New Applications of Time Memory Data Tradeoffs

Jin Hong[1] and Palash Sarkar[2]

[1] National Security Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-350, Korea
jinhong@etri.re.kr
[2] Cryptology Research Group, Applied Statistics Unit,
Indian Statistical Institute, 203, B.T. Road, Kolkata, India 700108
palash@isical.ac.in

Abstract. Time/memory tradeoff (TMTO) is a generic method of inverting oneway functions. In this paper, we focus on identifying candidate oneway functions hidden in cryptographic algorithms, inverting which will result in breaking the algorithm. The results we obtain on stream and block ciphers are the most important ones. For streamciphers using IV, we show that if the IV is shorter than the key, then the algorithm is vulnerable to TMTO. Further, from a TMTO point of view, it makes no sense to increase the size of the internal state of a streamcipher without increasing the size of the IV. This has impact on the recent ECRYPT call for streamcipher primitives and clears an almost decade old confusion on the size of key versus state of a streamcipher. For blockciphers, we consider various modes of operations and show that to different degrees all of these are vulnerable to TMTO attacks. In particular, we describe multiple data chosen plaintext TMTO attacks on the CBC and CFB modes of operations. This clears a quarter century old confusion on this issue starting from Hellman's seminal paper in 1980 to Shamir's invited talk at Asiacrypt 2004. We also provide some new applications of TMTO and a set of general guidelines for applying TMTO attacks.
Keywords: time memory data tradeoff

1 Introduction

Time memory tradeoff (TMTO) algorithm is a generic method of inverting otherwise well behaved oneway functions. The technique of using TMTO to invert oneway function was introduced by Hellman in his seminal paper [16] on the topic in 1980. This topic has two parts.

TMTO Algorithms: This covers development of new TMTO algorithms including use of multiple data and investigation of theoretical issues about general TMTO algorithms. Apart from Hellman's work, other contributions to this line of research include Rivest's idea of distinguished points, Fiat-Naor [10], Babbage [4], Golić [12], Biryukov-Shamir [7], Oechslin [23] and Kim-Matsumoto [20]. In this work, we will use some of the relevant results from the above papers, but we will not present any new contribution to this area.

TMTO Applications: Our contribution is to this area of TMTO research. As mentioned before, TMTO is applied to invert oneway function. Therefore an important question is to identify a target oneway function on which to apply TMTO. The initial work by Hellman [16] is a *chosen plaintext* attack and applies TMTO to the oneway function which maps the keyspace to the cipherspace by encrypting an *a priori* chosen message using a blockcipher. The work of Babbage [4], Golić [12] and Biryukov-Shamir [7] applies TMTO to the oneway function which maps the internal state space to a keystream segment of a streamcipher. See [13] for an adaptation of this application to the state space of a PRNG.

We would like to point out that this clear distinction between TMTO algorithm and the oneway function on which to apply it does not appear explicitly in the literature. On the other hand, with this distinction made clear one begins to search for suitable oneway functions hidden in cryptographic algorithms on which to apply TMTO.

In this paper, we present a systematic investigation of the above line of research. We consider a wide range of cryptographic algorithms and look for candidate oneway functions for TMTO applications. Our results on stream and block ciphers are the most interesting and also turns out to be quite important as discussed below. We also consider hash functions and asymmetric algorithms and finally describe a set of guidelines for applying TMTO to cryptographic algorithm. Due to lack of space, the last description as well as some of the other details are given in the Appendix. We next describe our contributions to stream and block ciphers.

1.1 Streamcipher

As mentioned before, the works of Babbage [4], Golić [12], and Biryukov-Shamir [7] have applied TMTO to the oneway function mapping internal state to a keystream segment. A suggested countermeasure for resisting TMTO has been to use a state whose size is double that of the key size. This can be seen from the following quote from [12].

> "... doubling the memory size, from 64 to 128 bits, is very likely to push the attacks beyond the current technological limits. Note that the secret session key size need not be increased to 128 bits."

Over the last few years, this has led streamcipher designers to incorporate huge internal states. Also, most recent streamcipher proposals have quoted their huge state size as indications of resistance to TMTO attacks.

We revisit TMTO on streamciphers. Most streamciphers use an initialization vector (IV) in addition to the secret key. We show that the function mapping (key, IV) to a keystream segment of suitable length is a candidate oneway function for TMTO application. In the case where the key is longer than the IV, the algorithm becomes vulnerable to TMTO *irrespective of the size of the internal state.* Thus, huge state size *does not* guarantee resistance to TMTO attacks. This

clears an almost decade old confusion on this issue. Further, our results shows that it does not make sense to increase the state size without a corresponding increase of the IV size. These results have been considered important enough to bring about a change in the recent ECRYPT call for streamcipher primitives.

Prior to our work, the only oneway function in a streamcipher considered for TMTO application was the state to keystream map. Our work shows that the (key, IV) to keystream map is another such function. It is an interesting problem to identify other possible candidate functions. Such functions may not be generic to all streamciphers (as the above two are), but may also be algorithm specific.

1.2 Blockcipher

Blockciphers are mostly used in an appropriate mode of operation. Hellman's attack applies to the ECB mode of operation. There is widespread belief in the cryptographic community that the following two points are true.

1. It is not possible to use multiple data with blockcipher tradeoffs.
2. Cipher block chaining with random IVs will foil tradeoff attacks on blockciphers.

The following quote from the invited talk by Adi Shamir at Asiacrypt 2004 [25] suggests that the first of these is a well settled fact (and not even an open problem).

> "Generic time/memory tradeoff attacks on stream ciphers ($TM^2D^2 = N^2$) are stronger than the corresponding attacks on block ciphers ($TM^2 = N^2$) since they can exploit the availability of a lot of data."

(In fact, the above statement was provided as one of the evidences that blockciphers are stronger than streamciphers.)

The second point is explicitly stated in Hellman's paper [16]. We quote the relevant portions from Hellman's paper. The first of these appears on Page 404, second column, third paragraph.

> "It should be remembered, however, that the time-memory trade-off does not work in a known plaintext attack if block chaining or cipher feedback is used. . . "

This appears even more explicitly on Page 405, second column, third paragraph of Section IV.

> "Even a block cipher can foil the time-memory trade-off in a known plaintext attack through cipher block chaining [7], [8] or other techniques which introduce memory into encipherment. . . . Again, proposed standards include provision for cipher block chaining with a random indicator."

The last sentence suggests that using CBC with a random IV will resist TMTO attacks. There is some confusion between chosen and known plaintext attacks. While the TMTO attack on the ECB mode developed by Hellman is itself a *chosen* plaintext attack, the above comments relate only to *known* plaintext TMTO attacks. We discuss this point in more details in Appendix A.

We investigate the possibility of TMTO application on various block cipher modes of operations. For every mode of operation that we consider, it turns out that there is a suitable oneway function to which *chosen plaintext* TMTO can be applied under appropriate conditions. The most interesting results are for the CBC and the CFB modes of operations. Contrary to Shamir's statement above on the use of multiple data, we show how to apply nontrivial multiple data TMTO to both the CBC and CFB modes of operations. Further, our results show that Hellman's statements above are not correct for chosen plaintext attacks (but they could still be true for known plaintext attacks). However, an algorithm which is not secure against chosen plaintext attacks cannot be considered to be secure. Hence, CBC and CFB modes of operations cannot be considered to be secure against TMTO attacks. This clears a quarter century old confusion on this issue.

Related Work: In a recent work, Biryukov [6] studies applications of multiple data TMTO. We would like to point out that the situation considered in [6] is different from the one we consider here. More specifically, Biryukov [6] considers the situation where a single message is encrypted with many keys and the corresponding ciphertexts are available to the attacker. The goal of the attacker is to obtain one of these keys. This situation applies to the ECB mode of operation of a block cipher, which is the mode usually considered for cryptanalysis of block ciphers. Detailed discussion on strengths of a block cipher in ECB mode and UNIX password hashing is presented in [6]. We would like to mention that one of the reviewers of this paper pointed out that obtaining one-out-of-many keys was earlier suggested in [12].

In contrast to [6], this work and its earlier version [17] considers the more general problem of identifying suitable oneway functions in cryptographic algorithms and possible access to multiple data. The more interesting cases considered here are streamciphers with IV, various modes of operation of block ciphers such as CBC, CFB, etcetra. We note that none of these cases are considered in [6].

Lastly, we would like to clarify some confusion regarding authorship. The work [6] and [22] has been merged and is due to appear as [5] in the proceedings of SAC'05. Thus, there is an overlap of authors between [5] and the current paper. However, the common author was in no way involved with either the preparation or the original submission of [6] to SAC'05.

2 Review of TMTO Algorithms

Time memory data tradeoff algorithms are applied to invert one-way functions. Let $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way function inverting which will break a cipher. We briefly describe the existing work on methodology of applying TMTO.

A TMTO algorithm has two phases. In the offline phase, a set of tables are prepared. In the online phase, the attacker is given $y_1, \ldots, y_D$ and has to find a pre-image for one of the y_i's, i.e., for some i, the attacker has to find one x_i such that $f(x_i) = y_i$.

We put $N = 2^n$ to be the size of the search space. The pre-computation time is denoted by P and the online search time is denoted by T. The number of data points $y_1, \ldots, y_D$ is D and the memory required to store (the required fraction of) the tables is denoted by M.

The original TMTO algorithm by Hellman [16] used $D = 1$ and satisfied the so-called TMTO curve: $TM^2 = N^2$ with a typical point of $T = M = N^{2/3}$. The pre-computation time is $P = N$.

Babbage [4] and Golić [12] considered TMTO on streamciphers. The tradeoff is basically a birthday attack, and the tradeoff curve is $TM = N$, $T = D$ and $P = M = N/D$. We will call this the BG attack. A typical point on the curve is $T = M = D = P = N^{\frac{1}{2}}$.

Biryukov and Shamir described a multiple data variation of the Hellman method to obtain a new TMTO on streamciphers. The tradeoff curve of $TM^2D^2 = N^2$, $1 \leq D^2 \leq T$, $P = N/D$ was given. We will call this the BS attack. A typical point on the curve is $T = M = N^{\frac{1}{2}}$, $D = N^{\frac{1}{4}}$, $P = N^{\frac{3}{4}}$.

Permutation: If the one-way function f to be inverted is a permutation, then even for $D = 1$, one can obtain the tradeoff curve $TM = N$ with a better tradeoff of $T = M = N^{1/2}$, $D = 1$.

Multiple Data: Availability of multiple data improves the effectiveness of a TMTO attack. In many cases with $D > 1$, the pre-computation time will also be less than N. On the other hand, we need to carefully examine the scenario under which multiple data attack is applied. For example, Hellman originally applied TMTO to find the key of a blockcipher used in the ECB mode of operation. An easy extension to multiple data attack would be for the attacker to target multiple keys and be satisfied with obtaining at least one of these. A similar situation applies to streamciphers as we point out later. A more nontrivial application of multiple data attack is to be able to identify a situation where all the obtained data corresponds to one single key. In this paper, we will mostly be concerned with TMTO attacks which uses multiple data corresponding to a *single* key.

Attack complexity: The complexity of a TMTO attack is usually taken to be the sum or maximum of T, M, and D. It is customary not to take the pre-computation time P as adding to the attack complexity. This is explicitly mentioned in the following quote from Hellman[16],

> "The N operations required to compute the table are not counted because they constitute a pre-computation which can be performed at the cryptanalyst's leisure."

Similarly, Biryukov-Shamir [7], writes that the pre-computation phase "can take a very long time". Following in these steps, it has been customary to ignore

pre-computation time for TMTO attacks. In the case $D = 1$, exhaustive search (or even more) pre-computation time is unavoidable. More generally, the pre-computation time is $P = N/D$ and N is of the form 2^{k+v}, where k is the key length and v is the length of associated data (IV, nonce, tweak, etcetra). If we put $D = N^a$, with $0 \leq a < 1$, then $P = N^{1-a}$ and is less than 2^k if $k > \frac{1-a}{a} \times v$. Since 2^k correspond to exhaustive search time, under the last condition the pre-computation time is less than exhaustive search.

3 Streamcipher

Let us be given a streamcipher algorithm that takes a k-bit key. Our search space is the key space of size $N = 2^k$. Consider the following oneway function f which takes a single k-bit key (and no IV) as input. The cipher algorithm specifies a key load mechanism and an initialization procedure. Take the first k bits of keystream as output for the function f.

Inverting f will provide the key. This approach of applying TMTO to the key space of a streamcipher is not a new idea. Hellman [16] briefly mentions this situation as one possible application. Also, in the appendix of a more recent paper [11], this situation is more definitely mentioned in relation to BG-tradeoff.

Let us consider multiple data when applying TMTO to f. Consider the situation of a dummy terminal session. Assume that each session is encrypted with a new key, and that the first encrypted text of a session is the (fixed) login screen so that the keystream prefix of each session is always exposed. Each session we observe gives one target data point. Inverting any one of the data points, gives us the corresponding secret key. Using the BS curve, if we can observe $D = N^{1/4} = 2^{k/4}$ sessions, then we have an attack with $T = M = N^{1/2} = 2^{k/2}$ and $P = 2^{3k/4}$. Depending on the amount of available data, one could also choose other suitable points of the BS curve. In any case, under this kind of an attack scenario, no streamcipher can provide security level equal to its key length.

3.1 Streamciphers with IV

The situation with streamciphers have changed somewhat since the early work of Hellman, and modern ciphers now use a nonce or an initial vector (IV) in addition to the secret key. Resynchronization is more common in this situation, and obtaining large sets of data is more realistic.

Consider an environment where many short messages are encrypted, each with a different IV. Assume that the master key is seldom changed. This may happen with wireless communication frames, or maybe a disk encryption scheme where each sector is encrypted with a different IV. Assume some of these frames or sectors are known to us in the form of bare keystream. Since IVs are usually public, if we can obtain the master key to one of these frames, all other frames using the same master key would be readable.

We first need to define an appropriate oneway function. Consider the function

$$f : \{\text{master keys}\} \times \{\text{IVs}\} \to \{\text{keystream prefix}\}. \tag{1}$$

Function f sends a random (k-bit key, v-bit IV) pair to a $(k+v)$-bit keystream prefix. So our search space is of size $N = 2^{k+v}$. For a good cipher, this mapping should behave like a random function. We consider three cases with different data requirements. The first of these follows from the BG curve ($TM = N$; $T = D$) , while the other two follow from the BS curve $TM^2D^2 = N^2$.

1. $(P, D, M, T) = (N^{1/2}, N^{1/2}, N^{1/2}, N^{1/2})$: $P = 2^{(k+v)/2} < 2^k$ for $k > v$.
2. $(P, D, M, T) = (N^{2/3}, N^{1/3}, N^{1/3}, N^{2/3})$: $P = 2^{2(k+v)/3} < 2^k$ for $k > 2v$.
3. $(P, D, M, T) = (N^{3/4}, N^{1/4}, N^{1/2}, N^{1/2})$: $P = 2^{3(k+v)/4} < 2^k$ for $k > 3v$.

If we ignore pre-computation time, then data requirement is the minimum in the third case above. In this case, we have an attack whenever $T = M = N^{1/2}$ is less than 2^k. The last condition holds for $k > v$ and hence we can say that *if IV is any shorter than key, the streamcipher is vulnerable to a TMTO attack.*

Pre-Computation Time: If we wish to take pre-computation time into account, then the third case gives an attack for $k > 3v$. If more data is available, then using the first two cases, we get attacks under different relations between k and v. As already mentioned before, if $D = N^a$ for some $0 \leq a < 1$, the pre-computation time is $P = N^{1-a}$ and is less than 2^k for $k > \frac{1-a}{a} \times v$. On the other hand, for a fixed value of k and v, if we wish to make the pre-computation time at least as expensive as exhaustive search, then we must ensure that the access to multiple data is restricted to the condition $a \leq v/(k+v)$. If $a > v/(k+v)$, then we have a TMTO attack where even the pre-computation time is less than exhaustive search.

Below, we state some remarks on this and give some variations to this method.

1. Putting a restriction on how many frames are encrypted before the master key is renewed does not stop this attack completely. The attacker still gets to know one of the many master keys.
2. Making the state initialization process more complex has completely no effect on this TMTO attack. *Neither does the size of the internal state of the stream cipher affect this TMTO in any manner.*
3. The known part of keystream need not be at the very beginning. As long as they are fixed positions in the keystream, they do not even need to be continuous. The oneway function can be defined to match the known part.
4. If IV is XORed into the key before being placed into the internal state, we could set the domain of the oneway function to be at that position. In general, the domain of f should be at the point of least entropy occurring during the initialization process.
5. Using IVs in a predictable manner effectively reduces the IV space, making TMTO more efficient.

3.2 State Versus Key Size

Previous multiple data attacks on streamciphers have targeted the internal state of the cipher. It has been suggested that to resist TMTO attacks, the internal

state size should be at least twice the key size. Our new attack shows that if IV is any shorter than the key, then the streamcipher is vulnerable to TMTO *irrespective* of the size of the internal state. There are two consequences.

First, simply increasing state size of a streamcipher does not make the algorithm TMTO resistant. Second, it does not make sense to increase the state size without a corresponding increase in the size of the IV. For example, if one believes that TMTO forces internal state of any streamcipher to be twice as big as key, as is requested in the ECRYPT Call for Stream Cipher Primitives [3], then one should also request IV size to be at least as big as key size.

Conversely, suppose one is on the other side of this argument, with the opinion that birthday attack based BG-tradeoff should not be taken seriously, and that BS-tradeoff with pre-computation time consideration only mandates IV size bigger than half of key size. Then one should demand state size of only 1.5 times key size.

3.3 ECRYPT Streamcipher Project

Consider a streamcipher taking 80-bit keys with 32-bit IVs. At first, this seems to be a perfectly normal use of key and IV. Actually, this is one of the mandatory parameter set for streamciphers aiming for Profile 2 of the recent ECRYPT Call for Stream Cipher Primitives [3].

Here $N = 2^{112}$ and using the BS curve $TM^2D^2 = N^2$, one sees that this is vulnerable under the tradeoff point $T = M = 2^{56}, \quad D = 2^{28}, \quad P = 2^{84}$. The pre-computation time is slightly more than exhaustive key search. The tradeoff point $T = 2^{74.7}, \quad M = D = 2^{37.3}, \quad P = 2^{74.7}$ is also applicable, and brings the offline complexity to under 80 bits. One weak point of this second approach is that the data must spread over multiple keys and the attacker recovers only one of these keys.

After a preliminary version [17] of our work was made public, members of ECRYPT STVL have posted a note [8], with the following modifications.

- 80-bit key with 32-bit IV can no longer be considered a secure parameter set for streamciphers.
- It makes no sense to increase internal state size of a streamcipher without increasing IV size.

Thus, even though our attack appears to be simple, it turns out to be important enough to bring changes to the ECRYPT call for streamcipher primitives. Actually, we were also *surprised* that such a simple and important observation as ours was actually missed by the entire large and active streamcipher community for so many years.

3.4 GSM

Our discussion so far on streamciphers has shown that security level reached by using a key of length longer than IV length, does not correspond to key length,

under the framework of TMTO attacks. In this section, we turn to a more specific example. It will illustrate that the actual joint entropy of key and IV matters more than just their length.

The encryption algorithm for GSM mobile phones [1] is called A5/3. It is a modified version of OFB mode of operation based on the KASUMI blockcipher. KASUMI is a 64-bit blockcipher with key length of 128 bits.

In the use of A5/3 for GSM encryption, most part of IV is fixed to some constant value. Only a 22-bit counter part is incremented each time the IV is changed. The 128-bit key is actually a concatenation of two copies of a single 64-bit key. Only 228 bits of keystream is used after initialization with a new IV, but this is not important for us.

We can define our oneway function as

$$(\text{64-bit key}, \text{22-bit counter value}) \mapsto \text{86-bit keystream prefix}.$$

There is an initialization process making A5/3 slightly different from the usual OFB mode of operation and the feedback itself is also a bit different, but as was already commented, this is immaterial. It suffices to know the exact specification for keystream production in order to be able to apply TMTO algorithms.

In this case, $N = 2^{86}$. If we choose $D = N^{1/4}$ and $T = M$ in the curve $TM^2D^2 = N^2$, then we get an attack with the parameters $D = 2^{21.5}$ and $T = M = 2^{43}$. The precomputation time is $P = 2^{64.5}$. Since the counter used in the IV is only 22 bits long, it seems more reasonable to collect data that correspond to multiple master keys. In practice, this may have been obtained from multiple users. When one of these keys is recovered, it can be used to decrypt messages encrypted with the same key and different IVs.

The authors are not aware of the actual situation, but if only a small portion of the possible counter values are used in real life (this would happen if the counter always started from zero), i.e., if the entropy of the counter is smaller, the attacker's position is strengthened further.

3.5 Designing TMTO Resistant Streamciphers With IV

The level of threat brought about by a TMTO attack depends largely on the environment. But a good streamcipher design would be aimed at resisting these threat under any plausible environment it could be in. If one views TMTO attacks as threat to streamciphers, one of the following measures should be taken.

1. Ensure that, in every implementation of the cipher, the collective entropy of key and IV will always be at least twice that of intended security level. In particular, the length of key and IV should add up to at least twice security level and the IV should not be used in a predictable way. During the state initialization process, the collective entropy of key and IV should not be allowed to decrease below twice key size.

2. If you are designing a general purpose streamcipher, and do not know in what manner your cipher is going to be used, claim security level corresponding to half your key size. Then, arbitrary use of IV may be allowed. Entropy of internal state after initialization should not be smaller than that provided by key size.

Here, in saying that the IV usage should be random, we mean it to be unpredictable from the viewpoint of a TMTO attacker preparing a table. So, for example, as long as the starting point is chosen at random, the IV may be supplied through a counter for a limited period of time. This possibility was pointed out in [8] in response to an earlier version of this paper.

Pre-Computation Time: As mentioned in Section 3.1, the pre-computation time can be less than exhaustive search if $D = N^a$ with $a > v/(k+v)$. Thus, one approach to securing streamciphers against TMTO with less than exhaustive pre-computation is to ensure that the access to multiple data is restricted to at most $N^{v/(k+v)}$. Any value of k and v satisfying this condition can then be used.

4 Blockcipher Modes of Operation

In this section, we consider several non-trivial applications of multiple data chosen plaintext TMTO attacks to different blockcipher modes of operations. We were able to do this successfully on every mode we have considered. This seems to indicate that, in general, *all blockcipher modes of operation are vulnerable to TMTO attacks.*

4.1 ECB, CTR, OFB

For the ECB mode of operation, TMTO that utilize multiple data may be used if the attacker's objective is to recover any single one of the multiple keys that encrypted the same chosen plaintext.

Counter mode is in a very similar situation if counter usage is predictable. The counter value predicted to be used gives us a basis for the chosen plaintext attack, and when the corresponding ciphertext is given, the key may be recovered in time shorter than key exhaustive search. After this, all other text encrypted with the same key may be decrypted.

As we already saw in the GSM example, OFB mode of operation is essentially a streamcipher with IV, and arguments of the previous section apply.

4.2 CBC Mode of Operation

Consider a blockcipher where message, IV and cipher lengths are b bits. Let the key length be lb bits. (Note that l need not be an integer and we denote $\lambda = \lceil l \rceil$, $\mu = l - \lfloor l \rfloor$.) The encryption function E_k maps a b-bit string to a b-bit string. For a plaintext $m_1, m_2, \ldots$, with each $|m_i| = b$, the CBC encryption with an IV V,

produces a ciphertext $c_1, c_2, \ldots$ as follows: $c_i = E_k(m_i \oplus c_{i-1})$, where we assume $c_0 = V$.

Let m be a fixed b-bit string. For example, if we are dealing with a 64-bit blockcipher, we let m be 8 ASCII space characters. This definition of m also appears in the original work by Hellman [16].

For any b-bit IV V and lb-bit key k, we define a one-way function $f : \{0,1\}^{(l+1)b} \to \{0,1\}^{(l+1)b}$ as $f(k||V) = c_1||c_2||\cdots||c_{\lambda+1}$ where

$$f(k||V) = \underbrace{E_k(m \oplus V)}_{c_1} || \underbrace{E_k(m \oplus c_1)}_{c_2} || \cdots || \underbrace{E_k(m \oplus c_{\lambda-1})}_{c_\lambda} || \mathsf{prefix}_{\mu b}(\underbrace{E_k(m \oplus c_\lambda)}_{c_{\lambda+1}}).$$

(Here $\mathsf{prefix}_i(x)$ denotes the i-bit prefix of the binary string x.) Then the output of f is the $(l+1)b$-bit prefix of the encryption of the plaintext M which consists of $\lambda + 1$ repetitions of the b-bit message m using the key k and IV V.

The f defined above is the target one-way function to be inverted. We incorporate multiple data in the following manner. Let $c_1 c_2 \ldots c_{D+\lambda}$ be a ciphertext obtained by encrypting a plaintext consisting of $D + \lambda$ many repetitions of the b-bit message m using an unknown key k and IV V. For $1 \leq i \leq D$, define $C_i = c_i \ldots c_{\lambda+i}$. Due to the self-similar structure of CBC chaining, we have the following relationships.

1. C_1 is the CBC encryption of M using key k and IV V.
2. C_2 is the CBC encryption of M using key k and IV c_1.
3. C_3 is the CBC encryption of M using key k and IV c_2.
4. In general, C_i is the CBC encryption of M using key k and IV c_{i-1}.

Then by the definition of f, we have $f(k||c_{i-1}) = D_i = \mathsf{prefix}_{(l+1)b} C_i$, for $1 \leq i \leq D$. Inverting f on any of the D_i's will yield k (and also c_{i-1}). If the IV V is not public, we could just ignore the first block and think of the second block as starting a CBC mode with the IV set to the first ciphertext block, decrypting from the second block onwards. Further, the repetitions of m need not be at the begining of the message. If there are $D+\lambda$ repetitions of m occurring somewhere in the message, then we can use the known ciphertext block preceding the $D+\lambda$ repetitions as the IV and obtain the required D data points.

This establishes a multiple data scenario for attacking the CBC mode of operation. Here the search space is $N = 2^{(l+1)b}$ while the key space is 2^{lb}. Assuming the curve $TM^2D^2 = N^2$ holds, an optimal point of $T = M = N^{1/2}$, $D = N^{1/4}$ yields an attack if and only if $N^{1/2} < 2^{lb}$, i.e., $2^{(l+1)b/2} < 2^{lb}$ which holds if and only if $l > 1$. Thus, $b = 128$ and $l = 2$ gives an attack. This situation corresponds to AES with message and cipher length equal to 128 bits and key length equal to 256 bits. Similarly, the parameters $b = 128$ and $l = 1.5$ gives an attack corresponding to AES with 128-bit message block and 192-bit key.

The discussion on pre-computation time is similar to that presented in Section 3.1 and hence is not repeated here.

OMAC OMAC [19] is a NIST standard for encryption and authentication. It is a one key CBC with the capability of producing an authentication tag. Ignoring the MAC, the TMTO attack on CBC also works for OMAC.

4.3 CFB and TBC Modes of Operation

CFB is the other mode of operation which Hellman remarked to be secure against known plaintext TMTO attacks. However, the situation with CFB is exactly the same with CBC, i.e., CFB is equally susceptible to chosen plaintext TMTO attacks.

As before, let E_k be the encryption function of a blockcipher with b-bit message, IV and cipher blocks and lb-bit key blocks. Given a plaintext of b-bit blocks $m_1, m_2, \ldots$, and an IV V, the CFB mode of operation produces a ciphertext $c_1, c_2, \ldots$, where $c_i = m_i \oplus E_k(c_{i-1})$. As before, we assume $c_0 = V$.

The one-way function to be inverted is defined from $(l+1)b$-bit strings to itself in the following manner. As before, let m be a fixed b-bit message string. Then, given a lb-bit key k and a b-bit IV V, we define,

$$f(k||v) = \underbrace{m \oplus E_k(V)}_{c_1} || \underbrace{m \oplus E_k(c_1)}_{c_2} || \cdots || \underbrace{m \oplus E_k(c_{\lambda-1})}_{c_\lambda} || \mathsf{prefix}_{\mu b}(\underbrace{m \oplus E_k(c_\lambda)}_{c_{\lambda+1}}).$$

Now the entire discussion given for CBC applies. Also, the same argument applies to tweakable blockciphers [21] running in TBC mode. It suffices to use tweak in place of IV.

4.4 Other Modes of Operation

We have considered OCB [24], CMC [14], and EME [15] modes of operation. With the attacker given full power with respect to pre-computation and data availability, if key (two keys are used for CMC, but we can treat them as one long key) is any longer than IV, nonce, or tweak, these modes cannot provide security level equal to key size.

4.5 OCB

The mode OCB [24] produces MAC in addition to the ciphertext. Encryption part of OCB is similar to ECB, except that one extra key-like element is used for each block of encryption. These key-like elements are derived from a key and nonce pair, and is updated for each block of additional encryption.

From the view point of TMTO, the MAC output part is no different from the ciphertext. As before, we use the chosen plaintext attack scenario and define the oneway function to send (key, nonce) pair to (ciphertext||MAC).

4.6 CMC, EME

Let us consider the CMC [14] and EME [15] modes of operation. A tweak in addition to a key (two keys are used for CMC, but we just consider them as one long key) is used. These are two-pass encryption modes and every bit of the ciphertext depends on the whole input text. TMTO should provide the attacker

with a key (in addition to the tweak). This can then be used on ciphertexts using different tweaks.

We fix a plaintext and define the oneway function f as follows. The function f takes as input a pair (key, tweak) and encrypts the plaintext to obtain the ciphertext. This is hashed (by a collision resistant hash function) to obtain a string of length equal to |key| + |tweak|. This string is the output of f. In the online phase, we will have a ciphertext and can hash it to obtain a string in the range of the oneway function. Finding a pre-image of the range element will provide the secret key (and also the tweak).

These modes of operations extend a small block length pseudorandom permutation to a wide block length pseudorandom permutation. The intended application is for in-place disk encryption, where the tweak is the sector address and the plaintext block consists of the contents of the corresponding sector. Thus, block length is quite large (around 512 bytes). The reason for using hash function in the definition of the oneway function is so that we do not record this long ciphertext in the table.

It is quite possible that the contents of many of the sectors are identical, which is especially true if the sectors are not in use. In such situation, we can utilize multiple data by obtaining the ciphertexts corresponding to different sector addresses (tweaks) among the sectors containing our fixed chosen plaintext. Inverting any of the points will reveal the master key (and the corresponding tweak), which can be used to decrypt other blocks.

5 Hash Function

With the demand for small hash functions increasing in relation to its possible use in RFIDs, the relatively less interesting results we have concerning hash functions may have implications on hash designed for those environments.

Simple hash We could not find reasonable application of TMTO to collision finding, but obtaining preimage or second preimage quickly with the added advantage of pre-computation time seem to be plausible attack scenarios not considered before. Applying TMTO to the oneway hash function itself, with the message space appropriately restricted, one can see that no hash function can achieve preimage resistance security level equal to its digest size.

Keyed hash and MAC Under the chosen plaintext attack model, keyed hash (or MAC) is very similar to the ECB mode of operation. Sending key to the keyed hash value of a fixed plaintext is the oneway function to be considered. Attacker's objective is to recover the key, given the keyed hash value corresponding to the chosen plaintext. Once the key is obtained, it could be used to forge other hash (or MAC) values. TMTO applies as before and security level equal to key size cannot be reached.

6 Asymmetric Algorithms

In many cases of public key algorithms, the relevant oneway functions satisfy the so-called random self reducibility property, i.e., solving one particular instance of the problem is as hard as solving a random instance of the problem. This is usually shown by converting a specific instance to a random instance. We would like to point out that this provides a natural way of applying multiple data TMTO, even when a single data item is obtained from the application domain. This is also true for the so-called homomorphic encryption algorithms, whereby knowing the encryption of a single message, it is possible to create encryptions of many related messages.

In symmetric key algorithms, usually the security level expected of an algorithm is equal to its key size. This is far from true in the asymmetric world. Hence TMTO algorithms, the best of which only halves the security level, is less interesting here. Nevertheless, to show that TMTO is a versatile tool, we shall apply tradeoff methods to some asymmetric algorithms.

6.1 NTRUEncrypt

Let us consider the 80-bit security version of NTRU public key cryptosystem [2]. Latest parameter set [18] specify a message space of 2^{251} size. Of the 251 bits, only about $\frac{2}{3}$ is used for the actual message and the rest is filled with a randomizing value. (This situation resembles the key+IV situation considered in previous sections and shows that even *probabilistic* algorithms are not completely out of reach from TMTO.) What is important is that, for a fixed public key, once the 251-bit input is formed, the rest of the encryption process is deterministic from that point on. We can take this deterministic encryption process as our oneway function f and apply the tradeoff point $T = M = N^{2/3}$ to obtain a message recovery attack of $2^{167.3}$ complexity.

Actually, we can do better. As mentioned in Section 2, if f is a permutation, then a better tradeoff point $T = M = N^{1/2}$ applies. Notice that encryption is a bijective process (the so called wrapping failure no longer occurs for parameters presented in [18]). Hence, even though there are some complications, arguing that f is a permutation is reasonable. In such a case, attack complexity goes down to $2^{125.5}$.

We have shown that at the cost of exhaustive pre-computed encryption with a fixed public key, one can decrypt any ciphertext with online time and memory complexity $2^{125.5}$. This is larger than, but close to, the best known attack on NTRU of complexity 2^{106}, which happens to be another time memory tradeoff called the meet-in-the-middle attack. To bring multiple data into the picture, one might consider the situation where multiple encrypted messages are given to the attacker and inverting just one is good enough.

Similar arguments as given above apply to all public key encryption schemes. Also, for other public key schemes there can be alternative oneway functions to consider. For example, one may consider the function from the decryption key to the plaintext for a fixed ciphertext. It might not always be valid to consider

this, but in the cases it is valid, applying TMTO to such a function will yield the decryption key. We do not discuss these issues further, since for such applications, TMTO does not appear to be a realistic threat.

6.2 Signature Schemes

Many signature schemes send a triple (m, k, r) consisting of message, key, and randomizing value to a signature (x, s). Here, x is a function of the random value r and sometimes also of m, and s is a function of all inputs.

One fixes a message m likely to be signed by the victim in the near future and apply TMTO to the function $(k, r) \mapsto (x, s)$. Depending on the relative size of k and r, this could be efficient than key exhaustive search. However, the attack complexity will not go anywhere near the claimed security level of the signature schemes. Alternatively, one could apply TMTO to the $(r, m) \mapsto x$ part first (under chosen plaintext scenario), and use the obtained r to recover k, for a more efficient attack. Thus, there are several possibilities for candidate oneway functions, possibly different from the oneway function the designer had in mind.

7 General Framework for TMTO Application

Through arguments of this paper, we have seen that TMTO can be applied to many different situations in a very versatile way. In this section, let us take for granted that TMTO is a general method for inverting well-behaved oneway functions, and explain a general method for applying it to cryptographic situations.

In all of the cryptographic situations considered in this paper, under an appropriate attack scenario, we could devise a oneway function of the following form.

$$f : K \times V \to C. \tag{2}$$

Here, K denotes the secret values the attacker is trying to obtain, and V refers to the set of auxiliary values which is, in many situations, public but not controllable by the attacker. The set C contains the output values and specific targets from this set is given to the attacker at the online stage of TMTO. What these sets refer to in the various situations considered in this paper is listed in Table 1. In some cases, V is missing from the cryptographic system, in which case we think of V as containing a single element.

Once a oneway function is fixed, in most cases, we will want to be able to apply f iteratively. This can be taken care of by applying a random hash

$$h : C \to K \times V. \tag{3}$$

The second thing we should consider is that most of the TMTO algorithm will apply with better success rate if $h \circ f$ is close to an injection so that a target uniquely determines the pre-image. As long as set C is larger than $K \times V$, for most cryptographic applications, this can be naturally expected of the system to some degree. If C is smaller than $K \times V$, one should find some way to deform f so

Table 1. Fitting various cryptographic situations into TMTO framework

situation		K	V	C
block	ECB [16], CTR	key	-	single ciphertext block
	OFB, CBC, CFB	key	IV	ciphertext blocks
	TBC	key	nonce	ciphertext blocks
	OCB	key	nonce	ciphertext blocks + MAC
	CMC, EME	key(s)	tweak	ciphertext blocks
stream	previous [4, 7, 12]	state	-	keystream of state size
	simple	key	-	keystream of key size
	with IV	key	IV	keystream of (key+IV) size
hash	preimage	message	-	hash value
	keyed	key	-	hash value
public key encryption		message	randomizing value	ciphertext
signature		key	randomizing value	signature

that the image space is larger. We saw through chosen plaintext attack scenarios that this could easily be done by simply increasing your plaintext length so that the output is long enough. In other situations, for example, if V contains publicly known values, using a hash $h' : V \to V'$ of appropriate length and setting

$$f' : K \times V \to C' = C \times V' \qquad (k, v) \mapsto f(k, v)||h'(v) \tag{4}$$

could be another solution. One should keep in mind that the image must be some value that is either public or can be calculated from publicly available data.

We can now write up a set of guidelines for applying TMTO to a cryptographic system.

1. Identify a (oneway) function $f : K \times V \to C$, inverting which will reveal a secret information of the attacker's interest, belonging to K. This function need not be the oneway function the designer of the system based his system on.
2. K and V should be taken as small as possible, allowing it to be just big enough to reflect the actual entropy of values used.
3. If needed, adjust the function so that the entropy of function image space is equal to its input space. This will help in making the function f injective, hence raising the success probability of attack.
4. Lower attack complexity can be achieved if it is possible to devise an attack scenario where the attacker is given multiple target points in the image space of f and finding the inverse image of any one of those points is good enough.
5. Depending on the reasonable amount of target points available, apply a suitable TMTO method to obtain a secret value in K.
6. When abundant data is at hand, TMTO with $D = N^{1/4}$; $T = M = N^{1/2}$ is applicable, and the attack is meaningful whenever $|K| > |V|$. At the other extreme, with one data point and oneway function of bad characteristics, we could apply the TMTO of Fiat and Naor [10], and the attack is successful when $|K| > |V|^3$.

We can summarize all this by saying that the most difficult task of applying the TMTO to a cryptographic system is finding a plausible scenario of attack, preferably in which a large set of data is available. Once this is done, the rest of the process comes naturally.

8 Conclusion

TMTO is basically a generic oneway function inverter. To attack a specific system with these TMTO methods, it suffices to identify a suitable oneway function, inverting which will provide one with a secret. In doing this, one should open their eyes to oneway functions hiding in the system, different from the one designer of the system had in mind. Success of TMTO depends heavily on the available amount of data, so devising an appropriate scenario of attack is also crucial.

By applying generic TMTO to blockciphers in ways not tried before, we have confirmed that TMTO has security implications, not only to ECB, but to most blockcipher modes of operation. We have also shown that TMTO affects the security of every streamcipher, not only those with small internal states.

We conclude with the remark that TMTO as a general oneway function inversion technique is more powerful and versatile a tool than is currently known to the crypto community.

References

1. 3GPP TS 55.215 V6.2.0 (2003-09), A5/3 and GEA3 Specifications. Available from `http://www.gsmworld.com`
2. Consortium for efficient embedded security. Efficient embedded security standards (EESS) #1. Version 2.0, June 2003. Available from `http://www.ceesstandards.org/`
3. ECRYPT. Call for stream cipher primitives. Version 1.2, Feb. 2004. `http://www.ecrypt.eu.org/stream/`
4. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp. 161–166, IEE, 1995.
5. A. Biryukov, S. Mukhopadhyay and P. Sarkar, Improved time-memory trade-offs with multiple data, *Proceedings of Selected Areas in Cryptography*, 2005, to appear.
6. A. Biryukov, Some thoughts on time-memory-data tradeoffs. Cryptology ePrint Archive, Report 2005/207, `http://eprint.iacr.org/2005/207`, 30 June, 2005.
7. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp. 1–13, Springer-Verlag, 2000.
8. C. De Cannière, J. Lano, and B. Preneel, Comment on the rediscovery of time memory data tradeoffs. Available as a link on the *ECRYPT Call for Stream Cipher Primitives* [3] page version 1.3, April 2005.
9. Denning, *Cryptography and data security,* Addison-Wesley, 1982.
10. A. Fiat and M. Naor, Rigorous time/space tradeoffs for invering functions. *SIAM J. on Computing,* vol 29, no 3, pp. 790–803, SIAM, 1999.
11. S. Fluhrer, I. Mantin, and A. Shamir, Weakness in the key scheduling algorithm of RC4. *SAC 2001,* LNCS 2259, pp. 1–24, Springer-Verlag, 2001.

12. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt'97,* LNCS 1233, pp. 239–255, Springer-Verlag, 1997.
13. Z. Gutterman and D. Malkhi, Hold your sessions: An attack on Java session-id generation. *CT-RSA 2005,* LNCS 3376, pp. 44–57, Springer-Verlag, 2005.
14. S. Halevi and P. Rogaway, A tweakable enciphering mode. *Crypto 2003,* LNCS 2729, pp. 482–499, Springer-Verlag, 2003.
15. S. Halevi and P. Rogaway, A parallelizable enciphering mode. *CT-RSA 2004,* LNCS 2964, pp. 292–304, Springer-Verlag, 2004.
16. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory,* **26** (1980), pp. 401–406.
17. J. Hong and P. Sarkar, Rediscovery of time memory tradeoffs. Cryptology ePrint Archive, Report 2005/090, `http://eprint.iacr.org/2005/090`, 22 March, 2005.
18. N. Howgrave-Graham, J. H. Silverman, and W. Whyte, Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3. *CT-RSA 2005,* LNCS 3376, pp. 118–135, Springer-Verlag, 2005.
19. T. Iwata and K. Kurosawa, OMAC: One-Key CBC MAC. *Fast Software Encryption, FSE 2003,* LNCS 2887, pp. 129–153. Springer-Verlag.
20. I.-J. Kim and T. Matsumoto, Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Trans. Fundamentals,* **E82-A**, pp. 123–129, 1999.
21. M. Liskov, R. L. Rivest, and D. Wagner, Tweakable block ciphers. *Crypto 2002,* LNCS 2442, pp. 31–46, Springer-Verlag, 2002.
22. S. Mukhopadhyay and P. Sarkar, TMTO with multiple data: Analysis and new single table trade-offs. Cryptology ePrint Archive, Report 2005/214, `http://eprint.iacr.org/2005/214`, 4 July, 2005.
23. P. Oechslin. Making a fast cryptanalytic time-memory trade-off. *Crypto 2003,* LNCS 2729, pp. 617–630, 2003.
24. P. Rogaway, M. Bellare, J. Black, and T. Krovetz, OCB: A block-ciper mode of operation for efficient authenticated ecryption. *8th ACM CCS,* ACM Press, pp. 196–205, 2001.
25. A. Shamir, Stream ciphers: Dead or alive? Presentation slides for invited talk given at *Asiacrypt 2004*. Available from `http://www.iris.re.kr/ac04/`

A Known Versus Chosen Plaintext TMTO Attacks on Blockciphers

The issue of known and chosen plaintext attacks was briefly mentioned in Section 1.2. We continue the discussion here.

Hellman's attack on the ECB mode of operation uses a oneway function f defined as follows. Fix a message block m and define a map from key to ciphertext by $f(k) = E_k(m)$. Suppose, we are given a ciphertext c which is the encryption of m under an unknown key k, i.e., $f(k) = c$. If we can invert f on c, then we can hope to find k. Clearly, for this attack to work, we must have an encryption of m and hence the attack is actually a chosen plaintext attack (CPA).

Hellman explains that this can also be turned into a known plaintext attack (KPA) or ciphertext only attack (COA) in the following sense. Suppose m is a block which occurs very frequently, for example a string of blanks. For a *KPA*, the cryptanalyst looks for the occurence of m in the plaintext and inverts f on

the corresponding ciphertext block to obtain k. For a *COA*, the cryptanalyst will look for repetitions among the encrypted message. For each frequently repeated ciphertext block, he will try to invert f. If the block encrypts m, then he finds k, else he fails. The time required for the COA increases, since many trials might have to be done before actually finding an encryption of m. Note that for successful conversion of CPA to KPA and to COA, the block m *must* occur in the (unknown) plaintext corresponding to target data. Thus, if the target data is given randomly, then the above conversions are not meaningful. Furthermore, in Hellman's CPA converted to KPA or COA, there is no way to utilise multiple data to bring down the pre-computation time.

Our attacks on the CBC and CFB modes of operations in Section 4 are *CPA*. As in Hellman, we need to fix a plaintext and then define the oneway function to be inverted. To utilise multiple data, our fixed plaintext consists of $D + \lambda$ repetitions of m. Again, as in Hellman, we need to choose m and D such that $D + \lambda$ repetitions of m is likely in an actual message. Then we can convert the CPA to KPA by inspecting the obtained plaintext for $D + \lambda$ repetitions of m. We consider the corresponding portion of $D + \lambda$ ciphertext blocks. Using the ciphertext block preceding this portion as the IV, we can use the $D + \lambda$ ciphertext blocks to obtain D data points required for the attack. Again, as in Hellman's case, this conversion is not meaningful if the data corresponding to random plaintext is given.

It might appear that for a meaningful KPA, we need a larger portion to be frequently repeated than is required by Hellman. Though this is true, the actual requirement might not be too high. For example, if $\lambda+1$ repetitions of m occur in the plaintext, we can launch an attack with $D = 1$. Having more blocks increases D and the efficiency of the attack.

Conversion of the CPA on CBC and CFB to COA is also possible, though it becomes less efficient. Suppose that we want to utilise D data points and in the pre-computation phase have prepared the tables to cover N/D data points. In the online stage, we do the following. We slide (one block at a time) a window of $D + \lambda$ blocks over the ciphertext. Each window gives us D data points and if we perform the online search of the TMTO, with a constant probability of success we will get a hit. However, the k obtained may not be the correct key since there is no guarantee that the $D + \lambda$ blocks correspond to an encryption of $D + \lambda$ repetitions of m. We can easily verify this by decrypting a portion or whole of the ciphertext using this k. On the other hand, if the window of $D + \lambda$ ciphertext blocks actually correspond to an encryption of $D + \lambda$ repetitions of m, then we have the correct key. Hence, if the unknown plaintext indeed contained $D + \lambda$ repetitions of m, then by trying out all possible windows we are assured of success. This pushes up the online time by a factor which in the worst case is equal to the number of blocks in the obtained ciphertext. This makes the attack less efficient, though it still remains meaningful under our assumption on the data.

B When should we start building a table?

We consider the question of whether it makes sense to start the long-term pre-computation search today.

Moore's law It has been observed that processor power doubles every 1.5 years. Let us assume that this will be true for the foreseeable future. Going back to high school mathematics, we can write the processing power $p(t)$ at time t as

$$p(t) = \alpha \cdot 2^{\frac{2}{3}t}. \tag{5}$$

We will take $t = 0$ to correspond to today, in which case, constant α will be our current computational power.

Example table creation Let us consider Hellmans's TMTO on AES as an example. The pre-computation stage will be an exhaustive processing of all 128-bit keys. On a desktop PC, AES encryption runs at 488 Mbps, which translates to about 2^{47}-many 128-bit blocks per year. We should consider the keyschedule also. Assuming that it runs at about the same speed as the encryption, we can take

$$\alpha = 2^{46} \quad \text{“key} \mapsto \text{ciphertext” mappings/year.} \tag{6}$$

So how long would the table creating take? Solving for T in

$$\int_0^T 2^{46+\frac{2}{3}t}\, dt = 2^{128}, \tag{7}$$

we find that the table creation will end $T = 121.3$ years from now. This assumes that the computer is constantly upgraded.

Starting later What happens if we do nothing for 120 years, and only then start building the table? Our computation power will be $\alpha = 2^{46} \cdot 2^{\frac{2}{3}120} = 2^{126}$. Solving for T' in

$$\int_0^{T'} 2^{126+\frac{2}{3}t}\, dt = 2^{128}, \tag{8}$$

we find that the table creation will take $T' = 2.3$ years, hence ending 122.3 years from now. So we are late by one year than what was achievable. But, is finishing one year earlier really worth the trouble of upgrading the computer constantly for 120 years?

In general, given any computation that takes n years from now to complete, if one starts the computation n years later, it can be finished in less than 1.5 years from then on.