Novel Method for Implementation of Certain Key Management Schemes to Minimize Secret Storage

Miodrag J. Mihaljević
Mathematical Institute
Serbian Academy of Sciences and Arts
Kneza Mihaila 35, Belgrade, Serbia and Montenegro
e-mail: miodragm@turing.mi.sanu.ac.yu

Hideki Imai
The University of Tokyo
Institute of Industrial Science
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan

Abstract—The problem of minimizing the amount of secret information (secret bits) required for certain key management schemes related to data access control techniques is addressed. Particularly note that importance of the secret storage minimization originates from the fact that this storage should be read-proof and tamper-proof one. This paper points out to a novel approach for minimization of the secret storage dimension and compare this approach with a straightforward one based on establishing a virtual secret storage employing data encryption. The novel approach yields: (i) provable security; (ii) it is not based on storing the encrypted data into a public storage, and (iii) it requires lower processing complexity. The proposed approach yields the secret storage minimization via exchange of a secret storage to a public one based on the efficient one-way mapping of the secret bits yielding significant additional flexibility and reduction of the secret storage overhead at the user’s side as an appropriate trade-off with the required public storage and processing complexity. The overheads of the proposed technique are compared with the related previously reported ones, and advantages of the novel approach are pointed out.

I. INTRODUCTION

The cryptographic keys management is usually a core issue of the digital rights management (DRM). Particularly, the keys management appears as the substantial issue regarding conditional access to data or contents delivered via broadcasting.

The main objective of this paper is to point out and compare (with the related previously reported methods) a novel method relevant for efficient implementation of certain key management schemes assuming the key management scenarios for broadcast encryption.

Broadcast encryption (BE) schemes define methods for encrypting content so that only privileged users are able to recover the content from the broadcast. Later on, this flagship BE application has been extended to another one - media content protection (see [8] or [3], for example).

When cryptography is used for securing broadcasting communications or stored data, usually it is based on encryption/decryption employing a session-encrypting key (SEK) shared by the parties. Ensuring that only the valid members of the group have 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) should be involved so that it can be used to encrypt and deliver the updated SEK to the valid members of the group. Hence, the key management problem reduces to the problem of distributing KEKs to the members such that at any given time instance all the valid members can be securely updated with the new SEK.

Among other possible classifications, according to the nature of employed KEKs, key management schemes could be categorized as follows: (i) schemes with updatable or fixed keys, and (ii) information theoretically secure or computationally secure schemes. The schemes with updatable keys assume that a receiver has an internal state which is updatable so that KEKs are time varying. The schemes with fixed keys assume that a set of the keys is assigned to a receiver once and should not be changed during the receiver entire life. These KEKs should be kept in a secret (protected) storage. It is common to call the receivers with updatable and fixed keys as the stateful and stateless ones, respectively. The information-theoretically secure scheme employ only KEKs which are realizations of a random process. The computationally secure schemes involve computationally secure cryptographic primitives for specification of KEKs.

The basic idea in the most efficient stateless (with not updatable keys) broadcast encryption schemes is to represent any privileged set of users as the union of subsets of a particular form. A different key is associated with each one of these subsets, and a user knows a key if and only if he belongs to the corresponding set.

Employment of any key management scheme in a communications system (for example) introduces certain system overheads. The main overheads of a computationally secure scheme with fixed keys at a receiver are the following ones: (i) required secret storage at receiver; (ii) required public storage at a receiver; (iii) processing overhead at a receiver; (iv) communications overhead.

The basic trade-offs between the overheads are secret-storage/communications and secret-storage/processing ones. Other trade-offs are possible as well.

Two remarkable mainly secret-storage/communications trade-offs are proposed in [8] and [2]. In [8], two methods called complete subtree (CST) and subset difference (SD) are proposed. 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 KEKs to subsets in the collection. Following [8], another trade-off approach called layered subset difference (LSD) is proposed in [2].

Motivation of the work.
A number of recently proposed key management schemes for broadcast encryption (SD [8], LSD [2], and the reconfigurable key management schemes [5]-[6]) requires a significant amount of the secret data to be stored at a receiver which appears to be not appropriate in certain scenarios implying the requests for minimization of the secret storage overhead. Particularly note that importance of the secret storage minimization originates from the fact that this storage should be read-proof and tamper-proof one. The above minimization implies a requirement for assigning multiple roles to the same secret bits via one-way mapping. The reported methods for one-way mapping of the secret bits do not yield the required low overheads at the receiver side because they are usually based on complex mappings (and some of them include employment of the public key cryptography) and imply large public storage and/or processing complexity overheads. An alternative approach for reducing the required hard secret storage is based on establishing a virtual secret storage based on encryption. Accordingly, the motivations for this work include: (i) to point out an alternative low complexity mapping method and particularly to avoid any employment of the public key cryptography in order to provide implementation based only on very simple arithmetic and logical operations like mod2 addition, integer addition and look-up table operations; and (ii) to compare the proposed mapping with the virtual secret storage approach based on KEKs encryption. An additional motivation for the work is to point out appropriate techniques required for implementation of the reconfigurable key management and to support the generic framework of assigning different roles to the secret key bits which is of substantial interest for obtaining flexible reconfigurable key management suitable to highly dynamical revocation scenarios.

Contributions of the paper.
This paper points out a novel method for efficient implementation of certain key management schemes for broadcast encryption yielding the minimization of the required secret storage and compare this implementation method with the related previously reported ones. The proposed minimization of the required secret storage is based on secret to public storage exchange and employment of one-way mapping. A particular approach for one-way mapping of the secret bits is pointed out. The employment of this mapping method in SD and LSD key management schemes for broadcast encryption is discussed and compared with the original scheme and a related scheme which employs the virtual secret storage instead of the real one. The discussed method for reducing the amount of secret information to be stored at a user based on an appropriate trade-off regarding reducing the secret storage overhead at the expense of public storage and certain processing, but without involvement of public key cryptography and related complex operations.

In comparison with the straightforward minimization of the required secret storage employing the virtual secret storage established via the data encryption, the considered novel approach has the following advantages: (i) it yields provable security; (ii) it is not based on storing the encrypted data into a public storage, and (iii) it requires lower processing complexity.

Organization of the paper.
Section II yields a preliminary analysis of the addressed problem. Section III proposes a generic framework for reducing the secret storage required by considered key management schemes and a particular technique for one-way mapping of the secret bits required for the minimization of secret storage. Modified SD and LSD based key management schemes with minimized secret storage requirements are discussed in Section IV. Comparison of the proposed technique and the related previously reported ones is given in Section V. Concluding discussions are given in Section VI. The main characteristics of the employed one-way mapping are summarized in Appendix A. An overview of SD and LSD based key management techniques is given in Appendix B.

II. Preliminaries

There are the following two main approach for providing secrecy of KEKs to be stored at a receiver:

- **direct**: establishing a protected storage which in a direct manner preserves the data secrecy;
- **simple indirect** establishing a protected storage for a secret seed only, employment encryption of the data and storing the encrypted data in a public storage which acts as a virtual secret storage.

Implementation of the direct approach assumes establishing of a read-proof and tamper-proof secure storage, and this issue is beyond the scope of this paper.

In a simplified scenario, we assume that implementation of a simple indirect approach is based only on establishing a virtual secret storage employing a suitable encryption technique providing that a small read-proof and tamper-proof storage for the secret seed is available. Let $S$ be the secret seed and, for simplicity, let a symmetric encryption technique be employed. So, let $E_S(\cdot)$ and $E_S^{-1}(\cdot)$ denote the encryption and decryption operations, respectively. Accordingly, the basic paradigm of a virtual secret storage implementation is as follows:

- keep $S$ in a (hard) secure storage;
- generate the encrypted version of the data $D$ employing $E_S(D, P)$ where $P$ are certain public data, and store the encrypted data in a public storage;
- when required, access to the stored data employing the corresponding decryption, $D = E_S^{-1}(E_S(D, P), P)$.

Both of the above approaches (direct and simple indirect) have certain drawbacks.
The main problem with the direct approach is that it requires employment of a tamper-proof storage for a huge data, and implementation of such a storage could be very complex.

The simple indirect approach significantly reduces the implementation problem of a tamper-proof storage because only the secret seed should be kept in this storage, and encrypted data could be stored in (literally speaking) any storage, but the simple indirect approach also suffers due to the following problems:

- the encrypted data are available for cryptanalysis;
- the security of simple virtual storage rests on the security of the employed encryption technique, and its security is usually only a heuristic one;
- the implementation requires certain additional overheads; the gain in reducing the required (hard) secret storage is obtained at the following expenses: (i) requirement for additional data processing (encryption/decryption), and (ii) requirement for a public storage which could be significantly larger than the (hard) secret storage because it should store the encrypted version of the data as well as the public information \( P \).

So, it is an important issue to consider a possibility for developing a virtual secret storage with the following attributes:

- does not require exposing of the encrypted data;
- yields not heuristic but provable security;
- require smaller processing and public storage overheads in comparison with the simple virtual approach.

III. A METHOD FOR MINIMIZATION OF REQUIRED SECRET STORAGE

This section overviews the method proposed in [7] for reducing the required secret storage in certain key management schemes.

A. A Generic Framework for Minimization of the Secret Storage

Suppose that a key management scheme with not-updateable keys requires that the following \( I \) KEKs should be stored in a protected storage (secret storage) at a receiver: \( KEK_1, KEK_2, ..., KEK_I \).

Let the following hold:
- \( S \) is a secret seed, \( R \) is a public randomization parameter, and \( f(S, R) \) is a cryptographic one-way mapping;
- \( \{ R_i \}_{i=1}^I \) is the public data set;
- \( S \) is \( k \)-dimensional binary vector, and \( KEK_i, R_i, Y_i = f(S, R_i), i = 1, 2, ..., I \), are binary \( n \)-dimensional vectors, \( k > n \).

Let \( Q_i, i = 1, 2, ..., I \), be \( n \)-dimensional binary vectors calculated as follows:

\[
Q_i = KEK_i \oplus f(S, R_i), \quad i = 1, 2, ..., I, 
\]

where \( \oplus \) denotes bit-by-bit \( mod2 \) addition.

The above statements imply a framework for minimization of the required secret storage at the receiver employing the following approach:

- keep \( S \) in the secret storage;
- keep \( \{ R_i \}_{i=1}^I \) and \( \{ Q_i \}_{i=1}^I \) in the public storage;
- when required, calculate any \( KEK_i \), employing the following:

\[
KEK_i = f(S, R_i) \oplus Q_i, \quad i = 1, 2, ..., I.
\]

Accordingly, assuming the existence of a suitable function \( f(\cdot) \), the proposed framework provides minimization of the secret key storage because instead of all \( I \) KEKs, only the seed \( S \) should be stored. The expense for the secret storage minimization is the requirement for an additional public storage for \( \{ R_i \}_{i=1}^I \) and \( \{ Q_i \}_{i=1}^I \), as well as an additional processing overhead for \( f(\cdot) \) calculation.

B. A Secret Bits Mapping Technique

This section points out to a particular secret bits mapping technique (i.e. \( f(\cdot) \) and from the previous section) which originates from the results reported in [1] and [6] and fulfills the requirements of the proposed generic framework, and it does not involve the public key cryptography.

Let \( G(s, p) \) be a cryptographic pseudorandom number generator which for a given secret seed \( s \) and a public randomization parameter \( p \) generate a pseudorandom binary sequence.

Let \( S \) be the mapping secret seed in form of a binary \( k \)-dimensional vector, and for \( i = 1, 2, ..., I \), let the public information associated to each \( K_i \), be in form of three binary vectors \( P_1, P_2, \) and \( P_3 \), of dimension \( n \), with \( k > 2n \). The vectors \( P_1 \) and \( P_2 \) are selected randomly and independently, and the vector \( P_3 \) is a deterministic function of \( S, P_1, P_2, \) and \( K_i \).

Mapping \( A \) is defined as follows:

1) employing \( G(S, P_1) \) generate an \( nq \)-length binary sequence yielding an equivalent length-\( n \) sequence \( D_i = \{d_{i,t}\} \) with symbols from \( \{0,1\} \);
2) employing \( G(S, P_2) \) generate length-\( m \) binary sequence \( X_i \) where \( m = \sum_{j=1}^{n} d_{i,j} \) and \( m \geq 2n \);
3) generate the binary sequence \( Y_i = \{y_{i,j}\} \) via decimation of \( X_i \) by \( D_i \) as follows:

\[
y_{i,j} = x_{i,j+\ell}, \quad \ell = \sum_{t=1}^{j} d_{i,t}, \quad j = 1, 2, ..., n; \quad (1)
\]

4) calculate \( K_i = Y_i \oplus P_3 \).

IV. SD AND LSD WITH MINIMIZED SECRET STORAGE REQUIREMENTS

This section points out SD and LSD based key management algorithms which include the proposed Mapping \( A \).

Assuming a broadcast encryption system with \( N \) receivers, the reported SD [8] and basic LSD [2] key management schemes require the secret storage overheads \( O((\log_2 N)^2) \) and \( O((\log_2 N)^{3/2}) \), respectively. A goal of this section is to propose modified SD and LSD based key management schemes with the secret storage overhead of only \( O(1) \) (independent of the system parameter \( N \)), public storage overheads \( O((\log_2 N)^2) \) and \( O((\log_2 N)^{3/2}) \), respectively, and a lower...
additional processing overhead, employing Mapping A over the master secret key bits.

For further considerations we assume that employment of the SD and LSD schemes requires that a receiver stores in secret storage a sequence of vectors \( K_i \), \( i = 1, 2, ..., I \), where for simplicity of the notations, the superscripts related to SD and LSD are omitted.

Let us consider the key management over \( N \) users. Assuming that \( I^{(SD)} \) and \( I^{(LSD)} \) denote values of \( I \) related to SD and basic LSD, respectively, they are given by the following (see [8] and [2]):

\[
I^{(SD)} = \frac{1}{2} \left[ (\log_2 N)^2 + \log_2 N \right] + 1 , \quad (2)
\]

\[
I^{(LSD)} = (\log_2 N)^{3/2} + 1 . \quad (3)
\]

This section proposes modified versions of SD and LSD as follows:
- In the modified SD each receiver keeps in secret storage the seed \( S \) in the form of \( 3n \)-dimensional binary vector, and employing the Mapping A evaluates any of the required \( I^{(SD)} \) vectors; all other issues are identical to the original and modified SD;
- In the modified LSD each receiver keeps in secret storage the seed \( S \) in the form of \( 3n \)-dimensional binary vector, and employing Mapping A evaluates any of the required \( I^{(LSD)} \) vectors; all other issues are identical to the original and modified LSD.

Recall that employment of Mapping A requires that certain information be stored in public storage and that certain processing be employed. Mapping A directly implies that the required public storage is proportional to \( I \).

The processing overhead of the modified SD/LSD is proportional to two additional executions of the pseudorandom generator.

V. COMPARISON

Tables I and II yield a summary comparison of the main overheads regarding the modified SD/LSD, the original ones and the ones with an additional encryption employing a simple master key technique, respectively.

The main characteristics of the virtual storage approach based on encryption (employing the master secret key) are the following:
- employed public storage contains encrypted form of the data which should be stored in a secret storage;
- security of the virtual storage depends on the employed encryption technique, and usually it is heuristically secure only;
- processing complexity of the virtual storage implementation depends on the implementation complexity of the involved encryption.

On the other hand the main characteristics of the proposed approach are as follows:
- public storage contains only non-secret data required for the employed Mapping A;
- Mapping A is provably secure (see Appendix A and [7]);
- implementation complexity of Mapping A is very low.

Table III compares the proposed secret storage minimization and the minimization based on a virtual secret storage established employing a master key encryption.

VI. CONCLUDING DISCUSSION

In comparison with the straightforward minimization of the required secret storage employing a virtual secret storage established via the data encryption, the considered novel approach has the following advantages: (i) it yields provable security; (ii) it is not based on storing the encrypted data into a public storage, and (iii) it requires lower processing complexity (see Table III).
The proposed method and key management schemes employ only secret-key cryptography and yields a reduced secret storage requirement. The achieved trade-offs are related to reducing the secret storage overhead at the expense of public storage and certain processing, but without involvement of complex operations or public key cryptography.

The proposed modification of the original SD and LSD schemes does not change the nature of theirs security: The modified schemes belong to the same class of the computational secure key management schemes as the original ones, and they have the same level of security. The proposal is compared with the originally reported ones, and advantages of the novel approach are shown. Accordingly, the proposed schemes yield an appropriate trade-off between the main system overheads in a number of scenarios.

Also note the following. Recently, in [5], with an additional refinement in [6], the reconfigurable key management is proposed as an advanced technique for broadcast encryption which is appropriate for a high dynamic of the legitimate users, and which yields minimization of the secret information volume to be stored at the terminals. The design is based on the appropriate mapping of the same secret bits so that the mapping results can play different roles. Accordingly, the proposed key management implementation method is also relevant for the reconfigurable key management.

REFERENCES


VII. APPENDIX A:
MAIN CHARACTERISTICS OF MAPPING A

This section overviews the main characteristics of Mapping A according [7].

A. Preliminaries and Underlying Idea

Assume that $X = \{x_i\}$ is a purely random binary sequence, that is, a sequence of balanced i.i.d. binary random variables. Also, assume that a random decimation sequence $D = \{d_i\}$ is a sequence of i.i.d. non-negative integer random variables that is independent of $X$. Let $P = \{P(d_i)\}$ denote the probability distribution of $d_i$, for any $i \geq 1$, where $D$ is the set of values with nonzero probability.

Let the random sequences $X$ and $D$ be combined by the following decimation equation $y_i = x_{i+j}$, $j = \sum_{i=1}^{i} d_i$, yielding the output random sequence $Y = \{y_i\}$. Accordingly $Y$ is a purely random binary sequence itself.

It is possible to define the joint probability distribution $P(X,Y)$ for all pairs of binary strings $X = \{x_i\}_{i=1}^{n}$ and $Y = \{y_i\}_{i=1}^{n}$, for any $m \geq n$, which is a basis for the statistically optimal recovering of $X$ given $Y$ employing the maximum posterior probability decision rule.

Given a set of non-negative integers $D$, we assume that a binary string $Y = \{y_i\}_{i=1}^{n}$, of length $n$ can be $D$-embedded into a binary string $X = \{x_i\}_{i=1}^{n}$ of length $m$ if there exists a non-negative integer string $D = \{d_i\}_{i=1}^{n}$ of length $n$ such that $d_i \in D$ and consistent with $\{y_i\}_{i=1}^{n}$.

To check whether $Y$ can be $D$-embedded into $X$, the direct matching algorithm can be employed.

Let $P_{D,Y}(n,m)$ be defined as the probability that a binary string $Y$ of length $n$ can be $D$-embedded into a purely random binary string $X$ of length $m$.

Here, we consider the unconstrained embedding case where $D$ is the set of positive integers, i.e. $D = Z^+$. For simplicity, denote the unconstrained embedding probability $P_{Z^+,Y}(n,m)$ by $P_Y(n,m)$.

Theorem 1. [1]. For an arbitrary binary string $Y$ of length $n$, the unconstrained embedding probability is given by

$$P_Y(n,m) = 1 - 2^{-m} \sum_{j=0}^{n-1} \binom{m}{j}. \quad (4)$$

Underlying Idea for the Mapping.

Theorem 1 indicates that with a probability close to one, any given length-$n$ binary sequence can be embedded into an arbitrary $m$-length binary sequence, assuming that the difference $\Delta = m - n$ is large enough. Accordingly, the infeasibility
of recovering the initial sequence from its decimated version opens a door for designing the desired one-way mapping of the seed into a desired KEK.

B. Main Characteristics

Existence of the Mapping

Proposition 1. Mapping A provides the mapping of an arbitrary binary \( k \)-dimensional vector \( S \) into a given binary \( n \)-dimensional vector \( K_i \).

One-Wayness

Proposition 2. When \( k > 2n \), the complexity of recovering any \( K_i \) is proportional to \( 2^n \) given all other vectors \( K_j, j \neq i \), \( i = 1, 2, \ldots, I \), and all public information.

Proposition 2 and Theorem 1 directly imply the following proposition.

Proposition 3. When \( k > 2n + \log \alpha \), the complexity of recovering \( S \) is greater than the complexity of recovering \( \alpha \) different \( K_i \)’s.

Implementation Complexity

Proposition 4. Assuming that the complexity to generate a length-\( \ell \) output sequence from the pseudorandom sequence generator \( G \) is \( O(\ell) \), the implementation complexity of Mapping A is \( O(m + nq) \).

VIII. APPENDIX B: BACKGROUND ON SD AND LSD KEY MANAGEMENT SCHEMES

Recent papers [8] and [2] have addressed the BE scenario with the stateless receivers. The basic idea in all the stateless encryption schemes is to represent any privileged set 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 the program key (SEK) \( s \) times under all the keys (KEKs) associated with the set in the cover. Consequently, each privileged user can easily access the program, but even a coalition of the non-privileged users cannot find the program key. The simplest implementation of this idea is to cover the privileged set with singleton sets. A better solution is to associate the users with the leaves of a binary tree, and to cover the privileged set of leaves with a collection of subtrees.

In [8], 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 (long-lived) keys to subsets in the collection. An important consequence of this framework is the separation between long-lived keys and short-term keys. Two types of revocation schemes in the subset-cover framework, are proposed in [8] 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 \( R \log N \) and storage of \( \log N \) keys at the receiver. The second technique for the covering is the Subset Difference (SD), [8]. 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 descendant 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 \( O((\log N)^2) \) keys.

What is shown in [2] is that SD collection of sets can be reduced: The basic idea of the Layered Subset Difference (LSD) scheme [2] 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 \( k = 1, \ldots, \sqrt{\log(N)} \), 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 [2], 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:

\[
S_{i,j} = S_{i,k} \cup S_{k,j}.
\]

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.