A practical state recovery attack on the stream cipher Sablier v1[⋆]

Xiutao FENG and Fan ZHANG

Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, CAS, China (e-mail: fengxt@amss.ac.cn)

Abstract. Sablier is an authenticated encryption cipher submitted to the CAESAR competition, which is composed of the encryption Sablier v1 and the authentication Au. In this work we present a state recovery attack against the encryption Sablier v1 with time complexity about 2^{44} operations and data complexity about 24 of 16-bit keywords. Our attack is practical in the workstation. It is noticed that the update of the internal state of Sablier v1 is invertible, thus our attack can further deduce a key recovery attack and a forgery attack against the authenticated encryption Sablier. The result shows that Sablier v1 is far from the goal of its security design (80-bit level).

Keywords: CAESAR, stream ciphers, Sablier, state recovery attack.

1 Introduction

Authenticated cipher is a cipher combining encryption algorithm with authentication algorithm, which can provides confidentiality, integrity and authenticity assurances on the data simultaneously and has been widely used in session communications such as SSL/TLS [1, 2], etc. Since the security of authenticated ciphers depends on both encryption ciphers and authentication ciphers, and an adversary possesses extra information such as authentication tag except for plaintext/ciphertext and has more choices in the execution of attacks (he may attack any one of the encryption cipher and the authentication cipher, or attack both of them simultaneously), thus it is more difficult to design a good authenticated cipher than a usual encryption cipher. CAESAR is a new competition calling for submissions of authenticated ciphers [3]. This competition follows a long tradition of focused competitions in secret-key cryptography. It is expected to have a tremendous increase in confidence in the security of authentication ciphers.

Sablier is an authenticated cipher designed by B. Zhang et al and has been submitted to the CAESAR competition [4]. The encryption of Sablier is a stream cipher named Sablier v1 with a new internal structure to generate the keystream

⋆ This work was supported by the Natural Science Foundation of China (Grant No. 61121062, 11071285), the 973 Program (Grant No. 2011CB302401)

from a 80-bit key and a 80-bit IV. As the designers said, since Sablier v1 is designed based on 16-bit words and adopts only some bitwise XOR, bitwise logical AND and bitwise intra-word rotation, its hardware implementation can be 16 times faster than that of Trivium [5]. The authentication mechanism of Sablier is similar to Grain-128a [6]. In this work we mainly focus on the encryption of Sablier, i.e., Sablier v1. As for Sablier v1, we present a state recovery attack with time complexity about 2^{44} operations and about 24 of 16-bit key words, which is practical in the workstation. What is more, since the update of the internal state of Sablier v1 is invertible, our attack can further deduce a key recovery attack and a forgery attack for Sablier. So Sablier is insecure.

The rest of this paper is organized as follows: in section 2 we recall Sablier v1 briefly, and in section 3 some observations on Sablier v1 are provided. Based on these observations, in section 4 we present a state recovery attack on Sablier v1 and further provide the complexity analysis of our attack. Finally section 5 concludes the paper.

2 Description of Sablier v1

In this section we will recall Sablier v1 briefly, and more details on Sablier v1 can be found in [4].

The structure of the authenticated encryption Sablier is shown in Fig. 1. It is composed of the encryption Sablier v1 and the authentication **Au**. The primary recommended parameter set of Sablier is 10-byte key, 10-byte nonce and 4-byte tag. The input of Sablier includes a plaintext P, associated data A and a public message number N, and the output of Sablier is (C, T), where C is an unauthenticated ciphertext and T is an authenticated tag.

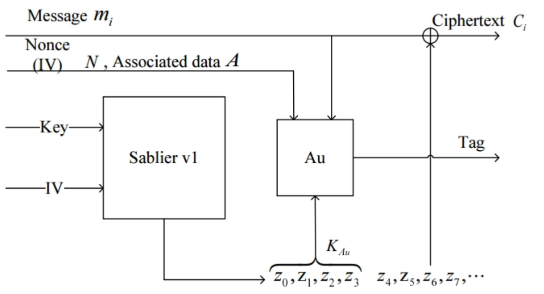

Fig. 1 The structure of the authenticated encryption Sablier

Sablier v1 is a 16-bit word-based stream cipher, which is shown in Fig. 2. The design of Sablier v1 is inspired by sandglass, and the operation of Sablier can be imaged as the mixing of the sand in a sandglass, or as the shaking of the cocktail in a shaker in the bar. Sablier v1 contains five registers L_i $(i = 1, 2, ..., 5)$, which are used as below:

- L_1 and L_5: the two largest registers, each with four 16-bit words, namely $L_{1,i}$,$L_{5,i}$ with $1 \leq i \leq 4$;

– L_2 and L_4: the two second largest registers, each with two 16-bit words, namely $L_{2,i}$,$L_{4,i}$ with $1 \leq i \leq 2$;
– L_3: the smallest register, consists of one 16-bit word.

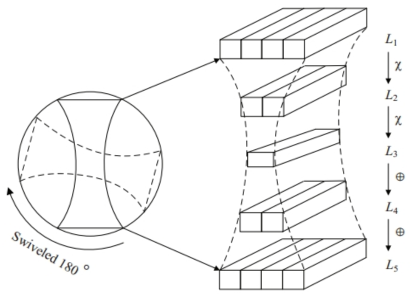

Fig. 2 The structure of the stream cipher Sablier v1

The keystream generation of Sablier v1 mainly consists of two half round operations, namely the lower half round and the upper half round, as shown below:

```
The keystream generation
$t = 0$;
repeat until enough keystream bits are generated.
{
    The lower half round :
    $1 : L_5 \leftarrow (L_{5,1} \oplus L_{5,2} \oplus L_{5,3} \oplus L_{4,2},\ L_{5,1} \oplus L_{5,2},\ L_{5,3} \oplus L_{5,4},$
                                  $L_{5,2} \oplus L_{5,3} \oplus L_{5,4} \oplus L_{4,1})$
    $2 : L_4 \leftarrow (L_{4,1} \oplus L_3 || L_{4,2} \oplus L_3) \ggg 5$
    $3 : L_3 \leftarrow L_3 \oplus ((L_{2,1} \oplus \mathbf{1}) \cdot L_{2,2}) \oplus C_1$
    $4 : L_2 \leftarrow (L_{2,1} \oplus ((L_{1,1} \oplus \mathbf{1}) \cdot L_{1,2}),\ L_{2,2} \oplus ((L_{1,3} \oplus \mathbf{1}) \cdot L_{1,4}))$
    $5 : (L_1, L_2, L_3, L_4, L_5) \leftarrow (L_5, L_4, L_3, L_2, L_1)$
    The upper half round :
    $6 : L_5 \leftarrow (L_{5,1} \oplus L_{5,2} \oplus L_{5,3} \oplus L_{4,2},\ L_{5,1} \oplus L_{5,2},\ L_{5,3} \oplus L_{5,4},$
                                  $L_{5,2} \oplus L_{5,3} \oplus L_{5,4} \oplus L_{4,1}))$
    $7 : L_4 \leftarrow (L_{4,1} \oplus L_3 || L_{4,2} \oplus L_3) \ggg 5$
    $8 : L_3 \leftarrow L_3 \oplus ((L_{2,1} \oplus \mathbf{1}) \cdot L_{2,2}) \oplus C_2$
    $9 : L_2 \leftarrow (L_{2,1} \oplus ((L_{1,1} \oplus \mathbf{1}) \cdot L_{1,2}),\ L_{2,2} \oplus ((L_{1,3} \oplus \mathbf{1}) \cdot L_{1,4}))$
    $10 : (L_1, L_2, L_3, L_4, L_5) \leftarrow (L_5, L_4, L_3, L_2, L_1)$
    **Output the keystream**
      11 : $z_t = L_{2,2} \oplus L_3 \oplus L_{5,3}$
      12 : $t = t + 1$
}
end-repeat
```

where $C_1 = 0x1735$ and $C_2 = 0x9cb6$.

Since our attack does not involve in the initialization of Sablier v1 and the authentication Au, thus we omit them here, and if the readers are interested in them, please refer to [4].

3 Some properties of Sablier v1

In this section we will reveal some properties of the keystream generation of Sablier v1. First we introduce some notations.

Sablier v1 contains 13 of 16-bit word registers. We denote by L these registers, that is,

$$L = (L_{1,1}, L_{1,2}, L_{1,3}, L_{1,4}, L_{2,1}, L_{2,2}, L_3, L_{4,1}, L_{4,2}, L_{5,1}, L_{5,2}, L_{5,3}, L_{5,4}).$$

For $0 \leq i \leq 15$, define

$$\begin{aligned} L[i] = (&L_{1,1}[i], L_{1,2}[i], L_{1,3}[i], L_{1,4}[i], L_{2,1}[i], L_{2,2}[i], L_3[i], L_{4,1}[i], L_{4,2}[i], \\ &L_{5,1}[i], L_{5,2}[i], L_{5,3}[i], L_{5,4}[i]), \end{aligned}$$

and call $L[i]$ the i-th facet of the registers L, where $x[i]$ means the i-th bit register of x for a 16-bit word register x, $0 \leq i \leq 15$. At time $t \geq 0$, we denoted by L^t and $L^t[i]$ the state of the registers L and the facet $L[i]$ respectively, where $0 \leq i \leq 15$.

Set the sequence

$$i_0 i_1 \cdots i_{16} = (0, 11, 6, 1, 12, 7, 2, 13, 8, 3, 14, 9, 4, 15, 10, 5, 0).$$

Our attack is mainly due to the following three observations:

Observation 1 *For any $1 \leq j \leq 16$ and time $t \geq 0$, the update of the state $L^t[i_j]$ of the facet $L[i_j]$ only depends on $L^t[i_j]$ and the 4-bit information coming from the state $L^t[i_{j-1}]$ of the facet $L[i_{j-1}]$.*

The above observation follows directly from the keystream generation of Sablier v1. Indeed the operations of all steps except steps 2 and 7 in the procedure of the keystream generation are done in the current facet since both the exclusive or "$\oplus$" and the dot multiplication "$\cdot$" are bitwise. At steps 2 and 7 only the rotation "<<<" needs the data from other facets. If both the 2-bit values of L_4 after the update at step 2 and the 2-bit values of L_4 after the update at step 7 are known, then the update of the state of the facet $L[i_j]$ is done well.

We consider the state $L^t[i_j]$ of the facet $L[i_j]$ at time t for some integer $1 \leq j \leq 16$, and view them as some unknown variables. Suppose that all extra bit information from the facet $L[i_{j-1}]$ are known, below we consider how to establish equations on the state variables $L^t[i_j]$ by the output keystream $\{z_t\}_{t\geq 0}$.

By step 11 in the keystream generation we have

$$z_{t+i}[i_j] = L_{2,2}^{t+i}[i_j] \oplus L_3^{t+i}[i_j] \oplus L_{5,3}^{t+i}[i_j], \quad i \geq 0. \tag{1}$$

For any $i \geq 1$, first we get by steps 8, 5 and 3

$$\begin{aligned}
L_3^{t+i}[i_j] &= L_3^{t+(i-1)+0.5}[i_j] \oplus (L_{2,1}^{t+(i-1)+0.5}[i_j] \oplus 1) \cdot L_{2,2}^{t+(i-1)+0.5}[i_j] \oplus C_2 \\
&= L_3^{t+(i-1)}[i_j] \oplus (L_{2,1}^{t+(i-1)}[i_j] \oplus 1) \cdot L_{2,2}^{t+(i-1)}[i_j] \oplus C_1 \\
&\quad \oplus (L_{2,1}^{t+(i-1)+0.5}[i_j] \oplus 1) \cdot L_{2,2}^{t+(i-1)+0.5}[i_j] \oplus C_2,
\end{aligned}$$

where $L_{2,1}^{t+(i-1)+0.5}[i_j]$ and $L_{2,2}^{t+(i-1)+0.5}[i_j]$ means the state of $L_{2,1}[i_j]$ and $L_{2,2}[i_j]$ respectively after the lower half round in the $(i-1)$-th round. Note that all $L_{2,2}^{t+i}[i_j]$, $L_{2,1}^{t+(i-1)+0.5}[i_j]$ and $L_{2,2}^{t+(i-1)+0.5}[i_j]$ come from the facet $L[i_{j-1}]$ and are known, thus at last $L_3^{t+i}[i_j]$ only depends on both $L_3^t[i_j]$ and $(L_{2,1}^t[i_j] \oplus 1) \cdot L_{2,2}^t[i_j]$.

Second, by steps 10, 5 and 1 we get

$$L_{5,3}^{t+i}[i_j] = L_{5,3}^{t+(i-1)}[i_j] \oplus L_{5,4}^{t+(i-1)}[i_j].$$

When $i \geq 2$, further we have

$$L_{5,3}^{t+i}[i_j] = L_{5,2}^{t+(i-2)}[i_j] \oplus L_{4,1}^{t+(i-2)}[i_j].$$

By steps 10, 9 and 5 it is easy to check that the update of $L_4^{t+i}[i_j]$ only depends on $L_4^{t+(i-1)}[i_j]$ and $L_5^{t+(i-1)}[i_j]$. Thus we have

Observation 2 *For any given $1 \leq j \leq 16$, we always assume that all extra bit information from the facet $L[i_{j-1}]$ are known during the update of the state of the facet $L[i_j]$. If we view the state $L^t[i_j]$ of the facet $L[i_j]$ at time t as the unknown variables, then we get an equation system on at most six variables $L_{5,1}^t[i_j]$, $L_{5,2}^t[i_j]$, $L_{5,3}^t[i_j]$, $L_{5,4}^t[i_j]$, $L_{4,1}^t[i_j]$, $L_{4,2}^t[i_j]$ and two intermediate variables $L_3^t[i_j] \oplus L_{2,2}^t[i_j]$ and $L_{2,1}^t[i_j] \cdot L_{2,2}^t[i_j]$.*

By Observation 2 it is known that the 4-bit information of $L_1^t[i_j]$ will never be got whatever we retrieve equations (1). Thus during the execution of the attack we always need to guess these bit variables.

For $i = 0, 1, \cdots, 7$, the exact equations on the state $L^t[i_j]$ of the facet $L[i_j]$ has been established in Appendix A. By these equations it is easy to see that the following conclusion holds:

Observation 3 *For any two distinct facets $L[i_{j_1}]$ and $L[i_{j_2}]$, the equation systems constructed by equations (1) at the facets $L[i_{j_1}]$ and $L[i_{j_2}]$ have the same form, which only depends on the values of seven parameters $L_{2,1}^{t+i+0.5}$, $L_{2,2}^{t+i+0.5}$ and $L_{2,2}^{t+3.5}$, where $i = 0, 1, 2$.*

4 A state recovery attack

In this section we will present a state recovery attack on Sablier v1 based on the above three observations and provide the complexity analysis of our attack.

4.1 The pre-computation

By Observation 2 and 3 we can set up a table to compute six variables $L^t_{5,1}[i_j]$, $L^t_{5,2}[i_j]$, $L^t_{5,3}[i_j]$, $L^t_{5,4}[i_j]$, $L^t_{4,1}[i_j]$, $L^t_{4,2}[i_j]$ and two intermediate variables $L^t_3[i_j] \oplus L^t_{2,2}[i_j]$ and $L^t_{2,1}[i_j] \cdot L^t_{2,2}[i_j]$ in the facet $L[i_j]$ at time t from equations (1) got by z_{t+i}, where $0 \le i \le 7$. Since these equations involve some bit information from other facet, we need to set up a table for each possible values of those involved bit information. Fortunately these involved bit information can be finally synthesized into seven independent intermediate variables $L^{t+i+0.5}_{2,1}$, $L^{t+i+0.5}_{2,2}$ and $L^{t+3.5}_{2,2}$, where $i = 0, 1, 2$. So in practice we only need to set up 2^7 tables, and each table contains 2^8 items, each item one byte. The time complexity of the pre-computation is about 2^{15} and the size of the memories used to storing the tables is 2^{15}B = 32KB.

4.2 Online attack

In the online attack we need to retrieve about 24 of 16-bit key words z_{t+i}, where $0 \le i \le 23$. Our attack may start on an arbitrary facet $L[i_j]$. Without loss of generality, we start at $j = 0$, and the details are shown as below:

1. Set $j = 0$;
2. For each time $t + i$ $(0 \le i \le 7)$, we first guess the 4-bit information from other facet and look up the table to get the 8-bit values of $L^t_5[i_j]$, $L^t_4[i_j]$, $L^t_3[i_j] \oplus L^t_{2,2}[i_j]$ and $L^t_{2,1}[i_j] \cdot L^t_{2,2}[i_j]$; We further guess the 5-bit values of $L^t_1[i_j]$ and $L^t_3[i_j]$, and recover 8 states $L^{t+i}[i_j]$ $(i = 0, 1, 2, \cdots, 7)$ of the facet $L[i_j]$.
3. Consider the next facet $L[i_{j+1}]$. According to each possible states $L^{t+i}[i_j]$ $(0 \le i \le 7 + j)$ of the previous facet $L[i_j]$, we can get $9 + j$ equations on the state $L^t[i_{j+1}]$. If these equations have no solution, then we try the next possible states $L^{t+i}[i_j]$ $(0 \le i \le 7 + j)$; otherwise, similar to step 2, we further guess the rest 5-bit values and finally get $9 + j$ states of the $L[i_{j+1}]$.
4. Set $j = j + 1$. If $j = 16$, output the state L^t; otherwise, go to step 3.

4.3 The complexity of our attack

In step 2 we need to guess total $4 \times 7 + 5 = 33$ bits and get about 2^{33} possible states $L^{t+i}[i_0]$ $(0 \le i \le 7)$ of the facet $L[i_0]$ on average. Suppose that there are N_j possible states of the facet $L[i_j]$. Since we get $9 + j$ equation on the state $L^t[i_{j+1}]$ of the facet $L[i_{j+1}]$ in step 3, thus about $N_j \times 2^{8-(9+j)}$ possible states can be remained on average. So $N_{j+1} \approx N_j \times 2^{8-(9+j)} \times 2^5 = N_j \times 2^{4-j}$, and we have

$$N_j \approx 2^{33} \times \prod_{i=0}^{j-1} 2^{4-i} = 2^{33+4j-\frac{1}{2}(j-1)j}.$$

For each possible solutions on the facet $L[i_j]$ the time of the computation of the state $L^t[i_j]$ by looking up the pre-computation table is very low and we ignore these time consuming. Thus we can get an evaluation of the total time complexity of our attack, that is,

$$T \approx \sum_{j=0}^{15} N_j \approx 2^{44}.$$

5 Conclusion

In this paper we study the encryption algorithm Sablier v1 of the authenticated encryption Sablier, and give a state key recovery attack on Sablier v1, whose time complexity is about 2^{44} operations and is practical in the workstation. Since the update of the state of Sablier v1 is invertible, thus our attack can further deduce a key recovery attack and a forgery attack on the authenticated encryption Sablier.

References

1. RFC 6101: The Secure Sockets Layer (SSL) Protocol Version 3.0.
2. RFC 5246: The Transport Layer Security (TLS) Protocol, Version 1.2.
3. CAESER: http://competitions.cr.yp.to/index.html.
4. Sablier v1: B. Zhang, Z.Q. Shi, C. Xu, Y. Yao, Z.Q. Li, submission to CAESAR, available from: http://competitions.cr.yp.to/round1/sablierv1.pdf.
5. Christophe De Canniere and Bart Preneel, Trivium specifications, eSTREAM Projet, http://www.ecrypt.eu.org/stream/e2-trivium.html.
6. Martin Agren, Martin Hell, Thomas Johansson and Willi Meier, Grain-128a: A New Version of Grain-128 with Optional Authentication, http://lup.lub.lu.se/record/2296437/file/2296485.pdf.

A Equations on the state $L^t[i_j]$ at time $t+i$ $(i = 0, 1, \cdots, 7)$

When all bit information from the facet $L[i_{j-1}]$ are known, we can establish equations on the state $L^t[i_j]$ of the facet $L[i_j]$, which are shown as below:

$$\begin{aligned}
z_t[i_j] &= L^t_{2,2}[i_j] \oplus L^t_3[i_j] \oplus L^t_{5,3}[i_j],\\
z_{t+1}[i_j] \oplus G_{t+1} &= x[i_j] \oplus L^t_{5,3}[i_j] \oplus L^t_{5,4}[i_j],\\
z_{t+2}[i_j] \oplus G_{t+2} &= x[i_j] \oplus L^t_{5,2}[i_j] \oplus L^t_{4,1}[i_j],\\
z_{t+3}[i_j] \oplus G_{t+3} &= x[i_j] \oplus (L^t_{5,1}[i_j] \oplus L^t_{5,2}[i_j])(L^t_{5,3}[i_j] \oplus L^t_{4,2}[i_j] \oplus 1),\\
z_{t+4}[i_j] \oplus G_{t+4} &= x[i_j] \oplus (L^t_{5,3}[i_j] \oplus L^t_{4,2}[i_j])((L^t_{5,3}[i_j] \oplus L^t_{5,4}[i_j] \oplus 1)(L^t_{5,2}[i_j] \oplus L^t_{4,1}[i_j] \oplus 1) \oplus L^{t+0.5}_{2,2}[i_j]),\\
z_{t+5}[i_j] \oplus G_{t+5} &= x[i_j] \oplus (L^{t+1}_{5,3}[i_j] \oplus L^{t+1}_{4,2}[i_j])((L^{t+1}_{5,3}[i_j] \oplus L^{t+1}_{5,4}[i_j] \oplus 1)(L^{t+1}_{5,2}[i_j] \oplus L^{t+1}_{4,1}[i_j] \oplus 1) \oplus L^{t+1.5}_{2,2}[i_j]),\\
z_{t+6}[i_j] \oplus G_{t+6} &= x[i_j] \oplus (L^{t+2}_{5,3}[i_j] \oplus L^{t+2}_{4,2}[i_j])((L^t_{5,3}[i_j] \oplus L^{t+2}_{5,4}[i_j] \oplus 1)(L^{t+2}_{5,2}[i_j] \oplus L^{t+2}_{4,1}[i_j] \oplus 1) \oplus L^{t+2.5}_{2,2}[i_j]),\\
z_{t+7}[i_j] \oplus G_{t+7} &= x[i_j] \oplus (L^{t+3}_{5,3}[i_j] \oplus L^{t+3}_{4,2}[i_j])((L^{t+3}_{5,3}[i_j] \oplus L^{t+3}_{5,4}[i_j] \oplus 1)(L^{t+3}_{5,2}[i_j] \oplus L^{t+3}_{4,1}[i_j] \oplus 1) \oplus L^{t+3.5}_{2,2}[i_j]),
\end{aligned}$$

where G_{t+i} $(1 \le i \le 7)$ only depends on the data from the facet $L[i_{j-1}]$ and

$$\begin{aligned}
x[i_j] &= L^t_3[i_j] \oplus L^t_{2,2}[i_j] \oplus L^t_{2,1}[i_j]L^t_{2,2}[i_j],\\
L^{t+i+1}_{5,1}[i_j] &= L^{t+i}_{5,1}[i_j] \oplus L^{t+i}_{5,2}[i_j] \oplus L^{t+i}_{5,3}[i_j] \oplus L^{t+i}_{4,2}[i_j],\\
L^{t+i+1}_{5,2}[i_j] &= L^{t+i}_{5,1}[i_j] \oplus L^{t+i}_{5,2}[i_j],\\
L^{t+i+1}_{5,3}[i_j] &= L^{t+i}_{5,3}[i_j] \oplus L^{t+i}_{5,4}[i_j],\\
L^{t+i+1}_{5,4}[i_j] &= L^{t+i}_{5,2}[i_j] \oplus L^{t+i}_{5,3}[i_j] \oplus L^{t+i}_{5,4}[i_j] \oplus L^{t+i}_{4,1}[k],\\
L^{t+i+1}_{4,1}[i_j] &= L^{t+i+0.5}_{2,1}[i_j] \oplus L^{t+i+1}_{5,1}[i_j] \cdot L^{t+i+1}_{5,2}[i_j] \oplus L^{t+i+1}_{5,2}[i_j],\\
L^{t+i+1}_{4,2}[i_j] &= L^{t+i+0.5}_{2,2}[i_j] \oplus L^{t+i+1}_{5,3}[i_j] \cdot L^{t+i+1}_{5,4}[i_j] \oplus L^{t+i+1}_{5,4}[i_j]
\end{aligned}$$

for $i = 0, 1, 2$.

It is easy to see that the above equation system only depends on the values of G_{t+i} $(i = 0, 1, \cdots, 7)$, $L^{t+i+0.5}_{2,1}[i_j]$, $L^{t+i+0.5}_{2,2}[i_j]$ $(i = 0, 1, 2)$ and $L^{t+3.5}_{2,2}[i_j]$. When these values are determined, on average we get one solution

$$(L^t_{5,1}[i_j], L^t_{5,2}[i_j], L^t_{5,3}[i_j], L^t_{5,4}[i_j], L^t_{4,1}[i_j], L^t_{4,2}[i_j], L^t_3[i_j] + L^t_{2,2}[i_j], L^t_{2,1}[i_j]L^t_{2,2}[i_j]).$$