Distributed Time-Memory Tradeoff Attacks on Ciphers
(with Application to Stream Ciphers and Counter Mode)

Howard M. Heys(✉)

Memorial University of Newfoundland, St. John's, Canada
hheys@mun.ca

Abstract. In this paper, we consider the implications of parallelizing time-memory tradeoff attacks using a large number of distributed processors. It is shown that Hellman's original tradeoff method and the Biryukov-Shamir attack on stream ciphers, which incorporates data into the tradeoff, can be effectively distributed to reduce both time and memory, while other approaches are less advantaged in a distributed approach. Distributed tradeoff attacks are specifically discussed as applied to stream ciphers and the counter mode operation of block ciphers, where their feasibility is considered in relation to distributed exhaustive key search. In particular, for counter mode with an unpredictable initial count, we show that distributed tradeoff attacks are applicable, but can be made infeasible if the entropy of the initial count is at least as large as the key. In general, the analyses of this paper illustrate the effectiveness of a distributed tradeoff approach and show that, when enough processors are involved in the attack, it is possible some systems, such as lightweight cipher implementations, may be susceptible to attack in practice.

Keywords: Cryptanalysis · Time-memory tradeoff attacks
Block ciphers · Stream ciphers · Counter mode

1 Introduction

Time-memory tradeoff (TMTO) attacks were first introduced by Hellman [1] to attack block ciphers using a chosen plaintext or easily predicted known plaintext. The basic concept involves two phases: Before system operation begins, the *preprocessing* (or offline) phase prepares a compact table from chains representing information from (almost) all keys, while the *online* phase efficiently searches the table in an attempt to identify which key is used to encrypt during system operation. Following Hellman's work, Babbage [2] and Golić [3] independently showed that a time-memory-data tradeoff based on the birthday paradox was applicable to stream ciphers by attacking the stream cipher state, rather than the key. This was subsequently combined with Hellman's approach by Biryukov and Shamir [4] to develop another, more flexible, tradeoff involving data and

W. Susilo and G. Yang (Eds.): ACISP 2018, LNCS 10946, pp. 135–153, 2018.
https://doi.org/10.1007/978-3-319-93638-3_9

targeting the stream cipher state. This approach was then extended by Hong and Sarkar [5] to attack directly the key and initialization vector (IV) of stream ciphers, as well as being applied to some block cipher modes.

Numerous papers have refined Hellman's approach trying various methods to improve the success rate and reduce the attack complexity. Most notably, the distinguished points method, attributed to Rivest in [6], can be used to minimize costly memory accesses, while the rainbow table method can be used to minimize memory accesses and improve the speed of the table search [7].

Although the concept of distributed cryptanalytic attacks is well known, no paper has systematically characterized the value of distributed time-memory tradeoff attacks. In this paper, we examine tradeoff expressions for a number of distributed TMTO approaches using the number of processors as a tradeoff parameter. Further, we explicitly examine the applicability of distributed TMTO attacks to stream ciphers and the counter mode operation of block ciphers.

2 Background on Time-Memory Tradeoff Attacks

In our discussion, complexities are given for time, memory, and data and the units of these complexities may differ by a modest multiplicative constant when comparing approaches. Time and memory complexities are often represented in units equivalent to the number of encryption operations and the number of key pairs stored, respectively, while data complexities are sometimes expressed as the number of contiguous bits of data or the number of data blocks, with each block corresponding to a unique IV. Also, as is usually done, we assume that when an algorithm complexity involves a factor that is logarithmic in a parameter, this factor is small enough to be ignored.

2.1 Hellman's Attack

The basic TMTO attack on block ciphers introduced by Hellman [1] works because memory is saved by storing in a table just the start and end of chains generated during the preprocessing phase, such that, in the online phase, the table can be efficiently searched while walking through a chain starting with the data captured from the system. As a result, the preprocessing phase requires a time complexity that is equivalent to the size of the key space, while the online time complexity and the memory complexity can be substantially less than the size of the key space.

The preprocessing phase of Hellman's approach involves constructing a table consisting of t subtables, each subtable consisting of m chains of keys of length t. Each chain is constructed by using a chaining function to map a cipher output to the next key input, using a fixed plaintext as input to the cipher in each step. Each subtable uses a different chaining function and picks m arbitrary keys as starting points for the chains. Only the first and last keys in a chain need to be stored, with the key pairs in a subtable sorted according to the last key, for easy search during the online phase of the attack. The table should cover most of the

key space, thus requiring a so-called stopping criterion of $mt^2 = K$, where K is the size of the key space. Because only the start and end of each chain is stored, the table requires a memory complexity of $M = mt$.

During the online phase, a subtable is searched by producing a chain of length t, starting from the intercepted ciphertext (produced by the plaintext used to the build the table). At each step in the chain, if the key is found to be one of the stored last keys of a chain in the subtable, then the cipher key can be determined by proceeding from the starting key of the chain until the ciphertext is generated. The corresponding key is very likely to be the correct cipher key. A chain is built for each of t subtables and, hence, the online time complexity is given by $T = t^2$.

Subsequently, it can be derived that the following tradeoff exists:

$$TM^2 = K^2. \tag{1}$$

The preprocessing time, P, is determined by the time to construct the table given by mt^2, and, hence, due to the stopping criterion relationship, $P = K$. Hellman uses the example that, if $T = M$, then both online time and memory are smaller than the key space and, in fact, $T = M = K^{2/3}$.

2.2 Babbage-Golić (BG) Tradeoff

Both Babbage [2] and Golić [3] independently proposed a tradeoff attack on stream ciphers, referred to as the BG attack. Assume that the size of the stream cipher's state space is N. A keystream prefix is a $\log_2 N$ sequence of keystream bits corresponding to the state at which the prefix starts. The BG tradeoff works by constructing, during preprocessing, a table of N/D pairs of the state and the corresponding keystream prefix. A total of $D + \log_2 N - 1 \approx D$ bits of keystream are acquired in the online phase resulting in the determination of D keystream prefixes, using a sliding window. Due to the birthday paradox, with high probability, one of the D keystream prefixes can be found in the table and the corresponding state derived, thus breaking the cipher.

For this attack, the tradeoff expression, involving online time complexity T and memory complexity M, is

$$TM = N \tag{2}$$

where $T = D$, $M = N/D$, and the preprocessing time complexity is $P = N/D$. Due to this attack, it is prudent to ensure that the state of the stream cipher (in bits) should be at least twice as large as the key (in bits) (i.e., $N \geq K^2$) to ensure that $T \geq K$ or $M \geq K$.

Note that a recent direction of research in the design of stream ciphers is to develop structures to provide security using a state with a size that is less than double the key size. The objective of such research is to minimize the hardware complexity of the ciphers. Designs to do this have been proposed by having the state update be a function of key [8,9] or by using a specific initialization approach and applying packet mode where the amount of keystream generated under one IV is constrained [10]. We do not address these designs in our discussion.

2.3 Biryukov-Shamir (BS) Tradeoff

In [4], Biryukov and Shamir combined Hellman's table and the BG tradeoff use of data to develop a new tradeoff involving time, memory, and data, applicable to stream ciphers. In the BS tradeoff, the Hellman table is derived from chains on the cipher state, rather than the key. During preprocessing, a total of t/D subtables are constructed, with each covering m chains of length t, for which only the first and last states are stored. Variable D represents the amount of data in the form of contiguous keystream bits used in the attack and now the memory complexity is $M = mt/D$. The preprocessing complexity is thus $P = N/D$, where $mt^2 = N$ is the stopping criterion for constructing the table.

During the online phase, t steps through the chain must be executed, with each of the t/D subtables being searched and this must be done for each of the D prefixes derived from a sliding window over the D bits. Hence, the online time takes $T = t(t/D)D = t^2$. As a result, the tradeoff in this case becomes

$$TM^2D^2 = N^2. \tag{3}$$

It should be noted that to ensure there is at least one complete subtable, it is assumed that $D \leq t$ and therefore the restriction of $D^2 \leq T$ exists. Letting $N \geq K^2$ results in $T \geq K$ or $M \geq K$, thereby ensuring that a BS TMTO attack cannot do better than exhaustive key search.

2.4 Hong-Sarkar (HS) Tradeoff

In [5], Hong and Sarkar explicitly relate the BS tradeoff for stream ciphers to the key and the IV, rather than the state. The key is secret and unknown when building the table during preprocessing and, while the IV is typically public and known during the online phase, it may be unpredictable and therefore also unknown when building the table during preprocessing. The HS tradeoff approach treats the input to be discovered in the tradeoff attack to be the key/IV combination. If the size of the IV space is defined to be V and the IVs to be used by the system are unknown during preprocessing, then the HS approach can be applied to a stream cipher with the tradeoff being

$$TM^2D_{iv}^2 = (KV)^2 \tag{4}$$

where the preprocessing complexity is given by $P = KV/D_{iv}$. The attack has a similar data restriction of $D_{iv}^2 \leq T$ as the BS approach. Note that the D term used in the BS tradeoff of (3) has been replaced by D_{iv} in (4) to emphasize that, rather than D contiguous bits, in fact, D_{iv} represents the number of $\log_2(KV)$ bit prefixes at the start of the keystream for different key/IV combinations.

In theory, each prefix used in the attack must be collected from different key/IV combinations and, hence, success in the attack may mean finding one key from among a number of keys used in encryption. In the single-key scenario, where it is assumed that data is only available from one key, if unpredictable IVs are to be used, then data could be collected from different IVs and the target key. Then the tradeoff of (4) can be applied, where D_{iv} represents the number of IVs under the one key and, hence, $D_{iv} \leq V$.

2.5 Dunkelman-Keller (DK) Approach

The HS tradeoff approach assumes that preprocessing is structured to consider the combination of key and IV as one input and builds the table based on this, resulting in the restriction on data. However, the HS method of attack does not take advantage of the fact that, during the online phase, the IV is known and only the key needs to be discovered. In [11], Dunkelman and Keller modify the HS approach by separating the key and IV in the attack. The preprocessing phase then builds a number of Hellman tables to cover keys, with each table built for a particular IV. This allows the online phase of the attack to simply consider whether an intercepted IV has been used to build a table. If so, the table corresponding to this IV can be searched for the key. In this approach, which we refer to as the DK approach, assuming equally likely occurences of any IV, if V/D_{iv} tables, each corresponding to a different IV, are built during preprocessing, then collected data from D_{iv} IVs during the online phase should result in one of the intercepted IVs being used in the tables with high probability. For this tradeoff, $M = (V/D_{iv})mt$ and $T = t^2$, where the stopping criterion of $mt^2 = K^2$ applies to the Hellman tables. Hence, the DK method has the tradeoff expression of (4) if the IV is unpredictable, but now has no restriction on the data, D_{iv}, other than $D_{iv} \leq V$ in the single-key scenario. Further, this approach has an advantage for applications where the IV is unpredictable but not equally likely in distribution, as this knowledge can be used to build tables for the most likely IVs.

2.6 Other Work on TMTO Attacks

We shall consider in our work both the distinguished points and rainbow table refinements of Hellman's TMTO attack. These refinements and their relative merits in terms of probability of success, detailed complexity analyses, and other practical performance related issues, are studied in a number of papers including [12–14]. The results of these comparisons indicate that these practical performance issues do not seem to have substantial implications (i.e., orders of magnitude effects on complexity) and, hence, we do not consider them significant for our discussion on distributed TMTO attacks.

It is known that it is possible to parallelize TMTO attacks. For example, distributed attacks are mentioned in [15] where it is noted that it is possible to divide the Hellman subtables into groupings and circulate to participating processors. Parallelizing TMTO attacks is further studied in [16,17]. However, no work has yet systematically characterized the tradeoff aspects of multiple processors. In our work, we will thoroughly characterize the distributed approach to various forms of time-memory tradeoffs.

3 Distributed Hellman Attack

We now consider the parallelization of Hellman's attack using distributed processors, as well as the related approaches of distinguished points and the rainbow

table. We assume that W processors, with independent memory, are available. This might represent, for example, W computers on the Internet with users willing to participate, or being duped into participating, in attacking some cryptographic system. We assume that any necessary communication complexity between these processors and a central controlling processor are negligible in comparison to the time and memory complexities associated with the attack.

In our discussion, we let T_0, M_0, and P_0 represent the online time complexity, memory complexity, and preprocessing time complexity, respectively, for an individual processor. It is these quantities, along with W, which determine the efficacy of the attack, since it is assumed that the individual processors can operate concurrently. For example, while a non-distributed attack might require an online time complexity of T, if it is possible to spread this work evenly between W processors, each processor would only require a time of $T_0 = T/W$, which could be done concurrently for all processors, and thus the overall duration of the attack could be dramatically reduced if W is large. As a point of comparison for distributed tradeoff attacks, we consider distributed exhaustive key search, which is expected to have a time complexity for an individual processor of $T_0 = K/W$ (with, of course, no preprocessing phase and negligible memory complexity).

3.1 Distributed Approach to the Original Hellman Attack

A distributed approach to Hellman's TMTO attack can proceed by distributing the responsibility for generating the t subtables to the W processors, so that each processor generates t/W subtables independently. When the necessary ciphertext data is captured during system operation, it will be distributed to all processors. Each processor will require a memory of M_0, where $M_0 = m(t/W) = M/W$ and M is the total memory requirement for the attack, with $W \leq t$ in order to ensure that each processor generates one or more subtables.

Since each processor only needs to implement t encryptions for each of t/W subtables, the time taken in a processor (and, if all processors operate concurrently, the overall time to search the full Hellman table) is $T_0 = t(t/W) = T/W$, where T is the time required for the non-distributed attack. When a key is found by a processor in its share of the table, it must communicate this back to the central processor that is overseeing the cryptanalytic process and that will be able to announce the successful completion of the attack.

Now $T_0 M_0^2 = (t^2/W)(mt/W)^2 = (mt^2)^2/W^3$ and assuming the Hellman stopping criterion of $mt^2 = K$ results in the tradeoff for an individual processor to be

$$T_0 M_0^2 W^3 = K^2 \tag{5}$$

where the constraint $W \leq t$, or equivalently $W \leq T_0$, applies. This expression captures the tradeoff of interest in a distributed Hellman attack and reflects that both time and memory can be improved by a factor of W. The preprocessing time for an individual processor is $P_0 = K/W$ and is improved by a factor of W over the time required in the non-distributed attack, since each processor only needs to construct chains covering a fraction of the table. Although we notate

this as the preprocessing cost of the individual processor, if we assume that all processors compute their tables concurrently, it also reflects the overall time complexity to prepare for the attack.

It is clear that using a number of processors to implement the attack potentially provides a very significant advantage and may actually make the attack possible in some practical scenarios. Although exhaustive key search can also be improved by a distributed approach, a distributed TMTO attack preserves the possibility for a significantly faster online processing time at the expense of more memory. Consider the following example applying to an implementation of AES-128 for which $K = 2^{128}$. Letting $W = 2^{20}$, the non-distributed exhaustive key search would require $T = 2^{128}$, while the distributed exhaustive key search would require $T_0 = 2^{108}$. In the case of a Hellman TMTO attack with equal online time and memory complexity, the non-distributed attack would take $T = M = 2^{85.3}$ (with $P = 2^{128}$), while the distributed approach would require $T_0 = M_0 = 2^{65.3}$ (with $P_0 = 2^{108}$). As another example, consider a lightweight block cipher with an 80-bit key so that $K = 2^{80}$. In this case, with $W = 2^{20}$, a distributed TMTO attack exists with $T_0 = M_0 = 2^{33.3}$ (and $P_0 = 2^{60}$), which is substantially less complex than the $T_0 = 2^{60}$ required for a distributed exhaustive key search.

3.2 Distributed Distinguished Points (DP) Method

One of the issues identified for the Hellman TMTO attack is that the cost of a memory access can vary by orders of magnitude depending on whether the access is to internal memory (RAM) or to an external memory (e.g. hard disk drive or a solid state drive) [18]. In order to mitigate the cost of slow memory accesses, the distinguished points (DP) method was proposed by Rivest [6]. In this approach, rather than build chains of fixed length t when constructing a Hellman table, the preprocessing phase can build a chain which terminates when a particular pattern (e.g. all zeroes) is recognized in the first $\log_2 t$ bits of the key. This means the length of a chain is variable but will be a length of t on average. When executing the online portion of the attack, since the end point of a chain must start with $\log_2 t$ zeroes, only about $1/t$ encryptions needs a look up to be executed in the subtable (which is likely stored in slow access external memory).

In the distributed Hellman attack, it is fully possible to execute the distinguished points approach to the attack. The amount of memory in a processor is still fixed at $M_0 = mt/W$, since there are t subtables split between the W processors. However, the time required to finish the concurrent computations of W processors is now more complex. Since there is an average of t steps in each chain, the number of encryptions per subtable must be more than t to cope with chains having more than t steps. Assume that, at most, γt encryptions are executed for each subtable. The DP method is likely to set γ to be a modest value, to keep the time complexity of the attack constrained. When preparing the table during the preprocessing phase, the DP method will stop a chain when a distinguished point is found or when γt steps in a chain have been reached without hitting a distinguished point. Similarly, during the online process, if,

after γt encryptions, a distinguished point is not reached for a subtable, the subtable is assumed to not contain the key. Of course, the value used for γ affects the probability of success, but as shown in [13], γ can effectively be a small constant. Hence, the online time complexity can be no worse than the maximum chain length, γt, multiplied by the number of subtables to search through, t/W, and, hence, $T_0 = \gamma t^2/W$ where T_0 now represents the maximum possible time taken at an individual processor.

This leads to a tradeoff of the form $T_0 M_0^2 W^3 = \gamma K^2$ which is slightly worse than the distributed Hellman tradeoff of (5). However, it is quite possible that implementing the distinguished points method when using a distributed approach will not be necessary. Since the memory size needed in the individual processors in a distributed attack is reduced by a factor of W, it is quite conceivable for some parameters that the processor memory complexity of M_0 is small enough that the processor's complete table portion could be stored in internal memory and slow accesses to external memory are not needed. In such a case, there would be no need to implement the DP approach.

3.3 Distributed Rainbow Table Method

In [7], Oechslin proposed an alternate formulation to represent the key chains in the TMTO attack. Hellman's approach was to use one chaining function for every step of a chain and for all the chains in one subtable, with different subtables then using different chaining functions. In contrast, the rainbow table approach uses a different chaining function for each step of the chain and then builds one table of such chains. It is argued that there are improvements to Hellman's approach [7,19]. For the online phase, t partial chains of length $\leq t$ are produced, starting with the intercepted ciphertext, requiring $t^2/2$ encryptions in total. Ignoring the somewhat insignificant factor of $1/2$ in the number of encryptions gives $T \approx t^2$ and results in the same tradeoff expression as in (1). However, since only at the end of one of the partial chains is it necessary to look up in the table, only t memory accesses to the table are required.

The distributed rainbow table approach can be accomplished by distributing the table so that $M_0 = mt/W = M/W$. However, for each processor, the time complexity involves reproducing t partial chains for a total of $T_0 = t^2/2 \approx t^2$ encryptions required in each processor. Hence, the time complexity cannot be improved by distributing the table since each processor must take $\sim t^2$ to consider their portion of the table, i.e., $T_0 = T$. The resulting tradeoff expression is

$$T_0 M_0^2 W^2 = K^2. \tag{6}$$

Rather than divide up the rainbow table between processors, an alternative approach for a distributed rainbow table attack would be to distribute the computation of t partial chains between W processors. In this case, $T_0 \approx t(t/W)$ would represent the online time complexity (again ignoring the factor of $1/2$). However, the resulting distributed computations would need to be checked in

one central table. In this case, $T_0 = T/W$, but $M_0 = M = mt$. Hence, the tradeoff becomes even worse as

$$T_0 M_0^2 W = K^2. \tag{7}$$

For the rainbow table approach, distributing the table and the computations is not feasible, since the end of each partial chain must be looked up in the full table. Hence, the distributed rainbow table approach is inferior to the distributed version of the original Hellman TMTO approach. In addition, when applying a distributed approach to time-memory tradeoffs, since the memory requirements could be substantially smaller on a per processor basis, reducing memory accesses (one of the advantages of the rainbow table) may not be important, since the necessary subtables of the Hellman approach may fit within a processor's RAM.

4 Applying Distributed TMTO Attacks on Stream Ciphers

In this section, we consider the application of distributed TMTO attacks to stream ciphers.

4.1 Distributed BG Attack

We first consider the distributed BG attack, which makes use of data collected and assumes D bits of keystream are available. In this case, the attack can be distributed by dividing up the work to prepare, and the memory to store, the BG table to W processors, so that $P_0 = N/(DW)$ and $M_0 = N/(DW)$. The time required in a processor during the online phase is directly proportional to the processing of all D prefixes, so that $T_0 = D$, which is unchanged from the non-distributed case. As a result, it can be shown that

$$T_0 M_0 W = N. \tag{8}$$

For a non-distributed attack, letting $N \geq K^2$ ensures that the BG tradeoff does not lead to a better attack than exhaustive key search. Placing this constraint on the stream cipher leads to the following proposition for the distributed BG attack.

Proposition 1
If $N \geq K^2$, there is no value of W for which a distributed BG TMTO attack on a stream cipher has a lower complexity for both online time and memory than the complexity of distributed exhaustive key search.

Proof
A distributed exhaustive key search has a complexity of K/W. Let $N = aK^2$, where $a \geq 1$. We can now adjust (8) to be $T_0 M_0 W = aK^2$. For the best TMTO

attack, we can minimize the maximum of either T_0 or M_0 in this equation by letting $T_0 = M_0$, leading to

$$T_0 = \frac{a^{1/2}K}{W^{1/2}} \tag{9}$$

which clearly implies $T_0 \geq K/W$ and $M_0 \geq K/W$ for all values of W. Since other tradeoffs lead to one of T_0 or M_0 being larger, there will always be at least one of T_0 or M_0 being at least as large as K/W. Hence, clearly the distributed BG tradeoff cannot have a lower complexity than distributed exhaustive key search for any number of processors. □

4.2 Distributed BS Attack

Consider now the distributed BS attack. With W processors and D contiguous data bits of keystream, the t/D subtables needed in the BS approach can be divided into W groups, resulting in the memory for individual processors being $M_0 = mt/(DW)$, where $W \leq t/D$ in order for each processor to have one or more subtables. The time in an individual processor to process the data and recover the state is given by $T_0 = t{\cdot}(t/(DW)){\cdot}D = t^2/W$, where the first term represents the t encryptions to reproduce a chain from the starting point of the captured data, the middle bracketed term represents the number of subtables to process in each processor, and the last term represents the data that each processor must consider. Combining the expressions for M_0 and T_0 leads to the following tradeoff:

$$T_0 M_0^2 D^2 W^3 = N^2 \tag{10}$$

where the amount of data and the number of processors must satisfy $D^2W \leq T_0$ (which is derived by combining the constraint on W with the expression for T_0). Since deriving the required subtables determines the preprocessing time in an individual processor, we also have $P_0 = N/(DW)$.

In the following proposition, we show that the constraint of $N \geq K^2$ ensures that the distributed BS attack performs no better than distributed exhaustive key search.

Proposition 2
If $N \geq K^2$, there is no value of W for which a distributed BS TMTO attack on a stream cipher, satisfying the constraint $D^2W \leq T_0$, has a lower complexity for both online time and memory than the complexity of distributed exhaustive key search.

Proof
Let $N = aK^2$, where $a \geq 1$. Minimizing T_0 and M_0 in the application of the BS tradeoff is done by maximizing the data in the tradeoff. Using the upper bound of $D \leq (T_0/W)^{1/2}$, it can be shown that (10) is equivalent to the tradeoff of $T_0 M_0 W = aK^2$. This is now identical in form to the distributed BG tradeoff of (8) and, hence, the remainder of the proof can follow similarly to the proof of Proposition 1. □

4.3 Distributed HS and DK Attacks

Targeting a stream cipher system which uses a single key and numerous IVs and applying a distributed HS approach results in the tradeoff

$$T_0 M_0^2 D_{iv}^2 W^3 = (KV)^2, \tag{11}$$

where D_{iv} represents the number of prefixes that are derived from the first $\log_2(KV)$ bits of the initial cipher state following the reinitialization from different IVs. The constraints $D_{iv}^2 W \leq T_0$ and $D_{iv} \leq V$ apply and the preprocessing complexity is $P_0 = (KV)/(D_{iv}W)$.

The distributed DK approach, which builds V/D_{iv} Hellman tables for different IVs results in the same tradeoff as (11), as well as the same constraint of $D_{iv} \leq V$ and the same preprocessing complexity of $P_0 = (KV)/(D_{iv}W)$. However, since the DK approach builds a Hellman table to cover just keys (rather than key/IV combinations), we can assume that each processor contains t/W of the Hellman subtables for all of the V/D_{iv} IVs. In this case, $M_0 = (V/D_{iv})m(t/W)$ and $T_0 = t(t/W)$, resulting in (11) with the contraint that $W \leq t$, or equivalently $W \leq T_0$, since at least one full subtable per IV must be stored in a processor.

Note that the HS and DK approaches of (11) require a total number of bits of data to be about $D_{total} = D_{iv}\mu_{iv}$, where μ_{iv} represents the average number of bits encrypted under one IV (although only the first $\log_2(KV)$ bits of each IV's keystream are used in the attack). Hence, substituting into (11) results in

$$TM^2 D_{total}^2 W^3 = (KV\mu_{iv})^2 \tag{12}$$

where D_{total} is the number of bits collected (although many are discarded) and, while it represents data collected from multiple IVs, it is similar to the D term in (10), implying that (12) is a better tradeoff when $KV\mu_{iv} < N$. In cases where $N = K^2$, which ensures security against BG and BS attacks and minimizes cipher implementation complexity, (12) is the better tradeoff when $V\mu_{iv} < K$. These arguments apply equally to the non-distributed and distributed HS and DK approaches.

5 Applying Distributed TMTO Attacks to Counter Mode

In this section, we describe how distributed TMTO attacks can be applied to counter mode [20]. This is of interest because when a block cipher operates in counter mode, in addition to the key, the initial count value can be unpredictable during the preprocessing phase of TMTO attacks, making the building of the Hellman table more challenging, even when a chosen plaintext approach can be applied during the online phase. When counter mode is operated with a predictable initial count, Hellman's TMTO attack (distributed or non-distributed) can be directly applied by constructing tables for this known initial count.

5.1 Distributed Attack Without Data

In this section, we consider the application of a distributed TMTO attack to counter mode with a single key and an unpredictable initial count. (The non-distributed attack can be considered by simply letting $W = 1$.) Here, we shall use the term IV to refer to the unpredictable portion of the initial count and assume that the non-IV portion is fixed and predictable. We let V represent the number of possible values for the IV and to apply the attack, V Hellman tables to cover the keys are built (using appropriate chaining functions to map the cipher operation output to the next key input), one for each IV. An attack which does not use data in the tradeoff can be performed by dividing the t subtables of the V Hellman tables between the W processors. Letting $\log_2 V$ represent the size of the IV, the tradeoff used in this approach would be a simple modification of (5), where K is replaced by KV:

$$T_0 M_0^2 W^3 = (KV)^2 \tag{13}$$

with $W \leq T_0$ and preprocessing requiring $P_0 = KV/W$ to cover all key/IV combinations across all processors. We now consider an expression which indicates the size of W necessary to allow a TMTO attack to outperform a distributed exhaustive key search. This is equivalent to saying that the online time complexity and memory complexity of the TMTO attack should both be less than K/W. The resulting analysis leads to Proposition 3.

Proposition 3
Consider counter mode such that the key and the IV portion of the initial count are unpredictable during the preprocessing phase and assumed to be randomly drawn from the K and V possible values, respectively. With $T_0 = M_0^r$, a distributed tradeoff approach can be applied to obtain an attack with an online time complexity and memory complexity less than the complexity of distributed exhaustive key search for the following conditions on W:

$$W > \begin{cases} V^{\frac{2}{1-r}}/K^{\frac{r}{1-r}} & , r < 1 \\ 0 & , r = 1, \text{ if } V < K^{1/2} \\ \infty & , r = 1, \text{ if } V \geq K^{1/2} \\ K^{\frac{r-2}{2r-2}} V^{\frac{2r}{2r-2}} & , r > 1 \end{cases} \tag{14}$$

Proof
We need to show the conditions on W for which $T_0 < K/W$ and $M_0 < K/W$. The proof considers the three cases for r. For $r > 1$, $T_0 > M_0$ and, hence, it is sufficient to consider scenarios for $T_0 < K/W$, while for $r < 1$, $M_0 > T_0$, and, therefore, it is sufficient to consider $M_0 < K/W$. For the case of $r = 1$, $T_0 = M_0$ and we can consider a bound on either T_0 or M_0.

From (13), it can be shown that, if $r > 1$, then

$$T_0 = \frac{(KV)^{\frac{2r}{r+2}}}{W^{\frac{3r}{r+2}}} \tag{15}$$

which, when letting $T_0 < K/W$, leads to the result for $r > 1$.

Similarly, for $r < 1$,

$$M_0 = \frac{(KV)^{\frac{2}{r+2}}}{W^{\frac{3}{r+2}}} \tag{16}$$

which, when letting $M_0 < K/W$, leads to the result for $r < 1$.

Finally, letting $T_0 = M_0$, gives

$$T_0 = \frac{(KV)^{2/3}}{W} \tag{17}$$

which, when compared to K/W, results in an inequality not involving W, but which shows that, for $V < K^{1/2}$, the TMTO attack can improve upon distributed exhaustive key search for any W, while, for $V \geq K^{1/2}$, the TMTO attack cannot improve upon distributed exhaustive key search for any W. □

The interpretation of Proposition 3 can be demonstrated by considering the following example where we let $K = 2^{128}$ and $V = 2^{32}$. From Proposition 3, we can determine: (1) if $T_0 = M_0$, then $W > 0$, (2) if $T_0 = M_0^{1/2}$, then $W > 1$, and (3) if $T_0 = M_0^2$, $W > 2^{64}$. So we can conclude that a distributed TMTO attack can be made more efficient than distributed exhaustive key search for cases 1 and 2 by using as few as 1 and 2 processors, respectively, while for case 3, the number of processors must be more than 2^{64}, an impractically large requirement. Hence, for case 3, although it may be theoretically possible to mount a distributed TMTO attack, it is not practical to do so. Other examples for values of K, V and r can be considered to determine their practicality in terms of the number of required processors in a distributed attack.

The following proposition gives the relationship between K and V in order to ensure that it is impossible for a distributed TMTO attack to outperform distributed exhaustive key search for any tradeoff of time and memory (i.e., any r).

Proposition 4

Consider counter mode such that the key and the IV portion of the initial count are unpredictable during the preprocessing phase and assumed to be randomly drawn from the K and V possible values, respectively. If $V \geq K^{1/2}$, the online time complexity or the memory complexity of a distributed TMTO attack (which does not use multiple data) is at least as large as the complexity of a distributed exhaustive key search.

Proof

The best tradeoff from (13) occurs when we minimize the maximum of either T_0 or M_0, which occurs for $T_0 = M_0$, leading to $T_0 = (KV)^{2/3}/W$. If $V \geq K^{1/2}$, in this case clearly $T_0 \geq K/W$ and $M_0 \geq K/W$ for any W, where K/W is the complexity of a distributed exhaustive key search. Reducing T_0 at the expense of M_0 (or vice versa) still clearly results in M_0 (or T_0) being at least K/W. □

Proposition 4 implies that the entropy of the initial count (which is $\log_2 V$ for a random IV) should be at least half the size of the key to ensure security against distributed TMTO attacks, which do not use data. This is also true for non-distributed TMTO attacks, where $W = 1$.

5.2 Incorporating Data into the Attack

Consider now incorporating the use of data into the distributed TMTO attack on a single-key implementation of counter mode. In doing so, the distributed DK approach can be applied and, hence, the tradeoff of (11) can be used, with the constraints $W \leq T_0$ and $D_{iv} \leq V$, and $P_0 = KV/(D_{iv}W)$. Extending Proposition 4 leads to the following proposition.

Proposition 5
Consider counter mode such that the key and the IV portion of the initial count are unpredictable during the preprocessing phase and assumed to be randomly drawn from the K and V possible values, respectively. Assume that a distributed TMTO attack on a single-key system is applied with data available from D_{iv} IVs, where $D_{iv} \leq V$. If $V/D_{iv} \geq K^{1/2}$, the online time complexity or the memory complexity of a distributed TMTO attack is at least as large as the complexity of a distributed exhaustive key search.

Proof
We can simply follow the proof of Proposition 4, but base it on the distributed DK tradeoff of (11), which can be rewritten to be

$$T_0 M_0^2 W^3 = (K[V/D_{iv}])^2. \tag{18}$$

This equation is similar to (13) used in the proof of Proposition 4, except that we have substituted V with V/D_{iv}. Proposition 4 now follows with the same substitution, resulting in the distributed TMTO attack with data not being able to improve on distributed exhaustive key search when $V/D_{iv} \geq K^{1/2}$. □

Proposition 5 increases the lower bound on V for which the distributed TMTO attack becomes infeasible. Assuming that it is impractical for $D_{iv} > K^{1/2}$, then letting $V \geq K$ is sufficient to ensure security against TMTO attacks which make use of data. Now if $D_{iv}W = \alpha V$, where $\alpha > 1$, then $P_0 < K$, meaning the preprocessing time is better than exhaustive search on a cipher with key space K. Further, $T_0 M_0^2 = K^2/(\alpha^2 W) < K^2/W$, which could be substantially better than the tradeoff of the non-distributed approach. Consider the following case of counter mode using AES-128: $K = 2^{128}$, $V = 2^{32}$ and $W = 2^{20}$. If we let $T_0 = M_0$ and $D_{iv} = 2^{20}$ (so that $\alpha = 256$), we get $T_0 = M_0 = 2^{73.3}$, with $P_0 = 2^{120}$. Hence, the complexity of the online phase of the distributed TMTO attack is much better than the complexity of distributed exhaustive key search, which would be $K/W = 2^{108}$. Of course, collecting more data D_{iv} and/or involving more processors W could be used to improve the attack even further, but is still subject to the DK approach constraints of $D_{iv} \leq V$ and $W \leq T_0$.

To this point, we have only considered single-key systems. Note that the concept of attacking a multi-key block cipher system [5,21] where the cipher uses counter mode can result in the tradeoff (11) targeting the key and unpredictable initial count and may result in some systems being vulnerable.

6 Conclusions

In this paper, we have discussed the characterization of distributed TMTO attacks on ciphers. A summary of the characteristics of tradeoff attacks, including the distributed versions discussed in this paper, is presented in Appendix A. In Appendix B, numerical examples are used to illustrate the effectiveness of the attacks against a lightweight cipher (80-bit key) and an AES-level cipher (128-bit key).

Not surprisingly, distributing Hellman's approach can be highly effective, scaling both time and memory by the number of processors. Other tradeoff approaches such as the rainbow table method and the BG method are not as well suited to a distributed approach. The BS method benefits from a distributed approach in both time and memory, but the benefit of data in the tradeoff is not scaled by the number of processors involved. We have also described the application of distributed tradeoff attacks in relation to stream ciphers and have shown that distributed TMTO approaches can be effectively applied to counter mode in scenarios where the entropy of the initial count is too small. In particular, distributed TMTO attacks are of concern in the context of lightweight cryptography, where key sizes are smaller and the cryptanalytic gain of distributing the attacks could seriously compromise the security of some systems.

Appendix A: Summary of Tradeoffs

Table 1 contains a summary of all tradeoffs discussed and applied in this paper. Tradeoff expressions and preprocessing complexity, as well as target applications and meaningful restrictions on tradeoff parameters, are presented.

Appendix B: Numerical Results for Some Tradeoffs

In this section, we highlight a few cases to illustrate the applicability of the distributed TMTO attack. The data presented considers two key sizes of 80 bits (Table 2) and 128 bits (Table 3) and represents results for both stream ciphers and block ciphers using counter mode. A key size of 80 bits is consistent with the typical use of a lightweight block or stream cipher, while the 128-bit key represents an application that uses AES-128 level security. The results in the tables represent a tradeoff attack using the DK approach of a single-key system and the table values assume equal complexity for the online time and memory,

Table 1. Summary of Tradeoffs

	Tradeoff	Preprocessing	Target applications and restrictions
Exhaustive Key Search	$T = K$, $M = 1$	$P = 0$	block cipher key stream cipher key
Full Dictionary Attack	$T = 1$, $M = K$	$P = K$	block cipher key stream cipher key
Hellman	$TM^2 = K^2$	$P = K$	block cipher key
BG	$TM = N$	$P = N/D$	stream cipher state $D = T$
BS	$TM^2D^2 = N^2$	$P = N/D$	stream cipher state $D^2 \leq T$
HS	$TM^2D_{iv}^2 = (KV)^2$	$P = KV/D_{iv}$	stream cipher key/IV counter mode key/IV $D_{iv}^2 \leq T$
DK	$TM^2D_{iv}^2 = (KV)^2$	$P = KV/D_{iv}$	stream cipher key counter mode key $D_{iv} \leq V$ for single-key
Distributed Exh Key Srch	$T_0 = K/W$, $M_0 = 1$	$P_0 = 0$	block cipher key stream cipher key
Distributed Full Dict Att	$T_0 = 1$, $M_0 = K/W$	$P_0 = K/W$	block cipher key stream cipher key
Distributed Hellman	$T_0M_0^2W^3 = K^2$	$P_0 = K/W$	block cipher key $W \leq T_0$
Distributed BG	$T_0M_0W = N$	$P_0 = N/(DW)$	stream cipher state $D = T_0$
Distributed BS	$T_0M_0^2D^2W^3 = N^2$	$P_0 = N/(DW)$	stream cipher state $D^2W \leq T_0$
Distributed HS	$T_0M_0^2D_{iv}^2W^3 = (KV)^2$	$P_0 = KV/(D_{iv}W)$	stream cipher key/IV counter mode key/IV $D_{iv}^2W \leq T_0$ $D_{iv} \leq V$ for single-key
Distributed DK	$T_0M_0^2D_{iv}^2W^3 = (KV)^2$	$P_0 = KV/(D_{iv}W)$	stream cipher key counter mode key $W \leq T_0$ $D_{iv} \leq V$ for single-key

i.e., $T_0 = M_0$. The tradeoff expression of (11) is applied and the constraints $D_{iv} \leq V$ and $W \leq T_0$ are satisfied. For $V > 1$, $P_0 = KV/(D_{iv}W)$ resulting in

$$T_0 = \frac{P_0^{2/3}}{W^{1/3}} \tag{19}$$

which can be used to derive the values in the tables. However, for the case of $V = 1$ (that is, a predictable initial count in counter mode or a stream cipher with no IV), data cannot be used in the tradeoff and $P_0 = KV/W$ with (19) still suitable.

For both key sizes, various IV sizes are given and the complexity presented for cases of differing amounts of data, D_{iv}, and number of processors, W. For reference, the appropriate distributed exhaustive key search complexity (DEKS) is also presented for each case. Each TMTO case given in the tables has the online time complexity and the preprocessing complexity for an individual processor presented in the format "T_0/P_0".

It is obvious from the tables that there are many scenarios in which distributed TMTO attacks could be made more effective than a distributed exhaustive key search. Most notably, if $V = 1$, one Hellman table can be constructed straightforwardly to cover just the keys. In this case, although the use of data from multiple IVs is not applicable, applying a distributed approach can result in extremely small online time complexities - as low as $2^{33.3}$ for a lightweight cipher with an 80-bit key using 2^{20} processors. For cases with $V > 1$, using data drawn from a modest number of IVs can result in a compromise of the security of the cipher. For example, with $K = 2^{128}$ and $V = 2^{32}$, using data from only 2^{20} IVs and applying 2^{20} processors results in a TMTO attack with an online time complexity of $2^{73.3}$ and a preprocessing time complexity of 2^{120}. Hence, the online time complexity is substantially better than the distributed exhaustive key search complexity of 2^{108}, while the preprocessing complexity is only slightly worse.

Table 2. TMTO Results T_0/P_0 for 80-bit Keys

$K = 2^{80}$	DEKS	$V = 1$	$V = 2^{20}$	$V = 2^{40}$
$W = 1$, $D_{iv} = 1$	2^{80}	$2^{53.3}/2^{80}$	$2^{66.7}/2^{100}$	$2^{80}/2^{120}$
$W = 1$, $D_{iv} = 2^{10}$	2^{80}	$2^{53.3}/2^{80}$	$2^{60}/2^{90}$	$2^{73.3}/2^{110}$
$W = 2^{20}$, $D_{iv} = 1$	2^{60}	$2^{33.3}/2^{60}$	$2^{46.7}/2^{80}$	$2^{60}/2^{100}$
$W = 2^{20}$, $D_{iv} = 2^{10}$	2^{60}	$2^{33.3}/2^{60}$	$2^{40}/2^{70}$	$2^{53.3}/2^{90}$

Table 3. TMTO Results T_0/P_0 for 128-bit Keys

$K = 2^{128}$	DEKS	$V = 1$	$V = 2^{32}$	$V = 2^{64}$
$W = 1$, $D_{iv} = 1$	2^{128}	$2^{85.3}/2^{128}$	$2^{106.7}/2^{160}$	$2^{128}/2^{192}$
$W = 1$, $D_{iv} = 2^{20}$	2^{128}	$2^{85.3}/2^{128}$	$2^{93.3}/2^{140}$	$2^{114.7}/2^{172}$
$W = 2^{20}$, $D_{iv} = 1$	2^{108}	$2^{65.3}/2^{108}$	$2^{86.7}/2^{140}$	$2^{108}/2^{172}$
$W = 2^{20}$, $D_{iv} = 2^{20}$	2^{108}	$2^{65.3}/2^{108}$	$2^{73.3}/2^{120}$	$2^{94.7}/2^{152}$

References

1. Hellman, M.E.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theory **26**(4), 401–406 (1980)
2. Babbage, S.: A space/time tradeoff in exhaustive search attacks on stream ciphers. In: European Convention on Security and Detection, IEEE Conference Publication No. 408, pp. 161–166 (1995)
3. Golić, J.D.: Cryptanalysis of alleged A5 stream cipher. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 239–255. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_17
4. Biryukov, A., Shamir, A.: Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 1–13. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44448-3_1
5. Hong, J., Sarkar, P.: New applications of time memory data tradeoffs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005). https://doi.org/10.1007/11593447_19
6. Denning, D.E.: Cryptography and Data Security. Addison-Wesley, Boston (1982)
7. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_36
8. Armknecht, F., Mikhalev, V.: On lightweight stream ciphers with shorter internal states. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 451–470. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48116-5_22
9. Mikhalev, V., Armknecht, F., Müller, C.: On ciphers that continuously access the non-volatile key. IACR Trans. Symmetric Cryptol. **2016**(2), 52–79 (2016)
10. Hamann, M., Krause, M., Meier, W.: LIZARD - a lightweight stream cipher for power-constrained devices. IACR Trans. Symmetric Cryptol. **2017**(1), 45–79 (2017)
11. Dunkelman, O., Keller, N.: Treatment of the initial value in time-memory-data tradeoff attacks on stream ciphers. Inf. Process. Lett. **107**(5), 133–137 (2008)
12. Avoine, G., Junod, P., Oechslin, P.: Characterization and improvement of time-memory trade-off based on perfect tables. ACM Trans. Inf. Syst. Secur. **11**(4), 17:1–17:22 (2008)
13. Hong, J., Moon, S.: A comparison of cryptanalytic tradeoff algorithms. J. Cryptol. **26**(4), 559–637 (2013)
14. van den Broek, F., Poll, E.: A comparison of time-memory trade-off attacks on stream ciphers. In: Youssef, A., Nitaj, A., Hassanien, A.E. (eds.) AFRICACRYPT 2013. LNCS, vol. 7918, pp. 406–423. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38553-7_24
15. Borst, J., Preneel, B., Vandewalle, J.: On the time-memory tradeoff between exhaustive key search and table precomputation. In: Proceedings of the 19th Symposium in Information Theory in the Benelux, WIC, pp. 111–118 (1998)
16. Hong, J., Lee, G.W., Ma, D.: Analysis of the parallel distinguished point tradeoff. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 161–180. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25578-6_14
17. Kim, J.W., Seo, J., Hong, J., Park, K., Kim, S.-R.: High-speed parallel implementations of the rainbow method based on perfect tables in a heterogeneous system. Softw. Pract. Exper. **45**(6), 837–855 (2015)

18. Avoine, G., Carpent, X., Kordy, B., Tardif, F.: How to Handle Rainbow Tables with External Memory. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 306–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_16
19. Lee, G.W., Hong, J.: Comparison of perfect table cryptanalytic tradeoff algorithms. Des. Codes Crypt. **80**(3), 473–523 (2016)
20. National Institute of Standards and Technology. NIST Special Publication 800–38A: Recommendation for Block Cipher Modes of Operation, December 2001. https://csrc.nist.gov/publications/detail/sp/800-38a/final
21. Biryukov, A., Mukhopadhyay, S., Sarkar, P.: Improved time-memory trade-offs with multiple data. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 110–127. Springer, Heidelberg (2006). https://doi.org/10.1007/11693383_8