Time-Memory-Data Tradeoff Attacks against Small-State Stream Ciphers

Matthias Hamann[1], Matthias Krause[1], Willi Meier and Bin Zhang[3]

[1] University of Mannheim, Germany, {hamann,krause}@uni-mannheim.de
[2] FH Nordwestschweiz, Switzerland, willi.meier@fhnw.ch
[3] Chinese Academy of Sciences, China, zhangbin@tca.iscas.ac.cn

Abstract. Time-memory-data (TMD) tradeoff attacks limit the security level of many classical stream ciphers (like E_0, A5/1, Trivium, Grain) to $\frac{1}{2}n$, where n denotes the inner state length of the underlying keystream generator. This implies that to withstand TMD tradeoff attacks, the state size should be at least double the key size. In 2015, Armknecht and Mikhalev introduced a new line of research, which pursues the goal of reducing the inner state size of lightweight stream ciphers below this boundary by deploying a key-dependent state update function in a Grain-like stream cipher. Although their design Sprout was broken soon after publication, it has raised interest in the design principle, and a number of related ciphers have been suggested since, including Plantlet, a follow-up of Sprout, and the cipher Fruit.
In this paper, existing TMD tradeoff attacks are revisited, and new insights on distinguishers and key recovery related to small-state stream ciphers are derived. A particular result is the transfer of a generic distinguishing attack suggested in 2007 by Englund, Hell, and Johansson to this new class of lightweight ciphers. Our analysis shows that the initial hope of achieving full security against TMD tradeoff attacks by continuously using the secret key has failed. In particular, we demonstrate that there are generic distinguishing attacks against Plantlet and Fruit with complexity significantly smaller than that of exhaustive key search. However, by studying the assumptions underlying the applicability of these attacks, we are able to come up with a new design idea for small-state stream ciphers which might allow to finally achieve full security against TMD tradeoff attacks.
Another contribution of this paper is the first key recovery attack against the most recent version of Fruit. We show that there are at least 2^{64} weak keys, each of which does not provide 80-bit security as promised by designers. This new attack against Fruit, together with previous attacks against Sprout, raises the question whether a more complicated key schedule than the basic one used in Plantlet is actually beneficial for the security of such ciphers.

Keywords: Stream Ciphers · Lightweight Cryptography · Time-Memory-Data Tradeoff Attacks · Plantlet · Fruit

1 Introduction

Stream ciphers have a long history when it comes to protecting digital communication. In 1987, Ronald L. Rivest designed RC4 [Sch95], which was later used in SSL/TLS [DR08] and the wireless network security protocols WEP [Ins97] and TKIP (often called WPA) [Ins04]. Other well-known stream cipher examples are E_0 of the Bluetooth standard [SIG14] and A5/1 of GSM [BGW99]. Unfortunately, E_0 and A5/1 have been shown to be highly insecure (see, e.g., [LMV05] and [BB06]) and RC4 also shows severe vulnerabilities, which led to its removal from the TLS protocol [Pop15] and rendered other protocols like WEP insecure [FMS01]. In 2004, the eSTREAM project [ECR08] was started in order to

identify new stream ciphers for different application profiles. In the hardware category, aiming at devices with restricted resources, three ciphers are still part of the eSTREAM portfolio after the latest revision in 2012: Grain v1 [HJM06], MICKEY 2.0 [BD06] and Trivium [CP05].

Common to these three ciphers is that they have an inner state length of at least twice the size of the targeted security level against key recovery attacks. This is due to the inherent vulnerability of classical stream ciphers (i.e., stream ciphers which compute the keystream based on a so-called *initial state*) against TMD tradeoff attacks like those of Babbage [Bab95] and Biryukov and Shamir [BS00], which allow to recover some inner state during keystream generation (and, usually, also the corresponding initial state by clocking the cipher backwards) with an overall attack complexity of $2^{n/2}$, where n denotes the inner state length of the underlying keystream generator (KSG). If the state initialization algorithm, which computes the initial state from a given key/IV pair, is efficiently invertible (as it is, e.g., for Trivium and Grain), knowing the initial state immediately reveals the secret key. And even if the state initialization algorithm is not efficiently invertible, variants of such TMD tradeoff attacks can allow for key recovery, e.g., by targeting the inner state at $t = 0$, which often contains the secret key (cf. Trivium and Grain). A generic view on these attacks is provided in [HK15] along with a corresponding complexity analysis.

In 2015, a new line of research emerged with the publication of Sprout [AM15] by Armknecht and Mikhalev, which pursues the goal of reducing the size of the volatile inner state of lightweight stream ciphers below this magic boundary formerly induced by TMD tradeoff attacks. We will refer to such ciphers, whose volatile inner state size is less than twice the key size, by the term *small-state stream cipher*. Sprout has a Grain-like structure and uses two 40-bit feedback shift registers. Compared to conventional stream ciphers like Grain v1, the characteristic difference of Sprout is that the 80-bit key is not only accessed during the state initialization but also continuously used as part of the state update during the subsequent keystream generation phase. Even though Sprout was broken shortly after publication (see, e.g., [LNP15], [ZG15], [Ban15], [EK16]), it has sparked interest in the underlying design principle and related ciphers like Fruit [GHX16] have been suggested since. Please note that as the designers of Fruit have changed the specification of their cipher several times in the past (e.g., ePrint versions 20160521:111224, 20161124:115414, 20170304:073404 of [GHX16] all contain different algorithms), we will refer to the most recent version of Fruit (20170304:073404) as Fruit v1 in the rest of this paper and hope that the designers will follow this versioning scheme for potential future updates.[1]

Unfortunately, though being elegant in theory, continuously accessing the secret key often comes at a heavy price in practice. For example, if the key is stored in an EEPROM, the corresponding access times may significantly slow down the operation speed of the KSG. This is particularly true if the key bits are not accessed sequentially (as in the case of Sprout and its successor Plantlet [MAM17]) but at random positions (as in the case of Fruit v1). In fact, Fruit v1 needs to access six different, non-sequential key bits per clock cycle. On the other hand, in a scenario where the key is fixed (e.g., via burning fuses), continuously accessing key bits (in potentially random order) is clearly feasible. Hence the question whether a more complicated key schedule as in the case of Fruit v1 does provide additional security when compared to, e.g., Plantlet, is not only interesting from a theoretical point of view but of actual practical relevance. In fact, the designers of Fruit v1 claim in [GHX16] (version 20170304:073404): "*The key bits are not properly used as a section of the internal state in Sprout and Plantlet (every 80 clocks, each key bit participates in internal state updating only one time). In Fruit, the balance of participation of the key (as a section of the internal state) is suitable in the internal state updating. We*

[1] This would not only be beneficial for the cryptographic community but also for the designers themselves as potential users will immediately recognize, which version of Fruit is actually considered secure. In analogy, when a weakness in the first version of Grain was discovered, the designers called the updated design Grain v1.

predict that very soon some attacks will be published against Plantlet [MS17].”

Ironically, while Plantlet has not been broken in the meantime, this paper now presents a key recovery attack against Fruit v1 that makes use of the cipher's insecure key schedule. In fact, our results seems to raise the question whether there is actually a need for (respectively a benefit of) a more complicated key schedule than the basic one used in Plantlet. Because not only Fruit v1 has now been broken due to its nonlinear key schedule but also Sprout initially suffered from this fate. It seems that simplicity may actually be preferable here. After all, the original idea underlying the use of key bits for the state update during keystream generation was simply to protect against TMD tradeoff attacks aiming at inner state recovery. Sequentially using one key bit per clock cycle as done by Plantlet already seems to do this job.

Another contribution of this paper is the discussion of TMD tradeoff distinguishing attacks against ciphers like Sprout, Fruit v1, and Plantlet, which continuously use the secret key as part of the state update. The existence of such attacks with a complexity below that of exhaustive key search was considered an exclusion criterion in the eSTREAM contest. Making use of a result by Englund, Hell and Johansson [EHJ07], we give a generic keystream-based TMD tradeoff distinguisher with attack complexity way below the complexity of exhaustive key search for Plantlet and Fruit v1. In fact, Banik already showed in 2015 [Ban15] that there is a similar distinguisher for Sprout. His attack can also be seen as a specific variant of the generic distinguisher by Englund, Hell and Johansson from 2007 (i.e., long before Sprout, Fruit, and Plantlet were introduced).

We would like to point out that, in our opinion, the existence of distinguishing attacks with a complexity below that of exhaustive key search should not be a knock-out criterion for stream ciphers targeting ultra-lightweight applications because such attacks might actually be tolerable depending on the application scenario. Nonetheless, the user needs to know about the corresponding complexities in order to be able to decide on this question. In [GHX16] (version 20170304:073404), however, the designers of Fruit v1 do not discuss the topic of distinguishing attacks at all, which might give potential users the impression that Fruit v1 is considered 80-bit secure (the general security level claimed in the abstract of the paper) against this type of attack. The designers of Plantlet, on the other hand, recognize the possibility of distinguishing attacks and discuss them as part of their cryptanalysis in [MAM17]. In particular, they point out the aforementioned distinguishing attack of Banik against Sprout and acknowledge that it would also be applicable to Plantlet (“*the memory complexity of this distinguishing attack against Plantlet is at least* 2^{58} *which is about 32,768 terabytes”* [MAM17]). However, they also state: “*This attack is possible due to the simplicity of the round key function. It would be possible to make the design resistant against this attack by either choosing a more complicated key-selection function or by further increasing internal state size.”* While increasing the size of the inner state would in fact be a valid counter measure (though it would somehow counteract the initial idea of having a smaller state), having a more complicated key schedule as suggested by the Plantlet designers would not help.

The fact that the designers of Fruit v1 do not discuss the topic of distinguishing attacks at all together with the above misconception of the Plantlet authors w.r.t. the effects of a more complicated key schedule seem to make it beneficial for future research in the direction of small-state stream ciphers to revisit the distinguishing attack by Englund, Hell, and Johansson and transfer it, along with a discussion of the now necessary assumptions, to this new application context. In particular, by analyzing those assumptions, we will then be able to come up with a new design idea for small-state stream ciphers, which finally delivers, what continuously using the secret key alone cannot: security against TMD tradeoff-based key recovery **and** distinguishing attacks.

Before we sketch the structure of our paper and continue with the details of the attacks, it should be mentioned that there is another recent lightweight stream cipher, which

also strives for beyond-the-birthday-bound security w.r.t. key recovery, but based on another design paradigm. Lizard [HKM17] was presented at FSE 2017 and implements the $FP(1)$-mode introduced in [HK15], which is a method for deriving the initial state from key and IV in a way that provides provable $\frac{2}{3}n$-security w.r.t. generic TMD tradeoff attacks aiming at key recovery. TMD tradeoff attacks aiming at recovering some initial state, on the other hand, are not hampered by the $FP(1)$-mode, even though they do not allow to straightforwardly derive the underlying secret key. As a consequence, not only for Sprout, Fruit v1, and Plantlet, but also for Lizard there are distinguishing attacks with a complexity below that of exhaustive key search. More precisely, the designers of Lizard claim 80-bit security against key recovery and 60-bit security against distinguishing (see [HKM17] for further details).

Structure of the paper: In section 2, we present a generic distinguishing attack against ciphers which continuously use the secret key, like Sprout, Fruit v1 and Plantlet. It is is based on a result by Englund, Hell and Johansson [EHJ07] from 2007 and also related to the distinguishing attack of Banik against Sprout [Ban15] from 2015. The generic nature of the attack will clarify that the security against distinguishing of ciphers which continuously use the secret key cannot be increased by choosing a more complicated key schedule. Instead, we will introduce a new design idea for small-state stream ciphers, which not only allows to achieve security against TMD tradeoff-based key recovery but also against distinguishing. In section 3, we demonstrate that the current version of Fruit (20170304:073404), which we denote Fruit v1, has at least 2^{64} weak keys, each of which does not provide the 80-bit security promised by the designers. The corresponding attack exploits the cipher's vulnerable key schedule and uses a variant of Babbage's TMD tradeoff attack [Bab95] to recover a subset of the unknown key bits together with a certain state of the feedback shift registers (FSRs) during state initialization. Based on this information, the cipher is then clocked back and the rest of the key bits is read from the FSRs at $t = 0$. Section 4 concludes the paper.

2 A Generic Distinguishing Attack against Stream Ciphers which Continuously use the Non-volatile Key

In this section, we present a generic distinguishing attack against stream ciphers which continuously use the non-volatile key. Our description is based on a result by Englund, Hell, and Johansson [EHJ07] from 2007 and hence also related to the distinguishing attack of Banik against Sprout [Ban15] from 2015.

2.1 Revisiting the Generic Attack of Englund, Hell, and Johansson

In 2007, Englund, Hell, and Johansson published a note [EHJ07], in which they presented a "*new distinguishing attack scenario for stream ciphers, allowing a resynchronization collision attack*" and pointed out that "*[t]he attack can succeed if the part of the state that depends on both the key and the IV is smaller than twice the key size.*" At the time of publication, the authors did not have ciphers like Sprout, Fruit v1, or Plantlet in mind (the first of which was only published in 2015), but they were targeting block ciphers in output feedback (OFB) mode. These can also be interpreted as (though not bitwise working) stream ciphers and, in fact, the eSTREAM phase-3 candidate LEX [Bir05] is very similar to a block cipher in OFB mode, so the attack of Englund, Hell, and Johansson could be applied.

The authors of [EHJ07] start the description of their idea as follows: "*Let us divide the internal state of the cipher into two parts, $State = (State_K, State_{K+IV})$, where $State_K$ is the part of the state that statically holds the key and $State_{K+IV}$ is the part of the state that*

is affected by both the key and the IV. Also, let N be the size of $State_{K+IV}$ in bits. If the key size $|K| > N/2$, then the new distinguishing attack will always succeed with complexity below exhaustive key search."

Note that this corresponds exactly to the scenario we are facing for Sprout, Fruit v1 and Plantlet.

The attack description in [EHJ07] then continues as follows: "*Consider a resynchronization scenario. We assume that the key is fixed and that the cipher is reinitialized using many different IVs. Furthermore, we assume that we have access to one long keystream sequence produced from one of the IVs, denoted IV_0. We then intercept many short ciphertext messages, each initialized using a different IV, and we assume that we know the first N plaintext bits in every ciphertext message. Our goal is to recover the rest of the plaintext for one of the messages.* [...] *We now proceed as follows. Let a keystream block be N consecutive bits in a considered keystream sequence. We first store $2^{N/2}$ different keystream blocks from the keystream generated from IV_0 in a sorted table. Thus, we have a table covering a fraction of $2^{-N/2}$ of all possible keystream blocks of length N. If the cipher is reinitialized with a new IV and we know the N first bits of the corresponding keystream, then the number of reinitializations needed in order to have a collision, i.e., receiving N bits that are present in the table, is geometrically distributed with expected value $2^{N/2}$. Denote the IV producing the collision by IV_c. If a collision is found, then with high probability the states are the same and the sequences following the colliding blocks of IV_0 and IV_c will be identical. That means that if we know only the first N keystream bits generated by IV_c, then we can predict future keystream bits from IV_c. In other words, by knowing only the first N plaintext bits of the message, we can decrypt the rest of the ciphertext without knowing the key."*

Depending on the application context and the attack complexity, the fact that the attacker is not only able to distinguish the cipher from a random bitstream but also obtains a potentially large amount of previously unknown keystream for the collision IV IV_c, might pose a serious security risk.

Also note that what this attack effectively does is to look for two different IVs which map to shifted versions of the same key stream. The same idea underlies, e.g., Banik's distinguishing attack against Sprout [Ban15].

However, the attack by Englund, Hell, and Johansson cannot be applied carelessly to stream ciphers like Sprout, Fruit v1, and Plantlet. The reason for this lies in their assumption that "[i]*f the cipher is reinitialized with a new IV and we know the N first bits of the corresponding keystream, then the number of reinitializations needed in order to have a collision, i.e., receiving N bits that are present in the table, is geometrically distributed with expected value $2^{N/2}$.*" This statement is obviously motivated by their application context of block ciphers in OFB mode. There, the IV serves as the initial state and thus, the IV space and the state of inner spaces have the same size. In fact, at a later point of the paper, the authors even write: "*This resynchronization scenario will not give an attack on a block cipher in counter mode.* [...] *However, the above scenario applies to block ciphers used in OFB mode since the output block z_i can be viewed as the part of the state that depends on both the key and the IV, $State_{K+IV}$. Similarly, the key used in the block cipher can be viewed as the part of the state that statically holds the key, $State_K$. By reinitializing the OFB mode stream cipher with a new IV, the cipher will enter a new random state after every encryption."* [EHJ07]

Note that for stream ciphers it is common to have IV sizes below the total size of the volatile inner state (see, e.g., Trivium and Grain, but also Sprout, Fruit v1, and Plantlet). Hence reinitializing with a new IV will only allow us to randomly draw elements from a **subset** of the set of all volatile inner states. In the next subsection, we will hence see that for a generic attack description against general stream ciphers, we will need additional assumptions.

Such assumptions are also necessary because, for a block cipher in OFB mode, the mapping of IVs to initial states is the identity and, hence, obviously bijective. While, under an arbitrarily fixed key, this is also the case for Sprout (cf. [Ban15]) and Plantlet (see Appenix A), it is not clear for Fruit v1 due to the fact that the IV is introduced 'from aside' during the state initialization. As a consequence, in our experiment, we would not draw **uniformly** at random from the set of all possible initial states as some of those states would be more likely due to the fact that they are the result of more than one IV.

Another important difference between a block cipher in OFB mode and common stream ciphers results from the fact that a block cipher realizes a permutation between plaintext blocks and ciphertext blocks. Hence, for a block cipher in OFB mode, the mapping of inner states (corresponding to the block cipher's plaintext blocks) to keystream blocks (corresponding to the block cipher's ciphertext blocks) is also a permutation. For common stream ciphers, however, we do not have this guarantee as different inner states of size n can actually lead to identical keystream blocks of size n. As a consequence, when straightforwardly applying the attack of Englund, Hell, and Johansson to ciphers like Sprout, Fruit v1, and Plantlet, we might actually experience keystream block collisions which are not the result of colliding inner states but of collisions in the output function. To avoid such 'false positives', we need to slightly increase the size of the keystream blocks which we save (for practical attacks, usually by only a few bits; see also subsection 2.3).

We will now describe a generic attack scheme for general stream ciphers, which takes the above considerations into account. In particular, explicitly specifying the required assumptions will enable us to come up with a new design idea for stream ciphers in subsection 2.4 that thwarts this type of distinguishing attack. More precisely, we extend the 2-tuple $State = (State_K, State_{K+IV})$ underlying the model of Englund, Hell, and Johansson (and also ciphers like Sprout, Fruit v1, and Plantlet) by a third component: $State_{IV}$.

2.2 A Generic Attack Scheme

Definition 2.1: Continuous-Key-Use (CKU) Stream Cipher

A CKU stream cipher is a classical, stepwise working keystream generator with the additional tweak that, even after the state initialization has been completed, the secret key is still used as an additional input to the state update function. The key schedule, which determines the way in which the secret key influences the state update, can depend on any part of the volatile inner state (FSRs, counters etc.) of the cipher.

Note that, e.g., Sprout, Fruit v1, and Plantlet are all CKU stream ciphers. Sprout's key schedule depends on a separate counter along with bits from the FSRs, while those of Fruit v1 and Plantlet only depend on a counter.

In the following definition, we do not not speak of any counters, but only of a *volatile inner state*, which we consider potential counters to be part of (along with the FSRs etc.). The reason for doing so is that we want to stay as generic as possible in our attack description in order to show its applicability for a wide range of CKU stream ciphers.[2]

[2] As mentioned in the introduction, the designers of Plantlet (in our opinion falsely) concluded that Banik's distinguishing attack against Sprout [Ban15] could have been avoided by using a more involved key schedule. With our generic attack scheme, we want to emphasize the general applicability of distinguishers of this type against such ciphers.

Definition 2.2

Let CIPHER be a CKU stream cipher with the following properties:

- n: size of the **volatile** inner state in bit;
- l: IV length in bit;
- 2^λ: limit of keystream bits per IV.

We define:

$$I_k = \left\{ \mathit{InitialState}_k(v); v \in \{0,1\}^l \right\},$$

where $\mathit{InitialState}_k(v) \in \{0,1\}^n$ denotes the initial state which the state initialization algorithm of CIPHER computes on the basis of the secret key k and the IV v.

For our generic distinguishing attack against CIPHER, we need the following two assumptions to hold.

Assumption 2.1: IV Near-Injectivity

CIPHER fulfills the *IV Near-Injectivity* assumption for the secret key k if $|I_k| \approx 2^l$.

Assumption 2.2: Initial State Randomness

Let $\sigma := \max\{0, (n/2 - \lambda)\}$, and let T be a set of about $2^{n/2}$ different $\tilde{n}$-bit keystream blocks (with $\tilde{n}$ slightly larger than n) that were obtained on the basis of $\lceil 2^\sigma \rceil$ different IVs under an arbitrarily fixed key k by sliding an $\tilde{n}$-bit window over each of the respective $\lceil 2^\sigma \rceil$ keystreams of length $\leq 2^\lambda$ bit without experiencing any collisions. Furthermore, let $S_k(T)$ denote the set of volatile inner states which underlie the keystream blocks in T for the secret key k.

CIPHER fulfills the *Initial State Randomness* assumption if for such sets T, it holds with high probability that:

$$\frac{|S_k(T) \cap I_k|}{|S_k(T)|} \gtrapprox \frac{|I_k|}{|\{0,1\}^n|}.$$

Note that if a stream cipher violates Assumption 2.1, this in itself opens the door for a distinguishing attack that looks for IVs which produce identical key streams. For example, if the IV size is smaller than the key size, even a single such IV collision allows for a distinguishing attack with complexity lower than that of exhaustive key search.[3]

Assumption 2.2 ensures that there will actually be a sufficient amount of collision candidates to look for. Banik makes a similar assumption in his distinguishing attack against Sprout in [Ban15] by expecting that for a randomly chosen inner state, there is a probability of 2^{-10} that an IV exists for which this state is "*the 80th keystream phase state*". In subsection 2.4, we will introduce a new design idea for stream ciphers, which thwarts such distinguishing attacks by ensuring that two IVs will never lead to shifted versions of the same keystream. But first, we will now describe the generic distinguishing attack against CKU stream ciphers and explain its consequences for Fruit v1 and Plantlet in subsection 2.3.

[3] A small amount of IV collisions may be tolerable depending on the security claims; see, e.g., LIZARD [HKM17], which claims 80-bit security against key recovery and 60-bit security against distinguishing.

So let Cipher be a CKU cipher with an $\tilde{n}$-bit volatile inner state, an IV length of l bit, and a limit of 2^λ keystream bits per IV. Moreover, let Cipher fulfill assumptions 2.1 and 2.2, and let $\sigma := \max\{0, (n/2 - \lambda)\}$. Then the following algorithm allows to distinguish the keystream produced by Cipher from a random bitstream with high probability:

1. Obtain about $2^{n/2}$ $\tilde{n}$-bit keystream blocks (with $\tilde{n}$ slightly larger than n) based on $\lceil 2^\sigma \rceil$ different IVs from the oracle by sliding an $\tilde{n}$-bit window over each of the respective $\lceil 2^\sigma \rceil$ keystreams of length $\leq 2^\lambda$ bit and save the keystream blocks in an efficiently searchable data structure like a hash table. If during this step, a collision occurs, we can already distinguish Cipher and stop.

2. For $2^{n/2}$ different IVs, obtain the corresponding $\tilde{n}$-bit keystream prefix from the oracle and look for a collision in the data structure created in *Step (1)*. Once such a collision is found, we can distinguish Cipher and stop.

The success probability of the attack can be derived from the birthday paradox. Under an arbitrarily fixed key k, the universe of the corresponding experiment will be the set of all possible initial states I_k, which, according to Assumption 2.1, has size $|I_k| \approx 2^l$. Assumption 2.2 ensures that, w.h.p., a subset of size about $2^{n/2-(n-l)} = 2^{l-n/2}$ of the volatile inner states underlying the $2^{n/2}$ keystream blocks collected in *Step (1)* will belong to the set I_k. Assumption 2.1 guarantees that in *Step (2)*, we draw uniformly and at random $2^{n/2}$ elements from I_k. According to the birthday paradox, when drawing uniformly and at random $2^{n/2}$ elements from a set of size 2^l, we are likely to find a collision with an arbitrarily fixed subset of size $2^{l-n/2}$.

The complexity of the above attack is:

(1) Obtain $2^{n/2}$ keystream blocks of size $\tilde{n}$ bit and store them in an efficiently searchable data structure:

- Data complexity (keystream): $2^{n/2}$;
- Memory complexity (keystream blocks): $2^{n/2} \cdot \tilde{n}$;
- Time complexity: $2^{n/2}$.

(2) Obtain $2^{n/2}$ keystream prefixes of size $\tilde{n}$ and search for a collision in the data structure created in *Step (1)*.

- Data complexity (keystream prefixes): $2^{n/2} \cdot \tilde{n}$;
- Memory complexity: negligible;
- Time complexity: $2^{n/2}$.

Note that, for the time complexity, we consider the computation of each $\tilde{n}$-bit keystream block to be an atomic operation, independent of whether it is a keystream prefix or a block that appears in the middle of a keystream. This is motivated by the comparison to the complexity of exhaustive key search, where, per key candidate, a keystream prefix needs to be generated and compared to the given keystream prefix for the secret key. In consequence, *Step (1)* and *Step (2)* each have a time complexity of $2^{n/2}$. (For *Step (1)*, where we slide an $\tilde{n}$-bit window over the keystream, we actually need less encryption operations when we assume that each atomic encryption operation gives $\tilde{n}$ new keystream bits. However, inserting the $2^{n/2}$ keystream blocks into the hash table takes $2^{n/2} \cdot O(1)$ time.)

So the overall complexity of the generic TMD tradeoff distinguishing attack against CKU stream ciphers is dominated by data and memory complexity, which are both about $2^{n/2} \cdot \tilde{n}$. Hence, if the key size of the attacked cipher is larger than $(n/2) + log_2(\tilde{n})$, we have a distinguishing attack complexity below the complexity of exhaustive key search.[4]

[4]Englund, Hell, and Johansson similarly concluded: "*If the key size* $|K| > N/2$*, then the new*

2.3 Applying the Attack to Plantlet and Fruit v1

In this subsection, we will discuss the consequences of the above distinguishing attack against the ciphers Plantlet and Fruit v1. Sprout will not be treated due to the respective distinguishing result already presented by Banik [Ban15]. The reason for including Plantlet despite the fact that the designers already acknowledged that Banik's attack against Sprout could also be transferred to Plantlet is that they point out at the same time: "*It would be possible to make the design resistant against this attack by either choosing a more complicated key-selection function or by further increasing internal state size.*" [MAM17] We will show that the above attack is completely independent of how complicated the key-selection function is. Fruit v1, on the other hand, is discussed as the designers do not mention the possibility of distinguishing attacks at all in their paper, which might give the impression that the security level against distinguishing is supposed to equal that against key recovery via exhaustive key search. We will show that this is not the case.

The main question when applying our generic attack scheme is obviously whether the targeted cipher fulfills the necessary assumptions 2.1 (*IV Near-Injectivity*) and 2.2 (*Initial State Randomness*). In Appendix A, we show for Plantlet that, under an arbitrarily fixed key, the mapping of IVs to the corresponding volatile initial state is injective; hence, Assumption 2.1 is fulfilled. For Fruit v1, we will also suppose Assumption 2.1 to be fulfilled, despite the fact that here, IV collisions could actually occur. However, as pointed out above, the existence of a large number of IV collisions would constitute a weakness on its own. In particular, as the IV space of Fruit has only size 2^{70} whereas the key size is 80 bit, the existence of even a single IV collision would be sufficient to launch a distinguishing attack with complexity below that of exhaustive key search by exhaustively searching the complete IV space for this collision.

The applicability of Assumption 2.2, on the other hand, can neither be proved for Plantlet nor for Fruit v1. Instead, we have to refer to a plausibility argument based on the structure of the respective cipher (similarly to what, e.g., Banik did for Sprout in [Ban15]):

- **Plantlet**: The IV space of Plantlet has size 2^{90} and the corresponding mapping (under an arbitrarily fixed key) to the set of initial states is injective (see Appendix A). As two bits of the 9-bit counter will not be used after the state initialization is completed, the relevant volatile inner state of Plantlet has size $61 + 40 + 7 = 108$ bit. From the cipher definition, we know that the remaining 7-bit counter will have the same binary value $0 \dots 0$ for all **initial** states. During the keystream generation, this counter, interpreted as a natural number, stepwise takes all values *mod 80*. In particular, in every 80th clock cycle, the counter takes binary value $0 \dots 0$. Based on general security assumptions for stream ciphers, we can also expect that the FSR states will evolve randomly during the keystream generation. As a consequence, every time the counter takes value $0 \dots 0$, we have a chance of 2^{90-101} (101 bit is the combined size of the FSRs) that there is an IV which has this inner state as an initial state. So when picking an arbitrary $(108 + \epsilon)$-bit keystream block, we have a chance of $80^{-1} \cdot 2^{90-101} > 2^{-18}$ that that underlying 108-bit inner state is also an initial state for some IV under the given key. This implies that Assumption 2.2 holds.

- **Fruit v1**: The IV space of Fruit has size 2^{70}. Due to the way the IV is introduced to the inner state, there is a possibility for IV collisions leading to the same initial state. However, as explained above, for Fruit v1 we will also suppose Assumption 2.1 to be fulfilled. So let 2^{70} be the approximate size of the set of initial states that are actually possible under the given secret key. Like Plantlet, Fruit v1 has a (7-bit wide)

distinguishing attack will always succeed with complexity below exhaustive key search." However, they left out the logarithmic factor, which we chose to include as exhaustive key search has negligible data and memory complexity, whereas in the above distinguishing attack, these complexities are actually each at a factor of $\tilde{n}$ higher than the time complexity and dominate the overall cost of the attack.

counter that influences the key bit selection. The volatile inner state of Fruit v1 has size $37 + 43 + 7 = 87$ bit. As in the case of Plantlet, the 7-bit counter is stepwise incremented during keystream generation. However, there are two differences: (1) the counter will actually cycle through all possible 2^7 values; (2) the counter value of the initial states will not be known as it is set based on FSR values during the state initialization. We know, however, that each initial state has the property that, when clocked back 80 times (the respective state transition function is bijective), a certain position of the LFSR has to be 1 and six positions of the NFSR have to equal six positions of the counter. One in 2^7 inner states of size 87 bit will fulfill this property. Using the same security argument about FSR state randomness as in the case of Plantlet, we can hence expect that when picking an arbitrary $(87 + \epsilon)$-bit keystream block, we have a chance of about $2^{-7} \cdot \left(2^{70-80}\right) = 2^{-17}$ that that underlying 87-bit inner state is also an initial state for some IV under the given key. This implies that Assumption 2.2 holds.

In order to substantiate our assumption w.r.t. the randomly evolving FSR states during keystream generation and the corresponding implications on the existence of a sufficient amount of collision candidates, we created a reduced (i.e., 'halved') version of Fruit v1, which we call *Shrunk Fruit v1* (see Appendix B for a specification). Shrunk Fruit v1 has a volatile inner state of size 46 bit (19-bit NFSR, 21-bit LFSR, 6-bit counter) and is operated with 40-bit keys and 35-bit IVs. Assuming that Shrunk Fruit v1 fulfills assumptions 2.1 and 2.2, the distinguishing attack described in the previous subsection should find a collision pair based on $2^{46/2}$ keystream **blocks** collected in *Step (1)* and $2^{46/2}$ keystream **prefixes** requested in *Step (2)* with high probability. We created a corresponding simulation with the computer algebra system Magma. In each iteration of the experiment, a random key was chosen and 2^{23} 50-bit keystream blocks were generated and stored as described in *Step (1)* of the above distinguishing algorithm. Then, in line with *Step (2)*, 50-bit keystream prefixes were computed for randomly chosen IVs until the first collision with the set generated in *Step (1)* occurred. We performed 25 such iterations and the average number of trials needed in *Step (2)* was $2^{22.3}$, which is in line with our theoretical results. The simulations also showed that as few as 4 additional bits per keystream block (i.e., 50 instead of 46) were sufficient to avoid false positives, i.e., keystream block collisions caused by collisions in the output function instead of by colliding inner states.

The complexities (*Step (1)* + *Step (2)*) of the above distinguishing attack when applied to Plantlet and Fruit v1 are as follows:

- **Plantlet**:
  - Data complexity: $2^{108/2} + 2^{108/2} \cdot (108 + 20) \approx 2^{61}$;
  - Memory complexity: $2^{108/2} \cdot (108 + 20) = 2^{61}$;
  - Time complexity: $2^{108/2} + 2^{108/2} = 2^{55}$.
- **Fruit v1**:
  - Data complexity: $2^{87/2} + 2^{87/2} \cdot (87 + 20) \approx 2^{50.2}$;
  - Memory complexity: $2^{87/2} \cdot (87 + 20) \approx 2^{50.2}$;
  - Time complexity: $2^{87/2} + 2^{87/2} = 2^{44.5}$.

In both cases, we added a 20-bit security margin to the block size in order to avoid false positives as explained in subsection 2.1. 20 bit are probably way to much as our experiments with Shrunk Fruit v1 indicate (see above), by the effects on the attack complexity are minimal anyhow, so we preferred to be on the safe side with our claims.

Note that for Plantlet, complexities could easily be improved by taking the cipher specifics into account. For example, the attacker actually knows which of the keystream

blocks collected in *Step (1)* are impossible to produce a collision (because he knows the underlying counter values). These values would not have to be stored in the first place, reducing the memory complexity of the attack. Moreover, the above attack complexities are based on the assumption that all values can occur in the counter register. For Plantlet, this is actually not the case, which would further reduce the attack complexity. However, as pointed out before, we wanted to have a scheme for distinguishing attacks against CKU ciphers which is as generic as possible.

Thereby, we hope to clarify that a more complicated key schedule (e.g., by having a nonlinear round key function, making use of FSR bits etc.) will not protect against the above distinguishing attack. So while continuously using the secret key may protect against TMD tradeoff **inner state recovery** attacks, it does not provide security (equal to the security against key recovery) against TMD tradeoff **distinguishing** attacks.

A trivial way to thwart the above distinguishing attacks would be to increase the size of the volatile inner state. However, this would clearly counteract the initial idea of having a smaller state. In the following subsection, we will introduce a new design idea, which can provide full security against distinguishing attacks without increasing the volatile inner state.

2.4 New Design Idea: Stream Ciphers which Continuously use the IV

The assumption which underlies the applicability of the above (and also, e.g., Banik's [Ban15]) distinguishing attack against CKU ciphers like Sprout, Fruit v1, and Plantlet, is that it is possible to find two IVs which, under an arbitrarily fixed key, lead to shifted versions of the same keystream. In this subsection, we suggest a potential countermeasure, which ensures that, under an arbitrarily fixed key, any two IVs will always map to different keystreams.

The prominent innovation of Sprout [AM15] was to continuously (i.e., also during keystream generation) use the secret key as part of the state update in order to protect against TMD tradeoff **inner state recovery** attacks. Our suggestion is now to also continuously use the public IV as part of the state update in order to protect against TMD tradeoff **distinguishing** attacks. By doing so, the public IV would become part of the inner state, just like the secret key became part of the inner state in Sprout.

Now one may argue from a hardware perspective that, while the secret key has to be stored anyhow (e.g., also for Trivium, Grain etc.) in order to be reused with other IVs, this would not be the case for the IV. Hence, at first sight, assuming that the IV is still accessible after state initialization might be considered cheating.

However, we do not think that this is the case for many application scenarios. Look, for example, at A5/1. There, the IV used in the encryption of a data packet is the respective (sequentially incremented) 22-bit frame number. Hence, any A5/1 device needs some memory containing this frame number anyhow. Generally, especially for ciphers with small IV spaces, there always has to be a mechanism like a stepwise incremented IV register to make sure that the same IV is not accidentally used twice under the same secret key. Similarly, in all communication scenarios like A5/1, where the packet counter serves at the same time as an IV source, we will always have this information.

Actually, any stream cipher definition which strictly requires "Never use the same IV twice under a single key!" also needs a mechanism to enforce this requirement.[5] Independent of whether this mechanism is to keep a stepwise incremented IV in a writable storage location like an EEPROM or another register on the device, or whether the device actually keeps a table of already used IVs, there will always be a source where a cipher can continuously get the current IV from.

[5] Though stream cipher designers seem to hardly talk about this issue in their suggestions but usually leave the problem of IV uniqueness to user.

In this paper, we will not suggest a specific instantiation of such a cipher, which continuously uses the secret key **and** the IV, but leave the development to future work. However, it might be as easy as changing the "*round key function*" of Plantlet [MAM17] from

$$\tilde{k}^t = k_{(t \bmod 80)},\ t \geq 0$$

to

$$\tilde{k}^t = k_{(t \bmod 80)} + iv_{(t \bmod 90)},\ t \geq 0.$$

As a final note, we would like to point out that our new design idea of continuously using the IV as part of the state update is mainly targeted at CKU ciphers. Other small-state ciphers like Lizard [HKM17] would hardly benefit for the following reason: If the secret key is not used after the state initialization, there is always the possibility of a TMD tradeoff inner state recovery attack like those by Babbage [Bab95] or Biryukov and Shamir [BS00]. Such an attack will have complexity half the size of the volatile inner state, independent of whether the IV is continuously used during state update, because the IV is public and the attacker will be able to evaluate the function $(\textit{volatile inner state}, IV) \rightarrow \textit{keystream block}$ for randomly chosen volatile inner states and the proper IV. The only advantage of continuously using the IV would be that TMD tradeoff precomputations could be prevented as, if it is not a chosen-IV attack scenario, the attacker would have to wait for which IV he actually obtains the required/attacked keystream.

3 A Key Recovery Attack against Fruit v1

3.1 Description

The following attack against Fruit v1 (our name for the most recent version of Fruit; cf. Sec. 1 for further remarks) works for the subset $\{k_0, \ldots, k_{63}, 0, \ldots, 0\} \subset \{0,1\}^{80}$ (i.e., for every 65536th key), for which it allows key recovery faster than exhaustive key search (i.e., with complexity $< 2^{64}$) against the remaining bits $k_0, \ldots, k_{63}$. In other words: Fruit has (at least) 2^{64} weak keys, which do not offer the promised 80-bit security.

Observation 3.1

The key schedule of Fruit v1 has the property that if $k_{64} = k_{65} = \ldots = k_{79} = 0$ holds, then the key schedule bit k'_t is always computed as $k'_t = k_{q+32}$, where $0 \leq q \leq 31$ and q depends on t. In particular, this means that if $k_{64} = k_{65} = \ldots = k_{79} = 0$ holds, then the key bits $k_0, \ldots, k_{31}$ are never involved in the computation of k'_t.

So let us suppose that $k_{64} = k_{65} = \ldots = k_{79} = 0$ holds. We will use a variant of Babbage's TMD tradeoff attack to recover the inner state (composed of the 80 bits of the FSRs **and** the 32 key bits $k_{32}, \ldots, k_{63}$, i.e., 112 bits in total) at $t = 130$ **before** the counter values and the value of l_{130} are overwritten. Observe that if we know this inner state and the IV underlying it, then we will be able to clock the cipher back and recover the inner state at $t = 0$, which contains the full 80-bit key. Also note that the counter value at $t = 130$ **before** it is overwritten is always publicly known.

Attack algorithm:

1. For 2^{52} different IVs obtain the 112-bit keystream prefix from the oracle and save the tuples (keystream prefix, IV) in an appropriate (i.e., efficiently searchable with respect to the keystream prefixes) data structure like a hash table.

2. For 2^{60} random choices of elements

$$((k^*_{32},\ldots,k^*_{63}),(l^*_{130},\ldots,l^*_{172}),(n^*_{130},\ldots,n^*_{166})) \in \{0,1\}^{112},$$

evaluate the function $F : \{0,1\}^{112} \to \{0,1\}^{112}$, which computes the first 112 keystream bits based on this random KSG state at $t = 130$, and look for a collision in the data structure created in *Step (1)*.[6] Once a collision is found, clock the cipher back based on the discovered (IV, inner state) pair and obtain the full secret key as the FSR contents at $t = 0$.

Theorem 3.1

According to the birthday paradox, with high probability, we'll find a collision between one of the keystream prefixes from *Step (1)* with a keystream prefix from *Step (2)*.

Proof. See subsection 3.2. □

The complexity of the above attack is:

(1) Get 2^{52} keystream prefixes of size 112 bit and store them, together with the respective IV of size 70 bit, in an efficiently searchable data structure:

- Time complexity: 2^{52};
- Data complexity (keystream prefixes): $2^{52} \cdot 112 \approx 2^{58.8}$;
- Memory complexity ((keystream prefix, IV) tuples):

$$2^{52} \cdot (112 + 70) \approx 2^{59.5}.$$

(2) Evaluate the function F 2^{60} times (we'll consider this as 2^{60} atomic operations, just like each encryption under a different key during exhaustive key search is usually treated as an atomic operation; actually, we are even faster as we do not need to perform the first 130 clocks of the initialization):

- Time complexity: 2^{60} keystream prefix generations.

Obviously, for the tradeoff parameters we chose (i.e., collecting 2^{52} keystream prefixes during *Step (1)* and performing 2^{60} evaluations of F for different random inputs in *Step (2)*), the overall attack cost is dominated by the time complexity 2^{60}. (Note that we needed to get below 2^{64} to show that for the targeted subset of 2^{64} keys, the remaining unknown 64 key bits can be recovered with complexity below exhaustive key search, i.e., below 2^{64}.)

Final remarks:

- For the related ciphers Grain and Plantlet, it can be shown that, under an arbitrarily fixed key, the mapping of IVs to initial states is injective. Due to the way the IV is introduced (i.e., 'from aside') in Fruit, it is not clear to what extend this property actually holds here. This leads to the situation that during *Step (1)* of the algorithm, we could need some more attempts to arrive at a subset of 2^{52} different keystream prefixes corresponding to a set S, $|S| = 2^{52}$, of different inner states as explained

[6] Note that the 112-bit input $((k^*_{32},\ldots,k^*_{63}),(l^*_{130},\ldots,l^*_{172}),(n^*_{130},\ldots,n^*_{166}))$ contains all necessary information to compute this keystream prefix, because: the counter at $t = 130$ before it is overwritten is publicly known, and we suppose the last 16 key bits to be 0, and the first 32 key bits are never needed again for the state update (and the keystream generation) after $t = 0$.

above. This, however, is not a problem as we could collect up to 2^{57} keystream prefixes during *Step (1)* and still stay below the overall target complexity 2^{64} of exhaustive key search against the bits $k_0, \ldots, k_{63}$ (note that only the data complexity would be increased as we only save different keystream prefixes, which leaves the memory complexity unchanged). If a set of 2^{57} randomly chosen different IVs would be mapped to fewer than 2^{52} different initial states under an arbitrarily fixed key, this would obviously constitute a massive weakness on its own (e.g., with respect to distinguishing attacks and the fact that non-negligible fractions of IVs would be mapped to identical keystrems).

- In an older version of Fruit ([GHX16], version 20161124:115414), the design description actually contained an upper bound (2^{15}) on the number of IVs that should be used under the same secret key. In the current version of the ePrint paper ([GHX16], version 20170304:073404), which underlies what we call Fruit v1 here, this restriction has been dropped.[7] Nonetheless, we would like to point out that the above attack could be easily adapted to a scenario with this restriction still in place. Instead of targeting the inner state at $t = 130$, one could also target an arbitrary inner state during keystream generation. However, in this case the counter would not be known any longer and it (more precisely, only the six bits $c_t^1, \ldots, c_t^6$) would have to be included in our TMD tradeoff attack. This would raise the data, memory and time complexities each by a factor of at most 2^3 (the counter is also 'halved' via the birthday paradox), which would still fall in the boundaries of a successful attack.[8] In *Step (1)* of the attack, one would then obtain 2^{55} 118-bit keystream blocks based on 2^{12} keystreams each of size 2^{43} bit (the limit set by the designers of Fruit v1) for 2^{12} different IVs. In *Step (2)*, one would pick 2^{63} random inputs for a modified function $F : \{0,1\}^{118} \rightarrow \{0,1\}^{118}$. The rest of the attack then works analogously to what we described before.

- An easy way to thwart our key recovery attack against Fruit v1 would obviously be to add the linear term '$\oplus\, k_s$', with s cyclically taking the values $0, \ldots, 31$, to the cipher's round key function. However, the question remains in what sense (i.e., w.r.t. which kind of attacks) a more complicated round key function like that of Fruit v1 is actually supposed to be superior to the basic one used in the yet unbroken Plantlet.

3.2 Proof of Theorem 3.1

In *Step (1)*, we collect keystream prefixes which correspond to a subset S of size 2^{52} of the set $\Omega = \{0,1\}^{32} \times \{0,1\}^{43} \times \{0,1\}^{37}$ of size 2^{112}. More precisely, the 112-bit inner states underlying the keystream prefixes collected in *Step (1)* all belong to the same secret key. In particular, this implies that S has a special structure:

$$\begin{aligned}
&\left(\left(\tilde{k}_{32}, \ldots, \tilde{k}_{63}\right), \left(\tilde{l}_{130}, \ldots, \tilde{l}_{172}\right), \left(\tilde{n}_{130}, \ldots, \tilde{n}_{166}\right)\right) \in S \\
&\text{and } \left(\left(\hat{k}_{32}, \ldots, \hat{k}_{63}\right), \left(\hat{l}_{130}, \ldots, \hat{l}_{172}\right), \left(\hat{n}_{130}, \ldots, \hat{n}_{166}\right)\right) \in S \\
&\Rightarrow \tilde{k}_i = \hat{k}_i \text{ for } i = 32, \ldots, 63.
\end{aligned}$$

However, we will see that the structure of S is completely irrelevant for the success probability of our attack. In particular, S does not have to be a random subset of Ω. The

[7] In fact, limiting a cipher that is designed for fixed-key scenarios (due to the continuous use of non-sequential key bits) to 2^{15} IVs per secret key, seems rather unrealistic as this would effectively mean that a corresponding RFID tag would have to be dumped after at most 32768 keystream generations.

[8] In fact, e.g. the data complexity would be increased by a factor below 2^3 (and possibly even decrease) as now, the keystream blocks can be derived via sliding a 118-bit window over the keystream, just like in the classical TMD tradeoff attack.

only important aspect is that during *Step (2)*, we are able to randomly select elements from the whole of Ω (which is the case in our attack as we are free to choose the inputs for $F : \{0,1\}^{112} \to \{0,1\}^{112}$; see above).

So let S with $|S| = 2^{52}$ be an arbitrarily fixed subset of Ω with $|\Omega| = 2^{112}$. The probability of finding a collision during *Step (2)* can be computed as $1 - p$, where p denotes the probability that the 2^{60} elements we draw during *Step (2)* all belong to the set $\Omega \setminus S$. For simplicity, we assume that our algorithm is 'stupid' and does not remember which elements it has already drawn uniformly and at random during *Step (2)*, i.e., the same non-collision elements in $\Omega \setminus S$ can be drawn multiple times during *Step (2)*. (This assumption is obviously a disadvantage for the attacker.) Then we get

$$p = \left(\frac{2^{112} - 2^{52}}{2^{112}}\right)^{2^{60}} = \left(1 - \frac{1}{2^{60}}\right)^{2^{60}} \approx e^{-1},$$

which proves our Theorem as $1 - p \approx 0.63$. □

4 Conclusion

In this paper, we presented a generic distinguishing attack against ciphers which continuously use the secret key, like Sprout, Fruit v1 and Plantlet. It is based on a result by Englund, Hell and Johansson [EHJ07] from 2007 and also related to the distinguishing attack of Banik against Sprout [Ban15] from 2015. The generic nature of the attack helps to clarify that the security against distinguishing of ciphers which continuously use the secret key cannot be increased by choosing a more complicated key schedule. Instead, we introduced a new design idea for small-state stream ciphers, which not only allows to achieve security against TMD tradeoff-based key recovery but also against distinguishing. Furthermore, we demonstrated that the current version of Fruit (20170304:073404), which we denoted Fruit v1, has at least 2^{64} weak keys, each of which does not provide the 80-bit security promised by the designers. The corresponding attack exploits the cipher's vulnerable key schedule and uses a variant of Babbage's TMD tradeoff attack [Bab95] to recover a subset of the unknown key bits together with a certain state of the feedback shift registers during state initialization. Based on this information, the cipher is then clocked back and the rest of the key bits is read from the FSRs at $t = 0$.

In general, our paper shows that the search for small-state stream ciphers which completely 'defeat' the birthday bound, is far from being finished. The initial hope that continuously using the secret key would fully solve this problem has been shattered by TMD tradeoff distinguishing attacks. While the new design principle of continuously using key **and** IV, which we introduced in this paper, might actually lead to ciphers that resist TMD tradeoff key recovery **and** distinguishing attacks, it is obvious that the necessary hardware conditions will not be present in all application scenarios. Hence, the search for alternative solutions remains a field which is not only interesting from a theoretical point of view, but also of actual practical relevance. The results presented in this paper seem to indicate, that a more complicated key schedule (as used by Fruit v1) will probably not be the way to go.

References

[AM15] Frederik Armknecht and Vasily Mikhalev. On Lightweight Stream Ciphers with Shorter Internal States. In Gregor Leander, editor, *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 451–470. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[Bab95] S.H. Babbage. Improved "exhaustive search" attacks on stream ciphers. In *Security and Detection, 1995., European Convention on*, pages 161–166, May 1995.

[Ban15] Subhadeep Banik. Some Results on Sprout. In Alex Biryukov and Vipul Goyal, editors, *Progress in Cryptology – INDOCRYPT 2015: 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, pages 124–139. Springer International Publishing, Cham, 2015.

[BB06] Elad Barkan and Eli Biham. Conditional Estimators: An Effective Attack on A5/1. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[BD06] Steve Babbage and Matthew Dodd. The stream cipher MICKEY 2.0. eSTREAM: the ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf.

[BGW99] Marc Briceno, Ian Goldberg, and David Wagner. A pedagogical implementation of A5/1, 1999. Available at http://www.scard.org/gsm/a51.html.

[Bir05] Alex Biryukov. LEX. eSTREAM: the ECRYPT Stream Cipher Project, 2005. hhttp://www.ecrypt.eu.org/stream/lexp3.html.

[BS00] Alex Biryukov and Adi Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings*, pages 1–13. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[CP05] Christophe De Cannière and Bart Preneel. Trivium – Specifications. eSTREAM: the ECRYPT Stream Cipher Project, 2005. http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.

[ECR08] ECRYPT – European Network of Excellence for Cryptology. eSTREAM: the ECRYPT stream cipher project, 2008. http://www.ecrypt.eu.org/stream/.

[EHJ07] H. Englund, M. Hell, and T. Johansson. A Note on Distinguishing Attacks. In *2007 IEEE Information Theory Workshop on Information Theory for Wireless Networks*, pages 1–4, July 2007.

[EK16] Muhammed F. Esgin and Orhun Kara. Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015: 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 67–85. Springer International Publishing, Cham, 2016.

[FMS01] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers*, pages 1–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[GHX16] Vahid Amin Ghafari, Honggang Hu, and Chengxin Xie. Fruit: Ultra-Lightweight Stream Cipher with Shorter Internal State. Cryptology ePrint Archive, Report 2016/355, 2016. http://eprint.iacr.org/2016/355.

[HJM06] Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM: the ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf.

[HK15] Matthias Hamann and Matthias Krause. On Stream Ciphers with Provable Beyond-the-Birthday-Bound Security against Time-Memory-Data Tradeoff Attacks. Cryptology ePrint Archive, Report 2015/636, 2015. http://eprint.iacr.org/2015/636.

[HKM17] Matthias Hamann, Matthias Krause, and Willi Meier. LIZARD – A Lightweight Stream Cipher for Power-constrained Devices. *IACR Transactions on Symmetric Cryptology*, 2017(1):45–79, 2017.

[Ins97] Institute of Electrical and Electronics Engineers. IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-1997*, pages i–445, 1997.

[Ins04] Institute of Electrical and Electronics Engineers. IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Amendment 6: Medium Access Control (MAC) Security Enhancements. *IEEE Std 802.11i-2004*, pages 1–190, July 2004.

[LMV05] Yi Lu, Willi Meier, and Serge Vaudenay. The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005. Proceedings*, pages 97–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[LNP15] Virginie Lallemand and María Naya-Plasencia. Cryptanalysis of Full Sprout. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 663–682. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[MAM17] Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On Ciphers that Continuously Access the Non-Volatile Key. *IACR Transactions on Symmetric Cryptology*, 2016(2):52–79, 2017.

[Pop15] A. Popov. Prohibiting RC4 Cipher Suites. RFC 7465 (Proposed Standard), February 2015.

[Sch95] Bruce Schneier. *Applied Cryptography (2nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[SIG14] Bluetooth SIG. Bluetooth Core Specification 4.2, 2014. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439.

[ZG15] Bin Zhang and Xinxin Gong. Another Tradeoff Attack on Sprout-Like Stream Ciphers. In Tetsu Iwata and Hee Jung Cheon, editors, *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*, pages 561–585. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

A Plantlet: Injectivity of IV → Initial State

The following algorithm computes (under an arbitrarily fixed key) for any given FSR content at $t = 320$ a corresponding FSR content at $t = 0$, which, according to the Plantlet algorithm, would lead to the given FSR content at $t = 320$. (Moreover, it returns the IV contained in the FSRs at $t = 0$.) This shows that the corresponding mapping of FSR contents at $t = 0$ to FSR contents at $t = 320$ under an arbitrarily fixed key is surjective. As domain and codomain have the same size, this also implies injectivity and shows that, under an arbitrarily fixed key, different IVs will always be mapped to different initial states by Plantlet's state initialization algorithm.

Given:

- 80-bit Key: $(k_0, \ldots, k_{79})$
- 60-bit LFSR part of initial state: $\left(l_0^{320}, \ldots, l_{59}^{320}\right)$
  Note that l_{60} is irrelevant for us as it is not used during initialization.
- 40-bit NFSR part of initial state: $\left(n_0^{320}, \ldots, n_{39}^{320}\right)$
- $\left(c_t^0, \ldots, c_t^8\right)$ is known for all $t = 0, \ldots$
  This is particularly important for c_4^t. (see below)

Algorithm:

for $t = 319$ down to 0 **do**
  for $i = 0$ to 58 **do**
    $l_{i+1}^t \leftarrow l_i^{t+1}$
  end for
  for $i = 0$ to 38 **do**
    $n_{i+1}^t \leftarrow n_i^{t+1}$
  end for
  $(x_0^t, \ldots, x_8^t) \leftarrow (n_4^t, l_6^t, l_8^t, l_{10}^t, l_{32}^t, l_{17}^t, l_{19}^t, l_{23}^t, n_{38}^t)$
  $h^t \leftarrow x_0^t \cdot x_1^t \oplus x_2^t \cdot x_3^t \oplus x_4^t \cdot x_5^t \oplus x_6^t \cdot x_7^t \oplus x_0^t \cdot x_4^t \cdot x_8^t$
  $z^t \leftarrow h^t \oplus l_{30}^t \oplus (n_1^t \oplus n_6^t \oplus n_{15}^t \oplus n_{17}^t \oplus n_{23}^t \oplus n_{28}^t \oplus n_{34}^t)$
  $l_0^t \leftarrow \oplus l_{59}^{t+1} \oplus l_{54}^t \oplus l_{43}^t \oplus l_{34}^t \oplus l_{20}^t \oplus l_{14}^t \oplus z^t$
  $\tilde{k}^t = k_{(t \bmod 80)}$
  $n_0^t \leftarrow n_{39}^{t+1} \oplus \tilde{k}^t \oplus l_0^t \oplus c_4^t \oplus n_2^t \cdot n_{25}^t \oplus n_3^t \cdot n_5^t \oplus n_7^t \cdot n_8^t \oplus n_{14}^t \cdot n_{21}^t \oplus n_{16}^t \cdot n_{18}^t \oplus n_{22}^t \cdot n_{24}^t \oplus n_{26}^t \cdot n_{32}^t \oplus n_{33}^t \cdot n_{36}^t \cdot n_{37}^t \cdot n_{38}^t \oplus n_{10}^t \cdot n_{11}^t \cdot n_{12}^t \oplus n_{27}^t \cdot n_{30}^t \cdot n_{31}^t$
end for
for $i = 0$ to 39 **do**
  $iv_i \leftarrow n_i^0$
end for
for $i = 40$ to 89 **do**
  $iv_i \leftarrow l_{i-40}^0$
end for
  return $(iv_0, \ldots, iv_{89})$

B Shrunk Fruit v1

- 40-bit Key $(k_0 \dots k_{39})$, 35-bit IV $(v_0 \dots v_{34})$
- Keystream limit per IV: 2^{21} bit (due to the 21-bit MLLFSR; corresponding to a limit of 2^{43} bit and a 43-bit MLLFSR in Fruit v1)
- 6-Bit Counter: $Cr = \left(c_t^1 c_t^2 c_t^3 c_t^4 c_t^5 c_t^6\right)$
- Key Schedule:
  $k'_t = k_s \cdot k_{y+32} \oplus k_{u+36} \cdot k_p \oplus k_{q+16} \oplus k_{r+32}$
  $s = \left(c_t^1 c_t^2 c_t^3 c_t^4 c_t^5\right)$
  $y = \left(c_t^4 c_t^5\right)$
  $u = \left(c_t^5 c_t^6\right)$
  $p = \left(c_t^1 c_t^2 c_t^3 c_t^4\right)$
  $q = \left(c_t^2 c_t^3 c_t^4 c_t^5\right)$
  $r = \left(c_t^4 c_t^5 c_t^6\right)$
- NFSR: 19 Bit

$$\begin{aligned} n_{t+19} =& k'_t \oplus l_t \oplus c_t^4 \oplus n_t \oplus n_{t+5} \oplus n_{t+10} \oplus n_{t+6} \cdot n_{t+2} \\ & \oplus n_{t+8} \cdot n_{t+13} \oplus n_{t+3} \cdot n_{t+11} \cdot n_{t+15} \\ & \oplus n_{t+4} \cdot n_{t+9} \oplus n_{t+14} \cdot n_{t+15} \cdot n_{t+16} \cdot n_{t+17} \end{aligned}$$

- LFSR: 21 Bit (MLLFSR like in Fruit v1)

$$l_{t+21} = l_t \oplus l_{t+4} \oplus l_{t+9} \oplus l_{t+12} \oplus l_{t+14} \oplus l_{t+17}$$

- Keybit z_t

$$\begin{aligned} h_t =& l_{t+3} \cdot l_{t+7} \oplus l_{t+1} \cdot l_{t+11} \oplus n_{t+18} \cdot l_{t+13} \\ & \oplus l_{t+5} \cdot l_{t+16} \oplus n_{t+1} \cdot n_{t+17} \cdot l_{t+20} \end{aligned}$$

$$\begin{aligned} z_t =& h_t \oplus n_t \oplus n_{t+3} \oplus n_{t+7} \oplus n_{t+9} \oplus n_{t+12} \\ & n_{t+14} \oplus n_{t+18} \oplus l_{t+19} \end{aligned}$$

- IV' (extension of 35-bit IV to 70 bits):
  $IV' := 10000 v_0 v_1 \dots v_{33} v_{34} 000 \dots 000$
- Key loading:
  $(n_0, \dots, n_{18}) := (k_0, \dots, k_{18})$
  $(l_0, \dots, l_{20}) := (k_{19}, \dots, k_{39})$
- Key schedule counter initialization:
  $\left(c_0^1, c_0^2, c_0^3, c_0^4, c_0^5, c_0^6\right) := (0, \dots, 0)$
- Initialization:
  1. 65 IV loading and mixing steps as described in the Fruit v1 paper [GHX16] (there: 130 steps)
  2. Set
     $$\left(c_{65}^1, c_{65}^2, c_{65}^3, c_{65}^4, c_{65}^5, c_{65}^6\right) := (n_{65}, n_{66}, n_{67}, n_{68}, n_{69}, l_{65})$$
     and then $l_{65} := 1$.
  3. Clock 40 times as described in the Fruit v1 paper paper (there: 80 times).
- The first keystream bit that is output is z_{105}.