Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack*

Yevgeniy Dodis and Nelly Fazio

Computer Science Department, New York University, USA
{dodis,fazio}@cs.nyu.edu

Abstract. A (public key) Trace and Revoke Scheme combines the functionality of broadcast encryption with the capability of traitor tracing. Specifically, (1) a trusted center publishes a single public key and distributes individual secret keys to the users of the system; (2) anybody can encrypt a message so that all but a specified subset of "revoked" users can decrypt the resulting ciphertext; and (3) if a (small) group of users combine their secret keys to produce a "pirate decoder", the center can trace at least one of the "traitors" given access to this decoder.
We construct the first *chosen ciphertext* (CCA2) secure Trace and Revoke Scheme based on the DDH assumption. Our scheme is also the first *adaptively secure* scheme, allowing the adversary to corrupt players at any point during execution, while prior works (e.g., [14, 16]) only achieves a very weak form of non-adaptive security even against chosen plaintext attacks.
Of independent interest, we present a slightly simpler construction that shows a "natural separation" between the classical notion of CCA2-security and the recently proposed [15, 1] relaxed notion of gCCA2-security.

1 Introduction

A *broadcast encryption* scheme allows the sender to securely distribute data to a dynamically changing set of users over an insecure channel. Namely, it should be possible to selectively exclude (i.e., "revoke") a certain subset of users from receiving the data. In particular, each user should receive an individualized decoder which decrypts only the ciphertexts intended for the given user. Broadcast encryption has numerous applications, including pay-TV systems, distribution of copyrighted material, streaming audio/video and many others.

The formal study of broadcast encryption was initiated by Fiat and Naor [8], who showed a scheme with message overhead roughly $O(z^2 \log^2 z \log N)$, where z is the maximum number of excluded users (so called *revocation threshold*) and N is the total number of users. Subsequent works include [12, 10], and, more recently, [13, 11].

Most of the above works primarily concentrate on the centralized setting, where only the trusted center (the entity who generates all the secret keys) can

* This proceedings version lacks most proof details; for a complete version see [7].

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 100–115, 2003.

send messages to the receivers. In the *public key* setting, studied in this paper, the center also prepares a fixed public key which allows any entity to play the role of the sender. The public key setting also allows the center to store secret keys in a secure place (e.g. off-line), and only use them when a new user join the system.

The only known public key Broadcast Encryption Schemes have been constructed by [14, 16] based on the DDH assumption, and achieve public key and message overhead $O(z)$. In fact, these schemes are essentially identical: in the following we will refer to the work of [16], who emphasize more the public key nature of their scheme.

Despite providing a simple and elegant scheme, the work of [16] has several noticeable shortcomings. First, the given (informal) notion of security makes little sense in a revocation setting. Indeed, to show the "security" of revocation, [16] show the following two claims: (1) the scheme is semantically secure when no users are revoked; (2) no set of z a-priori fixed users can compute the secret key of another user. Clearly, these properties do not imply the security notion we really care about and which informally states: (3) if the adversary controls some set $\mathcal{R}$ of up to z *revoked* users, then the scheme remains semantically secure. Actually, the scheme of [16] can be shown to satisfy (3) only when the set $\mathcal{R}$ is chosen by the adversary *non-adaptively*, and in fact only if it is chosen before the adversary learns the public key. Such weak non-adaptive security is clearly insufficient for realistic usages of a public key revocation scheme.

Most importantly, the extended scheme of [16] is proven to be CCA2-secure when none of the users is corrupted, but stops being such the moment just a single user is corrupted, even if this user is immediately revoked for the rest of the protocol. Again, this is too weak — the scheme should remain CCA2-secure *even after many users have been revoked*. As we will see, achieving this strong type of security is very non-trivial, and requires a much more involved scheme than the one proposed by [16].

Our Contributions. We construct the first *adaptive chosen ciphertext* (CCA2) secure public key Broadcast Encryption Scheme under the DDH assumption (with no random oracles). We remark that no CCA2 schemes were known even in the symmetric setting. Moreover, it doesn't seem obvious how to extend current symmetric schemes (e.g. [13]) to meet the CCA2 notion. Our public key scheme is based on the regular Cramer-Shoup encryption [5, 6], but our extension is non-trivial, as we have to resolve some difficulties inherent to Broadcast Encryption. Furthermore, we introduce for the first time a precise formalization of an appropriate notion of adaptive security for Broadcast Encryption (for both the CPA and the CCA2 setting). We also extend the CPA scheme of [16] to achieve such higher level of security, while maintaining essentially the same efficiency in all the parameters (up to a factor of 2).

Of independent interest, we also provide another scheme achieving a slightly weaker (but still very strong) notion of *generalized* CCA2 security (gCCA2) [15, 1]. As argued in [1], the gCCA2 security is much more robust to syntactic changes, while still sufficient for all known uses of CCA2-security. Interestingly,

all the examples separating CCA2 and gCCA2-secure encryption were "artificial" in a sense that they made a more complicated scheme from an already existing CCA2-secure encryption. Our work shows the first "natural" separation, but for the *broadcast* public key encryption.

A Note on Traitor Tracing. As first explicitly noticed by Gafni et al. [9], Broadcast Encryption is most useful when combined with a *Traitor Tracing* mechanism [4] by which the center can extract the identity of (at least one) "pirate" from any illegal decoder produced combining decryption equipments of a group of legal members (the "traitors"). By slightly modifying standard tracing algorithms from previous weaker schemes (e.g. [14, 16]), tracing algorithms can be added to our schemes, thus yielding fully functional *Trace and Revoke* schemes [14]. However, due to space limitations we omit the tracing part, focusing only on Broadcast Encryption (i.e. revocation), which is also the main novelty of this paper.

2 Notations and Basic Facts

Lagrange Interpolation in the Exponent. Let q be a prime and $f(x)$ a polynomial of degree z over $\mathbb{Z}_q$; let $j_0, \ldots, j_z$ be distinct elements of $\mathbb{Z}_q$, and let $f_0 = f(j_0), \ldots, f_z = f(j_z)$. Using Lagrange Interpolation, we can express the polynomial as $f(x) = \sum_{t=0}^{z}(f_t \cdot \lambda_t(x))$, where $\lambda_t(x) = \prod_{0 \le i \ne t \le z} \frac{j_i - x}{j_i - j_t}$, $t \in [0, z]$. Now, define the Lagrange Interpolation Operator as: $\mathsf{LI}(j_0, \ldots, j_z; f_0, \ldots, f_z)(x) \doteq \sum_{t=0}^{z}(f_t \cdot \lambda_t(x))$.

Now, consider any cyclic group $\mathbb{G}$ of order q and a generator g of $\mathbb{G}$. For any distinct $j_0, \ldots, j_z \in \mathbb{Z}_q$ and (non necessarily distinct) $v_0, \ldots, v_z \in \mathbb{G}$, define the Lagrange Interpolation Operator in the Exponent as: $\mathsf{EXP\text{-}LI}(j_0, \ldots, j_z; v_0, \ldots, v_z)(x) \doteq g^{\mathsf{LI}(j_0,\ldots,j_z;\log_g v_0,\ldots,\log_g v_z)(x)} = \prod_{t=0}^{z} g^{(\log_g v_t \cdot \lambda_t(x))} = \prod_{t=0}^{z} v_t^{\lambda_t(x)}$. The last expression shows that the function $\mathsf{EXP\text{-}LI}$ is poly-time computable, despite being defined in terms of discrete logarithms (which are usually hard to compute). We also remark on another useful property of the above operator: $\mathsf{EXP\text{-}LI}(j_0, \ldots, j_z; v_0^r, \ldots, v_z^r)(x) = [\mathsf{EXP\text{-}LI}(j_0, \ldots, j_z; v_0, \ldots, v_z)(x)]^r$. In what follows, we will refer to a function of the form $g^{f(x)}$, where $f(x)$ is a polynomial, as an EXP-polynomial.

DDH Assumption. The security of our schemes will rely on the Decisional Diffie-Hellman (DDH) Assumption in the group $\mathbb{G}$: namely, it is computationally hard to distinguish a random tuple (g_1, g_2, u_1, u_2) of four independent elements in $\mathbb{G}$ from a random tuple satisfying $\log_{g_1} u_1 = \log_{g_2} u_2$ (for a survey, see [3]).

A Probabilistic Lemma. The following useful lemma states that to estimate the difference between two related experiments U_1 and U_2, it is sufficient to bound the probability of some event F which "subsumes" all the differences between the experiments.

Lemma 1. *If U_1, U_2 and F are events such that $(U_1 \wedge \neg F)$ and $(U_2 \wedge \neg F)$ are equivalent events, then* $\left|\Pr[U_1] - \Pr[U_2]\right| \le \Pr[F]$.

3 Definition of Broadcast Encryption Scheme

Since a public-key broadcast encryption is typically used by encrypting a session key s for the privileged users (this encryption is called the *enabling block*), and then symmetrically encrypting the "actual" message with s, we will often say that the goal of a Broadcast Encryption Scheme is to *encapsulate* [6] a session key s, rather than to encrypt a message M.

Definition 1 (Broadcast Encryption Scheme).
A Broadcast Encryption Scheme BE *is a 4-tuple of* poly*-time algorithms* (KeyGen, Reg, Enc, Dec)*, where:*

- KeyGen*, the* key generation algorithm*, is a probabilistic algorithm used by the center to set up all the parameters of the scheme.* KeyGen *takes as input a security parameter* 1^λ *and a revocation threshold* z *(i.e. the maximum number of users that can be revoked) and generates the public key* PK *and the master secret key* SK_{BE}.
- Reg*, the* registration algorithm*, is a probabilistic algorithm used by the center to compute the secret initialization data needed to construct a new decoder each time a new user subscribes to the system.* Reg *receives as input the master key* SK_{BE} *and a (new) index* i *associated with the user; it returns the user's secret key* SK_i.
- Enc*, the* encryption algorithm*, is a probabilistic algorithm used to encapsulate a given session key* s *within an enabling block* $\mathcal{T}$*.* Enc *takes as input the public key* PK*, the session key* s *and a set* $\mathcal{R}$ *of revoked users (with* $|\mathcal{R}| \leq z$*) and returns the enabling block* $\mathcal{T}$.
- Dec*, the* decryption algorithm*, is a deterministic algorithm that takes as input the secret key* SK_i *of user* i *and the enabling block* $\mathcal{T}$ *and returns the session key* s *that was encapsulated within* $\mathcal{T}$ *if* i *was a legitimate user when* $\mathcal{T}$ *was constructed, or the special symbol* $\perp$.

3.1 Security of Revocation

Intuitively, we would like to say that even if a malicious *adversary* $\mathcal{A}$ learns the secret keys of at most z users, and these users are later revoked, then subsequent broadcasts do not leak any information to such adversary. The security threat posed by such adversary is usually referred to as *Chosen Plaintext Attack* (CPA), and a Broadcast Encryption Scheme withstanding such an attack is said to be z*-Resilient against* CPA.

For most realistic usages, however, it is more appropriate to consider the stronger *Chosen Ciphertext Attack* (CCA2), in which the adversary is allowed to "play" with the decryption machinery as she wishes, subject only to the condition that she doesn't ask about enabling blocks closely related to the "challenge" $\mathcal{T}^*$. In formalizing the notion of "close relationship", the usual treatment is to impose a minimal restriction to the adversary, forbidding just direct decryption of the challenge itself. As noted in [15, 1], such a mild constraint restricts too much the class of schemes that can be proven secure, excluding even schemes that

ought to be considered secure under a more intuitive notion. For this reason, it seems more reasonable to consider a variant of the CCA2, to which we will refer to as *Generalized Chosen Ciphertext Attack* (gCCA2), following the terminology introduced in [1].

In a Generalized Chosen Ciphertext Attack, the set of enabling blocks the adversary is forbidden to ask about is defined in term of an efficiently computable equivalence relation $\Re(\cdot,\cdot)$. In fact, in the case of a broadcast (as opposed to ordinary) encryption, there is no unique decryption machinery, since the decryption algorithm can be used with the secret key of any legitimate user. Hence, we need to consider a *family* of efficient equivalence relations $\{\Re_i(\cdot,\cdot)\}$, one for each user i. As in the regular case [1], the equivalence relation $\Re_i(\cdot,\cdot)$ corresponding to each user i needs to be *i-decryption-respecting*: equivalent enabling blocks under $\Re_i$ are guaranteed to have exactly the same decryption according to the secret data of user i. Finally, this family should form an *explicit* parameter of the scheme (i.e., one has to specify some decryption-respecting family $\{\Re_i\}$ when proving the gCCA2-security of a given scheme).

Formal Model. We now formalize the above attack scenarios, starting with the CPA.

First, $(PK, SK_{\mathsf{BE}}) \leftarrow \mathsf{BE.KeyGen}(1^\lambda, z)$ is run and the adversary $\mathcal{A}$ is given the public key PK. Then $\mathcal{A}$ enters the user corruption stage, where she is given oracle access to the *User Corruption Oracle* $\mathsf{Cor}_{SK_{\mathsf{BE}}}(\cdot)$. This oracle receives as input the index i of the user to be corrupted, computes $SK_i \leftarrow \mathsf{BE.Reg}(SK_{\mathsf{BE}}, i)$ and returns the user's secret key SK_i. This oracle can be called *adaptively* for at most z times. Let us say that at the end of this stage the set $\mathcal{R}$ of at most z users is corrupted.

In the second stage, a random bit σ is chosen, and $\mathcal{A}$ can query the *Encryption Oracle* (sometimes also called the *left-or-right* oracle) $\mathcal{E}_{PK,\mathcal{R},\sigma}(\cdot,\cdot)$ on any pair of session keys s_0, s_1.[1] This oracle returns $\mathsf{Enc}(PK, s_\sigma, \mathcal{R})$. Without loss of generality (see [2]), we can assume that the encryption oracle is called exactly once, and returns to $\mathcal{A}$ the *challenge enabling block* $\mathcal{T}^*$. At the end of this second stage, $\mathcal{A}$ outputs a bit σ^* which she thinks is equal to σ. Define the *advantage* of $\mathcal{A}$ as $\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{BE},\mathcal{A}}(\lambda) \doteq |\Pr(\sigma^* = \sigma) - \frac{1}{2}|$.

Additionally, in the case of a Chosen Ciphertext Attack (generalized or not), $\mathcal{A}$ has also access to a *Decryption Oracle* $\mathcal{D}_{SK_{\mathsf{BE}}}(\cdot,\cdot)$, which she can query on any pair $\langle i, \mathcal{T}\rangle$, where i is the index of some user and $\mathcal{T}$ is any enabling block of her choice. $\mathcal{A}$ can call this oracle at any point during the execution (i.e., both in the first and in the second stage, arbitrarily interleaved with her other oracle calls). To prevent the adversary from directly decrypting her challenge $\mathcal{T}^*$, the decryption oracle first checks whether $\Re_i(\mathcal{T}, \mathcal{T}^*)$ holds[2]: if so, $\mathcal{D}$ outputs $\perp$; if not, $\mathcal{D}$ computes $SK_i \leftarrow \mathsf{BE.Reg}(SK_{\mathsf{BE}}, i)$ and uses it to output $\mathsf{BE.Dec}(i, \mathcal{T})$.

[1] For the sake of generality, we could have allowed $\mathcal{A}$ to interleave the calls to $\mathsf{Cor}_{SK_{\mathsf{BE}}}(i)$ and $\mathcal{E}_{PK,\mathcal{R},\sigma}$ (where $\mathcal{A}$ can choose any i's and $\mathcal{R}$'s only subject to $i \notin \mathcal{R}$). However, this clumsier definition is easily seen to be equivalent to the one we present.

[2] This preliminary check applies to the standard Chosen Ciphertext Attack as well, which corresponds to all the $\Re_i$'s being the equality relation.

As before, we define the corresponding advantages $\mathsf{Adv}^{\mathsf{gCCA2}}_{\mathsf{BE},\mathcal{A}}(\lambda)$ and $\mathsf{Adv}^{\mathsf{CCA2}}_{\mathsf{BE},\mathcal{A}}(\lambda)$.

Definition 2 (z-Resilience of a Broadcast Encryption Scheme). *Let $\mu \in \{\mathsf{CPA}, \mathsf{gCCA2}, \mathsf{CCA2}\}$. We say that a Broadcast Encryption Scheme BE is z-resilient against a μ-type attack if the advantage, $\mathsf{Adv}^{\mu}_{\mathsf{BE},\mathcal{A}}(\lambda)$, of any probabilistic poly-time algorithm $\mathcal{A}$ is a negligible function of λ.*

4 Revocation Schemes

In this section, we present three Broadcast Encryption Schemes, achieving z-resilience in an adaptive setting for the case of a CPA, $\mathsf{gCCA2}$ and $\mathsf{CCA2}$ attack respectively. Subsequent schemes build on the previous one, in a incremental way, so that it is possible to obtain increasing security at the cost of a slight efficiency loss.

Considering the subtlety of the arguments, our proofs follow the structural approach advocated in [6] defining a sequence of attack games $\mathbf{G}_0$, $\mathbf{G}_1$, ..., all operating over the same underlying probability space. Starting from the actual adversarial game $\mathbf{G}_0$, we incrementally make slight modifications to the behavior of the oracles, thus changing the way the adversary's view is computed, while maintaining the view's distributions indistinguishable among the games. While this structural approach takes more space to write down, it is much less error-prone and much more understandable than a slicker "direct argument" (e.g., compare [5] and [6]).

4.1 z-Resilience against CPA Attack

As a warm-up before addressing the more challenging case of chosen ciphertext security, we describe a simpler CPA-secure scheme. Our scheme naturally builds upon previous works [14, 16], but achieves a much more appropriate notion of *adaptive* security, which those previous schemes do not enjoy.

The Key Generation Algorithm. The first step in the key generation algorithm $\mathsf{KeyGen}(1^\lambda, z)$ is to define a group $\mathbb{G}$ of order q, for a random λ-bit-long prime q such that $p = 2q + 1$ is also prime, in which the DDH assumption is believed to hold. This is accomplished selecting a random prime q with the above two properties and a random element g_1 of order q modulo p: the group $\mathbb{G}$ is then set to be the subgroup of $\mathbb{Z}_p^*$ generated by g_1, i.e. $\mathbb{G} = \{g_1^i \bmod p : i \in \mathbb{Z}_q\} \subset \mathbb{Z}_p^*$. A random $w \leftarrow_R \mathbb{Z}_q$ is then chosen and used to compute $g_2 = g_1^w$. (In what follows, all computations are mod q in the exponent, and mod p elsewhere.) Then, the key generation algorithm selects two random z-degree polynomials[3] $Z_1(\xi)$ and $Z_2(\xi)$ over $\mathbb{Z}_q$, and computes the values: $h_0 \doteq g_1^{Z_{1,0}} \cdot g_2^{Z_{2,0}}, \ldots, h_z \doteq g_1^{Z_{1,z}} \cdot g_2^{Z_{2,z}}$. Finally, the pair (PK, SK_{BE}) is given in output, where $PK \doteq \langle g_1, g_2, h_0, \ldots, h_z \rangle$ and $SK_{\mathsf{BE}} \doteq \langle Z_1, Z_2 \rangle$.

[3] For conciseness, we will use the following notation: $Z_{1,i} \doteq Z_1(i)$ and $Z_{2,i} \doteq Z_2(i)$.

Encryption algorithm $\mathsf{Enc}(PK, s, \mathcal{R})$	Decryption algorithm $\mathsf{Dec}(i, \mathcal{T})$
$E1.\ r_1 \leftarrow_R \mathbb{Z}_q$ $E2.\ u_1 \leftarrow g_1^{r_1}$ $E3.\ u_2 \leftarrow g_2^{r_1}$ $E4.\ H_t \leftarrow h_t^{r_1},\ t \in [0, z]$ $E5.$ **for** $t = 1$ **to** z **do** $\quad H_{j_t} \leftarrow \mathsf{EXP\text{-}LI}(0, .., z; H_0, .., H_z)(j_t)$ $E6.$ **end for** $E7.\ S \leftarrow s \cdot H_0$ $E8.\ \mathcal{T} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z})\rangle$	$D1.\ H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$ $D2.\ s \leftarrow \frac{S}{\mathsf{EXP\text{-}LI}(j_1,..,j_z,i;H_{j_1},..,H_{j_z},H_i)(0)}$

Fig. 1. Encryption and decryption algorithms for the CPA scheme

The Registration Algorithm. Each time a new user $i > z$ (in all our schemes, we reserve the indices $[0, z]$ for "special purposes") decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i \doteq \langle i, Z_{1,i}, Z_{2,i}\rangle$.

The Encryption Algorithm. The encryption algorithm Enc is given in Fig. 1. It receives as input the public key PK, a session key s and a set $\mathcal{R} = \{j_1, \ldots, j_z\}$ of revoked users and returns the enabling block $\mathcal{T}$. If there are less than z revoked users, the remaining indices are set to $1 \ldots (z - |\mathcal{R}|)$, which are never given to any "real" user.

The Decryption Algorithm. To recover the session key embedded in the enabling block $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z})\rangle$ a legitimate user i can proceed as in Fig. 1. If i is a revoked user (i.e. $i \in \{j_1, \ldots, j_z\}$), the algorithm fails in step $D2$, since the interpolation points $j_1, \ldots, j_z, i$ are not pairwise distinct.

Security. As shown in the theorem below, the z-resilience of the above scheme relies on the Decisional Diffie-Hellman (DDH) assumption.

Theorem 1. *If the* DDH *problem is hard in* $\mathbb{G}$*, then the above Broadcast Encryption Scheme is z-resilient against chosen plaintext attacks.*

Proof. We define a sequence of "indistinguishable" games $\mathbf{G}_0, \ldots$, where $\mathbf{G}_0$ is the original game, and the last game clearly gives no advantage to the adversary.

Game $\mathbf{G}_0$. In game $\mathbf{G}_0$, $\mathcal{A}$ receives the public key PK and adaptively queries the corruption oracle $\mathsf{Cor}_{SK_{\mathsf{BE}}}(\cdot)$. Then, she queries the encryption oracle $\mathcal{E}_{PK,\mathcal{R},\sigma}(\cdot,\cdot)$ on (s_0, s_1), where $\mathcal{R}$ must contain all users that $\mathcal{A}$ compromised through the oracle $\mathsf{Cor}_{SK_{\mathsf{BE}}}(\cdot)$; $\mathcal{A}$ receives back the enabling block $\mathcal{T}^*$. At this point, $\mathcal{A}$ outputs her guess $\sigma^* \in \{0, 1\}$. Let T_0 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_0$.

Game $\mathbf{G}_1$. Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that step $E4$ of the encryption algorithm in Fig. 1, is changed in: $\boxed{E4'.\ H_t \leftarrow u_1^{Z_{1,t}} \cdot u_2^{Z_{2,t}},\ t \in [0, z]}$. By the properties of the Lagrange Interpolation in the Exponent, it is clear that step $E4'$ computes the same values $\{H_t\}_{t=0}^{z}$ as step $E4$. The point of this change is just to make explicit any functional dependency of the above quantities on u_1 and u_2. Let T_1 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_1$; clearly, it holds that $\Pr[T_0] = \Pr[T_1]$.

Game $\mathbf{G}_2$. To turn game $\mathbf{G}_1$ into game $\mathbf{G}_2$ we again modify the encryption oracle used in $\mathbf{G}_1$, replacing step $E1$ with $\boxed{E1'.\ r_1 \leftarrow_R \mathbb{Z}_q,\ r_2 \leftarrow_R \mathbb{Z}_q \setminus \{r_1\}}$ and step $E3$ with $\boxed{E3'.\ u_2 \leftarrow g_2^{r_2}}$. Let T_2 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_2$. Notice that while in $\mathbf{G}_1$ the values u_1 and u_2 are obtained using the same value r_1, in $\mathbf{G}_2$ they are independent subject to $r_1 \neq r_2$. Therefore, using a standard reduction argument, any non-negligible difference in behavior between $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a PPT algorithm $\mathcal{A}_1$ that is able to distinguish Diffie-Hellman tuples from totally random tuples with non negligible advantage. Hence, $\left|\Pr[T_2] - \Pr[T_1]\right| \leq \epsilon_1$ for some negligible ϵ_1.

Game $\mathbf{G}_3$. To define game $\mathbf{G}_3$, we make another change to the encryption oracle in game $\mathbf{G}_2$, substituting step $E6$ with: $\boxed{E6'.\ e \leftarrow_R \mathbb{Z}_q,\ S \leftarrow g_1^e}$. Let T_3 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_3$. Because of this last change, the challenge no longer contains σ, nor does any other information in the adversary's view; therefore, we have that $\Pr[T_3] = \frac{1}{2}$. Moreover, we can prove (see Lemma 3 in [7]), that the adversary has the same chances to guess σ in both games $\mathbf{G}_2$ and $\mathbf{G}_3$, i.e. $\Pr[T_3] = \Pr[T_2]$.

Finally, combining all the intermediate results together, we can conclude that adversary $\mathcal{A}$'s advantage is negligible; more precisely: $\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \epsilon_1$.

4.2 z-Resilience against gCCA2 Attack

Once we have constructed a Broadcast Encryption Scheme z-resilient against CPA attacks, it is natural to try to devise an extension achieving adaptive chosen ciphertext security. This was already attempted by [16], but they do not elaborate (neither formally nor informally) on what an "adaptive chosen ciphertext attack" on a Broadcast Encryption Scheme exactly is. As a consequence, in their security theorem (Theorem 3 of [16]), the authors only show the security of their scheme against an adversary that does not participate to the system, while their scheme is certainly not CCA2-secure with respect to even a single malicious revoked user.

To achieve CCA2-security, we will first try to apply the standard technique of [5, 6] to the scheme presented in Section 4.1. Unfortunately, this natural approach does *not* completely solve the CCA2 problem; still it leads us to an interesting scheme that achieves the (sligthly weaker) notion of generalized chosen ciphertext security.

The Key Generation Algorithm. As before, the first task of the key generation algorithm is to select a random group $\mathbb{G} \subset \mathbb{Z}_p^*$ of prime order q and two random generators $g_1, g_2 \in \mathbb{G}$. Then, KeyGen selects six random z-degree polynomials[4] $X_1(\xi)$, $X_2(\xi)$, $Y_1(\xi)$, $Y_2(\xi)$, $Z_1(\xi)$ and $Z_2(\xi)$ over $\mathbb{Z}_q$, and computes the values $c_t \doteq g_1^{X_{1,t}} \cdot g_2^{X_{2,t}}$, $d_t \doteq g_1^{Y_{1,t}} \cdot g_2^{Y_{2,t}}$ and $h_t \doteq g_1^{Z_{1,t}} \cdot g_2^{Z_{2,t}}$, for $t \in [0, z]$.

Finally, KeyGen chooses at random a hash function $\mathcal{H}$ from a family $\mathcal{F}$ of collision resistant hash functions,[5] and outputs the pair (PK, SK_{BE}), where

[4] For conciseness, we will use the following notation: $X_{1,i} \doteq X_1(i), X_{2,i} \doteq X_2(i), Y_{1,i} \doteq Y_1(i), Y_{2,i} \doteq Y_2(i), Z_{1,i} \doteq Z_1(i)$ and $Z_{2,i} \doteq Z_2(i)$.

Encryption algorithm $\mathsf{Enc}(PK, s, \mathcal{R})$	Decryption algorithm $\mathsf{Dec}(i, \mathcal{T})$
$E1.\ r_1 \leftarrow_R \mathbb{Z}_q$	$D1.\ \alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}))$
$E2.\ u_1 \leftarrow g_1^{r_1}$	$D2.\ \bar{v}_i \leftarrow u_1^{X_{1,i}+Y_{1,i}\alpha} \cdot u_2^{X_{2,i}+Y_{2,i}\alpha}$
$E3.\ u_2 \leftarrow g_2^{r_1}$	$D3.\ v_i \leftarrow \mathsf{EXP\text{-}LI}(0, .., z; v_0, .., v_z)(i)$
$E4.\ H_t \leftarrow h_t^{r_1},\ t \in [0, z]$	$D4.$ **if** $v_i = \bar{v}_i$ **then**
$E5.$ **for** $t = 1$ **to** z **do** $H_{j_t} \leftarrow \mathsf{EXP\text{-}LI}(0, .., z; H_0, .., H_z)(j_t)$	$D5.\quad H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$
$E6.$ **end for**	$D6.\quad s \leftarrow \frac{S}{\mathsf{EXP\text{-}LI}(j_1,..,j_z,i;H_{j_1},..,H_{j_z},H_i)(0)}$
$E7.\ S \leftarrow s \cdot H_0$	$D7.\quad$ **return** s
$E8.\ \alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}))$	$D8.$ **else return** $\perp$
$E9.\ v_t \leftarrow c_t^{r_1} \cdot d_t^{r_1\alpha},\ t \in [0, z]$	$D9.$ **end if**
$E10.\ \mathcal{T} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}), v_0, .., v_z \rangle$	

Fig. 2. Encryption and decryption algorithms for the gCCA2 scheme

$PK \doteq \langle g_1, g_2, c_0, \ldots, c_z, d_0, \ldots, d_z, h_0, \ldots, h_z, \mathcal{H} \rangle$ and $SK_{\mathsf{BE}} \doteq \langle X_1, X_2, Y_1, Y_2, Z_1, Z_2 \rangle$.

The Registration Algorithm. Each time a new user $i > z$ decides to subscribe to the system, the center provides him with a decoder box containing the secret key $SK_i \doteq \langle i, X_{1,i}, X_{2,i}, Y_{1,i}, Y_{2,i}, Z_{1,i}, Z_{2,i} \rangle$.

The Encryption Algorithm. Using the idea of [5, 6], in order to obtain non-malleable ciphertexts, we "tag" each encrypted message so that it can be verified before proceeding with the actual decryption. In the broadcast encryption scenario, where each user has a different decryption key, the tag cannot be a single point — we need to distribute an entire EXP-polynomial $\mathcal{V}(x)$. This is accomplished appending $z + 1$ tags to the ciphertext: each user i first computes the tag v_i using his private key and then verifies the validity of the ciphertext by checking the interpolation of the $z + 1$ values in point i against its v_i.

The encryption algorithm Enc receives as input the public key PK, the session key s to be embedded within the enabling block and a set $\mathcal{R} = \{j_1, \ldots, j_z\}$ of revoked users. It proceeds as described in Fig. 2, and finally it outputs $\mathcal{T}$.

The Decryption Algorithm. To recover the session key embedded in the enabling block $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v_0, \ldots, v_z \rangle$, a legitimate user i can proceed as in Fig. 2. If i is a revoked user, the algorithm fails in step $D6$, since the interpolation points $j_1, \ldots, j_z, i$ are not pairwise distinct.

Security. As mentioned above, the presence of many decryption keys leads to the use of an EXP-polynomial $\mathcal{V}(x)$ to tag the encryption of the message. This in turn makes the ciphertext malleable: since each user i can verify the value of $\mathcal{V}(x)$ only in one point, the adversary can modify the v_j's values and construct a different EXP-polynomial $\mathcal{V}'(x)$ intersecting $\mathcal{V}(x)$ at point i — thus fooling user i to accept as valid a corrupted ciphertext. In the next section we show a non-trivial solution to this problem; here, we assess the z-resilience of

[5] Recall, it is hard to find $x \neq y$ such that $\mathcal{H}(x) = \mathcal{H}(y)$ for a random member $\mathcal{H}$ of $\mathcal{F}$.

the Broadcast Encryption Scheme presented above against a gCCA2 attack. As already discussed in Section 3.1, to this aim it is necessary to introduce a family of equivalence relations $\{\Re_i\}$: intuitively, two ciphertexts $\mathcal{T}$ and $\mathcal{T}'$ are equivalent for user i if they have the same "data" components, and the tag "relevant to user i" is correctly verified, i.e. $v_i = v'_i$ (even though other "irrelevant" tags could be different). Clearly, this relation is efficiently computable and i-decryption-respecting.

Definition 3 (Equivalence Relation).
Consider the EXP-polynomials $\mathcal{V}(x) = \mathit{EXP\text{-}LI}(0, \ldots, z; v_0, \ldots, v_z)(x)$ *and* $\mathcal{V}'(x) = \mathit{EXP\text{-}LI}(0, \ldots, z; v'_0, \ldots, v'_z)(x)$. *Given a user* i, *and the two enabling blocks* $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v_0, \ldots, v_z \rangle$ *and* $\mathcal{T}' = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v'_0, \ldots, v'_z \rangle$, *we say that* $\mathcal{T}$ *is* equivalent *to* $\mathcal{T}'$ *with respect to user* i, *and we write* $\Re_i(\mathcal{T}, \mathcal{T}')$, *if the two EXP-polynomials* $\mathcal{V}(x)$ *and* $\mathcal{V}'(x)$ *intersect at point* i, *i.e.* $v_i = \mathcal{V}(i) = \mathcal{V}'(i) = v'_i$.

Theorem 2. *If the* DDH *Problem is hard in* $\mathbb{G}$ *and* $\mathcal{H}$ *is chosen from a collision-resistant hash functions family* $\mathcal{F}$, *then the above Broadcast Encryption Scheme is* z*-resilient against generalized chosen ciphertext attacks, under the family of equivalence relations* $\{\Re_i\}$.

Proof. To prove this theorem, we pursue the same approach as in the proof of Theorem 1, where the starting scenario of the sequence of games is defined as in the definition of the adaptive gCCA2 attack.
Game $\mathbf{G}_0$. In game $\mathbf{G}_0$, $\mathcal{A}$ receives the public key PK and adaptively interleaves queries to the corruption oracle $\mathsf{Cor}_{SK_{\mathsf{BE}}}(\cdot)$ and to the decryption oracle $\mathcal{D}_{SK_{\mathsf{BE}}}(\cdot, \cdot)$. Then, she queries the encryption oracle $\mathcal{E}_{PK,\mathcal{R},\sigma}(\cdot, \cdot)$ on (s_0, s_1), where $\mathcal{R}$ must contain all users that $\mathcal{A}$ compromised through the oracle $\mathsf{Cor}_{SK_{\mathsf{BE}}}(\cdot)$; $\mathcal{A}$ receives back the enabling block $\mathcal{T}^*$. Then, $\mathcal{A}$ can again query the decryption oracle $\mathcal{D}_{SK_{\mathsf{BE}}}(i, \mathcal{T})$, restricted only in that $\neg\Re_i(\mathcal{T}, \mathcal{T}^*)$. Finally, she outputs her guess $\sigma^* \in \{0, 1\}$. Let T_0 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_0$.
Game $\mathbf{G}_1$. Game $\mathbf{G}_1$ is identical to game $\mathbf{G}_0$, except that steps $E4, E8$ of the encryption algorithm in Fig. 2, are changed in: $\boxed{E4'.\ H_t \leftarrow u_1^{Z_{1,t}} \cdot u_2^{Z_{2,t}},\ t \in [0, z]}$ and $\boxed{E8'.\ v_t \leftarrow u_1^{X_{1,t}+Y_{1,t}\alpha} \cdot u_2^{X_{2,t}+Y_{2,t}\alpha},\ t \in [0, z]}$. By the properties of the Lagrange Interpolation in the Exponent, it is clear that step $E4'$ computes the same values $\{H_{j_t}\}_{t=0}^{z}$ as steps $E4$; similarly, step $E8'$ computes the same values $\{v_t\}_{t=0}^{z}$ as step $E8$. The point of these changes is just to make explicit any functional dependency of the above quantities on u_1 and u_2. Let T_1 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_1$. Clearly, it holds that $\Pr[T_0] = \Pr[T_1]$.
Game $\mathbf{G}_2$. To turn game $\mathbf{G}_1$ into game $\mathbf{G}_2$ we again modify the encryption oracle used in $\mathbf{G}_1$, replacing step $E1$ with $\boxed{E1'.\ r_1 \leftarrow_R \mathbb{Z}_q,\ r_2 \leftarrow_R \mathbb{Z}_q \setminus \{r_1\}}$ and step $E3$ with $\boxed{E3'.\ u_2 \leftarrow g_2^{r_2}}$. Let T_2 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_2$. Notice that while in $\mathbf{G}_1$ the values u_1 and u_2 are obtained using the same value r_1, in $\mathbf{G}_2$ they are independent subject to $r_1 \neq r_2$. Therefore, using a standard

reduction argument, any non-negligible difference in behavior between $\mathbf{G}_1$ and $\mathbf{G}_2$ can be used to construct a PPT algorithm $\mathcal{A}_1$ that is able to distinguish Diffie-Hellman tuples from totally random tuples with non negligible advantage. Hence, $\left|\Pr[T_2] - \Pr[T_1]\right| \leq \epsilon_1$ for some negligible ϵ_1.

Game $\mathbf{G}_3$. To define game $\mathbf{G}_3$ we modify the decryption oracle, changing steps $D2, D4, D5$ with

$D2'.\ \bar{v}_i \leftarrow u_1^{(X_{1,i}+Y_{1,i}\alpha)+(X_{2,i}+Y_{2,i}\alpha)\cdot w}$, $D4'.$ **if** $(u_2 = u_1^w \wedge v_i = \bar{v}_i)$ **then**, $D5'.\ H_i \leftarrow u_1^{Z_{1,i}+Z_{1,i}\cdot w}$. The rationale behind these changes is that we want to strengthen the condition that the enabling block has to meet in order to be considered valid and hence to be decrypted. This will make it easier to show the security of the scheme; however, for these changes to be useful, there should be no observable difference in the way invalid enabling blocks are "caught" in games $\mathbf{G}_2$ and $\mathbf{G}_3$. To make it formal, let T_3 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_3$, and let R_3 be the event that $\mathcal{A}$ submits some decryption query that would have been decrypted in game $\mathbf{G}_2$ but is rejected in game $\mathbf{G}_3$; in other words, R_3 is the event that some decryption query that would have passed the test in step $D4$ of the decryption oracle used in $\mathbf{G}_2$, fails to pass the test in step $D4'$ used in $\mathbf{G}_3$. Clearly, $\mathbf{G}_2$ and $\mathbf{G}_3$ are identical until event R_3 occurs; hence, if R_3 never occurs, the adversary has the same chances to win in both the two games, i.e. (using Lemma 1) $T_3 \wedge \neg R_3 \equiv T_2 \wedge \neg R_3 \Rightarrow \left|\Pr[T_3] - \Pr[T_2]\right| \leq \Pr[R_3]$.

To bound the last probability, we consider two more games, $\mathbf{G}_4$ and $\mathbf{G}_5$.

Game $\mathbf{G}_4$. To define game $\mathbf{G}_4$, we change step $E6$ of the encryption oracle as follows: $E6'.\ e \leftarrow_R \mathbb{Z}_q,\ S \leftarrow g_1^e$. Let T_4 be the event that $\sigma = \sigma^*$ in game $\mathbf{G}_4$. Because of this last change, the challenge no longer contains the bit σ, nor does any other information in the adversary's view; therefore, we have that $\Pr[T_4] = \frac{1}{2}$.

Let R_4 be the event that $\mathcal{A}$ submits some decryption query that would have been decrypted in game $\mathbf{G}_2$ but is rejected in game $\mathbf{G}_4$; in other words, R_4 is the event that some decryption query that would have passed the test in step $D4$ of the decryption oracle used in $\mathbf{G}_2$, fails to pass the test in step $D4'$ used in $\mathbf{G}_4$. In [7], we prove (Lemma 4) that those events happen with the same probability as the corresponding events of $\mathbf{G}_3$, i.e. $\Pr[T_4] = \Pr[T_3]$ and $\Pr[R_4] = \Pr[R_3]$.

Game $\mathbf{G}_5$. We again modify the decryption algorithm, adding a *special rejection rule*, to prevent $\mathcal{A}$ from submitting illegal enabling blocks to the decryption oracle, once she has received her challenge $\mathcal{T}^* = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_z^*, H_{j_z^*}), v_0^*, \ldots, v_z^* \rangle$:

> After adversary $\mathcal{A}$ receives the challenge $\mathcal{T}^*$, the decryption oracle rejects any query $\langle i, \mathcal{T} \rangle$, with $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v_0, \ldots, v_z \rangle$ and $\langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}) \rangle \neq \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_z^*, H_{j_z^*}) \rangle$, but $\alpha = \alpha^*$, and it does so before executing the test in $D4'$.

Notice that in the gCCA2 setting the adversary is not allowed to query the decryption oracle $\mathsf{Dec}(i, \mathcal{T})$ on enabling blocks $\Re_i$-equivalent to the challenge $\mathcal{T}^*$.

Therefore, when the *special rejection rule* is applied, we already know that it holds $\neg\Re_i(\mathcal{T}, \mathcal{T}^*)$.

Let C_5 be the event that the adversary submits a decryption query that is rejected using the above *special rejection rule*; let R_5 be the event that $\mathcal{A}$ submits some decryption query that would have passed the test in step $D4$ of the decryption oracle used in $\mathbf{G}_2$, but fails to pass the test in step $D4'$ used in $\mathbf{G}_5$. Notice that this implies that such a query passed the $\Re_i$-equivalence test and the *special rejection rule*, because otherwise step $D4'$ wouldn't have been executed. Clearly, games $\mathbf{G}_4$ and $\mathbf{G}_5$ are identical until event C_5 occurs, i.e. by Lemma 1: $R_5 \wedge \neg C_5 \equiv R_4 \wedge \neg C_5 \Rightarrow \left|\Pr[R_5] - \Pr[R_4]\right| \leq \Pr[C_5]$.

Our final task is to show that events C_5 and R_5 occur with negligible probability: while the argument to bound event C_5 is based on the collision resistance assumption for the family $\mathcal{F}$ (using a standard reduction argument, we can construct a PPT algorithm $\mathcal{A}_2$ that breaks the collision resistance assumption with non negligible advantage), the argument to bound event R_5 hinges upon the fact that the adversary is not allowed to submit queries that are "$\Re_i$-related" to her challenge, and upon information-theoretic considerations (as proven in Lemma 5 of [7]). From these considerations, we obtain that $\Pr[C_5] \leq \epsilon_2$ and $\Pr[R_5] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$, where ϵ_2 is a negligible quantity and $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary.

Finally, combining the intermediate results, we can conclude that adversary $\mathcal{A}$'s advantage is negligible; more precisely: $\mathsf{Adv}^{\mathsf{gCCA2}}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \epsilon_1 + \epsilon_2 + Q_{\mathcal{A}}(\lambda)/q$.

4.3 z-Resilience against CCA2 Attack

In Section 4.2, we saw how a direct application of the standard technique of [5, 6] does not provide a complete solution to the CCA2 problem, but only suffices for gCCA2 security. As proven in Lemma 5 of [7], the restriction imposed by the gCCA2 attack (namely, forbidding the adversary to submit decryption queries $\langle i, \mathcal{T}\rangle$ such that $\Re_i(\mathcal{T}, \mathcal{T}^*)$ holds) is essential for the security of the previous Broadcast Encryption Scheme. Indeed, given a challenge $\mathcal{T}^*$ with tag sequence $v_0 \ldots v_z$, it is trivial to come up with a different sequence $v'_0 \ldots v'_z$ such that $v_i = v'_i$, resulting in a "different" enabling block $\mathcal{T}' \neq \mathcal{T}^*$: however, $\mathsf{Dec}(i, \mathcal{T}^*) = \mathsf{Dec}(i, \mathcal{T}')$, allowing the adversary to "break" the CCA2-security.

Although we feel that gCCA2-security is enough for most applications of Broadcast Encryption Schemes, it is possible to non-trivially modify the Broadcast Encryption Scheme presented in Section 4.2 to obtain CCA2 security (with only a slight efficiency loss). The modified scheme, presented in this section, maintains the same Key Generation and Registration algorithms described before; the essential modifications involve the operations used to construct the enabling block. In particular, to achieve CCA2 security, it is necessary to come up with some trick to make the tag sequence $v_0, \ldots, v_z$ non-malleable. To this aim, we will use any secure (deterministic) *message authentication code* (MAC) to guarantee the integrity of the entire sequence. In fact, we only need any *one-time* MAC, satisfying the following simple property: given a (unique) correct

Encryption algorithm $\mathsf{Enc}(PK, s, \mathcal{R})$	Decryption algorithm $\mathsf{Dec}(i, \mathcal{T})$
$E1.\ r_1 \leftarrow_R \mathbb{Z}_q$	$D1.\ \alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}))$
$E2.\ u_1 \leftarrow g_1^{r_1}$	$D2.\ \bar{v}_i \leftarrow u_1^{X_{1,i}+Y_{1,i}\alpha} \cdot u_2^{X_{2,i}+Y_{2,i}\alpha}$
$E3.\ u_2 \leftarrow g_2^{r_1}$	$D3.\ v_i \leftarrow \mathsf{EXP\text{-}LI}(0, .., z; v_0, .., v_z)(i)$
$E4.\ H_t \leftarrow h_t^{r_1},\ t \in [0, z]$	$D4.$ **if** $v_i = \bar{v}_i$ **then**
$E5.$ **for** $t = 1$ **to** z **do**	$D5.\quad H_i \leftarrow u_1^{Z_{1,i}} \cdot u_2^{Z_{2,i}}$
$E6.\quad H_{j_t} \leftarrow \mathsf{EXP\text{-}LI}(0, .., z; H_0, .., H_z)(j_t)$	$D6.\quad s \,\|\, k \leftarrow \frac{S}{\mathsf{EXP\text{-}LI}(j_1, .., j_z, i; H_{j_1}, .., H_{j_z}, H_i)(0)}$
$E7.$ **end for**	$D7.\quad$ extract s and k from $s \parallel k$
$E8.\ k \leftarrow_R \mathcal{K}$	$D8.\quad$ **if** $\tau \neq \mathsf{MAC}_k(v_0, .., v_z)$ **then**
$E9.\ S \leftarrow (s \parallel k) \cdot H_0$	$D9.\quad\quad$ **return** $\perp$
$E10.\ \alpha \leftarrow \mathcal{H}(S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}))$	$D10.\quad$ **else return** s
$E11.\ v_t \leftarrow c_t^{r_1} \cdot d_t^{r_1 \alpha},\ t \in [0, z]$	$D11.\quad$ **end if**
$E12.\ \tau \leftarrow \mathsf{MAC}_k(v_0, .., v_z)$	$D12.$ **else return** $\perp$
$E13.\ \mathcal{T} \leftarrow \langle S, u_1, u_2, (j_1, H_{j_1}), .., (j_z, H_{j_z}), v_0, .., v_z, \tau \rangle$	$D13.$ **end if**

Fig. 3. Encryption and decryption algorithms for the CCA2 scheme

value $\mathsf{MAC}_k(M)$ for some message M (under key k), it is infeasible to come up with a correct (unique) value of $\mathsf{MAC}_k(M')$, for any $M' \neq M$.

The Encryption Algorithm. The encryption algorithm Enc receives as input the public key PK, the session key s to be embedded within the enabling block and a set $\mathcal{R} = \{j_1, \ldots, j_z\}$ of revoked users. To construct the enabling block $\mathcal{T}$, the encryption algorithm (defined in Fig. 2) operates similarly to the gCCA2 encryption algorithm: the main difference is that now a MAC key k, randomly chosen from the MAC key space $\mathcal{K}$, is used to MAC the tag sequence $v_0, \ldots, v_z$, and is encapsulated within $\mathcal{T}$ along with the session key s.

The Decryption Algorithm. To recover the session key embedded in the enabling block $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v_0, \ldots, v_z, \tau \rangle$ a legitimate user i can proceed as in Fig. 3. If i is a revoked user, the algorithm fails in step $D6$, since the interpolation points $j_1, \ldots, j_z, i$ are not pairwise distinct.

Security. The security analysis for this scheme is very subtle, because there is the risk of circularity in the use of the MAC key k. Namely, k is part of the ciphertext (since it is encapsulated, along with the session key s, within S); this means that α, the hash of the ciphertext, depends on k (at least Information-Theoretically), and thus the sequence of tags depends on k. In other words, we are MAC-ing something that depends on the MAC key k, which could be a problem. Luckily, the Information-Theoretic nature of the structural approach to the security analysis that we are pursuing (following [6]) allows us to prove that actually k is completely hidden within S, so that MAC-ing the resulting tag with k is still secure.

The solution to the CCA2 problem for Broadcast Encryption Schemes and the relative security analysis can be viewed as the main technical contribution of this paper; at the same time, the capability to resolve the apparent circularity in the use of the MAC demonstrates the importance of providing a formal model

and precise definitions, without which it would have been much harder to devise a correct proof of security for the above scheme.

Theorem 3. *If the* DDH *Problem is hard in* $\mathbb{G}$, $\mathcal{H}$ *is chosen from a collision-resistant hash functions family* $\mathcal{F}$ *and* MAC *is a one-time message authentication code, then the above Broadcast Encryption Scheme is* z*-resilient against chosen ciphertext attacks.*

Proof. The proof proceeds defining a sequence of games similar to that presented in Theorem 2. The definition of games $\mathbf{G}_0, \ldots, \mathbf{G}_5$ closely follow the exposition given in Theorem 2: however, the statements of all lemmas (and their proofs) need to be changed to accommodate for the use of the MAC. In particular, we can easily state and prove a lemma analogous to Lemma 4 in [7], where the only difference is the presence of information about the MAC key k in the challenge (see Lemma 6 of [7]). More importantly, to bound the probability $\Pr[R_5]$ we introduce a new game $\mathbf{G}_6$ to deal with the use of the MAC in the enabling block, while a lemma similar to Lemma 5 is used to bound the probability of event R_6 defined in game $\mathbf{G}_6$ (see [7] for the details).

Game $\mathbf{G}_6$. We again modify the decryption algorithm, adding a *second special rejection rule* to detect illegal enabling blocks submitted by $\mathcal{A}$ to the decryption oracle, once she has received her challenge $\mathcal{T}^* = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_z^*, H_{j_z^*}), v_0^*, \ldots, v_z^*, \tau^* \rangle$. Notice that, while the *special rejection rule*, defined in game $\mathbf{G}_5$, is used to reject adversary's queries aiming at exploiting any weakness in the collision-resistant hash family $\mathcal{F}$, the *second special rejection rule* is used to reject ciphertexts aiming at exploiting any weakness in the MAC scheme.

> After adversary $\mathcal{A}$ receives the challenge $\mathcal{T}^*$, the decryption oracle rejects any query $\langle i, \mathcal{T} \rangle$, with $\mathcal{T} = \langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}), v_0, \ldots, v_z, \tau \rangle$ and $\langle S, u_1, u_2, (j_1, H_{j_1}), \ldots, (j_z, H_{j_z}) \rangle = \langle S^*, u_1^*, u_2^*, (j_1^*, H_{j_1^*}), \ldots, (j_z^*, H_{j_z^*}) \rangle$ and $(v_0, \ldots, v_z) \neq (v_0^*, \ldots, v_z^*)$, but $\tau = \mathsf{MAC}_{k^*}(v_0, \ldots, v_z)$, and it does so before executing the test in $D4'$, and before applying the *special rejection rule*.

Let M_6 be the event that the adversary submits a decryption query that is rejected in game $\mathbf{G}_6$ using the *second special rejection rule*; let C_6 be the event that the adversary submits a decryption query that is rejected in game $\mathbf{G}_6$ using the *special rejection rule*; let R_6 be the event that $\mathcal{A}$ submits some decryption query that would have passed both the test in step $D4$ and in step $D8$ of the decryption oracle used in game $\mathbf{G}_2$, but fails to pass the test in step $D4'$ used in game $\mathbf{G}_6$. Notice that this implies that such a query passed both the *second special rejection rule* and the *special rejection rule*, because otherwise step $D4'$ wouldn't have been executed at all.

Event M_6 is closely related to the security of the one time MAC used in the scheme; in particular, any difference in behavior between game $\mathbf{G}_5$ and game $\mathbf{G}_6$ can be used to construct a PPT algorithm $\mathcal{A}_3$ that is able to forge a legal authentication code under a one-message attack with non-negligible probability, thus breaking the MAC scheme. Hence, $\Pr[M_6] \leq \epsilon_3$, for some negligible ϵ_3.

Moreover, since $\mathbf{G}_5$ and $\mathbf{G}_6$ are identical until event M_6 occurs, if it doesn't occur at all, they will proceed identically; by Lemma 1: $C_6 \wedge \neg M_6 \equiv C_5 \wedge \neg M_6 \Rightarrow \left|\Pr[C_6] - \Pr[C_5]\right| \leq \Pr[M_6]$ and $R_6 \wedge \neg M_6 \equiv R_5 \wedge \neg M_6 \Rightarrow \left|\Pr[R_6] - \Pr[R_5]\right| \leq \Pr[M_6]$.

Our final task is to bound the probability that events C_6 and R_6 occur: the argument to bound $\Pr[C_6]$ is based on the collision resistance assumption for the family $\mathcal{F}$, while the argument to bound $\Pr[R_6]$ hinges upon information-theoretic considerations (as proven in Lemma 7 of [7]). From those facts, we obtain that $\Pr[C_6] \leq \epsilon_2$ and $\Pr[R_6] \leq \frac{Q_{\mathcal{A}}(\lambda)}{q}$, where ϵ_2 is a negligible quantity and $Q_{\mathcal{A}}(\lambda)$ is an upper bound on the number of decryption queries made by the adversary.

Finally, combining the intermediate results, we can conclude that adversary $\mathcal{A}$'s advantage is negligible; more precisely: $\mathsf{Adv}^{\mathsf{CCA2}}_{\mathsf{BE},\mathcal{A}}(\lambda) \leq \epsilon_1 + \epsilon_2 + 2\epsilon_3 + Q_{\mathcal{A}}(\lambda)/q$.

Acknowledgments

We wish to thank Jonathan Katz, Yevgeniy Kushnir, Antonio Nicolosi and Victor Shoup for helpful observations on an preliminary version of the paper and the anonymous referees for useful comments.

References

[1] J. H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. In *Advances in Cryptology - EuroCrypt '02*, pages 83–107, Berlin, 2002. Springer-Verlag. LNCS 2332. 100, 101, 103, 104

[2] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science - FOCS '97*, pages 394–403, 1997. 104

[3] D. Boneh. The Decision Diffie-Hellman Problem. In *Algorithmic Number Theory - ANTS-III*, pages 48–63, Berlin, 1998. Springer-Verlag. LNCS 1423. 102

[4] B. Chor, A. Fiat, and N. Naor. Tracing Traitors. In *Advances in Cryptology - Crypto '94*, pages 257–270, Berlin, 1994. Springer-Verlag. LNCS 839. 102

[5] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology - Crypto '98*, pages 13–25, Berlin, 1998. Springer-Verlag. LNCS 1462. 101, 105, 107, 108, 111

[6] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Scheme Secure against Adaptive Chosen Ciphertext Attack. Manuscript, 2001. 101, 103, 105, 107, 108, 111, 112

[7] Y. Dodis and N. Fazio. Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack. Full version of this paper, available at `http://eprint.iacr.org/`, 2002. 100, 107, 110, 111, 113, 114

[8] A. Fiat and M. Naor. Broadcast Encryption. In *Advances in Cryptology - Crypto '93*, pages 480–491, Berlin, 1993. Springer-Verlag. LNCS 773. 100

[9] E. Gafni, J. Staddon, and Y. L. Yin. Efficient Methods for Integrating Traceability and Broadcast Encryption. In *Advances in Cryptology - Crypto '99*, pages 372–387, Berlin, 1999. Springer-Verlag. LNCS 1666. 102

[10] A Garay, J. Staddon, and A. Wool. Long-Lived Broadcast Encryption. In *Advances in Cryptology - Crypto 2000*, pages 333–352, Berlin, 2000. Springer-Verlag. LNCS 1880. 100

[11] D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. In *Advances in Cryptology - Crypto '02*, pages 47–60, Berlin, 2002. Springer-Verlag. LNCS 2442. 100

[12] M. Luby and J. Staddon. Combinatorial Bounds for Broadcast Encryption. In *Advances in Cryptology - EuroCrypt '98*, pages 512–526, Berlin, 1998. Springer-Verlag. LNCS 1403. 100

[13] D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. In *Advances in Cryptology - Crypto '01*, pages 41–62, Berlin, 2001. Springer-Verlag. LNCS 2139. 100, 101

[14] M. Naor and B. Pinkas. Efficient Trace and Revoke Schemes. In *Financial Cryptography - FC 2000*, pages 1–20, Berlin, 2000. Springer-Verlag. LNCS 1962. 100, 101, 102, 105

[15] V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption. Manuscript, 2001. 100, 101, 103

[16] W. G. Tzeng and Z. J. Tzeng. A Public-Key Traitor Tracing Scheme with Revocation Using Dynamics Shares. In *Public Key Cryptography - PKC '01*, pages 207–224, Berlin, 2001. Springer-Verlag. LNCS 1992. 100, 101, 102, 105, 107