Chapter

# Tradeoff Attacks on Symmetric Ciphers

*Orhun Kara*

## Abstract

Tradeoff attacks on symmetric ciphers can be considered as the generalization of the exhaustive search. Their main objective is reducing the time complexity by exploiting the memory after preparing very large tables at a cost of exhaustively searching all the space during the precomputation phase. It is possible to utilize data (plaintext/ciphertext pairs) in some cases like the internal state recovery attacks for stream ciphers to speed up further both online and offline phases. However, how to take advantage of data in a tradeoff attack against block ciphers for single key recovery cases is still unknown. We briefly assess the state of art of tradeoff attacks on symmetric ciphers, introduce some open problems and discuss the security criterion on state sizes. We discuss the strict lower bound for the internal state size of keystream generators and propose more practical and fair bound along with our reasoning. The adoption of our new criterion can break a fresh ground in boosting the security analysis of small keystream generators and in designing ultra-lightweight stream ciphers with short internal states for their usage in specially low source devices such as IoT devices, wireless sensors or RFID tags.

**Keywords:** symmetric cipher, block cipher, stream cipher, tradeoff attack, keystream, keystream generator, Hellman table, rainbow table, one-way function, preimage

## 1. Introduction

In general, bulk encryption is performed through symmetric ciphers; that is, block ciphers or stream ciphers. Hash functions, message authentication codes and authenticated encryption schemes are also based on the quite similar design and security principles. All these cryptographic primitives are examples of one-way functions for which it must be computationally infeasible to find a preimage. Indeed, the only generic method to invert a given output is exhaustively searching for one of its inputs.[1] This may be embodied as brute force attacks on block ciphers and stream ciphers, internal state recovery attacks on keystream generators, preimage attacks on hash functions or constructing valid messages to given tag values for message authentication codes.

The brute force attacks can be expedited significantly by utilizing very large tables that have been already prepared during the offline phase. This phase is called the precomputation phase also and is usually equivalent to exhaustive search. Nevertheless, once it is executed, the prepared tables can be used several times.

[1] Permutations as one-way functions are out of scope of this chapter.

It may be possible to further improve a tradeoff attack by exploiting large amount of data (plaintext/ciphertext pairs). Biryukov-Shamir attack on keystream generators can be considered as a typical example of a tradeoff among time, memory and data [1]. One of the internal states of a long keystream sequence is recovered. However, it is still unknown how to use data to improve the tradeoff attacks on block ciphers.

The state sizes of block ciphers are not of security concern against tradeoff attacks, enabling to design ultra-lightweight block ciphers. In fact, we encounter several such block cipher designs in the literature during the last decades [2–9]. However, it seems to be almost impossible to design ultra-lightweight stream ciphers due to their strict security criterion on the lower bound of their internal state sizes to resist tradeoff attacks.

The tradeoff attacks can be quite effective against some real world cryptographic primitives. The tradeoff tables can be used in practical applications to break real life ciphers such as A5/1 for the GSM encryption [10–12] or to crack passwords by finding preimages to hash functions [13–17]. In this chapter, we introduce briefly how to use tradeoff tables to invert small sized one-way functions. Moreover, we evaluate the state of art of the applications, raise some open problems and come up with a discussion on the countermeasures against tradeoff attacks on keystream generators.

We argue that it is possible to loosen the lower bound for the state size without sacrificing the security against tradeoff attacks and this can enable designing ultra-lightweight stream ciphers. We claim that the lower bound for the internal state size can be diminished to $4n/3$ bits from $2n$ bits where $n$ is the key length. It is possible to design a keystream generator of size $4n/3$ bits, which remains still secure against tradeoff attacks and which presents a great advantage in low cost applications. Indeed, such ciphers are in real world demand due to the confidentially issues of lightweight devices such as RFID tags, wireless sensors or IoT devices.

It is straightforward that resistance against tradeoff attacks is not sufficient for security. Unfortunately, the security of small stream ciphers has not been studied sufficiently so far. We still do not know how to design secure and small stream ciphers. This is due to fact that almost all the stream ciphers in the literature have internal state sizes at least twice as large as their key sizes. Hence, there is almost no example in the literature to analyze. The recent small keystream generators such as Sprout [18] or Plantlet [19] are analyzed intensively in a short while and several weaknesses are discovered [20–26].

The tradeoff attacks on block ciphers so far are limited to the tradeoff between only time and memory. It is an open problem how to construct a tradeoff curve between memory and data or among memory, data and time for a single key recovery attack. We phrase the problem of *inverting one-way function with data*, the problem of *mutual inverting of multiple one-way functions* and the problem of *inverting only one of the several independent one-way functions*. Moreover, we address these problems with block ciphers and raise a question about the hierarchical relationships between any pair of them.

The outline of the chapter is as follows. We briefly overview the tradeoff attacks on symmetric ciphers, give some recent applications of these attacks and evaluate them in Section 2. Then, we assess the tradeoff attacks on stream ciphers and keystream generators in Section 3. We also introduce the tradeoff attacks on block ciphers, discuss the differences from those on stream ciphers and state some open problems in Section 4. We assess the internal state recovery tradeoff attacks and make an argument about the internal state sizes of keystream generators in Section 5. Finally, we introduce our concluding remarks in Section 6.

## 2. Inverting a one-way function through tradeoff

Let $f : GF(2)^m \rightarrow GF(2)^n$ be a one-way function of $m-$bit input and $n-$bit output. That is, it is easy to compute the output, $f(x) = y$, of a given input $x \in GF(2)^m$; but computationally infeasible to find a preimage $x \in f^{-1}(y)$ for a given output $y \in GF(2)^n$.

The phrase" computationally infeasible" is not a formal or a precise statement. Indeed, we mean that the fastest algorithm of finding a preimage $x \in f^{-1}(y)$ must be exhaustively searching for $x$, either online or offline. This definition is valid for random one-way functions which are generally not permutations. The one-way functions deduced from symmetric ciphers are examples and we consider them only throughout the chapter. The time complexity of recovering one preimage of a given value $y \in GF(2)^n$ is about $T = 2^n$ calls of the $f-$function (simply $2^n$) for a one-way function $f$. There is almost no memory or data complexity. Hence this can be considered as one of the extreme cases where only the time complexity dominates.

The time complexity may be substituted by the memory complexity if we compute all the $(x, f(x))$ values in advance during the offline phase (which we call precomputation phase) and save them in a sorted table with respect to the second column, $f(x)$. Then, the time complexity of the precomputation phase is $2^n$ and the memory complexity is $M = 2^n$. On the other hand, the time complexity of finding a preimage $x \in f^{-1}(y)$ for a given $y$ during the online phase is relatively negligible in comparison to the memory complexity. One needs to search for $y$ in the second column of the table and this search takes roughly $n$ steps since the table is sorted. This is also one of the extreme cases where only the memory complexity dominates.

In general, we can regard the tradeoff attacks as the attacks searching for a preimage of a one-way function by utilizing a significant memory prepared in the precomputation phase to reduce the time complexity from $2^n$. A tradeoff curve between memory and time is introduced with possibly some restrictions. The time complexity is decreased by increasing the memory complexity or vice versa. But the ratio of increase/decrease depends on the tradeoff curve. In general, the optimum point on the curve is considered as the point where $T = M$ if the restrictions permit to choose this point. Let us remark that the precomputation phases of these attacks must be the whole exhaustive search to provide significantly high success rates. But, since this offline phase is run only once, its complexity can be ignored in some applications where one uses the tables several times to invert enormous number of outputs. The Hellman tables or the rainbow tables for the GSM encryption algorithm A5/1 are typical real world applications [10, 12].

It is possible to ease the problem of inverting a one-way function $f$ by introducing large number of data. Then the corresponding tradeoff attacks can be further improved by constructing better tradeoff curves with the addition of the amount of data used.

We can define the problem of *inverting one-way function with data* as follows. Let $y_1, \ldots, y_D \in GF(2)^n$ be given. Then, find a preimage for one of them. That is, find $x_i$ such that $f(x_i) = y_i$. This problem is easier than finding a preimage of only one given element $y \in GF(2)^n$. Indeed, it is possible to prepare a sorted list of $y_1, \ldots, y_D$ and then search for $x$ such that $f(x)$ is in this sorted list. It is clear that the time complexity of the exhaustive search is $2^n/D$. Hence, the time complexity of the default attack for inverting one-way function with data is reduced by a factor of $D$.

It is possible to address the problem of inverting one-way function with data in stream ciphers and mount some tradeoff attacks for single key setting. We introduce these attacks in Section 3. However, it is not known in the literature yet how to

associate a single key recovery attack for a block cipher as a problem of inverting one-way function with data (see Section 4 for details of the tradeoff attacks in the case of block ciphers).

### 2.1 Hellman and rainbow tables

One very well known way of inverting a one-way function is using Hellman tables [27]. Initially, Hellman introduced the tables only for recovering the DES keys in his original work in [27] but it can be used to invert any one-way function.

Let us assume that the input and the output sizes of a one-way function, $f$, are equal. That is, $f : GF(2)^n \rightarrow GF(2)^n$. The general cases may easily be deduced by the reduction or enlargement techniques as Hellman applied for the DES encryption by reducing its block size to 56 bits. Let $x \in GF(2)^n$ be an input. Then, compute $f(x), f^2(x), \ldots, f^t(x)$ and save the pair, $\left(x, f^t(x)\right)$.

If a given value $y \in GF(2)^n$ is equal to $f^i(x)$ for some $i \in \{1, \ldots, t\}$ then we can find a preimage for y easily: $f^{i-1}(x)$ will be a preimage since $f\left(f^{i-1}(x)\right) = y$. We can check if $y = f^i(x)$ for some $i$ by checking the equality $f^{t-i}(y) = f^t(x)$. Indeed we have

$$f^{t-i}\left(f^i(x)\right) = f^t(x) = f^{t-i}(y). \tag{1}$$

Therefore, it is highly probable that $y = f^i(x)$. It may be possible that $y \neq f^i(x)$ even though $f^{t-i}(y) = f^t(x)$ since $f$ is not a permutation. This case is considered as a false alarm. The probability of the false alarms should be taken into account for the success rate of the attack. Gildas *et al.* introduce an efficient way of ruling out the false alarms, particularly in the perfect tables [28].

Choosing $m$ different $x$ points and preparing a table of $m$ pairs $\left(x, f^t(x)\right)$ sorted with respect to $f^t(x)$ (which is called a Hellman table), it is possible to find a preimage of a given output $y \in GF(2)^n$ if $y = f^i(x)$ for some $x$ in these $m$ pairs by calling the $f$ function and checking if the result is among the second (sorted) values of the pairs $\left(x, f^t(x)\right)$ at most $t$ times. Therefore, examining if $y$ is in the set $\left\{f^i(x)\right\}$ with $m \cdot t$ elements, costs $t$ calls of $f$ and the memory amount we need is $m$ since we save $m$ pairs for one table of $m \cdot t$ elements. These $m$ pairs consist of the initial and the final columns of the table.

The most significant disadvantage of Hellman tables is the high propagation of the collisions throughout the rows. If $f^i(x) = f^j(x')$ for some $1 \leq i, j < t$ and different starting points $x \neq x'$, then the collision is going to merge to the rest of the rows as $f^{i+k}(x) = f^{j+k}(x')\ \forall k = 1, \ldots, \min\{t-i, t-j\}$. This restricts the capacity of a Hellman table. Indeed, we should choose the number of the rows and the columns $m$ and $t$ such that $mt^2 \leq 2^n$ to optimize the probability of collisions according to the birthday paradox [27]. Therefore, we need roughly $t$ tables since one table can contain at most $mt$ different elements and each Hellman table must be prepared by using a different function deduced from a slight derivation of the $f$-function so as to ensure the independence of the tables.

The time complexity is $T = t^2$ since examining through one table costs $t$ calls of the $f$-function and we have $t$ tables. Similarly, we need $M = mt$ memory to save $t$ tables. As a corollary, the tradeoff curve $M^2T = 2^{2n}$ is deduced with $mt^2 = 2^n$. The optimum point on the curve is $T = M = 2^{2n/3}$. The precomputation phase for

preparing the tables is equivalent to the exhaustive search and hence its complexity is $2^n$.

Oechslin introduces another kind of tables to invert one-way functions, which he calls rainbow tables [29]. He proposes to use a different function for the computation of each column and hence each row is constituted as

$$f_1(x), f_2(f_1(x)), \ldots, f_t(f_{t-1}(\cdots f_1(x))) \tag{2}$$

instead of $f(x), f^2(x), \ldots, f^t(x)$ for a chosen starting point $x$ where $f_i$s are derived from $f$ by slight modifications. Only the initial point $x$ and its final evaluation $f_t(f_{t-1}(\cdots f_1(x)))$ are saved as in the case of Hellman tables.

Rainbow tables have a significant advantage over Hellman tables: The collisions in different columns do not propagate in rainbow tables. So, it is possible to use only one rainbow table for covering majority of the space $GF(2)^n$. The table contains $t$ columns and $mt$ rows. However, tracing through a rainbow table costs much more. For a given output $y$, check if it is in the last column. If not then check $f_t(y)$ and then $f_t(f_{t-1}(y))$ and then, $f_t(f_{t-1}f_{t-2}(y))$ and so on are in the last column one by one.

Both the Hellman tables and the rainbow tables have the same tradeoff curve. But, the time complexity is $t(t-1)/2$ for a rainbow table which is roughly twice less than $t^2$. This makes rainbow tables more popular in practical applications.

Barkan *et al.* compares these two methods and combine them in a general model based on stateful random graphs [30]. They also improve the time complexity of the rainbow tables [30]. Lu *et al.* use the unified rainbow tables to break GSM A5/1 algorithm and recover an A5/1 key in 9 s with a success rate of 81% by using general purpose GPUs with 3 NVIDIA GeForce GTX690 cards [12]. There are also FPGA implementation versions of tracing through the rainbow tables of the A5/1 states [10, 11]. The success rates of the rainbow tables for A5/1 are improved in [12]. Rainbow tables are commonly used to invert hash functions and crack passwords [13–17]. Even though rainbow tables are ubiquitously used in the real world applications, Biryukov *et al.* show that Hellman tables are superior to rainbow tables in multiple data scenario [31].

## 3. Tradeoff attacks on stream ciphers

The main building blocks of (synchronous) stream ciphers are keystream generators. The most general design principle of keystream generators make use of a state update function $\phi : GF(2)^s \rightarrow GF(2)^s$ and an output function $g : GF(2)^s \rightarrow GF(2)^r$ producing $r$-bit output from each $s$-bit internal state. An internal state $S_t$ is updated to the next internal state $S_{t+1}$ via $\phi$. The initial internal state $S_0$ is called the seed and produced from a key $K$ and an initial vector $IV$ through an initialization algorithm $InAlg$:

$$\begin{aligned} InAlg : GF(2)^n \times GF(2)^l &\rightarrow GF(2)^s \\ (K, IV) &\mapsto S_0. \end{aligned} \tag{3}$$

The objective of the attacks on stream ciphers is twofold in general. They aim at either recovering the key or an internal state. The same approach is adopted for tradeoff attacks. The state recovery attacks are conventional examples of the problem of inverting one-way function with data in a single key attack scenario. Indeed, it is enough to recover one of the internal states occurred during the encryption process.

Babbage [32] and Golić [33] independently introduce a natural way of recovering one of the internal states by using data. They define a one-way function by extending the output function which produces enough number of output bits by calling $\phi$ and $g$ certain number of times consecutively to identify the input state from its keystream piece uniquely. One can compute $M$ pairs of the states and their outputs during the precomputation phase and save them as sorted with respect to the outputs. Then, it is highly probable to recover one of the states which produce $D$ data when $MD \geq 2^s$ during the online phase. The optimum point on the tradeoff curve $MD = 2^s$ is $M = D = 2^{s/2}$. So, $s/2$ is supposed to be larger than the key length to ensure that the Babbage-Golić attack is slower than the exhaustive search. This imposes a well known and highly adopted security criterion on stream ciphers: The internal state size must be at least twice as large as the key size. It was one of the main security requirements for the stream ciphers in both the NESSIE project [34] and the eSTREAM project [35, 36].

Another tradeoff attack on keystream generators using data is introduced by Biryukov and Shamir [1]. They propose to use Hellman tables to recover one of the internal states which produce $D$ data. It is nothing but finding a preimage for one of the data. The optimum online complexity is achieved when only one Hellman table is constructed. So, $mt^2 = 2^s$ and $D = t$ with $M = m, T = tD$. Hence, we have the tradeoff curve given as $M^2D^2T = 2^{2s}$ with the restriction $D \leq \sqrt{T}$. The optimum point on the curve is achieved when $D^2 = T = M$ and this gives $T = 2^{s/2}$. Again, if $s \geq 2n$ then the online phase of the Biryukov-Shamir attack will be slower than the exhaustive search, confirming the security criterion that the internal state size should be at least twice as large as the key size.

Both the Babbage-Golić attack and the Biryukov-Shamir attack aim at recovering one of the internal states. The online phases of these attacks are compared with the exhaustive search rather than the default tradeoff attacks. The attacks use multiple data since the one-way function they would like to invert has several outputs available. On the other hand, it is possible to define the one-way function as the function taking the $n$-bit main key as input and producing the keystream of $n$-bits for a chosen fixed $IV$. The internal state size has no significance for inverting this one-way function. So, we have the classical complexities $T = M = 2^{2n/3}$. However, we can not exploit the multiple data for this function. Therefore the Babbage-Golić attack and the Biryukov-Shamir attack are superior when the internal state size is too short. The tradeoff attacks on the GSM encryption algorithm A5/1 with its 64 bit internal state are mostly the applications of the Biryukov-Shamir attack [10–12].

Armknecht and Mikhalev examine the keyed update functions and show that the keystream generators with keyed state update functions are secure against conventional tradeoff attacks no matter how small the internal state sizes are [18]. They also introduce an example cipher they call Sprout [18]. A keyed state update function takes the main key as the second parameter of the input to produce the next internal state from the current internal state.

The cipher Sprout is analyzed intensively in a short while and some weaknesses are discovered [20, 22]. More interestingly, special tradeoff attacks are mounted [21, 23]. Then, Armknecht and Mikhalev present another keystream generator with keyed state update. They call it Plantlet [19]. This cipher also attains significant interests of cyrptanalysts and several results are published including correlation attacks [24–26, 37, 38], some of them are even faster than exhaustive search [25]. It seems that it is indeed a challenging task for the crypto community to design keystream generators of small state sizes even if the tradeoff attacks are ignored in their security assessments.

## 4. Tradeoff attacks on block ciphers

Let $E : GF(2)^n \times GF(2)^m \rightarrow GF(2)^m$ be a block cipher of $n$-bit key and $m$-bit block size. $E(K,P) = E_K(P)$ and $E_K$ is a permutation for a fixed key $K$. We can define a one-way function $f(x) = E(x,P_0) = E_x(P_0)$ for a chosen fixed plaintext $P_0$. Finding a preimage for a given ciphertext is nothing but finding a key candidate that encrypts the plaintext $P_0$ to the given ciphertext.

It is possible to invert $f(x)$ by using tradeoff tables. Hellman initially mounted the tradeoff attack on the block cipher DES in his original work [27]. The online time complexity is reduced to $2^{2n/3}$. But preparing the tables requires as many encryption calls as in the exhaustive search.

There is no known method of using multiple data to improve the tradeoff curve $M^2T = 2^{2n}$ in the single key recovery setting for block ciphers yet. Choosing another plaintext will result in another one-way function to convert. So, using multiple data yields the following problem. Let $f_1, \ldots, f_D$ be $D$ independent one-way functions of $n$-bit inputs and $n$-bit outputs. We call the problem of finding $x$ as the problem of the *mutual inverting of multiple one-way functions* where

$$f_1(x) = y_1, f_2(x) = y_2, \ldots, f_D(x) = y_D \tag{4}$$

and $y_1, \ldots, y_D$ are given.

Choosing $D$ different plaintexts $P_1, \ldots, P_D$ for a block cipher $E$ is an example of the problem of the mutual inverting of multiple one-way functions given as: $f_1(x) = E_x(P_1), f_2(x) = E_x(P_2), \ldots, f_D(x) = E_x(P_D)$. Here $x$ is the key and we have $D$ chosen plaintexts encrypted with $x$. Then, finding $x$ becomes a mutual inverting problem of multiple one-way functions.

The problem may further be generalized as inverting only one of the $D$ independent one-way functions. Let

$$f_1(x_1) = y_1, f_2(x_2) = y_2, \ldots, f_D(x_D) = y_D \tag{5}$$

be given for $D$ independent one-way functions $f_1, \ldots, f_D$. The goal is to find one of $x_i$ for $i = 1, \ldots, D$.

The problem of mutual inverting multiple one-way functions can be applied to stream ciphers also. Several one-way functions may be defined by choosing several $IV$s. Each $IV$ determines a one-way function taking the key as the input and producing $n$-bit keystream. That is, each one-way function $f_{IV} : GF(2)^n \rightarrow GF(2)^n$ is defined as

$$f_{IV}(K) = (z_1, \ldots, z_n) \tag{6}$$

where $K$ is an $n$-bit key and $(z_1, \ldots, z_n)$ is the first $n$-bit keystream segment produced by the pair $(K, IV)$. The function $f_{IV}$ can be inverted by the conventional Hellman tables or rainbow tables. Finding preimage for one specific $f_{IV}$ can be considered as the default tradeoff attack on stream ciphers and its online complexity is given as $2^{2n/3}$.

It may be still possible to use any number of $IV$s. For the single key attack scenario, the keystream generator is initialized by several different $IV$s and the corresponding $n$-bit keystream segments are produced. Then, the unknown inputs of the one-way functions will be mutual, namely the main key.

It seems that inverting only one specific one-way function once is not easier than the other two problems. One can use the algorithm of inverting a one-way function

to invert one of $D$ one-way functions. So, an algorithm inverting a one-way function can be used to solve the problem of inverting at least one function among $D$ one-way functions. Similarly, any algorithm inverting one of $D$ one-way functions can straightforwardly be used to solve the mutual inverting problem.

It is not known yet if these three problems are of equal difficulty. It is an open problem if the mutual inverting problem is strictly easier than the problem of inverting one of the several one-way functions. It is also an open problem that inverting one of the several one-way functions is strictly easier than inverting only one one-way function. If there is an algorithm solving problem of mutual inverting problem but not solving the problem of inverting one-way function then the security levels and the key lengths for both block ciphers and stream ciphers must be assessed again. Because, the algorithms solving mutual inverting problems efficiently can be very powerful and serious attacks on symmetric ciphers.

## 5. Assessment of security criterion on state size

The online complexities of both the Babbage-Golić and the Biryukov-Shamir attacks are compared to the complexity of the exhaustive search and the security criterion on the state size of a stream cipher is imposed thereof. However, there is still a faster tradeoff attack even though the internal state size is larger than twice of the key size. It is possible to define a one-way function from a main key to its keystream piece of a stream cipher by choosing and fixing an $IV$. Then, one of the preimages of the keystream segment will be the main key. The attack complexity is derived from the key size rather than the internal state size. At the optimum point of the tradeoff curve, the online complexity is $2^{2n/3}$ where $n$ is the key length. This is the default Hellman or Oechslin tradeoff attacks and valid for block ciphers also. Note that the complexity is much smaller than $2^n$, the complexity of the exhaustive search.

Any tradeoff attack on symmetric ciphers should be compared with the default tradeoff attack with its complexity $2^{2n/3}$, instead of the exhaustive search. In this case, the strict criterion on the internal state size can be lightened, enabling to design ultra-lightweight stream ciphers. Indeed, a stream cipher of 128 bit key is required at least 256 bit internal state according to the conventional security criterion. If we assume one bit register is implemented by a flip flop of 6 GE (Gate Equivalent) area, we must allocate roughly 1.5 K GE only for the registers. This is why there is almost no stream cipher in the literature having a hardware implementation less than 1 K GE. However, there are several block cipher designs with hardware implementations less than 1 K GE such as Ktantan [9], PRINTCipher [39], SLIM [2] and LBlock [7].

Recall that we have the tradeoff curve $MD = 2^s$ for the Babbage-Golić attack with the optimum point $M = N = 2^{s/2}$ where $s$ is the internal state size of a given stream cipher. The online time complexity is also equal to the data complexity. Then, we simply should consider the attack to be successful if $2^{s/2} < 2^{2n/3}$. Therefore, the internal state size must be at least $4n/3$. An attacker may prefer to choose much larger $M$ on the curve $MD = 2^s$. For example, preparing a memory of $M = 2^n$, we have $D = 2^{n/3}$ for the case $s = 4n/3$. However, it is possible to restrict the total number of the keystream bits produced per one key and force the users to change the key before completing encrypting the amount of $2^{n/3}$ data.

Similarly, the optimum point of the tradeoff curve for the Biryukov-Shamir attack is $D^2 = M = T = 2^{s/2}$ where $M^2D^2T = 2^{2s}$. Then, the attack will be slower than the default key recovery tradeoff attack if again $2^{s/2} \geq 2^{2n/3}$. Once more, we achieve the

same security bound that the minimum size for the internal state must be $4n/3$. If the precomputation phase is required to be not faster than the exhaustive search, then the amount of data encrypted per one key can be bounded above by $2^{n/3}$.

As a result, the tradeoff attacks aiming at the internal state recovery should be compared to the default tradeoff key recovery attack. Then, it is possible to loosen the restriction on the state size from $2n$ to $4n/3$. This new criterion can enable novel designs of ultra-lightweight stream ciphers. However, stream ciphers with short internal states may prone to several other attacks. The attacks on Plantlet and Sprout are the examples [20–26, 37, 38]. Therefore, it seems to be a fruitful challenge for the cryptography community to design secure stream ciphers having quite short internal states. On the other hand, the real world applications such as IoT devices, RFID tags or wireless sensors require ultra-lightweight stream ciphers for confidentially.

## 6. Conclusions

We briefly introduce the tradeoff attacks on symmetric ciphers and initiate hopefully a fruitful discussion about how to assess the degree of precautions or countermeasure to be taken against these attacks.

The tradeoff attacks targeting at recovering one of the internal states producing a given keystream sequence are compared to the exhaustive search attack on the corresponding key used. However, a stream cipher key can be recovered much faster thorough the default tradeoff attack. Therefore, the internal state recovery tradeoff attacks should be compared to the default key recovery tradeoff attack. In this case, it is possible to loosen the bound for the countermeasure taken against state recovery tradeoff attacks.

The internal state size is supposed to be at least twice as large as the key size if the security threshold for tradeoff attacks is taken as the complexity of the exhaustive search. This is indeed a well known and worldwide adopted security criterion. We argue that it is indeed not necessary to allocate such large internal state just for the resistance against tradeoff attacks. The internal state size is enough to be at least $4n/3$-bits particularly for the lightweight applications where $n$ is the key length. Besides, there are several other cyrptanalytic techniques for internal state recovery that must be taken into account. It is an open problem how to design secure stream ciphers with short internal states. Such ciphers must be secure against other types of attacks such as divide-and-conquer attacks, guess and determine attacks or correlation attacks. It is interesting to study this generic problem.

We believe that it is a challenging task to design small stream ciphers and the industry requires such ciphers to use in lightweight applications such as IoT devices, wireless sensors or RFID tags.

## Acknowledgements

We would like to thank Mehmet Sabır Kiraz, Ali Aydın Selçuk and Sırrı Erdem Ulusoy for their helpful comments. We also would like to thank IntechOpen LIMITED for the grant.

## Conflict of interest

The authors declare no conflict of interest.

## Author details

Orhun Kara
Department of Mathematics, Faculty of Science, IZTECH Izmir Institute of Technology, Urla, Izmir, Turkey

*Address all correspondence to: orhunkara@iyte.edu.tr

## References

[1] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Advances in Cryptology - ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 1–13, Springer, 2000.

[2] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, and M. M. Dessouky, "SLIM: A lightweight block cipher for internet of health things," *IEEE Access*, vol. 8, pp. 203747–203757, 2020.

[3] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An ultra-lightweight blockcipher," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, pp. 342–357, Springer, 2011.

[4] L. Li, B. Liu, and H. Wang, "QTL: A new ultra-lightweight block cipher," *Microprocess. Microsystems*, vol. 45, pp. 45–55, 2016.

[5] Z. Gong, S. Nikova, and Y. W. Law, "KLEIN: A new family of lightweight block ciphers," in *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26–28, 2011, Revised Selected Papers* (A. Juels and C. Paar, eds.), vol. 7055 of *Lecture Notes in Computer Science*, pp. 1–18, Springer, 2011.

[6] H. AlKhzaimi and M. M. Lauridsen, "Cryptanalysis of the SIMON family of block ciphers," *IACR Cryptol. ePrint Arch.*, vol. 2013, p. 543, 2013.

[7] W. Wu and L. Zhang, "Lblock: A lightweight block cipher," in *Applied Cryptography and Network Security - 9th International Conference, ACNS 2011, Nerja, Spain, June 7–10, 2011. Proceedings* (J. López and G. Tsudik, eds.), vol. 6715 of *Lecture Notes in Computer Science*, pp. 327–344, 2011.

[8] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw, "The LED block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, pp. 326–341, Springer, 2011.

[9] C. D. Cannière, O. Dunkelman, and M. Knezevic, "KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers," in *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6–9, 2009, Proceedings* (C. Clavier and K. Gaj, eds.), vol. 5747 of *Lecture Notes in Computer Science*, pp. 272–288, Springer, 2009.

[10] M. Kalenderi, D. N. Pnevmatikatos, I. Papaefstathiou, and C. Manifavas, "Breaking the GSM A5/1 cryptography algorithm with rainbow tables and high-end FPGAS," in *22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29–31, 2012* (D. Koch, S. Singh, and J. Tørresen, eds.), pp. 747–753, IEEE, 2012.

[11] P. Papantonakis, D. N. Pnevmatikatos, I. Papaefstathiou, and C. Manifavas, "Fast, fpga-based rainbow table creation for attacking encrypted mobile communications," in *23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2–4, 2013*, pp. 1–6, IEEE, 2013.

[12] Z. Li, "Optimization of rainbow tables for practically cracking GSM A5/1 based on validated success rate modeling," in *Topics in Cryptology -*

*CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29–March 4, 2016, Proceedings* (K. Sako, ed.), vol. 9610 of *Lecture Notes in Computer Science*, pp. 359–377, Springer, 2016.

[13] J. Bieniasz, K. Skowron, M. Trzepinski, M. Rawski, P. Sapiecha, and P. Tomaszewicz, "Hardware implementation of rainbow tables generation for hash function cryptanalysis," in *Information Systems Architecture and Technology: Proceedings of 36th International Conference on Information Systems Architecture and Technology - ISAT 2015 - Part II, Karpacz, Poland, September 20–22, 2015* (A. Grzech, L. Borzemski, J. Swiatek, and Z. Wilimowska, eds.), vol. 430 of *Advances in Intelligent Systems and Computing*, pp. 189–200, Springer, 2015.

[14] G. Avoine, A. Bourgeois, and X. Carpent, "Analysis of rainbow tables with fingerprints," in *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29–July 1, 2015, Proceedings* (E. Foo and D. Stebila, eds.), vol. 9144 of *Lecture Notes in Computer Science*, pp. 356–374, Springer, 2015.

[15] J. Horalek, F. Holík, O. Horák, L. Petr, and V. Sobeslav, "Analysis of the use of rainbow tables to break hash," *J. Intell. Fuzzy Syst.*, vol. 32, no. 2, pp. 1523–1534, 2017.

[16] H. Ying and N. Kunihiro, "Decryption of frequent password hashes in rainbow tables," in *Fourth International Symposium on Computing and Networking, CANDAR 2016, Hiroshima, Japan, November 22–25, 2016*, pp. 655–661, IEEE Computer Society, 2016.

[17] G. Avoine, X. Carpent, and C. Lauradoux, "Interleaving cryptanalytic time-memory trade-offs on non-uniform distributions," in *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21–25, 2015, Proceedings, Part I* (G. Pernul, P. Y. A. Ryan, and E. R. Weippl, eds.), vol. 9326 of *Lecture Notes in Computer Science*, pp. 165–184, Springer, 2015.

[18] F. Armknecht and V. Mikhalev, "On lightweight stream ciphers with shorter internal states," in *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8–11, 2015, Revised Selected Papers* (G. Leander, ed.), vol. 9054 of *Lecture Notes in Computer Science*, pp. 451–470, Springer, 2015.

[19] V. Mikhalev, F. Armknecht, and C. Müller, "On ciphers that continuously access the non-volatile key," *IACR Trans. Symmetric Cryptol.*, vol. 2016, no. 2, pp. 52–79, 2016.

[20] V. Lallemand and M. Naya-Plasencia, "Cryptanalysis of full sprout," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2015, Proceedings, Part I* (R. Gennaro and M. Robshaw, eds.), vol. 9215 of *Lecture Notes in Computer Science*, pp. 663–682, Springer, 2015.

[21] B. Zhang and X. Gong, "Another tradeoff attack on sprout-like stream ciphers," in *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II* (T. Iwata and J. H. Cheon, eds.), vol. 9453 of *Lecture Notes in Computer Science*, pp. 561–585, Springer, 2015.

[22] S. Maitra, S. Sarkar, A. Baksi, and P. Dey, "Key recovery from state information of sprout: Application to cryptanalysis and fault attack," *IACR Cryptol. ePrint Arch.*, vol. 2015, p. 236, 2015.

[23] M. F. Esgin and O. Kara, "Practical cryptanalysis of full sprout with TMD tradeoff attacks," in *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12–14, 2015, Revised Selected Papers* (O. Dunkelman and L. Keliher, eds.), vol. 9566 of *Lecture Notes in Computer Science*, pp. 67–85, Springer, 2015.

[24] O. Kara and M. F. Esgin, "On analysis of lightweight stream ciphers with keyed update," *IEEE Trans. Computers*, vol. 68, no. 1, pp. 99–110, 2019.

[25] S. Banik, K. Barooti, and T. Isobe, "Cryptanalysis of plantlet," *IACR Trans. Symmetric Cryptol.*, vol. 2019, no. 3, pp. 103–120, 2019.

[26] Y. Todo, W. Meier, and K. Aoki, "On the data limitation of small-state stream ciphers: Correlation attacks on fruit-80 and plantlet," in *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12–16, 2019, Revised Selected Papers* (K. G. Paterson and D. Stebila, eds.), vol. 11959 of *Lecture Notes in Computer Science*, pp. 365–392, Springer, 2019.

[27] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[28] G. Avoine, P. Junod, and P. Oechslin, "Characterization and improvement of time-memory trade-off based on perfect tables," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 4, pp. 17:1–17:22, 2008.

[29] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 2003, Proceedings* (D. Boneh, ed.), vol. 2729 of *Lecture Notes in Computer Science*, pp. 617–630, Springer, 2003.

[30] E. Barkan, E. Biham, and A. Shamir, "Rigorous bounds on cryptanalytic time/memory tradeoffs," in *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20–24, 2006, Proceedings* (C. Dwork, ed.), vol. 4117 of *Lecture Notes in Computer Science*, pp. 1–21, Springer, 2006.

[31] A. Biryukov, S. Mukhopadhyay, and P. Sarkar, "Improved time-memory trade-offs with multiple data," in *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11–12, 2005, Revised Selected Papers* (B. Preneel and S. E. Tavares, eds.), vol. 3897 of *Lecture Notes in Computer Science*, pp. 110–127, Springer, 2005.

[32] S. Babbage, "Improved exhaustive search attacks on stream ciphers." Security and Detection 1995, European Convention IET, 1995.

[33] J. D. Golić, "Cryptanalysis of alleged A5 stream cipher," in *EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 239–255, Springer, 1997.

[34] B. Preneel, *NESSIE Project*, pp. 408–413. Boston, MA: Springer US, 2005.

[35] M. Robshaw, "The estream project," in *New Stream Cipher Designs - The eSTREAM Finalists* (M. J. B. Robshaw and O. Billet, eds.), vol. 4986 of *Lecture Notes in Computer Science*, pp. 1–6, Springer, 2008.

[36] V. Rijmen, "Stream ciphers and the estream project," *ISC Int. J. Inf. Secur.*, vol. 2, no. 1, pp. 3–11, 2010.

[37] J. Copeland and L. Simpson, "Finding slid pairs for the plantlet stream cipher," in *Proceedings of the Australasian Computer Science Week,*

*ACSW 2020, Melbourne, VIC, Australia, February 3–7, 2020* (P. P. Jayaraman, D. Georgakopoulos, T. K. Sellis, and A. Forkan, eds.), pp. 7:1–7:7, ACM, 2020.

[38] S. Wang, M. Liu, D. Lin, and L. Ma, "Fast correlation attacks on grain-like small state stream ciphers and cryptanalysis of plantlet, fruit-v2 and fruit-80," *IACR Cryptol. ePrint Arch.*, vol. 2019, p. 763, 2019.

[39] M. J. Mihaljević, S. Gangopadhyay, G. Paul, and H. Imai, "Generic cryptographic weakness of *k*-normal boolean functions in certain stream ciphers and cryptanalysis of Grain-128," *Periodica Mathematica Hungarica*, vol. 65, no. 2, pp. 205–227, 2012.