Time-Data-Memory Trade-Off Based Cryptanalysis of Certain Broadcast Encryption Schemes

Miodrag J. Mihaljević[1], Marc P.C. Fossorier[2] and Hideki Imai[3]

[1] Mathematical Institute, Serbian Academy of Sciences and Arts
Kneza Mihaila 35, 11001 Belgrade, Serbia and Montenegro
Email: miodragm@turing.mi.sanu.ac.yu

[2] Department of Electrical Engineering, University of Hawaii,
540 Dole St., Holmes Hall 483, Honolulu, HI 96822, USA
Email: marc@spectra.eng.hawaii.edu

[3] University of Tokyo, Institute of Industrial Science,
4-6-1, Komaba, Meguro-ku, Tokyo, 153-8505 Japan
Email: imai@iis.u-tokyo.ac.jp

Abstract

This paper points out to a generic vulnerability of certain broadcast encryption schemes. This vulnerability can be effectively explored assuming chosen plaintext attacks, and in some cases even under ciphertext only attack. The developed methods for cryptanalysis are based on an attacking approach not taken into account in the security evaluations of the reported broadcast encryption schemes. The proposed attacks are based on employment of a dedicated time-data-memory trade-off approach for cryptanalysis. Two algorithms for cryptanalysis are proposed and their main characteristics regarding the complexity and required sample are pointed out. The algorithms are applied for cryptanalysis of particular recently reported broadcast encryption schemes implying that their security is far below the claimed ones.

Keywords: broadcast encryption, key management, cryptanalysis, time-memory-data trade-off.

1 Introduction

Broadcast encryption (BE), initially reported in [2] and [4], is a cryptographic method for providing the conditional data access distributed via the public channels. BE schemes employ the following approach for controlling the access privileges: the data are encrypted and only legitimate users are provided with the information on how to decrypt them (for some recent issues and particular applications see [9], [14] and [10] for example). The data encryption is performed based on a symmetric cipher and the secret session encrypting key (SEK). Ensuring that only the valid members of the group have the SEK at any given time instance is the key management problem. To make this updating possible, another set of keys called the key-encrypting keys (KEKs) is involved. The KEKs are used to encrypt and deliver the updated SEK to the valid members of the group only. In order to obtain the desired security, the KEKs must be kept in a protected storage.

The underlying BE paradigm is to represent any privileged set of users as the union of s subsets of users of a particular form. A different key is associated with each one of these sets, and a user knows a key if and only if he belongs to the corresponding set. The broadcaster encrypts SEK s times employing the KEKs associated with the set in the cover. Consequently, each privileged user can easily access the data, but even a coalition of the non-privileged users cannot.

Security of the reported BE schemes was mainly considered via possible impacts of colluding the revoked users under assumption that the employed encryption techniques are secure ones.

Motivation for the Work. The main intention of this work was to consider some alternative attacking scenarios motivated by the following two issues: (i) KEKs have static nature - they are given to users at the very beginning and used later on during the entire "working life" of the system; (ii) a same SEK is encrypted a huge (usually) number of times by different KEKs and the corresponding ciphertexts are publicly available. As a result, knowledge of only one KEK can compromise the security of the entire BE system.

Particularly, a motivation was consideration of BE schemes resistance against dedicated time-data-memory trade-off attacks.

Contributions of the Paper. This paper points out to a generic vulnerability of certain BE schemes. This vulnerability can be effectively explored assuming chosen plaintext attacks, and in some cases even under ciphertext only attack. The developed methods for cryptanalysis are based on an attacking approach not taken into account in the security evaluations of the reported BE schemes. The proposed attacks are based on employment of a dedicated time-data-memory trade-off approach for cryptanalysis. Two algorithms for cryptanalysis are proposed, one related to the chosen plaintext attack scenario, and the other related to the ciphertext only scenario. The main characteristics of both algorithms are given regarding the required sample, and time and space complexities. The algorithms are employed for cryptanalysis some of the currently most interesting BE schemes showing that their security level is significantly below the claimed one. One of the main consequences of the proposed methods for cryptanalysis is the impact regarding the design requirements of the BE schemes in order to avoid the identified vulnerabilities.

Organization of the Paper. Section 2 contains a summary, relevant for this paper, of the background on BE and time-memory-data trade-off approaches for cryptanalysis. The attacking model and scenarios are specified in Section 3. Two developed algorithms for cryptanalysis are proposed in Section 4 including statements on theirs complexity and required sample. The evaluations of the currently most interesting BE schemes employing the proposed algorithms for cryptanalysis are given in Section 4 and high vulnerability of these schemes are pointed out. A concluding discussion is given in Section 5. The main characteristics of the particular BE schemes considered in Section 4 are summarized in Appendix A.

2 Background

2.1 Broadcast Encryption

Let KEK_i denotes a KEK employed in the system, and let ID_i denotes its name, i.e. its identification (ID). BE is based on the following approach. The system center generates all the employed KEKs. A user of the BE system is in advance provided with a subset of all KEKs employed in the system. Note that different users can have overlapping subsets of KEKs, but no one pair of users have the identical subset.

The BE procedures at the center and each of the users are based on the following. When the current SEK should be updated, the center finds a subset I of KEKs $\{KEK_i\}_{i\in I}$ such that each of the legitimate users possesses at least one of these keys and none of the un-legitimate users possesses any of these keys. The center encrypts the data with SEK, generates encrypted forms of SEK employing all KEK_i, $i \in I$, and broadcasts the following

$$< [header]; E_{SEK}(data) > = < [\{ \ (ID_i, E_{KEK_i}(SEK)) \ \}_{i\in I}]; E_{SEK}(data) > , \qquad (1)$$

where for simplicity we assume that the same encryption algorithm $E(\cdot)$ is employed for encryption of the data and KEKs.

Upon receiving (1), a legitimate receiver is able to find ID_i in its possession and based on the pair $(ID_i, E_{KEK_i}(SEK))$ it can recover SEK and the data based on the following:

$$SEK = E^{-1}_{KEK_i}(E_{KEK_i}(SEK)) \ , \tag{2}$$

$$data = E^{-1}_{SEK}(E_{SEK}(data)) \ , \tag{3}$$

where $E^{-1}(\cdot)$ denotes the decryption algorithm.

Note that the well recognized BE schemes reported in [13] and [7] follow the above paradigm, as well as the very recently reported BE scheme [8].

2.2 Time-Memory-Data Trade-Off Attack

This section yields an overview of the time-memory-data trade-off concept according to [6] and [3].

Let $f(\cdot)$ denote a one-way function, and K a secret key. Computing $f(K)$ is simple, but computing K from $f(K)$ is equivalent to cryptanalysis.

The time-memory trade-off concept is based on the following two phases: the precomputation phase which should be performed only once, and the processing phase which should be performed for reconstruction of a particular secret key.

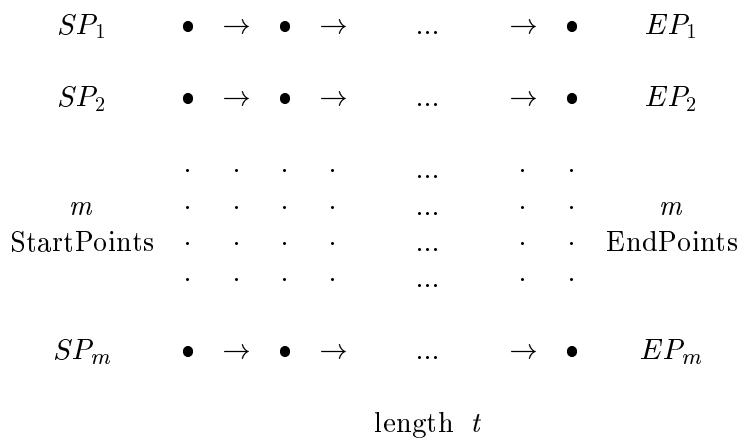

Figure 1: The underlying matrix for time-memory trade-off.

As part of the precomputation, the cryptanalyst chooses m starting points, $SP_1, SP_2, ..., SP_m$, each an independent random variable drawn uniformly from the key space $\{1, 2, ..., N\}$. For $1 \leq i \leq m$ he lets

$$X_{i0} = SP_i \tag{4}$$

and computes

$$X_{ij} = f(X_{i,j-1}) \ , \ \ 1 \leq j \leq t \ , \tag{5}$$

following the scheme given in Fig. 1. The parameters m and t are chosen by the cryptanalyst to trade-off time against memory.

The last element or endpoint in the ith chain (or row) is denoted by EP_i. Clearly,

$$EP_i = f^t(SP_i) \ , \tag{6}$$

where $f^t\cdot$ denotes the corresponding self-composition of $f(\cdot)$.

The complexity to construct the table is mt. However, to reduce memory requirements, the cryptanalyst discards all intermediate points as they are produced and sorts $\{SP_i, EP_i\}_{i=1}^{m}$ on the endpoints. The sorted table is stored as the result of this precomputation.

Now we explain how the above table can be employed for the cryptanalysis of a known encryption algorithm $E_K(\cdot)$ where K denotes the employed secret key. Suppose that the cryptanalyst has obtained the pair (Y_0, P_0) where

$$Y_0 = E_K(P_0) \; . \tag{7}$$

Accordingly, we consider the problem of recovering the secret key K when the encryption $E(\cdot)$ and corresponding decryption $D(\cdot)$ algorithms, the ciphertext Y_0, and the corresponding plaintext P_0 are known to the cryptanalyst.

Suppose that the following is valid

$$Y_1 = f(K) \; . \tag{8}$$

Note that the cryptanalyst can check if Y_1 is an endpoint in one "operation" because the $\{(SP_i, EP_i)\}$ are sorted on the endpoints. Accordingly, note the following:

- If Y_1 is not an endpoint, the key is not in the next to the last column in Fig. 1. (If it is there, Y_1, which is its image under f, would be an endpoint.)
- If $Y_1 = EP_i$, either $K = X_{i,t-1}$ (i.e., K is in the next to last column of Fig. 1), or EP_i has more than one inverse image. We refer to this latter event as a false alarm. If $Y_1 = EP_i$, the cryptanalyst therefore computes $X_{i,t-1}$ and checks if it is the key, for example by seeing if it deciphers Y_0 into P_0. Because all intermediate columns in Fig. 1 were discarded to save memory, the cryptanalyst must start at SP_i and recompute $X_{i,1}, X_{i,2}, ...$, etc. until he reaches $X_{i,t-1}$.
- If Y_1 is not an endpoint or a false alarm occurred, the cryptanalyst computes

  $$Y_2 = f(Y_1) \tag{9}$$

  and checks if it is an endpoint. If it is not, the key is not in the $(t-2)$-th column of Fig. 1, while if $Y_2 = EP_i$ the cryptanalyst checks if $X_{i,t-2}$ is the key.
- In a similar manner, the cryptanalyst computes

  $$Y_3 = f(Y_2), \; ..., \; Y_t = f(Y_{t-1})$$

  to check if the key is in the $(t-3)$-th, ..., or 1-st column of Fig. 1.

If all mt elements in the 1-st through t-th columns of Fig. 1 are different and if K is chosen uniformly from all possible values, the probability of success $Pr(S)$ would be mt/N. Only $2m$ words of memory and t operations are required, so the time-memory product has come into play. An exhaustive search with t operations has only $Pr(S) = t/N$, while a table lookup with m words of memory has only $Pr(S) = m/N$.

If the matrix in Fig. 1 has some overlap, but a fixed fraction of distinct elements, the probability of success is only lowered by the same fixed fraction. A mild amount of overlap therefore can be tolerated in the matrix without affecting the basic gain inherent in the time-memory trade-off. The analysis also neglects other constant and logarithmic factors (e.g., it counts an encipherment and check for Y_1 equal to an endpoint as one operation).

Finally, suppose that the secret key K initializes the internal state of a stream cipher, that this state has the same dimension as K, and that we have D different ciphertexts obtained by a stream cipher. The attack is successful if any one of the D given outputs can be found in the matrix corresponding to Fig. 1. Accordingly we can reduce the total number of points covered by the

matrix from about N to N/D, and still get (with high probability) a collision between the stored and actual states, obtaining a time-memory-data trade-off.

Assuming that P denotes the pre-processing complexity, the time-memory-data trade-off attack on stream ciphers proposed in [3] satisfies the following relation $P = N/D$, $tm^2D^2 = N^2$ for any $D^2 \leq t \leq N$. A typical point on this trade-off relation is $P = N^{2/3}$ pre-processing time, $t = N^{2/3}$ attack time, $m = N^{1/3}$ memory space, and $D = N^{1/3}$ available data. For $N = 2^{100}$ the parameters $P = t = 2^{66}$ and $m = D = 2^{33}$ are all (barely) feasible, [3].

3 Model and Scenarios for Cryptanalysis of Broadcast Encryption Schemes

3.1 Model under Cryptanalysis

We assume that the key management in the considered BE schemes is based on the following paradigm. In order to provide the legitimate users with the decryption key SEK_j at the time instance j, the following set of pairs H_j is publicly available

$$H_j = \{(ID_i, C_{i,j})\}_{i \in I(j)} \ , \tag{10}$$

where

$$C_{i,j} = E_{KEK_i}(SEK_j)) \ , \tag{11}$$

and ID_i is the name of the key KEK_i employed for encryption of SEK_j using the symmetric encryption algorithm $E(\cdot)$, and $I(j)$ is a time dependent subset of integers $\{1, 2, ..., I\}$.

Note that in the considered model $E(\cdot)$ could be a block-cipher or a stream-cipher.

Also, we assume that the following is valid:

- For each $i = 1, 2, ..., I$,
  - KEK_i is a randomly generated binary vector of dimension L and $2^L >> I$;
  - ID_i only indicates that the encrypted form $E_{KEK_i}(SEK_j)$ of SEK_j is obtained employing the key KEK_i and does not provide any information on the binary vector KEK_i;

- For each $j = 1, 2, ...,$
  - SEK_j is a binary vector of dimension L;
  - each $I(j)$ is a different subset of $\{1, 2, ..., I\}$;
  - a certain overlapping between different sets $I(j)$ could occur;

- The employed encryption algorithm $E(\cdot)$ is a secure one so that any $C_{i,j} = E_{KEK_i}(SEK_j))$ does not yield any information on KEK_i and SEK_j.

3.2 Scenarios under Cryptanalysis

The attacker's knowledge is limited as follows:
- The attacker knows the entire structure of the BE scheme under cryptanalysis including the employed encryption algorithm $E(\cdot)$;
- The attacker does not know any of the keys KEK_i, $i = 1, 2, ..., I$, employed in the considered BE scheme.

The goal of the attacker is to recover at least one of the secret keys KEK_i, $i = 1, 2, ..., M$, employed in the BE scheme. We emphasize this last point as it constitutes the main difference in comparison with recovery of a single key employed in a block or stream cipher, and it is one of the main origins for the weaknesses pointed out in this paper.

3.2.1 Scenario A

In this scenario, it is assumed that the attacker has the following data for cryptanalysis

$$(H_j, SEK_j = SEK)\ ,\ j = 1, 2, ..., J\ .$$

This scenario corresponds to the chosen plaintext based cryptanalysis.

3.2.2 Scenario B

In this scenario, it is assumed that the attacker has the following data for cryptanalysis

$$(H_j, SEK_j)\ ,\ j = 1, 2, ..., J\ .$$

This scenario corresponds to the ciphertext only based cryptanalysis assuming that the attacker is a legitimate user of the system.

4 Novel Methods for Cryptanalysis of Broadcast Encryption Schemes

4.1 Underlying Ideas

The main origins for developing the attacks are the following characteristics of the BE schemes:
(a) the entire secret key of a BE scheme, known only to the broadcasting center, consists of a huge number of the particular secret keys $\{KEK_i\}_{i=1}$;
(b) in a BE scheme, each session key is encrypted a number of times employing different KEKs;
(c) any user of a BE system does not know any of the assigned keys because they are in a tamper resistant storage, and accordingly the system should be considered as broken even if a user can recover only one of the KEKs employed in the system.

Illustrative Example. The CST BE [13] is based on a secret key corresponding to a binary balanced tree in which each KEK is assigned to a node of this tree as illustrated in Fig. 2. The secret key consists of $2N - 1 = 63$ KEKs, and it is assumed that there are $N = 32$ receiving entities. Accordingly, the secret key consists of $2N - 1$ independent parts which could be considered as the randomly generated ones.

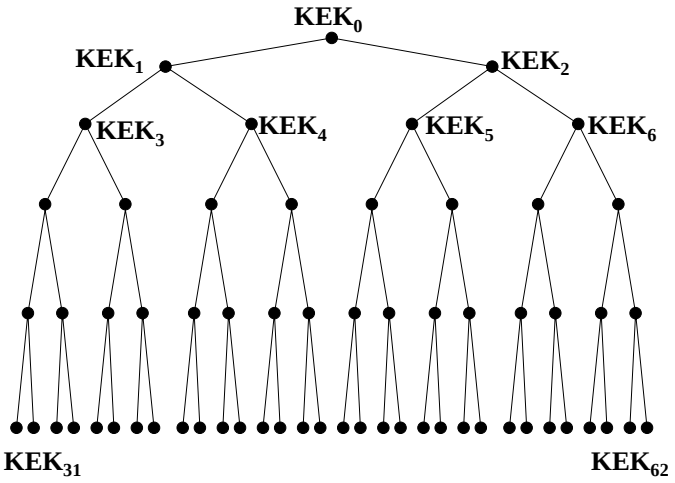

Figure 2: An illustration of the secret key employed in certain BE schemes.

The above characteristics of BE open a door for developing cryptanalytic methods based on an attacking approach not taken into account in the reported security evaluations of the schemes.

The developed cryptanalysis could be called "list cryptanalysis" following the term "list decoding" and its related similarity. Recall that "list decoding" assumes that we have succeeded in decoding if the codeword is in the list of the candidate codewords. In our cryptanalysis we assume that the goal is achieved if it is possible to recover at least one key from the list of the all KEKs employed in the BE scheme.

This paper proposes dedicated time-memory-data trade-off based methods for cryptanalysis of the BE key management scheme. The proposed methods employ the following underlying ideas:

- develop a dedicated time-memory-data trade-off based cryptanalysis assuming the chosen plaintext attack corresponding to the attacking Scenario A;
- develop a dedicated time-data trade-off based cryptanalysis assuming the ciphertext only attack corresponding to the attacking Scenario B.

These approaches are different from reported ones related to the time-memory trade-off based cryptanalysis of block ciphers and time-memory-data trade-off based ones for stream ciphers. The differences are consequences of the attacking nature regarding BE schemes on one hand side and block or stream ciphers on the other hand. Regarding these issues, as an illustration note the following:
- the time-memory trade-off based cryptanalysis of block ciphers [6] (see [15], as well) assumes that the attacker's goal is to recover the employed secret key when at least one plaintext-ciphertext pair is known; in the BE setting with a block cipher, the cryptanalysis is based on a collection of the ciphertext of a same message generated employing different secret keys, and the attacker's goal is to recover at least one of these employed secret keys;
- the time-memory-data trade-off cryptanalysis of the stream ciphers [3] assumes that all the issues are related to recovering an internal state of the considered stream cipher; in the BE case even if the employed $E(\cdot)$ is a stream cipher, again the entire consideration is related to the secret key only, i.e. the internal state evolution appears as not relevant.

Particularly note the following differences:
- when a block cipher is employed as the algorithm $E(\cdot)$ in the BE scheme, the developed methods provide a gain proportional to the number of available plaintext-ciphertext pairs; on the other hand note that the a time-memory trade-off attack [6] does not provide any additional gain when more than only one ciphertext-plaintext pair is available;
- when a stream cipher is employed in the BE scheme, the developed time-memory-data trade-off based attack is related only to the stream cipher secret key and not to its internal state; on the other hand, the time-memory-data trade-off based attacks reported in [3] are related to the internal state of the cipher and they become infeasible when the internal state size is much larger than the employed key.

4.2 Attacking Scenario A

The algorithm for cryptanalysis of BE schemes under the attacking Scenario A (a particular chosen plaintext attack) consists of the following two main phases:
- pre-processing phase with the following main characteristics:
  - it should be done only once;
  - it depends on the employed encryption $E(\cdot)$ and the chosen SEK;
  - it is independent of the secret keys KEKs employed in the system;
- processing phase with the following main characteristics:

- it should be performed for attacking a particular BE scheme where the employed set of keys $\{KEK_i\}_{i=1}^{i}$ is unknown assuming that a certain sample is available;
- it employs the output of the pre-processing phase; and
- it yields, as the expected output, in recovering at least one of the employed KEKs.

4.2.1 Algorithm for Cryptanalysis

Algorithm A

- *Pre-Processing*
  - *Input Data*: SEK, and the algorithm parameters M and T.
  - *Pre-Processing Steps*
    For $m = 1, 2, ..., M$, do the following:
    1. randomly select an L-dimensional binary vector $X_m(0)$
    2. For $t = 1, 2, ..., T$, perform the following recursive calculation
    $$X_m(t) = E_{X_m(t-1)}(SEK) \tag{12}$$
    3. Memorize the pair $X_m(0), X_m(T)$.
  - *Output*: The two-column matrix of the pairs memorized in the pre-processing step 3.
- *Processing*
  - *Input Data*: sequence of D different values $C_{i,j} = E_{KEK_i}(SEK))$, $i \in I(j)$, $j = 1, 2, ..., J$.
  - *Processing Steps*
    For each i, j, $i \in I(j)$, $j = 1, 2, ..., J$, do the following:
    1. Set $t = 0$ and $X_t = C_{i,j}$.
    2. Check the identity of the considered X_t to any of the second column elements $X_m(T)$ of the matrix generated in the pre-processing phase; if for some index m the identity appears, go to the processing step 4; otherwise go to the processing step 3.
    3. If $t \leq T$, calculate $X_{t+1} = E_{X_t}(SEK)$ and go to the processing step 2; if $t > T$, go to the processing step 5.
    4. (a) Select the corresponding $X_m(0)$ and set $X_0 = X_m(0)$;
       (b) perform the following iterative calculation: $X_{t+1} = E_{X_t}(SEK)$ until $X_{t+1} = C_{i,j}$;
       (c) memorize the pair $(X_t, C_{i,j})$.
    5. Select a previously not considered $C_{i,j}$ and go to the processing step 1.
  - *Output*: Set of the recovered KEKs obtained via the memorized pairs in the processing step 3.

Remark 1. For the simplicity of presentation, it is assumed that the sample for cryptanalysis $C_{i,j} = E_{KEK_i}(SEK))$, $i \in I(j)$, $j = 1, 2, ..., J$, is available before the processing phase starts, but this is not necessary, and the processing can work in the same manner when the data are available sequentially, i.e. the processing starts when $C_{i,1} = E_{KEK_i}(SEK))$, $i \in I(1)$, is available and continues when a new sample becomes available.

4.2.2 Complexity of Cryptanalysis

This section yields the complexity analysis of the proposed Algorithm A assuming that the expected number of KEKs it recovers is equal to k, and that D is the expected cardinality of the union of the sets $I(j)$, $j = 1, 2, ..., J$.

According to the Algorithm A structure and the results reported in [6] and [3], the following statements can be proved.

Proposition 1. The pre-processing phase time complexity of Algorithm A is $O(k2^L D^{-1})$.

Proposition 2. The processing phase time complexity of Algorithm A is $O(k^2 2^{2L} M^{-2} D^{-2})$.

Proposition 3. Algorithm A provides different possible trade-offs between the required memory M, sample dimension D and time complexity of the processing T, assuming that the following trade-off condition holds:

$$TM^2D^2 = k^2 2^{2L} \ . \tag{13}$$

Remark 2. Note that $k = D$ corresponds to [6] as our scheme can be viewed as [6] with kN/D instead of N.

4.3 Attacking Scenario B

For this attacking scenario, the developed algorithm for cryptanalysis does not require pre-processing phase. In fact, the pre-processing phase of the attacking Scenario A becomes now, the main processing phase. The main phases of this algorithm are: (i) sample collection; (b) processing over collected samples.

4.3.1 Algorithm for Cryptanalysis

Algorithm B

- *Input Data*:
  - sequence of D different values $C_{i,j} = E_{KEK_i}(SEK_j))$, $i \in I(j)$, $j = 1, 2, ..., J$, where SEK_j , $j = 1, 2, ..., J$, are known;
  - algorithm parameter T.
- *Processing Steps*
  For each SEK_j, $j = 1, 2, ..., J$, do the following:
  1. Select a previously not considered SEK_j, set $t = 0$ and randomly select an L-dimensional vector X_0.
  2. Calculate $X_{t+1} = E_{X_t}(SEK_j)$.
  3. Compare X_{t+1} with all $C_{i,j}$:
     (a) If X_{t+1} is identical to $C_{i,j}$ for some i, memorize the corresponding pair $(X_t, C_{i,j})$;
     (b) If $t \leq T$, set $t \rightarrow t + 1$, and go to the processing step 2;
     (c) If $t > T$ go to the processing step 1.
- *Output*: Set of the recovered KEKs obtained via the memorized pairs in the processing step 3(a).

4.3.2 Complexity of Cryptanalysis

According to the Algorithm B processing steps the following statements can be directly proved.

Proposition 4. Assuming that Algorithm B should recover k KEKs the processing time complexity is $O(k2^L D^{-1})$ where D is the expected cardinality of the union of the sets $I(j)$, $j = 1, 2, ..., J$.

Proposition 5. Algorithm B provides different possible trade-offs between the required sample dimension D and time complexity of the processing T, assuming that the following trade-off condition holds:

$$TD = 2^L k \ . \tag{14}$$

Remark 3. Note that $k = D$ corresponds to the preprocessing cost [6].

5 Vulnerability of Particular Broadcast Encryption Schemes CST, SD and LSD

The popular Complete Sub-Tree (CST), Subset Difference (SD) and Layered Subset Difference (LSD) based key management schemes have been reported in [13] and [7], and a number of related applications have been discussed in [9] and [10]. The main characteristics of these schemes are summarized in Appendix A.

This section shows that these schemes are vulnerable under the attacking Scenarios A and B, and it yields complexity of theirs cryptanalysis employing the proposed Algorithms A and B.

The results on the communications overhead of CST, SD and LSD, reported in [13] and [7], respectively, imply the following proposition.

Proposition 6. Assuming that there are J sessions of SEK updating, and that each of these sessions assumes R random revocations from a set of N users, the sample available for cryptanalysis of CST, SD and LSD based BE schemes is upper-bounded by $JR\log_2(N/R)$, $2JR$ and $4JR$, respectively.

According to Propositions 1, 2, 4 and 6, when $k = 1$, the complexities of cryptanalysis of CST, SD and LSD based key management schemes for BE are summarized in Table 1 and Table 2 regarding the attacking Scenarios A and B, respectively, assuming the following:
- the schemes include N users in total;
- each of the employed KEKs and SEKs consists of L bits;
- the sample for cryptanalysis is obtained from J sessions of SEK updating, and each of these sessions assumes R random revocations.

Illustrative numerical examples related to Tables 1 and 2 are given in Tables 3 and 4, respectively.

6 Concluding Discussion

The vulnerabilities of BE schemes identified in this paper originate from the following generic characteristics of these schemes: (i) BE schemes employ "internal" secret key consisting of a huge number of independent static components (KEKs); (ii) BE schemes encrypt the session key (SEK) a huge number of times employing different KEKs and these ciphertexts are publicly available; (iii) the possibility for recovering one active KEK, a part in use in the internal BE secret key implies weakness of the entire scheme.

Table 1: Attacking Scenario A (chosen plaintext attack): Complexity of recovering one KEK (k=1) in CST, SD and LSD based key management schemes assuming that the schemes include N users in total, each of the employed KEKs and SEKs consists of L bits, the sample for cryptanalysis is obtained from J sessions of SEK updating and each of these sessions assumes R random revocations, and a memory of dimension M is available.

	pre-processing time complexity	processing time complexity
CST [13]	$O(2^L(JR\log_2(N/R))^{-1})$	$O(2^{2L}(MJR\log_2(N/R))^{-2})$
SD [13]	$O(2^L(2JR)^{-1})$	$O(2^{2L}(2MJR)^{-2})$
LSD [7]	$O(2^L(4JR)^{-1})$	$O(2^{2L}(4MJR)^{-2})$

Table 2: Attacking Scenario B (ciphertext only attack): Complexity of recovering one KEK (k = 1)in CST, SD and LSD based key management schemes assuming that the schemes include N users in total, each of the employed KEKs and SEKs consists of L bits, the sample for cryptanalysis is obtained from J sessions of SEK updating and each of these sessions assumes R random revocations.

	processing time complexity
CST [13]	$O(2^L(JR\log_2(N/R))^{-1})$
SD [13]	$O(2^L(2JR)^{-1})$
LSD [7]	$O(2^L(4JR)^{-1})$

This paper proposes methods for cryptanalysis of BE schemes via KEK by KEK recovering with complexity significantly lower than an exhaustive search over all KEK possibilities. The developed methods for cryptanalysis are based on the dedicated time-data-memory and time-data trade-off approaches employing chosen plaintext and ciphertext only attacks, respectively.

The proposed algorithms for cryptanalysis are employed for security evaluation of the currently most interesting BE schemes, CST [13], SD [13] and LSD [7], and it is shown that these schemes are highly vulnerable, implying that the security level of these schemes are far below the claimed one, and at least from the information-theoretic point of view they appear as insecure ones. Also note that the very recently reported scheme [8] suffers from the same vulnerability as the CST, SD and LSD based schemes.

The developed methods for cryptanalysis indicate requests for developing improved BE schemes which should be resistant against the proposed attacking approaches.

Table 3: Attacking Scenario A (chosen plaintext attack): Illustrative numerical examples on complexity of recovering one KEK ($k = 1$)in CST, SD and LSD based key management schemes assuming that the schemes include $N = 10^8$ users in total, each of the employed KEKs and SEKs consists of $L = 128$ bits, the sample for cryptanalysis is obtained from $J = 1000, 10000$, sessions of SEK updating and each of these sessions assumes $R = 10^6$ random revocations, and a memory of dimension $M = 2^{57}$ is available.

	pre-processing time complexity		processing time complexity	
	$J = 1000$	$J = 10000$	$J = 1000$	$J = 10000$
CST [13]	$\sim 2^{98}$	$\sim 2^{95}$	$\sim 2^{69}$	$\sim 2^{62}$
SD [13]	$\sim 2^{104}$	$\sim 2^{101}$	$\sim 2^{80}$	$\sim 2^{73}$
LSD [7]	$\sim 2^{103}$	$\sim 2^{100}$	$\sim 2^{79}$	$\sim 2^{72}$

Table 4: Attacking Scenario B (ciphertext only attack): Illustrative numerical examples on complexity of recovering one KEK ($k = 1$)in CST, SD and LSD based key management schemes assuming that the schemes include $N = 10^6$ users in total, each of the employed KEKs and SEKs consists of $L = 64, 128$ bits, the sample for cryptanalysis is obtained from $J = 1, 100, 1000$ sessions of SEK updating and each of these sessions assumes $R = 10^4$ random revocations.

	processing time complexity					
	$L = 64$			$L = 128$		
	$J = 1$	$J = 100$	$J = 1000$	$J = 1$	$J = 100$	$J = 1000$
CST [13]	$\sim 2^{44}$	$\sim 2^{38}$	$\sim 2^{35}$	$\sim 2^{108}$	$\sim 2^{102}$	$\sim 2^{99}$
SD [13]	$\sim 2^{50}$	$\sim 2^{44}$	$\sim 2^{41}$	$\sim 2^{114}$	$\sim 2^{108}$	$\sim 2^{105}$
LSD [7]	$\sim 2^{49}$	$\sim 2^{43}$	$\sim 2^{40}$	$\sim 2^{113}$	$\sim 2^{107}$	$\sim 2^{104}$

References

[1] M. Abdalla, Y. Shavitt and A. Wool, "Key management for restricted multicast using broadcast encryption", *IEEE/ACM Trans. Networking*, vol. 8, pp. 443-454, Aug. 2000.

[2] S. Berkovits, "How to broadcast a secret", EUROCRYPT '91, *Lecture Notes in Computer Science*, vol. 547, pp. 536-541, 1991.

[3] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers", ASIACRYPT 2000, *Lecture Notes in Computer Science*, vol. 1976, pp. 1-13, 2000.

[4] A. Fiat and M. Naor, "Broadcast encryption", Advances in Cryptology - CRYPTO93, *Lecture Notes in Computer Science*, vol. 773, pp. 480-491, 1994.

[5] A. Fiat and M. Naor, "Rigorous time/space trade-offs for inverting functions", *SIAM J. Computing*, vol. 29, pp. 790-803, 1999.

[6] M.E. Hellman, "A cryptanalytic time-memory trade-off", *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 401-406, July 1980.

[7] D. Halevy and A. Shamir, "The LCD broadcast encryption scheme", CRYPTO 2002, *Lecture Notes in Computer Science*, vol. 2442, pp. 47-60, 2002.

[8] N. Jho, J. Y. Hwang, J. H. Cheon, M-H. Kim, D. H. Lee, E. S. Yoo, "One-way chain based broadcast encryption scheme", EUROCRYPT 2005, *Lecture Notes in Computer Science*, accepted for publication. (preliminary paper version available at http://eprint.iacr.org/2005/073b.pdf)

[9] J. Lotspiech, S. Nusser and F. Prestoni, "Broadcast encryption's bright future", *IEEE Computer*, vol. 35, pp. 57-63, Aug. 2002.

[10] J. Lotspiech, S. Nusser and F. Prestoni, "Anonymous trust: Digital rights management using broadcast encryption", *Proc. IEEE*, vol. 92, pp. 898-909, June 2004.

[11] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*. Boca Roton: CRC Press, 1997.

[12] J. Mitra and P. Sarkar, "Trade-Off Attacks on Multiplications and T-Functions", ASIACRYPT 2004, *Lecture Notes in Computer Science*, vol. 3329, pp. 468-482, Dec. 2004.

[13] D. Naor, M. Naor and J. Lotspiech, "Revocation and tracing schemes for stateless receivers", CRYPTO 2001, *Lecture Notes in Computer Science*, vol. 2139, pp. 41-62, 2001.

[14] D. Naor and M. Naor, "Protecting cryptographic keys: The trace-and-revoke approach", *IEEE Computer*, vol. 36, pp. 47-53, July 2003.

[15] P. Oechslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off", CRYPTO 2003, *Lecture Notes in Computer Science*, vol. 2729, pp. 617-630, 2003.

7 Appendix A: Background on CST, SD and LSD Key Management Schemes

In [13], a generic framework, is given by encapsulating several previously proposed revocation methods called Subset-Cover algorithms. These algorithms are based on the principle of covering all non-revoked users by disjoint subsets from a predefined collection, together with a method for assigning the "static" keys to subsets in the collection. An important consequence of this framework is the separation between long-lived keys (KEKs) and short-term keys (SEKs). Two types of revocation schemes in the subset-cover framework, are proposed in [13] with a different performance tradeoff. Both schemes are tree-based, namely the subsets are derived from a virtual tree structure imposed on all receivers in the system. The first proposed scheme, Complete Sub-Tree (CST) scheme, requires a message length of $Rlog(N/R)$ and storage of $logN$ keys at the receiver. The second technique for the covering is the Subset Difference (SD), [13]. The improved performance of SD algorithm is primarily due to its more sophisticated choice of the covering sets in the following way.

Let i be any vertex in the tree and let j be any descendent of i. Then $S_{i,j}$ is the subset of leaves which are descendents of i but are not descendents of j.

Note the following: (i) $S_{i,j}$ is empty when $i = j$; (ii) otherwise, $S_{i,j}$ looks like a tree with a smaller subtree cut out; (iii) an alternative view of this set is a collection of subtrees which are hanging off the tree path from i to j.

The SD scheme covers any privileged set P defined as the complement of R revoked users by the union of $O(R)$ of these $S_{i,j}$ sets, providing that a receiver stores $O((logN)^2)$ keys.

What is shown in [7] is that SD collection of sets can be reduced: The basic idea of the Layered Subset Difference (LSD) scheme [7] is to retain only a small collection of the $S_{i,j}$ sets used by the SD scheme, which suffices to represent any privileged set P as the union of $O(R)$ of the remaining sets, with a slightly larger constant.

The subcollection of sets $S_{i,j}$ in the LSD scheme is defined by restricting the levels in which the vertices i and j can occur in the tree. This approach is based by specifying some of the $log(N)$ levels as "special". The root is considered to be at a special level, and in addition we every level of depth $k \cdot \sqrt{log(N)}$ for $k = 1, ..., \sqrt{log(N)}$, as special (we assume that these numbers are integers). Thus, there are $\sqrt{log(N)}$ special levels which are equally spaced at a distance of $\sqrt{log(N)}$ from each other. The collection of levels between (and including) adjacent special levels is defined as a "layer".

Since there are fewer possible sets, it is possible to reduce the number of initial keys given to each user. In [7], it is shown that if we allow the number of sets in the cover to grow by a factor of two, we can reduce the number of keys from $O(log^2(N))$ to $O(log^{3/2}(N))$ and then this technique was extended and it has been shown how to reduce the number of keys to $O(log^{1+\epsilon}(N))$ for $\epsilon > 1$.

Suppose that nodes i, k, j, occur in this order on a path from the root to a leaf, i is not located on a special level, i and j do not belong to the same layer, and k is located on the first special layer from i to j. In this case a subset $S_{i,j}$ is not included in the basic LSD but it can be described using other subsets included in the LSD collection as follows:

$$S_{i,j} = S_{i,k} \bigcup S_{k,j} \quad .$$

Accordingly, instead of a ciphertext encrypted under the subset key $S_{i,j}$ as in SD, two ciphertexts obtained by $S_{i,k}$ and $S_{k,j}$ should be broadcasted in LSD scenario. Therefore, the communication overhead increases at most twice in comparison with SD, but on the other hand LSD yields the storage reduction at a receiver.