

Improved Time-Memory Trade-offs with Multiple Data

Alex Biryukov¹, Sourav Mukhopadhyay² and Palash Sarkar²

¹ Katholieke Universiteit Leuven,
Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10,
B-3001 Heverlee, Belgium
<http://www.esat.kuleuven.ac.be/~abiryuko/>

² Cryptology Research Group
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108

Abstract. In this paper we study time/memory/data trade-off attacks from two points of view. We show that Time-Memory trade-off (TMTO) by Hellman may be extended to Time/Memory/Key trade-off. For example, AES with 128-bit key has only 85-bit security if 2^{43} encryptions of an arbitrary fixed text under different keys are available to the attacker. Such attacks are generic and are more practical than some recent high complexity chosen related-key attacks on round-reduced versions of AES. They constitute a practical threat for any cipher with 80-bit or shorter keys and are marginally practical for 128-bit key ciphers. We show that UNIX password scheme even with carefully generated passwords is vulnerable to practical trade-off attacks. Our second contribution is to present a unifying framework for the analysis of multiple data trade-offs. Both Babbage-Golic (BG) and Biryukov-Shamir (BS) formulas can be obtained as special cases of this framework. Moreover we identify a new class of *single table* multiple data trade-offs which cannot be obtained either as BG or BS trade-off. Finally we consider the analysis of the rainbow method of Oechslin and show that for multiple data, the TMTO curve of the rainbow method is inferior to the TMTO curve of the Hellman method.

Keywords: time/memory/data trade-off, block-cipher, key sizes.

1 Introduction

In 1980, Hellman [8] introduced the technique of time-memory trade-off (TMTO) attack on block ciphers. In its more general form, it can be viewed as a general one-way function inverter. The original work by Hellman considered inverting a one-way function f at a single data point. Babbage [1] and Golic [7] (BG) have shown that in the context of stream ciphers multiple data points can be used by another trade-off attack relying on birthday paradox. Independently,

Biham [3, 2] has shown that birthday-paradox trade-off applies to block-ciphers in a frequently changing key scenario. Both BG and Biham's results show that theoretical strength of a block or stream cipher without an IV (nonce) can not exceed the square root of the size of the key space. However birthday trade-offs suffer from a weakness: they lack flexibility due to strong binding of memory complexity with the data complexity which typically results in unrealistic data or memory requirements. Later Biryukov and Shamir [5] (BS) have shown that multiple data can be combined with Hellman's tradeoff, resulting in a flexible time/memory/data tradeoff formula.

In the context of block ciphers with reasonably long keys, the original Hellman attack is typically not considered to be of a threat since its precomputation time is the same as the exhaustive search of the key. Moreover, the attack works for a single chosen plaintext encryption and cannot benefit if more plaintext-ciphertext pairs are available to the attacker since the precomputed tables are "wired" to a fixed plaintext. This is contrary to what happens in Babbage-Golic-Biham or Biryukov-Shamir's trade-off where precomputation time is way below the exhaustive key search complexity.

At the rump session of ASIACRYPT 2004, Hong and Sarkar [9] have demonstrated that stream ciphers with short IVs can be attacked via the Biryukov-Shamir time/memory/data trade-off [5] in a frequent re-synchronization scenario. More recently in [10] they also provide a careful study of time/memory/data trade-off attack in the context of various modes of operation of block-ciphers noticing that these essentially constitute a stream cipher. However, we believe that [10] does not consider in sufficient details the important cases of ECB (a mode typically assumed in theoretical cryptanalysis) or CBC with known IV's or counter and OFB modes in chosen IV scenarios, which directly lead to very powerful attacks. They also describe attacks which have preprocessing times higher than the complexity of exhaustive search and thus seem to be less relevant.

Our Contributions: Our contribution is two-fold. We present new applications as well as a new analysis of time/memory/data trade-off attacks.

We describe three applications.

- The usual time/memory/data trade-off attack on stream ciphers can be considered to be a time/memory/key trade-off attack on block ciphers. This attack applies to situations where the goal of the attacker is to obtain one out of many possible keys. See Table 1 for a picture how various block and stream cipher tradeoffs are related.
- We carefully consider the key size of block ciphers and conclude that in the view of trade-off attacks one may no longer assume k -bit security even for a good k -bit cipher. This is true for ECB mode but unfortunately also true for CBC mode even with proper IV choice. Counter and OFB modes are vulnerable to this attack in chosen IV scenarios.
- The last application is to Unix password scheme, where the short size of the "salt" is insufficient to stop trade-off attacks. We describe practical attacks on this scheme when a password file with reasonably many password hashes is available to the attacker.

Table 1. Relation between block and stream cipher trade-off attacks.

	Block ciphers (varying keys or IV's)	Stream ciphers (varying keys or IV's)
Type of	Biham's collision [2]	Babbage-Golic birthday [1, 7]
trade-off	this paper and [10]	Biryukov-Shamir TMD [5, 10]

In the second part of the paper, we perform a new unified analysis of Hellman's attack in the presence of multiple data. The BG and the BS attacks are obtained as special cases of the general attack. So far, it has been believed that any multiple data TMTO is either BG or BS. Our work reveals that there are other possible desirable trade-offs for *single table*, *multiple column* attacks which are not obtainable from either the BG or the BS attack. The time required for table look-ups can be reduced by using Rivest's idea of distinguished point (DP) method. We analyse this method in a general setting. Finally, we consider the rainbow method and show that in the presence of multiple data, the TMTO curve of the rainbow method is inferior to the TMTO curve of the Hellman+DP method.

2 Time/Memory/Data Trade-Off Methodology

In this section we give a quick introduction for the methodology behind Hellman's [8] time/memory trade-off as well as for the closely related time/memory/data trade-off attacks by Biryukov-Shamir [5].

Suppose that $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a one-way function, i.e. a function which can be efficiently evaluated in forward direction, but which is hard to invert. The goal of the attacker is to invert this function, i.e. given $f(x)$ to find x , while keeping complexity of the inversion algorithm (time T , data D , memory M and preprocessing time P) as low as possible. Throughout this paper we will denote by $N = 2^n$ the size of the domain of the function.

In [8] Hellman has proposed a general algorithm for inversion of an arbitrary one-way function which obeyed the formula: $N^2 = TM^2, D = 1, P = N$. In [5] Hellman's algorithm has been generalized for the case of multiple data, which resulted in a formula $N^2 = TM^2D^2, 1 \leq D^2 < T, P = N/D$. Here we will directly describe this generalized trade-off.

The algorithm consists of two stages: a one-time offline stage followed by an online stage. In the online stage, we will be given D points y_1, \dots, y_D in the range of f and our goal is to find the pre-image of any one of these points. In the offline stage, a set of tables are prepared covering N/D of the domain points. Since N/D domain points are covered and the online stage involves D domain points, by the birthday bound, we are assured of constant probability of success in finding the pre-image of one of the y 's.

Let f_1, \dots, f_r be simple output modifications of f ($f_i = g_i \circ f$, where g_i can be a simple bit permutation of the output bits), such that the f_i 's can be assumed to be pairwise independent. In the offline stage r tables (one per f_i) are prepared.

The i th table is prepared in the following manner. A total of m random domain points are chosen. For each domain point, the function f_i is iteratively applied t times to reach an end point. The pairs (start-point, end-point) are stored as part of the i th table sorted by the end points. The total storage requirement is rm pairs of points, while the total coverage is rm points.

In the online stage, for each data point y_j we look for a pre-image in the set of points covered by the tables. For searching in the i th table, we first apply g_i to y_j to obtain y'_j , then we iteratively apply f_i a total of t times to y'_j . After each application of f_i , we look in the end points of the i th table for a match. If a match is found, we go to the corresponding start point and iterate f_i until we reach y'_j . The preceding point is a possible pre-image of y , which is verified by applying f to it and checking whether we obtain y . This requires a total of t applications of f and t table look-ups per table per data item. In order to minimize the waste of table coverage due to birthday collisions the proper choice of parameters m and t would typically satisfy $N = mt^2$ (in Sec. 8 it will be shown how to obtain a new trade-off formula by using sub-optimal choices of m and t). Since a single matrix covers only mt points in order to cover the full space one will need $r = N/mt = t$ tables corresponding to different functions $f_i, i = 1, \dots, r$. However in generalized case we need to cover only a fraction N/D of space, and thus $r = t/D$ tables would suffice. By eliminating parameters r, m, t one gets a tradeoff formula $N^2 = TM^2D^2, 1 \leq D^2 < T, P = N/D$. For more details regarding the method see [8, 5].

We discuss some general issues about TMTO. In Hellman's original scheme, $D = 1$; the table preparation time is disregarded and only the online time and memory requirements are considered. The assumption is that the tables would be prepared once for all in an offline phase. Once the tables are prepared, they will not change and can be used to find different pre-images. In this scenario, the table preparation time can be huge and even larger than exhaustive search. Thus, the security of a cryptographic algorithm with respect to this kind of TMTO has a hidden cost of offline (and one time) exhaustive search.

If multiple data is available, the actual table preparation time will be less than exhaustive search. Since this is an offline activity, it might be reasonable to expect the table preparation time to be more than the online time but less than exhaustive search time.

The precomputation time will be in general more than the memory requirement. In the table preparation stage, the entire table will have to be computed and only a fraction of it stored. This shows that the offline time will be at least as large as the memory requirement. Hellman in his original paper [8], considered the condition where the online time is equal to the memory requirement. In the presence of multiple data, it is perhaps more practical to require the data and memory requirement to be less than the online and offline time requirements. This has been considered in [5].

3 Time/Memory/Key Trade-offs

It is easy to see that all the reasoning from the Time/Memory/Data trade-off in the case of stream ciphers [5] can be applied to the block-cipher “Time-Memory-Key” case. Namely we no longer need a full coverage of the space N , but rather can cover a fraction N/D_k , where we denote by D_k the number of possible keys at the online stage. Thus, we will use t/D_k tables instead of t , which means that memory requirements go down to $M = mt/D_k$ (here m is the number of Hellman’s tables). Our time requirements are $T = t/D_k \cdot t \cdot D_k = t^2$ (less tables to check but for more data points), which is the same as in the original Hellman’s trade-off. Finally, the matrix stopping rule is again: $N = mt^2$. Using the matrix stopping rule and eliminating the parameters m and t we get a trade-off formula:

$$N^2 = T(MD_k)^2.$$

This is exactly the same formula as the one derived in [5] for the case of stream ciphers. For example, for the case of AES with 128-bit key, assuming that one is given 2^{32} encryptions of a plaintext “all zeroes” (or any other fixed text, like 16 spaces, “Hello Joe, ” etc.) under different unknown keys, one can recover one of these keys after a single preprocessing of 2^{96} steps, and using 2^{56} memory for table storage and 2^{80} time for the actual key-search³. It is important to note that unlike in Hellman’s original trade-off, the preprocessing time is much lower than the exhaustive search and thus technically this is a break of cipher. Though even better theoretical attacks for block-ciphers exist in this setting [2] they are in direct correspondence to Babbage-Golic “birthday” trade-off attacks and thus suffer from the same lack of flexibility due to $T = D$. Such attack will require impractical amount of 2^{64} fixed text encryptions as well as high storage complexity of 2^{64} . We believe that if one would try to implement these attacks he would prefer to use less data and less memory at the expense of more preprocessing and longer attack time. In Table 2, we summarize complexities of TMD attacks for various schemes. For example we believe that the attack on full Skipjack with 2^{32} fixed plaintexts and 2^{48} preprocessing complexity, 2^{32} memory and time is tempting to implement and to try in practice. Another important observation is that the attack is not exactly a chosen plaintext attack – since the specific value of the fixed plaintext is irrelevant. Thus, in order to obtain the attack faster than exhaustive search the attacker will first check which plaintext is the most frequently used in the specific application, collect the data for various keys and then perform the attack. The attack is technically faster than the exhaustive search even if the attacker obtains a relatively small number of arbitrary fixed text encryptions. For example if the attacker obtains only 2^8 128-bit key AES encryptions, then after preprocessing of 2^{120} steps and using 2^{60} memory and 2^{120} analysis steps, one of the keys would be recovered. In practical applications, it might be a rather non-trivial task to ensure that the attacker never obtains encryptions of 2^8 fixed known plaintexts. This attack is much

³ At the moment of this writing 2^{85} computations is approximately the power of all computers on the internet during 1 year.

better than the existing state of the art attacks on 128-bit AES, which barely break 7-rounds of this cipher. Note that Biham’s attack for the same amount of fixed text would formally have the same 2^{120} total complexity but would require unrealistic amount of memory 2^{120} which is probably the reason why such trade-off attacks have not been viewed as a threat by the crypto community. In addition to all said above note that intentionally malicious protocol design may ensure that some fixed plaintext is always included into the encrypted stream (for example by fixing a header in communication, using communication to a fixed address or using fixed file header as is common in many applications). Results shown in Table 2 compare favorably to the best attacks on such ciphers

Table 2. Comparison of TMD attacks on various ciphers.

Cipher	Key size	Keys (Data)	Time	Memory	Preprocessing
DES	56	2^{14}	2^{28}	2^{28}	2^{42}
Triple-DES	168	2^{42}	2^{84}	2^{84}	2^{126}
Skipjack	80	2^{32}	2^{32}	2^{32}	2^{48}
AES	128	2^{32}	2^{80}	2^{56}	2^{96}
AES	192	2^{48}	2^{96}	2^{96}	2^{144}
AES	256	2^{85}	2^{170}	2^{85}	2^{170}
Any cipher	k	$2^{k/4}$	$2^{k/2}$	$2^{k/2}$	$2^{3k/4}$
Any cipher	k	$2^{k/3}$	$2^{2k/3}$	$2^{k/3}$	$2^{2k/3}$
Any cipher[2]	k	$2^{k/2}$	$2^{k/2}$	$2^{k/2}$	$2^{k/2}$

as DES, Triple-DES, Skipjack and AES. Moreover, the scenario of TMD attacks is much more practical than that of related key attacks as is discussed in more detail in Section 4. We believe that complexities of future cryptanalytic attacks should be benchmarked against the time-memory-key attacks.

Due to the importance of some trade-off points we provide Tables 3–6 for several important ciphers (key lengths) and compare them with best attacks known so far.

Table 3. Trade-off attacks on Skipjack (and any other 80-bit cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^8	2^{60}	2^{42}	2^{72}
BS TMD	FKP	2^{20}	2^{40}	2^{40}	2^{60}
BS TMD	FKP	2^{32}	2^{32}	2^{32}	2^{48}
Biham[2]	FKP	2^{40}	2^{40}	2^{40}	2^{40}
BBS Imp.Diff*[4]	CP	2^{34}	2^{78}	2^{64}	2^{64}

* — the attack breaks 31 out of 32 rounds of Skipjack, the data is encrypted under a single key. FKP – fixed known plaintext, CP – chosen plaintext.

Table 4. Trade-off attacks on 128-bit key AES (and any other 128-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^8	2^{120}	2^{60}	2^{120}
BS TMD	FKP	2^{20}	2^{100}	2^{58}	2^{108}
BS TMD	FKP	2^{32}	2^{80}	2^{56}	2^{96}
BS TMD	FKP	2^{43}	2^{84}	2^{43}	2^{85}
Biham[2]	FKP	2^{64}	2^{64}	2^{64}	2^{64}
GM collision*	CP	2^{32}	2^{128}	2^{80}	?
FSW partial sum*	CP	$2^{128}-2^{119}$	2^{120}	2^{64}	?

* — only 7 out of 10 rounds. FKP – fixed known plaintext, CP – chosen plaintext.

Table 5. Trade-off attacks on 192-bit key AES (and any other 192-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^{48}	2^{96}	2^{96}	2^{144}
BS TMD	FKP	2^{64}	2^{128}	2^{64}	2^{128}
Biham[2]	FKP	2^{96}	2^{96}	2^{96}	2^{96}

FKP – fixed known plaintext.

Table 6. Tradeoff attacks on 256-bit key AES (and any other 256-bit key cipher).

Attack	Data Type	Keys (Data)	Time	Memory	Preprocessing
BS TMD	FKP	2^{64}	2^{128}	2^{128}	2^{192}
BS TMD	FKP	2^{85}	2^{170}	2^{85}	2^{170}
Biham[2]	FKP	2^{128}	2^{128}	2^{128}	2^{128}

FKP – fixed known plaintext.

4 Types of Cryptanalytic Attacks and Key-Size Considerations

Cryptanalytic attacks may be divided into three main classes by the type of access to an encryption/device. In the first class of attacks, which we call *fixed key* attacks, we assume that a black box with the encryption/decryption device is given to the attacker. The attacker is then able to make arbitrary number of queries (with unknown, known or chosen inputs) to the device. The goal of the attacker is to find the secret key of the box, which remains unchanged during the attack. Note that the queries could be performed adaptively (i.e. based on the results of previous queries). For example: differential, linear, boomerang or multiset attacks are of this type. Moreover linear cryptanalysis *requires* a fixed key scenario, while differential, boomerang or multiset attacks may tolerate key changes which are not too frequent during the attack.

The second type of attack which we call *variable key* attacks, assumes that the attacker is given both the black box with the encryption/decryption device as well as a black box of the key-schedule device. The attacker can then perform both the fixed key attacks as well as re-keying the cipher to a new secret key value at any given moment. The goal of the attacker is to find one of the keys of the device. This scenario is strictly more powerful than the fixed key scenario and can be efficiently exploited for example in the “weak key” attacks or in time/memory/key trade-off attacks.

The third type of attacks is what is called *related key* scenario. In this case the attacker is not only allowed to change the keys of the device. He is essentially given access to two or more encryption/decryption devices and he knows or even chooses the relations between the keys used in these devices. This scenario is highly unrealistic in practice but may identify undesirable certification weaknesses in the key-schedule of a cipher.

Applicability of the attack scenarios described above in practice may be hindered by the use of certain *mode of operation* (which for example may preclude the use of chosen plaintext queries) or by the *key-change provision*, which may enforce a key-change every 1000 encryptions thus rendering statistical attacks which assume fixed key scenario — impractical.

4.1 Key-size Consideration

Modern symmetric ciphers typically have keys larger or equal to 128 bits and they assume that exhaustive search is the best way one could recover a secret key⁴.

As this note shows however in a *variable key* scenario no k -bit cipher can offer a k -bit security against some quite practical attacks. One may assume that this

⁴ Depending on the mode of operation used, there are also distinguishing attacks which may require about 2^{64} fixed key data, and do not lead to key-recovery. Those attacks are not considered to be of a threat by the community and are typically taken care of by key-change provisions.

problem can be cured by introducing the IV which has to be of the same size as the key. However in such popular block-cipher modes of operation like CBC due to a simple XOR of the *known* IV with the first plaintext block the attacker capable of mounting chosen plaintext attack can easily obtain encryptions of arbitrary fixed text under different keys. In the less likely but still not impossible case of a chosen IV attack other modes of operation like CFB, OFB or counter mode become vulnerable as well. A careful study of what should be the IV size in order to avoid trade-off attacks is given in [10], however a simple rule of a thumb is that the IV size should be at least equal to the key-size, since the state of the cipher at any given moment has to be twice the key-size in order to avoid birthday time-data attacks [2, 1, 7]. XORing of the IV into the plaintext/ciphertext should be avoided.

Following these simple observations it is clear that 80-bit (or less) key ciphers should not be used since they allow for practical attacks in real-life scenarios, while 128-bit ciphers (which in practice provide security of about 80-bits) should not be used when full 128-bit security is required. At least 192-bit keys should be used for this security level.

One may argue that generic trade-off attacks do not exploit weaknesses of specific designs and thus should be considered separately from other attacks. There are two counter-arguments to this point: first of all we have at the moment no proof that existing trade-off attacks (such as Hellman's attack) are the best possible and thus a popular maxim "The attacks only become better, they do not get worse" may still apply. Moreover trade-off attacks may be sped up by specific properties of the design, for example by what is called in a stream cipher case — cipher's sampling resistance [5]. In the case of stream cipher LILI-128 low sampling resistance was used to obtain trade-off attack [13] with a complexity much lower than a naive application of a trade-off technique would suggest.

It seems that we will have to give up the convenient world in which we assumed a k -bit security for a good k -bit cipher.

5 Application to the Unix Password Scheme

The attacks described in this paper are not limited to block or stream ciphers, they are applicable to other one-way constructions, for example to hash functions.

Time/memory/data trade-off [5] ($N = TM^2D^2$) could be used to analyze Unix password scheme for example, if the attacker obtains access to a file storing password hashes of a large organization ($D = 1000$ password hashes). Indeed the trade-off space consists of 56-bits of the unknown key (i.e. password) and 12-bits of known salt. Since the salt size is much shorter than the key-size its effect on making the trade-off harder is not very significant. Suppose that the attacker knows that passwords are selected from a set of arbitrary 8-character alphanumeric passwords, including capital letters and two additional symbols like dot and comma which in total can be encoded in 48-bits. Thus together with a 12-bit salt the state is $N = 2^{60}$ bits. For example the following attack

parameters seem quite practical: preprocessing time done once: $P = N/D = 2^{50}$ Unix hash computations, parallelizable. A memory of $M = 2^{34}$ 8-byte entries (12+48 bits) which takes one 128 Gbyte hard disk. This way we store 2^{34} start-end pointers. Attack time is then $T = 2^{32}$ Unix hash evaluations — about an hour on a fast PC or about 8 seconds on a BEE2 FPGA [11]. The attack will recover one password from about every 1000 new password hashes supplied. This is two – three orders of magnitude faster than the results described in [11]. The relatively lengthy preprocessing step may be performed in parallel on a network of PC’s (hundred PC’s may take less than a month) or it may take about 1.5 months for a single BEE2 FPGA. The number of tables computed in parallel may be as high as $t/D = 2^{17}/1000 = 2^7$. In order to reduce the number of hard disk accesses the attack will need to use distinguished points with 16-bit prefixes. This will allow to make only 2^{16} disk accesses (which is less than 6 minutes).

In fact it is clear that such trade-off can analyze all passwords typable on a keyboard. The space is $N = 84^8 \cdot 2^{12} = 2^{63}$. Assuming again $D = 2^{10}$, we get precomputation time $P = 2^{53}$, $M = 2^{35}$ 8-byte entries or one 256 Gb hard disk, $T = 2^{36}$ hash evaluations.

Table 7. Trade-off attacks on UNIX password scheme.

Passwords attacked	State Size (bits)	Data	Time	Memory	Preprocessing
Alphanumeric	60	2^8	2^{34}	2^{34} (128 Gb)	2^{52}
Alphanumeric	60	2^{10}	2^{32}	2^{34}	2^{50}
Full keyboard	63	2^{10}	2^{36}	2^{35} (256 Gb)	2^{53}
Alphanumeric ^a [11]	60	1	2^{40}	2^{40}	2^{60}

^a The paper provides analysis for a single fixed salt value.

6 A New Analysis

In this section, our goal is different from that of the previous sections. Here we present a new analysis of the general TMTO methodology described in Section 2. We show that the BG and the BS methods are special cases of the general analysis. In particular, we show that there are certain interesting trade-offs which were not known earlier. Some of these are summarized in Table 8.

Recall from Section 2 that the general description of the TMTO methodology uses r tables each of size $m \times t$. In the BG attack, $r = t = 1$ and hence is not very flexible. The BS attack is more flexible but assumes $mt^2 = N$ and $r = t/D$ leading to the constraint $D^2 \leq T$. The last restriction can prove to be a drawback of the BS method when the amount of available data is high. From the Hellman

Table 8. Some of the new trade-offs.

N	Precomputing time (P)	Memory (M)	Data (D)	Run time (T)
2^{80}	2^{50}	2^{30}	2^{30}	2^{50}
	$2^{46.6}$	$2^{33.3}$	$2^{33.3}$	$2^{46.6}$
2^{100}	$2^{62.5}$	$2^{37.5}$	$2^{37.5}$	$2^{62.5}$
	$2^{58.3}$	$2^{41.6}$	$2^{41.6}$	$2^{58.3}$
* N	$N^{\frac{1+d}{3}}$	$N^{\frac{2-d}{3}}$	$N^{\frac{2-d}{3}}$	$N^{\frac{1+d}{3}}$

* Here d is a constant such that $\frac{1}{2} < d < 1$.

attack we have the following relations.

$$\left. \begin{aligned}
 T_f &= r(t-1)D \quad (\# f \text{ invocations in the online phase}) \\
 T_t &= rtD \quad (\# \text{ table look-ups in the online phase}) \\
 P &= rmt \quad (\# f \text{ invocations in the pre-computation phase}) \\
 &= \frac{N}{D} \quad (\text{coverage}) \\
 M &= rm \quad (\text{memory}) \\
 mt^2 &\leq N \quad (\text{birthday bound})
 \end{aligned} \right\} \quad (1)$$

If $t \gg 1$, we can assume $t-1 \approx t$ and $T_f \approx rtD = T_t$. We will usually make this assumption, except for the analysis of the BG attack, where $t = 1$. Let γ be the ratio of the time required for performing one table look-up to the time required for one invocation of f , i.e.,

$$\gamma = \frac{\text{time for one table look-up}}{\text{time for one invocation of } f}. \quad (2)$$

We assume that one unit of time corresponds to one invocation of f (i.e., one unit of time = time required for completing one invocation of f) and also $\gamma \geq 1$. The time required for the table look-ups is then γrtD . Define $T = \max(T_f, \gamma T_t) = \gamma rtD$. The parameter T is a measure of the time required during the online phase. The actual online time is proportional to $T_f + \gamma T_t$. However, this is only at most twice the value of T . Thus, we will perform the analysis with T instead of $T_f + \gamma T_t$.

For the present, we will assume that $\gamma = 1$ (and $T = T_t \approx T_f$), i.e., the cost of one invocation of f is equal to the cost of one table look-up. The value of γ need not actually be one; even if it is a small constant (or a negligible fraction of N), we can assume it to be one and that will not affect the asymptotic analysis. On the other hand, [5] mentions that γ may be as large as one million ($\approx 2^{20}$). If N is only moderately large (like 2^{64} for A5/1), then γ can be a significant proportion of N . In such a situation, we cannot assume $\gamma = 1$ and the cost of table look-up will dominate the total online cost. This case will be considered later.

Using (1), we can solve for r , m and t as follows.

$$\left. \begin{aligned} t &= \frac{N}{MD} \geq 1 && \text{(number of columns)} \\ m &= \frac{N}{T} && \text{(number of rows)} \\ r &= \frac{MT}{N} \geq 1 && \text{(number of tables)} \\ mt^2 &= \frac{N^3}{TM^2D^2} \leq N && \text{(birthday bound)} \end{aligned} \right\} \quad (3)$$

Note that all three of r , m and t must be at least 1. Since $m = N/T$ and for a valid attack we must have $N > T$, the condition on m is trivially satisfied. The advantage of writing in the form of (3) is that given values for T , M and D satisfying the proper constraints, we can immediately design a table structure which achieves these values.

6.1 TMTO Curve

From Equations (3), we know $MT \geq N$ and $mt^2 \leq N$. Further, for a feasible attack we must have $1 \leq D < N$ and $M, T < N$. We capture these in the following manner:

$$D = N^a; \quad MT = N^b; \quad M = N^c; \quad mt^2 = N^d; \quad (4)$$

with $0 \leq a, c < 1$, $0 \leq d \leq 1$ and $b \geq 1$. Consequently, we have $P = N/D = N^{1-a}$; $T = N^b/M = N^{b-c}$, with $0 \leq b - c < 1$. Further, $t \geq 1$ implies $MD \leq N$ and hence $a + c \leq 1$. Substituting in the last equation of (3), we obtain $2a + b + c + d = 3$. Thus, any set of values for a, b, c and d which satisfy the following constraints constitute a valid attack.

$$\left. \begin{aligned} \mathbf{C1:} & 2a + b + c + d = 3 \\ \mathbf{C2:} & 0 \leq a < 1 \\ \mathbf{C3:} & 0 \leq c, b - c < 1 \leq b \\ \mathbf{C4:} & a + c \leq 1 \\ \mathbf{C5:} & 0 \leq d \leq 1 \end{aligned} \right\} \quad (5)$$

The so-called TMTO curve can be obtained as the following relations.

$$\left. \begin{aligned} TM^2D^2 &= N^{3-d} \\ PD &= N \\ MD &\leq N \leq MT \\ M, D, T &< N. \end{aligned} \right\} \quad (6)$$

Also, we have the following values of r , m and t .

$$r = N^{b-1}; \quad m = N^{1-(b-c)}; \quad t = N^{1-a-c}. \quad (7)$$

Since $MT = N^b \geq N$, we have $r = 1$ if and only if $MT = N$. With $r = 1$, we have only one table and hence if there are more than one tables, then MT is strictly greater than N .

BG Attack [1, 7]: In this case, we have $r = t = 1$. This implies $T_f = 0$, i.e., the online phase does not require invocation of f . The cost in the online phase is $T = T_t$ and we have $MD = N = MT$ and hence $T = D$; $M = N/D$. This corresponds to the conditions $a + c = 1$; $b = 1$; $d = 1 - a$.

BS Attack [5]: In [5], $r = t/D$ and $d = 1$ is used. Then $T = t^2$, $M = mt/D$ and hence $r = N^{-a+(b-c)/2}$. Since $r \geq 1$, we have the restriction $0 \leq 2a \leq b - c$ (i.e., $1 \leq D^2 \leq T$) in addition to (5).

The conditions $d = 1$ and $r = t/D$ are related (e.g., if $r = 1$, then $t = D$ and $T = t^2 = D^2$). In the following analysis, we will proceed without these two conditions. Later, we show the situation under which making these two assumptions is useful.

7 Distinguished Point Method

We now consider the case where $\gamma \gg 1$. In this case, a direct application of the Hellman method leads to $T = \gamma rtD$, i.e., the time required for the table look-ups dominate the online time. It is useful to consider the distinguished point method of Rivest to reduce the number of table look-ups. See [5] for a description of the DP method.

Using the distinguished point method results in reducing the number of table look-ups from rtD to rD , i.e., one table look-up per table per data. Then $T_t = rD = N^{a+b-1}$. (Note $T_t = N^a = D$, i.e., only one table look-up is required per data item if and only if $b = 1 = r$, i.e., $MT = N$.)

The total cost of the table look-ups is γrD whereas the cost of invoking the one-way function is rtD . In this case, the ratio of the two costs is γ/t . If $t \geq \gamma$, then the ratio is at most one. Hence, we can again ignore the cost of table look-up and perform the analysis by considering simply the cost of invoking the one-way function. The actual runtime will be at most twice the runtime obtained by such an analysis.

Suppose $t < \gamma$. Then the analysis performed above does not hold. We now investigate the situation under which $t < \gamma$ holds. This certainly holds for $t = 1$ (the BG attack), but in the BG attack the entire online computation consists of table look-ups and hence the general analysis is not required. Recall that $t = N^{1-(a+c)} = 2^{n(1-(a+c))}$, $D = N^a$ and $M = N^c$. Suppose $\gamma = 2^e$. Then $t \geq \gamma$ if and only if $a + c \leq 1 - (e/n)$. The value of e is a constant whereas n increases. Hence, $(1 - e/n) \rightarrow 1$ as n grows. Thus, we can have $a + c > 1 - e/n$ only for small values of n . The smallest value of n for which we can expect to have a secure cryptographic algorithm is 64. Further, as mentioned in [5], e can be at most around 20 and so $1 - e/n \geq 2/3$ for $n \geq 64$.

Consider $a = c = 1/3$, as in the solution $(a, b, c, d) = (1/3, 1, 1/3, 1)$ corresponding to $P = T = N^{2/3}$; $M = D = N^{1/3}$; $r = 1$ of [5]. If $n \geq 64$, then $a + c = 2/3 \leq 1 - e/n$ and the time analysis assuming $T = rtD = tD$ holds. On the other hand, for the solution $(a, b, c, d) = (3/8, 1, 3/8, 7/8)$ corresponding to $P = T = N^{5/8}$; $M = D = N^{3/8}$; $r = 1$ considered in Section 8, we have

$a + c = 3/4$. For $n = 64$, $a + c > 1 - e/n$ and we have to assume $T = \gamma r D = \gamma D$, whereas for $n = 100$, $a + c \leq 1 - e/n$ and we can assume $T = r t D = t D$. Thus, for relatively small n , we should solve (5) with the constraint $a + c \leq 1 - e/n$ instead of $a + c \leq 1$. This disallows some of the otherwise possible trade-offs.

There is another issue that needs to be considered. We have to ensure that t is large enough to ensure the occurrence of a DP in a chain. Let 2^{-p} be the probability of a point being a DP. Hence, we can expect one DP in a random collection of 2^p points. Thus, if $t \geq 2^p$, we can expect a DP in a chain of length t . This implies $p \leq \log_2 t$. Any attempt to design the tables with $t < 2^p$, will mean that several trials will be required to obtain a chain terminating in a DP. This will increase the pre-computation time. In fact, [5] has shown that use of the DP method in the BG attack divides into two different trade-offs leading to unrealistic requirements on data and memory.

Using (7), we have $p/n \leq 1 - (a + c)$. This leads to the condition $a + c \leq 1 - p/n$ ($MD \leq N^{1-p}$) instead of the condition $a + c \leq 1$ (resp. $MD \leq N$) in (5) (resp. (6)). For small n , this condition has to be combined with $a + c \leq 1 - e/n$ and we should solve (5) with the constraint $a + c \leq 1 - 1/n \times \max(p, e)$ instead of the constraint $a + c \leq 1$. This puts further restrictions on otherwise allowed trade-offs.

BSW Sampling There is an elegant application of TMTO in [6], which uses a special type of sampling technique called the BSW sampling. This technique uses only part of the available online data and also reduces the search space. The trade-off curve does not change, but the number of table look-ups reduces significantly. Use of this technique allowed particularly efficient attacks on A5/1.

Use of the BSW technique reduces the amount of available online data. This makes it difficult to use a single table to carry out the TMTO. In such a situation, our analysis does not lead to any new insight into the BSW technique. On the other hand, if the available online data (even after sampling) is large enough to allow the use of a single table, then our analysis applies and one can consider a wider variety of trade-offs.

8 Single Table Trade-Offs

The case $N = 2^{100}$ has been considered in [5]. It has been mentioned in [5] that the Hellman attack with $D = 1$; $T = M = N^{2/3} = 2^{66}$ requires unrealistic amount of disk space and the BG attack with $T = D = N^{2/3} = 2^{66}$; $M = N^{1/3} = 2^{33}$ requires unrealistic amount of data. (Note $T = M = D = N^{1/2} = 2^{50}$ also gives a BG attack. However, as mentioned in [5] in a different context, data and memory requirement of more than 2^{40} is unrealistic.) Further, [5] mentions $P = T = 2^{66}$ and $D = M = 2^{33}$ to be a (barely) feasible attack. This corresponds to the parameters $(a, b, c, d) = (1/3, 1, 1/3, 1)$ and $(r, m, t) = (1, N^{1/3}, N^{1/3})$.

From Proposition 2, if we choose $d = 7/8$, then we obtain $M = D = N^{3/8} = 2^{37.5}$ and $P = T = N^{5/8} = 2^{62.5}$. The corresponding parameters are $(a, b, c, d) = (3/8, 1, 3/8, 7/8)$ and $(r, m, t) = (1, N^{3/8}, N^{1/4})$. This brings down the attack

time while keeping the data and memory within feasible limits. Since $t > 1$, this cannot be obtained from the BG attack. Further, choosing $d = 7/8$ and $D^2 > T$ ensures that this attack cannot also be obtained from the BS attack. We would like to point out that [5] mentions that choosing $d < 1$ is “wasteful”. The above example shows that this is not necessarily the case and choosing $d < 1$ can lead to more flexible trade-offs. We show below the condition under which choosing $d < 1$ is indeed “wasteful”.

As mentioned earlier, we have one table (i.e., $r = 1$) if and only if $MT = N$. The reason for moving to more than one tables is when $mt^2 > N$ and we begin to have more and more repetitions within a table.

Proposition 1. *There is a solution to (6) with $r = 1 = b$ (and hence $MT = N = PD$) if and only if $2a + c \geq 1$.*

Proof : Suppose $r = 1$. Then $b = 1$ and $2a + c + d = 2$. Hence $d = 2 - (2a + c)$. Since $d \leq 1$, this shows $2a + c \geq 1$.

On the other hand assume that $2a + c \geq 1$. Choose $b = 1$ and set $d = 2 - (2a + c) \leq 1$. This choice satisfies the conditions of (6). Further, since $b = 1$, we have $r = 1$. \square

Suppose $2a + c < 1$. Then $b + d > 2$ and $b > 2 - d$. Since $MT = N^b$, we would like to minimize b and hence we choose $d = 1$. We can now modify the suggestion of [5] and say that it is “wasteful” to choose $mt^2 < N$ if there are more than one tables. Since $b > 1$, we have $2a + c < 1 < b$ and hence $2a < b - c$ which gives $D^2 < T$ and we are back to the situation described in [5].

Thus, the analysis of [5] actually applies to the situation where the data is small enough to require more than one tables. On the other hand, for the case of one table, the restrictions of [5] are not required and removing these restrictions provide more flexible trade-offs. We would like to point out that there are interesting situations where a single table can be used. Apart from the examples $D = M = N^{1/3}$ and $D = M = N^{3/8}$ already considered, other possible examples are $(D = N^{0.3}, M = N^{0.4})$; $(D = N^{0.25}, M = N^{0.5})$, etcetera.

8.1 Small N

Consider $N = 2^{64}$, as in A5/1. It is mentioned in [6] that $D \approx 2^{22}$ is a reasonable choice. Further, $M \approx 2^{40}$ is also feasible. We consider possible values of P and T satisfying these values of M and D .

Trade-Off 1: $(P, D, M, T) = (2^{48}, 2^{16}, 2^{40}, 2^{24})$: The table parameters are $(r, m, t) = (1, 2^{40}, 2^8)$ and $d = 7/8$.

Trade-Off 2: $(P, D, M, T) = (2^{42}, 2^{22}, 2^{40}, 2^{24})$: The table parameters are $(r, m, t) = (1, 2^{40}, 2^4)$ and $d = 11/16$.

None of the above two trade-offs are obtainable as BG attacks, since in both cases $t > 1$. Further, neither can any of them be obtained as BS trade-offs since in both cases $d < 1$ and hence $D^2 > T$. For both trade-offs, the data and memory are within reasonable limits and the online times are the same. The offline time is lower for the second trade-off and is within doable limits (especially as an offline one-time activity), while for the first attack it is probably just outside the

doable limit. Hence, both the attacks are feasible for any 64-bit function and hence also for A5/1. However, as mentioned before, using special properties of A5/1, it is possible to obtain faster attacks as in [6].

9 Certain Special Cases

Here we consider in detail two special cases. These should be considered to be mainly of theoretical interest. The first condition of $T = M$ was originally considered by Hellman in the situation where $D = 1$, while the second condition was briefly considered in [5] for the case $N = 2^{100}$.

9.1 Condition $T = M$

The condition $T = M$ was considered by Hellman [8]. We perform an analysis of (6) with $T = M$. Then $c = b - c$, whence $c = b/2$. Condition **C1** becomes $2a + 3c + d = 3$ and so $\frac{b}{2} = c = 1 - \frac{2a+d}{3}$. Using $a + c \leq 1$, we obtain $a \leq d$. Also since $b \geq 1$, we have $c = b/2 \geq 1/2$. This gives $d \leq 3/2 - 2a$. Since, we already know $d \leq 1$, we obtain $a \leq d \leq \min(1, \frac{3}{2} - 2a)$. Thus, any non-negative solution in a and d to this gives a valid attack with $T = M = N^c$.

We are interested in minimizing the value of c . We see that the value of c is minimized by maximizing the value of d . In fact, we can choose $d = 1$ as long as $1 \leq \frac{3}{2} - 2a$, i.e., $2 - (1/2a) \leq 0$ or $a \leq 1/4$. Thus, for $a \leq 1/4$, we obtain $T = M = N^{b/2} = N^{(2-2a)/3}$.

In the case $3/2 - 2a \leq 1$, we have $a \leq d \leq 3/2 - 2a$. For the gap to be non-empty we must have $a \leq 1/2$. For minimizing c , we use the upper bound, i.e., $d = 3/2 - 2a \leq 1$. Thus, for $1/4 \leq a \leq 1/2$, we have $c = 1/2$ and $T = M = N^{1/2}$. Finally, we obtain the following result.

Theorem 1. *If $T = M$, then $D \leq N^{1/2}$ and the following conditions hold.*

1. $N^{1/2} \leq T = M = N^{(2-2a)/3} \leq N^{2/3}$, for $1/4 \geq a \geq 0$.
2. $T = M = N^{1/2}$, for $1/4 \leq a \leq 1/2$.

For the first case we have, $(a, b, c, d) = (a, 2(2 - 2a)/3, (2 - 2a)/3, 1)$ and for the second case we have $(a, b, c, d) = (a, 1, 1/2, 3/2 - 2a)$. The corresponding values of (r, m, t) are $(N^{(1-4a)/3}, N^{(1+2a)/3}, N^{(1-a)/3})$ and $(1, N^{1/2}, N^{1/2-a})$ respectively.

In the second case of Theorem 1, exactly one table is required. However, it is not the BG attack, since the number of columns can be more than one. Also, we have $T \leq P \leq N$. The situation with $T < P < N$ is interesting, since the pre-computation time is less than exhaustive search. Even though P is more than T , since it is an offline activity, we might wish to spend more time in the pre-computation part than in the online attack.

In the second case of Theorem 1, we have $r = 1$ and $M = T = N^{1/2}$. The allowed range of a for this case is $1/4 \leq a \leq 1/2$. The case $a = 1/4$ can be obtained from the BS analysis and the case $a = 1/2$ can be obtained from the BG analysis. However, the range $1/4 < a < 1/2$ for which $T = M = N^{1/2}$ can be

attained, cannot be obtained from either the BG or the BS analysis and provide previously unknown trade-offs. The advantage is that the data can be increased (thus lowering offline time) without increasing either time or memory.

9.2 Condition $P = T$

Since both P and T represent time, the case $P = T$ puts equal emphasis on both the offline and the online times. The condition $P = T$ implies $P = N^{1-a} = T = N^{b-c}$ and so $m = N^{1-(b-c)} = N^a = D$. (On the other hand, $P = M$ is possible only if $t = 1$.) Since $PD = N$, we have $T = N/D$ and so the curve becomes $M^2D = N^{2-d}$. If $P = T$, then $r = M/D$. If further $M = D$, then $M = D = N^{(2-d)/3}$ and $P = T = N^{(1+d)/3}$.

Proposition 2. *If $P = T$ and $M = D$ in (6), then $M = D = N^{(2-d)/3}$ and $P = T = N^{(1+d)/3}$. Further, $r = 1$, i.e., exactly one table is required.*

Proposition 2 gives us a nice way to control the trade-off between time and data/memory requirement by varying d . Choosing $d = 1$, corresponds to choosing $(P, D, M, T) = (N^{2/3}, N^{1/3}, N^{1/3}, N^{2/3})$ and has been observed in [5]; choosing $d = 1/2$ corresponds to $(P, D, M, T) = (N^{1/2}, N^{1/2}, N^{1/2}, N^{1/2})$ which is the square root birthday (BG) attack.

From Proposition 2, we have $r = 1$, i.e., all trade-offs attaining this condition use a single table. In the plausible situation, $M = D \leq P = T$, we have $1/2 \leq d \leq 1$. The case $d = 1$ can be obtained from the BS analysis. In the BG analysis, we have $d = 1 - a$. Since $a - (2 - d)/3$, this condition leads to $d = 1/2$. Thus, the range $1/2 < d < 1$ for which the condition $P = T = N^{(1+d)/3}$; $M = D = N^{(2-d)/3}$ can be attained was not known earlier.

10 The Rainbow Attack

The rainbow attack was introduced in [12]. The number of table look-ups of the rainbow method is comparable to that of the Hellman+DP method. See [12] for a discussion of the relative advantages of the rainbow method with respect to the DP method.

In the rainbow attack, we use a table of size $m \times t$ and suppose there are D online data points. Then the total number of invocations of the one-way function is $t^2D/2$ while the cost of the table look-ups is tD . Again, we will ignore the factor of two in the runtime since it does not significantly affect the analysis. Then, the total number of invocations of f is t^2D and the total number of table look-ups is tD . Also, we have $mt = N/D$.

If we assume $\gamma \approx 1$, then the cost of invoking f dominates the online cost and we have $M = m$ and $T = t^2D$. Assume $D = N^a$ and $M = N^c$ as in the case of Hellman analysis. Then since $mt = N/D = N^{1-a}$, we have $t = N^{1-a-c}$ and $T = t^2D = N^{2-a-2c}$. Also, since $t \geq 1$, we must have $a + c \leq 1$. The TMTO curve for rainbow in the presence of multiple data is $TM^2D = N^2$ which is inferior to the Hellman TMTO curve when $D > 1$.

The rainbow parameters are $(P, D, M, T) = (N^{1-a}, N^a, N^c, N^{2-a-2c})$. We now compare the rainbow parameters with the Hellman parameters for same data and memory. For multiple table Hellman, we choose $d = 1$ and hence the corresponding Hellman parameters are $(P, D, M, T) = (N^{1-a}, N^a, N^c, N^{2-2a-2c})$. If $a > 0$, i.e., if multiple data is available, then clearly Hellman time is less than rainbow time.

If γ is a significant fraction of N , then the cost of table look-ups is $\gamma t D$ while the cost of invoking f is still $t^2 D$. In the case $\gamma > t$, which happens for relatively small N (around 2^{64} or so), the cost of table look-up dominates the online cost. To compare to the Hellman method we have to consider the Hellman+DP algorithm. For the case $\gamma > t$, the online cost of the Hellman method is also $\gamma t D$. Hence, for this case, the costs of online time for the rainbow and the Hellman+DP methods are equal. In this situation, one might prefer to use the rainbow method for the possibly lower rate of false alarms compared to the DP method [12].

Thus, we conclude that in the presence of multiple data, in general the Hellman attack is better than the rainbow attack. For the case of small N , the online times of both attacks can be comparable and one might prefer rainbow for obtaining other possible advantages.

11 Conclusion

In this paper, we have considered two aspects of time/memory/data trade-offs: 3 new application and new analysis.

We show several applications to block-ciphers. By applying Biham-Biryukov [5] multiple data trade-off to block ciphers we show that 80-bit ciphers allow practical attacks in real world scenarios (2^{32} data, memory and time, with 2^{48} steps for preprocessing), while 128-bit ciphers provide only about 80-bits of security against attacks with practical amounts of data and memory. We further show realistic attacks on Unix password hashing scheme even if strong random passwords are chosen.

In the second part of the paper we provide general analysis which shows that Hellman's attack, Babbage-Golic attack and the Biryukov-Shamir all fit into a single unifying general framework. Our new contribution is the identification of a new class of single table trade-offs which are not obtainable as either the BG or the BS attacks. Finally, we consider the rainbow attack of Oechslin and show that with the utilization of multiple data, the TMTO curve of the rainbow attack is inferior to the TMTO curve of the Hellman attack.

References

- [1] S. Babbage, "Improved "exhaustive search" attacks on stream ciphers," in *ECOS 95 (European Convention on Security and Detection)*, no. 408 in IEE Conference Publication, May 1995.

- [2] E. Biham, “How to decrypt or even substitute DES-encrypted messages in 2^{28} steps,” *Information Processing Letters*, vol. 84, pp. 117–124, 2002.
- [3] Eli Biham, “How to Forge DES-Encrypted Messages in 2^{28} Steps”, Technical report CS 884, August 1996.
- [4] E. Biham, A. Biryukov, and A. Shamir, “Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials,” in *Proceedings of Eurocrypt’99* (J. Stern, ed.), no. 1592 in Lecture Notes in Computer Science, pp. 12–23, Springer-Verlag, 1999. To appear in the Journal of Cryptology.
- [5] A. Biryukov and A. Shamir, “Cryptanalytic time/memory/data trade-offs for stream ciphers,” in *Proceedings of Asiacrypt’00* (T. Okamoto, ed.), no. 1976 in Lecture Notes in Computer Science, pp. 1–13, Springer-Verlag, 2000.
- [6] A. Biryukov, A. Shamir and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC, Proceedings of Fast Software Encryption 2000.
- [7] J. D. Golic, “Cryptanalysis of alleged A5 stream cipher,” in *Advances in Cryptology – EUROCRYPT’97* (W. Fumy, ed.), vol. 1233 of *Lecture Notes in Computer Science*, pp. 239–255, Springer-Verlag, 1997.
- [8] M. E. Hellman, “A cryptanalytic time-memory tradeoff,” *IEEE Transactions on Information Theory*, vol. 26, pp. 401–406, 1980.
- [9] J. Hong and P. Sarkar, “Time memory tradeoff attacks on streamciphers,” 2004. Rump session talk at ASIACRYPT’04.
- [10] J. Hong and P. Sarkar, “Rediscovery of time memory tradeoffs,” 2005. <http://eprint.iacr.org/2005/090>.
- [11] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, “Cracking Unix passwords using FPGA platforms,” 2005. Presented at SHARCS’05, in submission.
- [12] P. Oechslin, “Making a faster cryptanalytic time-memory trade-off,” in *Advances in Cryptology – CRYPTO 2003* (D. Boneh, ed.), vol. 2729 of *Lecture Notes in Computer Science*, pp. 617–630, Springer-Verlag, 2003.
- [13] M.-J. O. Saarinen, “A time-memory trade-off attack against LILI-128,” in *Proceedings of Fast Software Encryption – FSE’02* (J. Daemen and V. Rijmen, eds.), no. 2365 in Lecture Notes in Computer Science, pp. 231–236, Springer-Verlag, 2002.