Some Thoughts on Time-Memory-Data Tradeoffs

Alex Biryukov

Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10,
B–3001 Heverlee, Belgium

Abstract. In this paper we show that Time-Memory tradeoff by Hellman may be extended to Time-Memory-Key tradeoff thus allowing attacks much faster than exhaustive search for ciphers for which typically it is stated that no such attack exists. For example, as a result 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 also show that UNIX password scheme even with carefully generated passwords is vulnerable to practical tradeoff attacks. Finally we also demonstrate a combination of rainbow tables with the time-memory-data tradeoff which results in a new tradeoff curve.

1 Introduction

Hellman's tradeoff [6] is a well-known way to invert arbitrary one-way functions. In the context of block ciphers with reasonably long keys this 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 the case of stream ciphers, where two different tradeoffs both involving data are available: the Babbage-Golic Time-Memory, Data-Memory tradeoff [1, 5] and a more flexible Time-Memory-Data tradeoff by Biryukov-Shamir [4]. More importantly, precomputation in these attack is way below the exhaustive key search complexity.

It is easy to see that all the reasoning from the Time-Memory-Data tradeoff in the case of stream ciphers [4] 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. Thus we will use t/D_k tables instead of t, which means our 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 tradeoff. Finally the matrix stopping rule is: $N = mt^2$ which is the

condition to minimize the waste of matrix coverage due to birthday paradox effects. Using the matrix stopping rule and eliminating the parameters m and t we get a tradeoff formula:

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

This is exactly the same formula as the one derived in [4] 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-search[1]. It is important to note that unlike in Hellman's original tradeoff 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" tradeoff 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 1 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 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 tradeoff 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 1 compare favorably to the best attacks on such ciphers

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

Table 1. 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 Sect. 3. We believe that complexities of future cryptanalytic attacks should be benchmarked against the time-memory-key attacks.

Due to the importance of some tradeoff points we provide Tables 2–5 for several important ciphers (key lengths) and compare them with best attacks known so far.

Table 2. Tradeoff 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*[3]	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.

2 Related Work

In [2] Biham shows that theoretic strength of a block-cipher in ECB mode cannot exceed the square root of the size of the key space. What he shows is in fact a Time-Memory tradeoff $N = T \cdot M$ for the case when encryptions of a fixed plaintext under many varying keys are available. Interestingly this tradeoff is a direct analogy of the Babbage-Golic tradeoff for the case of stream ciphers (and discovered approximately at the same time). The tradeoff presented in this paper is an analogy of the Time-Memory-Data tradeoffs for stream ciphers given by Biryukov-Shamir [4]. See Table 6 for a description of this analogy.

Table 3. Tradeoff 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 4. Tradeoff 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 5. 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.

Table 6. Relation between block and stream cipher tradeoff 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, 5]
tradeoff	this paper and [8]	Biryukov-Shamir TMD [4, 8]

Both tradeoff attacks can be avoided by increasing the "state-size" of the cipher for example by introducing random IVs, or using any other way to make the state-size twice the keysize.

At the rump session of ASIACRYPT'04 Hong and Sarkar [7] have demonstrated that stream ciphers with short IVs can be attacked via the Biryukov-Shamir time-memory-data tradeoff [4] in a frequent re-synchronization scenario. More recently in [8] they also provide a careful study of time-memory-data tradeoff attack in the context of various modes of operation of block-ciphers noticing that these essentially constitute a stream cipher. However we believe that they overlooked important cases of ECB (a mode typically assumed in theoretical cryptanalysis) or CBC with known IV's 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.

3 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 tradeoff 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 certificational 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.

3.1 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 key[2].

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 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 tradeoff attacks is given in [8], 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, 5]. 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 tradeoff 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 tradeoff 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 tradeoff 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 [4]. In the case of stream cipher LILI-128 low sampling resistance was used to obtain tradeoff attack [11] with a complexity much lower than a naive application of a tradeoff 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.

[2] 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.

4 Time-Memory-Data tradeoffs with Rainbow Tables

In [10] Oechslin has proposed an alternative way to perform Hellman's tradeoff. His idea is to use a single table of mt starting points (called rainbow table), in which reduction function is changed every column. He has shown that this way one does not obtain asymptotic improvements, but practical implementations become easier and faster by a constant factor 2-8.

It is natural to check how rainbow tables would perform in the context of time-memory-data tradeoff if the attacker is given $D > 1$ data points (encryptions of the same plaintext under different keys in the case of block ciphers or stream prefixes produced by different (key, IV) pairs in the case of stream ciphers). It seems that this way one gets a flexible generalization of "birthday" tradeoff.

The main difference between Hellman's and Oechslin's approaches is the use of multiple small tables in the first case versus a single (rainbow) table in the later case. Inside each table in Hellman's case a single function is used, while in Oechslin's case each column has a different function. Given two parameters m and t the rainbow table has mt rows and t columns, and requires $M = mt$ memory for storage, and $T = t^2/2$ time for scanning. The matrix stopping rule is $N = mt^2$, which results in the same tradeoff formula as in Hellman's case up to a small constant speedup factor.

A natural way to cut the coverage of the rainbow table would be thus to cut its width. If we cut the width of the table from t to t/D, the time becomes $T \approx \frac{1}{2}(\frac{t}{D})^2 \cdot D = \frac{t^2}{2D}$. This is under condition that $t^2 > D^2$ and thus $T > D$. Memory and space size are parameterized as follows: $M = mt$ and $N = mt^2$ and thus after eliminating the parameters we get a tradeoff:

$$N^2 = M^2 T D, \qquad P = \frac{N}{D}, \qquad T > D.$$

This tradeoff formula is not particularly exciting compared to the other TMD tradeoff curve discussed in this paper though it is still more flexible than the "birthday" tradeoff curve.

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 tradeoff [4] ($N = TM^2D^2$) 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 tradeoff 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 tradeoff 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 [9]. 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 [9]. 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 tradeoff 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. Tradeoff 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][9]	60	1	2^{40}	2^{40}	2^{60}

[a] The paper provides analysis for a single fixed salt value.

6 Summary

In this paper we show several applications of Time-Memory-Data tradeoffs [4] to block-ciphers. As a simple application of this technique we argue 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. We also show practical attacks on Unix password hashing scheme even if strong passwords are chosen. Finally we apply TMD tradeoffs to rainbow tables [10].

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] 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.

[4] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs 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.

[5] 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.

[6] M. E. Hellman, "A cryptanalytic time-memory tradeoff," *IEEE Transactions on Information Theory*, vol. 26, pp. 401–406, 1980.

[7] J. Hong and P. Sarkar, "Time memory tradeoff attacks on streamciphers," 2004. Rump session talk at ASIACRYPT'04.

[8] J. Hong and P. Sarkar, "Rediscovery of time memory tradeoffs," 2005. `http://eprint.iacr.org/2005/090`.

[9] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "Cracking Unix passwords using FPGA platforms," 2005. Presented at SHARCS'05, in submission.

[10] 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.

[11] 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.