needed in order to make sure all strings are different. For the current work, approximately 6400000000 digits were taken from $b \cdot e$ or $b \cdot \pi$ products.

Each one of the 100 applications of the 320000000 56-bits strings to cipher the plaintext 0 gave a different encrypted string as result. Thus, the negation of the proposition “there is at least two equal encrypted strings as result of applying all the $2^{56}$ possible keys to plaintext 0” could be false with a probability of error of at most $\frac{1}{2^{100}}$. This result leads to the conclusion that it is highly probable that all encrypted strings of plaintext 0 are different from each other.

Below is a particular result of the process described before: first the product of the prime number 101 and $e$ is taken, then some of the resulting 64-bits strings are taken and used to encrypt the plaintext 0.

Example 3.1. Experiment process using the numbers 101 and $e$.

Results of the product of 101 and $e$:

- Block number 3627: FD 75 5F 28 5C 94 6C.
- Block number 35157: DE 86 05 A8 CA 82 F4.
- Block number 64000: 73 14 61 9B A7 16 AD

Cryptosystem input (key) and output (encrypted string).

- Input FD 75 5F 28 5C 94 6C → output 85 47 FB 61 00 50 54 0A.
- Input DE 86 05 A8 CA 82 F4 → output B7 1E DA DB F6 BF 35 F1.
- Input 73 14 61 9B A7 16 AD → output 69 FE FE 36 B0 27 6C 5F.

4. Conclusions and Future Work

The present work shows the robustness of the proposed modification to the DES algorithm, by repeating 100 times a decision problem (described in section 3) which manifests the high probability of the modified DES algorithm being efficient. Such efficiency is better illustrated when stated that, by increasing one
bit the length of the keys used in the process, the cryptosystem increases by a factor of 2 the computation complexity needed to break it, by a brute force attack.

As described at the beginning of the paper, the DES algorithm key is made up by two parts: one of approximately 296 bits long and another of 56 bits long. Thus, the computational complexity needed to break such cryptosystem is approximately \((2^{296})(2^{56}) = 2^{352}\). The latter implies that it is insufficient to know the \(K\) key, since it becomes necessary to also know the initial permutation used at the beginning of the encryption process.

It is in part thanks to this feature that the DES algorithm with initial variable permutation can avoid such threats as the Differential [3] and Linear [10] attacks; at least as they are currently known.

On the other hand, it is a common practice when using a Public Key Infrastructure (PKI) scheme to employ a symmetric cryptosystem (e.g. DES, Triple-DES, or AES) along with an asymmetric cryptosystem (e.g. RSA [11] or ElGamal [15], also known as public key with electronic signature [7]). The latter are adequate to transmit the number associated to a variable permutation.

As future work, it remains to be seen how likely it is to reach similar results by adding a variable permutation at the start of other encryption algorithms, such as Triple-DES [2], and the Advanced Encryption Standard, AES [8].

Acknowledgements. The authors would like to thank the Instituto Politécnico Nacional (Secretaría Académica, COFAA, SIP, CIDETEC, and ESFM), the CONACyT, and SNI for their economic support to develop this work.

References

Monte Carlo scheme: cryptography application


Received: September, 2012