2009 International Conference on Availability, Reliability and Security

Choosing Parameters to Achieve A Higher Success Rate for Hellman Time Memory Trade Off Attack

Nurdan Saran
Department of Computer Engineering
Cankaya University, Ankara, TURKEY
Email: buz@cankaya.edu.tr

Ali Doğanaksoy
Department of Mathematics
METU, Ankara, TURKEY
Email: aldoks@metu.edu.tr

Abstract

In 1980, Hellman proposed the Time Memory Trade Off (TMTO) attack and applied it on block cipher DES (Data Encryption Standard). Time Memory Trade Off attack is one of the methods that inverts a one way function. The resistance to TMTO attacks is an important criterion in the design of a modern cipher. Unlike the exhaustive search and table lookup methods, TMTO is a probabilistic method, that is, the search operation may not find a preimage even if there exists one. Up to now, there are some approximate bounds for success rates of Hellman table by Hellman and Kusuda et al. In this study, we give a more precise approximation for the coverage of a single Hellman table. There is no precise guideline in the literature that points out how to choose parameters for Hellman TMTO. We present a detailed analysis of the success rate of Hellman table via new parameters and also show how to choose parameters to achieve a higher success rate. The results are experimentally confirmed. We also discuss the Hellman's TMTO Curve.

Index Terms

Symmetric Key Cryptography, Cryptanalysis, Time Memory Trade Off Attack, Success Probability

1. Introduction

Inverting a one-way function is a basic problem in cryptanalysis. In most situations, the security of a cryptosystem depends on the invertibility of a one way function. Block ciphers and stream ciphers are some of the examples in which inverting the underlying function means breaking the system. In 1980, Hellman[12] proposed the Time Memory Trade Off (TMTO) method and applied it on block cipher DES (Data Encryption Standard). He mentioned that the method can be applied on any one-way function. This method suggests a trade off between time and memory by storing an amount of pre-computed data in the memory. It has a lower time complexity (in online phase) than exhaustive search and lower memory complexity than lookup table. If the attacker has access to a computer with a large amount of memory, the computation time will be less. Since it is a probabilistic method; the success rate depends on the time and memory allocated for cryptanalysis. This attack can be used as a known plaintext attack for block ciphers. Exhaustive searching consumes a lot of computing power when the same attack has to be carried out multiple times. It can be carried out when the attacker can guess some bytes of data such as password hashes. Moreover, it can also be used as a ciphertext-only attack by waiting for repeated ciphertext blocks and assuming the corresponding chosen plaintext.

There are two main improvements on TMTOs; distinguished points [8], [9], [20], [13] by Rivest and Rainbow tables [18], [1], [2] by Oechslin. Rivest noticed that the number of disk access operations is slower than than the evaluation of encryption function. He suggested to the use distinguished points(DP) to reduce the memory access in the online phase. In this method, rather than a fixed number of iterations, the encryption function is iterated until a point that satisfies a certain property(DP) is found. Oechslin suggested to use new tables (called Rainbow Tables) in which the online time complexity reduces by a factor of 2.

TMTO is also applied on stream ciphers in several papers, firstly [3], [11], [6], then A5 [7], Lili-128 [19]. Some Ecrypt stream ciphers such as Mickey, Grain are also analyzed from the view point of TMTO attacks.

In [15] TMTO on various block cipher modes of operations are given, in [14]-[5], it is shown how to

978-0-7695-3564-7/09 $25.00 © 2009 IEEE
DOI 10.1109/ARES.2009.140

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.

apply multiple data TMTO on block ciphers. The usual TMDTO (time memory data trade-off) attack on stream ciphers can be considered to be a TMK (time memory key trade-off) attack on block ciphers when a key is used to encrypt several plaintexts.

There are some approximate bounds for success rates of Hellman table in the literature. Up to now although very close bounds are known, there is no explicit expression for the coverage of a single Hellman table. We give an explicit formula for the number of distinct points covered by a single Hellman table, which in turn allows us to derive a formula regarding the overall success probability of the method. In this paper, we first describe the Hellman's construction. In his work, Hellman gives a lower bound for a coverage rate and states that it numerically evaluates to 0.80 for $mt^2 = N$. In [17] Kusuda and Matsumoto obtains an upper bound for the success rate in terms of m and t. They study to optimize a relation among the breaking cost and time. Kim and Matsumoto show that the table parameters can be adjusted to achieve higher success probability[16]. Firstly, we give a more precise approximation for success rate of a single Hellman Table. In the literature, there is no precise guideline about how to choose parameters. When applying the method in the sense of Hellman, the parameters in question are the length of each chain (t), the number of chains (m) and the number of tables (r). In general usage, success rate of a single table is defined to be $S = \frac{\#\text{number of distinct points}}{N}$. In [4], Barkan et al stated that the best collection of start points will even have the same coverage as a random Hellman table. Our paper examines how the parameters affect the coverage(#number of distinct points in a single table) which determines the success rate. Our main subject is to examine whether it is possible to achieve higher success rates by choosing different parameters or not. We present a detailed analysis via new parameters for a whole success rate of the attack. Finally, we give our experimental results on SHA-1 hash algorithm with a key space of size 2^{40} to compare with the theoretical results. And also we discuss Hellman's curve.

2. Hellman's construction

Let $E : K \times P \rightarrow C$ be an encryption function 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, respectively.

The encryption of one block is written as:

$$c = E(x,p) \text{ or } c = E_x(p).$$

Given $c_0 = E(x, p_0)$ where p_0 is a fixed plaintext and c_0 is the corresponding ciphertext, the aim of the attacker is to recover any of the key $x \in K$, more precisely some key x' such that $c_0 = E(x', p_0)$. We may assume that $k = n$. Otherwise employing a reduction or an expansion function R, it is possible to define $E_{p_0} \circ R : K \rightarrow C$ such that $(E_{p_0} \circ R) = R(E(x_0, p))$. Thus to generate a key from a ciphertext, we use $x_{i+1} = R(c_i)$ where $c_i = E(x_i, p_0)$.

A TMTO application is composed of two phases; an offline (precomputation) phase and an online phase. In the offline phase, the attacker constructs tables that contain possible keys; this process is approximately equivalent to the exhaustive search however only a part of the tables are stored. During online phase, the attacker expects to recover the particular key when the plaintext and ciphertext is known; his aim is to reduce the time of search. This leads to a trade-off between the memory required in the offline stage and time required in the online phase.

Offline Phase(Precomputation):

In the Hellman table, starting from a random x_0, iteratively evaluating $E \circ R = f$, a chain of length t is generated as follows:

$$x_0 \xrightarrow{f} x_1 \xrightarrow{f} x_2 \xrightarrow{f} \ldots \xrightarrow{f} x_t.$$

To construct a Hellman table of size $m \times t$, choosing m random start points, chains of length t are generated. To save memory, only the first and last terms of each chain are stored. Also the table is sorted with respect to end points for speeding up the search in online phase. Marginal profit of each new row is quite small when the number of rows exceeds some bound (given in the next section). Therefore rather than increasing the number of rows, it is much more reasonable to define new tables each of which depends on new function formed by combining the original function with permutation. By this way, r-Hellman tables are formed. It should be noted that chains computed using different functions can intersect, but they will not merge.

Online Phase:

For a preimage of a given ciphertext c_0, the aim of the attacker is trying to find out if the key is used to generate c_0 is among the one used in any

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.

of the generated tables. Since only start/end points are stored, a chain for c_0 is generated and after each encryption, the obtained value is compared to the endpoints of all tables. If a match is found, then from the corresponding start point, the whole chain is regenerated and the found key is expected to be the searched key. It should be noted that sometimes desired key is a part of a chain that is merged with another chain of the table, resulting false alarms. Therefore, the success rate of the attack is closely related to number of distinct keys that are covered in the offline phase.
When applying the method by means of a single Hellman table, the parameters in question are the length of each chain(t), the number of chains(m), number of tables (r).

Memory complexity is the amount of memory that is necessary to store pre-computed data. $M = 2 * m * r * m_0$ where M is the amount of memory; m_0 is the amount of memory that is necessary to store a start and end point(in general, it is omitted).

Time complexity is the amount of time required to perform the attack successfully. In time-memory trade offs time complexity is divided into two parts; precomputation and online time complexities.

$T = t * r * t_0$ where T is the worst case time required in the online phase; t_0 is the amount of time that is necessary to evaluate the function (in general, it is omitted). False alarms are neglected.

$P = m * t * r$ where P is the time complexity required in the precomputation phase. It is often compared to exhaustive key search. Generally P is not included in the attack complexity.

Data complexity(D) is the amount of data that is required to mount an attack (like ciphertexts, known plaintexts, chosen plaintexts). For block ciphers, it is accepted to be 1. But in multiple data [5], if some plaintexts are encrypted with the same key, it is in question to use data which is called TMDT (Time Memory Data Trade Off).

Effects of parameters

- t: If chosen to be too large chains will probably enter in loops which will undesirably increase both precomputation time and online time. If it is too small it will increase the memory requirement.
- m: If chosen to be too large the number of identical points will increase. If it is too small it is necessary to construct more tables.
- r: If it is too large, it will increase both memory and online time complexity.

2.1. Single Table

S is the success rate of Hellman tables, in other words, given $c_0 \in Im(E)$, the probability to find some x such that $E(x) = c_0$.

In general usage, success rate of a single table is defined to be $S = \frac{\#\text{number of distinct points}}{N}$. However, for a random mapping $F : X \to X$ with $|X| = N$, expectedly Ne^{-1} points will have no inverse images. Consequently, the success rate will be bounded by $(1 - e^{-1}) \approx 63\%$ if it is related with the probability of finding inverse image of an arbitrary $x \in X$. If we are focused on points $x \in F(x)$, the success rate can take any value in $(0, 1)$. Since it is a chosen plaintext attack for block ciphers we know that the given point has at least one inverse so the expected success rate of the table will be $S = \frac{\#\text{number of distinct points}}{N(1 - e^{-1})}$.

For a single table, it is possible to find x such that $E(x) = y$ if and only if x appears somewhere (starting from the second column since first column does not give a clue for x) in the table. Thus, the number of preimages than can be found in a table is equal to the number of distinct entries in the table which will be called $Y(m, t)$, the coverage of the table. Thus the success rate of a table is defined as

$$S(m, t) = \frac{Y(m,t)}{|Im(E)|}.$$

An important question is how to choose m and t to have a high table coverage, hence a high success rate.

Hellman suggested that m and t should be chosen to satisfy $mt^2 = N$. We will settle the mathematical background to explain the relation between mt^2 and N(however [6] explains with matrix stopping rule). For $m < \sqrt{N}$

$$Y(m, t) = \sqrt{2Nm}.tanh\left(\sqrt{\frac{mt^2}{2N}}\right). \qquad (1)$$

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.

Proof is in the Appendix. Refer to [10] for details.

From Eq. 1 we obtain the marginal change in the value of $Y(m,t)$ is

$$\Delta Y \approx \frac{\partial y}{\partial m}\Delta m + \frac{\partial y}{\partial t}\Delta t$$

that is

$$\Delta Y \approx \left(\frac{\sqrt{N}}{\sqrt{2m}}\tanh\left(\sqrt{\frac{mt^2}{2N}}\right) + \frac{t}{2}\sec h^2\left(\sqrt{\frac{mt^2}{2N}}\right)\right) + \left(m\sec h^2\left(\sqrt{\frac{mt^2}{2N}}\right)\right)\Delta t$$

Then for a fixed value of m ($\Delta m = 0$) and for $\Delta t = 1$, in other words adding one column to a table will effect the coverage by

$$\Delta Y = m\sec h^2\left(\sqrt{\frac{mt^2}{2N}}\right).$$

For example, let $N = 2^{48}$, $m = 2^{16}$, $t = 2^{19}$ then $\Delta Y = 2^{1,67} < 4$. This means that by a work of $m = 2^{16}$ encryptions, we obtain less than 4 'distinct' points in a table. So the value of m and t should be chosen properly. Now we will try to investigate the 'rules' how to choose the feasible values for m and t.

A natural way of measuring the feasibility of a single table is the ratio of the number of distinct points (outcomes) to the number of work done (encryptions) so we define

$$R(m,t) = \frac{Y(m,t)}{mt}$$

as the *feasibility* of the table. Then

$$R(m,t) = \sqrt{\frac{2N}{mt^2}}tanh\left(\sqrt{\frac{mt^2}{2N}}\right) \qquad (2)$$

It should be noted that R is indeed a function of 'mt^2'. This means that for any fixed $C \in R$ any choice of m and t with $mt^2 = C$ will yield the same feasibility. Then the success of a single table is

$$S(m,t) = \frac{Y(m,t)}{N'} \qquad (3)$$

where $N' = |Im(f)| = N(1-e^{-1})$.

Define $\alpha = \dfrac{mtr}{N}$ and $\lambda = \dfrac{mt^2}{N}$. In general, $\alpha = 1$ since precomputation time(P) is equal to mtr and precomputation time is expected to be not exceeding N. Then we have

$$R(m,t) = \sqrt{\frac{2}{\lambda}}tanh\left(\sqrt{\frac{\lambda}{2}}\right)$$

.

Since each table has different reduction functions, the coincidence between rows of a table does not occur among different searched tables. Expected number of distinct elements in the entire set of r tables is

$$S(m,t,r) = 1 - \left(1 - \frac{Y(m,t)}{N'}\right)^r \approx 1 - e^{-\frac{Y(m,t)r}{N'}}$$

Since $r = \frac{\alpha N}{mt}$
Δm

$$S(m,t,r) = 1 - e^{\frac{-Y(m,t)}{mt}\frac{N}{N'}} = 1 - e^{\frac{-R(m,t)\alpha}{\varphi}}$$

where $\varphi = 1 - e^{-1}$.

Figure 1 gives the $R(m,t)$ for different λ. For $\lambda = 1$, $R(m,t) = 0,86$.

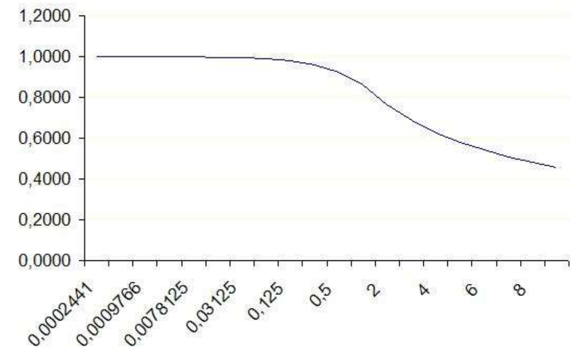

Figure 1. Feasibility of a table

2.2. Experimental Results

We give our experimental results on Secure Hash Algorithm (SHA-1). SHA is published by NIST (National Institute of Standards and Technology) as a U.S. government standard. SHA is a hash function family consisting of five algorithms; SHA- 1, SHA-224, SHA-256, SHA-384, and SHA-512. SHA-1 is widely used in various applications such as TLS and SSL, PGP, SSH, S/MIME, and IPsec.

1) To compare our analysis with empirical results, we give empirical table coverage, H(m,t), overall coverage H(m,t,r) for reduced Sha-1 with $N = 2^{21}$ in Table 1. $\alpha = 1$ in this table, so precomputation time(P) is not exceeding N.
2) Hellman suggested that m and t should be chosen to satisfy $mt^2 = N$ to obtain optimal coverage rate (the row colored in gray in Table 2). Now we demonstrate that with different M/T (memory/ time complexities), it is still possible to have the same success rate. The table is

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.

λ	4	2	1	1/2	1/4
m	128	64	128	64	128
t	256	256	128	128	64
r	64	128	128	256	256
R(m,t)	0,63	0,76	0,86	0,92	0,96
S(m,t)	0,02	0,01	0,01	0,01	0,01
S(m,t,r)	0,63	0,70	0,75	0,77	0,78
Y(m,t)	20459	12364	13979	7494	7733
H(m,t)	20584	12478	14108	7572	7867
Y(m,t,r)	782679	864433	918141	947460	960488
H(m,t,r)	838736	930556	988617	1019724	1036794

Table 1. Comparison for reduced Sha-1 with $N = 2^{21}$

m	t	r	S(m,t)	S(m,t,r)	M	Time
2^{20}	2^{10}	2^{10}	0,0013	0,744128319	30	20
2^{18}	2^{11}	2^{11}	0,0007	0,744012227	29	22
2^{16}	2^{12}	2^{12}	0,0003	0,743954199	28	24
2^{14}	2^{13}	2^{13}	0,0002	0,743925191	27	26
$2^{13,4}$	$2^{13,3}$	$2^{13,3}$	0,0001	0,743919745	26,7	26,6
2^{12}	2^{14}	2^{14}	8,31E-05	0,743910687	26	28
2^{10}	2^{15}	2^{15}	4,15E-05	0,743903436	25	30
2^{8}	2^{16}	2^{16}	2,07E-05	0,74389981	24	32
2^{6}	2^{17}	2^{17}	1,03E-05	0,743897998	23	34
2^{4}	2^{18}	2^{18}	5,19E-06	0,743897091	22	36

Table 2. For $N = 2^{40}$ Hellman Construction

constructed for $N = 2^{40}$, $\lambda = 1$, $\alpha = 1$,resulting in $R(m,t) = 0,861057172$.
For an instance, if an attacker has less memory and more computational power, he may try the last row of the above table. Hellman's choice is the case where the memory and time complexities are closest.

3) Since $R(m,t)$ is the single table coverage if $R(m,t) = 1$ then $\lambda \cong 2^{-12}$, this means $mt^2 = 2^{-12}N$ (See Table 3). Then, $r = \dfrac{\alpha t}{\lambda}$.

α	r	$S(m,t,r)$
1	$2^{12}t$	0.7955
2	$2^{13}t$	0.9582
3	$3 * 2^{12}t$	0.9915

It is clear that choosing m and t small, it will increase the single table coverage. However, it will increase the number of tables, r, which will cause excessive M and T.

3. Hellman's Curve

The number of tables constructed in Hellman attack is $r = t$, then $M = m * t$, $T = t * t$. The time required in offline phase is $P = mt^2 = N$. Then trade off curve is obtained as:

$$TM^2 = N^2 \text{ where } 1 \leq T \leq N$$

Hellman suggests to construct $N^{1/3}$ tables, each with $m = N^{1/3}$ words, and the number of operations per table is also as $t = N^{1/3}$ [12].

By choosing $\lambda \leq 1$, it is possible to achieve a higher success without increasing memory but additional cost of time complexity (See in Table 3). In the above table, $t = 2^{13,3}$,$\alpha = 1$.

m	r	λ	R(m,t)	S(m,t)	S(m,t,r)	M	T
11	10327588	2^{-10}	0,999	1,53E-07	0,794	26,7	36,6
169	645474	2^{-6}	0,997	2,44E-06	0,793	26,7	32,6
676	161369	2^{-4}	0,989	9,70E-06	0,791	26,7	30,6
2702	40342	2^{-2}	0,960	3,76E-05	0,781	26,7	28,6
5405	20171	2^{-1}	0,924	7,24E-05	0,768	26,7	27,6
10809	10086	2^{0}	0,861	0,0001	0,743	26,7	26,6
216189	5043	2^{1}	0,762	0,0002	0,700	26,7	25,6
43238	2521	2^{2}	0,628	0,0003	0,630	26,7	24,6

Table 3. $N = 2^{40}$ Hellman Construction for different λ s.

Some Remarks

Remark 1 : There are two ways to apply TMTO method on stream ciphers according to the one way function to be inverted [15]. First one is to invert the function from state to output prefix(which has equal size with the state). Second one is to invert the function from secret key (key+iv) to output prefix (which has equal size with secret key). In the literature TMTO Curve for stream ciphers are given as TM=N since a stream cipher is expected to be a permutation. Hong et al [15] stated that for a good stream cipher the second construction is very close to a permutation. But it is not a permutation, in most cases it is a random mapping. Moreover the first construction is not necessarily permutation.

Remark 2 : Generating a table without merges is another trade because it will increase the precomputation time. Perfect tables are tables in which there are no merges. Firstly more chains are generated since merging chains will have the identical endpoints, they can be easily discarded (in both rainbow and DP tables). In[1], Avoine et al stated that to construct a perfect Hellman table is not efficient since all chains have to be looked up. We give a more efficient method of generating perfect Hellman tables. Assuming encryption function is modeled as a random mapping the set into itself, if we consider the graph representation of a random mapping, we choose our start points as terminal nodes(we mean the points which have no preimage). It is clear that

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.

to find terminal nodes, it is necessary to increase precomputation from N to $\leq 2N$. Since there are $N * e^{-1}$ terminal nodes, the memory requirement will increase. But both online and offline time complexities will be smaller than other perfect tables.

4. Conclusion

In this paper we give an explicit formula for choosing parameters in Hellman's TMTO. We obtain a success formula regarding to $\alpha = \frac{mtr}{N}$ and $\lambda = \frac{mt^2}{N}$. For a fixed total complexity, we can easily adjust time and memory complexities to have a better coverage rate. For $\alpha = 1$ (fixed precompuation), it is possible to have equal success rates with different time and memory complexities, resulting different curves ($\lambda \neq 1$). Explicit formula for choosing parameters for other improvements such as DP and rainbow tables is a further research topic.

References

[1] G. Avoine, P. Junod, and P. Oechslin. Time-memory trade-offs: False alarm detection using checkpoints. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 183–196. Springer, 2005.

[2] G. Avoine, P. Junod, and P. Oechslin. Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Trans. Inf. Syst. Secur.*, 11(4):1–22, 2008.

[3] S. Babbage. Improved "‘exhaustive search"’ attacks on stream ciphers. In *ECOS 95 (European Convention on Security and Detection)*, number 408 in IEE Conference Publication, May 1995.

[4] E. Barkan, E. Biham, and A. Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 1–21. Springer, 2006.

[5] A. Biryukov, S. Mukhopadhyay, and P. Sarkar. Improved time-memory trade-offs with multiple data. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer, 2005.

[6] A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.

[7] A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of a5/1 on a pc. In B. Schneier, editor, *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2000.

[8] J. Borst. *Block Ciphers: Design, Analysis and Side-Channel Analysis*. PhD thesis, Katholieke Universiteit Leuven, 2001.

[9] J. Borst, B. Preneel, and J. Vandewalle. On time-memory tradeoff between exhaustive key search and table precomputation. In P. H. N. de With and M. van der Schaar-Mitrea, editors, *Proc. of 19th Symp. on Information Theory in the Benelux*, pages 111–118, Veldhoven (NL), 28-29 1998. Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL).

[10] C. Çalık. How to invert one-way functions: Time-memory trade-off method. Master's thesis, METU, 2007.

[11] J. D. Golic. Cryptanalysis of alleged a5 stream cipher. In *EUROCRYPT*, pages 239–255, 1997.

[12] M. E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[13] J. Hong, K. C. Jeong, E. Y. Kwon, I.-S. Lee, and D. Ma. Variants of the distinguished point method for cryptanalytic time memory trade-offs. In L. Chen, Y. Mu, and W. Susilo, editors, *ISPEC*, volume 4991 of *Lecture Notes in Computer Science*, pages 131–145. Springer, 2008.

[14] J. Hong and P. Sarkar. New applications of time memory data tradeoffs. In B. K. Roy, editor, *ASIACRYPT*, volume 3788 of *Lecture Notes in Computer Science*, pages 353–372. Springer, 2005.

[15] J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs, 2005.

[16] I.-J. Kim and T. Matsumoto. Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Transactions on Fundamentals*, E82-A(1):123–129, 1999.

[17] K. Kusuda and T. Matsumoto. Optimization of time-memory trade-off cryptanalysis and its application to des, feal-32 and skipjack. *IEICE Transactions on Fundamentals*, E79-A(1):35–48, 1996.

[18] P. Oechslin. Making a faster cryptanalytic time-memory trade-off. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer, 2003.

[19] M.-J. O. Saarinen. A time-memory tradeoff attack against lili-128. In J. Daemen and V. Rijmen, editors, *FSE*, volume 2365 of *Lecture Notes in Computer Science*, pages 231–236. Springer, 2002.

[20] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat. A time-memory tradeoff using distinguished points: New analysis & fpga results. In B. S. K. Jr., Çetin Kaya Koç, and C. Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 593–609. Springer, 2002.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 14,2023 at 20:37:00 UTC from IEEE Xplore. Restrictions apply.