Security Notions for Broadcast Encryption

Duong Hieu Phan[1,2], David Pointcheval[2], and Mario Strefler[2]

[1]LAGA, University of Paris 8
[2]ENS / CNRS / INRIA

Abstract. This paper clarifies the relationships between security notions for broadcast encryption. In the past, each new scheme came with its own definition of security, which makes them hard to compare. We thus define a set of notions, as done for signature and encryption, for which we prove implications and separations, and relate the existing notions to the ones in our framework. We find some interesting relationships between the various notions, especially in the way they define the receiver set of the challenge message. In addition, we define a security notion that is stronger than all previous ones, and give an example of a scheme that fulfills this notion.

Keywords: Broadcast Encryption, Adaptive Security, Security Models.

1 Introduction

Broadcast encryption (BE) is a cryptographic primitive that was first described by Fiat and Naor in [FN94]. It provides a content holder the ability to publish the content to a specific subset of the registered users. This is used in practice by copyright protection mechanisms for digital media such as DVDs, so that if the keys for a series of DVD players become known, this series will not be able to play DVDs produced after the series is revoked [NNL01]. But while work on the related topic of multi-cast encryption progressed, BE did not receive much attention until the last decade, when Naor, Naor, and Lotspiech presented their (symmetric-key) subset-cover framework along with a security model and a security analysis [NNL01]. Since then, many BE schemes have been proposed, but for each scheme the security proof was done in a new security model. Because of these various and often *ad-hoc* security models, it is hard to compare the merits of these schemes, as it is not always clear how the security notions relate to each other.

Gentry and Waters [GW09], for example, defined a security notion they call "adaptive", because the adversary can corrupt users adaptively before the challenge phase. But there is a notion that is "even more" adaptive, where the adversary can still corrupt users *after* the challenge phase. The goal of this paper is thus to provide a better picture of the meaningful security models for BE, and to compare them. In particular, we investigate whether the various adaptive notions of corruption coincide or not.

Related Work. The first scheme to come with a security argument was the subset-cover framework introduced by Naor, Naor, and Lotspiech [NNL01]. The framework uses symmetric keys, where the sender and the receivers share some secrets, so the security proof relies on assumptions about the symmetric primitives (one-way functions and block ciphers). Dodis and Fazio [DF03] presented the first CCA2-secure public-key Trace and Revoke (TR) scheme along with a security model covering CCA2 and generalized CCA2. When one considers possible corruption after the target ciphertext has been sent, one has to deal with forward-secrecy. This was done by Yao, Fazio, Dodis, and Lysyanskaya [YFDL04] who first considered forward-secrecy for HIBE and then by extension for BE. Boneh, Gentry, and Waters [BGW05] designed a fully collusion-resistant BE scheme and proposed a security model for it, where the adversary can corrupt all the users, except the target users. Thereafter, Boneh and Waters [BW06] presented a fully collusion-resistant TR scheme secure against adaptive attacks. Delerablée, Paillier, and Pointcheval [DPP07] also presented a fully collusion-secure dynamic BE scheme (DBE) and presented a new matching security model. More recently, Gentry and Waters [GW09] defined two additional security notions they call "semi-static" and "adaptive", as well as a generic transformation from a semi-static secure scheme into an adaptively secure scheme, and then a semi-static secure scheme to which they later apply the transformation.

Contribution. As shown above, many security notions were proposed in the literature. In this paper, we define a more systematic security model for broadcast encryption schemes, and construct a generic security framework for BE. We take into account, as usual in the "provable security framework", oracles to model the means available to the adversary, such as the possibility to join new users, to corrupt users, and to decrypt messages. It is worth noting that small details can have a high impact. For example, the choice of the set of users to which the challenge message is sent also plays a role in how the models relate to each other. We investigate the relationships between the different notions, and find that in some cases, two notions are equivalent or separated depending on the availability of some oracles or the collusion-resistance of a BE scheme. After describing the relationships between notions in our framework, we have a closer look at the security models and the schemes proposed in the literature, and discuss where they are in our framework, which then helps to compare them.

Our results are relevant for several existing BE schemes. For example, from the proof found in [GW09], it is clear that the two-key transformation actually achieves the stronger 2-adaptive-security level. We also examine the proof found in [DPP07], and see that it can fulfill a stronger security notion where the adversary can choose the target set, and then the scheme meets the requirements from [GW09]. This means that it can be made 2-adaptively secure using the generic transformation (although not efficiently).

Organization. In section 2 we provide a formal definition of broadcast encryption, or more precisely key encapsulation, and specify some terminology. In section 3 we define our security framework. Section 4 relates the different security notions to each other. In section 5 we embed the existing security models from the literature into our framework. In section 6, we describe which security notions have been achieved by existing protocols and describe an (inefficient) protocol that achieves the strongest notion.

2 Definitions

Broadcast encryption (BE) schemes enable the sender of a message to specify a subset of the registered users (the *target set* or *privileged set*), who will be able to decrypt the ciphertext sent to all users via a broadcast channel. The complement of the target set (in the set of the registered users) is called the *revoked set.* To accomplish user revocation when sending a message, a BE generally generates three parts: the *Id Header*, that is a bit-string that unambiguously identifies the target set/revoked set; the *Key Header*, that encapsulates a session key for the privileged users; and the *Message Body*, that contains the payload encrypted under the session key.

Since for all the schemes, the *Id Header* and the *Message Body* are similar, in this paper, we will focus on the *Key Header* part only, which can be seen as a key encapsulation mechanism (KEM). Furthermore, when no more information is given, we will consider a public-key key encapsulation system with possibly stateful decoders: encryption key is public, the decryption keys of the users can evolve, but the updates will be global and sent on a public channel, and ephemeral keys are distributed to be used together with symmetric encryption (DEM: Data Encapsulation Mechanism). We will nevertheless sometimes make remarks about alternative cases.

Definition 1 (Dynamic Broadcast Encapsulation). *A dynamic broadcast encapsulation scheme is a tuple of algorithms* $\mathcal{DBE} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Encaps}, \mathsf{Decaps})$*:*

- $\mathsf{Setup}(1^k)$*, where* k *is the security parameter, generates the global parameters* param *of the system (omitted in the following); and returns a master secret key* MSK *and an encryption key* EK*. It also initiates an empty list Reg. If the scheme is asymmetric,* EK *is public, otherwise it can be seen as a part of the* MSK*.*
- $\mathsf{Join}(\mathsf{MSK}, Reg, \mathsf{id})$ *takes as input the master secret key, the list Reg, and a user identifier* id*. If* $\mathsf{id} \in \mathcal{UI}$ *(where* $\mathcal{UI}$ *is the set of valid user identifiers, usually* $\mathbb{N}$*) and* $\mathsf{id} \notin Reg$*, outputs a user secret key* $\mathsf{usk}_{\mathsf{id}}$ *and a public user tag* $\mathsf{upk}_{\mathsf{id}}$*. The pair* $(\mathsf{id}, \mathsf{upk}_{\mathsf{id}})$ *is appended to Reg. Else, outputs* $\perp$*.*
- $\mathsf{Encaps}(\mathsf{EK}, Reg, S)$ *takes as input the encryption key, the list Reg, and a target set* S *and outputs a key header* H *and a session key* $K \in \{0,1\}^k$*.*
- $\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H)$ *takes as input a user secret key, the target set* S*, and the key header* H*. If* $\mathsf{id} \in S$*, outputs the session key* K*.*

The correctness requirement is that for any (polynomial size) set of joined users $U \subset \mathcal{UI}$, any target set $S \subset U$ and for any $\mathsf{id} \in \mathcal{UI}$, if $\mathsf{id} \in S$ then the decapsulation algorithm gives back the ephemeral session key. See figure 1.

```
$(\mathsf{MSK}, \mathsf{EK}, Reg) \leftarrow \mathsf{Setup}(1^k)$;
for all  $\mathsf{id} \in S$ :  $(\mathsf{usk}_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}}, Reg) \leftarrow \mathsf{Join}(\mathsf{MSK}, Reg, \mathsf{id})$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg, S)$.

$i \in S \Rightarrow \mathsf{Decaps}(\mathsf{usk}_i, S, H) = K$
```

Fig. 1. $\mathcal{DBE}$: Correctness

2.1 Terminologies and Various Types of Schemes

Join Algorithms. When the Join algorithm can be run at the setup phase only, with no later evolution of the group, we say the scheme is *static* (instead of *dynamic*). For a dynamic scheme, several kinds of Join functionalities are possible:

Passive, no input (except a counter i); it generates a public tag upk_i to identify the user;
Active, the input is id; it generates a public tag $\mathsf{upk}_{\mathsf{id}}$ to identify the user;
Identity-Based, the input is id, and the public tag $\mathsf{upk}_{\mathsf{id}}$ is simply id.

We stress that the default case in this paper (when no other version is specified) is that Join is passive.

Target Set. A broadcast encryption scheme is called *inclusive* when the target set is specified by the list of authorized users, and *exclusive* when the target set is specified by its complement $\mathcal{R}$, the set of revoked users.

Key Encapsulation Mechanisms. We described above a key encapsulation mechanism (KEM) where only a key is generated. The payload is then encrypted with a symmetric mechanism to get a full encryption scheme. All the broadcast encryption schemes known to the authors can be written as KEMs, e. g. the bilinear BE schemes from [BGW05, GW09] generate a random group element which is then multiplied to the message. This random group element can be considered as the symmetric key, and group multiplication as the symmetric encryption. To achieve CCA2-security for the full broadcast encryption, given a CCA2-secure key encapsulation, we additionally need to bind all the components of the ciphertext together.

Encryption and Decryption Keys. The encryption key can be either public (asymmetric) or private (symmetric), in the former case, we talk about *public-key* broadcast encryption, in the latter we say this is a *private-key* broadcast encryption. The decryption keys can either be defined and sent to the users at the join phase and never modified again, or be updated each time another user joins the system. In the former case, the decoders are said to be *stateless* since there is no state to evolve. In the latter case, the decoders are called *stateful* because they have to keep their state up-to-date. They thus have to be always on-line to receive the update information.

Default. As already mentioned, in this paper, we focus on public-key key encapsulation system with possibly stateful decoders.

3 Security Notions

Besides the various properties that a broadcast encryption scheme can satisfy, many security notions have been defined to take all the threats into consideration. We will thus review them, and try to give a cleaner view. As usual, security notions are defined by the goal the adversary want to achieve, and by the means that are available. We first define our standard security notions, and then compare them with some alternatives defined in the literature.

3.1 Standard Security Notions

Since we are studying a KEM [CS03], the goal of the adversary is to distinguish two keys in a key encapsulation, noted IND for *key indistinguishability*: after having received the public parameters, in the first phase (the FIND phase) the adversary outputs a target set S; then the challenger runs the key encapsulation algorithm, on this set S, that outputs the ephemeral K and the encapsulation H. It then chooses a random key K' and a random bit b and sets $K_b = K$ and $K_{1-b} = K'$. Upon receiving (H, K_0, K_1), the adversary runs the second phase (the GUESS) during which it has to decide whether H encapsulates K_0 or K_1, which means it has to guess the bit b.

Oracles can be available at different periods of time (Setup, FIND-phase, or GUESS-phase) which defines several kinds of attacks. Figure 2 shows the experiment $\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z}_{\mathcal{DBE},\mathcal{A}}(k)$, where the oracles OJoin_1, $\mathsf{OCorrupt}_1$ and $\mathsf{ODecaps}_1$ are available during the FIND-phase, and the oracles OJoin_2, $\mathsf{OCorrupt}_2$ and $\mathsf{ODecaps}_2$ are available during the GUESS-phase. According to the exact definition of these oracles, we have an IND-Dynx-Ady-CCAz security game, for x-Dynamic (Join), y-Adaptive (Corrupt) and CCA-z (Decaps). If not otherwise specified, use of the variables x, y, z means that they can be replaced by any level defined below.

The Join *Oracle.* It can be available at the Setup-time only. In this case, the adversary can make a number of non-adaptive Join-queries, where he receives the results only at the end of the Setup-phase, together with the parameters and MSK, EK. As said above, we then talk about a **static scheme**, and the attack is *S-Dynamic* (or Dyn**S**), and both the oracles OJoin_1 and OJoin_2 output $\perp$. The Join-oracle can be available during the first phase only, then $\mathsf{OJoin}_1 = \mathsf{Join}$ but the OJoin_2 oracle outputs $\perp$, and the attack is *1-Dynamic* (or Dyn**1**); it can be available always, then $\mathsf{OJoin}_1 = \mathsf{OJoin}_2 = \mathsf{Join}$, and the attack is *2-Dynamic* (or Dyn**2**).

The Corrupt *Oracle.* Corruptions can be more or less adaptive. Again, the adversary may have to decide before the Setup-time which users will be corrupted. This is a **selective attack** or *S-Adaptive* (also denoted Ad**S**), which is meaningful for static schemes (Dyn**S**) only (otherwise there are no users to corrupt during the Setup-phase), and then both the oracles $\mathsf{OCorrupt}_1$ and $\mathsf{OCorrupt}_2$ output $\perp$. It can be available during the first phase only, then $\mathsf{OCorrupt}_1 = \mathsf{Corrupt}$ but the

```
$\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z-b}_{\mathcal{DBE},\mathcal{A}}(k)$
    $(\mathsf{MSK}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^k)$;
    $\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
    $(st, S) \leftarrow \mathcal{A}^{\mathsf{OJoin}_1(\cdot),\mathsf{OCorrupt}_1(\cdot),\mathsf{ODecaps}_1(\cdot,\cdot,\cdot)}(\mathsf{param})$;
    $(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg, S)$; $K_b \leftarrow K$; $K_{1-b} \overset{\$}{\leftarrow} \mathcal{K}$;
    $b' \leftarrow \mathcal{A}^{\mathsf{OJoin}_2(\cdot),\mathsf{OCorrupt}_2(\cdot),\mathsf{ODecaps}_2(\cdot,\cdot,\cdot)}(st; S, H, K_0, K_1)$;
    if $\exists i \in S, (i, S, H) \in \mathcal{Q}_D$ or $i \in \mathcal{Q}_C$
    then return $0$;
    else return $b'$;
```

where $x \in \{s, 1, 2\}$, $y \in \{0, s, 1, 2\}$, $z \in \{0, 1, 2\}$.

```
$\mathsf{OJoin}(i)$
    $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathsf{Join}(\mathsf{msk}, i)$;
    return $\mathsf{upk}_i$;
```

```
$\mathsf{OCorrupt}(i)$
    $\mathcal{Q}_C \leftarrow \mathcal{Q}_C \cup \{i\}$;
    return $\mathsf{usk}_i$;
```

```
$\mathsf{ODecaps}(i, S, H)$
    $\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{(i, S, H)\}$
    $K \leftarrow \mathsf{Decaps}(\mathsf{usk}_i, S, H)$;
    return $K$;
```

Fig. 2. $\mathcal{DBE}$: Key Privacy (IND-Dynx-Ady-CCAz)

$\mathsf{OCorrupt}_2$ oracle outputs $\perp$, and the attack is *1-Adaptive* (or Ad**1**). It can be available during the full security game, then $\mathsf{OCorrupt}_1 = \mathsf{OCorrupt}_2 = \mathsf{Corrupt}$, and the attack is *2-Adaptive* (or Ad**2**). Eventually, the adversary can have no access at all to the Corrupt oracle: we say the attack is *0-Adaptive* (or Ad**0**).

The Decaps *Oracle.* As usual for chosen-ciphertext security, the Decaps-oracle can be available or not. It can never be available in the CPA (or CCA**0**) scenario, and both the oracles $\mathsf{ODecaps}_1$ and $\mathsf{ODecaps}_2$ output $\perp$; it can be available during the first phase only, then $\mathsf{ODecaps}_1 = \mathsf{Decaps}$ but the $\mathsf{ODecaps}_2$ oracle outputs $\perp$, and the attack is CCA**1**; it can be available during the full security game, then $\mathsf{ODecaps}_1 = \mathsf{ODecaps}_2 = \mathsf{Decaps}$, and the attack is CCA**2**.

For the IND-goal, the natural restriction for the adversary is not to ask for the decapsulation of the challenge header H nor corrupt any user in the target set S.

Remark 2. For private-key schemes, the adversary is granted access to the encapsulation oracle instead of the encryption key. In the rest of the paper, we will focus on the public-key setting for dynamic broadcast encryption schemes (noted PKDBE).

Definition 3. *A public-key DBE scheme $\mathcal{DBE}$ is said to be $(t, N, q_C, q_D, \varepsilon)$-IND-Dynx-Ady-CCAz-secure if in the security game presented in figure 2, the advantage, denoted $\mathrm{Adv}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z}_{\mathcal{DBE}}(k, t, N, q_C, q_D)$, of any t-time adversary $\mathcal{A}$ registering at most N users (*OJoin *oracle), corrupting at most q_C of them (*OCorrupt *oracle), and asking for at most q_D decapsulation queries (*ODecaps *oracle), is bounded by ε:*

$$\mathrm{Adv}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z}_{\mathcal{DBE}}(k, t, N, q_C, q_D) =$$
$$\max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z-1}_{\mathcal{DBE},\mathcal{A}}(k) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{a}y\mathsf{cca}z-0}_{\mathcal{DBE},\mathcal{A}}(k) = 1]\} .$$

3.2 Alternatives and Variants

Forward-Secrecy. For dynamic exclusive schemes (the target set is defined by the list of revoked users), new users are by definition included in the target sets of the message headers, even if they did not exist at the time the header was sent. Furthermore, since new users are included in the challenge set S, the adversary is not allowed to corrupt them. This means the encryption does not provide forward-secrecy. To model forward-secrecy, we can allow corruption of joined users, and in this case the encryption key EK must evolve when a new user joins the system.

For dynamic inclusive schemes (the target set is defined by the list of authorized users), the $\mathsf{Ad}\mathbf{2}$ notion provides *forward-secrecy* since any user not in the target set can be corrupted in the second phase.

Target Set. In the default security game, the adversary chooses the target set S at the end of the first phase, the FIND phase which consists in finding the best S for winning the game. But some papers in the literature restrict this choice:

– The adversary announces the target set before the setup phase [BGW05]. We call this *selective security*, denoted $\mathsf{Targ}\mathbf{S}$. This can only happen in static schemes, because the adversary needs to know the set of users to choose the target set from.
– The target set is automatically set to all uncorrupted users at the end of the first phase [DF03]. We call this *fixed-target-set security*, denoted $\mathsf{Targ}\mathbf{F}$.

When needed, the default case (the adversary chooses the target set S at the end of the FIND-phase) is denoted $\mathsf{Targ}\mathbf{C}$.

Security Models in the Literature. We can now characterize all the security models defined in the literature into our formalism: These notions are summarized in table 1, when S is the target set and C the corrupted users set.

– In [YFDL04], the authors defined the full access to the $\mathsf{Corrupt}$ oracle, but for a static scheme (no Join oracle). In order to accommodate the forward-secrecy, they included time slots. Disregarding the latter, the security model is similar to IND-$\mathsf{Dyn}\mathbf{S}$-$\mathsf{Ad}\mathbf{2}$-$\mathsf{CCA}\mathbf{2}$-$\mathsf{Targ}\mathbf{F}$. Essentially the adversary is restricted to corrupting only users from a time slot later than the one the challenge message was sent in. In our model, IND-$\mathsf{Dyn}\mathbf{x}$-$\mathsf{Ad}\mathbf{2}$-$\mathsf{CCA}\mathbf{z}$-$\mathsf{Targ}\mathbf{F}$-security does only make sense for $x = 2$, as otherwise no users can be corrupted in the GUESS-phase (because the target set is fixed to $U \setminus C$ and the adversary cannot join new users after the challenge phase).
– In [Del08] the authors define a security model for IBBE they call IND-sID-CCA (selective ID CCA-security), which is IND-$\mathsf{Dyn}\mathbf{S}$-$\mathsf{Ad}\mathbf{2}$-$\mathsf{CCA}\mathbf{2}$-$\mathsf{Targ}\mathbf{S}$-security in our notation.

Table 1. Adversarial Capabilities

Security	before setup	FIND-Phase	Challenge-Phase	GUESS-Phase
Ad**2**		Corrupt	S	Corrupt
Ad**2**Targ**F**		Corrupt		Corrupt
Ad**1**		Corrupt	S	
Ad**1**Targ**F**		Corrupt		
semi-static	C		S	
static	C			

- The partial access to the Corrupt oracle has been used in [BW06] and [GW09]. In our notation, the authors used IND-Dyn**S**-Ad**1**-CCA**0** security, since no decapsulation queries were available.
- As noted above, the the fixed-target-set security was introduced in [DF03], but no Corrupt queries were allowed in the second phase, and the system was static (no Join query). In our formalism, this is IND-Dyn**S**-Ad**1**-CCA**z**-Targ**F**, according to the Decaps-oracle access.
- *Semi-static* security has been introduced in [GW09] in order to build a generic conversion into Adaptive-1. In this setting, the adversary must announce the set of corrupted users before the setup phase, as we defined as *selective attack*. In our notation, this is IND-Dyn**S**-Ad**S**-CCA**0** security.[1]
- In the *static* model due to [BGW05], the adversary also has to announce its target set before the setup phase (selective attack). In our notation, this is IND-Dyn**S**-Ad**S**-CCA**2**-Targ**F** security with fixed target set. The authors also define a CPA version.[2]

Collusion Resistance. We can also distinguish between two types of collusion-resistance: full collusion-resistance, where there is no limit on the number of Corrupt-queries, and t-collusion-resistance, where the number of queries is bounded by t (which can depend on the number of users N). With our parameters, we implicitly consider all the cases.

[1] More precisely, in the *semi-static* version of the experiment, the adversary must commit to a set $\tilde{S}$ before the setup phase. He is allowed to corrupt any user not in $\tilde{S}$ after the setup phase, and must choose a challenge set $S \subseteq \tilde{S}$. An equivalent formulation is that the adversary chooses the set C of users to corrupt before the setup phase (because he can corrupt all users not in $\tilde{S}$), but chooses S at the challenge phase. This formulation is only equivalent for fully collusion resilient schemes, but it is for these schemes that the notions were designed.

[2] In the *static* version of the experiment [BGW05], the adversary has to announce the set S of users he wants to attack before the setup phase. He then receives the private keys of all users not in S after the setup phase. An equivalent definition is that he chooses the set C of corrupted users, and the S is fixed to be all the users except C. To allow the adversary to choose the target set, the adversary announces both C and S before the setup phase. This definition where the adversary chooses both C and S can also be used in not fully collusion-secure schemes and is the one considered in this section.

4 Relationship between the Security Notions

In this section, we shed light on the relationships between the security notions we defined in the last section. We start in section 4.1 with the hierarchy of Decaps-oracles, where we expect no surprises. In section 4.2, we explore the Join-oracle, of which we defined three different versions: For the passive version, which takes no input, all notions are equivalent; For the active version, which takes input and outputs a user tag, we can separate all notions. For the IBBE version, which takes an arbitrary string as input, but does not output a user tag (the upk is the identity of the user), we can show equivalences and separations based on the availability of a $\mathsf{Corrupt}$-oracle. In section 4.3, we address the $\mathsf{Corrupt}$-queries, and gaps appear according to the number of such queries, and thus the level of collusion-resistance. In section 4.4, we examine the various ways in which the target set can be chosen. Due to space limitations, many proofs have been removed from this version, but are given in the full version [PPS11].

4.1 Separating CPA and CCA

We remember the well-known separation between CPA ($\mathsf{CCA}\mathbf{0}$), $\mathsf{CCA}\mathbf{1}$, and $\mathsf{CCA}\mathbf{2}$-security for PKE from [BDPR98]. The same separation applies in the case of broadcast encryption, because if we set $\mathsf{KeyGen}(1^k)$ to
$(\mathsf{MSK}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^k)$; $(\mathsf{usk}_1, \mathsf{upk}_1) \leftarrow \mathsf{Join}(\mathsf{MSK}, 1)$; $\mathsf{dk} \overset{\mathsf{def}}{=} \mathsf{usk}_1, \mathsf{ek} \overset{\mathsf{def}}{=} \mathsf{EK}||\mathsf{upk}_1$, we obtain a single-user KEM scheme.

Theorem 4. *The following implications are strict:*

$$\mathit{IND}\text{-}\mathsf{Dyn}\mathbf{x}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{2} \Rightarrow \mathit{IND}\text{-}\mathsf{Dyn}\mathbf{x}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{1} \Rightarrow \mathit{IND}\text{-}\mathsf{Dyn}\mathbf{x}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{0}.$$

4.2 Separating Notions of Dynamicity

In this section, in order to compare the Join-oracle access, we also have to consider the three versions of the Join-algorithm, as defined in section 2.1: *passive*-Join, if it takes no input; *active*-Join, if it takes an input; *ID-based*-Join, if the output tag upk is the input identity.

Easy Implications. As above, there is a clear hierarchy on the Join oracle access: at the setup time only, in the first phase, or at anytime.

Theorem 5. *The following implications hold for all versions of the* Join oracle: $\mathsf{IND}\text{-}\mathsf{Dyn}\mathbf{2}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{z} \Rightarrow \mathsf{IND}\text{-}\mathsf{Dyn}\mathbf{1}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{z} \Rightarrow \mathsf{IND}\text{-}\mathsf{Dyn}\mathbf{S}\text{-}\mathsf{Ad}\mathbf{y}\text{-}\mathsf{CCA}\mathbf{z}$.

Passive Join. This is a standard definition in the literature. Interestingly enough, in this context all the notions are equivalent, since the adversary cannot influence the output.[3]

[3] It is interesting to note that the equivalence is for our above notions only: for passive-Join, a query in the first phase is strictly more useful than a query in the second phase. As a consequence, if we consider in details the number of queries in each phase, as done in [PP04] for the encryption and decryption oracles, we can show that $\mathsf{Dyn}(N_1 + N_2, 0) \rightarrow \mathsf{Dyn}(N_1, N_2) \rightarrow \mathsf{Dyn}(0, N_1 + N_2)$, and these implications are strict. However, in the above theorem, we do not fix the number of queries.

Theorem 6. *If* Join *takes no input,we have the following equivalences*

$$\mathit{IND}\text{-Dyn}\mathbf{2}\text{-Ady-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{1}\text{-Ady-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ady-CCAz}.$$

Proof. Because of the trivial implications, it remains to show that DynS $\Rightarrow$ Dyn2. Given a successful Dyn2-adversary $\mathcal{A}^d$ that makes N_1 queries to the Join-oracle in phase 1, and N_2 queries to the Join-oracle in phase 2, we construct a successful DynS-adversary $\mathcal{A}^s$ that joins $N = N_1 + N_2$ users before the setup phase. Because the Join-oracle takes no input, its behavior is exactly the same in phase 1 and phase 2. Therefore $\mathcal{A}^s$ can store the results and then answer all Join-queries made by $\mathcal{A}^d$ later.

Active Join with Large Input. If the Join-algorithm is interactive or takes input from the adversary (that can be sufficiently large, i. e. $|\mathcal{UI}|$ is superpolynomial), the adversary can influence the Join-process:

Theorem 7. *If* Join takes input and outputs a public tag, the following implications are strict

$$\text{IND-Dyn}\mathbf{2}\text{-Ady-CCAz} \Rightarrow \text{IND-Dyn}\mathbf{1}\text{-Ady-CCAz} \Rightarrow \text{IND-Dyn}\mathbf{S}\text{-Ady-CCAz}.$$

Identity-Based. In this case, the OJoin-oracle only outputs a user secret key $\mathsf{usk}_{\mathsf{id}}$ (because $\mathsf{upk}_{\mathsf{id}} = \mathsf{id}$). This means that in order to gain anything from the output, the adversary must also be able to corrupt users.

Theorem 8. *For ID-Based Broadcast Encryption, the following implications are strict*

$$\begin{array}{l}
\mathit{IND}\text{-Dyn}\mathbf{2}\text{-Ad}\mathbf{2}\text{-CCAz} \Rightarrow \mathit{IND}\text{-Dyn}\mathbf{1}\text{-Ad}\mathbf{2}\text{-CCAz} \Rightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{2}\text{-CCAz}\\
\mathit{IND}\text{-Dyn}\mathbf{2}\text{-Ad}\mathbf{1}\text{-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{1}\text{-Ad}\mathbf{1}\text{-CCAz} \Rightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{1}\text{-CCAz}\\
\mathit{IND}\text{-Dyn}\mathbf{2}\text{-Ad}\mathbf{S}\text{-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{1}\text{-Ad}\mathbf{S}\text{-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{S}\text{-CCAz}\\
\mathit{IND}\text{-Dyn}\mathbf{2}\text{-Ad}\mathbf{0}\text{-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{1}\text{-Ad}\mathbf{0}\text{-CCAz} \Leftrightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{0}\text{-CCAz}
\end{array}$$

4.3 Separating Forms of Corruption

Theorem 9.

$$\begin{array}{l}
\mathit{IND}\text{-Dynx-Ad}\mathbf{2}\text{-CCAz} \Rightarrow \mathit{IND}\text{-Dynx-Ad}\mathbf{1}\text{-CCAz}\\
\qquad \Rightarrow \mathit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{S}\text{-CCAz} \Rightarrow \mathit{IND}\text{-Dynx-Ad}\mathbf{0}\text{-CCAz},
\end{array}$$

and for BE schemes that are not fully collusion-secure all implications are strict.

Proof. The implications are clear, since having access to an oracle never makes an adversary weaker. The separations follow from lemmas 10, 11, 12, and 14.

Separation of no Corruption from Selective Corruption. Recall that for AdS, the only version of Dyn that makes sense is DynS (section 3.1).

Lemma 10. *IND*-Dyn**x**-Ad**0**-CCA**z** $\nRightarrow$ *IND*-Dyn**S**-Ad**S**-CCA**z**.

Separation of Selective Corruption from 1-Adaptive Corruption. In a model with selective corruption, the adversary must announce the set C of corrupted users before seeing the encryption key. To make a difference, we would have to give some information on the subset of the users to corrupt in the encryption key: we thus embed such information using a secret sharing scheme to make sure all of the identified users (special users) have to be corrupted.

We need to make sure that the subset is hard to guess by chance: this is the case for IBBE, where the size of the set $\mathcal{UI}$ is exponential and any user is hard to guess. In case $\mathcal{UI}$ is of polynomial size, the size of the subset must not be too small, otherwise all of them will be corrupted even by a selective-corruption adversary with significant probability. If t is the number of special users, there are $\binom{N}{t}$ ways of choosing them, where N is polynomial in the security parameter. To make the binomial be super-polynomial for a polynomial N, we need t to be non-constant.

How can we be sure the adversary corrupts at most t users? First, it can be set by definition, using the t-collusion secure level. For IBBE, $\binom{|\mathcal{UI}|}{1}$ is already exponential in the security parameter. However, without any additional constraint, for a basic broadcast encryption scheme, if $N-t$ is constant, then $|S|$ must also be constant, and we are actually dealing with a simple multi-encryption scenario. Multi-cast security of encryption schemes has been considered in [BPS00]. The authors proved that standard IND-CPA encryption schemes remain secure even if the same message is sent to different users in parallel. This makes the case where the adversary is always sending only to a constant number of users less interesting to us. It thus seems reasonable to exclude these cases from BE. In the following, we thus focus on t-collusion secure schemes, where t must be less than the total number of users minus a non-constant number.

Lemma 11. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),*

IND-Dyn**S**-Ad**S**-CCA**z** $\nRightarrow$ *IND*-Dyn**S**-Ad**1**-CCA**z**.

Separation of 1-Adaptive Corruption from 2-Adaptive Corruption.

Lemma 12. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),*

IND-Dyn**x**-Ad**1**-CCA**z** $\nRightarrow$ *IND*-Dyn**x**-Ad**2**-CCA**z** *for* $z \in \{0,1\}$.

As noted, the proof requires t and $N-t$ to be non-constant. But we can also note that it does not work in the CCA**2**-setting, because on the one hand the scheme is malleable, and on the other hand the adversary could simply query the H_i's to the Decaps-oracle.

For the following lemma, we need the notion of a homomorphic OWF.

Definition 13 (Homomorphic One-Way Function). *Let* $(G,+)$ *and* $(H,*)$ *be two groups with* $2^{k-1} \leq |G| \approx |H| \leq 2^k$*. A PPT-computable function* $f: G \rightarrow H$ *is*

– one-way *if* $\forall \mathcal{A} : \Pr[x \overset{\$}{\leftarrow} G; y \leftarrow \mathcal{A}(1^k, f(x)); f(y) = f(x)]$ *is negligible.*
– homomorphic *if* $f(x + y) = f(x) * f(y)$.

An example of a homomorphic OWF is discrete exponentiation (assuming DLOG is hard). The security of MAC and symmetric encryption is defined as usual.

Lemma 14. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),if strongly-UF-CMA-secure MAC, IND-CCA2-secure symmetric encryption and homomorphic OWF exist,*

$$\textsf{\textit{IND}-Dynx-Ad}\mathbf{1}\textsf{-CCA}\mathbf{2} \not\Rightarrow \textsf{\textit{IND}-Dynx-Ad}\mathbf{2}\textsf{-CCA}\mathbf{2}.$$

4.4 Choice of the Target Set

Selective Security

Lemma 15. *The following implication is strict:*

$$\textsf{\textit{IND}-Dynx-Ady-CCAz-Targ}\mathbf{C} \Rightarrow \textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ady-CCAz-Targ}\mathbf{S}.$$

Fixed Target Sets. In our definition the adversary chooses the target set S of the challenge. In the DPP security model [DPP07], S is automatically the set of all non-compromised users. The same situation appears in [BGW05], where the adversary outputs S before the setup and receives the secret keys for all users in $U \setminus S$. A similar definition is given for the BGW model. We could reformulate the BGW model so that the adversary outputs the set C of the keys he wants to know, and S is set to $U_1 \setminus C_1$. This formulation is obviously equivalent. We want to investigate the relationship between these two notions.

Note that under the "fixed" definition, the notions IND-Dynx-Ad**1**-CCAz and IND-Dynx-Ad**2**-CCAz for $x \in \{s, 1\}$ are equivalent since in any case the adversary cannot corrupt users after the challenge phase (all the non-corrupted users at the end of the first phases are in the target set and cannot be corrupted).

Theorem 16. *All the following implications are strict*

$$\begin{aligned}
&\textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ad}\mathbf{S}\textsf{-CCAz-Targ}\mathbf{C} \Rightarrow \textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ad}\mathbf{S}\textsf{-CCAz-Targ}\mathbf{S}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\Leftrightarrow \textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ad}\mathbf{S}\textsf{-CCAz-Targ}\mathbf{F}\\
&\textsf{\textit{IND}-Dynx-Ad}\mathbf{0}\textsf{-CCAz-Targ}\mathbf{C} \Rightarrow \textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ad}\mathbf{0}\textsf{-CCAz-Targ}\mathbf{S}\\
&\qquad\qquad\qquad\qquad\qquad\qquad\Rightarrow \textsf{\textit{IND}-Dynx-Ad}\mathbf{0}\textsf{-CCAz-Targ}\mathbf{F}
\end{aligned}$$

The theorem follows from lemmas 15, 17, 18, and 19.

Lemma 17. $\textsf{\textit{IND}-Dyn}\mathbf{S}\textsf{-Ady-CCAz-Targ}\mathbf{S} \Rightarrow \textsf{\textit{IND}-Dynx-Ady-CCAz-Targ}\mathbf{F}$ *for $y \in \{0, s\}$.*

Proof. From an adversary $\mathcal{A}^f$ against the IND-Dynx-Ady-CCAz-Targ**F**-security of a BE scheme, we build an adversary $\mathcal{A}^S$ against the IND-Dyn**S**-Ady-CCAz-Targ**S**-security. If the model has no corruption or static corruption, $\mathcal{A}^S$ runs $\mathcal{A}^f$, who outputs C, chooses the same C and sets his target set $S = U \setminus C$.

Lemma 18. $\mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{S}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{S}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}}$.

Proof. Given a successful adversary $\mathcal{A}^S$, we construct an adversary $\mathcal{A}^f$ as follows. $\mathcal{A}^S$ outputs his target set S and the set of users to corrupt C before the Setup phase. $\mathcal{A}^f$ chooses $C' = U \setminus S$.

Lemma 19. $\mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \not\Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}}$.

Proof. In the $\mathsf{IND\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}}$-experiment, the target set is always fixed to $S = U$. Given a $\mathsf{IND\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}}$-secure scheme Π, we modify it into a scheme Π' that is still $\mathsf{IND\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}}$-secure, but not $\mathsf{IND\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{0}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}}$. The only change is that if $|S| = 1$, $\Pi'.\mathsf{Encaps}$ sets $K = 0$ (or determines the key in a deterministic way by fixing all random coins e. g. to 0).

Theorem 20. *For fully collusion-resilient BE schemes, the following implications are strict*

$$\mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \Leftrightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}} \quad (y \in \{1,2\})$$

The theorem follows from lemmas 15 and 21. It seem curious at first that the relationship between fixed target set and selective security is inverted for models with no corruption, but in this case the fixed target set means that it is always set to U, while the selective security allows some freedom of the adversary to choose.

Lemma 21. *For fully collusion-resistant BE schemes*

$$\mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \Leftrightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \quad (y \in \{1,2\}).$$

Proof. It is clear that if the adversary can choose S freely, he can set it to $U \setminus C$. Let $\mathcal{A}^{choice}$ be a successful adversary against a BE scheme that can choose his target set S. Then we construct $\mathcal{A}^{fixed}$ as follows: $\mathcal{A}^{fixed}$ faithfully forwards all queries. When $\mathcal{A}^{choice}$ outputs his challenge target set S, $\mathcal{A}^{fixed}$ first issues corrupt queries so that $U \setminus C = S$, then asks for the challenge and forwards it to $\mathcal{A}^{choice}$. He forwards the guess bit b and wins with the same probability as $\mathcal{A}^{choice}$.

Note that $\mathcal{A}^{fixed}$ corrupts more users, which could reduce the tightness of a security proof, and causes the proof to fail in a t-resilient setting where $t < N-1$ (if $t = N - 1$, the scheme is fully collusion-resistant).

In the following, we denote by $\not\Leftrightarrow$ the fact that $\not\Rightarrow$ in both directions.

Theorem 22. *For BE schemes where the adversary must leave at least two users uncorrupted, the following implications are strict:*

$$\mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \not\Leftrightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}}$$

and $\mathsf{\mathit{IND}\text{-}Dyn\mathbf{x}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{y}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}} \quad (y \in \{1,2\})$

The theorem follows from lemmas 15, 23, 24, and 25.

Lemma 23. *If the adversary is restricted to leaving at least 2 users uncorrupted, the following implication is strict*

$$\textit{IND}\text{-Dynx-Ady-CCAz-Targ}\mathbf{C} \Rightarrow \textit{IND}\text{-Dynx-Ady-CCAz-Targ}\mathbf{F} \quad (y \in \{1,2\}).$$

Lemma 24. *If the adversary is restricted to leaving at least 2 users uncorrupted,*

$$\textit{IND}\text{-Dynx-Ady-CCAz-Targ}\mathbf{F} \not\Rightarrow \textit{IND}\text{-Dyn}\mathbf{S}\text{-Ady-CCAz-Targ}\mathbf{S}.$$

We can easily see that the adversary does not get weaker if he can choose the target set freely from the set of uncorrupted users $U \setminus C$, because he can choose $S = U \setminus C$ as in the fixed case.

Lemma 25.

$$\textit{IND}\text{-Dyn}\mathbf{S}\text{-Ady-CCAz-Targ}\mathbf{S} \not\Rightarrow \textit{IND}\text{-Dyn}\mathbf{S}\text{-Ady-CCAz-Targ}\mathbf{F} \textit{ for } y \in \{1,2\}.$$

5 Relationships between Notions from the Literature

A security notion that our model does not cover is defined in [DPP07]. In this model, the adversary accesses a JoinCorrupted oracle instead of the Corrupt oracle. That means he must decide whether to corrupt a user before the user is joined, but the choice can depend on information gained previously. The model defined in [DPP07] is Dyn**1**, as the adversary has access to a Join oracle before the challenge phase, CCA**0** and Targ**F**, as the challenge set is *fixed* to $S = U \setminus C$, so it is rather similar to IND-Dyn**1**-Ad**1**-CCA**0**-Targ**F**-model in our framework, except that the Corrupt oracle is replaced with JoinCorrupted. We call it the *partially adaptive* model. As in the previous section, we also denote Targ**C** the default case where the adversary can choose S as any subset of $U \setminus C$.

Theorem 26. *We have the following implications*

$$\begin{aligned} &\textit{IND}\text{-Dyn}\mathbf{1}\text{-Ad}\mathbf{1}\text{-CCAz-Targ}\mathbf{F} \Rightarrow \textit{partially adaptive} - \text{CCAz} - \text{Targ}\mathbf{C} \\ &\Rightarrow \textit{partially adaptive} - \text{CCAz} - \text{Targ}\mathbf{F} \Rightarrow \textit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{1}\text{-CCAz-Targ}\mathbf{S} \end{aligned}$$

that are all strict (the first one only if t-collusion secure with t and $N - t$ non-constant).

The proof can be found in the full version of this paper [PPS11].

We now have almost all the results we need to establish the relationship between the security notions that can be found in the existing literature to fill the picture on figure 3. We now complete it.

Theorem 27. *The following implication is strict*

$$\textit{Partially adaptive} - \text{CCAz} - \text{Targ}\mathbf{C} \Rightarrow \textit{IND}\text{-Dyn}\mathbf{S}\text{-Ad}\mathbf{S}\text{-CCA}\mathbf{0}\text{-Targ}\mathbf{C}.$$

Proof. From any semi-static adversary $\mathcal{A}^S$ we construct a partially adaptive adversary as follows. $\mathcal{A}^S$ announces N and C before the setup phase. $\mathcal{A}^{pa}$ asks for $\mathsf{JoinCorrupted}$ on all users in C and simply joins all users in $U \setminus C$. The separation is analogous to the one in lemma 11.

We relate semi-static security to the version of static security with 1-adaptive corruption defined in [GW09].

Theorem 28. *The following implication is strict*

$$\mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{S}\text{-}CCA\mathbf{0}\text{-}Targ\mathbf{C}} \Rightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{1}\text{-}CCA\mathbf{0}\text{-}Targ\mathbf{S}}.$$

Proof. From any selectively 1-adaptive adversary $\mathcal{A}^a$ we construct a semi-static adversary $\mathcal{A}^s$. $\mathcal{A}^a$ announces N and S before the setup phase. $\mathcal{A}^s$ forwards N and sets $C = U \setminus S$. He now has enough information to answer all $\mathsf{Corrupt}$-queries. The separation is analogous to the one in lemma 15.

Theorem 29. *Partially adaptive*-$\mathsf{CCA\mathbf{z}\text{-}Targ\mathbf{F}} \not\leftrightarrow \mathsf{\mathit{IND}\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{S}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}}$.

*: for t-collusion secure schemes with t and $N - t$ non-constant
(all implications are strict)

$$\begin{tikzcd}
& \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{2}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \arrow[d, "\text{lemma 23*}"] & \\
& \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{2}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \arrow[d, "\text{theorem 9*}"] & \\
& \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{1}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \arrow[d, "\text{lemma 23*}"] & \\
& \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{1}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{F}} \arrow[d, "\text{theorem 26}"] & \\
& \text{partially adaptive-}\mathsf{CCA\mathbf{z}\text{-}Targ\mathbf{C}} \arrow[dl, "\text{theorem 26}"'] \arrow[dr, "\text{theorem 27}"] & \\
\text{partially adaptive-}\mathsf{CCA\mathbf{z}\text{-}Targ\mathbf{F}} \arrow[dr, "\text{theorem 26}"'] & & \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{S}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{C}} \arrow[dl, "\text{theorem 28}"] \\
& \mathsf{IND\text{-}Dyn\mathbf{S}\text{-}Ad\mathbf{1}\text{-}CCA\mathbf{z}\text{-}Targ\mathbf{S}} &
\end{tikzcd}$$

Fig. 3. Relations between Security Notions from the Literature

Remark 30. To conclude this section on the security notions found in the literature, we place some of the existing schemes more precisely. First, one can show that the DPP dynamic BE scheme [DPP07] fulfills the "choice"-definition without changing the security proof. Using theorem 28, we see that this notion implies semi-static security, so the two-key transformation from [GW09] can be used to achieve 1-adaptive security. The transformation is not efficient in this case, because the length of the ciphertext depends on the number of r revoked users, and to use the transformation, $N + r$ of $2N$ users have to be revoked. Looking at the proof in [GW09], we see that the after applying the two-key transformation, a scheme can be proved 2-adaptively secure using the same proof, because the simulator has the secret keys of each user and can answer $\mathsf{Corrupt}$-queries in the GUESS-phase as easily as in the FIND-phase.

Table 2. Comparison between schemes

	DF03	BGW05	DPP07	Del08	GW09a	GW09b	Naive
Dyn	Dyn**S**	Dyn**S**	Dyn**S**	Dyn**1**	Dyn**S**	Dyn**S**	Dyn**2**
Ad	Ad**1**	Partially adaptive	Ad**2**	Ad**2**	Ad**2**	Ad**2**	Ad**2**
CCA	CCA**2**	CCA**2**	CCA**2**	CCA**0**	CCA**2**	CCA**2**	CCA**2**
Targ	Targ**S**	Targ**S**	Targ**S**	Targ**F**	Targ**C**	Targ**C**	Targ**C**

6 Previous Schemes

Let us now discuss on the previous schemes in order to compare them. Table 2 sums up the security levels for each of them.

DF03. Dodis and Fazio[DF03] proposed the first scheme that is secure against adaptive adversaries. However, their scheme is in the Targ**F** model. Consequently, the scheme can only be Ad**1**-secure, because any corrupted user in the second phase is implicitly included in the target set and can thus decrypt. One of the main disadvantages of the DF03 scheme is that the bound of maximum revoked user r_{max} should be fixed before the setup and as soon as there are more than r_{max} corrupted users, the scheme can be totally broken. The DF03 scheme can be shown to be Ad**2**-secure when the target set is adversarially chosen with the size of the revoked set bounded by r_{max} and the total number of corrupted users in both first and second phases is also bounded by r_{max}.

BGW05. In [BGW05], Boneh, Gentry, and Waters presented new methods for achieving fully collusion-resistant systems with short ciphertexts. However, the scheme is only proved secure in the static model (Dyn**S**). As discussed in [GW09], the BGW proof of security requires an "exact cancellation" and there is not an obvious way to prove BGW05 to be semi-statically secure.

DPP07. In [DPP07], Delerablée, Paillier, and Pointcheval proposed a dynamic scheme that is partially adaptive secure. As pointed out in remark 30, we can show that the adversary can be allowed to choose the target set, which implies that this scheme is semi-statically secure. Therefore, by using Gentry-Waters transformation, one can obtain a (very inefficient) adaptive secure scheme.

Del08. The identity-based broadcast encryption in [Del08] deals with 2-adaptive corruption and enjoys CCA security with constant ciphertext and private key sizes. However, the adversary has to announce its target set before the setup phase which corresponds to our selective security model.

GW09. In [GW09], the authors aim to construct efficient schemes that are adaptively secure and that resist to full collusion. The adaptive security mentioned in the paper correspond to our Ad**1** model. However, their schemes can be easily proved secure in Ad**2** model. They introduced a two-key transformation that convert a semi-static system of $2N$ users into a adaptive secure system of N users. Their schemes are not dynamic.

A Secure Broadcast Encryption Scheme

Let us now propose a simple scheme that is IND-Dyn**2**-Ad**2**-CCA**2**-secure to show that it is possible. The naive BE scheme where the center shares a key with every user is not IND-Dyn**2**-Ad**2**-CCA**2**-secure, but adding a MAC makes it secure.

Definition 31. *Let $\mathcal{PKE}$ be an IND-CCA2 secure public-key encryption scheme with key length κ, $\mathcal{MAC}$ a strongly-UF-CMA MAC. We build a BE scheme Π in the following way.*

```
– Setup(1^k) $MSK \overset{\text{def}}{=} \emptyset; EK \overset{\text{def}}{=} \emptyset; Reg \overset{\text{def}}{=} \emptyset$
– Join(MSK, i) $(pk_i, sk_i) \leftarrow \mathcal{PKE}.\mathsf{KeyGen}(1^k)$. return $(sk_i, pk_i)$.
– Encaps(EK, S): $K, \mathcal{K}_m \overset{\$}{\leftarrow} \{0,1\}^k$;
      for all $i \in S : c_i \leftarrow \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{pk}_i, K||\mathcal{K}_m)$;
      $\sigma \leftarrow \mathcal{MAC}_{\mathcal{K}_m}(c_1|| \dots ||c_{|S|})$;
      $H \overset{\text{def}}{=} c_1|| \dots ||c_{|S|}||\sigma$
– Decrypt(sk_i, S, H): $K||\mathcal{K}_m = \mathcal{PKE}.\mathsf{Decrypt}(\mathsf{sk}_i, c_i)$
      if $\mathcal{MAC}.\mathsf{Verify}(\mathcal{K}_m, \sigma, c_1|| \dots ||c_{|S|})$ return $K$,
      else return $\perp$
```

Theorem 32. *The above BE scheme is IND-Dyn**2**-Ad**2**-CCA**2**-secure.*

The proof can be found in the full version of this paper [PPS11].

Acknowledgments

This work was supported in part by the French ANR-09-VERS-016 BEST Project.

References

[BDPR98] Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998), Full version available at `http://www.di.ens.fr/~pointche/pub.php`

[BGW05] Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)

[BPS00] Baudron, O., Pointcheval, D., Stern, J.: Extended notions of security for multicast public key cryptosystems. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 499–511. Springer, Heidelberg (2000)

[BW06] Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: ACM CCS, pp. 211–220. ACM, New York (2006), Full version available at Cryptology ePrint Archive `http://eprint.iacr.org/2006/298`

[CS03] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)

[Del08] Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007)

[DF03] Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 100–115. Springer, Heidelberg (2002), Full version available at Cryptology ePrint Archive `http://eprint.iacr.org/2003/095`

[DPP07] Delerablée, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007)

[FN94] Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)

[GW09] Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009), Full version available at Cryptology ePrint Archive `http://eprint.iacr.org/2008/268`

[NNL01] Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001), Full version available at Cryptology ePrint Archive `http://eprint.iacr.org/2001/059`

[PP04] Phan, D.H., Pointcheval, D.: On the security notions for public-key encryption schemes. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 33–46. Springer, Heidelberg (2005)

[PPS11] Phan, D.H., Pointcheval, D., Strefler, M.: Security notions for broadcast encryption. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS. Springer, Heidelberg (2011); Full version available on the web page of the authors

[YFDL04] Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In: ACM CCS 2004. ACM, New York (2004), Full version from `http://www.cs.brown.edu/~anna/research.html`