How to Distinguish Between a Block Cipher and a Random Permutation by Lowering the Input Entropy

[Extended Abstract]

Hernández J.C.[1], Isasi P.[2], Sierra J.M.[1], Gonzalez-Tablas A.[1]

1

Computer Science Department
Security Group
Carlos III University
28911 Leganés-Madrid-Spain
{jcesar,sierra,aigonzal}@inf.uc3m.es

2

Computer Science Department
Artificial Intelligence Group
Carlos III University
28911 Leganés-Madrid-Spain
isasi@ia.uc3m.es

Abstract. - In this paper a new cryptanalysis technique is presented, and its suitability in distinguishing a block cipher algorithm or a hash function from a random permutation is explained. Additionally, we propose a genetic algorithm based implementation and show some preliminary results of these ideas on reduced rounds versions of the block cipher TEA.

Introduction

A theoretically perfect block cipher, or a hash function, must behave as being a random permutation for any given key. So a block cipher or a hash function that can be consistently and efficiently distinguished from a random permutation must be rejected for a large number of applications. Unfortunately, this undesirable behaviour is usually difficult to find because cryptanalysis is nowadays more an art than a science, and it can take large hours of work of a talented cryptanalyst to find these kinds of weaknesses.

In this article we will try to show how genetic algorithms could be useful to perform automated cryptanalysis of some cryptographic primitives, specifically at the new cryptanalytic technique we propose.

Meaningless measures

There is no point at all in performing some statistical tests that cryptographers usually do on some block ciphers, for example [1] when some random data is ciphered and then its output analysed to search for some statistical measures that are supposed to be found in truly random data. In most of the cases, these tests conclude that the behaviour of the block cipher is truly random. These conclusions are very common because even the worst block cipher algorithm, the identity, given a sufficiently random input, will produce a sufficiently random output.

0-7803-6636-0/01/$10.00 ©2001 IEEE

The point is that these tests are meaningless if the block cipher algorithm has been designed and implemented with care, and only useful if the algorithm is extremely weak.

Lowering the input entropy

Instead of feeding the block cipher with data as random as possible (we can call this classical approximation high-entropy-feeding), we propose the use of low-entropy-feeding: the fixing of some bits of the input of every block ciphered to look for how this affects the corresponding output.

This bit fixing in the input must be reasonable in its length to allow for sufficiently large and different output, but large enough as to be able to generate some undesirable behaviour in the output. If this deviation from the random behaviour is observed, then we could have a way of distinguishing the given block cipher from a random permutation.

The problem is how to decide which input bits have to be fixed and to which values. For this, we propose the use of a genetic algorithm. In particular, we will use a genetic algorithm to search for individuals that codify bitmasks. These bitmasks will be used to perform a logical AND with the random input of the block cipher, thus fixing some of the input bits to 0. We will observe some of the output bits, that are supposed to have a random and uniform distribution, and evolve the individuals (or input bitmasks) to prefer those that generate strongest deviations from the expected random behaviour of the output.

Results

We have implemented our ideas about low-entropy cryptanalysis and found them useful in this preliminary work about TEA cryptanalysis. TEA stands for Tiny Encryption Algorithm, and is a well-known block cipher algorithm that was invented by David Wheeler and Roger Needham [2] of the Computer Laboratory of Cambridge University. It is a strong block cipher algorithm to which only one attack is known [3], but this attack is related with its simplistic key schedule and does not ease our cryptanalysis at all. The security of TEA relies on the number of rounds used. So our results, that are applicable only to two rounds (the authors recommend eight at least) although promising and interesting, could not be seen as a prove of its weaknesses.

In the following, we will describe our experiments in detail to allow other researchers to verify and extend them. We have used a genetic algorithm implementation from William M. Spears, from the Navy Centre for Applied Research in Artificial Intelligence, with a crossover probability of 0.95 and a mutation rate of 0.05. The fitness function we were trying to maximize was related with the chi-square statistic of the output. We decided to observe the ten rightmost bits of the first output word of the TEA block cipher algorithm because some authors, notably [4], have shown that in ciphers like TEA, the rightmost bits of the output words generally present a worse distribution.

So we will observe the distribution of the ten rightmost bits of the first output word of TEA. These bits can be seen as representing numbers from 0 to 1023 and they should be uniformingly distributed. To test this, we can calculate a chi-square statistic that should correspond to a chi-square distribution with 1023 degrees of freedom. The values for different percentiles are shown in Table 1 below:

p-value	0.5	0.75	0.90	0.95	0.99
X^2 statistic	1022.33	1053.13	1081.37	1098.52	1131.15

Table 1
Different values of the chi-square distribution with 1023 degrees of freedom for some p-values

Our objective is to find bitmasks for the input (both the input block and the input key) that provokes a value in the chi-square statistic as far as possible from the expected ones. In this vein, we must obtain the highest possible value for the chi-square statistic. This suggests the use of the chi-square statistic as the fitness function to maximize. This is exactly what we did in our first attempts. Finally, as the genetic algorithm implementation we used only supports selection of the individuals proportional to their fitness and not to their rank, we modified the fitness function slightly (by raising it to the third power) to amplify little fitness differences.

Every bitmask was evaluated by performing an AND over 2^{18} different random inputs, and then calculating the chi-square statistic based fitness function. An example of one of the bitmasks we found is the following:

```
{0,0,0,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,1,1,0,1,1,0,0,1,0,1,0,0,1,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,1,0,0,1,1,1,1,1,1,0,0,0,1,1,1,0,0,0,1,1,0,1,0,0,0,0,0,1,0,0,0
,1,0,0,0,1,0,0,0,1,1,1,1,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0,1,0,0,1,1,0,1,0,1,0,1,0
,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,0,1,0,0,1,0,1,1,0,0,1,0,1,0,0,0,0,1,0,1
,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,1,0,1,0,0,0,1,1,0,0}
```

This bitmask has length 192 because TEA has an input block length of 64 and an input key length of 128. Although the 2^{18} random inputs used to calculate the fitness of each generation of the genetic algorithm were different, thus searching for more general results but making convergence harder, we tested this bitmask against a new, previously unseen set of another 2^{18} random inputs. The results obtained over this test set confirm that the bitmask above has a very strong influence over the output of the ten bits we observe, so these bits do not depend uniformingly of every input bit and those that have a 0 value in the bitmask above have, obviously, more influence.

This information can be used to mount a successful cryptanalytic attack over TEA reduced to two rounds, or at least to distinguish it from a truly random permutation. A partial representation of the output generated by this bitmask is shown in Figure 1. As shown below, there are not values of the form 4K-1, and there are much more values of the form 4K+1 than the expected value of 16. The distribution repeats approximately in groups of four consecutive values. The corresponding chi-square statistic value is 23591.875, extremely high.

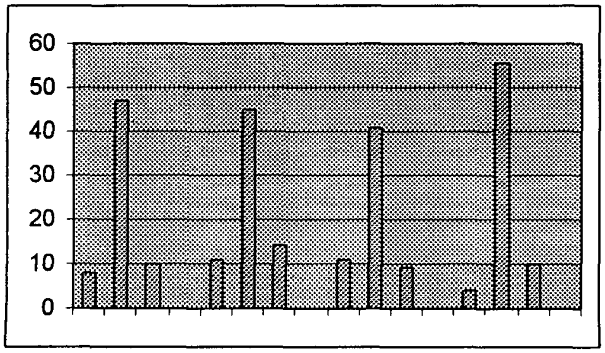

Figure 1
Partial representation of the output values of the observed bits.

Conclusions

In this preliminary work we have shown how a new cryptanalytic technique can be implemented with the aid of genetic algorithms to perform automatic cryptanalysis. We have also proved that a correlation between input an output bits in the block cipher TEA reduced to two rounds can be discovered with this approach.

We believe we have provided enough reasons to believe this is a promisingly technique. Moreover, the fact that this is a blind and automatic way of testing new cryptographic primitives such as hash functions or block ciphers increases its interest, because it allows its use as an evaluation criterion.

Further Research

Many improvements and extensions are possible over this first implementation. For example, using the bitmasks to fix some input bits to zero is interesting, but in some primitives fixing some input bits to 1 could have even more effect over the output. So probably a ternary representation for the individuals of our population (0,1,*) could be useful. Analogously, our chose of observing the ten rightmost bits of the first output word, although justified, is somehow arbitrary. Perhaps this can be also part of the genetic algorithm and be evolved to also search for the weakest output bits. After these ideas are implemented, more tests on other cryptographic primitives and more rounds of TEA will be carried.

References

[1] NIST Randomness Testing for Round 1 AES Candidates
Security Technology Group - Information Technology Laboratory (NIST)
http://csrc.nist.gov/encryption/aes/round1/conf2/NIST-randomness-testing.pdf

[2] TEA, a Tiny Encryption Algorithm
D. Wheeler and R. Needham
Fast Software Encryption Proceedings, Springer-Verlag, 1995, pp. 97-110.

[3] Related-Key Cryptanalysis of 3-WAY, Biham-DES,CAST, DES-X, NewDES, RC2, and TEA
John Kelsey, Bruce Schneier, David Wagner
ICICS '97 Proceedings, Springer-Verlag, November 1997, pp. 233-246.

[4] Mod *n* Cryptanalysis, with Applications against RC5P and M6
J. Kelsey, B. Schneier, and D. Wagner
Fast Software Encryption Proceedings, Springer-Verlag, 1999, pp. 139-155.