Proceedings of the first International Conference on Pairing-based Cryptography (Pairing 2007)
(2 – 3 july 2007, Tokyo, Japan)
T. Takagi, t. Okamoto, E. Okamoto and T. Okamoto Eds. Springer-Verlag, LNCS 4575, pages 39–59.

Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys

Cécile Delerablée[1,3], Pascal Paillier[2], and David Pointcheval[3]

[1] France Telecom Division R&D, cecile.delerablee@orange-ftgroup.com
[2] Gemalto Security Labs, pascal.paillier@gemalto.com
[3] ENS-CNRS, david.pointcheval@ens.fr

Abstract. This paper puts forward new efficient constructions for public-key broadcast encryption that simultaneously enjoy the following properties: receivers are stateless; encryption is collusion-secure for arbitrarily large collusions of users and security is tight in the standard model; new users can join dynamically i.e. without modification of user decryption keys nor ciphertext size and little or no alteration of the encryption key. We also show how to permanently revoke any subgroup of users. Most importantly, our constructions achieve the optimal bound of $O(1)$-size either for ciphertexts or decryption keys, where the hidden constant relates to a couple of elements of a pairing-friendly group. Our broadcast-KEM trapdoor technique, which has independent interest, also provides a dynamic broadcast encryption system improving all previous efficiency measures (for both execution time and sizes) in the private-key setting.

1 Introduction

Broadcast Encryption. The concept of stateless broadcast encryption was introduced by Fiat and Naor in [5]. In this paradigm, a broadcaster encrypts messages and transmits these to a group of users $\mathcal{U}$ who are listening to a broadcast channel and use their private keys to decrypt transmissions. The broadcaster may exclude any subset of users $\mathcal{R} \subseteq \mathcal{U}$ from being able to decrypt the contents of the broadcast thanks to a one-time exclusion or revocation mechanism. The subset of revoked users $\mathcal{R}$ is chosen at encryption time and may change from one encryption to the next. A broadcast encryption scheme is said to be (t, n)-collusion secure if for any r-subset $\mathcal{R} \subseteq \mathcal{U}$ with $r \leq t$ and $|\mathcal{U}| = n$, users in $\mathcal{R}$ can by no means infer information about the broadcast message. It is said to be fully collusion secure when it is (n, n)-collusion secure. There are mainly two categories of broadcast encryption systems:

$r \ll n$: meaning that we broadcast to all but a small set of revoked users. A number of systems [14, 9, 8] have been suggested that achieve at best $O(r)$-size ciphertexts and $O(\log n)$ private key size. This paper specifically focuses on this case and puts forward constructions which improve these bounds in the public-key setting and even reach their information-theoretic values in the private-key setting.

$n - r \ll n$: we broadcast only to a few users in the group. The best known systems are the scheme of Boneh, Gentry and Waters [3] which achieves $O(\sqrt{n})$-size ciphertexts and private keys, and the trivial scheme where users hold independent keys pairs and the message is sequentially encrypted under the $n - r$ public keys of non-revoked users. This trivial scheme achieves

$O(n-r)$-size ciphertexts and $O(1)$ private keys and is more efficient than [3] when $n - r < O(\sqrt{n})$.

Although previous works often use as an efficiency measure the size of public and private keys, we choose to rigorously separate encryption from decryption key material by taking into account all elements required to perform encryption or decryption. We will denote by λ_c the size of the broadcast ciphertext, λ_{dk} the (maximal) size of a user decryption key (which may then contain private and public parts) and λ_{ek} the size of the encryption key (which may be public or private). τ_{ek} and τ_{dk} denote the execution time of encryption and decryption respectively.

Related work in the $r \ll n$ case. Naor et al. [14] suggested two fully collusion secure broadcast systems: NNL_1 (based on the Complete-Subtree method) which achieves $\lambda_c = O(r \log n/r)$ and $\lambda_{\mathsf{dk}} = O(\log n)$ and NNL_2 (Subtree-Difference method) where $\lambda_c = O(r)$ and $\lambda_{\mathsf{dk}} = O(\log^2 n)$. Originally the (private) encryption key has size linear in n, but by using a PRF to generate user decryption keys, the size of the encryption key can be reduced to $O(1)$ in both NNL_1 and NNL_2. Note however that these two systems do not support public-key encryption. Dodis and Fazio [4] later refined NNL_2 into a public key broadcast encryption scheme with $O(1)$-size encryption key. We also mention the work of Dodis and Fazio [4] which by using parallelized schemes lead to a broadcast system that has essentially the same characteristics as those of NNL_2.

Related work in the $n - r \ll n$ case. The so-called trivial broadcast system consists in multiple encryptions of the message under individual and unrelated public keys. It is easily seen that this gives a public-key broadcast encryption scheme with $\lambda_c = O(n-r)$, $\lambda_{\mathsf{dk}} = O(1)$ and $\lambda_{\mathsf{ek}} = O(n)$. Recently, Boneh, Gentry and Waters [3] proposed a very efficient public-key broadcast encryption system (called BGW_1 hereafter) where both ciphertexts and private keys are of constant size while the public key has size $O(n)$. However, in order to decrypt ciphertexts, users need to store this public key in addition to their $O(1)$-size private keys. Thus each user has to store an actual decryption key of size $\lambda_{\mathsf{dk}} = O(n)$. The same authors suggested a second system (BGW_2) that achieves a trade-off between the key and ciphertext sizes and gives $\lambda_c = \lambda_{\mathsf{dk}} = \lambda_{\mathsf{ek}} = O(\sqrt{n})$. Overall, BGW_2 provides the best broadcast system known so far for general r's i.e. when $r \in [O(\sqrt{n}), n - O(\sqrt{n})]$.

All the above systems make use of the hybrid (KEM-DEM) encryption paradigm where the broadcast ciphertext only encrypts a symmetric key used to encrypt the broadcast contents. We mention that a number of other systems rely on an asymmetric encryption of the whole broadcast data. The encryption rate may asymptotically tend to one in these schemes, thereby reaching transmission sizes similar to hybrid encryption. However, asymmetrically encrypting the whole contents is often unrealistic in practice for performance reasons.

Dynamic broadcast encryption. A basic property very much desired in broadcast encryption (and other group-based protocols) is that the group should be

dynamic in the sense that the group manager can invite new members to join or permanently revoke undesired members in a very efficient way. Although long-term revocation necessarily implies a modification of the keys, there is no such theoretical requirement when a new member joins the group. In this respect, we say that a broadcast system is *dynamic* when

i) the system setup as well as the ciphertext size are fully independent from the expected number of users or an upper bound thereof,

ii) a new user can join anytime without implying a modification of preexisting user decryption keys,

iii) the encryption key is *unchanged* in the private-key setting or *incrementally updated* in the public-key setting, meaning that this operation must be of complexity at most $O(1)$.

Hence, by definition, dynamic systems support arbitrarily many users. In [3] as well as in NNL_1 and NNL_2, either a large upper bound on the number of possible users is chosen at initialization time or the decryption keys have to be recomputed when a user joins the group, resulting in that those systems are not dynamic. Similarly, the trivial broadcast system is not dynamic since the ciphertext must include one additional element per new user, irremediably altering its size. As discussed in [14, p. 56], the property of being dynamic is incompatible with forward-secrecy because new group members can actually decrypt all previously encrypted messages. This feature may however be desirable; a newly manufactured DVD player is expected to play any properly encrypted DVD issued in the past. Achieving forward-secrecy requires the long-term revocation and a re-keying of user decryption keys.

Our contributions. Introducing a new multi-receiver encryption trapdoor based on bilinear maps, we suggest broadcast encryption systems which improve the points discussed above. First, in all our schemes, either the broadcast ciphertext or the decryption key dk_i containing all the information required by the receiver to decrypt is of constant size. Second, the group manager can dynamically include new members while preserving previously computed information: in particular, user decryption keys need not be recomputed, the morphology and size of ciphertexts are unchanged and the group encryption key ek requires minimal or no modification. Thirdly, our constructions provably resist full collusions relative to a bilinear map related computational problem denoted (t, n)-GDDHE. Our security reductions are *tight* and do not rely on random oracles. We also show that (t, n)-GDDHE has generic security. Finally, we introduce the most efficient private-key broadcast encryption scheme known so far which features constant-size encryption and decryption keys and information-theoretically minimal ciphertext size.

For the sake of completeness, Figure 1 compares our schemes with the previous proposals in terms of ciphertext and key sizes. We consider the private-key broadcast encryption schemes NNL_1 and NNL_2 proposed by Naor et al. [14] (where the group encryption key ek remains private) as well as the public-key broadcast encryption schemes BGW_1 and BGW_2 proposed by Boneh et al. [3]. We

denote by $\mathsf{BGW'}_1$ a slightly modified version of BGW_1 where the public parameters needed by the decryption procedure are included in the ciphertext rather than in the decryption key. BGW_1 and $\mathsf{BGW'}_1$ are described in more detail in Appendix A.

Schemes	Key/Ciphertext Sizes			Time Complexity		Dynamic?
Public-key	λ_{ek}	λ_{dk}	λ_c	τ_{ek}	τ_{dk}	
Trivial	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	$O(1)$	no
BGW_1	$O(n)$	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	no
$\mathsf{BGW'}_1$	$O(n)$	$O(1)$	$O(n-r)$	$O(n-r)$	$O(n-r)$	no
BGW_2	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$	no
Construction 1	$O(n)$	$O(1)$	$O(r)$	$O(r^2)$	$O(r)$	yes
Construction 2	$O(n)$	$O(n)$	$O(1)$	$O(r^2)$	$O(r^2)$	no
Private-key	λ_{ek}	λ_{dk}	λ_c	τ_{ek}	τ_{dk}	
NNL_1	$O(1)$	$O(\log n)$	$O(r \log \frac{n}{r})$	$O(r \log \frac{n}{r})$	$O(\log \log n)$	no
NNL_2	$O(1)$	$O(\log^2 n)$	$O(r)$	$O(r)$	$O(\log n)$	no
Construction 3	$O(1)$	$O(1)$	$O(r)$	$O(r)$	$O(r)$	yes

Fig. 1. Comparing the efficiency of fully collusion secure broadcast encryption schemes

Remark 1. It is a common practice in broadcast systems (and other group-oriented protocols as well) to ignore the part of the broadcast ciphertext that identifies the target subset of users (our $\mathcal{R}$, or the subset S of effective receivers in [3]). What is called ciphertext size usually refers to the size of the header alone, not the size of the *full* header. As discussed in [14], transmitting the r-subset $\mathcal{R} \subseteq \{1, \ldots, n\}$ requires an extra $O(r \log n/r)$ bits. We note however that transmitting $\mathcal{R}$ is actually not necessary in Constructions 1 and 3.

Roadmap. Section 2 provides a number of definitional facts about bilinear maps and the General Diffie-Hellman Exponent assumption. We define dynamic broadcast encryption schemes and related security notions in Section 3. We describe our main construction in Section 4 and prove its security in Section 5. We suggest a number of related variants and discuss these in Section 6. We finally conclude on a number of open issues.

2 Preliminaries

2.1 Bilinear Maps

We briefly review the necessary facts about bilinear maps. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three cyclic groups of prime order p. The group laws in $\mathbb{G}_1$ and $\mathbb{G}_2$ are noted additively using elliptic curve conventions, whereas the inner law of $\mathbb{G}_T$ is noted multiplicatively. A bilinear map $e(\cdot, \cdot)$ is a map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that for any $G \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$,

$$e([a]G, [b]H) = e(G, H)^{ab}$$

and $e(G,H) \neq 1$ unless $G = 1$ or $H = 1$. A bilinear map group system $\mathbb{S}$ is a tuple

$$\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$$

composed of the objects as described above. $\mathbb{S}$ may also include group generators in its description. We impose all group operations as well as the bilinear map $e(\cdot,\cdot)$ to be efficiently computable i.e. in time $\mathsf{poly}(|p|)$. We know three categories of bilinear map group systems that are of interest in cryptography:

- the symmetric case $\mathbb{G}_1 = \mathbb{G}_2$ and by extension the one where an efficient and efficiently invertible isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is known [10, 11, 2],
- the asymmetric case where an efficient isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ is known but no efficient algorithm is known to invert ψ or more generally to isomorphically map $\mathbb{G}_1$ onto $\mathbb{G}_2$ [12, 13],
- the dissociate case where no efficient isomorphism $\mathbb{G}_2 \to \mathbb{G}_1$ or $\mathbb{G}_1 \to \mathbb{G}_2$ is known.

As seen later, we make use of an arbitrary bilinear map group system in our constructions. In particular, we do not need $\mathbb{G}_1$ and $\mathbb{G}_2$ to be distinct nor equal. Neither do we require the existence[1] of an efficient isomorphism going either way between $\mathbb{G}_1$ and $\mathbb{G}_2$. Practical implementations may therefore rely on any category of bilinear maps and select an $\mathbb{S}$ that optimizes the size of group elements or the performance of group operations.

2.2 The General Diffie-Hellman Exponent Assumption

We make use of a nice generalization of the Diffie-Hellman Exponent assumption due to Boneh, Boyen and Goh [1]. The framework suggested in [1] applies to symmetric and asymmetric bilinear map group systems but can easily be extended to the dissociate case, adopting the notations of [1]. We give here a rough overview in the symmetric case. Let then $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system such that $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Let $G_0 \in \mathbb{G}$ be a generator of $\mathbb{G}$, and set $g = e(G_0, G_0) \in \mathbb{G}_T$. Let s, m be positive integers and $P, Q \in \mathbb{F}_p[X_1, \ldots, X_m]^s$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$ and impose that $p_1 = q_1 = 1$. For any function $h : \mathbb{F}_p \to \Omega$ and vector $(x_1, \ldots, x_m) \in \mathbb{F}_p^m$, $h(P(x_1, \ldots, x_m))$ stands for $(h(p_1(x_1, \ldots, x_m)), \ldots, h(p_s(x_1, \ldots, x_m))) \in \Omega^s$. We use a similar notation for the s-tuple Q. Let $F \in \mathbb{F}_p[X_1, \ldots, X_m]$. It is said that F *depends* on (P, Q), which we denote by $F \in \langle P, Q \rangle$, when there exists a linear decomposition

$$F = \sum_{1 \leq i,j \leq s} a_{i,j} p_i p_j + \sum_{1 \leq i \leq s} b_i q_i \ ,$$

with coefficients $a_{i,j}, b_i \in \mathbb{Z}_p$. Let P, Q be as above and $F \in \mathbb{F}_p[X_1, \ldots, X_m]$. The (P, Q, F)-General Diffie-Hellman Exponent problems are defined as follows.

[1] A one-way isomorphism is often necessary to prove the security of pairing-based systems.

Definition 2 ((P, Q, F)-GDHE)**.** Given the vector

$$H(x_1, \ldots, x_m) = \left([P(x_1, \ldots, x_m)]\, G_0, g^{Q(x_1,\ldots,x_m)}\right) \in \mathbb{G}^s \times \mathbb{G}_T^s \ ,$$

compute $g^{F(x_1,\ldots,x_m)}$.

Definition 3 ((P, Q, F)-GDDHE)**.** Given $H(x_1, \ldots, x_m) \in \mathbb{G}^s \times \mathbb{G}_T^s$ as above and $T \in \mathbb{G}_T$, decide whether $T = g^{F(x_1,\ldots,x_m)}$.

We refer to [1] for a proof that (P, Q, F)-GDHE and (P, Q, F)-GDDHE have generic security when $F \notin \langle P, Q\rangle$. We prove that our constructions are fully collusion secure based on the assumption that (P, Q, F)-GDDHE is intractable for some well-defined P, Q, F with $F \notin \langle P, Q\rangle$ and for polynomial parameters $s, m = \mathsf{poly}(\lambda)$ where λ is a security parameter.

3 Dynamic Public-Key Broadcast Encryption

We give a formal definition of a dynamic broadcast encryption scheme and discuss security notions that are associated to the concept. We basically refine the definition of [3] by including a join procedure.

3.1 Definition

A dynamic broadcast encryption scheme involves two authorities: a group manager and a broadcaster. The group manager grants new members access to the group[2] by providing to each new member a public label lab_i and a decryption key dk_i. The generation of $(\mathsf{lab}_i, \mathsf{dk}_i)$ is performed using a secret manager key mk. The broadcaster encrypts messages and transmits these to the whole group of users through the broadcast channel. In a public-key broadcast encryption scheme, the broadcaster does not hold any private information and encryption is performed with the help of a public group encryption key ek containing, possibly among other things, all user labels. When the broadcaster encrypts a message, some group members can be revoked temporarily from decrypting the broadcast content thanks to a one-time revocation mechanism. Following the KEM-DEM methodology, broadcast encryption is viewed as the combination of a specific key encapsulation mechanism (a Broadcast-KEM) with a symmetric encryption (DEM) that shall remain implicit throughout the paper. More formally, a dynamic public-key broadcast encryption scheme $\mathcal{DBE}$ with security parameter λ is a tuple of probabilistic algorithms $\mathcal{DBE} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Encrypt}, \mathsf{Decrypt})$ described as follows:

$\mathsf{Setup}(\lambda)$. Takes as input the security parameter λ and outputs a manager key mk and an initial group encryption key ek. The group manager is given mk, and ek is made public.

[2] Note that given our definition of dynamic broadcast encryption, the group manager cannot revoke users permanently since keys cannot be changed. See Section 6 for more detail.

$\mathsf{Join}(\mathsf{mk}, i)$. Takes as input the manager key mk and a user counter i. Join generates a user label lab_i and a user decryption key dk_i. The user label lab_i is added to the group encryption key $\mathsf{ek} := \mathsf{ek} \cup \{\mathsf{lab}_i\}$ and the user decryption key dk_i is sent to the i-th user securely.

We denote by n the total number of users (evolving over time) and by $\mathcal{U} = \{1, \ldots, n\}$ the set of all users.

$\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$. Takes as input the group encryption key ek and a set of revoked users $\mathcal{R} \subseteq \mathcal{U}$ and outputs a random pair (hdr, K).

When a message $M \in \{0,1\}^*$ is to be broadcast to users in $\mathcal{U} \setminus \mathcal{R}$, the broadcaster generates $(\mathsf{hdr}, K) \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$, computes the encryption C_M of M under the symmetric key K and broadcasts $(\mathsf{hdr}, \mathcal{R}, C_M)$. We will refer to hdr as the header or broadcast ciphertext, $(\mathsf{hdr}, \mathcal{R})$ as the full header, K as the message encryption key and C_M as the broadcast body.

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathcal{R}, \mathsf{hdr})$. Takes as input a header hdr, a subset $\mathcal{R} \subseteq \mathcal{U}$ and a user decryption key dk_i. If $i \in \mathcal{U} \setminus \mathcal{R}$, the algorithm outputs the message encryption key K which is then used to decrypt the broadcast body C_M and recover M.

3.2 Security Notions for Dynamic Public-Key Broadcast Encryption

Semantic Security against Static Adversaries. The standard security notion for broadcast encryption schemes is semantic security against static colluders. Since we consider dynamic public-key broadcast encryption, we extend the security definition to one that is a bit more general than in [3]. More specifically, we allow the adversary to see the group encryption key before choosing the corrupted users:

1. The challenger first runs $\mathsf{Setup}(\lambda)$ to generate a manager key mk and an initial group encryption key ek. The adversary $\mathcal{A}$ is given ek.
2. $\mathcal{A}$ runs exactly n times the Join procedure, but before each invocation $\mathcal{A}$ specifies whether the corresponding new group member is honest or corrupted. Following the i-th call to Join, a user label lab_i is created and added to ek and therefore given to the adversary. If user i is corrupted, $\mathcal{A}$ receives in addition the decryption key dk_i. The user counter i is then incremented and so forth. Eventually, $\mathcal{A}$ ends up with the decryption keys of all corrupted users $\mathcal{C} \subseteq \mathcal{U}$, that is $\{\mathsf{dk}_i\}_{i \in \mathcal{C}}$. Let $t = |\mathcal{C}|$.
3. The challenger runs algorithm $\mathsf{Encrypt}$ with $\mathcal{R} = \mathcal{C}$ i.e. by revoking all corrupted users to randomly generate $(\mathsf{hdr}, K) \leftarrow \mathsf{Encrypt}(\mathsf{ek}, \mathcal{C})$. The challenger randomly selects $b \leftarrow \{0,1\}$, sets $K_b = K$ and sets K_{1-b} to a random value in the appropriate range. The tuple (hdr, K_0, K_1) is returned to $\mathcal{A}$.
4. $\mathcal{A}$ eventually outputs a guess $b' \in \{0,1\}$.

The adversary wins the above game when $b' = b$. Viewing t, n as attack parameters, we denote by $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A})$ the advantage of $\mathcal{A}$ in winning the game:

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n, \mathcal{A}) = |2 \times \Pr[b' = b] - 1| = |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|$$

where the probability is taken over the random coins of $\mathcal{A}$, the challenger and all probabilistic algorithms run by the challenger.

Definition 4 ((t,n)-Collusion Resistance). Let

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t,n) = \max_{\mathcal{A}} \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t,n,\mathcal{A})$$

where the maximum is taken over all probabilistic algorithms $\mathcal{A}$ running in time $\mathsf{poly}(\lambda)$. A dynamic public-key broadcast encryption scheme $\mathcal{DBE}$ is said to be semantically secure against (t,n)-colluders if $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t,n) = \mathsf{negl}(\lambda)$.

Definition 5 (Full Collusion Resistance). Note that for any integers t_1, t_2 such that $0 \leq t_1 \leq t_2 \leq n$ one has $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t_1,n) \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t_2,n)$. $\mathcal{DBE}$ is said to be semantically secure against full collusions if $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(n,n) = \mathsf{negl}(\lambda)$ for $n = \mathsf{poly}(\lambda)$.

Chosen-ciphertext security. Given a chosen-plaintext secure Broadcast-KEM, one can realize CCA secure public-key broadcast encryption using generic security enhancers such as Fujisaki-Okamoto [7, 6] or REACT [15]. Applying the Fujisaki-Okamoto transform guarantees CCA security in the random oracle model assuming the one-wayness of the Broadcast-KEM. REACT gives the same guarantee (and a performance gain at decryption time) assuming one-wayness under plaintext-checking attacks. We therefore do not consider CCA security in our constructions, given that it can be realized at negligible cost[3].

Beyond static adversaries. We comment that the above definition captures adversaries that are less static than in previous schemes because the adversary may choose $\mathcal{C}$ somewhat adaptively while seeing how the group encryption key ek evolves while new users join the system. Up to our knowledge, no public-key broadcast encryption scheme is known to resist fully adaptive adversaries (i.e. where the adversary determines $\mathcal{C}$ after seing all the public information) in the standard model[4]. As commented in [3], any static adversary that has success probability ε in the (t,n)-collusion security game leads to an adaptive adversary with success probability $\varepsilon \cdot 2^{-n}$. However, in practice this reduction is only meaningful for small values of n and building systems resisting fully adaptive colluders is still an open problem in the field.

4 Public-Key DBE with Constant-Size Decryption Keys

4.1 A New Multi-Receiver Encryption Trapdoor

We describe a public-key encryption scheme with multiple receivers featuring $O(1)$-size ciphertexts, decryption keys and encryption key. This system, which we call Construction 0, does not support user revocation and therefore is not

[3] In the RO model. However combining the Broadcast-KEM with ID-based encryption as in [3] may allow to avoid random oracles when such a combination is possible.

[4] This is known to be achievable in the random oracle and generic group models.

a broadcast encryption system. We describe Construction 0 to show the basic trapdoor mechanism which we generalize later to achieve broadcast encryption.

Let $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ be a bilinear map group system with two randomly selected generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$. Assume one publishes $\mathsf{ek} = (H, W, V)$ where $W = [\gamma] G$ for some $\gamma \leftarrow \mathbb{Z}_p$ and $V = e(G, H)$ and keeps $\mathsf{mk} = (G, \gamma)$ secret. Now given $\mathsf{mk} = (G, \gamma)$ and a user counter i, we generate a unique decryption key by randomly selecting a fresh $x_i \leftarrow \mathbb{Z}_p$ and defining $\mathsf{dk}_i = (x_i, A_i, B_i)$ where

$$A_i = \left[\frac{x_i}{\gamma + x_i}\right] G \ , \qquad B_i = \left[\frac{1}{\gamma + x_i}\right] H \ .$$

To generate a random encryption key K given $\mathsf{ek} = (H, W, V)$, the broadcaster randomly picks $k \leftarrow \mathbb{Z}_p^\star$, computes

$$C_1 = [k] W \ , \qquad C_2 = [k] H \ , \qquad K = V^k$$

and broadcasts $\mathsf{hdr} = (C_1, C_2)$. To recover K from hdr with dk_i, the i-th user computes

$$e(C_1, B_i) \cdot e(A_i, C_2) = e(G, H)^{\frac{k \cdot \gamma}{\gamma + x_i}} \cdot e(G, H)^{\frac{k \cdot x_i}{\gamma + x_i}} = V^k = K \ .$$

It turns out that this encryption scheme achieves chosen-plaintext security under the co-DDH assumption. Namely, given $H, [a] H \in \mathbb{G}_2$ and $v, t \in \mathbb{G}_T$, it must be hard to decide whether $t = v^a$. Construction 0 may have applications on its own in contexts where user revocation is not required.

4.2 Achieving User Revocation: A Full Construction

We now proceed to describe our main dynamic public-key broadcast encryption scheme which we refer to as Construction 1 throughout the paper. Our scheme allows decoders to store constant-size decryption keys i.e. $\lambda_{\mathsf{dk}} = O(1)$ and features $O(r)$ ciphertext size where r is the number of revoked users. This is of particular interest when r is small (i.e. $r < \sqrt{n}$).

$\mathsf{Setup}(\lambda)$. Given the security parameter λ, a bilinear map group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot,\cdot))$ is constructed such that $|p| = \lambda$. Also, two generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$ are randomly selected as well as a secret value $\gamma \leftarrow \mathbb{Z}_p^\star$. The manager key is defined as $\mathsf{mk} = (\mathbb{S}, G, H, \gamma)$. The initial group encryption key is $\mathsf{ek} = (\mathbb{S}, H, W, V)$ where $W = [\gamma] G$ and $V = e(G, H)$.

$\mathsf{Join}(\mathsf{mk}, i)$. Given $\mathsf{mk} = (\mathbb{S}, G, H, \gamma)$ and the user counter i, Join randomly selects a fresh $x_i \leftarrow \mathbb{Z}_p^\star$ (thus $x_i \neq x_j$ for $j < i$) and sets $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$ and $\mathsf{lab}_i = (x_i, V_i, B_i)$ where

$$A_i = \left[\frac{x_i}{\gamma + x_i}\right] G \ , \qquad B_i = \left[\frac{1}{\gamma + x_i}\right] H \qquad \text{and} \qquad V_i = V^{\frac{1}{\gamma + x_i}} \ .$$

dk_i is securely given to user i and lab_i is appended to the group encryption key ek.

$\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$. Assume for notational simplicity that $\mathcal{R} = \{1, \ldots, r\}$. Given $\mathsf{ek} = (\mathbb{S}, H, W, V, (x_1, V_1, B_1), \ldots, (x_n, V_n, B_n))$, the broadcaster computes

$$\begin{aligned} P_1 &= \left[\frac{1}{\gamma + x_1}\right] H, \\ P_2 &= \left[\frac{1}{(\gamma + x_1)(\gamma + x_2)}\right] H, \\ &\vdots \\ P_r &= \left[\frac{1}{(\gamma + x_1)\ldots(\gamma + x_r)}\right] H \ . \end{aligned}$$

We describe below a quadratic time algorithm $\mathsf{Aggregate}$ which $\mathsf{Encrypt}$ may use to compute

$$P_r = \mathsf{Aggregate}\left(\mathbb{G}_2, \left(x_1, \left[\frac{1}{\gamma + x_1}\right] H\right), \ldots, \left(x_r, \left[\frac{1}{\gamma + x_r}\right] H\right)\right) \ .$$

By running the $\mathsf{Aggregate}$ algorithm on $(x_1, B_1), \ldots, (x_r, B_r)$ and storing the intermediate variables $P_j = P_{j-1,j}$, $\mathsf{Encrypt}$ also computes P_j for $j < r$. The $\mathsf{Aggregate}$ algorithm given on Fig. 2 precisely goes through the successive evaluations of $P_1, \ldots, P_r$ and these can be stored at no extra cost. The broadcaster then picks $k \leftarrow \mathbb{Z}_p^\star$ at random and sets

$$C_1 = [k]\, W \ , \qquad\qquad C_2 = \left[\frac{k}{(\gamma + x_1)\ldots(\gamma + x_r)}\right] H = [k]\, P_r \ .$$

The same $\mathsf{Aggregate}$ algorithm can also be applied to compute

$$K' = V^{\frac{1}{(\gamma+x_1)\ldots(\gamma+x_r)}} = \mathsf{Aggregate}\left(\mathbb{G}_T, \left(x_1, V^{\frac{1}{\gamma+x_1}}\right), \ldots, \left(x_r, V^{\frac{1}{\gamma+x_r}}\right)\right) \ ,$$

wherefrom $K = K'^k$ is obtained. These two computations are performed in time $O(r^2)$, see below. $\mathsf{Encrypt}$ then defines

$$\mathsf{hdr} = (C_1, C_2, (x_1, P_1), \ldots, (x_r, P_r))$$

and outputs (hdr, K). The ciphertext thus contains r elements of $\mathbb{Z}_p$, $r + 1$ elements of $\mathbb{G}_2$ and one element of $\mathbb{G}_1$.

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathcal{R}, \mathsf{hdr})$. In order to retrieve the message encryption key K encapsulated in the header hdr, the i-th user uses $\{(x_j, P_j)\}_{j=1}^r \subseteq \mathsf{hdr}$ and $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$ to compute

$$K = e\,(C_1, B_{i,\mathcal{R}}) \cdot e\,(A_i, C_2)$$

where

$$B_{i,\mathcal{R}} = \left[\frac{1}{\prod_{j=1}^r (\gamma + x_j)}\right] B_i = \left[\frac{1}{(\gamma + x_i)\prod_{j=1}^r (\gamma + x_j)}\right] H \ .$$

Here $B_{i,\mathcal{R}}$ is computed in time $O(r)$ (instead of $O(r^2)$) given $(x_i, B_i) \subseteq \mathsf{dk}_i$ and $\{(x_j, P_j)\}_{j=1}^r \subseteq \mathsf{hdr}$ by using the $\mathsf{Aggregate}'$ algorithm given later on in the paper. This requires $x_i \neq x_j$ i.e. $i \notin \mathcal{R}$, otherwise $\mathsf{Aggregate}'$ faces a division by zero and returns an error.

Finally note that when $\mathcal{R} = \emptyset$, Construction 1 boils down to Construction 0 except for the encryption key $\mathsf{ek} = (\mathbb{S}, H, W, V) \cup \{(x_i, V_i, B_i)\}_{i=1}^n$ which, in Construction 0, does not include user labels.

4.3 Aggregation of 1-degree terms: Aggregate

The key encapsulation mechanism of Section 4.2 requires the computation of

$$P_r = \left[\frac{1}{(\gamma + x_1)\dots(\gamma + x_r)}\right] H \in \mathbb{G}_2 \quad \text{and} \quad K' = e\,(G, H)^{\frac{1}{(\gamma+x_1)\cdots(\gamma+x_r)}} \in \mathbb{G}_T$$

given $\{\mathsf{lab}_j = (x_j, B_j, V_j)\}_{j=1}^r$ where the x_j's are pairwise distinct. We proceed to describe how $\mathsf{Aggregate}(\mathbb{G}_2, \cdots)$ allows to compute P_r from the x_j's and the B_j's. The same algorithm applies over $\mathbb{G}_T$ as well to compute K' from the x_j's and the V_j's.

Description. Given $x_1, \dots, x_r$ and $B_j = \left[\frac{1}{\gamma+x_j}\right] H$ for $1 \leq j \leq r$, let us define for any (j, ℓ) such that $1 \leq j < \ell \leq r$,

$$P_{j,\ell} = \left[\frac{1}{\prod_{\kappa=1}^{j}(\gamma + x_\kappa)}\right] B_\ell = \left[\frac{1}{(\gamma + x_\ell)} \cdot \frac{1}{\prod_{\kappa=1}^{j}(\gamma + x_\kappa)}\right] H\ .$$

The $\mathsf{Aggregate}$ algorithm consists in computing sequentially $P_{j,\ell}$ for $j = 1, \dots, r-1$ and $\ell = j+1, \dots, r$ using the induction

$$P_{j,\ell} = \left[\frac{1}{x_\ell - x_j}\right] (P_{j-1,j} - P_{j-1,\ell}) \tag{1}$$

and posing $P_{0,\ell} = B_\ell$ for $\ell = 1, \dots, r$. The algorithm finally outputs $P_r = P_{r-1,r}$. $\mathsf{Aggregate}$ is displayed in more detail on Fig. 2. It is easily shown that Equ. 1 is sound. Note however that computing $P_{j,\ell}$ by Equ. 1 assumes $x_j \neq x_\ell$. If $x_j = x_\ell$ for some $j < \ell$ then $P_{j-1,j} = P_{j-1,\ell}$ and $P_{j,\ell}$, which then contains a $(\gamma + x_j)^2$ factor, cannot be computed (this is computationally infeasible without γ). In this case, we force $\mathsf{Aggregate}$ to abort and return an error symbol.

Complexity. One sees that the computation of a term of the form

$$\left[\frac{1}{\prod_{j\in R}(\gamma + x_j)}\right] H$$

for any subset $R \subseteq \{1, \dots, n\}$ from the 1-degree terms $[1/(\gamma + x_j)]\,H$ where $j \in R$ is quadratic in the cardinality of R. More precisely, generalizing to $\mathbb{G} \in \{\mathbb{G}_2, \mathbb{G}_T\}$:

$$\mathsf{Time}\,[\mathsf{Aggregate}(\mathbb{G}, r \text{ terms})] \simeq \frac{r(r-1)}{2} \cdot (\tau_p + \tau_{\mathbb{G}})\ ,$$

where τ_p is the execution time of a subtraction and an inversion modulo $p = |\mathbb{G}|$ and $\tau_{\mathbb{G}}$ the total time of a division and an exponentiation in $\mathbb{G}$.

4.4 Aggregation of terms of increasing degree: Aggregate′

Description. The Aggregate algorithm can be accelerated if one is given $P_j = P_{j-1,j}$ for $j = 1, \ldots, r$ instead of B_j. It is easily seen that Equ. 1 provides a way to compute $P_1, \ldots, P_r$ from $B_1, \ldots, B_r$ (and vice versa) in quadratic time. So knowing either vector is quadratic-time equivalent. However, given $P_1, \ldots, P_r$ and $(x_i, B_i = \left[\frac{1}{\gamma + x_i}\right] H)$, Equ. 1 is reformulated as

$$\left[\frac{1}{(\gamma + x_i)\prod_{\kappa=1}^{j}(\gamma + x_\kappa)}\right] H = \left[\frac{1}{x_i - x_j}\right]\left(P_j - \left[\frac{1}{(\gamma + x_i)\prod_{\kappa=1}^{j-1}(\gamma + x_\kappa)}\right] H\right)$$

for any $j = 1, \ldots, r$. Note that the left-hand term gives $B_{i,\mathcal{R}}$ when $j = r$. This leads us to the iterative computation of $B_{i,\mathcal{R}}$ as depicted on Fig. 2.
Complexity. It is easily seen that Time [Aggregate′$(\mathbb{G}, r$ terms$)] \simeq r \cdot (\tau_p + \tau_{\mathbb{G}})$.

Aggregate	Aggregate′
Input: two r-arrays $x = [x_1, \ldots, x_r]$ and $P = [B_1, \ldots, B_r]$ Output: P_r as defined above or $\perp$	Input: x_i, B_i, $x = [x_1, \ldots, x_r]$ and $P = [P_1, \ldots, P_r]$ Output: $B_{i,\mathcal{R}}$ as defined above or $\perp$
1. for $j = 1$ to $r - 1$ (a) for $\ell = j + 1$ to r i. if $x[j] = x[\ell]$ output $\perp$ ii. $P[\ell] = \left[\frac{1}{x[\ell] - x[j]}\right](P[j] - P[\ell])$ 2. output $P[r]$	1. set $temp = B_i$ 2. for $j = 1$ to r (a) if $x_i = x[j]$ output $\perp$ (b) set $temp = \left[\frac{1}{x_i - x[j]}\right](P[j] - temp)$ 3. output $temp$

Fig. 2. The Aggregate and Aggregate′ algorithms

5 Security Analysis

5.1 Security Reduction to (t, n)-GDHE

We prove the semantic security of our system by reformulating the security game in terms of sequences of polynomials and relying on the GDHE/GDDHE framework of [1]. We start by defining the following intermediate computational problem.

Definition 6 ((t, n)-GDHE)**.** Let $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ be a bilinear map group system and let f and g be the two random univariate polynomials

$$f(X) = \prod_{i=1}^{t}(X + x_i) = \sum_{i=0}^{t}\mu_i X^i\,, \qquad g(X) = \prod_{i=t+1}^{n}(X + x_i) = \sum_{i=0}^{n-t}\nu_i X^i\,,$$

where all the x_i's are random and pairwise distinct elements of $\mathbb{Z}_p^\star$. Let G_0 be a generator of $\mathbb{G}_1$ and H_0 a generator of $\mathbb{G}_2$. Solving the (t, n)-GDHE problem

consists, given

$$\begin{array}{lll} G_0\,, [\gamma]\,G_0\,, \ldots, [\gamma^{t-1}]\,G_0\,, & [\gamma \cdot f(\gamma)]\,G_0\,, & [k \cdot \gamma \cdot f(\gamma)]\,G_0\,, \\ H_0\,, [\gamma]\,H_0\,, \ldots, [\gamma^{n}]\,H_0\,, & & [k \cdot g(\gamma)]\,H_0\,, \\ e\,(G_0, H_0)^{f^2(\gamma) \cdot g(\gamma)}\,, & & \end{array}$$

in computing $e\,(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$.

As usual, we denote by $\mathsf{Succ}^{\mathsf{gdhe}}(t, n, \mathcal{A})$ the success probability of a randomized algorithm $\mathcal{A}$ solving (t, n)-GDHE. Similarly to the above, one defines the decisional version of (t, n)-GDHE, which we call (t, n)-GDDHE. In the (t, n)-GDDHE game, an additional input $T \in \mathbb{G}_T$ is provided and solving (t, n)-GDDHE consists in deciding whether $T = e\,(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$. We then denote by $\mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A})$ the advantage of an algorithm $\mathcal{A}$ in distinguishing the two distributions and set $\mathsf{Adv}^{\mathsf{gddhe}}(t, n) = \max_{\mathcal{A}} \mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A})$ over $\mathsf{poly}(|p|)$-time $\mathcal{A}$'s. The following statement is a corollary of Theorem 9 that can be found in Section 5.2.

Corollary 7 (Generic security of (t, n)-GDDHE). *For any probabilistic algorithm $\mathcal{A}$ that totalizes at most q queries to the oracles performing group operations in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and evaluations of the bilinear map $e\,(\cdot, \cdot)$,*

$$\mathsf{Adv}^{\mathsf{gddhe}}(t, n, \mathcal{A}) \leq \frac{(q + 2(n + t + 4) + 2)^2 \cdot (t + n)}{2p}\,.$$

Let $\mathcal{DBE}$ denote our construction (Construction 1) as per Section 4. We state:

Theorem 8. *For any n, t such that $0 \leq t \leq n$, one has $\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t, n) \leq 2 \cdot \mathsf{Adv}^{\mathsf{gddhe}}(t, n)$.*

Proof. The rest of this section is dedicated to proving Theorem 8. To establish the semantic security of $\mathcal{DBE}$ against static adversaries, we assume to be given an adversary $\mathcal{A}$ breaking $\mathcal{DBE}$ under a (t, n)-collusion and we build a reduction algorithm $\mathcal{B}$ that distinguishes the two distributions of the GDDHE decision problem.

Generation of system parameters and initial ek. The reduction algorithm $\mathcal{B}$ is given as input a group system $\mathbb{S} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\,(\cdot, \cdot))$ and a (t, n)-GDDHE instance in $\mathbb{S}$. Let then $f(X) = \prod_{i=1}^{i=t}(X + x_i)$ and $g(X) = \prod_{i=1}^{i=n-t}(X + x'_i)$ be two random polynomials of respective degree t and $n - t$ with non-zero pairwise distinct roots. $\mathcal{B}$ is also given a generator G_0 (resp. H_0) of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) and

$$\begin{array}{lll} G_0, [\gamma]\,G_0, \ldots, [\gamma^{t-1}]\,G_0 & [\gamma \cdot f(\gamma)]\,G_0 & [k \cdot \gamma \cdot f(\gamma)]\,G_0 \\ H_0, [\gamma]\,H_0, \ldots, [\gamma^{n}]\,H_0 & & [k \cdot g(\gamma)]\,H_0 \\ e\,(G_0, H_0)^{f^2(\gamma) \cdot g(\gamma)}\,, & & \end{array}$$

as well as $T \in \mathbb{G}_T$ which is either equal to $e\,(G_0, H_0)^{k \cdot f(\gamma) \cdot g(\gamma)}$ or to some random element of $\mathbb{G}_T$. $\mathcal{B}$ formally sets $G = [f(\gamma)]\,G_0$ (i.e. without computing it) and

computes

$$\begin{array}{ll} H = [f(\gamma) \cdot g(\gamma)]\, H_0 & (\text{from } H_0, [\gamma]\, H_0, \ldots, [\gamma^n]\, H_0) \\ W = [\gamma]\, G = [\gamma \cdot f(\gamma)]\, G_0 & (\text{given as input}) \\ V = e\,(G_0, H_0)^{f^2(\gamma)\cdot g(\gamma)} = e\,(G, H) & (\text{given as input}) \end{array}$$

$\mathcal{B}$ then defines the group encryption key as $\mathsf{ek} = (H, W, V)$. Note that $\mathcal{B}$ can by no means compute the value of G. $\mathcal{B}$ then runs $\mathcal{A}$ on the system parameters $(\mathbb{S}, H)$ and ek.

Generation of corrupted keys. The adversary $\mathcal{A}$ chooses a t-subset $\mathcal{C} \subseteq \{1, \ldots, n\}$ that indicates which users are corrupted: the users whose index lies in $\mathcal{C}$ are corrupted, whereas the users with index in $\bar{\mathcal{C}} = \{1, \ldots, n\} \setminus \mathcal{C}$ are honest. To generate the keys of the corrupted users, we define $f_i(X) = f(X)/(X + x_i)$ for $i \in [1, t]$, which are polynomials of degree $t - 1$, and then set

$$A_i = [x_i \cdot f_i(\gamma)]\, G_0 = \left[\frac{x_i}{\gamma + x_i}\right] G\,, \qquad B_i = [f_i(\gamma) \cdot g(\gamma)]\, H_0 = \left[\frac{1}{\gamma + x_i}\right] H$$

and

$$V_i = e\,(G_0, H_0)^{f_i(\gamma)\cdot f(\gamma)\cdot g(\gamma)} = e\,([f(\gamma)]\, G_0, [f(\gamma) \cdot g(\gamma)]\, H_0)^{\frac{1}{\gamma + x_i}} = e\,(G, H)^{\frac{1}{\gamma + x_i}}\,.$$

To generate the labels of honest users, we define $g_i(X) = g(X)/(X + x'_i)$ for $i \in [1, n - t]$ and

$$B'_i = [f(\gamma) \cdot g_i(\gamma)]\, H_0 = \left[\frac{1}{\gamma + x'_i}\right] H\,, \qquad V'_i = e\,(G_0, H_0)^{f^2(\gamma)\cdot g_i(\gamma)}\,.$$

Note that the total number of users is n. Since the polynomials $f_i g$ and $f g_i$ are of degree $n-1$, B_i and B'_i can be computed easily. V_i can also be computed given $[\gamma^j]\, G_0$ for $j \in [0, t-1]$ and $[\gamma^j]\, H_0$ for $j \in [0, n]$ since these allow to compute $e\,(G_0, H_0)^{\gamma^k}$ for $k \in [0, n + t - 1]$ and since polynomials $f_i f g$ and $f^2 g_i$ are of degree $n+t-1$. The user labels and keys $((x_i, A_i, B_i), (x_i, V_i, B_i))$ for $i \in [1, t]$ as well as the user labels (x'_i, B'_i, V'_i) for $i \in [1, n-t]$ are then given to the adversary $\mathcal{A}$.

Challenge broadcast ciphertext. $\mathcal{B}$ now builds a header $\mathsf{hdr} = (C_1, C_2, \{x_i, B_i\}_{i=1}^t)$ decryptable by the honest users i.e. where the users $\mathcal{C} \subseteq \mathcal{U}$ are revoked. Given the GDDHE instance, $\mathcal{B}$ computes

$$\begin{aligned} C_1 &= [k \cdot \gamma \cdot f(\gamma)]\, G_0 = [k]\, W\,, \\ C_2 &= [k \cdot g(\gamma)]\, H_0 = \left[\frac{k}{f(\gamma)}\right] H = \left[\frac{k}{(\gamma + x_1) \ldots (\gamma + x_t)}\right] H\,. \end{aligned}$$

Doing so, $\mathcal{B}$ implicitly defines the message encryption key K as

$$K = e\,(G, H)^{\frac{k}{(\gamma + x_1)\cdots(\gamma + x_t)}} = e\,(G_0, H_0)^{\frac{k \cdot f^2(\gamma)\cdot g(\gamma)}{f(\gamma)}} = e\,(G_0, H_0)^{k \cdot f(\gamma)\cdot g(\gamma)}\,.$$

$\mathcal{B}$ now selects a random bit $b \leftarrow \{0, 1\}$, sets $K_b = T$ and sets K_{1-b} to a random element of $\mathbb{G}_T$. $\mathcal{B}$ sends the tuple (hdr, K_0, K_1) to $\mathcal{A}$.

Final outcome. $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. $\mathcal{B}$ then outputs real if $b' = b$ or random otherwise. One has

$$\begin{aligned}\mathsf{Adv}^{\mathsf{gddhe}}(t,n,\mathcal{B}) &= \Pr[b'=b|\mathsf{real}] - \Pr[b'=b|\mathsf{random}] \\ &= \frac{1}{2} \times (\Pr[b'=1|b=1 \wedge \mathsf{real}] - \Pr[b'=1|b=0 \wedge \mathsf{real}]) \\ &\quad -\frac{1}{2} \times (\Pr[b'=1|b=1 \wedge \mathsf{random}] + \Pr[b'=1|b=0 \wedge \mathsf{random}]) .\end{aligned}$$

Now in the random case, the distribution of b is independent from the adversary's view wherefrom

$$\Pr[b'=1|b=1 \wedge \mathsf{random}] = \Pr[b'=1|b=0 \wedge \mathsf{random}] .$$

In the real case however, the distributions of all variables defined by $\mathcal{B}$ perfectly comply with the semantic security game since all simulations are perfect. Therefore

$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(\mathcal{A}) = \Pr[b'=1|b=1 \wedge \mathsf{real}] - \Pr[b'=1|b=0 \wedge \mathsf{real}] .$$

Summing up, we get that $\mathsf{Adv}^{\mathsf{gddhe}}(t,n,\mathcal{B}) = \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{DBE}}(t,n,\mathcal{A})/2$. It is obvious that $\mathcal{B}$ runs in time similar to the one of $\mathcal{A}$. □

5.2 Proving the Intractability of (t,n)-GDHE

In this section, we prove the intractability of distinguishing the two distributions involved in the (t,n)-GDDHE problem cf. Corollary 7. We first review known results on the General Diffie-Hellman Exponent problem from [1]. Sticking to the most general security result, we assume a bilinear map system of the most favorable type for the adversary i.e. $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ or there is an efficient isomorphism going both ways between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Theorem 9 ([1]). *Let $P, Q \in \mathbb{F}_p[X_1,\ldots,X_m]$ be two s-tuples of m-variate polynomials over $\mathbb{F}_p$ and let $F \in \mathbb{F}_p[X_1,\ldots,X_m]$. Let d_P (resp. d_Q, d_F) denote the maximal degree of elements of P (resp. of Q, F) and pose $d = \max(2d_P, d_Q, d_F)$. If $F \notin \langle P, Q\rangle$ then for any generic-model adversary $\mathcal{A}$ totalizing at most q queries to the oracles (group operations in $\mathbb{G}, \mathbb{G}_T$ and evaluations of e) which is given $H(x_1,\ldots,x_m)$ as input and tries to distinguish $g^{F(x_1,\ldots,x_m)}$ from a random value in $\mathbb{G}_T$, one has*

$$\mathsf{Adv}(\mathcal{A}) \leq \frac{(q+2s+2)^2 \cdot d}{2p} .$$

Proof (of Corollary 7). In order to conclude with Corollary 7, we need to prove that the (t,n)-GDHE problem lies in the scope of Theorem 9. As already said, we consider the weakest case $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and thus pose $H_0 = [\alpha]\,G_0$. In the (t,n)-GDHE problem, if one replaces γ by x, k by y and α by z, we see that our problem is reformulated as (P,Q,F)-GDHE where

$$\begin{aligned}P &= \begin{pmatrix} 1, x, x^2, \ldots, x^{t-1}, & x \cdot f(x), & y \cdot x \cdot f(x) \\ z, z \cdot x, \ldots, z \cdot x^n, & y \cdot z \cdot g(x) & \end{pmatrix} \\ Q &= (1, z \cdot f(x)^2 g(x)) \\ F &= y \cdot z \cdot f(x) g(x),\end{aligned}$$

and thus $m = 3$ and $s = t + n + 4$. We have to show that F is independent of (P, Q) i.e. that no coefficients $\{a_{i,j}\}_{i,j=1}^{s}$ and $\{b_1, b_2\}$ exist such that $F = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{2} b_k q_k$ where the polynomials p_i and q_k are the one listed in P and Q above. By making all possible products of two polynomials from P which are multiples of $y \cdot z$, we want to prove that no linear combination among the polynomials from the list R below leads to F:

$$R = \begin{pmatrix} z \cdot y \cdot x^{n+1} \cdot f(x), \ldots, \ z \cdot y \cdot x^2 \cdot f(x), \ z \cdot y \cdot x \cdot f(x), \\ z \cdot y \cdot x^{t-1} \cdot g(x), \ldots, \ z \cdot y \cdot x \cdot g(x), \ z \cdot y \cdot g(x), \\ z \cdot y \cdot x \cdot f(x)g(x) \end{pmatrix} .$$

Note that the last polynomial can be written as $z \cdot y \cdot x \cdot f(x)g(x) = \sum_{i=0}^{i=n-t} \nu_i \cdot z \cdot y \cdot x^{i+1} \cdot f(x)$, and thus as a linear combination of the polynomials from the first line. Moreover, $z \cdot y \cdot x^{n+1} \cdot f(x)$ is the unique polynomial of degree $t+n+1$ in x and such a monomial does not appear in F. Similarly, $z \cdot y \cdot x^n \cdot f(x)$, ..., $z \cdot y \cdot x^{n-t+1} \cdot f(x)$ cannot appear in the combination. We therefore simplify the task to refuting a linear combination of elements of the list R' below which leads to $f(x)g(x)$:

$$R' = \begin{pmatrix} x^{n-t} \cdot f(x), \ldots, \ x^2 \cdot f(x), \ x \cdot f(x), \\ x^{t-1} \cdot g(x), \ldots, \ x \cdot g(x), \ g(x) \end{pmatrix} .$$

Any such linear combination can be written as

$$f(x)g(x) = A(x) \cdot f(x) + B(x) \cdot g(x)$$

where A and B are polynomials such that $A(0) = 0$, $\deg A = n - t$ and $\deg B = t - 1$. Since f and g are coprime by assumption, we must have $f \mid B$. Since $\deg f = t$ and $\deg B = t - 1$ this implies $B = 0$. Hence $A = g$ resulting in that $g(0) = \prod_{i=1}^{n-t} x'_i = 0$ which contradicts $x'_i \neq 0$ for $i \in [1, n-t]$. □

6 Related Constructions

From a design perspective, our main construction is fairly simple and can therefore be refined in many ways. We now present a few broadcast encryption systems derived from Construction 1. Although we do not provide proof details here, it can be shown that all constructions inherit full collusion resistance under the GDDHE assumption from Construction 1.

6.1 Public-Key Broadcast Encryption with Constant-Size Ciphertexts (Construction 2)

We suggest a simple variant of the previous construction that reaches optimally short ciphertexts i.e. $\lambda_c = O(1)$. The basic observation is that the fraction of the ciphertext containing $\{(x_j, P_j)\}_{j=1}^{r}$ can be computed from $\mathcal{R}$ and $\{(x_j, B_j)\}_{j=1}^{r}$ which is a subset of the public encryption key. Therefore P_j for $j \in \mathcal{R}$ can be removed completely from hdr if the complete set $\{(x_1, B_1), \ldots, (x_n, B_n)\}$ is made available at decryption time. Construction 2 therefore redefines

$$\mathsf{hdr} = (C_1, C_2) \qquad \text{where} \qquad C_1 = [k]\, W \ , \quad C_2 = [k]\, H \ ,$$

and

$$\mathsf{dk}_i = (\mathbb{S}, (x_i, A_i, B_i), (x_1, B_1), \ldots, (x_n, B_n)) \ .$$

As a result, the new system enjoys constant-size ciphertexts at the expanse of linear-size user decryption keys. Decryption can be performed as

$$K = e\left(C_1, B_{i,\mathcal{R}}\right) \cdot e\left(A_i, C_2\right)$$

where $B_{i,\mathcal{R}}$ is computed in time $O(r^2)$ using the $\mathsf{Aggregate}$ algorithm with the list $\{(x_j, B_j)\}_{j\in\mathcal{R}} \subseteq \mathsf{dk}_i$ and $(x_i, B_i) \subseteq \mathsf{dk}_i$ as inputs. Note here that the presence of $\mathcal{R}$ in the full broadcast ciphertext is mandatory to allow proper decryption[5].

We also note that the dynamic aspect is lost when doing these changes since all users have to update their decryption key whenever a new member joins the group. It is worthwhile noting that the characteristics of this construction are exactly those of BGW_1. Our scheme is slightly less efficient since decryption keys are bigger than those of in BGW_1 by the inclusion of n extra integers modulo p.

6.2 Private-Key DBE with Constant-Size Decryption & Encryption Keys (Construction 3)

In this scheme, the broadcaster and the group manager are the same entity. All algorithms are unchanged except for the join procedure which now exclusively employs the encryption key ek, mk being obsolete. The random selection of $x_i, i = 1, \ldots, n$ is now replaced by a pseudo-random generation. More precisely, we modify Construction 1 as follows:

1. $\mathsf{Setup}(\lambda)$ generates $\mathbb{S}$, randomly selects a hash function $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_p^\star$ and outputs $\mathsf{ek} = (\mathbb{S}, G, H, \gamma, W, V, \mathcal{H})$,
2. $\mathsf{Join}(\mathsf{ek}, i)$ sets $x_i = \mathcal{H}(i)$, computes A_i and B_i as in Construction 1 and outputs the user decryption key $\mathsf{dk}_i = (\mathbb{S}, x_i, A_i, B_i)$,
3. $\mathsf{Encrypt}(\mathsf{ek}, \mathcal{R})$ picks a random $k \leftarrow \mathbb{Z}_p^\star$ and computes on the spot

$$\frac{1}{\gamma + x_1}, \quad \frac{1}{(\gamma + x_1)(\gamma + x_2)}, \quad \ldots, \quad \frac{1}{(\gamma + x_1)\ldots(\gamma + x_r)},$$

$$\frac{k}{(\gamma + x_1)\ldots(\gamma + x_r)}, \quad k\gamma$$

   over $\mathbb{Z}_p^\star$, thereby allowing to compute $P_1, \ldots, P_r, C_2, C_1, K$ directly by exponentiating H, V or W in their respective groups,
4. decryption is identical to the one of Construction 1.

The main difference with Construction 1 is that the broadcaster needs no linear-size group encryption key nor user labels: these are recovered whenever necessary using G, H, γ and the user counter i. Therefore $\lambda_{\mathsf{ek}} = O(1)$ and both encryption and decryption stages process in time $O(r)$. Note that all efficiency measures do not depend on the number of users n.

[5] Including $\mathcal{R}$ in the full header was unnecessary in Construction 1, see Remark 1.

6.3 Realizing Long-term Revocation

To revoke user i, the group manager broadcasts (x_i, B_i, V_i) and resets $H := B_i$ and $V := V_i$. A non revoked user $j \neq i$ can update his label as

$$B_j := \left[\frac{1}{\gamma + x_i}\right] B_j = \left[\frac{1}{x_j - x_i}\right] (B_i - B_j) \qquad V_j := V_j^{\frac{1}{\gamma + x_i}} = \left(\frac{V_i}{V_j}\right)^{\frac{1}{x_j - x_i}} .$$

Therefore the broadcaster does not have to repeat (x_i, B_i, V_i) in future encryptions. In virtue of the full collusion resistance of Construction 1, long-term revocations can be shown to be forward-secure.

7 Conclusion

We introduced new alternatives to design public and private key fully collusion secure broadcast encryption. Our designs support the inclusion of new users at minimal cost and achieve the best known security level relative to a generically secure computational problem via tight reductions in the standard model. We leave as an open problem to realize dynamic public-key broadcast encryption with an encryption key substantially shorter that $O(n)$. Resisting fully adaptive adversaries would also be a significant improvement. Finally, we expect our trapdoor mechanism to find other cryptographic applications in the future.

References

1. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer-Verlag, Berlin, Germany. Extended version available at `http://eprint.iacr.org/2005/015`.
2. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.
3. Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany.
4. Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 100–115, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany.
5. Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491, Santa Barbara, CA, USA, August 22–26, 1994. Springer-Verlag, Berlin, Germany.
6. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 53–68, Kamakura, Japan, March 1–3, 1999. Springer-Verlag, Berlin, Germany.
7. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer-Verlag, Berlin, Germany.
8. Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 511–527, Santa Barbara, CA, USA, August 15–19, 2004. Springer-Verlag, Berlin, Germany.
9. Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 47–60, Santa Barbara, CA, USA, August 18–22, 2002. Springer-Verlag, Berlin, Germany.

10. Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS*, pages 385–394, 2000.
11. Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from computational diffie-hellman in cryptographic groups. *Journal of Cryptology*, 16(4):239–247, 2003.
12. Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In *ICISC*, pages 90–108, 2000.
13. Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. New explicit conditions of elliptic curve traces for fr-reduction. *IEICE Transactions on Fundamentals*, E84-A(5):1234–1243, 2001.
14. Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62, Santa Barbara, CA, USA, August 19–23, 2001. Springer-Verlag, Berlin, Germany.
15. Tatsuaki Okamoto and David Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–175, San Francisco, CA, USA, April 8–12, 2001. Springer-Verlag, Berlin, Germany.

A BGW_1 and $\mathsf{BGW'}_1$

We briefly describe the system BGW_1 put forward by Boneh et al. [3]. As BGW_1 is not dynamic, the Setup algorithm takes as input the number n of users in addition to the security parameter λ, and is in charge of computing the user decryption keys since there is no Join algorithm.

$\mathsf{Setup}(\lambda, n)$. Given the security parameter λ, a symmetric bilinear map group system

$$\mathbb{S} = (p, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$$

is constructed such that $|p| = \lambda$. The algorithm first picks a generator $G \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $G_i = [\alpha^i] G \in \mathbb{G}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \leftarrow \mathbb{Z}_p$ and sets $W = [\gamma] G \in \mathbb{G}$. The encryption key is

$$\mathsf{ek} = (\mathbb{S}, G, G_1, \ldots, G_n, G_{n+2}, \ldots, G_{2n}, W) \in \mathbb{G}^{2n+1} .$$

The decryption key of user $i \in \{1, \ldots, n\}$ is set as $\mathsf{dk}_i = [\gamma] G_i \in \mathbb{G}$. The algorithm outputs the encryption key ek and the n decryption keys $\mathsf{dk}_1, \ldots, \mathsf{dk}_n$.

$\mathsf{Encrypt}(\mathsf{ek}, S)$. Given the encryption key ek and a subset $S \subseteq \{1, \ldots, n\}$ of users, the broadcaster randomly picks $k \leftarrow \mathbb{Z}_p^\star$, sets $K = e(G_{n+1}, G)^k = e(G_n, G_1)^k \in \mathbb{G}_T$ and computes

$$\mathsf{hdr} = \left([k] G,\ [k] \left(W + \sum_{j \in S} G_{n+1-j} \right) \right) \in \mathbb{G}^2$$

and outputs (hdr, K).

$\mathsf{Decrypt}(\mathsf{dk}_i, \mathsf{ek}, S, \mathsf{hdr})$. In order to retrieve the message encryption key K encapsulated in the header $\mathsf{hdr} = (C_1, C_2)$, the i-th user computes

$$K = e(G_i, C_2) \ / \ e(\mathsf{dk}_i + P_i, C_1) \quad \text{where} \quad P_i = \sum_{j \in S, j \neq i} G_{n+1-j+i} .$$

We define $\mathsf{BGW'}_1$ as a modification of BGW_1 where the decryption algorithm does not take ek as input parameter anymore but directly includes $\{P_i\}_{i \in S}$ in the header hdr instead. This change excepted, the encryption and decryption algorithms are unchanged. The motivation for considering $\mathsf{BGW'}_1$ is that the decryption key material at the user side does not include the encryption key ek, resulting in that $\lambda_{\mathsf{dk}} = O(1)$ instead of $\lambda_{\mathsf{dk}} = O(n)$ in BGW_1. This comes at the cost of a ciphertext size in $O(n - r)$ instead of $O(1)$ in BGW_1.