FPGA's as Cryptanalytic Tools

Rouvroy Gael, Standaert Francois-Xavier

UCL Crypto Group
Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium
{rouvroy,standaert}@dice.ucl.ac.be

Abstract

This paper presents FPGA[1] implementations of two cryptanalytic attacks against DES[2]. Linear cryptanalysis results from Matsui's work [2] but could not be applied as such and had to be modified to face hardware constraints. We broke a key in about 14 hours on one single FPGA[3], becoming the fastest implementation to our knowledge. In parallel, we evaluated the possibility of a cryptanalytic time-memory tradeoff using distinguished points. The original idea from Hellman [3] has never been implemented. We performed first experimental results and designed a machine that can break a 40-bit DES in about 15 seconds, with a high success rate (72%), using one PC[4]. An exhaustive search of the key on the same PC would have taken about 50 days.

1 Introduction

One objective of cryptography is to solve the confidentiality problem that can be explained as follows. Suppose that someone wants to send a message to a receiver, and wants to be sure that noone else can read the message. However, there is the possibility that someone else opens the letter or hears the electronic communication. In cryptographic terminology, the message is called a plaintext. Encoding the contents of the message in such a way that it hides its contents from outsiders is called encryption. The encrypted message is called ciphertext. The process of retrieving plaintext from ciphertext is called decryption. Encryption and decryption usually make use of a key, and the coding method is such that decryption can be performed only by knowing the proper key.

The Data Encryption Standard is an encryption/decryption algorithm developed in the mid-1970s. It was turned into a standard by the US National Institute of Standards and Technology and was also adopted by several other governments worldwide. It was and still is widely used in the financial industry, even though DES is supposed to be replaced by AES[5] for the next decade. Some Internet protocols still uses DES with a 40-bit key and even today, there are no cryptanalytic techniques that would completely break DES in a structural way. Indeed, the only real weakness known is the short key size.

Cryptanalysis is the science of breaking ciphers. In this work, we focused on cryptanalytic techniques to break the DES cipher.

2 Overview of Cryptanalytic Techniques

Generally speaking, a block cipher allows to encrypt a n-bit plaintext, using a k-bit key to produce a n-bit ciphertext. Cryptanalysis is equivalent to the searching problem of finding the correct secret key K in a set of 2^k possible keys and allows two extreme solutions: exhaustive search and table lookup. In exhaustive search, the ciphertext can be deciphered under each key and the result compared with the known plaintext. If they are equal,

[1] FPGA: Field Programmable Gate Array

[2] DES: Data Encryption Standard

[3] VIRTEX1000BG560

[4] 256MbytesRAM/350MHz

[5] AES: Advanced Encryption Standard

* This work has been funded by the Walloon region (Belgium) through the research project TACTILS.

the key tried is probably correct. Occasional false alarms can be rejected by additional tests. In table lookup, the cryptanalyst first enciphers some fixed plaintext P_0 under all possible keys to produce 2^k ciphertexts. These are sorted and stored in a table with their associated key. When a user chooses a new key, he provides (in a chosen plaintext attack) the cryptanalyst with the encipherment of P_0:

$$C_0 = E_K(P_0) \tag{1}$$

Where $E_K(*)$ denotes the enciphering operation under key K. Because the table is sorted by ciphertexts, the cryptanalyst can find C_0 and its associated key in at most $log_2 N$ operations using a binary search. The 2^k operations required to compute the table are not counted here because they constitute a precomputation task that can be performed beforehand. However, we must ensure that the precomputation is not excessive.

The objective of linear or differential cryptanalysis is to recover some key-bits in less operations than an exhaustive key search over all possible keys. Linear cryptanalysis takes advantage of possible input-output correlations over a few rounds of an algorithm[6] and differential cryptanalysis is possible if there are predictable difference propagation over a few rounds of the algorithm.

Finally, the aim of a time-memory tradeoff is to mount an attack which has a lower online processing complexity than exhaustive search and a lower memory complexity than table lookup. In the next sections, we detailed first hardware implementations of linear cryptanalysis and time-memory tradeoff. Other attacks exists and are out of the scope of this work.

3 Our Implementation of DES

DES is a block cipher with 64-bit block size and 56-bit keys. As the understanding of some parts of the algorithm are necessary in the next sections, we give a short description of it. A complete description can be found in [5]. The algorithm proceeds in three steps:

1. The given plaintext P_0 is divided in two parts of 32 bits according to an initial permutation IP: $IP(P_0) = L_0 R_0$.

2. 16 iterations of a round function are computed and sixteen keys $K_1, ..., K_{16}$, each bit strings of length 48 are derived from the key K.

3. The inverse permutation IP^{-1} is applied to the bit string $L_{16}R_{16}$, obtaining the ciphertext C_0.

The key point of the algorithm is the round function illustrated by Figure 1 where F is a non-linear function and the symbol $\oplus$ is the bitwise XOR operation. The non-linear function is divided into different steps:

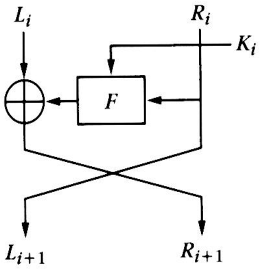

Figure 1: DES encrypt round

1. Permutations of the bits

2. Addition of the key K_i using the $\oplus$ operation.

3. Non-linear substitutions represented by s-boxes.

Our hardware implementation of DES is fully pipelined which means that we introduced registers between every round function. It allows to encrypt one plaintext in one clock step. Therefore, the encryption rate only depends on the work frequency. We carried out our experiments on a VIRTEX1000BG560 FPGA and reach a work frequency of 66.6 MHz. Moreover, we could fit 5 DES blocks on one FPGA. It means a final encryption rate of $5 \times 66.6 MHz = 333\ Mencryptions/sec$ or $5 \times 66.6 MHz \times 64 bits = 21\ Gbits/sec.$

[6] Encryption/decryption algorithms usually uses consecutive identical rounds

4 Linear Cryptanalysis

Linear cryptanalysis is a cryptanalytic technique that takes advantage of eventual input-output correlations over a few rounds of an algorithm. DES presents this interesting property and therefore is susceptible to be broken by a linear cryptanalysis attack. In its basic version, linear cryptanalysis is a known plaintext attack. The purpose of this method is to obtain a linear approximation of the DES.

To find this expression, we looked for boolean relations between inputs and outputs of s-boxes with a probability as different as possible from $\frac{1}{2}$. The approach chosen by Matsui was to investigate the probability that a XOR relation between input bits coincides with a XOR relation between output bits, for every s-box. After an empiric search of all the possible linear relations, Matsui found the best relation $x_2 = y_1 \oplus y_2 \oplus y_3 \oplus y_4$ for s-box 5[7] that holds with probability $\frac{12}{64}$. Other relations were found with worse probabilities.

The next step was to extend these relations to one round and finally to combine them in order to get a linear approximation of DES, involving plaintext bits, key bits and ciphertext bits, with the probability that this approximation holds. Matsui found:

$$\begin{aligned} P_H[7,18,24] \oplus F_1(P_L, K_1)[7,18,24] \oplus \\ C_H[15] \oplus C_L[7,18,24,29] \oplus \\ F_{16}(C_L, K_{16})[15] = K_3[22] \oplus \\ K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus \\ K_8[44] \oplus K_9[22] \oplus K_{11}[22] \oplus \\ K_{12}[44] \oplus K_{13}[22] \end{aligned} \tag{2}$$

Where P_H and P_L are plaintext bits, C_H and C_L are ciphertext bits and K_i are key bits. Moreover, $F_1(P_L, K_1)[7,18,24]$ denotes the first F function and $F_{16}(C_L, K_{16})[15]$ the last one. Both F functions involves 6 key bits.

According to Matsui's article, this equation holds with probability $\frac{1}{2} + 1.19 \times 2^{-21}$. In order to recover the 12 key bits involved in F_1 and F_{16}, we will compute equation 2 for all the $2^6 \times 2^6 = 4096$ possible keys and a large number of plaintexts. Knowing that only one of the keys is correct, there is one of the 4096 equations that will hold more significantly, corresponding to the correct key. The following algorithm summarizes this idea:

Algorithm

1. For each candidate K_n^i (i=1,2,...4096) of K_n, let T_i be the number of plaintexts such that the left side of the equation (7.12) is equal to zero.

2. Let T_{max} be the maximal value, T_{min} the minimal value of all T_i's and N the number of plaintexts tried.
   If $|T_{max} - \frac{N}{2}| > |T_{min} - \frac{N}{2}|$, then adopt the key candidate corresponding to T_{max}.
   If $|T_{max} - \frac{N}{2}| < |T_{min} - \frac{N}{2}|$, then adopt the key candidate corresponding to T_{min}.

The success rate of the algorithm depends on the number of plaintexts tried and the probability that equation 2 holds. Approximately, if the probability that the linear approximation holds is $\frac{1}{2} + 1.19 \times 2^{-21}$, we will need $\simeq (2^{21})^2 = 2^{42}$ plaintexts to recover the key bits.

Implementation problems: The hardware requirements of this algorithm depends on the number of times equation 2 has to be computed, corresponding to a number of counters. It was practically impossible to fit 4096 counters on our FPGA and therefore, we had to modify equation 2 to face hardware constraints.

Looking back at equation 2, we see that the counters are introduced by terms F_1 and F_{16}, each of which depends on 6 key bits. If we could force one of these terms (say F_1) to a constant value, we would get rid of 2^6 counters. Due to the non-linear character of F_1, the only way to force it at a constant value is to fix its inputs. As the key bits are obviously constant, all we have to do is to fix 6 bits of P_L to a constant value. As a consequence, our attack becomes a chosen-plaintext attack. Finally, we implemented the following equation:

$$\begin{aligned} P_H[7,18,24] \oplus F_{16}(C_L, K_{16})[15] \oplus \\ C_H[15] \oplus C_L[7,18,24,29] = \\ F_1(P_L, K_1)[7,18,24] \oplus K_3[22] \oplus \end{aligned}$$

[7] DES has 8 s-boxes mapping 6 input bits on 4 output bits

$$K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus K_9[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \qquad (3)$$

Where the right term is a constant.

Experimental results: We carried out the experiments on a single Xilinx FPGA, at a work frequency = 66.6 MHz (=2^{26}) and we parallelized 6 attack blocks. Therefore, we were able to compute 6×2^{26} equations per second and 2^{42} evaluations of equation 3 took about 3 hours. We performed tests with 50 different keys and the next table summarizes the experimental success rate for different amounts of chosen-plaintexts/ciphertexts pairs:

N	2^{38}	2^{39}	2^{40}	2^{41}	2^{42}
Success rate	4%	8%	14%	33%	77%

Completing the attack: Once these 6 key bits of information have been obtained, the question becomes: ”How can they be exploited to obtain the complete key?”. Since the 6 bits recovered belong to the 12 involved in equation 2, we can now implement this equation with 6 key bits fixed. This would require 2^6 counters, which is clearly achievable by the FPGA. Applying the same treatment to the dual equation (where we just invert the way we cover the rounds) would provide another 12 bits. Therefore, we could find 24 bits in about 12 hours. The exhaustive search of the remaining 32 bits would take about 20 seconds.

Conclusion: We implemented a first FPGA implementation of linear cryptanalysis. Due to hardware constraints, the attack had to be adapted to make it less memory-consuming. The resulting design is deployed on reasonably expensive hardware (3500$) and is capable to break a full DES key in 12-15 hours, including a final exhaustive search.

5 Time-Memory Tradeoff

In 1980, Hellman introduced the concept of cryptanalytic time-memory tradeoff, which allows the cryptanalysis of any N key symmetric cryptosystem in $O(N^{\frac{2}{3}})$ operations with $O(N^{\frac{2}{3}})$ storage, provided a precomputation of $O(N)$ is performed beforehand. This idea never led to realistic implementations. This section refers to Rivest [4] who introduced the idea of a time-memory tradeoff using distinguished points. We present the first experimental results of a cryptanalytic time-memory tradeoff using distinguished points in the context of a chosen-plaintext attack against a 40-bit DES.

Basic scheme: The time-memory tradeoff method for breaking ciphers is composed of a precomputation task and an online attack. We briefly introduce these steps with two intuitive schemes.

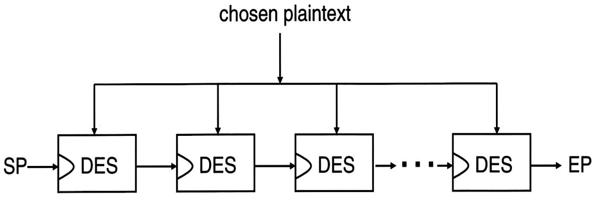

Figure 2: Precomputation

1. A chain is formed by a number l of encryptions using a chosen plaintext and l different keys. A defined property holds for the first and last keys and we call them distinguished points (DP). During the precomputation, we compute a number of chains and store start points (SP), end points (EP) and the corresponding length of chains in a table.

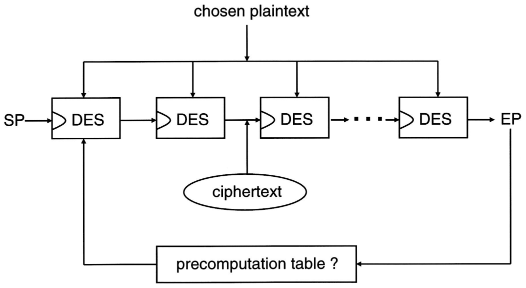

Figure 3: Online attack

2. Let the chosen plaintext be encrypted with a secret key. During the attack, we can use the resulting ciphertext as a key and start a chain until finding a distinguished point. Then, we check if this end point is in our table, take the

corresponding start point and restart chaining until we find the ciphertext again. The secret key is its predecessor in the computed chain.

This basic scheme illustrates that the success rate of the attack depends on how well the computed chains "cover" the key space. In order to improve this cover, we made use of different mask function that mixes the output bits of DES, allowing different covers of the key space. Let r be the number of different mask functions.

Stategy: In the tradeoff method, one tries to store information about as many different keys as possible by taking as long chains as possible. However, different critical overlap situations can appear during the precomputation and add constraints to the tradeoff.

1. A chain can cycle. This is the case where we find an already computed key of the chain before finding a distinguished point.
2. Two chains computed with the same mask function can merge. This is the case where two start points give two different chains with the same image. This means that from the moment of the merge until the end of at least one chain, both chains contains the same keys.
3. A chain can collide with another chain computed with a different mask function. This is the situation where two chains computed with different mask functions have some common points between SP and EP. It means that some keys are stored several times, which is not efficient.

We considered different solutions to deal with overlaps.

1. A cycle would produce a loop during the precomputation. We reject them by determining a maximum length for the chains.
2. Mergers are processed after the precomputations. If the same DP is an end point for different chains, then only the longest chain will be stored. Consequently, the effectiveness of the precomputation decreases when the number of triples already stored raises. Moreover, long chains are more involved in mergers than short chains. From this, we can reduce overlaps caused by mergers by decreasing the length of chains and number of chains stored. Therefore, we took different mask functions in order to fulfil the following condition:

$$r \times Nbr.Chains.Stored \times Average.length.of.chains = 2^{40} \tag{4}$$

3. Finally, collisions can not be rejected by a processing on the triples stored. The amount of collisions only appears in the final success rate of the algorithm. Therefore, we took a security coefficient in precomputation task and modify equation 4:

$$r \times Nbr.Chains.Stored \times Average.length.of.chains \geq 2^{40} \tag{5}$$

Implementation: a hardware/software co-design. The implementation choices were enforced by precomputation task and online attack. Obviously, the online attack had to be efficient on every PC and therefore is being dealt with in the software part. On the other hand, we performed the precomputation task with an optimal usage of the FPGA considering its limited size. It led us to carry out some parts of the precomputation by software like the sort on end points. We also wanted our hardware circuit to be parametric in order to change the tradeoff parameters by software. Therefore, some tasks are hardware implemented with a software control (SWC). The next list summarizes the hardware vs software design decisions.

Task	HW	SW	SWC
SP generation	X	-	-
DES chaining	X	-	-
Mask functions	X	-	X
Rejection of long chains	X	-	X
Rejection of short chains	X	-	X
DP detection	X	-	-
Length computation	X	-	-
Triples storage	-	X	-
Sort on EP	-	X	-
Merger rejection	-	X	-
Online attack	-	X	-

Results: The DES encrypts a 64-bit plaintext using a 56-bit secret key. We defined a 40-bit DES by fixing 16 key bits to arbitrary values and propose a chosen-plaintext attack to recover the 40 bits of the secret key.

Using our hardware implementation, we succeeded in storing 2^{20} chains with an average length of $2^{10.47}$ for every mask function and therefore decided to fulfil condition 5 by taking 2^{10} different mask functions. In terms of precomputation time, we ran the FPGA during about one week. In terms of memory requirements, we stored all chains on 16 CDROM's[8], corresponding to 16 sets of 2^6 mask functions.

Concerning the online attack, we first define a theoretical key space cover ($TKSC$), a practical key space cover ($PKSC$) and a practical success rate (SR) in terms of i, the number of CDROM's used.

$$TKSC(i) = i \times 2^{36.47} \tag{6}$$

$$PKSC(i) = \frac{Number.of.keys.found(i)}{Number.of.keys.tried(i)} \times 2^{40} \tag{7}$$

$$SR(i) = \frac{Number.of.keys.found(i)}{Number.of.keys.tried(i)} \tag{8}$$

The next tables summarizes the online attack results. Differences between $TKSC$ and $PKSC$ are due to the collision problem.

Nbr of CDROM's	SR
1	8.6%
2	17%
4	31.4%
8	49.7%
16	72%

Nbr of CDROM's	TKSC	PKSC
1	$2^{36.47}$	$2^{36.47}$
2	$2^{37.47}$	$2^{37.44}$
4	$2^{38.47}$	$2^{38.33}$
8	$2^{39.47}$	$2^{38.99}$
16	$2^{40.47}$	$2^{39.53}$

The online attack was performed on a single PC[9] and we recovered one key in about 10 seconds with a success rate of 72%. An exhaustive search of the key on the same PC would have taken 50 days.

Conclusion: We performed a first implementation of a time-memory tradeoff using distinguished points and presented experimental results that confirm its effectiveness in cryptanalytic contexts. The resulting chosen-plaintext attack significantly improves all existing complete cryptanalytic techniques attempted against DES in terms of speed. The method is general an could be applied to other block ciphers or one-way functions. Note that time-memory tradeoff attacks can be dangerous even when the key space is too large to be exhaustively precomputed (say 2^{80}). Consider an application where immediate inversion of a single cipher can be disastrous (e.g. an online bank transfer), then, constructing tables that would cover "only" 2^{60} keys would allow online inversion with probability 2^{-20}, which is not negligible.

6 Conclusion

In cryptanalytic applications, FPGA's can be used as dedicated computers in order to perform some computations at very high frequencies. Consequently, they speed up the attacks and improve their effectiveness, with some additional constraints that can be solved either by modifying algorithms or by hardware/software co-designs.

Practically, we evaluated two hardware implementations of cryptanalytic attacks and both cases improved existing results. We performed the fastest implementation of the linear cryptanalysis and the first experimental results of a time-memory tradeoff using distinguished points.

References

[1] Mitsuru Matsui: *Linear Cryptanalysis Method for DES Cipher*, EUROCRYPT93 : 386-397.

[2] Mitsuru Matsui: *The First Experimental Cryptanalysis of the DES*, CRYPTO94 : 1-11.

[3] M.Hellman, *A Cryptanalytic Time-Memory Tradeoff*, IEEE transactions on Information Theory, Vol 26, 1980, pp.401-406.

[4] D.Denning, *Cryptography and Data Security*, p.100, Addison-Wesley, 1982, Out of Print.

[5] Douglas R. Stinson: *Cryptography, Theory and Practice*, CRC press, 1995.

[6] Dieter Gollman: *Computer Security*, Wiley, 1999.

[7] Xilinx: *Virtex 2.5V Field Programmable Gate Arrays Data Sheet*, http://www.xilinx.com.

[8] 650Mbytes

[9] 18Gbytes memory/256MbytesRAM/350MHz