Cryptanalysis of Bluetooth Keystream Generator Two-level E0

Yi Lu⋆ and Serge Vaudenay

EPFL
http://lasecwww.epfl.ch

Abstract. In this paper, we carefully study both distinguishing and key-recovery attacks against Bluetooth two-level E0 given many short frames. Based on a flaw in the resynchronization of Bluetooth E0, we are able to fully exploit the largest bias of the finite state machine inside E0 for our attacks. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks against two-level E0, this is the best one so far.

1 Background

The short-range wireless technology Bluetooth uses the keystream generator E0 to produce the keystream for encryption. After the earlier results [10, 9, 6] of correlation (also called bias) properties inside the Finite State Machine (FSM) towards the one-level E0, most recently, [12] systematically studied the biases and proved two previously known large biases to be the only largest up to 26 consecutive bits of the FSM output sequences. Attacks against E0 mostly focus on one-level E0 only and the best attacks [12, 1, 5] work on one impractically long frame of keystream without exception. Nevertheless, a few attacks [15, 11, 7–9] apply to two-level E0; compared with feasible attack complexities on one-level E0, attack complexities on two-level E0 are extremely high and make the practical Bluetooth E0 unbroken.

The main contribution of this paper is that first based on one of the two largest biases inside the FSM within one-level E0, we identify the bias at two-level E0 due to a resynchronization flaw in Bluetooth E0. Unlike the traditional approach to find the bias, the characterized bias *does not* involve the precomputation of the multiple polynomial with low weight. Second, to utilize the identified bias, we develop a novel attack to directly recover the original encryption key for two-level E0 without reconstructing the initial state of E0 at the second level. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks [15, 11, 7–9] against two-level E0, this is the best so far.

The rest of the paper is structured as follows. In Section 2 we review description of two-level E0. In Section 3 we study the attack against one-level E0. Then, we investigate the E0 resynchronization flaw, which allows to develop the basic attack of previous section into the distinguishing and key-recovery attacks against two-level E0 in Section 4; we further extend our key-recovery attack in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

2.1 The Core of Bluetooth E0

To briefly outline, the core of E0 (both dashed boxes in Fig. 1) can be viewed as a nonlinear filtering generator. The filtering generator consists of four LFSRs

⋆ supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the Swiss National Science Foundation under the grant number 5005-67322.

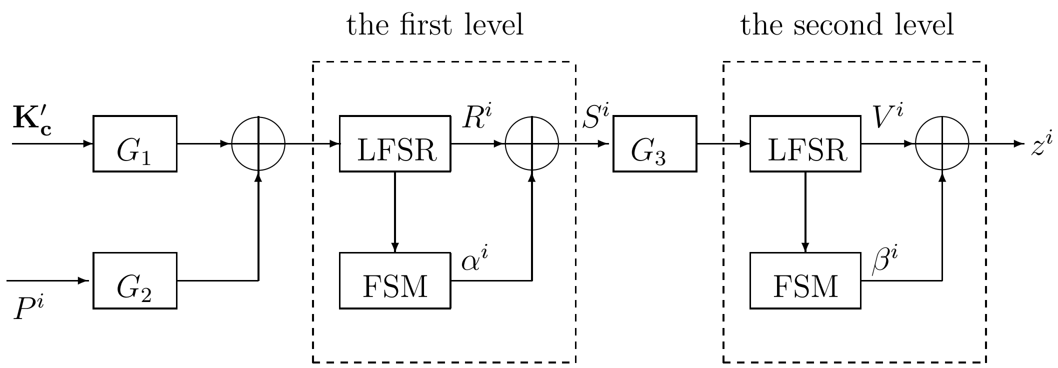

Fig. 1. Diagram of two-level E0 keystream generation

$(R_1, \ldots, R_4)$ which are equivalent to a single L-bit LFSR with connection polynomial[1] $p(x)$ and a 4-bit FSM, where $L = 128$. The keystream bit of the generator is obtained by xoring the output bit of the regularly-clocked LFSR with that of the FSM, which takes the current state of the LFSR as input and emits one bit (denoted by c_t^0 in Bluetooth specification) out of its 4-bit memory.

2.2 Review on Two-level Bluetooth E0

Let $\mathbf{K_c}$ be the L-bit secret key computed by the key generation algorithm E3 [3, p783]. According to [3], the effective key $\mathcal{K}$ of length 8ℓ $(1 \leq \ell \leq 16)$ is computed by

$$\mathcal{K}(x) = \mathbf{K_c}(x) \mod g_1^{(\ell)}(x),$$

where the polynomial $g_1^{(\ell)}(x)$ is specified in [3, p770] and has degree 8ℓ. Bluetooth two-level E0 (depicted in Fig. 1) uses two L-bit inputs: one is the known nonce[2] P^i, the other is the linearly expanded L-bit key $\mathbf{K_c'}$ from $\mathcal{K}$ by $\mathbf{K_c'}(x) = g_2^{(\ell)}(x) \cdot \mathcal{K}(x)$, where the polynomial $g_2^{(\ell)}(x)$ is also specified in [3, p770] and has degree no larger than $L - 8\ell$; or equivalently, we can rewrite it as

$$\mathbf{K_c'} = E(\mathcal{K}), \tag{1}$$

where E is a linear mapping. After initialization of the equivalent LFSR for the first level E0, we can express its initial state[3] $R^i_{[-199,\ldots,L-200]} = (R^i_{-199}, \ldots, R^i_{L-200})$ as

$$R^i_{[-199,\ldots,L-200]} = G_1(\mathbf{K_c'}) \oplus G_2(P^i), \tag{2}$$

for $i = 1, \ldots, m$, where G_1 and G_2 are affine transformations over $GF(2)^L$. Next comes the so-called two-level E0:

[1] Note that the connection polynomial of the equivalent single LFSR equals the product of those of the four LFSRs.

[2] By convention, hereafter we always use the superscript i to indicate the context of the i-th frame.

[3] Throughout the rest of the paper, we use the unified notation $\Omega^i_{[a,\ldots,b]}$ with the formatted subscript to denote the vector $(\Omega^i_a, \ldots, \Omega^i_b)$.

- During the first level, with the FSM initial state preset to zero, E0 runs L clocks producing 200-bit output $S_t^i = R_t^i \oplus \alpha_t^i$ and updating $R^i_{[t,\ldots,t+L-1]}$ by $R^i_{[t+1,\ldots,t+L]} = M(R^i_{[t,\ldots,t+L-1]})$ for $t = -199, \ldots, 0$, where M is the linear mapping over $GF(2)^L$ that corresponds to the companion matrix associated with $p(x)$. Note that first, α_t^i is the output bit of the FSM fed with $R^i_{[t,\ldots,t+L-1]}$; second, the last L-bit output at the first level E0 is $S^i_{[1-L,\ldots,0]}$; last, $R^i_{[1-L,\ldots,0]} = (M^{72} \circ G_1)(\mathbf{K_c'}) \oplus (M^{72} \circ G_2)(P^i)$.
- At the beginning of the second level, the equivalent single LFSR is initialized by $V^i_{[1,\ldots,L]} = G_3(S^i_{[1-L,\ldots,0]})$, where $G_3 : GF(2)^{128} \rightarrow GF(2)^{128}$ is another affine transformation (see [3, p772]); the FSM initial state at the second level remains the same as the one in the end of the first level. Note that the present time is $t = 1$.
- During the second level, for $t = 1, \ldots, 2745$, E0 produces the keystream $z_t^i = V_t^i \oplus \beta_t^i$ for encryption of the i-th frame and updates $V^i_{[t,\ldots,t+L-1]}$ by $V^i_{[t+1,\ldots,t+L]}$.

2.3 An Important Note on G_3

We observe that G_3 is implemented in such a simple way[4] that the last L-bit output sequence of the first level E0 is *byte-wise* reloaded into the four component LFSRs in parallel at the second level E0 with only a few exceptions, which turns out to be a flaw as introduced later in Section 4. For completeness, Table 1 lists in time order the first 24 output bits of $R_1, \ldots, R_4$ individually at the beginning of E0 level two, in terms of the L-bit input $v_0, \ldots, v_{L-1}$.

Table 1. The first 24 output bits of LFSRs at E0 level two

LFSR	output bits
R_1	$v_{71} \cdots v_{64},\ v_{39} \cdots v_{32}, v_7 \cdots v_0$
R_2	$v_{79} \cdots v_{72},\ v_{47} \cdots v_{40}, v_{15} \cdots v_8$
R_3	$v_{111} \cdots v_{104}, v_{87} \cdots v_{80}, v_{55} \cdots v_{48}$
R_4	$v_{119} \cdots v_{112}, v_{95} \cdots v_{88}, v_{63} \cdots v_{56}$

2.4 The Bias inside the FSM

Our starting point would be the bias inside the FSM, which was discovered by [9, 6] and further proved in [12] to be the the largest bias up to 26-bit output sequence of the FSM involving the smallest number of consecutive bits. Let $\lambda = \frac{25}{256}$, we have

$$\Pr(c_t^0 \oplus c_{t+1}^0 \oplus c_{t+2}^0 \oplus c_{t+3}^0 \oplus c_{t+4}^0 = 1) = \frac{1}{2} + \frac{\lambda}{2},$$

for any integer t, assuming that the $L + 4 = 132$-bit initial state of E0 is random and uniformly distributed. Hereafter, we analyze as exactly described in Bluetooth specification [3]. For convenience, we denote c_t^0 used for the first and second level keystream generation by α_t^i, β_t^i respectively. Therefore $\{\alpha_t^i\}, \{\beta_t^i\}$ being separated sequences of $\{c_t^0\}$ both satisfy the same statistical property:

$$\Pr(\alpha_t^i \oplus \alpha_{t+1}^i \oplus \alpha_{t+2}^i \oplus \alpha_{t+3}^i \oplus \alpha_{t+4}^i = 1) = \frac{1}{2} + \frac{\lambda}{2}, \tag{3}$$

$$\Pr(\beta_t^i \oplus \beta_{t+1}^i \oplus \beta_{t+2}^i \oplus \beta_{t+3}^i \oplus \beta_{t+4}^i = 1) = \frac{1}{2} + \frac{\lambda}{2}, \tag{4}$$

[4] It is believed to help increase the rate of keystream generation.

for any t and any i.

3 Security Analysis on E0 Level One

The goal of the attacker in this section is to recover the effective 8ℓ-bit encryption key $\mathcal{K}$ with knowledge of m L-bit output sequences $S^i_{[1-L,\ldots,0]}$ of the first level E0 for $i = 1, \ldots, m$ and the corresponding m nonces $P^1, \ldots, P^m$.

3.1 Finding the Closest Sequences with Fixed Differences

We begin with a very simple problem: given $2m$ L-bit sequences $s^1, \ldots, s^m$ and $\delta^1, \ldots, \delta^m$, where $\delta^1 = \mathbf{0}$ and $\delta^i \neq \delta^j$ for all $i \neq j$, find the L-bit sequence r^1 that maximizes $N(r^1) = \sum_{i=1}^{m} \sum_{t=1}^{L} (s^i_t \oplus r^i_t)$ where $r^i_t = r^1_t \oplus \delta^i_t$ for $i = 1, \ldots, m$ and $t = 1, \ldots, L$.

Similar to the well-known approach (see [9, p251]), the solution based on the idea of minority vote goes fairly easy. We have

$$N(r^1) = \sum_{t=1}^{L} \sum_{i=1}^{m} (s^i_t \oplus r^1_t \oplus \delta^i_t).$$

Thus, in order to maximize $N(r^1)$, we must have

$$r^1_t = \text{minority}\{s^i_t \oplus \delta^i_t : i = 1, \ldots, m\}$$

for all $t = 1, \ldots, L$. Note that in case of a tie for r^1_t, we have two answers for this t-th bit regardless of all the other bits. We finally obtain all the answers that achieve the same maximal $N(r^1)$. The time and memory complexities of the above algorithm both equal the data complexity $O(mL)$.

3.2 Attack against E0 Level One

Let $\Delta^i_{[1-L,\ldots,0]} = R^1_{[1-L,\ldots,0]} \oplus R^i_{[1-L,\ldots,0]}$ for $i = 1, \ldots, m$. By Eq.(2) we have $\Delta^i_{[1-L,\ldots,0]} = (M^{72} \circ G_2)(P^1 \oplus P^i)$. We further set

$$r^i_t = \bigoplus_{j=0}^{4} R^i_{t+j},$$

$$\delta^i_t = \bigoplus_{j=0}^{4} \Delta^i_{t+j},$$

$$s^i_t = \bigoplus_{j=0}^{4} S^i_{t+j},$$

for $i = 1, \ldots, m$ and $t = 1 - L, \ldots, -4$. Note that $s^i_t \oplus r^i_t = \bigoplus_{j=0}^{4} \alpha^i_{t+j}$ follows the biased distribution by Eq.(3). As long as $\sum_{i=1}^{m} \sum_{t=1-L}^{-4} (s^i_t \oplus r^i_t)$ is the maximal and Δ^i_t, S^i_t are known, we can apply the preceding algorithm to recover $(L-4)$ bits of r^1 followed by an exhaustive search on the remaining 4 bits, next solve R^1, then $\mathbf{K'_c}$ by Eq.(2), and finally deduce $\mathcal{K}$ from $\mathbf{K'_c}$ by Eq.(1). The time/memory/data complexities all equal $O(mL+2^4L)$, i.e. $O((m+16)L)$. No precomputation is needed.

About the minimal m to guarantee the valid precondition $\sum_{i=1}^{m} \sum_{t=1-L}^{-4} (s^i_t \oplus r^i_t)$ is the maximal, we use the result in [12, Eq.(10)] based on [2] that says regardless of the value of L and ℓ, we need the minimum

$$m \approx \frac{4 \log 2}{\lambda^2} \text{ (frames).} \tag{5}$$

Consequently, we require $m = 512$ to recover $\mathcal{K}$ from S^i and P^i for $i = 1, \ldots, m$. This results in the time/data/memory complexities all the same as $O(2^{16})$. To verify this, we ran experiments on 512 frames of the randomly-chosen 132-bit E0 initial state 2^{25} times. It turned out that we had 1.5 errors and 0.4 tie in average, which means we can easily correct all errors by an extra checking step in the end in negligible time. Finally, Table 2 compares our result with the only known[5] four attacks [7, 8, 11, 15] working on frames of L-bit consecutive keystreams. Note that existing attacks [7, 8, 11, 15] directly apply to two-level E0 as well with the level-by-level key-recovery scheme[6]; in contrast, our attack is completely disabled against two-level E0 with this scheme as the attack is based on a naive assumption that we directly observe the output of E0 level one[7]. In the next section, we introduce a resynchronization flaw in Bluetooth E0 that leads to a shortcut extended attack against the two-level E0.

Table 2. Comparison of our attack with existing attacks against E0 level one given frames of L bits

Attack Type	Precomputation	Time	Frames	Data	Memory
Divide & Conquer [15]	-	2^{93}	1	2^7	-
BDD [11]	-	2^{77}	1	2^7	-
Algebraic Attack [7]	-	2^{51}	2	2^8	2^{51}
Algebraic Attack [8]	-	$2^{23.4}$	3	$2^{8.6}$	$2^{23.4}$
Our Correlation Attack	-	2^{16}	2^9	2^{16}	2^{16}

4 Security Analysis on Two-level E0

4.1 The resynchronization Flaw in Bluetooth Two-level E0

Define

$$U^i = (U_1^i, \ldots, U_L^i) = G_3(R^i_{[1-L,\ldots,0]}). \tag{6}$$

Following the description of G_3 in Subsection 2.3, we can easily verify that

$$\begin{aligned}
V_t^i &= U_t^i \oplus \alpha^i_{-56-t} \oplus \alpha^i_{-48-t} \oplus \alpha^i_{-16-t} \oplus \alpha^i_{-8-t}, && \text{for } t = 1, \ \ldots, 8,\\
V_t^i &= U_t^i \oplus \alpha^i_{-80-t} \oplus \alpha^i_{-72-t} \oplus \alpha^i_{-32-t} \oplus \alpha^i_{-24-t}, && \text{for } t = 9, \ \ldots, 16,\\
V_t^i &= U_t^i \oplus \alpha^i_{-104-t} \oplus \alpha^i_{-96-t} \oplus \alpha^i_{-56-t} \oplus \alpha^i_{-48-t}, && \text{for } t = 17, \ldots, 24.
\end{aligned}$$

From the above equations, we summarize the characteristics of V_t^i by

$$V_t^i = U_t^i \oplus \alpha^i_{a_t} \oplus \alpha^i_{a_t+8} \oplus \alpha^i_{b_t} \oplus \alpha^i_{b_t+8}, \tag{7}$$

for $t = 1, \ldots, 24$, where $a_t = -t + \text{const}_{\lfloor \frac{t-1}{8} \rfloor}$ and $b_t = -t + \text{const}'_{\lfloor \frac{t-1}{8} \rfloor}$. Note that Eq.(7) is our crucial observation about Bluetooth E0 resynchronization flaw which

[5] In the similar approached paper [9], m is chosen as 45 for E0 level one without the complexity estimate, because the authors focused on the two-level E0 and traded m with the time complexity of E0 level one, whose time complexity is negligible with that of the E0 level two.

[6] namely, the initial state at the first level is reconstructed after the initial state at the second level is recovered

[7] As a matter of fact, according to [3, p763], Bluetooth takes the correlation properties into account and adopts the two-level scheme of keystream generation in practice on purpose.

enables a shortcut attack throughout the two levels of E0. Now, we express the output bit z_t^i of the second level E0 keystream by

$$z_t^i = U_t^i \oplus \alpha_{a_t}^i \oplus \alpha_{a_t+8}^i \oplus \alpha_{b_t}^i \oplus \alpha_{b_t+8}^i \oplus \beta_t^i, \tag{8}$$

for $t = 1, \ldots, 24$. Let $u_t^i = \bigoplus_{j=0}^4 U_{t+j}^i$ and $s_t^i = \bigoplus_{j=0}^4 z_{t+j}^i$. From Eq.(8), we have that

$$s_t^i \oplus u_t^i = \bigoplus_{j=0}^{4} \alpha_{a_t-j}^i \oplus \bigoplus_{j=0}^{4} \alpha_{a_t-j+8}^i \oplus \bigoplus_{j=0}^{4} \alpha_{b_t-j}^i \oplus \bigoplus_{j=0}^{4} \alpha_{b_t-j+8}^i \oplus \bigoplus_{j=0}^{4} \beta_{t+j}^i \tag{9}$$

for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$. Therefore, we deduce an important correlation concerning the practical implementation of E0 from Eq.(3,4,9):

Theorem 1. *Assuming independence of α_t^i's and β_t^i's, we have*

$$\Pr(s_t^i \oplus u_t^i = 1) = \frac{1}{2} + \frac{\lambda^5}{2}$$

for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$.

4.2 A Near-practical Distinguishing Attack against Two-level E0

Using the standard technique of linear cryptanalysis, we expect that $s_t^i \oplus u_t^i$ equals one most of the time for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$ with a total of $\lambda^{-10} \approx 2^{34}$ samples. Since the difference

$$U^i \oplus U^j = G_3(R^i_{[1-L,\ldots,0]}) \oplus G_3(R^j_{[1-L,\ldots,0]}) = (G_3 \circ M^{72} \circ G_2)(P^i \oplus P^j),$$

is known for all i and j by Eq.(2,6), we apply the algorithm in Subsection 3.2 to recover the bit u_1^1 separately with two sets of 2^{34} frames sharing only one common frame denoted as the first frame for both sets. If we get a unique solution, we conclude the keystreams are generated by E0; otherwise, we accept them as truly random sources. The time/data complexities are $O(2 \times 2^{34} \times 5)$, i.e. $O(2^{37})$. In contrast to the conventional treatment based on finding a multiple polynomial with low weight, no precomputation is needed in our scenario. So far, this is the only known near-practical attack against the full two-level E0. We can further improve the distinguisher by recovering u_t^1 for $t = 1, \ldots, 4$ with two different sets of frames of the first 8 bits. Comparing two sets of solutions for the four bits, if we get a majority of identical bits, then we conclude the keystreams are generated by E0, otherwise we accept them as truly random sources. The number of frames we need is $2 \times 2^{34}/4 = 2^{33}$. This results in time/data complexities $O(2^{33} \times 8)$, i.e. $O(2^{36})$.

4.3 The Key-recovery Attack against Two-level E0

From last subsection and Theorem 1, we know that with 2^{34} frames of keystreams, we can recover twelve bits, i.e. u_t^1 for $t = 8k+1, \ldots, 8k+4$, $k = 0, 1, 2$. After that, we try exhaustively for the remaining $|\mathcal{K}| - 12$ bits assuming linear independency of the twelve bits[8]. Note that we have

$$U^i = (G_3 \circ M^{72} \circ G_1)(\mathbf{K}'_{\mathbf{c}}) \oplus (G_3 \circ M^{72} \circ G_2)(P^i), \tag{10}$$

[8] We tested and found that the twelve bits are linearly independent for all choices of effective keylength $|\mathcal{K}| = 8\ell$ except for $|\mathcal{K}| = 8$ in which case our attack is worse than the brute force attack and becomes meaningless anyway.

by Eq.(6,2). So, we deduce from Eq.(10,1) that

$$U^i = (G_3 \circ M^{72} \circ G_1 \circ E)(\mathcal{K}) \oplus (G_3 \circ M^{72} \circ G_2)(P^i),$$

which means U^i is an affine transformation of $\mathcal{K}$ given P^i and so is u^i. Thus, we can ultimately solve the effective key $\mathcal{K}$ from u^i. The total time complexity of our attack is computed as

$$2^{34} + 2^{|\mathcal{K}|-13} = \begin{cases} 2^{34}, & |\mathcal{K}| < 48 \\ 2^{|\mathcal{K}|-13}, & |\mathcal{K}| \geq 48 \end{cases}$$

The data complexity of our attack is $(2^{34} - 1) \cdot 24 + 128$, i.e. $O(2^{38.6})$, as we need $2^{34} - 1$ frames of the first 24 bits plus one frame of 128 bits. Table 3 compares our attack with existing attacks [15, 11, 7–9] against the two-level Bluetooth E0. Note that the number of required frames completely depends on the frame size in [7] to meet the requirement of data amount. This is, to our best knowledge, the first non-trivial[9] attack against practical E0 with various key length. Notice that when $40 \leq |\mathcal{K}| \leq 80$, our attack offers the best performance over the others.

Table 3. Comparison of our attack with existing attacks against two-level Bluetooth E0

Attack	Precomputation	Time	Frames	Data	Memory
exhaustive search	-	$2^{\lvert\mathcal{K}\rvert-1}$	1	$\lvert\mathcal{K}\rvert$	-
[15]	-	2^{93}	1	2^7	-
[11]	-	2^{113}	1	2^7	-
[7]	-	2^{73}	-	2^{43}	2^{51}
[8]	2^{80}	2^{65}	2	$2^{12.4}$	2^{80}
[9]	2^{80}	2^{70}	45	2^{17}	2^{80}
Our Attack	-	$2^{\lvert\mathcal{K}\rvert-13} + 2^{34}$	2^{34}	$2^{38.6}$	2^{34}

Remark 2. Note that our attack is based on one of the largest two (linearly dependent) biases which is introduced in Eq.(3). As time/data tradeoff, we might also expect to have some other linearly independent biases to be large enough so that the time is decreased at somewhat reasonably increasing cost of data/memory complexities. Nonetheless, using the computation formula of [12], we find none such bias that leads to the data complexity of less than 2^{50}.

Remark 3. As the nonces P^i's are affine transformation of a 26-bit clock and a master device address, our attack requiring much more than 2^{26} frames of keystreams still remains impractical unfortunately.

5 Extended Key-recovery Attack against Two-level E0

5.1 A Partial Key-recovery Attack

Notice that on one hand, each of the four leftmost biased bits on the right-hand side of Eq.(9) is computed only with a certain subset of fixed key bits, the known nonce and the unknown variable FSM initial state; on the other hand, the value of Eq.(9) can be easily predetermined from the left-hand side, after we recover u_t^i's with 2^{34} frames by the distinguisher in Subsection 4.2. Consequently, the well-known technique of *statistical cryptanalysis* leads us to the following approach to advance

[9] in contrast to the brute force attack

our key-recovery attack: supposing we manage to guess one of those four biased bits for all frames by guessing only the related key bits, then, for each frame, we XOR the guess on the biased bit with the predetermined value of Eq.(9) to obtain one bit. Thanks to Eq.(9), this bit shows bias for the right guess and almost balancedness for the wrong guess (which is also called statistical distinguishable); we're able to spot out the right guess of all guesses finally.

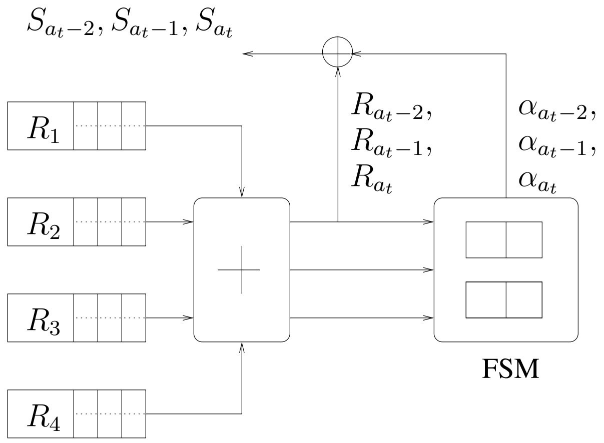

Fig. 2. Computation diagram of $S_{a_t-2}, S_{a_t-1}, S_{a_t}$

More specifically, we observe two important points about E0 FSM state: first, the FSM state at time t always contains the two bits c_t^0, c_{t-1}^0; second, the 4-bit FSM state is updated by its current state together with four current output bits of LFSRs. Therefore, for fixed $t \in \{8k+1, \ldots, 8k+4\}$ and $k \in \{0,1,2\}$, the bit[10] $\bigoplus_{j=0}^{4} \alpha_{a_t-j}^i$ computed from 5 consecutive bits $\alpha_{a_t}^i, \ldots, \alpha_{a_t-4}^i$, is derived from the same subset of $4 \times 3 = 12$ bits of the shared key $\mathcal{K}$ in all frames given P^i together with the unknown frame-dependent FSM state at time $a_t - 3$ (see Fig. 2). We can compute all the possible sequences[11] $\alpha'_{a_t}, \ldots, \alpha'_{a_t-2}$ in according to every possible FSM state for each frame i with $t \in \{8k+1, \ldots, 8k+4\}$ and $k \in \{0,1,2\}$. Within one frame, of all the choices of 12-bit K' of $\mathcal{K}$ and the FSM state, the sequence computed with the right shared 12-bit K and the right FSM state yields $\bigoplus_{j=0}^{4} \alpha_{a_t-j}^i$, which equals $\bigoplus_{j=0}^{4} (\alpha_{a_t-j+8}^i \oplus \alpha_{b_t-j}^i \oplus \alpha_{b_t-j+8}^i \oplus \beta_{t+j}^i)$ with bias λ^4 when xoring with the computable bit $s_t^i \oplus u_t^i$ by Eq.(9); meanwhile, the sequence obtained with the wrong FSM state and/or the wrong shared 12-bit K' is expected to produce a new biased bit $\bigoplus_{j=0}^{4} \alpha'_{a_t-j}$ (with bias λ) which when xoring with $s_t^i \oplus u_t^i$ finally generates a bit with much smaller bias λ^6 that could be approximated by a randomly and uniformly distributed bit. Therefore, we estimate that for every frame, the 12-bit guess K' would yield 2^4 randomly and uniformly distributed bits, except for the correct guess that produces $2^4 - 1 = 15$ randomly and uniformly distributed bits as well as one biased bit (with bias λ^4).

Alternatively, for every frame i, we can guess $4(2+\tau)$ bits K' of $\mathcal{K}$ together with the FSM state at time $a_t - \tau - 2$ to compute consecutively τ bits $\bigoplus_{j=0}^{4} \alpha'_{a_t-j}, \ldots,$

[10] For our convenience, we discuss the first biased bit on the right-hand side of Eq.(9) from now on; however, due to symmetry of the subscripts on the right-hand side of Eq.(9), our discussion also applies to the other three biased bits but the last.

[11] We omit the superscript i and use α'_t to denote the candidate for α_t^i.

$\bigoplus_{j=0}^{4} \alpha'_{a_t-j-\tau+1}$ with $\tau \le 5 - (t \mod 8)$ for fixed $t \in \{8k+1,\ldots,8k+4\}$ and $k \in \{0,1,2\}$. Denote the parameter m as the required number of frames to be discussed later. For the same reason as before, when we xor the τ bits with $s_t^i \oplus u_t^i, \ldots, s_{t+\tau-1}^i \oplus u_{t+\tau-1}^i$, we expect the $16m$ sequences to comply with a truly random distribution $\mathcal{D}_0$ of τ-bit vectors for all wrong guesses K', and the $16m$ sequences for the right guess K to comply with the biased distribution $\mathcal{D}_1$ of τ-bit vectors approximated by

$$\mathcal{D}_1 \approx \frac{\mathcal{D}' + 15\mathcal{D}_0}{16}, \tag{11}$$

where $\mathcal{D}' \stackrel{\text{def}}{=} \mathcal{D}^{\otimes 4}$ with $\otimes$ representing the regular convolutional product (see [12]), and $\mathcal{D}$ is the distribution of $\bigoplus_{j=0}^{4} c^0_{t-j}, \ldots, \bigoplus_{j=0}^{4} c^0_{t-\tau-j+1}$. Note that Eq.(11) means all the biases in $\mathcal{D}'$ dwindles 16 times in $\mathcal{D}_1$, i.e. we have the following relation between the two Walsh coefficients $\hat{\mathcal{D}}_1(x), \hat{\mathcal{D}}'(x)$ of any nonzero τ-bit vector x:

$$\hat{\mathcal{D}}_1(x) = \frac{1}{16}\hat{\mathcal{D}}'(x). \tag{12}$$

Let $f : GF(2)^\tau \to \mathbf{R}$ be a weighted grade for those resultant sequences $\chi^1_{K'}, \ldots, \chi^{16m}_{K'}$ from the guess K'. We accordingly grade each guess K' by

$$G_{K'} = \sum_{j=1}^{16m} f(\chi^j_{K'}). \tag{13}$$

Using analysis of [16] and [2] (see Appendix for complete treatment), we show that with minimal

$$m \approx \frac{\tau+2}{2\tau-1} \cdot 2^{34.5},$$

the score G_K of the right guess tops the chart by choosing $f(x) = \mathcal{D}_1(x) - \frac{1}{16}$ for all $x \in GF(2)^\tau$. Note that with $f(x) = \mathcal{D}'(x)$ we obtain equivalently the same resultant score $G_{K'}$ for all K'. Also, recall that to predetermine $u_t^i, \ldots, u_{t+\tau-1}^i$ we need 2^{34} frames for the distinguisher. Thus we must have

$$m \approx \max\left(2^{34}, \frac{\tau+2}{2\tau-1} \cdot 2^{34.5}\right). \tag{14}$$

Algorithm 1 details the above partial key-recovery attack for $4(2+\tau)$ bits.

5.2 Complexities and Optimization Issues

About the performance of the partial key-recovery attack, it is seen from Algorithm 1 that to recover $4(2+\tau)$ bits, it runs $2^{4(2+\tau)} \cdot m \cdot 2^4(2+\tau) = m(2+\tau) \cdot 2^{12+4\tau}$ times to compute each α'_t for grading $2^{4(2+\tau)}$ candidates. In total, we have to perform $T = m(2+\tau) \cdot 2^{12+4\tau}$ operations, where m is set by Eq.(14).

Additionally, the loop from Line 9 to Line 15 can be done by one operation after precomputation as detailed below, which makes $T = m \cdot 2^{4(2+\tau)}$. During preprocessing, we run through every $y_{a_t-1}, \ldots, y_{a_t-\tau-2}$ (where y_t denotes the Hamming weight of the original four component LFSRs' output bits at time t) to compute the 2^4 possible sequences $\alpha'_{a_t}, \ldots, \alpha'_{a_t-\tau-1}$ which yield 2^4 sequences $b' = \bigoplus_{j=0}^{4} \alpha'_{a_t-j}, \ldots, \bigoplus_{j=0}^{4} \alpha'_{a_t-j-\tau+1}$ accordingly; and then for each τ-bit b'', we increment the counter $\mu^{b''}_{b' \oplus b''}$; last, we build up a table to store $h(y_{a_t-1}, \ldots, y_{a_t-\tau-2}; b'') = \sum_{b'} \mu^{b''}_{b' \oplus b''}(\mathcal{D}_1(b' \oplus b'') - \frac{1}{16})$. The precomputation needs memory $2^\tau \cdot 5^{2+\tau} \approx 2^{3.32\tau+4.6}$ and time $2^4 \cdot 5^{2+\tau} \cdot (2+\tau) \cdot 2^\tau \approx (\tau+2)2^{3.32\tau+8.6}$. After that, in real-time processing, for each frame i, we just compute $b^{i''} = b_0^{i''}, \ldots, b_{\tau-1}^{i''}$ with $b_n^{i''} = u_{t+n}^i \oplus s_{t+n}^i$

Algorithm 1 The extended attack against two-level E0 to recover $4(2+\tau)$ bits

Parameters:
$\mathcal{D}_1$ by Eq.(11)
$\tau \in [1, 4]$
m by Eq.(14)
Input:
keystreams $z_1^i \cdots z_{24}^i$ generated by the same $\mathcal{K}$ together with P^i under two-level E0 for $i = 1, \ldots, m$
Processing:
choose $k \in \{0, 1, 2\}$ and $t \in \{8k+1, \ldots, 8k+4\}$ with $\tau \leq 5 - (t \mod 8)$
for $4(2+\tau)$ bits K' of $\mathcal{K}$ that are used to compute $\alpha'_{a_t-\tau-1}, \ldots, \alpha'_{a_t}$ and are consistent with previously recovered bits **do**
  initialize $G_{K'}$ to zero
  for $i = 1, \ldots, m$ **do**
    initialize counters $\mu_0, \mu_1, \ldots, \mu_{2^\tau-1}$ to zero
    for all 4-bit FSM state at time $a_t - \tau - 2$ **do**
      compute $\alpha'_{a_t-\tau-1}, \ldots, \alpha'_{a_t}$
      compute $b = b_0, \ldots, b_{\tau-1}$ with $b_n = u_{t+n}^i \oplus s_{t+n}^i \oplus \bigoplus_{j=0}^{4} \alpha'_{a_t-j-n}$
      increment μ_b
    end for
    increment $G_{K'}$ by $\sum_b \mu_b(\mathcal{D}_1(b) - \frac{1}{16})$
  end for
  add the pair $(G_{K'}, K')$ into the list
end for
find the largest G_K in the list and output K

for $n = 0, \ldots, \tau - 1$, deduce $y_{a_t-1}^i, \ldots, y_{a_t-\tau-2}^i$ from K', P^i and increment $G_{K'}$ by $h(y_{a_t-1}^i, \ldots, y_{a_t-\tau-2}^i; b^{i''})$. Thus, we get $T = m \cdot 2^{4(2+\tau)}$.

Moreover, when $2^{4(2+\tau)} \cdot 2^\tau \leq m$, i.e. $2^{8+5\tau} \leq m$, we can further reduce T down to $m + 2^{16+9\tau}$. Notice that it is the same subset of $4(2+\tau)$-bit Ω^i of P^i that is used to compute $y^i = (y_{a_t-1}^i, \ldots, y_{a_t-\tau-2}^i)$ with K'. For convenience, let $g : GF(2)^L \to GF(2)^{4(2+\tau)}$ map P^i to Ω^i. We precompute a table $h'(\Omega, q)$ for every $4(2+\tau)$-bit Ω and τ-bit q defined by:

$$h'(\Omega, q) = \sum_{i=1}^{m} \mathbf{1}_{\Omega = g(P^i), q = u_t^i \oplus s_t^i}$$

with $u^i = (u_t^i, \ldots, u_{t+\tau-1}^i)$ and $s^i = (s_t^i, \ldots, s_{t+\tau-1}^i)$. This takes time $O(m)$ with memory $O(2^{8+5\tau})$. Recall that u^i is determined independent of K' by the distinguisher in Subsection 4.2, and u^i, s^i, y^i completely determine how to increment $G_{K'}$ for frame i, i.e. by $h(y^i; u^i \oplus s^i)$ from last paragraph. So, in real-time processing, for every K', instead of processing frame by frame to update $G_{K'}$, we simply go through every $(8 + 5\tau)$-bit pair (Ω, q), deduce $y = (y_{a_t-1}, \ldots, y_{a_t-\tau-2})$ from Ω and K', then increment $G_{K'}$ by $h'(\Omega, q)h(y; q)$. We reach the time complexity $T = m + 2^{4(2+\tau)} \cdot 2^{8+5\tau} = m + 2^{16+9\tau}$ for $2^{8+5\tau} \leq m$.

To summarize, we have $T = m + 2^{4(2+\tau)} \cdot \min(m, 2^{8+5\tau})$. Table 4 lists the best complexities of our partial key-recovery attack corresponding to $\tau = 1, \ldots, 4$. Note that the success probability of the attack in the table is estimated according to the hypothesis test result of Eq.(11), i.e. the percentage of the $4(2+\tau)$-bit keys to generate a non-uniformly distributed sequence $\alpha'_{a_t-\tau-3}, \ldots, \alpha'_{a_t}$ with all the possible FSM state.

Table 4. Performance of our partial key-recovery attack against two-level E0

τ	Frames m	Time T	Prob. of Success	recovered key bits $4(2+\tau)$
1	2^{36}	2^{36}	50.8%	12
2	2^{35}	2^{35}	87.0%	16
3	$2^{34.5}$	2^{43}	99.0%	20
4	$2^{34.3}$	2^{52}	99.9%	24

5.3 The Overall Key-recovery Attack

Now we discuss how we proceed with the optimized Algorithm 1 to recover the full $\mathcal{K}$. With $\tau = 2$ and fixed k, we independently run Algorithm 1 three times with $t = 8k + 1, \ldots, 8k + 3$. And we expect at least two successes out of three runs. After checking consistency of all the overlapping bits for every possible pair of the algorithm outputs, we identify all the successful runs and obtain the minimum $16 + 4 = 20$ key bits.

We can easily adjust Algorithm 1 to target at any of the middle three biased bits on the right-hand side of Eq.(9) to recover 16 bits. With each modified partial key-recovery algorithm, we repeat previous procedure to recover minimum 20 bits. In total, we are sure to gather $4 \times 20 = 80$ bits. Since we already have 12 bits by the distinguisher, we finally exhaustively search the remaining $L - 80 - 12 = 36$ bits within one frame. Algorithm 2 gives the abstract strategy of our complete attack. Therefore, to recover L-bit $\mathcal{K}$, our key-recovery attack works on $m = 2^{35}$ frames, in time $24m + 4 \times 3 \times 2^{35} \approx 2^{40}$. The comparison of our attack with the best known attacks [7–9] against two-level E0 for $|\mathcal{K}| = L$ is available in Table 5.

Algorithm 2 The abstract of the complete attack against two-level E0 for $|\mathcal{K}| = L$

Parameters:
m by Eq.(14)
Input:
m frames of 24-bit keystreams generated by the same $\mathcal{K}$ together with known nonces under two-level E0
Processing:
for each of the leftmost four biased bits on right-hand side of Eq.(9) **do**
  choose $k \in \{0, 1, 2\}$ and set $\tau = 2$
  for $t = 8k + 1, 8k + 2, 8k + 3$ **do**
    use the optimized partial key-recovery attack to obtain 16 bits
  end for
  checking consistency among pairs of those 16 bits to obtain minimum 20-bit $\mathcal{K}$
end for
exhaustively search the remaining 36 key bits within one frame
output the L-bit $\mathcal{K}$

6 Conclusion

In this paper, based on one of the two largest biases inside the FSM within one-level E0, for the first time, we identify the bias at two-level E0 due to a resynchronization flaw in Bluetooth E0. Unlike the traditional approach to find the bias, the characterized bias *does not* involve the precomputation of the multiple polynomial

Table 5. Comparison of our attack with the best attacks [7–9] against two-level E0 for $|\mathcal{K}| = L$

Attack	Precomputation	Time	Frames	Data	Memory
[7]	-	2^{73}	-	2^{43}	2^{51}
[8]	2^{80}	2^{65}	2	$2^{12.4}$	2^{80}
[9]	2^{80}	2^{70}	45	2^{17}	2^{80}
Our Attack	-	2^{40}	2^{35}	$2^{39.6}$	2^{35}

with low weight. Second, to utilize the identified bias, we develop a novel attack to directly recover the original encryption key for two-level E0 without reconstructing the initial state of E0 at the second level. Our key-recovery attack works with 2^{40} simple operations given the first 24 bits of 2^{35} frames. Compared with all existing attacks [15, 11, 7–9] against two-level Bluetooth E0, this is the best one so far, although the impossibly high amount of frames thwarts our attack to be practical. It remains an open challenge to decrease the data complexity with practical time and memory complexities. Finally, our attack illustrates the theory of statistical attacks in [2, 16] with an example which is not based on linear cryptanalysis.

Acknowledgments

We owe a lot grateful thanks to Anne Canteaut and Willi Meier. And we would also like to thank the anonymous reviewers for many helpful suggestions.

References

1. Frederik Armknecht, Matthias Krause, *Algebraic Attacks on Combiners with Memory*, Advances on Cryptography - CRYPTO 2003, Lecture Notes in Computer Science, vol.2729, D. Boneh Ed., Springer-Verlag, pp. 162-175, 2003
2. Thomas Baignères, Pascal Junod, Serge Vaudenay, *How Far Can We Go Beyond Linear Cryptanalysis?*, in these proceedings
3. Bluetooth^{TM}, *Bluetooth Specification*, version 1.2, pp. 903-948, November, 2003, available at `http://www.bluetooth.org`
4. Philippe Chose, Antoine Joux, Michel Mitton, *Fast Correlation Attacks: An Algorithmic Point of View*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol.2332, L. R. Knudsen Ed., Springer-Verlag, pp. 209-221, 2002
5. Nicolas T. Courtois, *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Advances on Cryptography - CRYPTO 2003, Lecture Notes in Computer Science, vol.2729, D. Boneh Ed., Springer-Verlag, pp. 176-194, 2003
6. Patrik Ekdahl, Thomas Johansson, *Some Results on Correlations in the Bluetooth Stream Cipher*, Proceedings of the 10th Joint Conference on Communications and Coding, Austria, 2000
7. Scott Fluhrer, Stefan Lucks, *Analysis of the E0 Encryption System*, Selected Areas in Cryptography- SAC 2001, Lecture Notes in Computer Science, vol. 2259, S. Vaudenay and A. Youssef Eds., Springer-Verlag, pp. 38-48, 2001
8. Scott Fluhrer, *Improved Key Recovery of Level 1 of the Bluetooth Encryption System*, available at `http://eprint.iacr.org/2002/068`
9. Jovan Dj. Golić, Vittorio Bagini, Guglielmo Morgari, *Linear Cryptanalysis of Bluetooth Stream Cipher*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol. 2332, L. R. Knudsen Ed., Springer-Verlag, pp. 238-255, 2002
10. Miia Hermelin, Kaisa Nyberg, *Correlation Properties of the Bluetooth Combiner*, Information Security and Cryptology- ICISC'99, Lecture Notes in Computer Science, vol. 1787, JooSeok. Song Ed., Springer-Verlag, pp. 17-29, 2000

11. Matthias Krause, *BDD-Based Cryptanalysis of Keystream Generators*, Advances in Cryptology - EUROCRYPT 2002, Lecture Notes in Computer Science, vol. 2332, L. R. Knudsen Ed., Springer-Verlag, pp. 222-237, 2002
12. Yi Lu, Serge Vaudenay, *Faster Correlation Attack on Bluetooth Keystream Generator E0*, Advances on Cryptography - CRYPTO 2004, Lecture Notes in Computer Science, vol.3152, M. Franklin Ed., Springer-Verlag, pp. 407-425, 2004
13. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology - EUROCRYPT'93, Lecture Notes in Computer Science, vol.765, Springer-Verlag, pp. 386-397, 1993
14. Alfred J. Menezes, Paul C. van. Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC, 1996
15. Markku Saarinen, *Re: Bluetooth and E0*, Posted at `sci.crypt.research`, 02/09/00
16. Serge Vaudenay, *An Experiment on DES - Statistical Cryptanalysis*, Proceedings of the 3rd ACM Conferences on Computer Security, pp. 139-147, 1996

Appendix

All our analysis here is similar with [16] and inspired by [2]. First, by Eq.(13), we have

$$\mathrm{Exp}(G_{K'}) = \frac{16m}{2^\tau} \sum_b f(b)$$

for a random wrong guess K', and by Eq.(11) we have

$$\mathrm{Exp}(G_K) = \frac{15m}{2^\tau} \sum_b f(b) + m \sum_b \mathcal{D}'(b) f(b),$$

for the right K. Hence,

$$\Delta \mathrm{Exp}(G_{K'}) = m \sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau} \right) f(b). \tag{15}$$

Meanwhile, we compute the variance of $G_{K'}$ as

$$\mathrm{Var}(G_{K'}) = \frac{16m}{2^\tau} \sum_b (f(b))^2 - \frac{16m}{2^{2\tau}} \left(\sum_b f(b) \right)^2. \tag{16}$$

We can estimate the rank of G_K over all possible $G_{K'}$ by

$$\mathrm{Exp}(\mathrm{Rank}_{G_K}) \approx 2^{4(2+\tau)} \Phi \left(- \frac{\Delta \mathrm{Exp}(G_{K'})}{\sqrt{2 \mathrm{Var}(G_{K'})}} \right). \tag{17}$$

In order to achieve the top rank for G_K, we see that the fraction

$$\frac{\Delta \mathrm{Exp}(G_{K'})}{\sqrt{\mathrm{Var}(G_{K'})}} \tag{18}$$

must be large enough. This can be satisfied as long as the number m of available frames is sufficiently large. However, aiming at a practical attack, we are concerned with the question of how to choose f in order to minimize m under the constraint of the top rank G_K. In order to maximize the fraction (18), we first maximize the numerator with the constraint that the denominator is a constant and then try to maximize the fraction over all the solutions. Define the multivariate polynomial

$$g_f = m \sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau} \right) f(b) + \frac{16m\gamma}{2^\tau} \sum_b (f(b))^2 - \frac{16m\gamma}{2^{2\tau}} \left(\sum_b f(b) \right)^2.$$

Using Lagrange's multiplier, we have

$$\frac{\partial g_f}{\partial f(b)} = m\left(\mathcal{D}'(b) - \frac{1}{2^\tau}\right) + \frac{32m\gamma}{2^\tau} f(b) - \frac{32m\gamma}{2^{2\tau}} \sum_{b'} f(b') = 0, \tag{19}$$

for all $b \in GF(2)^\tau$. From Eq.(19) we infer that

$$\frac{f(b) - f(b')}{\mathcal{D}'(b) - \mathcal{D}'(b')} = \text{const.}$$

for all $b \neq b'$. Therefore we have a universal expression of f as

$$\frac{f(b) - \text{const.}}{\mathcal{D}'(b)} = \text{const}'.$$

for all $b \in GF(2)^\tau$, which yields the same quantity of (18) regardless of the constants in f. So the easiest way to define f could be $f(b) = \mathcal{D}'(b) - \frac{1}{2^\tau}$ for all $b \in GF(2)^\tau$. Then Eq.(15) reduces to

$$\begin{aligned} \Delta\text{Exp}(G_{K'}) &= m \sum_b \left(\mathcal{D}'(b) - \frac{1}{2^\tau}\right)^2 \\ &= \frac{m}{2^\tau} \sum_{b \neq \mathbf{0}} \left(\hat{\mathcal{D}}(b)\right)^8 \\ &\approx \frac{m}{2^\tau}(2\tau - 1)\lambda^8. \end{aligned}$$

On the other hand Eq.(16) reduces to

$$\text{Var}(G_{K'}) \approx \frac{16m}{2^{2\tau}}(2\tau - 1)\lambda^8.$$

So we deduce from Eq.(17) that

$$\begin{aligned} \text{Exp}(\text{Rank}_{G_K}) &\approx 2^{4(2+\tau)} \Phi\left(-\frac{\lambda^4}{4}\sqrt{\frac{m(2\tau-1)}{2}}\right) \\ &\approx \frac{2^{4(2+\tau)}}{\sqrt{2\pi}} e^{-\frac{m(2\tau-1)}{64}\lambda^8}. \end{aligned}$$

This means Rank_{G_K} is expected to top the chart with

$$m \approx \frac{256(2+\tau)\log 2}{\lambda^8(2\tau - 1)} \approx \frac{\tau + 2}{2\tau - 1} \cdot 2^{34.5}.$$