Cryptanalysis of Draco

Subhadeep Banik

Universita della Svizzera Italiana, Lugano, subhadeep.banik@usi.ch

Abstract. Draco is a lightweight stream cipher designed by Hamann et al. in IACR ToSC 2022. It has a Grain-like structure with two state registers of size 95 and 33 bits. In addition, the cipher uses a 128-bit secret key and a 96-bit IV. The first 32 bits of the key and the IV forms a non-volatile internal state that does not change during the time that the cipher produces keystream bits.

The authors claim that the cipher is provably secure against Time-Memory-Data (TMD) Tradeoff attacks. However in this paper, we first present two TMD tradeoff attacks against Draco. Both attacks leverage the fact that for certain judiciously chosen IVs the state update function of the cipher depend on only a small fraction of the non-volatile internal state. This makes the state update function in Draco essentially a one way function over a much smaller domain and range. The first attack requires around $2^{114.2}$ Draco iterations and requires that the adversary has access to 2^{32} chosen IVs. The second attack is such that the attack parameters can be tuned as per the requirements of the attacker. If the attacker prioritizes that the number of different chosen IVs is limited to 2^{20} say, then the attack can be done in around time proportional to 2^{126} Draco rounds. However if the total attack complexity is to be optimized, then the attack can be performed in 2^{107} time using around 2^{40} chosen IVs.

1 Introduction

In FSE 2015, Armknecht and Mikhalev proposed the stream cipher Sprout [AM15] whose internal state of was equal to the size of its key. This was counter-intuitive since after [BS00], it was widely accepted that to be secure against generic TMD Tradeoff attacks, the internal state of a stream cipher needed to be at least twice the size of the secret key. However one novelty of the Sprout design ensured that the cipher remained secure against generic TMD tradeoffs like the one in [BS00]. The state update function of Sprout required additional input from the secret key and so the effective internal state still was double size of the key.

Stream ciphers with internal state makes the cipher particularly attractive for compact lightweight implementations. Hence, although Sprout was cryptanalyzed in subsequent papers [Ban15,EK15,LNP15,ZG15], there have been lots of research into designing secure stream ciphers with short internal states: Lizard [HKM17], Plantlet [MAM16], Atom [BCI+21] are some of the constructions that have been recently designed.

In [HMKM22], the authors designed the stream cipher Draco which has a Grain like structure. It uses two non linear registers of sizes 95 and 33 respectively. Additionally, it uses a 128 bit secret key and a 96 bit IV: the first 32 bits of the key and the IV forms a non-volatile 128-bit internal state that does not change during the operation of stream cipher. This helps reduce the power consumption of the stream cipher, since a part of the finite state machine is held at a constant value. The keystream and state update functions are derived as Boolean functions of both the non-volatile and the volatile internal state (i.e. the non-linear registers). The authors stipulate that the maximum amount of keystream that a key-IV pair can produce is 2^{32} bits.

1.1 Contributions and Organization

In this paper, we present a key recovery attack on Draco that requires a computational complexity of around $2^{114.2}$ but requires access to 2^{32} chosen IVs. We present a second attack, the parameters of which can be tuned as per the requirements of the attacker: if the attacker prioritizes the optimization of attack complexity then the 2nd attack can be done in 2^{107} time using around 2^{40} chosen IVs. Both attacks take advantage of the fact for certain well chosen IVs the state update function of the cipher depends on only a small fraction of the non-volatile internal state. This makes the state update function in Draco essentially a one way function over a much smaller domain and range. This allows the attacker to construct tables in the offline phase over a much smaller domain and range which can be used in the online stage to look for collisions. The paper is organized in the following manner.

a. In Section 2, we present the mathematical description of the Draco stream cipher.
b. In Sections 3 and 4 respectively, we present the first and second attacks.
c. Section 5, concludes the paper.

2 Description of Draco

The exact structure of Draco is explained in Figure 1. Draco uses a 128-bit key $K = k_0, k_1, \dots, k_{127}$ and a 96-bit initial vector $IV = v_0, v_1, \dots, v_{95}$. It consists of two NFSRs of size 95 and 33 bits each. Certain bits of both the shift registers are taken as inputs to a combining Boolean function, whence the keystream is produced. The volatile 128-bit inner state of Draco is distributed over the two NFSRs, NFSR1 and NFSR2, whose contents at time $t = 0, 1, \dots$ is denoted by $B^t = (b_0^t, b_1^t, \dots, b_{94}^t)$ and $S^t = (s_0^t, s_1^t, \dots, s_{32}^t)$ respectively.

The non-volatile state consists of the key-bits $k_0, k_1, \dots, k_{31}$ and the IV. For each t, the non volatile state produces a bit d_t defined as follows: first define an augmented IV vector of 97 bits defined thus $x_0 = 0$ and $x_i = v_{i-1}$ for $i \in [1, 96]$, then d_t is defined thus:

$$d_t = \begin{cases} x_{t \bmod 97} & \text{if } t \leq 255 \\ x_{t \bmod 97} \oplus k_{t \bmod 32} & \text{otherwise.} \end{cases}$$

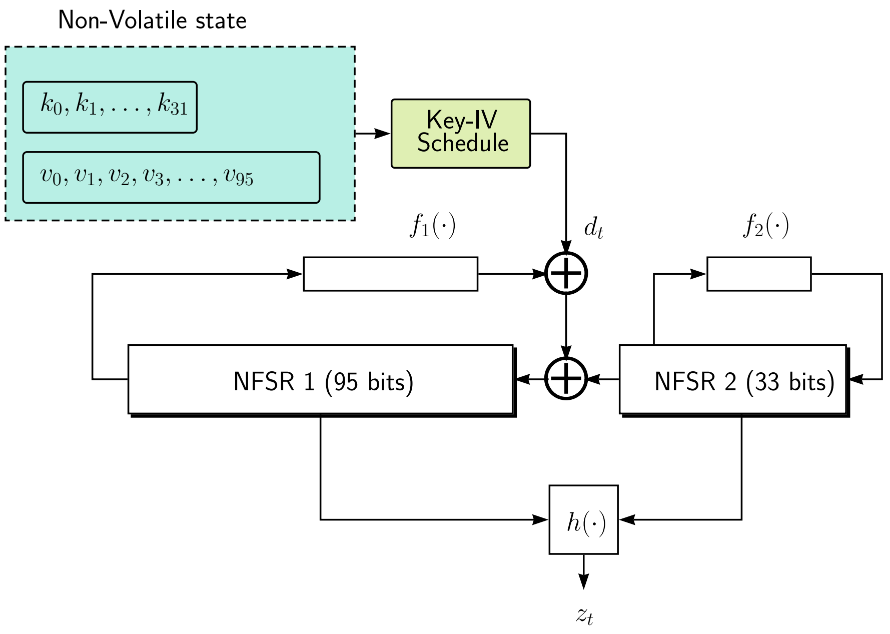

Fig. 1: Block Diagram for Draco

The keystream is produced after performing the following steps:

Phase 1: Key-IV loading: Let $K = (k_0, k_1, \ldots, k_{127})$ denote the 128-bit key and $IV = (v_0, v_1, \ldots, v_{95})$ be the 96-bit public IV. The registers of the keystream generator are initialized as follows:

$$b_j^0 = \begin{cases} k_j \oplus 1, & \text{for } j = 0 \\ k_j, & \text{for } j \in \{1, 2, 3, \ldots, 94\} \end{cases}$$

The remaining key is loaded on to NFSR2, i.e. $s_i^0 = k_{i+95}$, for $i \in \{0, 1, 2, \ldots, 32\}$

Phase 2: Mixing: During this phase the cipher is clocked for 512 cycles without producing any keystream bits. During this phase the registers are updated as follows. For $t = 0, 1, 2, \ldots, 511$, we compute:

$$\begin{aligned} b_i^{t+1} &= b_{i+1}^t, \quad \text{for} \quad i \in \{0, 1, \ldots, 93\} \\ b_{94}^{t+1} &= z_t \oplus s_0^t \oplus f_1(B^t) \oplus d_t \end{aligned}$$

$$\begin{aligned} s_i^{t+1} &= s_{i+1}^t, \quad \text{for} \quad i \in \{0, 1, \ldots, 31\} \\ s_{32}^{t+1} &= z_t \oplus f_2(S^t) \end{aligned}$$

where $f_1(S^t), f_2(B^t)$ and z_t are computed as follows:

$$\begin{aligned}
f_2(S^t) = {} & s_0^t \oplus s_2^t \oplus s_7^t \oplus s_9^t \oplus s_{10}^t \oplus s_{15}^t \oplus s_{23}^t \oplus s_{25}^t \oplus s_{30}^t \oplus s_8^t \cdot s_{15}^t \oplus \\
& s_{12}^t \cdot s_{16}^t \oplus s_{13}^t \cdot s_{15}^t \oplus s_{13}^t \cdot s_{25}^t \oplus s_1^t \cdot s_8^t \cdot s_{14}^t \oplus s_1^t \cdot s_8^t \cdot s_{18}^t \oplus \\
& s_8^t \cdot s_{12}^t \cdot s_{16}^t \oplus s_8^t \cdot s_{14}^t \cdot s_{18}^t \oplus s_8^t \cdot s_{15}^t \cdot s_{16}^t \oplus s_8^t \cdot s_{15}^t \cdot s_{17}^t \oplus \\
& s_{15}^t \cdot s_{17}^t \cdot s_{24}^t \oplus s_1^t \cdot s_8^t \cdot s_{14}^t \cdot s_{17}^t \oplus s_1^t \cdot s_8^t \cdot s_{17}^t \cdot s_{18}^t \oplus \\
& s_1^t \cdot s_{14}^t \cdot s_{17}^t \cdot s_{24}^t \oplus s_1^t \cdot s_{17}^t \cdot s_{18}^t \cdot s_{24}^t \oplus s_8^t \cdot s_{12}^t \cdot s_{16}^t \cdot s_{17}^t \oplus \\
& s_8^t \cdot s_{14}^t \cdot s_{17}^t \cdot s_{18}^t \oplus s_8^t \cdot s_{15}^t \cdot s_{16}^t \cdot s_{17}^t \oplus s_{12}^t \cdot s_{16}^t \cdot s_{17}^t \cdot s_{24}^t \oplus \\
& s_{14}^t \cdot s_{17}^t \cdot s_{18}^t \cdot s_{24}^t \oplus s_{15}^t \cdot s_{16}^t \cdot s_{17}^t \cdot s_{24}^t \oplus \prod_{i=1}^{32} (1 \oplus s_i^t)
\end{aligned}$$

$$\begin{aligned}
f_1(B^t) = {} & b_0^t \oplus b_{26}^t \oplus b_{56}^t \oplus b_{89}^t \oplus b_{94}^t \oplus b_3^t \cdot b_{67}^t \oplus b_{11}^t \cdot b_{13}^t \oplus b_{17}^t \cdot b_{18}^t \oplus \\
& b_{27}^t \cdot b_{59}^t \oplus b_{36}^t \cdot b_{39}^t \oplus b_{40}^t \cdot b_{48}^t \oplus b_{50}^t \cdot b_{79}^t \oplus b_{54}^t \cdot b_{71}^t \oplus b_{58}^t \cdot b_{63}^t \oplus \\
& b_{61}^t \cdot b_{65}^t \oplus b_{68}^t \cdot b_{84}^t \oplus b_8^t \cdot b_{46}^t \cdot b_{87}^t \oplus b_{22}^t \cdot b_{24}^t \cdot b_{25}^t \oplus b_{70}^t \cdot b_{78}^t \cdot b_{82}^t \oplus \\
& b_{86}^t \cdot b_{90}^t \cdot b_{91}^t \cdot b_{93}^t
\end{aligned}$$

$$L_t = b_7^t \oplus b_{15}^t \oplus b_{32}^t \oplus b_{47}^t \oplus b_{66}^t \oplus b_{80}^t \oplus b_{92}^t$$

$$Q_t = b_5^t \cdot b_{85}^t \oplus b_{12}^t \cdot b_{74}^t \oplus b_{20}^t \cdot b_{69}^t \oplus b_{34}^t \cdot b_{57}^t$$

$$\begin{aligned}
T_t^1 = {} & b_{53}^t \oplus b_{38}^t \cdot b_{44}^t \oplus b_{23}^t \cdot b_{49}^t \cdot b_{83}^t \oplus b_6^t \cdot b_{33}^t \cdot b_{51}^t \cdot b_{73}^t \oplus \\
& b_4^t \cdot b_{29}^t \cdot b_{43}^t \cdot b_{60}^t \cdot b_{81}^t \oplus b_9^t \cdot b_{14}^t \cdot b_{35}^t \cdot b_{42}^t \cdot b_{55}^t \cdot b_{77}^t \oplus \\
& b_1^t \cdot b_{16}^t \cdot b_{28}^t \cdot b_{45}^t \cdot b_{64}^t \cdot b_{75}^t \cdot b_{88}^t
\end{aligned}$$

$$T_t^2 = s_{26}^t \oplus s_5^t \cdot s_{19}^t \oplus s_{11}^t \cdot s_{22}^t \cdot s_{31}^t$$

$$T_t^3 = b_{76}^t \oplus s_3^t \cdot b_{10}^t \oplus s_{20}^t \cdot b_{21}^t \cdot b_{30}^t \oplus s_6^t \cdot s_{29}^t \cdot b_{62}^t \cdot b_{72}^t$$

$$z_t = L_t \oplus Q_t \oplus T_t^1 \oplus T_t^2 \oplus T_t^3$$

Phase 3: Keystream Generation: During this phase the feedback of the keystream bit is discontinued, and the cipher starts producing keystream bits. Thus for $t = 512, 513, 514, \ldots$, we compute:

$$
\begin{aligned}
b_i^{t+1} &= b_{i+1}^t, \quad \text{for} \quad i \in \{0,1,\ldots,93\}\\
b_{89}^{t+1} &= s_0^t \oplus f_2(B^t) \oplus d_t\\
\\
s_i^{t+1} &= s_{i+1}^t, \quad \text{for} \quad i \in \{0,1,\ldots,31\}\\
s_{30}^{t+1} &= f_1(S^t)
\end{aligned}
$$

The cipher produces the keystream bit z_t using the expression given above. Note that the authors stipulate that the maximum amount of keystream bits generated from any key-IV pair is limited to 2^{32}. Also note that throughout the paper we use one Draco *iteration/round* interchangeably as a unit of time required for cryptanalysis. This unit refers to the set of operations required to update the Draco volatile state by one clock cycle.

3 First Attack

We begin this section by making a few observations about the algebraic structure of Draco. The first is that given access to the non-volatile state, the state update routines during both the initialization and keystream generation phase are one-to-one and invertible. Note that the functions f_1, f_2 can be written as $f_1(B^t) = b_0^t \oplus s_0^t \oplus d_t \oplus f_1'(\overline{B}^t)$ and $f_2(S^t) = s_0^t \oplus f_2'(\overline{S}^t)$, where $\overline{B}^t = [b_1^t, b_2^t \ldots, b_{94}^t]$ and $\overline{S}^t = [s_1^t, s_2^t \ldots, s_{32}^t]$ are all the trailing bits of S^t, B^t. We present in Table 1 and 2 algorithms Init^{-1} and Update^{-1} which clocks back the state registers by one round during the initialization and keystream generation phases respectively.

Note that during Phase 3, when the cipher starts producing keystream bits the value d_t is always calculated as $k_{t \bmod 32} \oplus x_{t \bmod 97}$. This sequence d_t plays a major role in updating the volatile 128-bit internal state of the stream cipher, i.e. the state contained in the two NFSRs. We make the following observations:

Observation 1 *The sequence d_t is periodic with period equal to 3104. This is easy to see since the key bits and augmented IV bits repeat in cycles of 32 and 97 respectively. Since $32 * 97 = 3104$, after 3104 cycles the key and IV bits get synced again. Consider some $\tau \equiv 0 \bmod 3104$. Then we have*

$$
d_{\tau+i} = \begin{cases} k_0 & \textit{if } i = 0\\ k_i \oplus v_{i-1} & \textit{if } i \in \{1,2,3,\ldots,31\}\\ k_{i-32} \oplus v_{i-1} & \textit{if } i \in \{32,33,34,\ldots,63\}\\ k_{i-64} \oplus v_{i-1} & \textit{if } i \in \{64,65,66,\ldots,95\}\\ k_0 \oplus v_{95} & \textit{if } i = 96 \end{cases}
$$

Consider an initial vector IV^* of the form

$$IV^* = k_1, k_2, \ldots, k_{31} \mid\mid k_0, k_1, \ldots, k_{31} \mid\mid k_0, k_1, \ldots, k_{31} \mid\mid k_0,$$

```
Algorithm $\mathsf{Init}^{-1}$

1. Input: $S^t, B^t$: The NFSR states at time $t$
2. Input: $k_0, k_1, \ldots, k_{31}, v_0, v_1, \ldots, v_{95}$: The non-volatile state
3. Output: $S^{t-1}, B^{t-1}$: The NFSR states at time $t-1$
    • $s \leftarrow s_{32}^t, \quad b \leftarrow b_{94}^t$.
    • Note that $\overline{B}^{t-1} = (b_0^t, b_1^t \ldots, b_{93}^t), \quad \overline{S}^{t-1} = (s_0^t, s_1^t \ldots, s_{31}^t)$
    • Compute $d_{t-1}$ from the non-volatile state.
    • Compute $\hat{z}$ from $\overline{B}^{t-1}, \overline{S}^{t-1}$ .
    • $\hat{s} = s \oplus f_2'(\overline{S}^{t-1}) \oplus \hat{z}, \quad \hat{b} = b \oplus f_1'(\overline{B}^{t-1}) \oplus \hat{s} \oplus \hat{z} \oplus d_{t-1}$
    • $S^{t-1} \leftarrow (\hat{s}, s_0^t, s_1^t \ldots, s_{31}^t)$, $B^{t-1} \leftarrow (\hat{b}, b_0^t, b_1^t \ldots, b_{93}^t)$
    • Return $S^{t-1}, B^{t-1}$
```

Table 1: Algorithm Init^{-1} inverts one initialization round

```
Algorithm $\mathsf{Update}^{-1}$

1. Input: $S^t, B^t$: The NFSR states at time $t$
2. Input: $k_0, k_1, \ldots, k_{31}, v_0, v_1, \ldots, v_{95}$: The non-volatile state
3. Output: $S^{t-1}, B^{t-1}$: The NFSR states at time $t-1$
    • $s \leftarrow s_{32}^t, \quad b \leftarrow b_{94}^t$.
    • Note that $\overline{B}^{t-1} = (b_0^t, b_1^t \ldots, b_{93}^t), \quad \overline{S}^{t-1} = (s_0^t, s_1^t \ldots, s_{31}^t)$
    • Compute $d_{t-1}$ from the non-volatile state.
    • $\hat{s} = s \oplus f_2'(\overline{S}^{t-1}), \quad \hat{b} = b \oplus f_1'(\overline{B}^{t-1}) \oplus \hat{s} \oplus d_{t-1}$
    • $S^{t-1} \leftarrow (\hat{s}, s_0^t, s_1^t \ldots, s_{31}^t)$, $B^{t-1} \leftarrow (\hat{b}, b_0^t, b_1^t \ldots, b_{93}^t)$
    • Return $S^{t-1}, B^{t-1}$
```

Table 2: Algorithm Update^{-1} inverts one round during keystream generation phase

or in other words $IV^* = (\kappa||\kappa||\kappa) \lll 1$, where $\kappa = k_0, k_1, \ldots, k_{31}$. For such an IV it is not too difficult to see that the 96-bit sequence $d_{\tau+1}, d_{\tau+2}, \ldots, d_{\tau+96} = 0^{96}$, $\forall \tau \equiv 0 \bmod 3104$.

The above observation is significant since it shows us that for every 32 bit key prefix there exists one IV so that the contribution of the non-volatile state to the update function of the volatile internal state is 0 for 96 consecutive cycles. Once this happens, (a) the state update function and (b) the keystream produced by the cipher in these 96 consecutive cycles, i.e. $\tau + 1$ to $\tau + 96$ is completely independent of the non-volatile state.

For the next three cycles $\tau + \{97, 98, 99\}$, although the state update becomes once again dependent on the non-volatile state, the keystream bits produced in these cycles are still independent. This is because b^t_{92} is the highest tap location in NFSR1 that is input to the output filter function. Since $d_{\tau+96} = 0$, all state bits at $\tau + 97$ are independent of the non-volatile state and so is the keystream bit. At $\tau + \{98, 99\}$, the non-volatile state affects the register locations 94 and $\{93, 94\}$ respectively of NFSR1, and so the keystream bit is still independent of the non-volatile state.

This gives rise to a set of weak configurations in the cipher which can be exploited in the following way.

Offline Stage In the offline stage, the attacker tries to form a table containing tuples [State, Keystream] for cipher states in the weak configuration described above. More formally the steps can be described thus:

1. For $i = 1$ to N do the following:
   - Select $State_i \xleftarrow{R} \{0,1\}^{128}$ randomly.
   - Assume $State_i$ is the volatile internal state of Draco with $d_t = 0$ for 96 consecutive cycles.
   - Generate the 99 bit keystream vector Z_i for $State_i$.
   - Store $State_i$ in a hash table Tab indexed by Z_i.
2. End for

Note that if $N > 2^{99}$, then whp, there will be multiple state vectors that produce the same 99 bit keystream segment. Hence the table cells should be equipped to accommodate multiple state vectors. The total time complexity required to generate the table is around $P = 99 \cdot N$ Draco rounds. This is typically less than N Draco encryptions. The total memory complexity is $128 \cdot N$ bits. By standard randomness assumptions, we can assume that the N states are distributed uniformly among the 2^{99} table entries, and so each table cell has on average $J = N \cdot 2^{-99}$ state vectors.

Online Stage In the online, stage the attacker queries 2^{32} IVs, where each IV is of the form $(V||V||V) \lll 1$, $\forall\ V \in \{0,1\}^{32}$. Note that when $V = \kappa$, then there will several instances during the keystream generation phase when $d_t = 0$ for 96 consecutive cycles. In fact this will happen whenever $t = \tau \equiv 0 \bmod 3104$.

Since we are allowed to extract 2^{32} keystream bits from every key-IV pair, such instances will occur around $U = \frac{2^{32}}{3104} \approx 2^{20.4}$ times.

However the attacker naturally does not know the exact value of κ beforehand. So for each IV of form $(V||V||V) \lll 1$ the attacker does the following. For all $\tau \equiv 0 \bmod 3104$, the attacker takes the keystream vector $Z = [z_{\tau+1},\ z_{\tau+2},\ \ldots, z_{\tau+99}]$ and tries to locate the corresponding entry in the table Tab. In other words he proceeds under the assumption that the keystream was produced by the cipher in a weak configuration, i.e. he assumes that the value of V is indeed equal to κ. For all the states St_j found in the Z entry of Tab, the attacker takes the state vector and computes few more keystream bits $\hat{z}_{\tau+100+w}$, (for $w = 0, 1, 2\ldots$) using St_j and the assumed non-volatile state $V, (V||V||V) \lll 1$, and compares them with the keystream bits $z_{\tau+100+w}$ obtained from the key-IV query. More often than not after a few iterations some computed keystream bit and keystream bit obtained from query will not match and the attacker can discard the state.

However when $V = \kappa$, then the attacker may actually find a state stored in Tab after querying it with one of the keystream segments Z. When this happens, each successive computed keystream bit and keystream bit obtained from query will be always equal. The attacker then clocks the state back to the beginning of the key-IV loading phase, i.e. he finds B^0, S^0 using repeated executions of the algorithms in tables 1 and 2, using non-volatile internal state is V, $(V||V||V) \lll 1$. Then it can be easily seen that $[b^0_0 \oplus 1, b^0_1, b^0_2, \ldots, b^0_{94}, s^0_0, s^0_1, \ldots, s^0_{32}]$ is a reveals the secret key K.

More formally the algorithm can be presented as follows. First the attacker queries all 2^{32} IVs of the form $(V||V||V) \lll 1$. For each IV he does the following:

```
1. Assume non-volatile state is $A = (V, (V||V||V) \lll 1)$
2. For all $\tau \equiv 0 \bmod 3104$ (there are $U = 2^{20.4}$ iterations here):
     – Denote $Z = [z_{\tau+1},\ z_{\tau+2},\ \ldots, z_{\tau+99}]$
     – Find all the states stored in $\mathsf{Tab}[Z]$.
     – For each such state $St_j$
         → Set $w \leftarrow 0$
         → Do the following
             • Compute the keystream $\hat{z}_{\tau+100+w}$ using $St_j, A$
             • If $\hat{z}_{\tau+100+w} \neq z_{\tau+100+w}$ discard and exit the do loop.
             • Else update $w \leftarrow w + 1$
             • If $w$ is sufficiently large, then $St^* = St_j$ is the correct candidate.
             • If so exit the all loops and goto step 4.
3. End for
4. Clock back $St^*$ to get the state $B^0, S^0$ at the beginning of key-IV loading.
5. Return $K = [b^0_0 \oplus 1, b^0_1, b^0_2, \ldots, b^0_{94}, s^0_0, s^0_1, \ldots, s^0_{32}]$
```

The above algorithm is guaranteed to succeed whp, if when $V = \kappa$, there is a collision between one of the register states encountered during the production of 2^{32} keystream bits and the states stored in Tab. We encounter around $U \approx 2^{20.4}$ internal states during the keystream generation, whereas N state vectors are stored in the table. By standard birthday assumptions, we will get a collision with high probability when $N \cdot U = 2^{128}$, i.e. we need $N \approx 2^{107.6}$.

The total complexity of the online phase is determined by the number of Draco rounds required to eliminate all candidate states. Note that for each IV, we generate $2^{20.4} \cdot J = 2^{20.4} \cdot N \cdot 2^{-99} \approx 2^{20.4-99+107.6} = 2^{29}$ internal states on average. And so the total number of internal states extracted from the table, during the course of the entire online phase, is around $2^{29+32} = 2^{61}$. By standard randomness assumptions, half of them get eliminated after 1 iteration, $\frac{1}{4}$ after 2 iterations, $\frac{1}{8}$ after 3 iterations etc. Hence the total number of iterations required to eliminate all incorrect candidate internal states is given by

$$\sum_{i=1}^{61} i \cdot 2^{-i} \cdot 2^{61} \approx 2^{62}$$

A correct internal state is therefore obtained, when it does not get eliminated after sufficiently many iterations (i.e. some integer higher than 61, say ≈ 100). After recovering the correct state we have to clock the cipher back to the key-IV loading stage. This requires a maximum of $2^{32} + 512$ inverse Draco iterations. Assuming that one forward and inverse iteration take the same time to compute, the total complexity of the online stage is given as $T = 2^{62} + 2^{32} + 512 \approx 2^{62}$ Draco iterations. The total time complexity of the online and offline stages is given by $P + T = 99 \cdot N + T \approx 2^{114.2}$ iterations of Draco. The memory complexity is $M = 128 \cdot N \approx 2^{114.6}$ bits.

4 Second Attack

In the first attack we banked on the fact that if the attacker queried the correct IV, then he would be able to encounter states which can update itself and produce keystream bits without any contribution from the non-volatile internal state. In the second attack the idea is to store more tables in the offline stage, ideally one for every 32-bit key prefix.

Before we look at the attack formally, let us look at it informally. Let's say that the IV is fixed to 0^{96}. In that case the non-volatile state effectively only contains the 32-bit key prefix. For all 32-bit key prefixes $\mathbf{k}$, the attacker could take around 2^{96-D} (for some integer D) randomly chosen internal states St and generate 128 bit keystream bit vector Z from it. As before store $\mathsf{Tab}_{\mathbf{k}}[Z] = St$. The total time complexity of this phase $2^{32} \cdot 2^{96-D} \cdot 128 = 2^{135-D}$ Draco iterations, and so is the total memory complexity.

In the online stage the attacker requests keystream for the all zero IV. For each 128-bit keystream segment so obtained (there are a total of $2^{32} - 128$ of them), the attacker will then try to locate the internal state in each of the 2^{32} tables. Note that, as before, each incorrect internal state so obtained can be eliminated by computing a few more keystream bits and comparing with the keystream bits obtained from the oracle. Again, as before, if the state does not get eliminated after sufficiently many iterations, it can be considered to be correct. We then clock it back to the key-IV loading stage to recover the key. For the attack to succeed whp, by standard randomness assumptions, we need $(2^{32} - 128) \cdot 2^{96-D} = 2^{128}$. In other words, we need $D \approx 0$ for which the offline complexity alone becomes 2^{135} Draco iterations. Although one Draco iteration takes time much less than one encryption, 2^{135} Draco iterations is very close to the complexity of exhaustive search.

4.1 Query more IVs

Although we did not compute the online complexity above, it is much less than the offline complexity. This is easy to see since, for every keystream segment, we generate 2^{32} states (one from each of the 2^{32} tables) and so the total number of states generated is $2^{32} \cdot (2^{32} - 128) \approx 2^{64}$. We have already seen that we need only around $\sum_{i=1}^{64} i \cdot 2^{64-i} \approx 2^{65}$ Draco iterations to eliminate all of them. Now if we generated more internal states in the online stage, simply by querying more IVs, we may be able to make the attack work for larger D, and at the same time not increase the online complexity too much.

Let us say we take 2^E initial vectors of the form $0^{96-E}||e$, where $e \in \{0,1\}^E$, i.e. $v_i = 0$ for all $i < 96 - E$. Now consider the value of $d_t = k_{t \bmod 32} \oplus x_{i \bmod 97}$ during the keystream generation phase in each cycle of 97 iterations. Consider some $\lambda \equiv 0 \bmod 97$. Then it can be seen that the $(97-E)$-bit sequence $d_\lambda, d_{\lambda+1}, \ldots, d_{\lambda+96-E}$ does not depend on the last E bits of the IV. This is because **(a)** $d_\lambda = k_{\lambda \bmod 32} \oplus x_0 = k_{\lambda \bmod 32}$, and **(b)** for all other $i \in [1, 96-E]$, we have

$$\begin{aligned} d_{\lambda+i} = k_{\lambda+i \bmod 32} \oplus x_{\lambda+i \bmod 97} &= k_{\lambda+i \bmod 32} \oplus x_{i \bmod 97} \\ &= k_{\lambda+i \bmod 32} \oplus v_{i-1 \bmod 97} = k_{\lambda+i \bmod 32} \end{aligned}$$

This implies that for all such IVs the state update function (and hence the keystream bit) for $97 - E$ consecutive cycles depends only on the key-prefix. By arguments similar to those presented in Section 3, we can deduce that the keystream bits in the next 3 cycles are also independent of the last E bits of the initial vector and dependent only on the key prefix and the register state. Therefore if the initial vectors are of the form $0^{96-E}||e$, then in every cycle of 97 iterations, a total of $\mathbf{min}(97, 100 - E)$ keystream bits depends only on the key and register state. If $E \geq 3$, then this figure is exactly $100 - E$. This again gives rise to a set of weak configurations, which we exploit in the following way.

Offline Stage In the offline stage, the attacker tries to form tables for each 32-bit key prefix $\mathbf{k}$. So for each $\mathbf{k} \in \{0,1\}^{32}$ the attacker does the following:

1. Choose some integers D and $E \geq 3$.
2. For $i = 1$ to 2^{96-D} do the following:
   – Select $State_i \xleftarrow{R} \{0,1\}^{128}$ randomly.
   – Assume $State_i$ is the volatile internal state of Draco for some $IV \in 0^{96-E}||e$

```
    – Generate the $(100 - E)$-bit keystream vector $Z_i$ for $State_i$.
    – Store $State_i$ in a hash table $\mathsf{Tab}_{\mathbf{k}}$ indexed by $Z_i$.
3. End for
```

Again if $96-D > 100-E$, each table cell may need to accommodate multiple internal states, and each table cell will on average have $J_1 = 2^{96-D-100+E} = 2^{E-D-4}$ state vectors. The total offline time complexity is $P_1 = 2^{32} \cdot 2^{96-D} \cdot (100-E) = (100-E) \cdot 2^{128-D}$ number of Draco rounds. The total memory complexity is $M_1 = 2^{32} \cdot 2^{96-D} \cdot 128 = 2^{135-D}$ bits.

Online Stage The algorithm in the online stage is similar to the previous attack. Now the attacker queries 2^E IVs, where each IV is of the form $0^{96-E}||e$. For each IV, and for each $\lambda \equiv 0 \bmod 97$, the attacker extracts the keystream vector $Z = [z_\lambda, z_{\lambda+1}, \ldots, z_{\lambda+99-E}]$, the attacker tries to locate Z in each of the 2^{32} tables constructed in the offline phase. Since the number of keystream bits per key-IV pair is limited to 2^{32}, there are $U_1 = \frac{2^{32}}{97} \approx 2^{25.4}$ keystream segments for every IV.

Given any fixed IV, for each keystream segment Z the attacker gets $J_1 = 2^{E-D-4}$ states from each table $\mathsf{Tab}_{\mathbf{k}}$, and so around $U_1 \cdot 2^{32+E-D-4} = 2^{53.4+E-D}$ states are generated from all the tables and from all the keystream segments. As before each incorrect internal state so obtained can be eliminated by computing a few more keystream bits and comparing with the keystream bits obtained from the oracle. If the state does not get eliminated after sufficiently many iterations, it can be considered to be correct. We then clock it back to the key-IV loading stage to recover the key. We state the algorithm formally here. First the attacker queries all 2^E IVs of the form $0^{96-E}||e$. For each IV he does the following:

```
1. For all $\lambda \equiv 0 \bmod 97$ (there are $U_1 = 2^{25.4}$ iterations here):
    – Denote $Z = [z_{\lambda+1},\ z_{\lambda+2},\ \ldots, z_{\lambda+99-E}]$
    – For all $\mathbf{k} \in \{0,1\}^{32}$
      A: Find all the states stored in $\mathsf{Tab}_{\mathbf{k}}[Z]$.
      B: For each such state $St_j$
          → Set $w \leftarrow 0$
          → Do the following
              • Non-volatile state is $A_1 = (\mathbf{k}, 0^{96-E}||e)$
              • Compute the keystream $\hat{z}_{\lambda+99-E+w}$ using $St_j, A_1$
              • If $\hat{z}_{\lambda+99-E+w} \neq z_{\lambda+99-E+w}$ discard and exit the do loop.
              • Else update $w \leftarrow w + 1$
              • If $w$ is sufficiently large, $St^* = St_j$ is the correct candidate.
              • If so exit the all loops and goto step 3.
2. End for
3. Clock back $St^*$ to get the state $B^0, S^0$ at the beginning of key-IV loading.
4. Return $K = [b_0^0 \oplus 1, b_1^0, b_2^0, \ldots, b_{94}^0, s_0^0, s_1^0, \ldots, s_{32}^0]$
```

Complexity Estimation To estimate the complexity, let us look at a few details regarding the state update function of Draco. We have already seen that when queried with an IV of the form $0^{96-E}||e$, at all $\lambda \equiv 0 \bmod 97$ in the keystream phase, the state update is determined the $d_\lambda, d_{\lambda+1}, \ldots$ sequence which happens to be $\mathbf{k}_\lambda||\mathbf{k}_\lambda||\cdots$ where $\mathbf{k}_\lambda := k_{\lambda \bmod 32}, k_{\lambda+1 \bmod 32}, \ldots, k_{\lambda+31 \bmod 32} = (k_0, k_1, \ldots, k_{31}) \lll (\lambda \bmod 32)$. Thus during the course of keystream generation $\mathbf{k}_\lambda$ can take 32 values depending on the residue $\lambda \bmod 32$. In the offline phase we have constructed tables for all possible keystream prefixes. In the online phase our goal is to find a collision in one of these 32 tables $\mathsf{Tab}_{\mathbf{k}_\lambda}$.

Let us suppose we want a collision for some $\lambda = \lambda^*$. Whatever be the value of the unknown 32-bit key prefix, the value of $\mathbf{k}_\lambda$ equals $\mathbf{k}_{\lambda^*}$ exactly when the following system of modular equations are simultaneously satisfied:

$$\begin{aligned} \lambda &\equiv 0 \bmod 97 \\ \lambda &\equiv \lambda^* \bmod 32 \end{aligned} \tag{1}$$

By Chinese remainder theorem, we know that the above always has a unique solution $\bmod\ 3104$. Thus for any single IV we would be encountering $\mathbf{k}_{\lambda^*}$ a total of $U_2 = \frac{2^{32}}{3104} \approx 2^{20.4}$ times. For 2^E initial vectors, this figure is $2^E \cdot U_2 = 2^{20.4+E}$. Thus in the online phase we generate $2^{20.4+E}$ states corresponding to the $\mathsf{Tab}_{\mathbf{k}_{\lambda^*}}$ table. Since $\mathsf{Tab}_{\mathbf{k}_{\lambda^*}}$ has 2^{96-D} states stored in it, this algorithm is guaranteed to succeed whp, when $2^{20.4+E} \cdot 2^{96-D} = 2^{128}$, i.e. we need $E - D > 11.6$.

The total complexity of the online phase is determined by the number of Draco rounds required to eliminate all candidate states. We have seen that for each IV, we generate $2^{53.4+E-D}$ internal states on average. And so the total number of internal states extracted from the tables, during the course of the entire online phase, is around $2^{53.4+E-D+E} = 2^{53.4+2E-D}$. By arguments introduced in the previous section, the total number of iterations required to eliminate all incorrect candidate internal states is given by

$$\sum_{i=1}^{\lceil 53.4+2E-D \rceil} i \cdot 2^{-i} \cdot 2^{53.4+2E-D} \approx 2^{54.4+2E-D}$$

After recovering the correct state we have to clock the cipher back to the key-IV loading stage. This requires a maximum of $2^{32} + 512$ inverse Draco iterations. Thus the total complexity of the online stage is given as $T_1 = 2^{54.4+2E-D} + 2^{32} + 512$ Draco iterations.

We need to choose D, E judiciously to balance attack complexities. For example, $D = 28, E = 40$ seems to balance the online and offline complexities: we get $P_1 = 2^{105.9}$, $T_1 = 2^{106.4}$ iterations of Draco and the memory complexity is $M_1 = 2^{107}$ bits. The total attack complexity is therefore $P_1 + T_1 \approx 2^{107.1}$. However this requires keystream bits from 2^{40} IVs. If we want to limit the total number of different IVs to 2^{20} say, we can choose $D = 8, E = 20$ for which we get $P_1 = 2^{126.3}$, $T_1 = 2^{86.4}$ iterations of Draco and the memory complexity is $M_1 = 2^{127}$ bits.

5 Conclusion

In this paper, we look at the security of the stream cipher Draco. Although the cipher comes with provable security against TMD attacks, we introduce two different flavors of chosen IV attacks, both of which seem to disprove the security claims. Both attacks leverage the fact that for certain judiciously chosen IVs the state update function of the cipher depend on only a small fraction of the non-volatile internal state. In order to be secure the design should probably correct these flaws.

References

AM15. Frederik Armknecht and Vasily Mikhalev. On Lightweight Stream Ciphers with Shorter Internal States. In Gregor Leander, editor, *FSE*, volume 9054 of *Lecture Notes in Computer Science*, pages 451–470. Springer, 2015.

Ban15. Subhadeep Banik. Some Results on Sprout. In Alex Biryukov and Vipul Goyal, editors, *INDOCRYPT*, volume 9462 of *Lecture Notes in Computer Science*, pages 124–139. Springer, 2015.

BCI$^+$21. Subhadeep Banik, Andrea Caforio, Takanori Isobe, Fukang Liu, Willi Meier, Kosei Sakamoto, and Santanu Sarkar. Atom: A stream cipher with double key filter. *IACR Trans. Symmetric Cryptol.*, 2021(1):5–36, 2021.

BS00. Alex Biryukov and Adi Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2000.

EK15. Muhammed F. Esgin and Orhun Kara. Practical Cryptanalysis of Full Sprout with TMD Tradeoff Attacks. In Orr Dunkelman and Liam Keliher, editors, *SAC*, volume 9566 of *Lecture Notes in Computer Science*, pages 67–85. Springer, 2015.

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

HMKM22. Matthias Hamann, Alexander Moch, Matthias Krause, and Vasily Mikhalev. The draco stream cipher: A power-efficient small-state stream cipher with full provable security against tmdto attacks. *IACR Transactions on Symmetric Cryptology*, 2022(2):1–42, Jun. 2022.

LNP15. Virginie Lallemand and María Naya-Plasencia. Cryptanalysis of Full Sprout. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 663–682. Springer, 2015.

MAM16. Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On Ciphers that Continuously Access the Non-Volatile Key. *IACR Trans. Symmetric Cryptol.*, 2016(2):52–79, 2016.

ZG15. Bin Zhang and Xinxin Gong. Another Tradeoff Attack on Sprout-Like Stream Ciphers. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT (2)*, volume 9453 of *Lecture Notes in Computer Science*, pages 561–585. Springer, 2015.