A Cryptanalytic Time-Memory Tradeoff: First FPGA Implementation

Quisquater Jean-Jacques, Standaert Francois-Xavier,
Rouvroy Gael, David Jean-Pierre, Legat Jean-Didier
{quisquater,standaert,rouvroy,david,legat}@dice.ucl.ac.be

UCL Crypto Group
Laboratoire de Microelectronique
Universite Catholique de Louvain
Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium

Abstract. A cryptanalytic time-memory tradeoff allows the cryptanalysis of any N key symmetric cryptosystem in $O(N^{\frac{2}{3}})$ operations with $O(N^{\frac{2}{3}})$ storage, if a precomputation of $O(N)$ operations has been done in advance. This procedure is well known but did not lead to any realistic implementations. In this paper, the experimental results for the cryptanalysis of DES[1] that are presented are based on a time-memory tradeoff using distinguished points, a method which is referenced to Rivest [2]. For this task, a fast hardware implementation of DES was designed using FPGA technology. The target is a 40-bit DES which is obtained from DES by fixing 16 key bits to arbitrary values. The precomputation task is performed with a purpose-built FPGA design, whereas the search algorithm corresponding to the online attack is reported to be feasible on any PC within about 10 seconds, with a success rate of 72%. The cost of an expansion to 56-bit DES is evaluated.

1 Introduction

Generally speaking, a block cipher allows to encrypt a n-bit text using a k-bit key to produce a n-bit ciphertext. Let $q = \lceil \frac{k}{n} \rceil$. If q plaintext/ciphertext pairs are known, with a high probability, the key can be determined by exhaustive key search, but it requires a too long processing time. Another possibility is a chosen plaintext attack using a precomputation table where an attacker precomputes the encryptions of q chosen plaintexts under all possible keys and stores the corresponding ciphertext/key pairs, but it requires a too large memory. The aim of a time-memory tradeoff is to mount an attack which has a lower online processing complexity than exhaustive key search and lower memory complexity than a precomputation table.
In this paper, we present a cryptanalytic time-memory tradeoff in the context of a chosen plaintext attack on DES. In this way, we designed fast hardware implementations of DES using FPGA technology. Basically, the programmable

[1] DES : Data Encryption Standard

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

hardware device was used as a dedicated computer in order to perform some precomputation tasks (like cryptographic encryption and chaining) at very high frequencies. We designed a machine that can break a 40-bit DES block cipher in about 10 seconds, using one PC[2], with a high success rate (72%). An exhaustive search of the key on the same PC would have taken about 50 days. We also evaluate the hardware cost of an expansion to a 56-bit DES block cipher. Finally, this paper underlines the efficiency of FPGA's in cryptanalytic applications in terms of brute computational power.

2 Definitions

Let E : $\{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n$ be a block cipher with block length n and key length k. The encryption of one block is written as:

$$C = E_K(P) \tag{1}$$

Where $C \in \{0,1\}^n$, $K \in \{0,1\}^k$ and $P \in \{0,1\}^n$ denote the ciphertext, the secret key and the plaintext.
We define two functions. The first one just mixes its arguments and rejects z bits to reach the key size $k = n - z$. $g : \{0,1\}^n \rightarrow \{0,1\}^k$.

$$g(C) = g(C_1, C_2, ..., C_n) = (C_{perm(1)}, C_{perm(2)}, ..., C_{perm(n-z)}) \tag{2}$$

Where $C \in \{0,1\}^n$ is the ciphertext and *perm* denotes a simple permutation. We call g a mask function. There are many possibilities to define g, namely $(n-z)!$ We also define a function $f : \{0,1\}^k \rightarrow \{0,1\}^k$

$$f(K) = g(E_K(P)) \tag{3}$$

Finally, for a random start point $SP \in \{0,1\}^k$, we define a chain $K_0, K_1, K_2, ..., K_t$ of length t as

$$K_0 = SP \tag{4}$$

$$K_i = f(K_{i-1}) = f^i(K_0) \tag{5}$$

Definition of a DP-property: Let $\{0,1\}^k$ be the key space and $d \in$ {1,2,3,...,k-1}. Then $DP - d$ is a DP-property of order d if there is an easily checked property which holds for 2^{k-d} different elements of $\{0,1\}^k$. In our application, having d bits locked to a fixed value, say 0, is a DP-property of order d.

Definition of a distinguished point: Let $K \in \{0,1\}^k$ and $d \in$ {1,2,3,...,k-1}. Then K is a distinguished point (DP) of order d if the DP-property defined beforehand holds for K. Note that using this definition of distinguished point, we do not need to store the fixed bits and reduce the memory requirements of the tradeoff.

[2] 256Mbytes RAM/350MHz

3 Algorithms

The algorithm proposed requires the choice of a DP-property of order d and a maximum chain length t. We precompute r tables by choosing r different mask functions. For each mask function m different start points (DP) will be randomly chosen. For each start point a chain will be computed until a DP is encountered or until the chain has length $t+1$. Only start points that iterate to a DP in less than t iterations will be stored with the corresponding chain length, the others will be discarded. Moreover, if the same DP is an end point for different chains, then only the chain of maximal length will be stored.

Precomputation algorithm: Generate r tables with (SP,EP,l)-triples, sorted on EP.

1. Choose a DP-property of order d.
2. Choose r different mask functions g_i, $i = 1, 2, ..., r$. It defines r different f functions: $f_i = g_i(E_K(P))$, $i = 1, 2, ..., r$.
3. Choose the maximum chain length t.
4. For $i = 1$ to r
   (a) Choose m random start points $SP_1^{(i)}, SP_2^{(i)}, ..., SP_m^{(i)}$.
   (b) For $j = 1$ to m, $l = 1$ to t
      i. Compute $f_i^l(SP_j^{(i)})$.
      ii. If $f_i^l(SP_j^{(i)})$ is a DP then store the triple $(SP_j^{(i)}, EP_j^{(i)} = f_i^l(SP_j^{(i)}), l)$ and take next j.
      iii. If $l > t$ "forget" $SP_j^{(i)}$ and take next j.
   (c) Sort triples on end points. If several end points are identical, only store the triple with the largest l.
   (d) Store the maximum l for each table: l_{max}^i.

For the search algorithm, a table only has to be accessed when a DP is encountered during an iteration. Moreover, if the encountered DP is not in the table, then one will not find the target key by iterating further. Hence the current search can skip the rest of this table.

Search algorithm: Given $C = E_K(P)$ find K.

1. For $i = 1$ to r
   (a) Look up l_{max}^i.
   (b) $Y = g_i(C)$.
   (c) For $j = 1$ to l_{max}^i
      i. If Y is a DP then
         A. If Y in table i, then
            – Take the corresponding $SP^{(i)}$ and length l in the table.
            – If $j < l$
               • Compute predecessor $\tilde{K} = f_i^{l-1-j}(SP_l^{(j)})$.
               • If $C = E_{\tilde{K}}(P)$ then $K = \tilde{K}$: STOP.
               • If $C \neq E_{\tilde{K}}(P)$, take next i.
         B. Else take next i.
      ii. Set $Y = f(Y)$.

4 Hardware description

All our experiments were carried out on a Virtex1000BG560-4 FPGA board, developed by DICE[3]. The board is composed of a control FPGA (FLEX 10K) and a VIRTEX1000 FPGA associated with several processors (ARM and PIC) and fast access memories. We used a PCI[4] to communicate between the PC and the FPGA.

5 Implementation: a hardware/software co-design

The implementation choices were enforced by both precomputation and search algorithms. Obviously, the search algorithm corresponds to an online attack that we want to be efficient on any 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 EP. 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. The next list summarizes the hardware vs software design decisions.

Task	HW	SW	SW controlled
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	-

In a cryptanalytic point of view, the main advantage of FPGA's is to parallelize algorithms in order to reach very high encryption rates. Practically, we implemented DES in the following way:
DES is a block cipher with 64-bit block size and 56-bit keys. The algorithm proceeds in three steps:

1. The given plaintext P_0 is divided into two parts of 32 bits according to an initial permutation $IP : IP(P_0) = L_0R_0$.
2. 16 iterations of a round function are computed and sixteen keys $K_1, K_2, ..., K_{16}$, each bit strings of length 48 are derived from the initial key K.

[3] Microelectronics laboratory, Universite Catholique de Louvain, Belgium
[4] Parallel Computer Interface

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

A fast implementation can be obtained by inserting registers between each round of the algorithm as suggested by Figure 1. The following table summarizes the

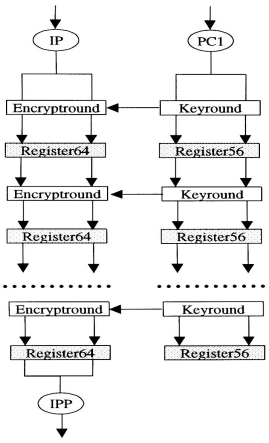

Fig. 1. Fast DES implementation

practical results of our implementation:

	Number of LUT	Work Frequency
Pipeline DES	4736	66 MHz

Finally, the start point of our hardware design (Figure2) is a pipeline DES which runs at 66M encryptions/second. As a plaintext is encrypted in 16 clock edges, we deal simultaneously with 16 start points: $K1, K2, ..., K16$. Initially, they do not need to be different because each start point has its own mask function. The design of Figure 2 computes triples for $16{=}2^4$ different mask functions. We parallelized 4 units on the FPGA which corresponds to $64{=}2^6$ different mask functions.
The triples computed by the FPGA are communicated to the PC via a FIFO memory.

6 Cryptanalysis of a 40-bit DES block cipher: experimental results

The Data Encryption Standard encrypts a 64-bit plaintext using a 56-bit secret key. We define 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.

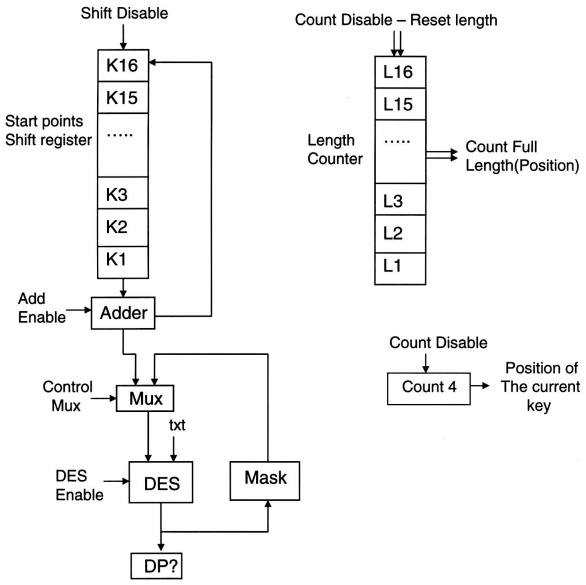

Fig. 2. Precomputation design

6.1 Precomputation task

The objective of the precomputation task is to reach the condition:

$$r \times Nbr.Triples.Stored \times Average.chain.length \geq 2^{40} \tag{6}$$

We define three quantities to evaluate the precomputation task:

1. Let $E =$ (Number of triples stored after sort and mergers rejection)/(Number of triples computed) be the effectiveness of the precomputation task.
2. Let $ALBS$ be the average chain length before sort and mergers rejection.
3. Let $ALAS$ be the average chain length after sort and mergers rejection.

The tradeoff parameters should be chosen such as to avoid the following situations:

- A saturation phenomenon. After the computation of an amount of triples, mergers of chains become critical and the effectiveness of the computation decreases.
- A variation of the average length of chains: $ALBS \gg ALAS$.

Consequently, we performed a heuristic evaluation of the parameters and defined a set of adequate values.

1. DP-property: $d = 11$.

2. Length of chains: $2^9 < l < 2^{13}$.

We succeeded in storing $2^{19.5}$ triples with an average length of $2^{10.97}$ for every mask function and therefore decided to fulfil condition (6) by taking 2^{10} different mask functions. In terms of precomputation time, we ran the FPGA with 16 different configurations because we could only fit 2^6 mask functions on one FPGA. It took us about one week. In terms of memory requirements, we stored all triples on 16 CDROM's[5] corresponding to 16 sets of 2^6 mask functions. Finally, we give the precomputation results.

Number of triples stored	$2^{19.5}$
Average length of chains	$2^{10.97}$
Number of mask functions	2^{10}
Memory requirements	16 CDROM's
Key space cover of one CDROM	$2^{36.47}$

6.2 Online attack

In order to evaluate the practical success rate of our attack and the amount of overlaps between mask functions, we define the following quantities:
Let the theoretical key space cover be defined in terms of memory usage as:

$$TKSC = Number.of.CDROM's \times 2^{36.47} \tag{7}$$

It corresponds to the cover that we would have observed without any overlap between the chains computed with different mask functions.
If i is the number of CDROM's used for the online attack, let the practical key space cover be defined in function of the experimental success rate of the attack:

$$PKSC(i) = \frac{Number.of.keys.found(i)}{Number.of.keys.tried(i)} \times 2^{40} \tag{8}$$

Differences between $TKSC$ and $PKSC$ are due to the overlap problem and we define an overlap factor such as:

$$TKSC = PKSC \times Overlap.factor \tag{9}$$

Finally, we define the experimental success rate of the attack as:

$$SR(i) = \frac{Number.of.keys.found(i)}{Number.of.keys.tried(i)} \tag{10}$$

We summarize the online attack results:

Nbr of CDROM's	Nbr of keys tried	Nbr of keys found	SR
1	3000	260	8.6%
2	3000	510	17%
4	3000	941	31.4%
8	3000	1490	49.7%
16	3000	2160	72%

[5] 650Mbytes

Nbr of CDROM's	TKSC	PKSC	Overlap factor
1	$2^{36.47}$	$2^{36.47}$	1
2	$2^{37.47}$	$2^{37.44}$	1.02
4	$2^{38.47}$	$2^{38.33}$	1.1
8	$2^{39.47}$	$2^{38.99}$	1.39
16	$2^{40.47}$	$2^{39.53}$	1.92

The online attack was performed on a single PC[6] and we recovered one key in about 10 seconds with a success rate of 72%. An exhaustive key search on the same PC would have taken about 50 days. Remark that the overlap factor denotes a saturation problem when multiplying the number of mask functions.

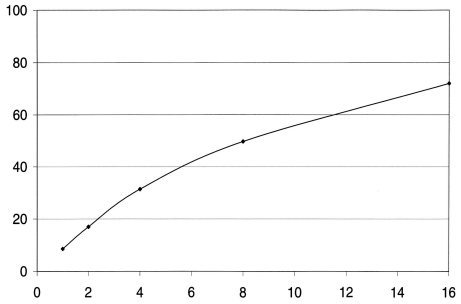

Fig. 3. Success rate (%) in terms of memory usage (CDROM's).

7 Expansion to a 56-bit DES block cipher

In order to evaluate the possibility of a cryptanalytic tradeoff on a complete DES block cipher (with a 56-bit key), we modified our hardware design and performed a sample of the precomputation task with the following parameters:

1. DP-property: $d = 18$.
2. Chain lengths: all.
3. Storage until saturation.
4. Number of mask functions: $r = 16$.

Figure 4 illustrates the saturation phenomenon encountered in the precomputation task which restricts the number of triples stored to 2^{22}. It corresponds to 5 days of computation. As the average chain length after sort was $\simeq 2^{18}$, we concluded that 2^{16} mask functions are necessary to fulfil the condition:

$$r \times Nbr.Triples.Stored \times Average.length.of.chains \simeq 2^{56} \qquad (11)$$

[6] 18Gbytes memory/256Mbytes RAM/350MHz

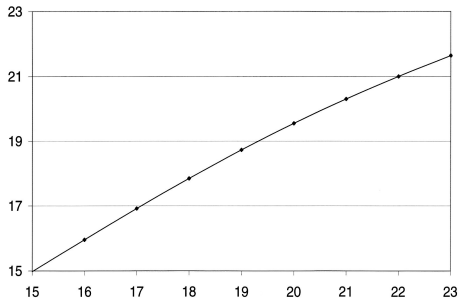

Fig. 4. X = Triples computed in log_2 scale - Y = Triples stored in log_2 scale.

However, a key space cover of 2^{46} only needs 2^6 mask functions and therefore 5 $\times$ 5 = 25 days of computation. It should allow us to reach a success rate close to $\frac{1}{2^{10}} \simeq 0.1\%$.
Finally, the next list indicates the possibilities to increase the precomputation power by using larger and faster FPGA's.

Family	Device	System gates	Internal clock
VIRTEX 2.5V FPGA	XCV1000	1M	200 MHz
VIRTEX-E 1.8V FPGA	XCV3200E	4M	300 MHz
VIRTEX-2 1.5V FPGA	XC2V8000	8M	400 MHz

For example, a VIRTEX-2 FPGA with 8M gates would certainly allow to gain a factor 16 in the precomputation time. Consequently, a parallelization of several VIRTEX-2 FPGA's would allow us to reach significant success rates (1% to 10%).
Concerning the online attack, the processing complexity roughly depends on the product: $ALAS \times r$. Then we can compare different attacks:

1. DES-40 with cover = 2^{40} : Complexity $\simeq 2^{10.5} \times 2^{10} = 2^{20.5}$.
2. DES-56 with cover = 2^{46} : Complexity $\simeq 2^{18} \times 2^6 = 2^{24}$.
3. DES-56 with cover = 2^{56} : Complexity $\simeq 2^{18} \times 2^{16} = 2^{34}$.

We conclude that an attack against DES-56 with 0.1% success rate is only 16 times slower than the attack developed in this paper against DES-40. Therefore, a software implementation of the attack is thinkable. However, high success rate attacks against DES-56 involve long processing times and should probably use a hardware coprocessor (for example a FPGA) for the distinguished points computation during the attack.

8 Conclusion

We performed a first implementation of a time-memory tradeoff using distinguished points and presented experimental results that confirm its effectiveness

in the context of block ciphers cryptanalysis. The resulting chosen plaintext attack significantly improves all existing complete cryptanalytic attacks attempted against DES or other block ciphers in terms of speed. Note that the attack is general and could be applied to any block cipher without changing algorithms. In case of a 40-bit DES and compared to exhaustive key search, we recover a key in 10 seconds instead of 50 days with a success rate of 72%.
Moreover, we evaluated the cost of a possible expansion to 56-bits DES. With our current equipment[7], a success rate of 0.1% is achievable. With recent FPGA's, we could gain a factor from 8 to 16 or more if the parallelization of several devices is considered. Anyway, the time-memory tradeoff attack can be dangerous even when the size of the key space is too large to be exhaustively searched (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} points would allow online inversion with probability 2^{-20}, which is not negligible.
Finally, this work confirms the effectiveness of FPGA's in cryptanalytic applications. Due to hardware constraints, some parts of the precomputation had to be software-implemented to make it less memory-consuming, but the resulting design is deployed on reasonably expensive hardware and allows a first implementation of a cryptanalytic time-memory tradeoff against a practical cipher. Future devices combined with high speed processors should reduce hardware constraints and permit the implementation of a variety of cryptanalytic attacks on FPGA's. In terms of cost and effectiveness, this could replace distributed computations in the coming years.

References

1. M.Hellman, *A Cryptanalytic Time-Memory Tradeoff*, IEEE transactions on Information Theory, Vol 26, 1980, pp.401-406.
2. D.Denning, *Cryptography and Data Security*, p.100, Addison-Wesley, 1982, Out of Print.
3. National Bureau of Standards. *Data Encryption Standard*, U.S.Department of Commerce, Washington DC, USA, January 1977.
4. J.Borst, B.Preneel and J.Vandewalle, *On the Time-Memory Tradeoff Between exhaustive key search and table precomputation*, Proc. of the 19th Symposium in Information Theory in the Benelux, WIC, 1998,
5. K.Kusuda and T.Matsumoto, *Optimization of Time-Memory Tradeoff Cryptanalysis and its Applications to DES, FEAL-32 and Skipjack*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, EP79-A, 1996, pp.35-48.
6. J.J.Quisquater, F.X.Standaert,G.Rouvroy,J.P.David,J.D.Legat, *A Cryptanalytic Time-Memory Tradeoff: First FPGA Implementation*, UCL Technical Report CG-2002/2, http://www.dice.ucl.ac.be/crypto/techreports.html
7. A.J.Menezes, P.C.van Oorschot and S.A.Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
8. Xilinx: *Virtex 2.5V Field Programmable Gate Arrays Data Sheet*, http://www.xilinx.com.

[7] We used only one VIRTEX1000BG560-4