A Linearization Attack on the Bluetooth Key Stream Generator

Frederik Armknecht *

University of Mannheim
68131 Mannheim, Germany
armknecht@th.informatik.uni-mannheim.de

Abstract. In this paper we propose an attack on the key stream generator underlying the encryption system E_0 used in the Bluetooth specification. We show that the initial value can be recovered by solving a system of nonlinear equations of degree 4 over the finite field GF(2). This system of equations can be transformed by linearization into a system of linear equations with at most $2^{24.056}$ unknowns. To our knowledge, this is the best attack on the key stream generator underlying the E_0 yet.

1 Introduction

The encryption system E_0, which is the encryption system used in the Bluetooth specification [1], is a two level system. In both levels the same key stream generator (KSG) is used. In this paper, we propose an attack on the underlying KSG. We regard its initial value as the secret key and try to recover it if only output bits are given. The result of the paper is that this can be done by solving a system of nonlinear equations (SNE) of degree 4 over the finite field GF(2). This system of equations can be transformed by linearization into a system of linear equations (SLE) with at most $2^{24.056}$ unknowns.

The paper is structured as follows. In Section 2, the E_0 key stream generator is described. In Section 3, we will show how to get for each clock t a new equation of degree 4 in the unknown initial value bits which holds with probability 1. Solving this SNE provides the initial value of the key stream generator. In Section 4, we discuss how the system of nonlinear equations can be solved and which obstacles have to be overcome. The attack is compared to other published attacks. Section 5 concludes the paper.

2 Description of the Bluetooth key stream generator

The E_0 encryption system is used in the Bluetooth specification [1] for wireless communication. The encryption is done in two levels. In the first level, a key and a nonce are given to a key stream generator which produces the key for the

* This work has been supported by grant 620307 of the DFG (German Research Foundation)

second level. In the second level, the same key stream generator uses the second level key to generate the key stream bits which are XORed to the plaintext bits. In this paper, we will concentrate on the key stream generator (KSG).

The KSG consists of four regularly clocked Linear Feedback Shift Registers (LFSR) and four memory bits. With each clock, an output bit z_t is produced depending on the outputs a_t, b_t, c_t and d_t of the four LFSRs and the four memory bits $(Q_t, P_t, Q_{t-1}, P_{t-1})$. Then, the next memory bits $\mathcal{C}_{t+1} := (Q_{t+1}, P_{t+1})$ are calculated and so on. The exact definitions are

$$z_t = a_t \oplus b_t \oplus c_t \oplus d_t \oplus P_t \tag{1}$$

$$\mathcal{C}_{t+1} = \mathcal{S}_{t+1} \oplus \mathcal{C}_t \oplus T(\mathcal{C}_{t-1}) \tag{2}$$

where $T(x_1, x_0) := (x_0, x_0 \oplus x_1)$ and

$$\mathcal{S}_{t+1} = (\mathcal{S}^1_{t+1}, \mathcal{S}^0_{t+1}) = \left\lfloor \frac{a_t + b_t + c_t + d_t + 2Q_t + P_t}{2} \right\rfloor \tag{3}$$

The values for $\mathcal{C}_0$ and $\mathcal{C}_1$ and the contents of the LFSR must be set before the start, the other values will then be calculated. The LFSRs have the lengths $n_1 = 25$, $n_2 = 31$, $n_3 = 33$, and $n_4 = 39$, and $n = n_1 + n_2 + n_3 + n_4 = 128$ is the size of the secret key.

3 Building a system of nonlinear equations for the key stream generator

In this section, we will show that the initial state of the keystream generator can be reconstructed by solving a system of nonlinear equations of degree 4. With each clock t, the new output z_t is produced and the next memory bits Q_{t+1} and P_{t+1} are computed. This is done by the following equations (see Appendix A for the proof):

$$z_t = a_t \oplus b_t \oplus c_t \oplus d_t \oplus P_t \tag{4}$$

$$Q_{t+1} = \Pi_4(t) \oplus \Pi_3(t)P_t \oplus \Pi_2(t)Q_t \oplus \Pi_1(t)P_tQ_t \oplus Q_t \oplus P_{t-1} \tag{5}$$

$$P_{t+1} = \Pi_2(t) \oplus \Pi_1(t)P_t \oplus Q_t \oplus Q_{t-1} \oplus P_{t-1} \oplus P_t \tag{6}$$

Here, a_t, b_t, c_t, d_t are the outputs of the four LFSRs and $\Pi_k(t)$ is the XOR over all possible products in $\{a_t, b_t, c_t, d_t\}$ of degree k. E. g.,

$$\begin{aligned} \Pi_1(t) &= a_t \oplus b_t \oplus c_t \oplus d_t \\ \Pi_2(t) &= a_tb_t \oplus a_tc_t \oplus a_td_t \oplus b_tc_t \oplus b_td_t \oplus c_td_t \\ &\vdots \end{aligned}$$

If we define the following additional variables

$$\begin{aligned} A(t) &= \Pi_4(t) \oplus \Pi_3(t)P_t \oplus P_{t-1} \\ B(t) &= \Pi_2(t) \oplus \Pi_1(t)P_t \oplus 1 \end{aligned}$$

equations (5) and (6) can be rewritten to

$$Q_{t+1} = A(t) \oplus B(t)Q_t \tag{7}$$
$$P_{t+1} = B(t) \oplus 1 \oplus P_{t-1} \oplus P_t \oplus Q_t \oplus Q_{t-1} \tag{8}$$

By multiplying (7) with $B(t)$ we get another equation

$$0 = B(t)(A(t) \oplus Q_t \oplus Q_{t+1}) \tag{9}$$

Equation (8) is equivalent to

$$Q_t \oplus Q_{t-1} = B(t) \oplus 1 \oplus P_{t-1} \oplus P_t \oplus P_{t+1} \tag{10}$$

Now we insert (10) into (9) with index $t+1$ instead of t and get

$$0 = B(t)\,(A(t) \oplus B(t+1) \oplus 1 \oplus P_t \oplus P_{t+1} \oplus P_{t+2})$$

Using (4) we eliminate all memory bits in the equation and get the following equation which holds for every clock t:

$$\begin{aligned}
0 = 1 &\oplus z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2} \\
&\oplus \Pi_1(t) \cdot (z_t z_{t+2} \oplus z_t z_{t+1} \oplus z_t z_{t-1} \oplus z_{t-1} \oplus z_{t+1} \oplus z_{t+2} \oplus 1) \\
&\oplus \Pi_2(t) \cdot (1 \oplus z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2}) \oplus \Pi_3(t) z_t \oplus \Pi_4(t) \\
&\oplus \Pi_1(t-1) \oplus \Pi_1(t-1)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_1(t-1)\Pi_2(t) \\
&\oplus \Pi_1(t+1) z_{t+1} \oplus \Pi_1(t+1)\Pi_1(t) z_{t+1}(1 \oplus z_t) \oplus \Pi_1(t+1)\Pi_2(t) z_{t+1} \\
&\oplus \Pi_2(t+1) \oplus \Pi_2(t+1)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_2(t+1)\Pi_2(t) \\
&\oplus \Pi_1(t+2) \oplus \Pi_1(t+2)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_1(t+2)\Pi_2(t)
\end{aligned}$$

With every clock t we get an additional equation which holds with probability 1. This makes it possible to compile a system of nonlinear equations degree 4. As the unknowns are just the bits of the initial value of the four LFSRs, the secret key can be recovered by solving the SNE.

Remark that in an LFSR all output bits are a linear combination of the initial values bits so the number of possibly occurring terms is limited, the upper bound being $17,440,047 \approx 2^{24.056}$ (see appendix B for details). In the following section, we will discuss how this knowledge can be used to solve the SNE.

4 Solving the system of nonlinear equations

We will now concentrate on the question how to solve the system of nonlinear equations of the previous section.

The simplest method is the so called linearization algorithm. Assume that a system of nonlinear equations (SNE) with l linearly independent equations is given. Let T be the number of all different terms occurring in the SNE. If we replace every occurring term by a new variable, we get a system of linear equations (SLE) with T unknowns. If $l = T$, then the SNE can be solved by the usual methods. Obviously, the algorithm can be only successful if enough linearly independent equations are present.

The maximum number of possibly occurring terms in the SNE is $T = 17,440,047 \approx 2^{24.056}$ (see appendix B for details). The fastest practical method we are aware of to solve a system of linear equations is Strassen's algorithm [8]. It requires about $7 \cdot T^{\log_2 7}$ operations. In this case, the secret key can be recovered with approximately $2^{70.341}$ operations if at least $2^{24.056}$ linearly independent equations are available. Observe that the system of linear equations can be constructed in precomputation time. For each attack, it is only necessary to insert the actual values of the key stream bits z_t.

Note that the best known attack against the E_0 was proposed by *Krause* [5] with time effort 2^{77}, given only 128 known key stream bits. The attack by *Fluhrer* and *Lucks* [4] needs about 2^{73} operations if 2^{43} bits are available.

Let us face again the problem how to get enough linearly independent equations. With each clock t, we get a new equation in the bits of the secret key. Hence, one strategy is to produce new equations until the number of linearly independent equations is equal to the number of terms. Obviously, we have to clock at least T times to achieve this goal. The question is whether we have to clock more often. Until now, there is no satisfying answer to this question. Our assumption is that approximately T clocks should be enough, meaning that about $2^{24.056}$ key stream bits would be sufficient to mount the attack. Tests confirmed our assumption for simpler systems of equations. For the future, tests with more E_0 like systems of equations are planned.

If not enough linearly independent equations are available, improved versions of linearization may succeed. We give a brief overview over existing methods.

In [6], Shamir and Kipnis presented the relinearization algorithm for solving a system of quadratic equations over the finite field GF(2). They expect that SNEs can be solved with this algorithm even if l is only one fifth of T.

The XL algorithm (XL stands for eXtended Linearization), introduced at Eurocrypt'00 [7] by Shamir, Patarin, Courtois and Klimov, can be seen as a simplified and improved version of relinearization. Given a system of nonlinear equations, additional equations (and possibly new terms also) may be gained by multiplying all equations with all possible monoms of some degree $\leq D$. If we are lucky we get enough new equations to achieve $l = T$. It is an open question under which conditions the XL algorithm is successful. The authors proved that XL is as least as powerful as relinearization. Notice that the XL algorithm was used by Courtois to attack the Toyocrypt cipher [2].

In [3], the XSL algorithm, an extension of XL which uses the sparsity of SNEs, is introduced by Courtois and Pieprzyk. XSL stands for eXtended Sparse Linearization. In the XSL algorithm, we multiply the equations by carefully selected monomials. The idea is to use products of monomials that already appear in the SNE. In their paper, they discuss how the XSL algorithm may be useful to mount attacks on AES and Serpent.

Each algorithm may be applicable to our SNE. Therefore, we expect that the key stream generator can be broken with our attack at least in theory. Notice that all described methods have been analyzed only heuristically. Hence, it is neither sure that the E_0 system of equations can be solved nor how many key stream bits are necessary. On the other side, there is no proof that the attack fails. We hope that computer simulations will partly confirm our assumption.

The question if a similar SNE for the whole two level E_0 encryption system exists will be investigated in the future.

5 Conclusion

We discussed how the initial value of the key stream generator used in the E_0 stream cipher can be obtained by solving a system of nonlinear equations of degree 4. If enough linearly independent equations are provided, the system can be solved by simple linearization. As the number of occurring terms is limited by $T \approx 2^{24.056}$, the secret key can be recovered in this case with $2^{70.341}$ work using Strassen's algorithm.

It is an open question how many keystream bits are needed to get enough linearly independent equations. We assume that it should not be much more than T but at the moment, meaningful tests are missing. Using better algorithms than linearization (e. g. XL, XSL) may reduce the number of needed key stream bits significantly.

Obviously, the next step is to test whether a similar system of equations can be found for the whole E_0 cipher.

Acknowledgment

The author would like to thank Erik Zenner, Stefan Lucks and Matthias Krause for helpful comments and discussions.

References

1. Bluetooth SIG, *Specification of the Bluetooth system*, Version 1.1, 1 February 22, 2001, available at http://www.bluetooth.com/.
2. Nicolas Courtois: *Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt*, http://eprintiacr.org/2002/87.ps. To appear in the Proceedings of ICISC '02, Springer LNCS 2587.
3. Nicolas Courtois, Josef Pieprzyk: *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*, Proceedings of Asiacrypt '02, Springer LNCS 2501, 2002, pp. 267-287.
4. Scott R. Fluhrer, Stefan Lucks: *Analysis of the E0 Encryption System*, Proceedings of Selected Areas of Cryptography '01, Springer LNCS 2259, 2001, pp. 38-48.
5. Matthias Krause: *BDD-Based Cryptanalysis of Keystream Generators*; Proceedings of Eurocrypt '02, Springer LNCS 2332, 2002, pp. 222-237.
6. Adi Shamir, Aviad Kipnis: *Cryptanalysis of the HFE Public Key Cryptosystem*; Proceedings of Crypto '99, Springer LNCS 1666, 1999, pp. 19-30.

7. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov: *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Proceedings of Eurocrypt '00, Springer LNCS 1807, pp. 392-407.
8. Volker Strassen: *Gaussian Elimination is Not Optimal*; Numerische Mathematik, vol 13, pp 354-356, 1969.

A The equations for Q_{t+1} and P_{t+1}

In this section we prove the correctness of equations (5) resp. (6) for Q_{t+1} resp. P_{t+1}. Let us recall equation (2) for $\mathcal{C}_{t+1}$

$$\begin{aligned}\mathcal{C}_{t+1} &= (Q_{t+1}, P_{t+1}) && (11)\\ &= \mathcal{S}_{t+1} \oplus \mathcal{C}_t \oplus T(\mathcal{C}_{t-1}) && (12)\\ &= (\mathcal{S}^1_{t+1} \oplus Q_t \oplus P_{t-1}, \mathcal{S}^0_{t+1} \oplus P_t \oplus Q_{t-1} \oplus P_{t-1}) && (13)\end{aligned}$$

where

$$\mathcal{S}_{t+1} = (\mathcal{S}^1_{t+1}, \mathcal{S}^0_{t+1}) = \left\lfloor \frac{a_t + b_t + c_t + d_t + 2Q_t + P_t}{2} \right\rfloor \tag{14}$$

Let f_0 resp. f_1 be the two boolean functions for which the following equations hold

$$\mathcal{S}^i_{t+1} = f_i(a_t, b_t, c_t, d_t, Q_t, P_t) \tag{15}$$

for $i \in \{0, 1\}$. It is easy to find f_0 and f_1 with the help of computers. If we write down f_0 and f_1 in algebraic normal form, we get

$$\begin{aligned}f_1 &= \Pi_4(t) \oplus \Pi_3(t)P_t \oplus \Pi_2(t)Q_t \oplus \Pi_1(t)P_tQ_t && (16)\\ f_0 &= \Pi_2(t) \oplus \Pi_1(t)P_t \oplus Q_t && (17)\end{aligned}$$

See section 3 for the definition of $\Pi_k(t)$. In table 1 f_0 and f_1 are evaluated for all possible inputs and compared with $\mathcal{S}_{t+1}$. It is easy to see that f_0 and f_1 fulfill the requirements. Together with (13) we get the following expressions for Q_{t+1} and P_{t+1}

$$\begin{aligned}Q_{t+1} &= S^1_{t+1} \oplus Q_t \oplus P_{t-1} && (18)\\ &= \Pi_4(t) \oplus \Pi_3(t)P_t \oplus \Pi_2(t)Q_t \oplus \Pi_1(t)P_tQ_t \oplus Q_t \oplus P_{t-1} && (19)\\ P_{t+1} &= \mathcal{S}^0_{t+1} \oplus P_t \oplus Q_{t-1} \oplus P_{t-1} && (20)\\ &= \Pi_2(t) \oplus \Pi_1(t)P_t \oplus Q_t \oplus Q_{t-1} \oplus P_t \oplus P_{t-1} && (21)\end{aligned}$$

Table 1. f_0 and f_1 evaluated for all possible inputs and compared with $\mathcal{S}_{t+1}$

a_t	b_t	c_t	d_t	Q_t	P_t	S_{t+1}	f_1	f_0		a_t	b_t	c_t	d_t	Q_t	P_t	S_{t+1}	f_1	f_0
0	0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0		1	0	0	0	0	1	1	0	1
0	0	0	0	1	0	1	0	1		1	0	0	0	1	0	1	0	1
0	0	0	0	1	1	1	0	1		1	0	0	0	1	1	2	1	0
0	0	0	1	0	0	0	0	0		1	0	0	1	0	0	1	0	1
0	0	0	1	0	1	1	0	1		1	0	0	1	0	1	1	0	1
0	0	0	1	1	0	1	0	1		1	0	0	1	1	0	2	1	0
0	0	0	1	1	1	2	1	0		1	0	0	1	1	1	2	1	0
0	0	1	0	0	0	0	0	0		1	0	1	0	0	0	1	0	1
0	0	1	0	0	1	1	0	1		1	0	1	0	0	1	1	0	1
0	0	1	0	1	0	1	0	1		1	0	1	0	1	0	2	1	0
0	0	1	0	1	1	2	1	0		1	0	1	0	1	1	2	1	0
0	0	1	1	0	0	1	0	1		1	0	1	1	0	0	1	0	1
0	0	1	1	0	1	1	0	1		1	0	1	1	0	1	2	1	0
0	0	1	1	1	0	2	1	0		1	0	1	1	1	0	2	1	0
0	0	1	1	1	1	2	1	0		1	0	1	1	1	1	3	1	1
0	1	0	0	0	0	0	0	0		1	1	0	0	0	0	1	0	1
0	1	0	0	0	1	1	0	1		1	1	0	0	0	1	1	0	1
0	1	0	0	1	0	1	0	1		1	1	0	0	1	0	2	1	0
0	1	0	0	1	1	2	1	0		1	1	0	0	1	1	2	1	0
0	1	0	1	0	0	1	0	1		1	1	0	1	0	0	1	0	1
0	1	0	1	0	1	1	0	1		1	1	0	1	0	1	2	1	0
0	1	0	1	1	0	2	1	0		1	1	0	1	1	0	2	1	0
0	1	0	1	1	1	2	1	0		1	1	0	1	1	1	3	1	1
0	1	1	0	0	0	1	0	1		1	1	1	0	0	0	1	0	1
0	1	1	0	0	1	1	0	1		1	1	1	0	0	1	2	1	0
0	1	1	0	1	0	2	1	0		1	1	1	0	1	0	2	1	0
0	1	1	0	1	1	2	1	0		1	1	1	0	1	1	3	1	1
0	1	1	1	0	0	1	0	1		1	1	1	1	0	0	2	1	0
0	1	1	1	0	1	2	1	0		1	1	1	1	0	1	2	1	0
0	1	1	1	1	0	2	1	0		1	1	1	1	1	0	3	1	1
0	1	1	1	1	1	3	1	1		1	1	1	1	1	1	3	1	1

B The number of terms

In this section we estimate the maximum number T of different terms in the equations system. With each clock t the following equation is produced

$$\begin{aligned}0 = 1 &\oplus z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2}\\ &\oplus \Pi_1(t) \cdot (z_t z_{t+2} \oplus z_t z_{t+1} \oplus z_t z_{t-1} \oplus z_{t-1} \oplus z_{t+1} \oplus z_{t+2} \oplus 1)\\ &\oplus \Pi_2(t) \cdot (1 \oplus z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2}) \oplus \Pi_3(t) z_t \oplus \Pi_4(t)\\ &\oplus \Pi_1(t-1) \oplus \Pi_1(t-1)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_1(t-1)\Pi_2(t)\\ &\oplus \Pi_1(t+1) z_{t+1} \oplus \Pi_1(t+1)\Pi_1(t) z_{t+1}(1 \oplus z_t) \oplus \Pi_1(t+1)\Pi_2(t) z_{t+1}\\ &\oplus \Pi_2(t+1) \oplus \Pi_2(t+1)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_2(t+1)\Pi_2(t)\\ &\oplus \Pi_1(t+2) \oplus \Pi_1(t+2)\Pi_1(t)(1 \oplus z_t) \oplus \Pi_1(t+2)\Pi_2(t)\end{aligned}$$

As we can see, every occurring term has to be one of the following types

$$\begin{aligned}&a, b, c, d, ab, ac, ad, bc, bd, cd, abc, acd, abd, bcd, abcd, aa'bc, aa'cd, aa'bd, bb'ac,\\ &bb'cd, bb'ad, cc'ab, cc'ad, cc'bd, dd'ab, dd'ac, dd'bc, aa'bb', aa'cc', aa'dd', bb'cc', bb'dd',\\ &cc'dd', aa'b, aa'c, aa'd, bb'a, bb'c, bb'd, cc'a, cc'b, cc'd, dd'a, dd'b, dd'c, aa', bb', cc', dd'\end{aligned}$$

Here, $a, a' \in \{a_1, \ldots, a_{n_1}\}$ with $a \neq a'$, etc. In table 2 the number of possible terms for each type is presented depending on the values n_1, n_2, n_3, and n_4. In addition, we give for each type one product in which it can occur. Note, that some terms may occur in other products too[1]. Of course, these types have to be counted only once. The sum is the number of possible terms T. In E_0, it is $n_1 = 25$, $n_2 = 31$, $n_3 = 33$ and $n_4 = 39$, so $T = 17{,}440{,}047$, which is approximately $2^{24.056}$.

[1] For example, a term of type abc can occur in $\Pi_1(t)\Pi_2(t')$ and in $\Pi_2(t)\Pi_2(t')$

Table 2. All possible terms and their number depending on n_1, n_2, n_3 and n_4

type	occur in	number
a,b,c,d	$\Pi_1(t)$	$n_1+n_2+n_3+n_4$
ab,ac,ad,bc,bd,cd	$\Pi_2(t)$	$n_1(n_2+n_3+n_4)+n_2(n_3+n_4)+n_3n_4$
abc,acd,abd,bcd	$\Pi_3(t)$	$n_1(n_2n_3+n_2n_4+n_3n_4)+n_2n_3n_4$
$abcd$	$\Pi_4(t)$	$n_1n_2n_3n_4$
aa',bb',cc',dd'	$\Pi_1(t)\cdot\Pi_1(t')$	$n_1(n_1-1)+n_2(n_2-1)+n_3(n_3-1)+n_4(n_4-1)$
$aa'b,aa'c,aa'd$	$\Pi_1(t)\cdot\Pi_2(t')$	$n_1(n_1-1)(n_2+n_3+n_4)$
$bb'a,bb'c,bb'd$	$\Pi_1(t)\cdot\Pi_2(t')$	$n_2(n_2-1)(n_1+n_3+n_4)$
$cc'a,cc'b,cc'd$	$\Pi_1(t)\cdot\Pi_2(t')$	$n_3(n_3-1)(n_1+n_2+n_4)$
$dd'a,dd'b,dd'c$	$\Pi_1(t)\cdot\Pi_2(t')$	$n_4(n_4-1)(n_1+n_2+n_3)$
$aa'bc,aa'cd,aa'bd$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_1(n_1-1)(n_2n_3+n_2n_4+n_3n_4)$
$bb'ac,bb'cd,bb'ad$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_2(n_2-1)(n_1n_3+n_1n_4+n_3n_4)$
$cc'ab,cc'ad,cc'bd$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_3(n_3-1)(n_1n_2+n_1n_4+n_2n_4)$
$dd'ab,dd'ac,dd'bc$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_4(n_4-1)(n_1n_2+n_1n_3+n_2n_3)$
$aa'bb',aa'cc',aa'dd'$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_1(n_1-1)\,[n_2(n_2-1)+n_3(n_3-1)+n_4(n_4-1)]$
$bb'cc',bb'dd'$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_2(n_2-1)\,[n_3(n_3-1)+n_4(n_4-1)]$
$cc'dd'$	$\Pi_2(t)\cdot\Pi_2(t')$	$n_3(n_3-1)n_4(n_4-1)$