Time and Space Complexity Reduction of a Cryptanalysis Algorithm

Mohammad Ghasemzadeh
Electrical and Computer Engineering Department,
Yazd University, Yazd, Iran
m.ghasemzadeh@yazduni.ac.ir

Received: 2011/04/16; Accepted: 2011/05/24 Pages: 39-46

Abstract

Binary Decision Diagram (in short BDD) is an efficient data structure which has been used widely in computer science and engineering. BDD-based attack in key stream cryptanalysis is one of the best forms of attack in its category. In this paper, we propose a new key stream attack which is based on ZDD(Zero-suppressed BDD). We show how a ZDD-based key stream attack is more efficient in time and space complexity over its BDD-based variant against the E_0 type of the Bluetooth security mechanism. We implemented it by using the CUDD - Colorado University Decision Diagram package. Experimental results show great improvements. We have also derived a mathematical proof, which shows that it is better than the BDD-based attack method even for the worst case analysis.

Keywords: *Binary Decision Diagram, Cryptanalysis, Algorithm complexity*

1. Introduction

In cryptography, pseudo random sequences are frequently used. A pseudo random sequence generator requires to be uniformly distributed, independent, and non-correlated [8]. In implementation of key stream generators, the LFSR (Linear Feedback Shift Register) is being used because all above conditions are met and the corresponding algebraic analysis is quite simple.

The LFSR-based key stream generators consist of two components: a linear bit stream generator *L* and a nonlinear compression function *C, i.e. K=(L,C)*. First they generate the key stream *Y=C(L(k))*, for the cipher key *k*, then *Y* and the plain text *P* are bitwise XORed to produce the cipher text *E*. In cryptanalysis of these generators, the encryption system is supposed to be known and we are interested in finding *k*.

BDD and its variants are data structures that are used effectively in computer science and engineering. These data structures give compact and canonical representations for Boolean functions. Recently, a new attack against LFSR-based key stream generators is introduced by Krause [3] which is based on a variant of BDD known as FBDD. Later Shacked and Wool [9] introduced their OBDD-based attack to E_0 key stream generator. In this paper, we introduce a new attack to key stream generators which uses ZDD. Experimental results show that it makes a remarkable reduction in time and space complexity regarding OBDD and FBDD based attacks. We have also derived a proof which confirms the experimental results.

This paper is organized as follows. Section 2 provides the basic definitions and the main concepts: E_0 encryption system and a brief introduction to BDD and ZDD. In section 3 the proposed attack is introduced. First the FBDD attack is discussed, then the attack to *E₀* with OBDD is reviewed. Finally our ZDD-based attack is introduced. Section 4 is dedicated to the theoretical complexity analysis of our method. Section 5 provides concludes.

2. Preliminaries

2.1 E_0 Key Stream Generator

E_0 is an LFSR-based key stream generator which is used in Bluetooth security mechanism. LFSR-based key stream generators consist of two components, a linear bit stream generator and a nonlinear compression function. After initialization, the linear bit stream generator L, generates the bit stream Z. It employs four Linear Feedback Shift Registers(LFSR), whose output is the input to the compression function C. The output of the compression function would be the key stream $Y = C(L(k))$. The lengths of the four LFSR are $|L_0| = 25$, $|L_1| = 31$, $|L_2| = 33$ and $|L_3| = 39$, and their feedback polynomials are:

$$p_0(x) = x^{25} + x^{20} + x^{12} + x^8 + 1$$

$$p_1(x) = x^{31} + x^{24} + x^{16} + x^{12} + 1$$

$$p_2(x) = x^{33} + x^{28} + x^{24} + x^4 + 1$$

$$p_3(x) = x^{39} + x^{36} + x^{28} + x^4 + 1$$

At the beginning, the linear generator needs to be loaded with an initial value for the four LFSRs(128 bits in total). Summation of the four output bits of the LFSRs make the input of the compression function. The compression function is usually organized with a finite state machine $C_{E_0} : (Q, \Sigma, \Gamma, I, F, d)$, States of the FSM are $Q = \{q_i : 0 \le i \le 15\}$,its input alphabet $\Sigma = \{0,1,2,3,4\}$, output alphabet $\Gamma = \{0,1\}$ and I, F stand for the set of initial and final states. The set of FSM transition rules $d \subseteq Q \times \Sigma \times \Gamma \times Q$ have elements in the form of $(q_n, a) \rightarrow q_{n+1}$ [2, 9, 4].

2.2 BDD versus ZDD

There are several known methods for representing Boolean formulas. The most important of them are: Truth table, Karnough map and Boolean expressions. BDD or more precisely ROBDD is also a data structure invented for this purpose. This data structure is a graph which can be obtained from the binary decision tree of the Boolean formula by applying merging and removing rules [1, 6]. Altogether this method is better than other methods. The benefits of ROBDD are: 1. Provides a canonical representation, 2.Represents Boolean functions more compactly and 3.Offers faster Boolean operations.

A *set* can be represented by its *characteristic function*. In this regard, according to each element/subset we consider a minterm in the corresponding characteristic function. Theoretical analysis and practical experiments has shown that a variant of BDD called ZDD (Zero suppressed Binary Decision Diagrams) [7] is more suitable for representing such a characteristic function.

A ZDD can also be obtained from binary decision tree of a Boolean formula. In a BDD whenever 1-edge and 0-edge of a node point to the same node that node must be removed, but in a ZDD whenever the 1-edge of a node points to 0-terminal, that node must be removed. The merging rule is the same for both of them. In a ZDD each path from the root to the 1-terminal stands for an element of the set [3, 5].

3. ZDD Based Cryptanalysis Of *E₀*

In this section we first introduce the FBDD based attacker of Krause [3], then we reveal the motivation led us using ZDD instead of FBDD or OBDD. Finally we introduce and discuss our ZDD-based attacker.

3.1 FBDD Based Cryptanalysis Of Key Stream Generator

Krause in his work [3] assumes that except for key k, all other parameters are known, also he assumes that the attacker is able to obtain the first bits of the key stream Y. The goal of the attacker is computing $k \in \{\{0,1\}^n\}$. Since in an LFSR, the first output bits are the same as its initial values, $Z = L(k)$ would contain k in the first bits. Therefore the problem reduces to finding a bit stream Z satisfying the following conditions:

1. Z can be produced by the linear bit stream generator L.
2. $C(Z)$ is prefix of the observed key stream Y.

For $m \geq 1$, and the bit stream $z \in \{0,1\}^m$ the following items are defined:

- G_m^C is an oracle graph representing the order in which the bits of Z are being read by the compression function C.
- R_m is a minimal $G_m^C - FBDD$ graph which decides whether Z can be produced by L or not.
- Q_m is a minimal $G_m^C - FBDD$ graph which decides whether $C(Z)$ is a prefix of Y or not.
- P_m is a minimal $G_m^C - FBDD$ graph which decides whether Z can be produced by L where $C(Z)$ is a prefix of Y or not.

In this method, the key is considered to be n bits and it computes m^*, where m^* denotes the length of the consecutive bits required for finding the key k. Considering above formulations, the following algorithm can compute k:

```
1. $P \leftarrow Q_n$.
2. for $m \leftarrow n+1$ to $m^*$ do:
        $P \leftarrow (P \wedge Q_m \wedge R_m)$
3. return $Z^*$ where $P(Z^*) = 1$.
```

On the other words, the above loop iterates until P_{m^*} has only one assignment $z^* \in \{0,1\}^m$ where $P(Z^*) = 1$.

3.2 Reduction of FBDD-based Cryptanalysis using OBDDs

The algorithm described by Krause is generic and needs to be adapted. Shacked and Wool [9] made reductions and adopted it for E_0, by using OBDD instead of FBDD.

Krause in [4] generalized OBDD attack to oblivious key stream generator. In the OBDD attack the output bits of $L(k)$ are considered as: $Z = (..., z_{4j}, z_{4j+1}, z_{4j+2}, z_{4j+3}, ...)$, where $z_{4j+i} \in L_i(k_i)$. This ordering leads to the following equations for the linear key stream generator L:

$$\forall i = 4j : z_i = z_{i-32} \oplus z_{i-48} \oplus z_{i-80} \oplus z_{i-100} \quad (1)$$

$$\forall i = 4j+1 : z_i = z_{i-48} \oplus z_{i-64} \oplus z_{i-96} \oplus z_{i-124}$$

$$\forall i = 4j+2 : z_i = z_{i-16} \oplus z_{i-96} \oplus z_{i-112} \oplus z_{i-132}$$

$$\forall i = 4j+3 : z_i = z_{i-16} \oplus z_{i-112} \oplus z_{i-144} \oplus z_{i-156}$$

Afterwards, according to the obtained equations, R_m graph is produced by building OBDDs for each z_i.

In building OBDDs which check bits for each L_i, the algorithm calls the first $|L_i|$ bits in its bit stream. The goal of the algorithm is to compute these leading bits of L_i. According to above equations, an algorithm must build OBDDs for $z_j : |L| \leq j \leq 4|L|$. A BDD structure called basic chain is used to compute Q_m graph which represents sums of 4 bits. For each state and each of the 5 possible sums, if the output bit matches the bit given in the key stream Y, it can proceed to next chain; otherwise this path would lead to a $0-Terminal$.

3.3 ZDD-Based Cryptanalysis Of E₀

Combinations of n items can be represented by an n-bit vector, $(x_1, ...x_n)$, where $x_i \in \{0,1\}$ determines whether x_i is included in the combination or not. In this way, a set of combinations can be represented with a Boolean function. Such a Boolean function is called *characteristic function* of the set. In general, OBDDs are more efficient in compact representation of characteristic functions than other methods, but Minato[7] has shown that if we change the elimination rule, we can represent characteristics functions much more efficiently.

The goal of key stream Cryptanalysis is to analyse all possible keys and find the right one. FBDD attack can be reduced by using OBDD, because these generators have the same ordering, in building R_m and Q_m graphs as well as in building P_m. The compression function of these generators can be shown with a finite state machine. We may use ZDD to construct a more efficient attack on this kind of key stream generators (to attack E_0 key stream generator).

In our ZDD attack against E_0 generator, we implemented the R_m graph in a similar way as in OBDD attack, the only difference is using ZDD instead of OBDD. Each synthetic ZDD contains of 5 variables and 9 vertices, therefore, it requires 3456 vertices. We computed the Q_m graph by the following method; Since finite state machine of E_0 generator has 16 states, we used 4 variables $0 \leq i \leq 3, q_i^n$ to mark the states. Thus the following function can be computed:

$$Q_m = F(q_3^{m+1}, q_2^{m+1}, q_1^{m+1}, q_0^{m+1},\ z_{4m+3}, z_{4m+2},, z_{4m+1},.., z_0)$$

Clearly, Q_m consists of $4m+4$ variables. It stands for all the possible paths in the finite state machine after reading $m+1$ input symbols. We implemented Q_m using the following algorithm:

1. If C_{E_0} includes transition rule $(q_m, a) \to q_{m+1}$ AND correspondent output rule $(q_m, a) \to b$ AND $b = b_m (b_m$ is m th bit in known key stream Y):

(a) Compute q_m and q_{m+1} based on q_m^i:

$$q_m = (q_m^3)^* \wedge (q_m^2)^* \wedge (q_m^1)^* \wedge (q_m^0)^*$$

where $(q_m^i)^*$ is q_m^i or $(q_m^i)^*$ is $(q_m^i)'$ according to labels of the states of the machine. For example in step m, the 5th state is : $(q_m^3)' \wedge (q_m^2) \wedge (q_m^1)' \wedge (q_m^0)$.

(b) For all $\sum z_{4m+i} = a$, compute:

$$X_j = (q_{m+1} \wedge z_{4m+3} \wedge z_{4m+2} \wedge z_{4m+1} \wedge z_{4m} \wedge q_m)$$

2. Compute Q_m' function based on:

$$Q_0' = (X_0 \vee ... \vee X_j)$$

$$Q_m' = ((X_0 \wedge Q_{m-1}) \vee ... \vee (X_j \wedge Q_{m-1})$$

3. Compute Q_m by removing $(q_m^i)^*$ from Q_m'.

We need to mention that finally we are interested in computing Q_{128} . The constructed Q_m correctly decides whether $C(Z)$ is prefix of Y or not. By scanning all the paths from root to $1-T$, we compute all Z s which produce the same prefix as Y .

A pseudo random sequence must be distributed uniformly, i.e., the probability of 0 occurrence must be equal to the probability of 1 occurrence. This property along with other required properties, enforce the constructed Q_m to be a sparse graph. In implementing our proposed attack, we mapped the problem to a combinatorial set problem. In fact, in each iteration of computing Q_m, we checked all possible combinations of input bits and final states.

Most operation on sets such as union, intersect, difference are already defined and implemented for ZDD. In addition some other useful functions like:

- Z.onset(N) selects the subset of the combinations including N, and then deletes N from each combination.
- Z.offset(N) returns the subset of the combinations excluding N.
- Z.Count(N) returns number of combinations in the ZDD Z .

are available in most BDD packages.

We ran our algorithm in C along with the CUDD package[10]; Our algorithm can be displayed with the following pseudo code:

```
For $\forall$ element $\in d$
  {
  If $((q_m, a) \to q_{m+1} \wedge (q_m, a) \to b \wedge (b = b_m))$
    {
      $q_m = ZDDIntersect(q_m^3)^*, (q_m^2)^*, (q_m^1)^*, (q_m^0)^*)$
      $q_{m+1} = ZDDIntersect((q_{m+1}^3)^*, (q_{m+1}^2)^*, (q_{m+1}^1)^*, (q_{m+1}^0)^*)$
      For $\forall Z_{4j+i}, 0 \le i \le 3$
       {
        if $\sum Z_{4j+i} = a$
          $X_j = ZDDIntersect(q_{m+1}, z_{4m+3}, z_{4m+2}, z_{4m+1}, z_{4m}, q_m)$
       }
    }
  $Q_m' \leftarrow ZDDUnion(\forall j, ZDDIntersct(X_j,$
  $Q_{m-1}.Oneset(q_m), if((q_m)_{X_j} == (q_m)_{Q_{m-1}})$
 }

For every $q_m$
  $Q_m \leftarrow Q.Oneset(q_m)$
```

4. Theoretical Complexity Analysis

The time complexity of the algorithm is determined by the space complexity of the constructed ZDD during the entire process of construction. First, lets take a look at the complexity of functions which are used in the algorithm:

- The time complexity of producing the ZDD representing $F(x_0,...,x_n)$ is $O(|G_F|)$, where $|G_F|$ denotes the number of vertexes in constructed graph.
- Time complexity of each set operation such as union and intersect of two graph F,G is $O(|G_F|.|G_G|)$

In the algorithm, during the $|L_0|$ steps, it introduces 4 new variables, and one constraint $\sum z_{4j+i} = a$, then the number of assignments is multiplied by 2^3. After $|L_1| - |L_0|$ steps it has two constraints, $z_{4j} \in L_0$ is determined , then the number of assignments is multiplied by 2^2. After $|L_2| - |L_1|$ steps it has three constraints, $z_{4j} \in L_0$ and $z_{4j+1} \in L_1$ are determined, then the number of assignments is multiplied by 2^1. After $|L_3| - |L_2|$ step it has four constraints and there are no more choices, then the number of assignments will be constant. In the next steps, the number of assignments start to decrease to half. On the other hand, due to ZDD properties, the average number of vertices in each path would be

$$\frac{\sum C(4,i).i.m}{2^4} = 2m$$

therefore, based on above arguments, we can compute the higher bound as $(m : 0 \rightarrow 128)$:

$$m \le |L_0| : |P_m| = 2m \times 2^{3m}$$

$$|L_0| \le m \le |L_1| : |P_m| = 2m \times 2^{3|L_0|} \times 2^{2(m-|L_0|)}$$

$$|L_1| \le m \le |L_2| : |P_m| = 2m \times 2^{3|L_0|} \times 2^{2(|L_1|-|L_0|)} \times 2^{m-|L_1|}$$

$$|L_2| \le m \le |L_3| : |P_m| = 2m \times 2^{3|L_0|} \times 2^{2(|L_1|-|L_0|)} \times 2^{2(|L_2|-|L_1|)}$$

$$|L_3| \le m : |P_m| = 2m \times 2^{3|L_0|} \times 2^{2(|L_1|-|L_0|)} \times 2^{|L_2|-|L_1|} \times 2^{|L_3|-m}$$

On the other hand, P_m is obtained by intersection of Q_m and R_m, then we can compute the other higher bound:

$$m \le |L_0| : Time(P_m) = |Q_m| \times |R_{4m}| = |Q_m|$$

$$|L_0| \le m \le |L_1| : Time(P_m) = |Q_m| \times (m - |L_0|) \times 2^3$$

$$|L_1| \le m \le |L_2| : Time(P_m) = |Q_m| \times (m - |L_0| - |L_1|) \times 2^3$$

$$|L_2| \le m \le |L_3| : Time(P_m) = |Q_m| \times (m - |L_0| - |L_1| - |L_2|) \times 2^3$$

$$|L_3| \le m : Time(P_m) = |Q_m| \times (m - |L_0| - |L_1| - |L_2| - |L_3|) \times 2^3 = |Q_m| \times (m - 128) \times 2^3$$

In practice, $|Q_m|$ has approximately 2^{14} nodes. The overall upper bound of complexity can be obtained from intersection of the above two bounds, which will give a space complexity of 2^{23}, and time complexity of 2^{82}. We need to mention that this is a non-refined approximation bound, accurate analysis would give even better values. Here we can see that using ZDD gives a graph with 2^8 nodes less than its predecessor which used OBDD.

5. Conclusion

Zero-suppressed Binary Decision Diagram (in short ZBDD or ZDD) is a variant of BDD. While BDD gives more compact representation and more efficient operations on Bollean formulas, ZDD gives more compact representation and more efficient operations on characteristics functions representingd sets of subsets. This research shows, by utilizing this property, how ZDD can be used to construct an attacker more efficient than the outstanding OBDD-based attacker.

6. References

[1] Randal E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8):677-691, 1986.

[2] Fluhrer, Scott R. and Lucks, Stefan. Analysis of the E0 Encryption System. *8th Annual International Workshop on Selected Areas in Cryptography*, pages 38-48, London, UK, 2001. Springer-Verlag.

[3] Matthias Krause. BDD-Based Cryptanalysis of Keystream Generators. *EUROCRYPT*, pages 222-237, 2002.

[4] Matthias Krause. OBDD-Based Cryptanalysis of Oblivious Keystream Generators. *Theor. Comp. Sys.*, 40(1):101-121, 2007.

[5] Matthias Krause and Dirk Stegemann. Reducing the Space Complexity of BDD-Based Attacks on Keystream Generators. *13th annual Fast Software Encryption Workshop*, pages 163-178, 2006.

[6] Christoph Meinel and Thorsten Theobald. *Algorithms and data structures in VLSI design: OBDD - foundations and applications.* Berlin, Heidelberg, New York: Springer-Verlag, 1998.

[7] Shin-ichi Minato, Zero-suppressed bdds and their applications, *in:Proceedings of International Journal on Software Tools for Technology Transfer*, Springer, 2001, pp. 156-170.

[8] Matt Robshaw. Stream Ciphers. Technical report, RSA Laboratories, 1995.

[9] Yaniv Shaked and Avishai Wool. Cryptanalysis of the Bluetooth E0 cipher using OBDD's. *Proceedings of 9th Information Security Conference, LNCS 4176*, pages 187-202, 2006.

[10] Fabio Somenzi. CUDD: Colorado University Decision Diagram Package. http://vlsi.colorado.edu/~fabio/CUDD/ , 2009.