Tatra Mt. Math. Publ. **33** (2006), 1–14
Mathematical Publications

EFFICIENT USER REMOVAL IN BROADCAST CHANNEL WITH SYMMETRIC ENCRYPTION

Jarosław Byrka

ABSTRACT. We present a broadcast protocol that enables to remove some receivers by updating a key used for encrypting transmission. The scheme is based on a set of keys properly distributed among users. Our protocol requires relatively small transfer for updating the set of authorized users. In a system with n users, our protocol requires $2 \cdot c \cdot \log_c n$ keys per user. To exclude k users it needs to broadcast $2 \cdot \log_c n \cdot k$ symmetric ciphertext blocks.

Our construction outperforms simple tree-like key designs and is a good alternative for polynomial-based protocols (like the one of Watanabe et al.) using asymmetric cryptography.

Known polynomial-based protocols require a broadcast of length $O(k)$ for removing k users. We get the same message size by setting $c = n^{1/d}$ for some constant d. It would result in giving $2 \cdot n^{1/d} \cdot d$ keys to each user. If users are able to store so many keys, then our protocol appears to be useful since it requires much less computation time and is better if we want to update the set of users quite often.

1. Introduction

We consider a situation where lots of data is broadcasted to a group of users (like in a pay-per-view TV). To make the service available only for the registered users, the whole transmission must be encrypted. One of the main problems is how to change the set of users who may decrypt the broadcasted data. This paper concentrates on the problem of removing groups of unwanted users.

In general, it is likely that other actors (not users) obtain the broadcasted information. However, the transmitted data will be encrypted and non-users will not be able to decode it (in fact, we will make our system resistant to attacks of groups of cooperating users, which is a stronger property). Therefore, we will consider the broadcaster and the registered users as the only actors of the described scenario.

2000 Mathematics Subject Classification: 94A60, 68M12, 68P25, 68R05.
Keywords: broadcast encryption, broadcast exclusion, symmetric cryptography.

The paper contains results of the author's Master's Thesis at the University of Wrocław.

We assume that there exists a reliable symmetric key encryption method. We will use such a method to encrypt broadcasted data. The problem we concentrate on is to deliver a new symmetric key (called a broadcast key) to users who are entitled to decrypt the broadcasted data without revealing it to the others. We provide a solution that is based on a proper assignment of symmetric keys.

1.1 Problem description

In our model there is a broadcast channel, a broadcaster and many receivers (called users). We assume that broadcaster has unlimited storage and is capable to perform complicated computation. Typically, receivers are small computing units with limited storage. The users are given unique identifiers (e.g., numbers from 0 to $n-1$). When a user registers in the system, he may be given appropriate keys (for the later user management procedures). Further communication with the user is limited to his listening to what is broadcasted in the public broadcast channel.

Users and user slots. In the broadcast system described, there is a constant set of user slots. To register an actor as a new user of the system, the broadcaster uses some free user slot. When an actor becomes a new user, he receives secret information of his user slot. To enable such "user addition" a certain amount of free user slots is necessary in the system. A possible solution is to make the number of user slots in the system twice bigger than it is needed when the system is built. When there are no free slots left, a reconstruction of the system is necessary to "introduce a new user".

Inactivating users instead of removing them. Since we decided that the set of user slots is constant, we need to simulate the operation of "removing a user". Each user in our system is in one of the two possible states (*active* or *inactive*). Active users are those who have the current broadcast key and may decrypt broadcasted data. At the beginning, all users are active (authorized). To "remove a user" (disable him to decrypt the transmission) the system makes him inactive. To change the set of active (authorized) users, the broadcaster generates a new broadcast key and distributes it only to users who should be active. Some protocols allow reactivating a user without changing the broadcast key (by "giving him" the current broadcast key).

System works in phases. There are two alternating types of system activity: *"regular broadcast" phase* and *"users management" phase.* In this paper we concentrate on the second type of system's activity. Each "regular broadcast" phase uses some broadcast key for encrypting the transmitted data. When describing a "users management" phase, we will sometimes refer to a broadcast key of some past "regular broadcast" phase (denoted by K_{old}) and the key of the next "regular broadcast" phase (denoted by K_{new}).

Using old broadcast keys. Generally, an old broadcast key K_{old} may be used while distributing K_{new}. Let A_{old} denote the set of users that received K_{old} and A_{new} denotes the set of users that should receive K_{new}. It is usually useful to think of distributing K_{new} as of taking some old set of authorized users A_{old}, removing some set of users R from A_{old}, and possibly adding (reactivating) some users to result in a set A_{new}. The cost of distributing K_{new} depends mainly on the number of users being removed. The usage of K_{old} may reduce the transfer volume, because it enables an easy elimination of users out of A_{old}. In order to take the full advantage of old broadcast keys when distributing K_{new}, users should store some of these keys. Nevertheless, the possible policies of storing old broadcast keys are not the part of our interest and are not discussed in this paper.

Important parameters. The problem we will concentrate on is how to remove k out of n users in the system. The purpose is to minimize both the transmission length a protocol requires when removing users, and the number of keys one user has to store. We are mainly interested in the asymptotic behavior of these parameters. However, when real live applications are concerned, a solution should behave well for $n \approx 10^6$ and $k \ll n$. When designing our protocol we aimed to make it applicable for such values of these parameters.

1.2 Possible solutions

In this section we describe several user removal methods. These, based on symmetric encryption, seem to fit into the following framework.

Framework for symmetric protocols. In order to remove (inactivate) some set R of users the broadcaster:

1. generates a new broadcast key K_{new};
2. chooses a subset F_R of the distributed keys that will be used to provide K_{new} to authorized users;
3. for each $f_i \in F_R$ prepares a message $M_i = E_{f_i}(K_{\mathrm{new}})$ by encrypting K_{new} with a symmetric key f_i;
4. broadcasts messages M_i.

After such an operation each user attempts to decrypt each of the messages M_i with each of the symmetric keys he was given. If the number of messages M_i and the number of keys each user stores are both quite big, it would be reasonable to tell the user which key was used to encrypt M_i.[1]

[1] We may easily produce appropriate key's identifiers with a hash function, and attach to each of messages M_i an identifier I_i. A user that have received a pair (M_i, I_i) knows that he should use a key f_i to decrypt M_i.

Usage of this framework results in providing K_{new} to all users that know at least one of the keys $f_i \in F_R$.

Ad hoc solution. The simplest idea is to give each user his individual (symmetric) key. When updating the broadcast key, a new key K_{new} is encrypted individually for each authorized user with his key and broadcasted. This requires transmission of length $O(n)$ for every update.

If each user keeps a single key shared with the broadcaster, it is impossible to reduce significantly the transmission volume comparing to the ad hoc solution (see [1]).

Tree-like solutions. There is a group of protocols corresponding to tree-like key distributions. Imagine a binary tree with users in its leaves and symmetric keys in all its nodes. Each user knows $\log n$ keys, namely the keys on the path from his leaf to the root of the tree.

When removing users we use the framework 1.2. To remove a set R of k users the set of keys F_R is chosen as follows: A leaf of the tree is called *dirty* iff a user u corresponding to this leaf is being removed ($u \in R$). A node v is dirty iff the subtree rooted at v contains a dirty leaf. F_R is the set of such nodes u where u is not dirty, but u has a dirty father in the tree.

PROPOSITION 1. *Using the above described method it is possible to remove k users by broadcasting $O(k \cdot \log_2 n)$ ciphertexts.*

Proof. When k users are being removed, at most $k \cdot \log_2 n$ nodes of the tree are dirty. Each dirty node has at most one child that is not dirty. Hence, there are at most $k \cdot \log_2 n$ nodes in F_R. □

Faster removal of a single user. It is possible to modify the above described schema by taking a tree of higher degree. Again, there is a key in each node of the tree, but the distribution of these keys to the users is different.

Let P_l denote a path from leaf l to the root of the tree. The user in a leaf l knows the key in node v iff v is a son of a node u, $u \in P_l$, and $v \notin P_l$. User removal is described according to the common framework 1.2. The set F_R of keys to be used is chosen as follows: To remove a user in a leaf l the broadcaster uses keys from vertices $v \in P_l$.

Such a protocol needs a broadcast of $O(\log_d n)$ ciphertexts to remove a single user, and $O(d \cdot \log_d n)$ keys per user, where d is a degree of the tree. This method requires lower transmission volume than the one with a binary tree. It is, however, unacceptable for real applications, because it is impossible to remove a pair of cooperating users.

Polynomial based solutions. Another group of solutions is based on polynomial interpolation and asymmetric cryptography. These protocols use the fact that one cannot reconstruct a polynomial of degree k from values of this polynomial in k points, while $k+1$ values suffice.

In the paper [2] A n z a i *et al.* presented a protocol that requires a broadcast of length $O(k)$ for removing k users. In their method, each user has a key which size is independent from n and k. Unfortunately, in this solution each user must perform $O(k)$ modular exponentiations when decrypting a new broadcast key.

Protocol from [3] requires only $O(1)$ modular exponentiations per update, but it is insecure when used repeatedly (see [4]). In [4] another protocol was proposed by W a t a n a b e *et al.* It allows a multi-round usage and requires $O(k)$ modular exponentiations, but most of the computation might be performed in a preprocessing phase.

Random key distributions. A broadcast encryption method based on randomly distributed hashed symmetric keys was recently proposed by R a m k u m a r [5]. Such a solution, for the price of storing more keys per user, provides a low cost removal of users. Moreover, it enables to implement the system that gives each user the possibility to broadcast his signal to any chosen subset of users.

1.3 Our contribution

We propose a protocol that is a generalization of tree-like key distribution protocols. It enables removing an arbitrary number of users at once (like in a simple binary tree) which is better than in [4], where at most k_0 users could be removed (k_0 is a protocol parameter).

In our solution removing k users requires broadcasting $2 \cdot \log_c n \cdot k$ symmetric ciphertext blocks. In the construction we propose each user must store $2 \cdot c \cdot \log_c n$ symmetric keys. These parameters are almost like for a tree of degree c, and removing the users one by one. Our contribution might be interpreted as a modification of a tree method to enable a removal of any subset of users at once.

Our method performs well in practice. For 10^6 users we might choose $c = 16$ and have 160 keys per user. Broadcasting $10 \cdot k$ symmetric ciphertext blocks would then suffice to remove k users.

2. New user removal protocol

Here we present our protocol for removing unwanted users. To describe it we use the framework 1.2. The core of our construction is a proper distribution of symmetric keys that will be used for the removal procedure.

Let $U = \{0 \ldots n-1\}$ be the set of identifiers of users in the broadcast system. We construct a family S of subsets of U. Each subset $s \in S$ corresponds to a key k_s that is known exactly to the users with identifiers from s and to the broadcaster. All logs in this section are to base 2, unless stated otherwise.

2.1 Construction of family $S(c,n)$

The parameters are: a constant c and the number of users(user slots) n. We put all user identifiers on a circle and take subsets of identifiers corresponding to arcs of this circle.

For $0 \leqslant x < y < 2n$ let $\langle x, y\rangle$ be defined as

$$\langle x,y\rangle = \begin{cases} \{a \in U\colon x \leqslant a < y\}\,, & y < n\,;\\ \{a \in U\colon x \leqslant a \vee a < y-n\}\,, & \text{otherwise}\,. \end{cases}$$

Definition 2.1. A *layer* $L(l,s)$ is defined as:

$$L(l,s) = \left\{\langle x,y\rangle\colon\, x = s\cdot i \bmod n \wedge y = x + l \bmod n \wedge i \in \left\{0,1,\dots,\left\lceil \frac{n}{s}\right\rceil - 1\right\}\right\}.$$

Intuitively, it is a family of intervals with length l and shift s (see Figure 1).

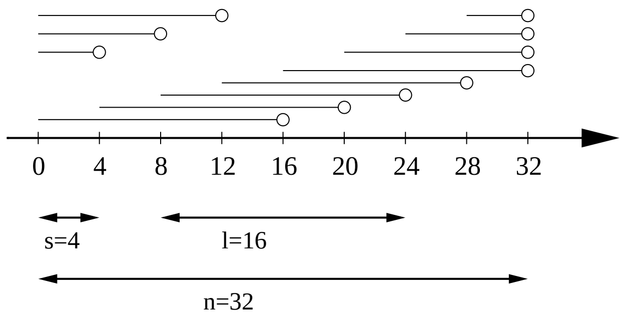

Figure 1. Example of a layer: $L(16,4)$ for $n = 32$.

Definition 2.2. A *level* $\mathrm{Lv}(l,s)$ is a sum of layers with shift s and lengths $s, 2s, \dots, 2^{\lceil \log(l/s)\rceil}\cdot s$. Formally:

$$\mathrm{Lv}(l,s) = \bigcup_{i\in\{0,1,\dots,\lceil \log(l/s)\rceil\}} L(2^i\cdot s, s)$$

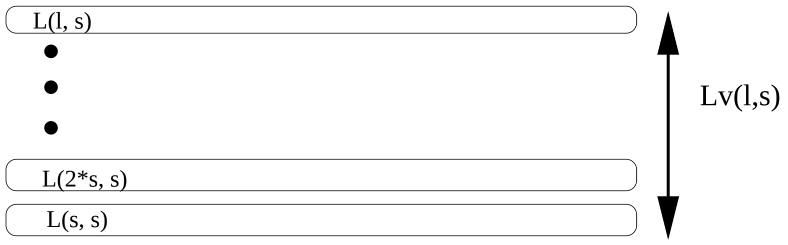

Figure 2. Composition of a level from layers.

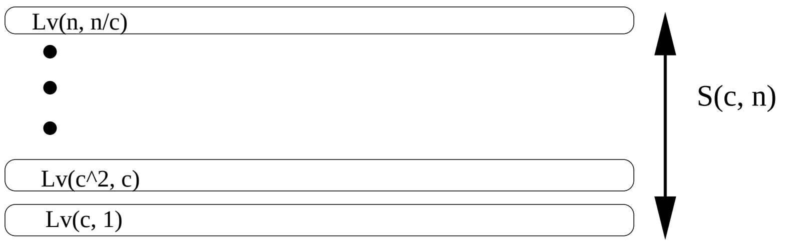

FIGURE 3. Composition of the family $S(c,n)$ from levels.

DEFINITION 2.3. Family $S(c,n)$ is composed as a sum of Levels;

$$S(c,n) = \bigcup_{i\in\{0,1,\ldots,\lceil\log_c n\rceil-1\}} \mathrm{Lv}(c^{i+1},c^i)\,.$$

2.2 Main properties

Let us first estimate the number of keys that a user must keep:

LEMMA 2.1. *Each user identifier is a member of at most* $(4c+\lceil\log c\rceil)\cdot\lceil\log_c n\rceil$ *sets in* $S(c,n)$*, and if* c *is a power of* 2 *and* n *is a power of* c *each user identifier is a member of at most* $2\cdot c\cdot\log_c n$ *sets in* $S(c,n)$*.*

Proof. It is an easy observation that if s divides n, then in any *layer* $L(l,s)$ there are at most $\lceil\frac{l}{s}\rceil$ sets containing any user identifier. In case n is not a multiply of s, some of the first l identifiers may occur $\lceil\frac{l}{s}\rceil+1$ times.

We now concentrate on *levels*. In a *level* $\mathrm{Lv}(l,s)$ there are $\lceil\log l/s\rceil+1$ layers. We first deal with the case when: $l=s\cdot 2^r$ for some integer r (it happens when c is a power of 2) and l divides n (this is true when n is a power of c). If these conditions hold, we may count occurrences of any identifier in sets from a level $\mathrm{Lv}(l,s)$ as follows: it occurs $\frac{l}{s}$ times in a layer $L(l,s)$, $\frac{l}{2s}$ in a layer $L(l/2,s)$, $\frac{l}{4s}$ in $L(l/4,s)$, ... , 1 in $L(s,s)$. Summing up, there are $2\cdot\frac{l}{s}-1$ sets containing identifier i in a level $\mathrm{Lv}(l,s)$ for any identifier $i\in U$.

If we drop the assumptions on the parameters l, s and n, we might get an extra layer that would at most double the number of identifier occurrences, and the number of occurrences in each layer may be increased by 1. It gives at most $4\cdot\frac{l}{s}+\lceil\log l/s\rceil$ occurrences in a level $\mathrm{Lv}(l,s)$.

Finally we get to the family $S(c,n)$. It is constructed as a sum of $\lceil\log_c n\rceil$ levels $\mathrm{Lv}(c^i,c^{i+1})$. If c is a power of 2 and n is a power of c, there are $2c-1$ occurrences in each level, and therefore less then $2\cdot c\cdot\log_c n$ occurrences of an identifier in the whole family $S(c,n)$.

Generally, there are at most $4c+\lceil\log c\rceil$ occurrences in each level, and therefore at most $(4c+\lceil\log c\rceil)\cdot\lceil\log_c n\rceil$ occurrences of an identifier in the whole family $S(c,n)$.

In particular, without the assumption that n is a power on c we get an additional level; without the assumption that c is a power of 2 we get an additional layer for each level and for some users an additional occurrence per layer. □

The next lemma indicates that any set of users can be quickly removed.

Lemma 2.2. *Let $R \subset U$ and $|R| \leqslant k$. Then there is a set $S_R \subset S(c, n)$ such that $(U \setminus R) = \bigcup_{s \in S_R} s$ and $|S_R| \leqslant 2 \cdot \lceil \log_c n \rceil \cdot k$.*

A proof of this lemma will be given at the end of the next subsection.

In order to remove a set of unwanted users R we use the family S_R from Lemma 2.2. Keys corresponding to sets in S_R are used for the removal procedure as described in framework 1.2. For each $s \in S_R$ the new broadcast key is encrypted with the key k_s. The ciphertexts obtained this way are sent through the broadcast channel. Each authorized user i $(i \notin R)$ finds $s \in S_R$ such that $i \in s$, and retrieves the new broadcast key from the ciphertext using k_s.

2.3 Choice of keys—construction of S_R

To prove Lemma 2.2 we give an algorithm constructing a set S_R. One step of the algorithm is a choice of a set from $S(c, n)$. Let $A = U \setminus R$ be the set of authorized users, and let S_{curr} be the family of already chosen sets.

Let

$$C = \bigcup_{s \in S_{\mathrm{curr}}} s$$

be the set of users that would receive a new broadcast key K if we send it to all users from sets within family S_{curr}. Initially both S_{curr} and C are empty. During an execution of the algorithm $C \subset A$ (i.e., the set C never contains any removed user).

To explain the usage of sets from the family $S(c, n)$ we name some characteristic configurations of elements from R and C. When describing these configurations we need to have users ordered in a circle like in 2.1.

We say a is the right neighbor of b and b is the left neighbor of a iff $a = b + 1 \mod n$.

Definition 2.4. We say that users $u_1, u_2, \ldots, u_h$ form a *hole* iff all the following conditions hold:

- u_{i+1} is the right neighbor of u_i for $i = 1, 2, \ldots, h - 1$;
- $u_i \in (A \setminus C)$ for $i = 1, 2, \ldots, h$;
- the left neighbor of u_1 is in R;
- the right neighbor of u_h is in R.

At the beginning, the set A may be interpreted as a collection of holes. Each hole is treated independently, thus the whole algorithm is nothing more than filling in one hole after another. We need one more notation to analyze this filling in process.

Definition 2.5. We say that users $u_1, u_2, \ldots, u_h$ form a *left side hole* of size h with buffer size b iff there are users $u_{h+1}, u_{h+2}, \ldots, u_{h+b}$ such that all the following conditions hold:

- u_{i+1} is a right neighbor of u_i for $i = 1, 2, \ldots, h+b-1$;
- $u_i \in (A \setminus C)$ for $i = 1, 2, \ldots, h$;
- left neighbor of u_1 is in R;
- $u_i \in C$ for $i = h+1, h+2, \ldots, h+b$.

In other words, a *left side hole* is a segment of h users that must get the new key K; it has a user which is being removed at the left end, and another b users that have already received K at the right end. The definition of a *right side hole* is symmetric.

Definition 2.6. We say a *left* (*right*) *side hole* fits into level i iff it's size h is at most c^i and it's buffer size b is at least c^i.

Lemma 2.3. *Given a left (right) side hole $u_1, u_2, \ldots, u_h$ that fits into level $i > 1$, there is a set $s_i \in S(c,n)$ such that if we add s_i to S_{curr} the remaining left side hole $u_1, u_2, \ldots, u_{h'}$ (right side hole $u_{h'}, u_{h'+1}, \ldots, u_h$) will fit into level $i-1$.*

Proof. We choose the leftmost (rightmost) set from the layer $L(c^i, c^{i-1})$ that does not contain the left neighbor of u_1 (right neighbor of u_h). (See Figure 4.) □

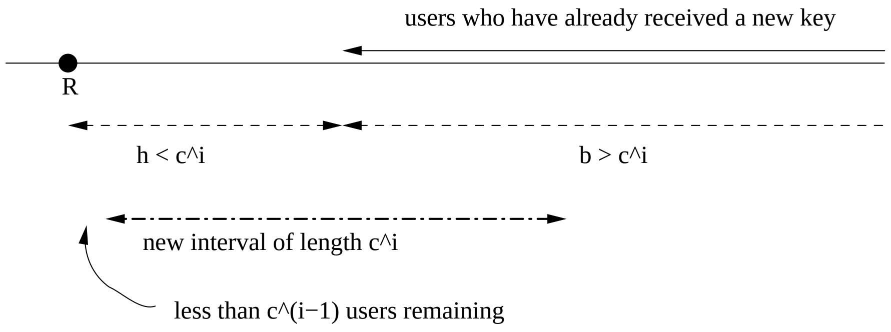
users who have already received a new key
R
h < c^i
b > c^i
new interval of length c^i
less than c^(i−1) users remaining

Figure 4. Filling a left side hole.

Lemma 2.4. *x sets from the family $S(c,n)$ suffice to fill in a left (right) side hole that fits into level x.*

Proof. It is a simple induction on x.

If $x=1$, the *left (right) side hole* size is $h \leqslant c$. Recall that $S(c,n)$ contains $\mathrm{Lv}(c,1)$. In turn, $\mathrm{Lv}(c,1)$ contains intervals that start at arbitrary locations, and have lengths which are powers of 2, not exceeding c.

If $x>1$, we use Lemma 2.3 as an induction step. □

Now we are ready to prove Lemma 2.2.

Proof of Lemma 2.2.

Since there are k users to be removed, the set of authorized users $A = U \setminus R$ is a sum of at most k holes. We fill each hole using at most $2 \cdot \lceil \log_c n \rceil$ sets from $S(c,n)$.

For a given hole $H = \{u_1, u_2, \dots u_h\}$ we choose a layer $L \subset S(c,n)$ which contains an interval of the highest length among intervals from $S(c,n)$ that are subsets of H. We choose the leftmost interval s_l and the rightmost interval s_r from L. (See Figure 5.)

Observe that $|s_l \cap s_r| > 0$. Otherwise there would be $s_{\text{sup}} \in S(c,n)$ such that $|s_{\text{sup}}| > |s_l|$, and $s_{\text{sup}} \subset H$, and it contradicts our choice of layer L.

Let $\mathrm{Lv}(l,s)$ be the level in $S(c,n)$ containing layer L. Let i be an integer such that $s = c^i$. Since we have chosen the leftmost and the rightmost interval from L that are inside H, the remaining users in this hole form one left and one right side hole that fit into level i ($i \leqslant \lceil \log_c n \rceil - 1$). From Lemma 2.4 it suffices to use i sets from $S(c,n)$ to fill each of them. □

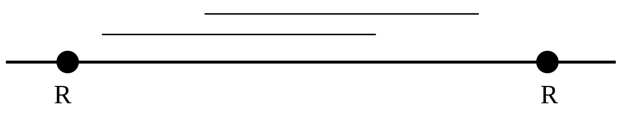

Figure 5. To fill in a hole, we start with the leftmost and the rightmost interval from longest intervals contained in the hole.

3. Alternatives for $S(c,n)$

In this section we discuss a possibility to find a family F of subsets of U better for building a user removal protocol. Each subset $s \in F$ corresponds to a key k_s that is known exactly to the users form s and to the broadcaster. To remove a set R of users from the system, we need to 'construct' a set $A = U \setminus R$ of authorized users from sets $s \in F$.

This construction may be a simple sum like in framework 1.2 or a more complicated composition. For example, we may use intersection of two subsets by sending one part of a secret information encrypted with the first key, and the other part encrypted with the second key. If decoding further transmission requires both parts of the secret, exactly the users from the intersection of sets corresponding to the used keys will have sufficient information to remain active.

There is a serious disadvantage of any construction using intersections. If we would like to remove two users and they together have both parts of the secret, they may exchange information and remain active. It seems to be an interesting question, whether using intersections could reduce the transmission volume significantly. This paper, however, concentrates on the constructions based on sums, as in framework 1.2.

3.1 Strongly selective families and their application to user removal problem

Definition 3.1. Let $k \leqslant n$. A family $\mathcal{F}$ of subsets of $\{0, 1, \ldots, n-1\}$ is (n, k)-*strongly-selective* if for every nonempty subset Z of $\{0, 1, \ldots, n-1\}$ such that $|Z| \leqslant k$ and for every element $z \in Z$ there is a set F in $\mathcal{F}$ such that $Z \cap F = \{z\}$.

Strongly selective families (SSF) (also called *superimposed codes*) are well described in the literature, see [7–9].

Consider a broadcast system where symmetric keys are distributed according to some family $\mathcal{F}$ of subsets of user identifiers. The following lemma gives a necessary and sufficient condition for the family $\mathcal{F}$ to allow a removal (in the sense of framework 1.2) of any set of at most k users in this system.

Lemma 3.1. *Let $U = \{0, 1, \ldots, n-1\}$, $\mathcal{F} \subset \mathcal{P}(U)$ and $k < n$. Then (for any $R \subset U$, such that $|R| \leqslant k$, there exists a set $\mathcal{F}_R \subset \mathcal{F}$ such that $U \setminus R = \bigcup_{s \in \mathcal{F}_R} s$) if and only if (the family $\mathcal{F}$ is a $(n, k+1)$-strongly-selective family).*

P r o o f. The "if"part:

Let $\mathcal{F}$ be a $(n, k+1)$-strongly-selective family. Let R be any subset of U with at most k elements. We will show that for any element $x \in U \setminus R$ there exists a set $F_x \in \mathcal{F}$ such that $|F_x \cap R| = 0$ and $x \in F_x$. Let us take $Z = R \cup \{x\}$, and from the definition of (n, k)-strongly-selective family there exists a set $F_x \in \mathcal{F}$ such that $F_x \cap Z = \{x\}$. Then the family $\mathcal{F}_R = \bigcup_{x \in U \setminus R} F_x$ has the desired property where $U \setminus R = \bigcup_{F \in \mathcal{F}_R} F$.

The "only if" part:

If $\mathcal{F}$ is not a $(n, k+1)$-strongly-selective family, there exists a set Z such that $|Z| \leqslant k+1$ and there is an element $x \in Z$ such that for any $F \in \mathcal{F}$ either

$\left|(Z \setminus \{x\}) \cap F\right| > 0$ or $x \notin F$. Let $R = Z \setminus \{x\}$. We show that there is no family $\mathcal{F}_R \subset \mathcal{F}$ such that $U \setminus R = \bigcup_{F \in \mathcal{F}_R} F$. Suppose there exists such family $\mathcal{F}_R$. There are two cases possible:

$$\forall_{F \in \mathcal{F}_R}\ x \notin F, \text{ then } x \in (U \setminus R) \wedge x \notin \left(\bigcup_{F \in \mathcal{F}_R} F\right);$$

$$\exists_{F \in \mathcal{F}_R}\ x \in F, \text{ then } \exists_{y \in (R \cap F)}\ y \in \left(\bigcup_{F \in \mathcal{F}_R} F\right) \wedge y \notin (U \setminus R).$$

Each of the cases results in a contradiction which completes the proof. □

3.2 Known bounds on the size of SSF

There are some bounds known for the minimal number of sets (n,k)-strongly--selective family must contain.

The following theorem gives a nonconstructive upper bound.

Theorem 1. *For any $n \geqslant 3$ and $k \geqslant 2$ there exist (n,k)-strongly-selective families of size $O(k^2 \log n)$.*

The proof of this theorem is based on a probabilistic argument. The probability that a random family is a strongly selective family is too small to give a simple randomized algorithm for constructing such families.

The following theorem indicates that smaller strongly selective families do not exist.

Theorem 2. *Let $\mathcal{F}$ be a (n,k)-strongly-selective family.*

1. *If $2 \leqslant k \leqslant \sqrt{2n} - 1$, then $|\mathcal{F}| \geqslant \frac{k^2}{16 \log k} \log n$.*
2. *If $k \geqslant \sqrt{2n}$, then $|\mathcal{F}| \geqslant n$.*

In the context of the problem of removing users the size of the family corresponds to the number of keys in the whole system. For it is only the broadcaster which stores all these keys, and we assumed it has a huge computational power and unlimited storage, we do not intend to optimize the total number of subsets in the family.

The important parameters are: the number of keys per user, and the transmission length needed for removing k users. Trying to minimize only one of them results in extremal solutions. When minimizing only the number of keys per user, we may take the family of all singletons with elements from U (give each user his own key). This is the ad hoc solution mentioned in Section 1. If we concentrated only on the transmission length, we would take the family of all subsets of U, and would enable the session key update with a transfer of a constant length. This family, however, cannot be used due to the exponential in n number of keys each user must store. When thinking about systems with number of users $n \geqslant 10^6$, we may not afford having more than $O(n)$ keys. In case receivers are small computing units (e.g., chip cards), the number of keys they may store is much less then a thousand.

We consider our construction of family $S(c,n)$ as a compromise between minimizing the number of keys and the transmission length.

4. Conclusion

This paper presents a protocol for efficient removal of users from an encrypted broadcast channel. Our protocol uses symmetric cryptography, and among symmetric protocols it requires the shortest transfer to remove groups of cooperating users. Compared to asymmetric cryptography protocols like in [4], it requires less computational power of receivers, and allows frequent updates of the authorized users set.

Another important issue is the hardware support of encryption and decryption especially at the receivers side. Nowadays, there is much more hardware support for symmetric cryptography protocols, which makes these protocols more practical and realizable for small computing units.

An intriguing problem that remains unresolved is finding a nontrivial lower bound on the transmission length, depending on the number of keys one user can store. Another interesting question is whether there is a more effective protocol that, for any set R of users we would like to exclude, enables an exclusion of set R' of users which differs from R very little (e.g., $|R \setminus R'| < c$ for some small constant c and $R' \subset R$).

Acknowledgments. I would like to thank Mirosław Kutyłowski and Krzysztof Loryś for valuable suggestions, comments and encouragement to write this paper. Moreover I want to thank Maciej Modelski, my wife Katarzyna and the referee for making this text more readable.

REFERENCES

[1] CICHOŃ, J.—KRZYWIECKI, Ł.—KUTYŁOWSKI, M.—WLAŹ, P.: *Anonymous distribution of encryption keys in cellular broadcast systems*, in: MADNES '05, Lecture Notes in Comput. Sci., Vol., Springer-Verlag, Berlin (to appear), in `http://im.pwr.wroc.pl./~cichon/jcipubl.htm`.

[2] ANZAI, J.—MASUZAKI, N.—MATSUMOTO, T.: *A quick group key distribution schema with "entity revocation"*, in: ASIACRYPT '99, Lecture Notes in Comput. Sci., Vol. 1716, Springer-Verlag, Berlin, 1999, pp. 333–347.

[3] ANZAI, J.—MASUZAKI, N.—MATSUMOTO, T.: *Light weight broadcast exclusion using secret sharing*, in: ACISP '00, Lecture Notes in Comput. Sci., Vol. 1841, Springer-Verlag, Berlin, 2000, pp. 313–327.

[4] WATANABE, Y.—NUMAO, M.: *Moulti-round secure light-weight broadcast exclusion protocol with pre-processing*, in: ESORICS '03, Lecture Notes in Comput. Sci., Vol. 2808, Springer-Verlag, Berlin, 2003, pp. 85–99.

[5] RAMKUMAR, M.: *Broadcast encryption with random key pre-distribution schemes*, Cryptology ePrint Archive, Report 2005/142.

[6] COURTOIS, N.—KLIMOV, A.—PATARIN, J.—SHAMIR, A.: *Efficient algorithms for solving overdefined systems of multivariate polynomial equations*, in: EUROCRYPT '00, Lecture Notes in Comput. Sci., Vol. 1807, Springer-Verlag, Berlin, 2000, pp. 392–407.

[7] INDYK, P.: *Deterministic superimposed coding with application to pattern matching*, in: FOCS '97, IEEE, 1997, pp.127–136.

[8] ERDÖSZ, P.—FRANKL, P.—FÜREDI, Z.: *Families of finite sets in which no set is covered by the union of r others*, Israel J. Math. **51** (1985), 79–89.

[9] CLEMENTI, A.—MONTI, A.—SILVESTRI, R.: *Selective families, superimposed codes, and broadcasting on unknown radio network*, in: SODA '01, 2001, pp. 709–718.

Received September 29, 2004

Centrum voor Wiskunde en Informatica
Kruislaan 413
NL–1098 SJ Amsterdam
NETHERLANDS
E-mail: J.Byrka@cwi.nl

Institute of Computer Science
University of Wrocław
ul. Przesmyckiego 20
PL–51-151 Wrocław
POLAND
E-mail: jbyrka@wp.pl