IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 1, 20 pages.

https://doi.org/10.62056/av4fe0iuc

Broadcast Encryption using Sum-Product decomposition of Boolean functions

Aurélien Dupin and Simon Abelard

Thales SIX France, Service de Cryptologie, Gennevilliers, France

Abstract. The problem of Broadcast Encryption (BE) consists in broadcasting an encrypted message to a large number of users or receiving devices in such a way that the emitter of the message can control which of the users can or cannot decrypt it. Since the early 1990s, the design of BE schemes has received significant interest and many different concepts were proposed. A major breakthrough was achieved by Naor, Naor and Lotspiech (CRYPTO 2001) by partitioning cleverly the set of authorized users and associating a symmetric key to each subset. Since then, while there have been many advances in public-key based BE schemes, mostly based on bilinear maps, little was made on symmetric cryptography.
In this paper, we design a new symmetric-based BE scheme, named ΣΠBE, that relies on logic optimization and consensual security assumptions. It is competitive with the work of Naor et al. and provides a different tradeoff: the bandwidth requirement is significantly lowered at the cost of an increase in the key storage.

Keywords: Broadcast Encryption · Boolean Functions

1 Introduction

Broadcast Encryption

The problem of Broadcast Encryption consists in broadcasting an encrypted message to a large number of users or receiving devices in such a way that the emitter of the message can control which of the users can or cannot decrypt it. A typical setting is the case of access control to a service such as pay-per-view TV, satellite Internet service, etc. In this case, one or several emitting facilities can distribute access keys through a Broadcast Encryption Scheme (BES or BE scheme) so that only the legitimate users (e.g. the ones who are still paying for the service) are able to recover keys while the revoked users (e.g. those who did not renew their subscription) are not. The advantage of a BE scheme is that the status of a user (i.e. revoked or authorized) can be modified at each new broadcast, which means that a user who stopped paying for the service can easily re-subscribe to it later on.

In the literature, Broadcast Encryption is often associated to Traitor Tracing which is a slightly different problem motivated by copyright issues in the context of digital media such as CD/DVD/Blu-Rays. In this context, no one can prevent a legitimate user (e.g. the purchaser of a DVD) to disseminate the content of the media, but a Traitor Tracing Scheme ensures that such a user cannot mask his identity and will therefore have to face the subsequent legal procedures. The present paper focuses on Broadcast Encryption and does not further explore the context of Traitor Tracing.

E-mail: aurelien.dupin@thalesgroup.com (Aurélien Dupin), sabelard@protonmail.com (Simon Abelard)

Received: 2024-01-08 Accepted: 2024-03-05

Since the early 1990s, the design of BE schemes has received significant interest and many different concepts were proposed. We give an overview of the literature on the subject in the next paragraph but let us first give a few criteria that are of importance in order to compare the relative performance of BE schemes. An obvious criterion is security: robust BE schemes offer **full collusion resilience**, meaning that no matter how many revoked users gather their credentials, they remain unable to decrypt the broadcast message. On the other hand, some BE schemes only resist to collusions of up to k revoked users and are thus called **k-collusion resistant** while some are probabilistic, meaning that there is a slim chance that a receiver or a collusion might still be able to decrypt the broadcast message. Another important distinction is between **static** schemes where credentials are initially generated once and for all by the emitter (meaning that beyond a certain number of users the emitter will have to re-run a potentially costly setup procedure in order to accommodate for new users) and **dynamic** schemes for which new users' credentials can be generated on the fly. Although it was not a concern at the time most of the BE schemes were invented, it is now particularly relevant to distinguish them based on their resistance to a quantum attacker, in which case we write that a BE scheme is **postquantum**. Finally, the other criteria are related to the performance of the BE schemes and are often subject to a trade-off:

- Bandwidth consumption (and in particular its dependence on the number of revoked users)
- Key storage both at emitter and receiver level
- Amount of computations performed both at emitter and receiver level (the latter usually being the most critical)

State of the art on Broadcast Encryption

A naive approach to BE consists in giving each user a symmetric key also stored by the emitter and encrypt the broadcast message with the key of each authorized user. This has unrealistic bandwidth requirements when there are too many users. The opposite extreme is to affect a symmetric key to each subset of the set of users and use the corresponding key to broadcast a message to any set of authorized users. While this is bandwidth-optimal, the key-storage is exponential in the total number of users and becomes unpractical even for 100 users. The seminal approach to BE [FN93] used an approach based on combinatorics which offers only probabilistic revocation at the price of heavy bandwidth requirements. This approach was later improved using code-based constructions [KRS99] but the resulting scheme still performs poorly and does not have full collusion resilience.

A major breakthrough was achieved by Naor, Naor and Lotspiech [NNL01], introducing the Subset-Cover paradigm. The main idea is to associate symmetric keys to specific subsets of users so that any set S of authorized users can be expressed as a partition involving such subsets. This framework is highly versatile as it offers a trade-off: increasing the number of special subsets will increase the storage requirement but on average reduce the bandwidth (more sets means that less sets should be used to partition S). While such a construction is not very practical on pure combinatorial grounds, [NNL01] cleverly organizes the users as leaves of a complete binary tree in order to turn this concept into an efficient BE scheme. Two instantiations are proposed in [NNL01]: the well-named Complete Subtree (CS) where the special sets correspond to complete subtrees of the initial tree and the Subset Difference (SD) where the special sets are differences of subtrees (i.e. a complete subtree T minus a complete subtree of T). Both constructions are fully collusion resilient and rely on simple and consensual security assumptions (resilience of the underlying blockcipher and PRF). The main differences between CS and SD is that the first option requires less storage at the price of a heavier bandwidth consumption.

These two schemes were customized and refined into other BE schemes in order to address a wider range of trade-offs between storage and bandwidth, such as the Layered Subset Difference [HS02]. Note that the LSD scheme follows the opposite purpose to our work as it increases the bandwidth consumption in order to reduce the storage requirements at user level. The consequence is that LSD never outperforms SD in terms of bandwidth and this is why we preferably compare our work to SD rather than LSD. Another customizable variant of the SD was proposed by [BS16]. In the extreme customization, it becomes equivalent to a naive solution where each possible subset of users is assigned a key. We have analyzed that it needs an important key storage per user ($> 10^6$ keys for 1000 users) to outperform our scheme in terms of bandwidth, although it will scale better when the number of users becomes very large ($n \geq 10^6$).

While we only mentioned BE schemes relying on symmetric cryptography, many public-key BE schemes were also proposed in the past decades, such as [TT01, DF03, AWY20]. Most of these schemes mimic classical public-key constructions such as Diffie-Hellman or RSA. Among them, a very interesting construction relies on bilinear maps [BGW05], which can be instantiated using pairings [PPSS13], yielding a very efficient BE scheme. Indeed, although it may be a bit hard on storage or on the computations to be performed by receiving devices, it is the only practical construction offering a constant overhead, meaning that the broadcast message sent by the emitter contains the necessary information (i.e. the list of revoked or authorized users and the message) plus an overhead whose size is constant (in practice it consists of two points on elliptic curves). A practical implementation is described in [DGB12]. The main weakness of such schemes is that they rely on number-theoretic problems that can be solved in polynomial time using quantum computers. Finding postquantum public-key BE schemes is an active topic of research but the currently available options [Wee22, SDSP23] are not competitive compared to the CS and SD schemes. Indeed, Table 2 of [SDSP23] shows that for n the total number of users, the size of the ciphertext is proportional to n^9 while the bandwidth consumption of the SD is linear in the number of revoked users. It is harder to assess the bandwidth requirements of [Wee22] since no cryptographic parameters are provided in the paper. However, although the growth of the ciphertext is shown to be polynomial in $\log n$, the constants hidden in the $O()$ are likely to make the scheme competitive only for very large n. By extrapolating the parameters recommended in FrodoKEM, a KEM relying on LWE, we assess that the ciphertext would always be greater than a megabit, regardless of the number of (denied) users. In our experiments, our scheme has a ciphertext smaller by one to two orders of magnitude.

Summary of our contribution

In this paper, we present a new BE scheme that offers various advantages: just like the ones from [NNL01], it relies on simple and consensual security assumptions (resilience of a blockcipher and PRF) with lower bandwidth requirements, at the price of a significant increase in key storage and potentially heavy computations at the emitter facility. While our BE scheme can be seen as an adaptive CS where the binary tree structure is re-generated at every broadcast in order to optimize bandwidth, we emphasize that this BE scheme **does not** follow the philosophy of the Subset-Cover because we do not compute a partition of the set S. Indeed, we rather see S as a Boolean function and compute a $\Sigma\Pi$-factorization yielding subgroups S_i that cover S but we do not preclude one user from belonging to two distinct S_i's as this does not affect the security of our scheme.

Considering only postquantum BE schemes, the most bandwidth-efficient to our knowledge is the SD introduced in [NNL01] as shown in Table 1. We report implementation results showing that our BE scheme can easily accommodate for thousands of users even when the emitter consists in a single laptop, and that it quickly outperforms the SD in terms of bandwidth when the number of revoked users grow: in our test scenarios the

Table 1: Performance comparisons for n users of which r are revoked

	Key storage		Average
	Emitter	Receiver	overhead
[NNL01] CS	$n-1$	$\log(n)+1$	$O(r\log_2(n/r))$
[NNL01] SD	$n-1$	$\frac{1}{2}\log(n)^2$	$1.25r$
This work	$2\log(n)$	n	$\leq r+\log_2(n)$ (empirical)

turning point is between 1 and 2% of revoked users and for more than 5% of revoked users it already provides a 25% decrease in bandwidth compared to the SD.

This makes our BE scheme particularly suitable to the setting where bandwidth is expensive or limited, such as space transmissions or communications with several layers of protection. We also emphasize that the constraints on the receiving devices are more than reasonable even for hand-held devices such as smartphones or radios: they only need to support a block cipher and PRF and the key-storage requirements are kept below 1 GB even for 2^{26} (i.e. > 67 million) users when using 128-bit symmetric keys.

An implementation of our solution is available at:

https://github.com/88abaa99/SPBE

2 Preliminaries

2.1 Broadcast encryption

A typical symmetric BE scheme needs four procedures:

- *Setup*$(n, \lambda, \$) \rightarrow k_{\text{master}}$: using a security parameter λ, some source of randomness \$ and the number of users n (or an upper-bound), the emitter derives some key material k_{master}.
- *Join*$(u, k_{\text{master}}) \rightarrow k_u$: the emitter derives and sends the key material k_u of user u $(0 \leq u < n)$.
- *Encrypt*$(m, \mathcal{A}, k_{\text{master}}) \rightarrow (h, c)$: using its key material k_{master}, the emitter encrypts the message m so that only users in the set of authorized users $\mathcal{A}$ are able to decrypt. It outputs the ciphertext c and some header h containing decryption instructions. It can be as simple as $h = \mathcal{A}$.
- *Decrypt*$(h, c, k_u) \rightarrow m$: the user u uses the header h and its key k_u to decrypt c to obtain m. In the event that the decryption fails because user u is not authorized, the output of the procedure is null.

Symmetric-based BE schemes generally relies on one or two block cipher modes of operation (that provides confidentiality and optionally authenticity), that we denote by (E, D) and $(E_{\text{payload}}, D_{\text{payload}})$. The output of the *Encrypt* procedure can often be seen as follows:

$$\begin{aligned} h &= i_1 \| i_2 \| \cdots \| i_l \\ c &= \underbrace{E(k_1, k_{\text{e}}) \| E(k_2, k_{\text{e}}) \| \cdots \| E(k_l, k_{\text{e}})}_{\text{overhead}} \| E_{\text{payload}}(k_{\text{e}}, m) \end{aligned}$$

where h encodes which keys are to be used (it can be as simple as the list of revoked users or be a more evolved construction) and k_{e} is an ephemeral key used to encrypt m. The key k_{e} is then encrypted several times with the keys $k_j (1 \leq j \leq l)$ that are possessed or that can be derived by authorized users. Note that this paradigm is slightly different from the traditional one of Naor et al. [NNL01]:

- the intermediate encryptions of the ephemeral key are part of the ciphertext c instead of the header h. We prefer to merge the elements that must be indistinguishable from random in c.
- the information $i_j (1 \leq j \leq l)$ are not necessarily indexes that refer to disjoint subsets. It may contain more general information that helps authorized users to decrypt. In particular, an authorized user may have several ways of decrypting the ciphertext.
- we add that notion of *overhead* for performance comparison. It contains all c but the encrypted message.

Most BE schemes focus on generating as few encryptions of k_e as possible, thus reducing the overhead.

2.2 Boolean functions

Definition 1 (Boolean Function - “don’t care” values)**.** A Boolean function in l variables is of the form $f : \mathbb{F}_2^l \to \mathbb{F}_2$. Its evaluation in the vector $\mathbf{x} = (x_1, \cdots, x_l)$ may equivalently be referred to as $f(\mathbf{x})$ or $f(x_1, \cdots, x_l)$.

Optionally, f may have “don’t care” values, denoted by a star $*$ (for example $f(0, 1, 1) = *$). This fictive third value represents the case where we are not interested in some evaluations of f.

With the notion of “don’t care” values, we need to redefine the concept of support and introduce the one of strict support.

Definition 2 (Support - Strict support)**.** Let f be a Boolean function in l variables. The support of f is the set of inputs $\mathbf{x} \in \mathbb{F}_2^l$ such that $f(\mathbf{x}) = 1$ or $*$.

The strict support of f is the set of inputs $\mathbf{x} \in \mathbb{F}_2^l$ such that $f(\mathbf{x}) = 1$. “Don’t care” values do not belong to the strict support.

The cornerstone of our BE schemes relies on representing a Boolean function as a sum of products.

Definition 3 (Sum of products - Product term - ΣΠ-form)**.** Let f be a Boolean function in l variables x_1 to x_l. A product term is a conjunction of variables or negation of variables (e.g. x_1, $x_2\bar{x_3}$, $\bar{x_1}\bar{x_2}x_4$).

A sum of products is a disjunction of product terms (e.g. $x_1 \vee x_2\bar{x_3} \vee \bar{x_1}\bar{x_2}x_4$). A ΣΠ-form of a function f refers to a sum of products representing f.

Any function f has a ΣΠ-form. A naive solution would be to define a product term for each element of the support of f (for example if $f(0, 1, 1) = 1$, then define $\bar{x_1}x_2x_3$). The function f can clearly be expressed as the sum of these terms. For instance, taking the example of Section 4 where $f(u) = 0$ for $u \in \{1, 3, 5\}$ and 1 anywhere else for u a 3-bit number, one can naively express $f = \bar{x_1}\bar{x_2}\bar{x_3} \vee \bar{x_1}x_2\bar{x_3} \vee x_1\bar{x_2}\bar{x_3} \vee x_1x_2\bar{x_3} \vee x_1x_2x_3$ because $f(u) = 1$ if and only if $u \in \{(0, 0, 0), (0, 1, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)\}$.

A ΣΠ-form of f is generally not unique and we are interested in ΣΠ-forms having a small number of products in order to minimize the overhead. The Quine-McCluskey algorithm allows to solve the problem of finding the smallest sum of products, which appears to be an NP-hard problem. Any alternative algorithm would fit our needs. In particular, the ESPRESSO algorithm should be considered when dealing with a large number of variables, at the cost of finding a good but possibly non-optimal solution.

In Section 3, we describe (a close variant of) the Quine-McCluskey algorithm. A reader only interested in the BE schemes may directly go to Section 4 and take this algorithm in black-box.

2.3 Pseudorandom functions

One of our constructions requires the use of pseudorandom functions.

Definition 4 (Pseudorandom Function [KL14])**.** Let $F : \mathcal{K} \times \mathcal{X} \mapsto \mathcal{Y}$ be an efficient keyed function. F is a pseudorandom function (PRF) if for all polynomial-time distinguisher D, there is a negligible function negl such that:

$$|Pr[D^{F(k,\cdot)}(1^\lambda) = 1] - Pr[D^{f(\cdot)}(1^\lambda) = 1]| \leq \mathrm{negl}(\lambda)$$

for some security parameter λ, where the first probability is taken over uniform choice of $k \in \mathcal{K}$ and the second probability is taken over a random function $f : \mathcal{X} \mapsto \mathcal{Y}$.

Informally, F is a pseudorandom function if any adversary having access to an oracle $F(k, \cdot)$ cannot distinguish F from a truly random function f.

In the rest of the paper, we actually require a weaker notion of pseudorandom function: instead of being given access to the oracle $F(k, \cdot)$, the adversary is only given a few known (but not chosen) evaluations. Then, the adversary must not be able to evaluate $F(k, \cdot)$ on other points. This security notion is implied by the indistinguishability as described above.

3 The Quine-McCluskey algorithm

In this section, we detail the Quine-McCluskey algorithm, which allows to solve the problem of finding the smallest sum of products, i.e. among all the $\Sigma\Pi$-forms defined in Definition 3, find one which minimizes the number of $\vee$ operands. We actually describe a close variant, that allows to obtain a degraded solution when the number of variables becomes too large to be solved in reasonable time.

3.1 Description

This approach tolerates that a Boolean function has "don't care" values, introduced in Definition 1. The Quine-McCluskey algorithm then solves the problem without being constrained by these "don't care" values.

First, the notions of implicant and prime implicant needs to be introduced.

Definition 5 (Implicant - Size-2^α implicant)**.** Let f be a Boolean function in l variables and p be a product term. We say that p is an implicant of f if and only if p implies f (i.e. for all vector $\mathbf{x} \in \mathbb{F}_2^l$, $p(\mathbf{x}) = 1 \implies f(\mathbf{x}) = 1$).

The implicant p is a size-2^α implicant if its support contains 2^α elements.

Property 1. Remark that since an implicant is a product term, its support necessarily contains a power of two elements. Moreover an implicant is a size-2^α implicant if and only if it is a product of $l - \alpha$ variables (or negation of variables).

Definition 6 (Prime implicant)**.** Let p be a product term. We say that p is a prime implicant of the Boolean function f if:

- it is an implicant of f,
- there exists no other implicant $p' \neq p$ of f such that p is an implicant of p'.

Note that if p' exists, then it is a "more general" implicant of f, and consists in a strict subset of variables (or negation of variables) of p.

Let f be a Boolean function in l variables, the Quine-McCluskey algorithm is a 2-step process which consists in:

1. generating all prime implicants of the Boolean function f.

x_0	x_1	x_2	x_3	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	*
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	*
1	1	1	1	1

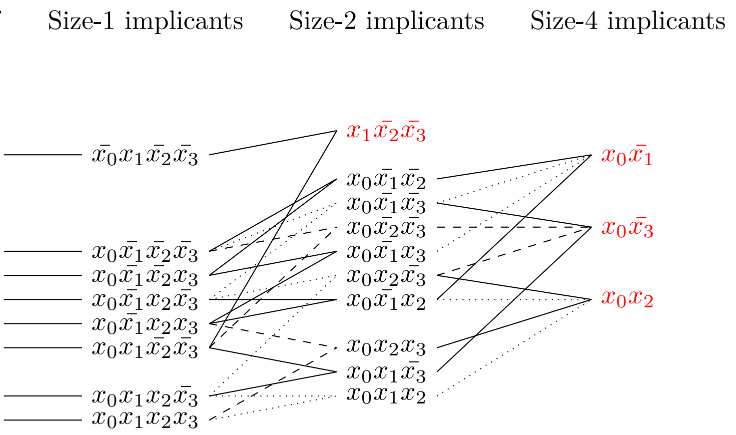

Size-1 implicants
Size-2 implicants
Size-4 implicants
$\bar{x}_0 x_1 \bar{x}_2 \bar{x}_3$
$x_0 \bar{x}_1 \bar{x}_2 \bar{x}_3$
$x_0 \bar{x}_1 \bar{x}_2 x_3$
$x_0 \bar{x}_1 x_2 \bar{x}_3$
$x_0 \bar{x}_1 x_2 x_3$
$x_0 x_1 \bar{x}_2 \bar{x}_3$
$x_0 x_1 x_2 \bar{x}_3$
$x_0 x_1 x_2 x_3$
$x_1 \bar{x}_2 \bar{x}_3$
$x_0 \bar{x}_1 \bar{x}_2$
$x_0 \bar{x}_1 \bar{x}_3$
$x_0 \bar{x}_2 \bar{x}_3$
$x_0 \bar{x}_1 x_3$
$x_0 x_2 \bar{x}_3$
$x_0 \bar{x}_1 x_2$
$x_0 x_2 x_3$
$x_0 x_1 \bar{x}_3$
$x_0 x_1 x_2$
$x_0 \bar{x}_1$
$x_0 \bar{x}_3$
$x_0 x_2$

Figure 1: Generation of prime implicants with the QuineMcCluskey algorithm. Prime implicants are in red.

	4	8	10	11	12	15
$p_0 = x_1\bar{x}_2\bar{x}_3$	X				X	
$p_1 = x_0\bar{x}_1$		X	X	X		
$p_2 = x_0\bar{x}_3$		X	X		X	
$p_3 = x_0 x_2$			X	X		X

Petrick's function: $p'_0 \wedge (p'_1 \vee p'_2) \wedge (p'_1 \vee p'_2 \vee p'_3) \wedge (p'_1 \vee p'_3) \wedge (p'_0 \vee p'_2) \wedge p'_3$

Figure 2: Prime implicant chart and Petrick's function

2. finding the smallest subset of prime implicants that describes f. This part is similar to the set cover problem. The sum of the obtained implicants is the smallest $\Sigma\Pi$-form of f.

The algorithm starts from the naive solution stated earlier: for each element of the support of f (i.e. $f(\mathbf{x}) = 1$ or $f(\mathbf{x}) = *$), create the size-1 implicant that is true only for that element. An example is given in Figure 1. Using Property 1, such implicants all have l variables.

Then, iteratively, the algorithm builds size-$2^{\alpha+1}$ implicants by combining two size-2^α implicants as follows: let p_1 and p_2 be two size-2^α implicants such that they share the same variables and differ in a single negation. They can be written as $p_1 = p_3 x_i$ and $p_2 = p_3 \bar{x}_i$ for some product term p_3. Therefore, p_3 is a size-$2^{\alpha+1}$ implicant of f. This can be easily proven using Property 1 and the fact that $p_3 = p_1 \oplus p_2$. Such combinations are searched exhaustively. There may exist several combinations ending in the same size-$2^{\alpha+1}$ implicant.

When implicants cannot be combined any further, the remaining implicants are prime implicants. Although it is not a minimal decomposition yet, f can already be written as the sum (i.e. disjunction) of these prime implicants.

In the second step of the algorithm, we aim at extracting the smallest subset of prime implicants. We start by constructing the prime implicant chart, as shown in Figure 2. Each row is mapped to a prime implicant generated in Step 1 and each column correspond to an element $\mathbf{x}$ of the strict support of f (i.e. $f(\mathbf{x}) = 1$). A cell then indicates whether a prime implicant covers an element $\mathbf{x}$.

From this chart, one can easily compute the corresponding Petrick's function also

Petrick's function: $p'_0 \wedge (p'_1 \vee p'_2) \wedge (p'_1 \vee p'_2 \vee p'_3) \wedge (p'_1 \vee p'_3) \wedge (p'_0 \vee p'_2) \wedge p'_3$

Linear system:	$p'_0 \geq 1$	$p'_1 + p'_2 \geq 1$	$p'_1 + p'_2 + p'_3 \geq 1$
	$p'_1 + p'_3 \geq 1$	$p'_0 + p'_2 \geq 1$	$p'_3 \geq 1$
Function to minimize:	$p'_0 + p'_1 + p'_2 + p'_3$		

Figure 3: Linear modelization of the Petrick's function

shown in Figure 2. For each prime implicant p_i, a binary variable p'_i is generated. It takes the value 1 if p_i is kept in the smallest subset, 0 otherwise. The Petrick's function consists in a conjunction of disjonctions of such binary variables. A disjunction indicates which implicants are necessary to cover a specific element of the strict support of f. Therefore, the Petrick's function is true if at least one implicant in each disjonction is kept in the smallest $\Sigma\Pi$-form of f.

In order to find the smallest solution in the Petrick's function, the so-called Petrick's method introduced in [Pet56] is generally considered. However, it has an exponential complexity that makes it unsuitable for a large number of variables or, more importantly, when the size of the support of f is big, which is true in our scenario. We do not detail more this method. Instead, we follow the idea of [CFN61] and use Integer Linear Programming to solve this problem. ILP is unlikely to have a lower complexity than other approaches to solve the set cover problem, but it has the advantage of having meaningful intermediate results that are hopefully close to optimality.

The Petrick's function directly leads to a linear model:

- for each disjunction $\bigvee_{i\in\mathcal{S}} p'_i$ (for some subset $\mathcal{S}$), the linear constraint $\sum_{i\in\mathcal{S}} p'_i \geq 1$ is added to the model,
- since we search for the smallest subset of implicants, the objective function to minimize is the sum of all p'_i.

An example is given in Figure 3.

Finally, the smallest $\Sigma\Pi$-form of f is defined by the sum of implicants p_i, for which $p'_i = 1$ in the minimal solution given by the ILP solver.

Remark that some disjunctions may have a single variable (for example p'_0 and p'_3 in Figures 2 and 3). The corresponding implicants (p_0 and p_3) are necessarily in the smallest $\Sigma\Pi$-form. Therefore, the Petrick's function can be simplified by setting these variables to 1 ($p'_0 = p'_3 = 1$), thus reducing the number of variables and disjunctions. This phenomenon is referred as "essential prime implicants" in the original Quine-McCluskey algorithm.

Optionally, once an optimal or suboptimal solution is found, "don't care" values of the Boolean function f (if any) can finally be assigned 0 or 1, depending on the solution.

3.2 Practical considerations

Computing the prime implicants may require exponential time, as there may exist $O(3^l/\sqrt{l})$ prime implicants, see [CM78]. However, when implemented properly, this part is not an issue compared to the second step for the considered parameters. In particular, the choice of the data structure has a considerable impact on the efficiency: for building size-$2^{\alpha+1}$ implicants, we start by partitioning size-2^{α} implicants by the variables they possess and by the number of negations they have. As already mentioned, an implicant can only be combined with an implicant having the same variables but differing in a single negation. Thus, the partition we propose makes the search of appropriate implicants more efficient. Also remark that there may exist several combinations ending in the same higher-size implicant. Following these observations, our python code easily reaches $l = 13$.

For larger values, a C code will be preferred and parallelisation should be deployed. We recommend to parallelise on the subsets of the partition described above: each thread is given a subset of implicants and tries to combine them with subsets having the same variables and an extra negation.

The second step is equivalent to the set cover problem, which is itself an NP-hard problem. The complexity of our technique, based on Integer Linear Programming (including simplex and branch-and-bound/cut), is unclear. In our experiments, it behaves exponentially. We use CPLEX as ILP solver. Up to $l = 9$, an optimal solution is found and proved in less than a second. With $l \geq 10$, unless f has some specific structure, the ILP solver fails to output an optimal solution within a few hours and using less than our 16Go memory. However, even for $l = 12$ (the maximal value for our experiments), CPLEX outputs a high-quality solution within few seconds.

4 The non-collusion-resistant BE scheme

In this section, we introduce ΣΠBE-NCR, a novel symmetric broadcast encryption scheme based on a completely new paradigm. For n users among which r are unauthorized, it requires $\log_2(n)$ keys per user and $O(r)$ messages. In practice, the number of messages is lower than r, except for very small values of r. However, the solution is **not secure against** (almost) **any collusion** of at least two users, but is used as a first step towards a fully secure scheme.

4.1 Description

We now describe our solution. For the sake of simplicity, let us consider that the number of users n is a power of two and let $l = \log_2(n)$. To illustrate the different procedures, let the system have $n = 8$ users, among which users labelled 1, 3 and 5 are revoked. We also use two encryption schemes E and E_{payload} that may or may not be the same. In particular, E_{payload} may be an authenticated encryption scheme. The scheme also needs a one-way function F.

4.1.1 Setup

The key material k_{master} of the emitter consists in $2l$ keys. For $0 \leq i < l$ and $0 \leq j \leq 1$, the key k_i^j is randomly generated and stored.

In our example, the system contains six keys: k_0^0, k_0^1, k_1^0, k_1^1, k_2^0 and k_2^1.

4.1.2 Join

Let $u_0 \| u_1 \| ... \| u_{l-1}$ be the binary decomposition of u. When user u needs to join the broadcast protocol, the emitter sends it $k_i^{u_i}$ for all $0 \leq i < l$.

In the above example, user $4 = 0b100$ receives k_0^1, k_1^0 and k_2^0.

4.1.3 Encrypt

In the encryption procedure, the emitter starts by generating an ephemeral key k_e and by encrypting the message: $E_{\text{payload}}(k_e, m)$ which is part of c. (Note that this step can and should be discarded when there is no revoked user, the message is directly encrypted using a common pre-shared key).

Let $\mathcal{A} \subset [\![0; n-1]\!]$ be the set of authorized users for this session. Let f be the Boolean function in l variables defined by $f(u) = 1$ if $u \in \mathcal{A}$, $f(u) = 0$ otherwise. The emitter then computes the minimal ΣΠ-form of f using our variant of the Quine-McCluskey algorithm from Section 3. It can be replaced by other algorithms, but depending on the parameters of the systems, we highly suggest to choose an approach that allows to find a good suboptimal solution when an optimal solution cannot be found using reasonable time or resources.

In our scenario, the minimal ΣΠ-form of f is:

$$f(u) = \bar{u_2} \vee u_0 u_1.$$

The header h consists in an encoding of f (see Section 5.3). The ciphertext c contains several encryptions of the ephemeral key k_e, which needs to be encrypted for every product term of the ΣΠ-form. Let $p_1 = \bigwedge_{i \in \mathcal{S}_1} u_i \bigwedge_{i \in \mathcal{S}_2} \bar{u_i}$ be the first product term (w.l.o.g.) of the ΣΠ-form for some ordered subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of $[\![0; n-1]\!]$. For each such product term, the emitter encrypts k_e as follows:

$$c_1 = E(k_1, k_e) \ \text{ with } \ k_1 = F\left((\|_{i \in \mathcal{S}_1} k_i^1) \parallel (\|_{i \in \mathcal{S}_2} k_i^0)\right)$$

In our example, the ciphertext is composed of two ciphertexts:

$$c = \underbrace{E(k_1, k_e)}_{c_1} \| \overbrace{E(k_2, k_e)}^{c_2} \| E_{\text{payload}}(k_e, m) \qquad f(u) = \underbrace{\bar{u_2}}_{k_1 = F(k_2^0)} \vee \overbrace{u_0 u_1}^{k_2 = F(k_0^1 \| k_1^1)} .$$

The header h and ciphertext c are broadcast.

For large values of l, the computation of the ΣΠ-form of f soon becomes the bottleneck of this procedure. However, it only needs to be computed when $\mathcal{A}$ is modified. If the set $\mathcal{A}$ does not evolve too often (i.e. less than every minute), this part can be made offline, as it does not depend on the message.

4.1.4 Decrypt

While decrypting, user u first checks with the header h that $f(u) = 1$. In the opposite case, u is not authorized and aborts. The user then searches which product term p_i of f is such that $p_i(u) = 1$. There might be several such terms, in which case the user arbitrarily choose one of them (in practice the first convenient factor encountered). As in the encryption procedure, it derives k_i with the possessed key materials and F, decrypts k_e using the corresponding part of the ciphertext and lastly decrypts m.

Here are a few examples:

user	used p_i	decryption of k_e
$u = 0b100$	$\bar{u_2} = 1$	$k_e = D(F(k_2^0), c_1)$
$u = 0b111$	$u_0 u_1 = 1$	$k_e = D(F(k_0^1 \| k_1^1), c_2)$
$u = 0b001$	None	cannot decrypt

4.2 Analysis

The scheme is correct: an authorized user is able to decrypt the plaintext.

Correctness. By construction of f, an authorized user u satisfies $f(u) = 1$. The Quine-McCluskey algorithm (or a suboptimal variant) expresses f as a sum of products $f = \bigvee_i p_i$. If $f(u) = 1$, then there exists at least one product p_i such that $p_i(u) = 1$.

Let $p_1 = \bigwedge_{i \in \mathcal{S}_1} u_i \bigwedge_{i \in \mathcal{S}_2} \bar{u_i}$ be this product. The corresponding ephemeral key is encrypted with:

$$k_1 = F\left((\|_{i \in \mathcal{S}_1} k_i^1) \parallel (\|_{i \in \mathcal{S}_2} k_i^0)\right)$$

Since $u = u_0 \| u_1 \| ... \| u_{l-1}$ and $p_1(u) = 1$, the user knows $k_i^{u_i} = k_i^1$ for $i \in \mathcal{S}_1$ and $k_i^{u_i} = k_i^0$ for $i \in \mathcal{S}_2$. As a consequence, it has the necessary key materials to reconstruct k_1, then to decrypt the ephemeral key k_e and finally m. □

The scheme is secure as long as E, E_{payload} and F are secure: revoked users are unable to decrypt.

Security. By construction of f, an unauthorized user $v = v_0\|v_1\|...\|v_{l-1}$ satisfies $f(v) = 0$. Let us represent f as a sum of products $f = \bigvee_i p_i$. If $f(v) = 0$, then all p_i are such that $p_i(v) = 0$.

Let $p_1(u) = \bigwedge_{i\in\mathcal{S}_1} u_i \bigwedge_{i\in\mathcal{S}_2} \bar{u}_i$ be one of the products of f. Since $p_1(v) = 0$, there exists at least one index j such that either $v_j = 0$ and $j \in \mathcal{S}_1$ or $v_j = 1$ and $j \in \mathcal{S}_2$. By construction k_1 is computed from k_i^1 for $i \in \mathcal{S}_1$ and from k_i^0 for $i \in \mathcal{S}_2$. Then, user v misses at least $k_j^{\bar{v}_j}$, and thus cannot reconstruct k_1 if F is secure. The same argument holds for every product p_i of the $\Sigma\Pi$-form of f.

If E and E_{payload} are secure, the unauthorized user is unable to retrieve either k_e or m, which ends the proof. □

However, any collusion of users u and v that differ in at least two positions $u_i \neq v_i$ and $u_j \neq v_j$ for distinct indexes i and j, is able to decrypt transmissions that neither u nor v is able to decrypt alone. In particular, let u and v be two users such that $u_i \neq v_i$ for all $0 \leq i < l$. Following the join procedure, one can easily see that they possess the whole key materials of the system and thus are able to decrypt any broadcast.

5 The collusion-resistant BE scheme

In this section, we modify the ΣΠBE-NCR scheme to prevent any collusion of unauthorized users from decrypting, at the cost of n keys being stored by each user. The encryption size remains unchanged.

The one-way function F is now replaced by a pseudorandom function, as defined in Section 2.3, still referred to as F.

We now describe the four new procedures, taking as example $n = 8$.

5.0.1 Setup

The key material of the emitter now consists in a single key k_{PRF}, that is randomly generated and stored.

As in the previous scheme, for $0 \leq i < l$ and $0 \leq j \leq 1$, values k_i^j are generated and stored. However, unlike the ΣΠBE-NCR scheme, they need not be random or secret, as long as it is guaranteed that they are all distinct and that any concatenation is unambiguous. It can be as simple as $k_i^j = i\|j$ encoded on a fixed number of bits. In order to be consistent with the previous scheme, we keep the notation k_i^j, but it should be considered as a label and not as a key.

In our example, the system contains a key k_{PRF} and six labels k_0^0, k_0^1, k_1^0, k_1^1, k_2^0 and k_2^1.

5.0.2 Join

When user $u = u_0\|u_1\|...\|u_{l-1}$ needs to join the broadcast protocol, it suffices that the emitter sends it the key $F(k_{\text{PRF}}, \|_{i\in\mathcal{S}} k_i^{u_i})$ for all ordered $\mathcal{S} \subset [\![0; l-1]\!]$. For simplicity, we suggest to sort the elements of $\mathcal{S}$ in increasing order. There are $2^l = n$ of them. Note that these keys have a one-to-one correspondence with the product terms p such that $p(u) = 1$.

In the above example, user $4 = 0b100$ receives:

$$\begin{array}{lll} F(k_{\text{PRF}}, k_0^1) & F(k_{\text{PRF}}, k_0^1\|k_1^0) & F(k_{\text{PRF}}, \emptyset) \\ F(k_{\text{PRF}}, k_1^0) & F(k_{\text{PRF}}, k_0^1\|k_2^0) & F(k_{\text{PRF}}, k_0^1\|k_1^0\|k_2^0) \\ F(k_{\text{PRF}}, k_2^0) & F(k_{\text{PRF}}, k_1^0\|k_2^0)\ . & \end{array}$$

5.0.3 Encrypt

Let $\mathcal{A} \subset [\![0; n-1]\!]$ be the set of authorized users for this session. Let f be the Boolean function in l variables defined by $f(u) = 1$ if $u \in \mathcal{A}$, $f(u) = 0$ otherwise.

The beginning of the encryption procedure is identical to the ΣΠBE-NCR scheme, the emitter generates an ephemeral key k_e and encrypts the message: $E_{\text{payload}}(k_e, m)$. The emitter computes the minimal ΣΠ-form of f using the Quine-McCluskey algorithm of Section 3.

In our scenario, suppose the minimal ΣΠ-form of f is:

$$f(u) = \bar{u_2} \vee u_0 u_1.$$

The header h consists in an encoding of f, as in the ΣΠBE-NCR scheme. The ciphertext c contains several encryptions of the ephemeral key k_e, which needs to be encrypted for every product term of the ΣΠ-form. Let $p_1 = \bigwedge_{i \in \mathcal{S}_1} u_i \bigwedge_{i \in \mathcal{S}_2} \bar{u_i}$ be the first[1] product term (w.l.o.g.) of the ΣΠ-form for some subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of $[\![0; n-1]\!]$. For each such product term, the emitter encrypts k_e as follows:

$$c_1 = E(k_1, k_e) \ \text{ with } \ k_1 = F\left(k_{\text{PRF}}, \|_{i \in \mathcal{S}_1 \cup \mathcal{S}_2} \left\{ \begin{array}{ll} k_i^1 & \text{if } i \in \mathcal{S}_1 \\ k_i^0 & \text{if } i \in \mathcal{S}_2 \end{array} \right. \right)$$

The set $\mathcal{S}_1 \cup \mathcal{S}_2$ must be ordered to fit the ordering of the join procedure. As aforementioned, we recommend to sort the elements of $\mathcal{S}_1 \cup \mathcal{S}_2$ in increasing order. In our example, the header is composed of f and two ciphertexts:

$$c = \underbrace{E(k_1, k_e)}_{c_1} \| \overbrace{E(k_2, k_e)}^{c_2} \| E_{\text{payload}}(k_e, m) \qquad f(u) = \underbrace{\bar{u_2}}_{k_1 = F(k_{\text{PRF}}, k_2^0)} \vee \overbrace{u_0 u_1}^{k_2 = F(k_{\text{PRF}}, k_0^1 \| k_1^1)} .$$

The header h and ciphertext c are broadcast.

A method for encoding f is detailed in Section 5.3. An alternative encoding is possible as long as it allows to easily build a mapping between the product terms p_i of f and the encrypted keys k_i.

As in the ΣΠBE-NCR scheme, the computation of the ΣΠ-form of f can be made offline, as it only depends on the set of authorized users.

5.0.4 Decrypt

While decrypting, user u first checks that $f(u) = 1$. In the opposite case, u is not authorized and aborts. The user then searches which product term p_i of f is such that $p_i(u) = 1$. There might be several such terms, in which case the user arbitrarily chooses one of them. As we mention in the Join procedure, such a product term corresponds to one key k_i received by the user and defined as:

$$p_i = \bigwedge_{j \in \mathcal{S}_1} u_j \bigwedge_{j \in \mathcal{S}_2} \bar{u_j} \qquad k_i = F\left(k_{\text{PRF}}, \|_{j \in \mathcal{S}_1 \cup \mathcal{S}_2} \left\{ \begin{array}{ll} k_j^1 & \text{if } j \in \mathcal{S}_1 \\ k_j^0 & \text{if } j \in \mathcal{S}_2 \end{array} \right. \right)$$

Note that the computation of k_i is done by the emitter and the receiver is only injected with the list of keys described in the join procedure, contrary to the ΣΠBE-NCR scheme where the user could freely derive k_i. This more complicated design is precisely what makes our scheme collusion resistant: even knowing the k_j^i and the value of u, two colluding users are not able to compute more keys than they already have because they do not know the master key k_{PRF}.

[1]The notion of "first" depends on the encoding method of f. Our approach is described in Section 5.3.

Finally, thanks to k_i, the user decrypts k_e using the corresponding part c_i of the ciphertext and subsequently decrypts m.

Here are a few examples:

user	used p_i	decryption of k_e
$u = 0b100$	$\bar{u_2} = 1$	$k_e = D(F(k_{\text{PRF}}, k_2^0), c_1)$
$u = 0b111$	$u_0 u_1 = 1$	$k_e = D(F(k_{\text{PRF}}, k_0^1 \| k_1^1), c_2)$
$u = 0b001$	None	cannot decrypt

5.1 Analysis

The scheme is correct: an authorized user is able to decrypt the plaintext.

Correctness. The correctness can actually be reduced to the one of the ΣΠBE-NCR scheme. Instead of being derived by the user, all possible derivations of the key/label materials $k_i^{u_i}(0 \leq i < l)$ are now made by the emitter then sent to the user u. □

The scheme is collusion-resistant: any collusion of revoked users is unable to decrypt.

Collusion-resistance. Consider a set of authorized users $\mathcal{A} \subset [\![0; n-1]\!]$ and of revoked users $\mathcal{R} \subset [\![0; n-1]\!]/\mathcal{A}$. Let f be the Boolean function defined by $f(u) = 1 \iff u \in \mathcal{A}$. As usual, f is expressed in a ΣΠ-form: $f = \bigvee_i p_i$.

Assuming that E_{payload} and E are secure encryption schemes, the collusion $\mathcal{R}$ is able to decrypt if only if it manages to obtain a key k_i corresponding to a product term p_i. Recall that they are of the form:

$$p_i = \bigwedge_{j \in \mathcal{S}_1} u_j \bigwedge_{j \in \mathcal{S}_2} \bar{u_j} \qquad k_i = F\left(k_{\text{PRF}}, \|_{j \in \mathcal{S}_1 \cup \mathcal{S}_2} \begin{cases} k_j^1 & \text{if } j \in \mathcal{S}_1 \\ k_j^0 & \text{if } j \in \mathcal{S}_2 \end{cases}\right).$$

Although they possibly know all labels k_j^0 and k_j^1, the key k_{PRF} is known only to the emitter. If the collusion is able to decrypt, then:

- either k_i is known by at least one user $v \in \mathcal{R}$. Following the join and encrypt procedures, it implies that $f(v) = 1$, meaning that $v \in \mathcal{A}$. This contradicts the initial definition of $\mathcal{R}$.
- or the collusion is able to compute k_i from the other evaluations of the PRF F it possesses from the join procedure. If so, the collusion can be used to build an adversary that distinguishes F from a random function (see Section 2.3).

□

The security of the scheme against a single revoked user is implied by the collusion-resistance.

5.2 The case of "don't care" users

The encryption procedures of the ΣΠBE-NCR and ΣΠBE schemes first define a function f such that $f(u) = 1$ if and only if user u is authorized. Until now, it does not exploit the possibility of setting "don't care" values[2] in the Boolean function f. We see several applications of these values.

If the number of users n of the system is not a power of two, let $n' > n$ be the closest greater power of two. We recommend to instantiate the schemes with n' users and we talk about "don't care" users u for all $n \leq u < n'$. In the encryption procedures, we suggest to

[2]See Section 1.

define $f(u) = *$ for all $n \leq u < n'$. The Quine-McCluskey algorithm then affect either 0 or 1 to these "don't care" users, such that the $\Sigma\Pi$-form of f is minimal.

As another application, in the context where it is not needed to deliver a message to a user u but it is also not needed to protect this message from u (for instance a pay-TV subscriber who already has his credentials), u can be defined as "don't care" user. Therefore, the set of authorized users $\mathcal{A}$ of the encryption procedures must be separated in two sets :

- $\mathcal{A}_M$ for mandatory users. Let $f(u) = 1$ for all $u \in \mathcal{A}_M$.
- $\mathcal{A}_{DC}$ for "don't care" users. Let $f(u) = *$ for all $u \in \mathcal{A}_{DC}$.

An unauthorized user u, which is not in $\mathcal{A}_M \cup \mathcal{A}_{DC}$, remains unchanged: $f(u) = 0$. As a consequence, a "don't care" user u might be able to decrypt, which does not affect the security against unauthorized users.

Similarly, "don't care" users may also be used as spare slots for future needs: new users can be dynamically added using these spare slots. However, in that case, forward secrecy is not guaranteed: a newly added user u might be able to decrypt ciphertexts generated before u joined the system.

5.3 An encoding for sums of products

In this section, we propose a method for encoding the $\Sigma\Pi$-form of a Boolean Function f in a compact way. As usual, let n be the number of users and 2^l the closest greater or equal power of two.

The encoding of f requires an encoding of each product term. A product term contains at most l variables, numbered from 0 to $l-1$, which can be negated. For a given product term p, we denote by $\mathcal{S}_0$ the set of negated variables and by $\mathcal{S}_1$ the set of non-negated variables. $\mathcal{S}_0$ and $\mathcal{S}_1$ are (disjoint) subsets of $[\![0; l-1]\!]$ of size at most l. The product term is written as:

$$p = \bigwedge_{i \in \mathcal{S}_0} \bar{x_i} \ \wedge \ \bigwedge_{i \in \mathcal{S}_1} x_i \ .$$

We propose to encode $\mathcal{S}_0$ (resp. $\mathcal{S}_1$) as an l-bit string where the i-th bit is set to 1 if and only if $i \in \mathcal{S}_0$ (resp. $i \in \mathcal{S}_1$). The term p is then encoded as the $2l$-bit string defined by the concatenation of the encoding of $\mathcal{S}_0$ and $\mathcal{S}_1$. For example with $l = 4$:

$$x_0 x_1 \bar{x_2} \ \rightarrow \ \mathcal{S}_0 = \{2\}, \mathcal{S}_1 = \{0,1\} \ \rightarrow \ \overbrace{0010}^{\mathcal{S}_0}\ \overbrace{1100}^{\mathcal{S}_1} \ .$$

Since f is a disjunction of product terms, it now suffices to concatenate the number of product terms and the encoding of each one of them. A (very large) upper bound on this number is n, the number of users. Indeed, as mentioned in Section 3, a naive $\Sigma\Pi$-form of f involves all elements of the support of f. Therefore the number of product terms is represented by an l-bit integer. As example, taking $n = 16$ and $l = 4$:

$$x_0 x_1 \bar{x_2} \vee \bar{x_1} \bar{x_3} x_2 \vee \bar{x_0} x_1 \ \rightarrow \ \overbrace{0011}^{3}\ \overbrace{0010\ 1100}^{x_0 x_1 \bar{x_2}}\ \overbrace{0101\ 0010}^{\bar{x_1} \bar{x_3} x_2}\ \overbrace{1000\ 0100}^{\bar{x_0} x_1} \ .$$

If $f = 1$ (i.e. all users are authorized), then there are no product term and the f is encoded as an l-bit zero.

An even more compact way is to instead represent f by using an l-digit number in base 3, whose j-th digit would determine each of the three possibilities: either u_j does not appear, or u_j appears, or $\bar{u_j}$ appears in f. Such a number is represented on at most $l \log_3(2) + 1$ bits, which represents a gain of about 20% compared to using $2l$ bits as above.

6 Comparison with other schemes

6.1 Comparison with Complete Subtree

In this section, we show that our BE scheme systematically outperforms the Complete Subtree in terms of bandwidth, i.e. that no matter the status of the users, the number of encryptions needed by our BE scheme is no greater than the one needed for CS. To do so, we recall the principle of the Complete Subtree: the n users are numbered from 0 to n and arranged as leaves of a complete binary tree T. Any set S of authorized users is then represented as a disjoint union of sets S_i, each of them being the set of leaves of a subtree of T of height h.

In the case where all users are authorized, the CS uses the key corresponding to the root and our BE scheme uses a key common to all users, so they both require one encryption. Let us now assume that at least one user is revoked, meaning that the sets S_i involved in the CS comes from strict subtrees of T. We now explain how every S_i can be viewed as a product term of f. For instance, if S_i corresponds to the leaves descending from the left (resp right) child of the root, then the set S_i is exactly the set of leaves whose labels are $< n/2$ (resp $\geq n/2$), it is therefore described by the product term $\bar{u_h}$ (resp u_h). Inductively, if the set S_i corresponds to the leaves of a subtree rooted at a node of depth d, then it will be described by a product term involving either u_i or $\bar{u_i}$ for $d \leq i \leq h$, the occurrence of one or the other option being determined by the critical path from the root of the subtree to the root of T.

This means that every subset cover used in the CS can be represented as a ΣΠ-form but since our BE scheme uses a ΣΠ-form whose number of terms is minimal, it necessarily involves no more encryptions than the CS. Based on the results that were proven for the CS in [NNL01], we get the following theorem as a corollary.

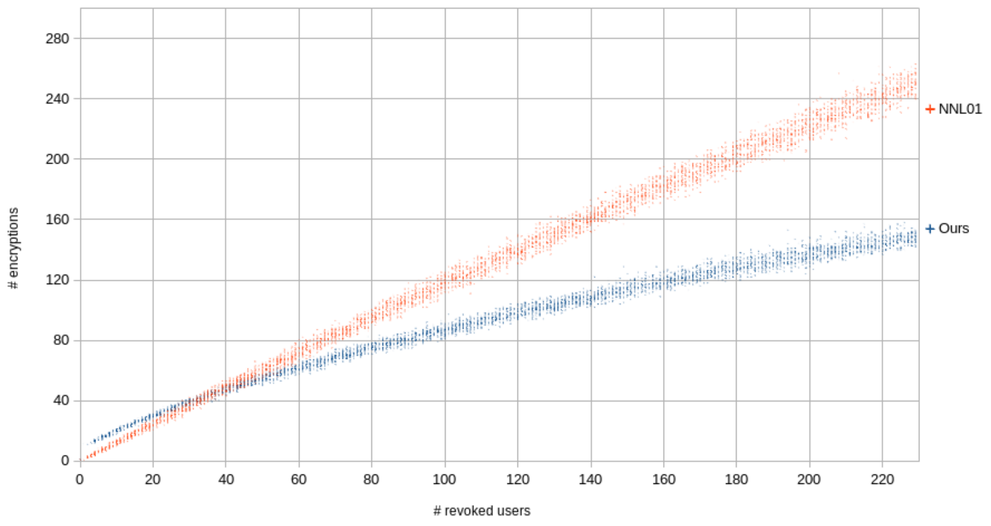
Theorem 1. *For a total of n users among which r users have been revoked, the bandwidth required by the* ΣΠ*BE scheme is in* $O(r\log(n/r))$.

In the next section, we present experimental results showing that the performance of our BE scheme is on average much better than this bound. We also remark that there is a worst case in which this bound is actually reached.

6.2 A practical comparison with Subset Difference

We compare our ΣΠBE scheme to the Subset Difference (SD) method proposed by Naor et al. [NNL01]. Let us consider a system of n users and $r \leq n$ revoked users. The SD scheme requires each user to store $\frac{1}{2}\log_2(n)^2$ keys. In comparison, our solution is heavier, since n keys are stored per user.

On the contrary, the encryption cost and bandwidth consumption, expressed in number of encryptions of the ephemeral key, behaves better in our scheme. It has been proved in [BS13] that $1.25r$ encryptions are needed in the average case and $2r$ in the worst case. However, this study is asymptotic and the scheme actually performs better in practice (e.g. $\approx 1.15r$ for $n = 2048$). In comparison, we cannot give an average case, as it relies on the average solution size of the NP-hard set cover problem, and even so, our approach described in Section 3 outputs a suboptimal solution for large n. Our comparison is therefore empirical, we choose $n = 2048$ (resp. $n = 4096$) and $r \in [\![0; 230]\!]$ (resp. $r \in [\![0; 520]\!]$). For each value of r, 20 tests are run with a set of r revoked users drawn pseudo-randomly. Since the Quine-McCluskey algorithm generally does not end in reasonable time for such parameters, we limit the ILP part to 30 seconds of computation on our laptop (i5-1135G7 with 16GB RAM). Depending on the context, it may be acceptable to run it for longer, resulting in a better solution and saving even more bandwidth. We



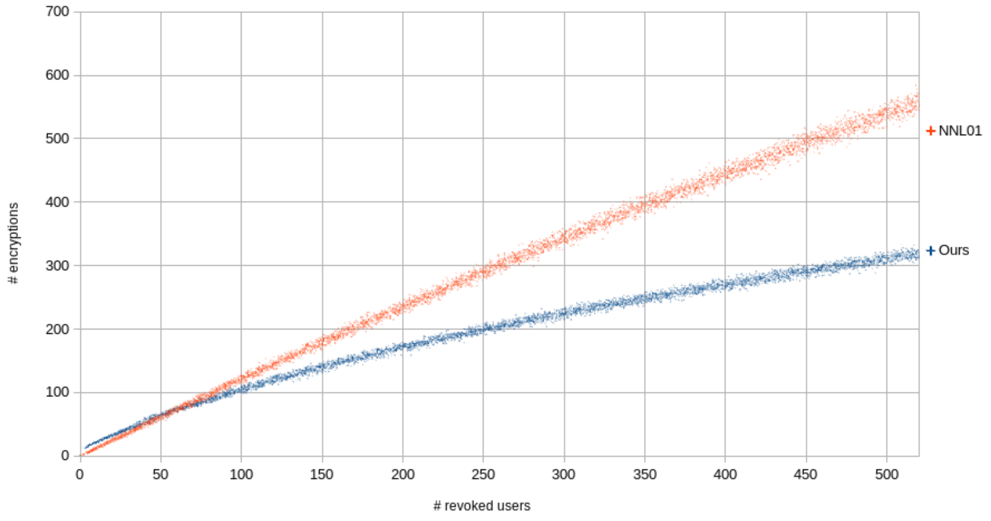


Figure 4: Performance comparison for $n = 2048$ (above) and $n = 4096$ (below)

discuss this question further. The experimental encryption cost for the ΣΠBE and SD schemes are given in Figure 4.

We observe that our solution outperforms the SD scheme except for very small values of r, which are unlikely to be the bottleneck anyway. For a fixed r, both schemes show a low variance in the encryption cost.

Concerning the computation time of the encryption process, our ΣΠBE scheme is obviously much heavier. It is exclusively due to the Quine-McCluskey algorithm. If the set of revoked users does not evolve too often (i.e. less than every minute), we stress that this part can be made offline, as it does not depend on the message. We believe most practical scenarios would fit in that description and recall that in the setting of BE, the power requirements are less stringent for the emitter than for the receivers. We also observed that very good solutions are found within the few first seconds of the ILP part. Our experiments suggest that running the ILP solver for 30 seconds (resp. 2 minutes and 20 minutes) improves the solution found after 5 seconds by less than 0.5% (resp. 1.7% and

4.2%) on average. Therefore, only 5 seconds of computation would have given roughly the same results as shown in Figure 4.

Beside the Quine-McCluskey part, our approach behaves slightly better, since no key derivation is necessary compared to the SD scheme. Similarly, the decryption process of ΣΠBE requires a few less computations thanks to the absence of key derivation.

6.3 Further comments on the comparison with SD

Contrary to the result proven for CS in Section 6.1 it is not true that our BE scheme systematically outperforms the Subset Difference. A typical example is the case where one single user is denied. With the SD framework, only one encryption is needed using the key associated to the whole tree minus the leaf corresponding to the revoked user. For our BE scheme, $\lambda = \log_2(n)$ encryptions are required. For instance is the revoked user is identified by the number 0, we have $f(u) = \bigwedge_{i=1}^{\lambda} u_i$ (u is authorized iff it has a 1 in its binary decomposition). This decomposition is already optimal because it is impossible to ensure that $u \neq 0$ without looking at all its bits.

On the other hand, however, there exist cases in which our BE scheme performs significantly better than the SD, in particular when a large number of users have been revoked. For instance, if one revokes all the users whose corresponding number is even then the SD will need exactly $n/2$ encryptions using, for every authorized leaf, the key corresponding to the difference of the subtree rooted at its parent minus the subtree rooted at its sibling. Using our BE scheme, such a denial is straightforward because the corresponding $f(u)$ is simply the value of the LSB of u, i.e. only one encryption is needed.

Following this reasoning, one gets the intuition that SD will perform well when denied users are clustered and poorly when they are interspersed among the authorized users, which should be a relatively frequent situation when users are randomly denied as in our experiments. Our BE scheme, on the other hand, is more adapted to handle this situation as it can easily cluster the denied users using a “comb” which groups users based on their remainders modulo a power of 2. We believe that this is the explanation behind our experimental results, but we reckon that, as we mentioned before, it is extremely challenging to provide a quantitative analysis of our BE scheme’s behaviour.

An important point that may come in mind is that, in practice, users are absolutely not revoked at random but based on features that can be used to arrange the tree used within the SD. For instance, if users are put in the tree based on their subscription date, their corresponding leaves will be close to one another and they will be roughly revoked at the same time, when the subscription expires. While this is a perfectly valid argument, we claim that it applies identically to our BE scheme (“close” users have consecutive numbers, so their most significant bits are identical) and that our BE scheme should actually benefit even more from this fact.

Indeed, in the case where several meaningful features need to be encoded, one could simply assign the first half of the identifier to one feature (e.g. geographic location, type of subscription) while the other half would just be a customer number related to the time of subscription. This would make it extremely convenient for our BE scheme to perform effective revocations based on these features or a combination thereof (for instance if there are various options for subscription duration, users will not be revoked based on their date of subscription but on a combination of both subscription date and subscription duration).

7 Future works

As briefly mentioned in Section 2.2, the ESPRESSO algorithm [BHMS84] is an interesting alternative to the Quine-McCluskey algorithm when dealing with a large number of users.

As for the Quine-McCluskey, it aims at representing a Boolean function as a small sum of products. However, the prime implicants are built following some customizable heuristic and is likely to output a suboptimal solution. We did not feel confident about using such a complex tool in a black box fashion without having some intuition about how it works and why it would be close or far from optimality. This question would need further investigation, but we point out that both the security and correctness of our BE scheme do not depend on the method used for minimizing the number of terms of the $\Sigma\Pi$-form of f, leaving room for numerous options including the pre-computation and storage of some $\Sigma\Pi$-forms in advance when revocations can be anticipated (e.g. expiration of subscriptions).

References

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 149–178. Springer, 2020. doi:10.1007/978-3-030-64375-1_6.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005. doi:10.1007/11535218_16.

[BHMS84] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*, volume 2 of *The Kluwer International Series in Engineering and Computer Science*. Springer, 1984. doi:10.1007/978-1-4613-2821-6.

[BS13] Sanjay Bhattacherjee and Palash Sarkar. Complete tree subset difference broadcast encryption scheme and its analysis. *Des. Codes Cryptogr.*, 66(1-3):335–362, 2013. doi:10.1007/s10623-012-9702-6.

[BS16] S. Bhattacherjee and P. Sarkar. Reducing communication overhead of the subset difference scheme. *IEEE Transactions on Computers*, 65(08):2575–2587, aug 2016. doi:10.1109/TC.2015.2485231.

[CFN61] A. Cobham, R. Fridshal, and J. H. North. An application of linear programming to the minimization of boolean functions. In *2nd Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1961)*, pages 3–9, 1961. doi:10.1109/FOCS.1961.5.

[CM78] Ashok K. Chandra and George Markowsky. On the number of prime implicants. *Discret. Math.*, 24(1):7–11, 1978. doi:10.1016/0012-365X(78)90168-1.

[DF03] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 2003. doi:10.1007/3-540-36288-6_8.

[DGB12] Renaud Dubois, Aurore Guillevic, and Marine Sengelin Le Breton. Improved broadcast encryption scheme with constant-size ciphertext. In Michel Abdalla and Tanja Lange, editors, *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, volume 7708 of *Lecture Notes in Computer Science*, pages 196–202. Springer, 2012. doi:10.1007/978-3-642-36334-4_12.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993. doi:10.1007/3-540-48329-2_40.

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002. doi:10.1007/3-540-45708-9_4.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014. URL: https://www.crcpress.com/Introduction-to-Modern-Cryptography-Second-Edition/Katz-Lindell/p/book/9781466570269.

[KRS99] Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 609–623. Springer, 1999. doi:10.1007/3-540-48405-1_38.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001. doi:10.1007/3-540-44647-8_3.

[Pet56] Stanley R Petrick. A direct determination of the irredundant forms of a boolean function from the set of prime implicants. *Air Force Cambridge Res. Center Tech. Report*, pages 56–110, 1956. URL: https://books.google.fr/books?id=mgKnAQAACAAJ.

[PPSS13] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. *Int. J. Inf. Sec.*, 12(4):251–265, 2013. doi:10.1007/s10207-013-0190-0.

[SDSP23] Vikas Srivastava, Sumit Kumar Debnath, Pantelimon Stanica, and Saibal Kumar Pal. A multivariate identity-based broadcast encryption with applications to the internet of things. *Adv. Math. Commun.*, 17(6):1302–1313, 2023. doi:10.3934/amc.2021050.

[TT01] Wen-Guey Tzeng and Zhi-Jia Tzeng. A public-key traitor tracing scheme with revocation using dynamic shares. In Kwangjo Kim, editor, *Public Key Cryptography, 4th International Workshop on Practice and Theory in Public*

Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings, volume 1992 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2001. doi:10.1007/3-540-44586-2_16.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 217–241. Springer, 2022. doi:10.1007/978-3-031-07085-3_8.