A Time-Memory Tradeoff using Distinguished Points: New Analysis & FPGA Results

Standaert Francois-Xavier, Rouvroy Gael
Quisquater Jean-Jacques, Legat Jean-Didier
{standaert,rouvroy,quisquater,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. In 1980, Martin Hellman [1] introduced the concept of cryptanalytic time-memory tradeoffs, 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 procedure is well known but did not lead to realistic implementations. This paper considers a cryptanalytic time-memory tradeoff using distinguished points, a method referenced to Rivest [2]. The algorithm proposed decreases the expected number of memory accesses with sensible modifications of the other parameters and allows much more realistic implementations of fast key search machines. We present a detailed analysis of the algorithm and solve theoretical open problems of previous models. We also propose efficient mask functions in terms of hardware cost and probability of success. These results were experimentally confirmed and we used a purpose-built FPGA design to perform realistic tradeoffs against DES. The resulting online attack is feasible on a single PC and we recover a 40-bit key in about 10 seconds.

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 usually 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 usually requires a too large memory. The aim of a time-memory tradeoff is to mount an attack of which the online processing complexity is lower than an exhaustive key search and the memory complexity is lower than a precomputation table.
In [3–5], Borst et Al. propose a theoretical analysis of the time-memory tradeoff using distinguished points. They conclude that of theoretical interest remains the problem of determining two parameters: the expected number of chains and

average chain length after sort. In this paper, we present a theoretical analysis of a time-memory tradeoff using distinguished points and evaluate the different parameters introduced by this variant of Hellman's method. We discuss the complexity of the attack as well as the resulting probability of success, isolate the different phenomenons involved in a tradeoff using distinguished points and evaluate their practical influence. Precisely, we propose approximations to solve the open problem of [5] and correct the probability of success to correspond with practical implementations. The resulting analysis was confirmed by experimental results and allowed to mount realistic attacks against the block cipher DES.
The paper is organized as follows. In section 2, some basic schemes are given to help the understanding of the tradeoff. Formal definitions and algorithms are in section 3 and 4. Section 5 identifies the critical situations due to the use of distinguished points in the tradeoff. Section 6 proposes efficient mask functions in terms of success rate and hardware cost. The main contribution of this paper lies in section 7 and 8. We evaluate the different parameters of the tradeoff and compare the resulting theory with experimental results. Conclusions are in section 9.

2 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:

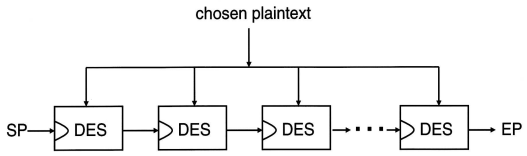

Fig. 1. Precomputation task

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. During the precomputation, we compute a number of chains and store start points, end points and the corresponding chain length in a table.
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 we find 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 the next sections, we develop these two schemes and present effective algorithms for precomputation and online attack.

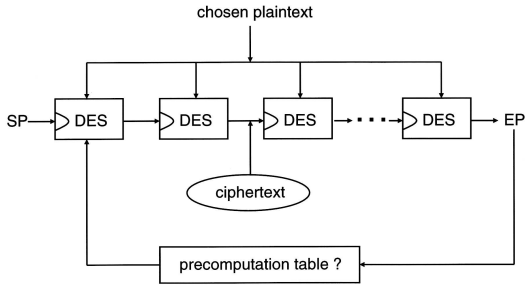

Fig. 2. Online attack

3 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 $k = n - z$.

$$g : \{0,1\}^n \rightarrow \{0,1\}^k. \tag{2}$$

We call g a mask function. There are many possibilities to define g. Earlier papers proposed to use permutations. In section 6, we suggest efficient mask functions in terms of implementation cost and probability of success.
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

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

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

In the tradeoff, only the start point SP and end point $EP = K_t$ are stored.

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.

4 Algorithms

The algorithm proposed requires to choose 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 (which are distinguished) will be randomly chosen. For each start point a chain will be computed until a DP is encountered or until the chain length is $t+1$. Only start points iterating 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. This involves a lower memory complexity than Hellman's tradeoff.

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 which allows efficient implementations of the online attack. 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 $Y = f(Y)$.

5 Overlap situations

In the tradeoff method described, one tries to store information about as many different keys as possible by taking as long chains as possible. Consequently, the very short chains increase the memory complexity of the tradeoff and could be rejected. However, different critical overlap situations can appear and add constraints to the tradeoff.

1. A chain can cycle. This is the case where we find i and j with $i \neq j$ and $K_{i+1} = K_{j+1}$.
2. Two chains computed with the same mask function can merge. This is the case where two different start points have the same image. This means that from the moment of the merge until the end of at least one chain, both chains contain 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.

As suggested by the precomputation algorithm, we dealt with cycles by choosing an adequate maximum chain length t and the mergers were rejected after the precomputation by keeping the longest of two merging chains. Consequently, the effectiveness of the tradeoff highly depends on the choice of its parameters. In the section 7, we evaluate the different parameters of the tradeoff as well as the resulting complexity and probability of success of both precomputation and online attack.

6 Efficient mask functions

In previous papers about time-memory tradeoffs, mask functions were implemented as permutations. We propose efficient mask functions in terms of collisions and implementation cost.
Basically, the problem when using a permutation is the possibility that two mask functions have some common keys after a collision. For example, if the mask function corresponds to a permutation of two bits, the common length after a collision is obviously $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ...$
Moreover, if a large number of mask functions is needed (which is practically

probable), their implementation becomes fastidious (specially in hardware designs that are particularly efficient in time-memory tradeoffs). We suggest the following definition of efficient mask functions. Let $m_i(x)$ and $m_j(x)$ be two mask functions:

$$m_i(x) : \{0,1\}^n \rightarrow \{0,1\}^n$$
$$m_j(x) : \{0,1\}^n \rightarrow \{0,1\}^n$$

Let $n(x) = m_i(x) - m_j(x)$. Efficient mask functions are such that for every $i \neq j$, we have

$$Ker(n(x)) = 0^1. \tag{6}$$

We define mask functions as:

$$m_i(x) : \{0,1\}^n \rightarrow \{0,1\}^n : x \rightarrow m_i(x) = x \oplus i \tag{7}$$

Where $\oplus$ denotes the bitwise XOR operator. These mask functions fulfill condition 6 and are specially easy to implement in hardware or software.

7 Theoretical analysis

As mentioned in section 5, the effectiveness of the tradeoff highly depends on the choice of its parameters. Consequently, their correct prediction before implementation is crucial. Although existing papers evaluate the processing complexity, memory complexity and success rate of the tradeoff and mention that length of chains, number of chains and number of mask functions play a crucial role in the performance of the tradeoff, there exist no precise indications about how to choose these parameters. Actually, only [3–5] focus on the distinguished points variant but their analysis is not complete and some problems remained open. In the next section, we propose a theoretical model for cryptanalytic time-memory tradeoffs using distinguished points that takes into account the new situations due to distinguished points. Approximations are proposed to solve the open problem of [5].

7.1 Probability to reach a distinguished point

The main modification caused by the introduction of distinguished points is the variable chain length. Consequently, we computed the probability to reach a distinguished point in less than l iterations. In the following computation, we guess that the maximum chain length is such that no cycle could appear. Let $P_1(l)$ be the probability that a DP is reached in less than ($\leq$) l iterations. Let $P_2(l)$ be the probability that no DP is reached in less than l iterations. We have $P_1 = 1 - P_2$ and we can easily compute $P_2(l)$:

$$P_2(l) = \prod_{i=0}^{l-1} (1 - \frac{2^{k-d}}{2^k - i}) \tag{8}$$

[1] $Ker(n(x)) = \{x \in \{0,1\}^n | n(x) = 0\}$

An approximate expression can be obtained knowing that $i \ll 2^k$, by fixing i to $\frac{l-1}{2}$:

$$P_2(l) \simeq (1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}})^l \tag{9}$$

Finally, we have

$$P_1(l) = 1 - \prod_{i=0}^{l-1}(1 - \frac{2^{k-d}}{2^k - i}) \tag{10}$$

$$P_1(l) \simeq 1 - (1 - \frac{2^{k-d}}{2^k - \frac{l-1}{2}})^l \tag{11}$$

Obviously, the probability to reach a distinguished point in exactly l iterations can be derived from $P_1(l)$ - $P_1(l-1)$ if we guess that $P_1(0) = 0$. Important information about the efficient areas of computation can be observed when representing the amount $l \times P$(A DP is reached in exactly l iterations) as shown by Figure 3 for the DP-10 property.

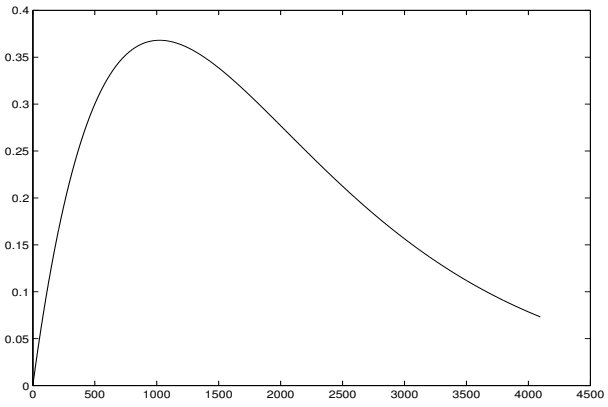

Fig. 3. X = l, Y=$l \times P$(A DP is reached in exactly l iterations)

7.2 Compute the average chain length β

Figure 3 suggests that practical precomputations should possibly be performed for an efficient interval of chain lengths. Therefore, we evaluate the average chain length in a region between lengths t_{min} and t_{max}:

$$\beta = \frac{\sum_{l=t_{min}}^{t_{max}} l.P(DP.in.exactly.l.iterations)}{\sum_{l=t_{min}}^{t_{max}} P(DP.in.exactly.l.iterations)} \tag{12}$$

The denominator is easy to estimate and corresponds to the quotient between the number of chains included in region $[t_{min}, t_{max}]$ and the total number of chains. We call it cover and denote it γ:

$$\gamma = \sum_{l=t_{min}}^{t_{max}} P(DP.in.exactly.l.iterations) = P_1(t_{max}) - P_1(t_{min} - 1) \tag{13}$$

Numerator can be estimated with the next formula:

$$\sum_{l=t_{min}}^{t_{max}} l.P(DP.in.exactly.l.iterations)$$

$$= \sum_{l=t_{min}}^{t_{max}} l.(\prod_{i=0}^{l-2}(1-\frac{2^{k-d}}{2^k-i}) - (\prod_{i=0}^{l-1}(1-\frac{2^{k-d}}{2^k-i}))$$

$$\simeq \sum_{l=t_{min}}^{t_{max}} l.((1-\frac{2^{k-d}}{2^k-\frac{t}{2}})^{l-2} - (1-\frac{2^{k-d}}{2^k-\frac{t}{2}})^{l-1}) \quad (14)$$

Where $t = \frac{t_{max}+t_{min}}{2}$. We can rewrite equation 14 in a simpler form:

$$\sum_{l=t_{min}}^{t_{max}} l.((1-x)^{l-2} - (1-x)^{l-1}) \quad (15)$$

Where $x = \frac{2^{k-d}}{2^k-\frac{t}{2}}$. Hence

$$\sum_{l=t_{min}}^{t_{max}} l.((1-x)^{l-2} - (1-x)^{l-1})$$

$$= t_{min}.(1-x)^{t_{min}-2} - t_{max}.(1-x)^{t_{max}-1} + \sum_{l=t_{min}-1}^{t_{max}-2}(1-x)^l$$

$$= (1-x)^{t_{min}-2}.(t_{min}+\frac{1-x}{x}) - (1-x)^{t_{max}-1}.(t_{max}+\frac{1}{x}) \quad (16)$$

Finally, the average chain length is:

$$\beta \simeq \frac{(1-x)^{t_{min}-2}.(t_{min}+\frac{1-x}{x}) - (1-x)^{t_{max}-1}.(t_{max}+\frac{1}{x})}{\gamma} \quad (17)$$

We designed some experiments in order to confirm this analysis and computed chains using a block cipher DES with a reduced key of 40 bits. First, we evaluated the influence of the DP-property:

DP-property	Length region (log_2)	Experimental β (log_2)	Theoretical β (log_2)
DP-11	9-13	11.2140	11.2140
DP-12	10-14	12.2137	12.2139
DP-13	12-14	13.0965	13.0966
DP-14	13-15	14.0967	14.0966
DP-15	11-18	15.0771	15.0836

Then, we observed the influence of the chain lengths:

DP-property	Length region (log_2)	Experimental β (log_2)	Theoretical β (log_2)
DP-13	10-13	11.9790	11.9987
DP-13	12-14	13.0965	13.0966
DP-13	13-16	14.0107	13.9955

7.3 Previous proposals for the success rate

In this section, we consider the success rate when using one table generated with one mask function and denote it SR. The probability of success when using several mask functions is evaluated later and we denote it PS. For all the following computations, the function f is modeled as a random function mapping the key set onto itself if the key K is randomly chosen. Actually, earlier evaluations of the success rate, [1, 7], do not consider the distinguished point variant and SR was always estimated in the following way: first the probability $P(K_{ij}$ is new) that a newly generated key K_{ij} is different from all keys generated previously is evaluated by:

$$P(K_{ij}.is.new) \geq (1 - \frac{it}{N})^{j+1} \tag{18}$$

Where i is the number of chains already computed, j the length of current chain, t the fixed chain length and $N = 2^k$ is the size of the key space. Then a lower bound of the success rate is evaluated:

$$SR \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} (1 - \frac{it}{N})^{j+1} \tag{19}$$

Where m is the number of chains computed. Using $e^{-x} \simeq 1 - x$, we have the following approximation:

$$\begin{aligned} (1 - \frac{it}{N}) &\simeq e^{\frac{-it}{N}} \\ (1 - \frac{it}{N})^{j+1} &\simeq e^{\frac{-ijt}{N}} \end{aligned} \tag{20}$$

Equation 20 indicates that for a fixed value of N, there is not much to be gained by increasing m and t beyond the point at which $mt^2 = N$. Because when $e^{\frac{-ijt}{N}} \simeq e^{\frac{-mt^2}{N}}$ and $mt^2 \gg N$, most terms will be small. Finally, we can evaluate the success rate:

$$\begin{aligned} SR &\geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} (1 - \frac{it}{N})^{j+1} \\ &\simeq \frac{1}{t} \sum_{i=1}^{m} \frac{1 - e^{\frac{-it^2}{N}}}{\frac{it}{N}} \frac{t}{N} \\ &\simeq \frac{1}{t} \int_0^{\frac{mt}{N}} \frac{1 - e^{-tx}}{x} dx \\ &\simeq h(u) \frac{mt}{N} \end{aligned} \tag{21}$$

Where $u = \frac{mt^2}{N}$ and $h(u) = \frac{1}{u} \int_0^u \frac{1-e^{-x}}{x} dx$. The function $h(u)$ denotes a lower bound of coverage. The next table evaluates $h(u)$ for different values of u and illustrates that $h(u)$ rapidly decreases after u exceeds 1.

u	$h(u)$	u	$h(u)$	u	$h(u)$
2^{-4}	0.99	2^{-1}	0.89	2^{2}	0.49
2^{-3}	0.97	2^{0}	0.80	2^{3}	0.33
2^{-2}	0.94	2^{1}	0.66	2^{4}	0.21

However, in case of a tradeoff using distinguished points, equation 19 is not correct anymore when the merger problem appears. Indeed, the keys are stored in terms of a number of chains and the relevant probability is the probability to find a new chain, not a new key. The practical consequence is that the success rate (21) does not correctly take the mergers into account. This point was neglected in [5] and in the next section, we propose other approximations and show that the possibility to carry on computing chains after $mt^2 = N$ has to be considered if the objective is to get the fastest online attack.

7.4 A prediction of the mergers

Let $s(j)$ be a storage function denoting the number of keys stored after sort and rejection of mergers and $j = \gamma m$ be the number of chains computed in region $[t_{min}, t_{max}]$ (γ is the cover defined in section 7.2 and m is the number of start points considered). Let $p(j)$ be the probability that a new chain is found after a storage $s(j)$. We have:

$$s(j) = s(j-1) + \beta \times p(j-1) \tag{22}$$

$$p(j) = \prod_{l=0}^{\beta-1} \frac{2^k - s(j) - l}{2^k} \tag{23}$$

Because $l \ll 2^k$, e can reduce this system to a non-linear difference equation:

$$s(j+1) = s(j) + \beta \times \prod_{l=0}^{\beta-1} 1 - \frac{s(j)}{2^k} - \frac{l}{2^k}$$

$$s(j+1) \simeq s(j) + \beta \times (1 - \frac{s(j)}{2^k})^{\beta} \tag{24}$$

First proposal for $s(j)$: We computed a lower bound for $s(j)$ by using the linear approximation: $(1 - \frac{s(j)}{2^k})^{\beta} = (1 - \beta \times \frac{s(j)}{2^k} + O(s(j)^2))$ and solving the resulting linear recurrence:

$$s(j+1) \simeq (1 - \frac{\beta^2}{2^k}) \times s(j) + \beta \tag{25}$$

Which has solution:

$$s(j) \simeq (s(0) - \frac{2^k}{\beta}) \times (1 - \frac{\beta^2}{2^k})^j + \frac{2^k}{\beta} \tag{26}$$

The function $(1 - \frac{s(j)}{2^k})^{\beta}$ represents the probability that a new chain is found. If we define the saturation as the moment when we have $s(j+1) = s(j)$ corresponding to a probability zero that a new chain is found, Figure 4 illustrates

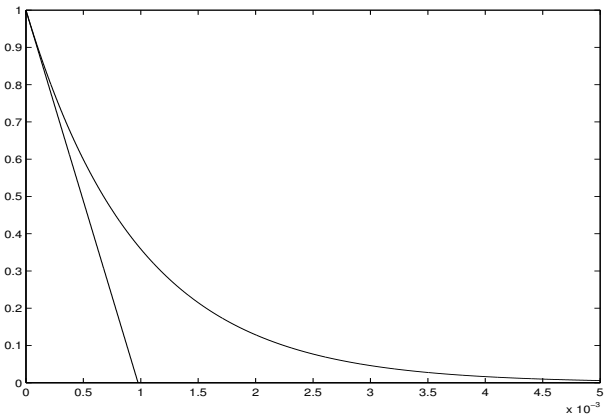

Fig. 4. Linear approximation of $(1-x)^k$ with $k = 2^{10}$

that our linear approximation involves a too fast saturation. As a consequence, we can derive a lower bound for the success rate: $SR \leq \frac{s(\gamma m)}{2^k}$.
Comparing this result with precedent conclusions about the success rate, we observe that the amount $s(\gamma m)$ is defined as a number of keys stored and β is the average chain length. This means that $\frac{s(\gamma m)}{\beta}$ is a number of chains like m. Therefore the condition suggested by Hellman: $mt^2 = N$ is similar to the condition of saturation when considering a linear approximation of the probability to find a new chain: $s(\gamma m) = \frac{2^k}{\beta}$ is equivalent to $\frac{s(\gamma m)}{\beta} = \frac{2^k}{\beta^2}$. This last point suggests that the decision to stop computations as soon as $mt^2 = N$ is not always optimal. For example if the objective is to get the fastest online attack, we will try to minimize the number of mask functions which involves the largest number of chains stored for every mask function.

Second proposal for $s(j)$: An improved approximation of the storage function is based on the convergence of Euler's methods described in [8]. The following equation:

$$\frac{s_{n+1} - s_n}{1} = \beta \times (1 - \frac{s_n}{2^k})^\beta \tag{27}$$

Can be approximated by

$$s'(j) = f(s, j) \tag{28}$$

Where $f(s,j) = \beta \times (1 - \frac{s(j)}{2^k})^\beta$ if $|\frac{df}{ds}| < L$ and $|s''(j)| < Y$ (L and Y are fixed constants). Because $s(j)$ is a storage function, it is obvious that $s''(j) < 0$ and $lim(s''(j)) = 0$. As $\frac{df}{ds} = \frac{-\beta^2}{2^k}(1 - \frac{s}{2^k})^{\beta-1}$ and $s \in [0, 2^k]$, the second condition is also fulfilled. Therefore, we solved the following differential equation:

$$\frac{s'(j)}{\beta} = (1 - \frac{s(j)}{2^k})^\beta \tag{29}$$

Using $t(j) := (1 - \frac{s(j)}{2^k})$ and $t'(j) := \frac{-s'(j)}{2^k}$, equation 29 is equivalent to:

$$\frac{-2^k}{\beta} \times t'(j) = t(j)^\beta \tag{30}$$

Which has an exact solution. Finally, we found

$$s(j) = 2^k \times \left(1 - \left(\frac{2^k}{-\beta j + \beta^2 j + K}\right)^{\frac{1}{\beta - 1}}\right) \tag{31}$$

With $K = 2^k \times (1 - \frac{s(0)}{2^k}) \times (\frac{1}{1 - \frac{s(0)}{2^k}})^\beta$ and the discretization error is estimated by:

$$e_{n+1} - e_n = \frac{df}{ds}(j, s) \times e_n + \frac{1}{2} s''(j) \tag{32}$$

These computations are in accordance with experimental results presented in section 8. We can conclude that:

1. The success rate evaluated in precedent papers is not directly applicable to tradeoffs using distinguished points. Mergers were not correctly taken into account.
2. The decision to stop precomputations at $mt^2 = N$ is not always optimal, depending on the objective: optimal precomputation time or optimal attack time.

7.5 A prediction of the average chain length after sort

An important consequence of the mergers is the possible modification of the average chain length after sort. Intuitively, this modification depends on the number of chains rejected and the choice of t_{min} and t_{max}. Looking at equation 23, we can observe that the probability that a chain merges is increasing with the length of this chain. Consequently, the average length of rejected chains is larger than the initial average chain length predicted by equation 17.
Practically, the rejection process is such that if two chains merge, only the longest one is stored. Therefore, a possible prediction of the modification of the average chain length could be achieved by computing the mergers prediction for every possible chain length, using initial conditions for the storage:

$$\beta_{mod} = \frac{\sum_{t_{min}}^{t_{max}} l.n_l}{\sum_{t_{min}}^{t_{max}} n_l} \tag{33}$$

Where n_l denotes the number of chains of length l after sort and is evaluated using the storage function of section 7.4. As the evaluation of $t_{max} - t_{min}$ difference equations is fastidious, we can divide the chain lengths in a number of sets. Let m be the number of chains computed, with average length β and cover γ. Let p be the number of sets used to evaluate β_{mod} (we chose $p = 4$). According to section 7.2, we evaluate β and γ for each set and denote them β_i and γ_i (good

approximations need $\gamma_i \simeq \gamma_j$ for every i, j). If m is the total number of chains computed, the number of chains computed in each region, say N_i is:

$$N_i = m \times \gamma_i \tag{34}$$

Then we solve for $i = p$ to 1:

$$s_i(j+1) \simeq s_i(j) + \beta_i \times (1 - \frac{s_i(j)}{2^k})^{\beta_i} \tag{35}$$

With the initial condition $s_p(0) = 0$ and $s_{i-1}(0) = s_i(N_i)$. From equation (26) or (31), we can derive the quantities $s_i(N_i)$ and $s_1(N_1)$ denotes the final number of keys stored. Finally, we compute the number of chains in each region: $c_i(N_i) = \frac{s_i(N_i) - s_i(0)}{\beta_i}$ and approximate the modified average length of chains as:

$$\beta_{mod} \simeq \frac{\sum_{i=1}^{p} c_i(N_i) \times \beta_i}{\sum_{i=1}^{p} c_i(N_i)} \tag{36}$$

7.6 Prediction of collisions and final probability of success

According to previous sections, the expected success rate using only one mask function is:

$$SR \simeq \frac{s(\gamma m)}{2^k} \tag{37}$$

If we use r different mask functions, the resulting probability of success is:

$$PS(r) = 1 - (1 - SR)^r \tag{38}$$

7.7 Memory complexity

Storage is needed for r tables. Each table contains about $\frac{s(\gamma m)}{\beta_{mod}}$ chains. These are in the form of triples and if we denote by e the actual size of an entry in the table, using the result of the previous section for r, the memory complexity C_{mem} can be expressed as:

$$C_{mem} \simeq e \times \frac{s(\gamma m)}{\beta_{mod}} \times r \tag{39}$$

7.8 Precomputation complexity

During the precomputation, we iterate for each point until either a distinguished point is reached or t_{max} iterations have been made. If a distinguished point is reached, then on average β iterations are computed. If a distinguished point is not reached, then t_{max} iterations are computed. We define the expected number of iterations for one chain as δ:

$$\delta = t_{max} \times (1 - P_1(t_{max})) + \beta \times P_1(t_{max}) \tag{40}$$

As iterations are made on $r \times m$ start points, the precomputation complexity C_{prec} can be estimated by:

$$C_{prec} \simeq r \times m \times \delta \tag{41}$$

7.9 Processing complexity

In a tradeoff using distinguished points, a table only has to be accessed when a DP is encountered during an iteration which means that for every mask function, only one chain has to be computed. Moreover, if the encountered DP is not in the table, then one will not find the target key by iterating further. Therefore, if r is the number of tables generated and β_{mod} is the average length of chains after sort, the processing complexity C_{proc} can be lower bounded by:

$$C_{proc} \leq r \times \beta_{mod} \tag{42}$$

Note that the possible reduction of the average chain length improves C_{proc} and the resulting online attack is faster. This last point strengthens the assumption that, for a fixed DP-property, computations beyond $mt^2 = N$ should be considered if the fastest online attack is to be reached.

8 Practical experiments

To confirm our analysis, we compare some theoretical predictions with experimental results. All our precomputations were carried out on a VIRTEX1000 FPGA board developed by the Microelectronics Laboratory at UCL. The board is composed of a control FPGA (FLEX10K) and a VIRTEX1000 FPGA associated with several processors (ARM, PIC) and fast access memories. The description of the hardware/software co-design used to perform the tradeoff and the complete experimental results can be found in [6].

Practically, we implemented two tradeoffs:

1. A first one against DES with a 40-bit key where we optimized the online attack. In this way, we illustrated a situation where condition $mt^2 = N$ is not optimal. The 40-bit DES is obtained from DES by fixing 16 key bits to arbitrary values.
2. A second one against DES with a 56-bit key where the precomputation task was critical. Therefore, we only implemented one mask function in order to evaluate the mergers and average chain length.

Both experiments confirmed our theoretical estimations. It is important to notice that experimental storage values were counted in terms of chains and therefore $\frac{s(\gamma m)}{\beta_{mod}}$ is the most significant data to compare. β and β_{mod} were evaluated on a sample of the results.

8.1 DES-40

Precomputation task: Table 1 summarizes our experimental results and theoretical predictions (in a log_2 scale) with a DP-property DP-11 and chain lengths $\in [2^9 - 2^{13}]$. It confirms our analysis to be a correct prediction of the tradeoff. Remark that $mt^2 = N$ would mean to limit the precomputations to $m = 2^{17.5738}$ which would not lead to an optimal online attack. As the precomputation complexity was easily reachable using FPGA's, we maximized the storage $s(\gamma m)$. Moreover the diminution of the average chain length also improves the online attack efficiency.

Table 1. Experimental results - Theoretical predictions

m	β	β_{mod}	$s(\gamma m)$	$\frac{s(\gamma m)}{\beta_{mod}}$
23.4219	11.2140	10.9721	30.4181	19.4460

m	β	β_{mod}	$s(\gamma m)$	$\frac{s(\gamma m)}{\beta_{mod}}$
21	11.2140	11.0400	29.7609	18.7209
22	11.2140	10.9653	30.0504	19.0851
23	11.2140	10.8847	30.2713	19.3866
24	11.2140	10.8040	30.4447	19.6407

Online attack: As the success rate of one single table is $\frac{2^{30.4181}}{2^{40}}$, we can derive the final probability of success in terms of a number of mask functions r. Experimentally, we used 2^{10} mask functions and observed a probability of success of 72% which is to compare with the 73.74% theoretically predicted. The online attack[2] was performed on a single PC[3]. Thanks to the optimized C_{proc}, we recovered a key in about 10 seconds. An exhaustive key search on the same PC would have taken about 50 days.

8.2 DES-56

Precomputation task: For this experiment, we experimentally observed the storage function $s(\gamma m)$ at different levels of precomputation. Table 2 summarizes our experimental results and theoretical experiments (in a log_2 scale) with a DP-property DP-18 and chain lengths $\in [2^0 - 2^{30}]$. Due to the critical pre-

Table 2. Experimental results - Theoretical predictions

m	β	β_{mod}	$s(\gamma m)$	$\frac{s(\gamma m)}{\beta_{mod}}$
20	18	17.8284	37.3781	19.5497
21	18	17.8261	38.1309	20.3048
22	18	17.6150	38.6140	20.9990
23	18	17.2983	38.9408	21.6425

m	β	β_{mod}	$s(m)$	$\frac{s(\gamma m)}{\beta_{mod}}$
20	18	17.8310	37.3022	19.4712
21	18	17.7242	37.8599	20.1357
22	18	17.5819	38.2685	20.6866
23	18	17.4104	38.5461	21.1357

computation task, this experiment was not likely to be optimized in terms of processing complexity. Anyway, it confirms our analysis to be a correct prediction of the tradeoff. Online attacks against DES-56 offer a variety of compromises between time and memory. The average chain lengths and number of mask functions needed to reach high success rates ($\beta = 2^{18}$ and $r \simeq 2^{18}$ in our example) will make FPGA's the relevant tools to mount efficient online attacks. They offer the high encryption rates and reconfigurability needed for cryptanalytic applications.

[2] First presented at the rump session of CRYPTO2001
[3] 18Gbytes ROM/256Mbytes RAM/350MHz

9 Conclusion

Confirmed by experimental results, we propose a new theoretical analysis of cryptanalytic time-memory tradeoffs using distinguished points and underline particularities of this variant of Hellman's proposal. The model allows the prediction of both precomputation and online attack parameters: the complexity of the tradeoff is evaluated as well as its resulting probability of success. Predictions of earlier papers are modified in order to correctly take the mergers into account and we suggest situations where our theoretical predictions induce practical improvements.

We implemented the tradeoff against DES with a 40-bit key and recovered a key in 10 seconds, with a success rate of 72%, using one PC. The exhaustive search of the key on the same PC would have taken about 50 days. In parallel, practical experiments against DES with a 56-bit key confirmed the effectiveness of FPGA's in cryptanalytic applications. We used FPGA's to perform precomputation tasks but in case of tradeoffs implemented against ciphers with large keys, an FPGA-based implementation of the online attack would be very efficient and reasonably expensive compared with software ones.

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. 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, pp.111-118.
4. J.Borst, B.Preneel and J.Vandewalle, *A Time-Memory Tradeoff using Distinguished Points*, Technical report ESAT-COSIC Report 98-1, Departement of Electrical Engineering, Katholieke Universiteit Leuven, 1998.
5. J.Borst, *Block Ciphers: Design, Analysis and Side-Channel Analysis*, Chapter 3: A Time-Memory Tradeoff attack, Phd Thesis, Departement of Electrical Engineering, Katholieke Universiteit Leuven, 2001.
6. J.J.Quisquater, F.X.Standaert, G.Rouvroy, J.P.David, J.D.Legat, *A Cryptanalytic Time-Memory Tradeoff: First FPGA Implementation*, To appear in the Proceedings of FPL 2002 (The Field Programmable Logic Conference) .
7. 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.
8. S.D.Conte, C.de Boor, *Elementary of Numerical Analysis*, Chapter 8.4, Mc Graw-Hill, 1981.
9. Amos Fiat and Moni Naor, *Rigorous Time/Space Tradeoffs for Inverting Functions*, SIAM J. Computing 29(3), 1999 pp. 790-803.