Highly Efficient Consistent Broadcast Encryption

Konstantin Gegier[1], Eike Kiltz[2], Roman Langrehr[3]*, and Guilherme Rito[2]*

[1] Department of Computer Science, ETH Zurich, Switzerland
konstantin.gegier@ethz.ch
[2] Ruhr-Universität Bochum, Germany
{eike.kiltz,guilherme.teixeirarito}@rub.de
[3] University of Waterloo, Canada
roman.langrehr@uwaterloo.ca

Abstract. Public Key Encryption for Broadcast (PKEBC) is a multi-recipient encryption primitive that guarantees decryption consistency across all designated recipients. Concretely, if a ciphertext c is encrypted for Bob and Charlie, and Bob's decryption yields a message m, then Charlie's decryption of c must also succeed and produce the same m. This property, though seemingly natural, is essential in secure group messaging, where consistent message delivery is often implicitly assumed. However, no efficient constructions of PKEBC currently exist: known approaches achieve consistency through Non-Interactive Zero-Knowledge (NIZK) proofs of generic statements. Not only is the complexity of the NIZK statements already prohibitively expensive, but, in addition, it is not even clear if these can be turned into purely algebraic statements while retaining linear-sized ciphertexts. This is crucial to enable the use of efficient NIZK constructions.

This paper presents new generic PKEBC constructions along with optimized instantiations of each. Concretely, we introduce $\mathsf{PKEBC}^{\mathsf{SM}}$ and $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$, and prove the security of these constructions in the standard and random oracle models, respectively.

- $\mathsf{PKEBC}^{\mathsf{SM}}$ achieves consistency via NIZK proofs. Crucially, we hand-tuned the NIZK statements of our instantiation to ensure the NIZK is only used to prove very simple and carefully optimized purely algebraic statements.
- $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ achieves consistency via the Fujisaki-Okamoto (FO) transform (CRYPTO 1999 and Journal of Cryptology 2013) applied to a Multi-Recipient Public Key Encryption scheme (mPKE). Specifically, FO's ciphertext recomputation during decryption ensures ciphertexts are well-formed, eliminating the need for costly NIZK proofs. We then give two suitable mPKE instantiations: one based on Kurosawa's mPKE (PKC 2002) and the other from Hash Proof Systems.

For all our schemes and their instantiations, ciphertext sizes and encryption and decryption times grow linearly with the number of receivers. Our constructions therefore enable the first practical group messaging applications with consistency guarantees.

* Part of the work was done while the author was affiliated with ETH Zurich.

1 Introduction

Group messaging is an important feature of every major messaging application, which is supposed to give the same experience as an in-person conversation during a group meet-up. Most messaging applications focus on achieving *confidentiality* and *authenticity*. However, this is insufficient to properly replicate in-person conversations, as these also ensure *consistency*.

Imagine, for example, an auctioneer (Alice) auctioning an item in a group chat. Each member of the group chat is eligible to bid and simply announces their bid by sending it in the group chat. In the end, the highest bidder wins and has to pay his bid (public-bid, first-price auction). A malicious bidder (Bob) could send a low bid that only slightly exceeds existing bids to Alice, and a very high bid to all other group members. Other group members can then be discouraged from placing their own bids in the belief that they would have to overbid Bob's high bid to win. However, if Bob wins, he only has to pay the low bid to Alice, because Alice will believe that this was his bid and no one else was willing to bid higher than this. This attack would not be possible in an in-person auction, when everyone can hear what anyone else says.

Several prior works have already identified consistency as an important security goal [60,64,1,20,6], and we believe that group messaging systems should aim for consistency, because users might naturally expect consistency to hold and act accordingly, which can lead to problems not only for auctions.

Yet, currently deployed messengers do not guarantee consistency (in the sense explained above), be it at the level of single messages or at the broader level of a chat. In fact, existing security models for group messaging explicitly allow an adversary to mount *group-splitting attacks*, wherein a group chat can evolve into a state where group members are partitioned, and members of different partitions cannot communicate (through that chat) with members of other partitions (see [2, Section 3.1]). This is problematic because group members are not informed that the group has been partitioned [1,2,3,47].[4] The main reason for the lack of consistency is presumably that it is challenging to achieve, and existing solutions are very inefficient. In this work, we show how consistency can be achieved in a very efficient way, without any significant overhead compared to IND-CCA secure multi-recipient public-key encryption (mPKE).

The difficulty of achieving consistency. Consistency needs to be ensured across all layers of the messaging application: from the network layer—which needs to ensure that messages are delivered to users consistently—to the cryptography layer and even the user interface layer—which needs to ensure that messages are displayed consistently. This work focuses on the cryptography layer. For completeness, in Section A.1, we briefly explain why consistent message delivery is a mild and realistic assumption.

[4] The MLS standard explicitly states that it is the upper application's responsibility to detect and recover (if necessary) from such attacks: [5, Section 16.12, "Group Fragmentation by Malicious Insiders"].

Preserving consistency when we convert an insecure broadcast channel into a confidential one is challenging. For example, if we were to use an ordinary PKE scheme to create ciphertexts for every recipient, a malicious sender Bob could simply encrypt a different message for each receiver (say, Alice and Charlie). Even if Alice is also given Charlie's ciphertext, she cannot check that it contains the same message as she obtained from decrypting her ciphertext because she does not know Charlie's secret key. In fact, she may not even be able to tell if Charlie can decrypt his ciphertext.

Thus, achieving desirable security properties like confidentiality and anonymity becomes much more challenging when consistency needs to be preserved, and many existing techniques to achieve these properties cannot be used.

Public Key Encryption for Broadcast. PKEBC is an encryption primitive introduced by Maurer, Portmann, and Rito [51] which enables consistent broadcast encryption in standard Public Key Infrastructure settings. Roughly, one selects a set of recipients and then encrypts a message to them. Each of these recipients can then decrypt the resulting ciphertext, and to do that they only need to know their secret key. In particular, they do not need to know who the other recipients of the ciphertext are. Instead, decryption outputs the public keys of all other recipients (plus the encrypted message). Consistency guarantees that each of the recipients whose public key was output by decryption can successfully decrypt the same ciphertext as well—and obtain the same set of public keys and the same message. In addition to confidentiality and consistency, PKEBC schemes also guarantee anonymity.

In addition to being useful per se, PKEBC schemes can also be combined with Multi-Designated Verifier Signature (MDVS) into Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE) schemes. These schemes were introduced specifically to enable Off-The-Record (OTR) group messaging; they provide all the guarantees that PKEBC schemes provide plus *authenticity* and *Off-The-Record* (i.e. plausible deniability).

Applications to messaging. Recent work shows how to construct a *consistent*, fully decentralized, and policy-enforcing messaging application that is (essentially) as efficient as its underlying communication channels [47]. Their messenger is proven secure—and in particular, *consistent*—without any party honesty assumptions: all it requires from the underlying channels are message *consistency* and *replay protection* guarantees. A very useful property of their messenger is that, as they show, the resulting messenger inherits any additional guarantees—such as authenticity, confidentiality, anonymity, and Off-The-Record—from the underlying communication channels (in the sense that the resulting messenger preserves these properties from the underlying channels).

Their messenger can be instantiated from MDRS-PKE; in turn we know how to construct these schemes by a simple combination of PKEBC and MDVS schemes (plus a signature scheme). This positions PKEBC as the key tool to enable consistent group messaging.

State of the art for PKEBC schemes. The first PKEBC scheme was given by Maurer et al. [51]. Essentially, their PKEBC scheme generates a separate ciphertext for each user and a NIZK that proofs validity of the ciphertext. Since each ciphertext needs to contain the set of recipients and the actual payload message, this results in ciphertexts growing quadratic in the number of receivers and poor scalability.

In follow up work, Chakraborty, Hofheinz, Maurer, and Rito [10] give a new PKEBC construction which resolves these issues and is asymptotically efficient. Their PKEBC ciphertexts grow only linear in the number of users and even the time for encryption and decryption is only linear. They also prove that their PKEBC is tightly secure from primitives that have tightly secure instantiations from standard assumptions. When plugged into the MDRS-PKE compiler, this constructions result in a tightly secure MDRS-PKE scheme which enjoys linear ciphertext sizes—matching a known lower bound [18]—as well as linear time encryption and decryption. Despite this, their PKEBC construction [10] does not admit an efficient concrete instantiation and is not practically efficient. The problem with instantiating their construction stems from their reliance on a NIZK to prove a rather involved Naor–Yung type of statement [56], as explained in [10, Section 5.4] (Relation $R_{\mathsf{PKEBC}^{\mathsf{adap}}}$). In particular, this relation involves the relation induced by the encryption algorithm of a secret key encryption (SKE) scheme. For practical SKE schemes like AES, no efficient NIZKs that can prove such relations are known (even more so if one focuses on NIZKs proven secure in the standard model). Instead, one has to use a NIZK for an NP-complete language and reduce their relation to this NP-complete language. This is extremely inefficient and does not result in a scheme that is usable in practice. A slightly better instantiation of their construction—using structure preserving schemes, to enable purely algebraic NIZK statements—seems to require that the symmetric encryption scheme is essentially a group-based one-time-pad. But this then results in quadratic ciphertext sizes because one has to encrypt, per receiver, one group element for each group element of the symmetric ciphertext—which encrypts the public key of each receiver.

Our Main Contribution: Practical PKEBC Constructions. We resolve the limitations of prior constructions by presenting two new efficient PKEBC schemes with all the desired properties. Our ROM construction, in particular, is efficient enough for practical use. Our schemes are built from generic building blocks, but they are designed with efficient instantiability in mind. The construction that we prove in the standard model follows the Naor–Yung (NY) paradigm to guarantee consistency [56]. To reduce the NIZK-associated costs we make significant optimizations to the statements that have to be proven. Our random oracle model construction is based on the Fujisaki–Okamoto (FO) transformation [25]. Roughly, we leverage FO's reconstruction of ciphertexts during decryption to ensure the well-formedness of other recipients' ciphertexts. This new technique to obtain consistency allows for highly efficient (and practical, in fact) instantiations. More concretely:

$\mathsf{PKEBC}_{\mathsf{SM}}$: Our standard model construction is based on the NY paradigm and can be instantiated using an ElGamal-style mPKE and a simulation-sound Groth-Sahai NIZK [33]. We make significant simplifications to the NIZK language to obtain better efficiency.

$\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$: Our Random Oracle Model (ROM) construction is based on the FO-transform, and relies on a generic adaptively secure mPKE to obtain more compact ciphertexts (relative to the standard model construction). It can be instantiated with an mPKE based on post-quantum assumptions, like mPKEs based on lattices[5] or isogenies. In addition, we also show how to obtain an adaptively secure mPKE instantiation from hash proof systems.
A very nice feature of our construction is that receivers can even have different types of public keys set up. And in fact, even in that case our FO-construction provides high efficiency. We believe this property is helpful for the ongoing messenger interoperability efforts pushed by the European Union.

We prove that $\mathsf{PKEBC}_{\mathsf{SM}}$ and $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ are tightly secure under adaptive corruptions and $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{HPS}]}$ is tightly secure in the single-challenge setting with a security loss linear in the number of challenge ciphertexts in the multi-challenge setting. Figure 1 gives an overview of the efficiency of algebraic instantiations of our schemes.

Construction	$\lvert pk\rvert$	$\lvert c\rvert - \lvert m\rvert$	sec. loss	Assumption
$\mathsf{PKEBC}^{\mathsf{SXDH}}_{\mathsf{SM}}$	$2\lvert\mathbb{G}_1\rvert$	$(4n{+}1)\lvert\mathbb{G}_1\rvert{+}\lvert\mathbb{G}_2\rvert{+}\lvert\mathcal{P}\rvert{+}\lvert\pi_n\rvert{+}\lvert\mathsf{SKE}\rvert$	$\mathcal{O}(1)$	Pairings
$\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}_{\mathbb{G}}$	$2\lvert\mathbb{G}\rvert$	$(4n+1)\lvert\mathbb{G}\rvert + 2\lvert\mathcal{X}\rvert + \lvert\mathsf{SKE}\rvert$	$\mathcal{O}(1)$	ROM,DDH
$\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}[\mathsf{HPS}]]}_{\mathbb{G}}$	$\lvert\mathbb{G}\rvert$	$(n+2)\lvert\mathbb{G}\rvert + (n+2)\lvert\mathcal{X}\rvert + \lvert\mathsf{SKE}\rvert$	$\mathcal{O}(E)$	ROM,DDH

Fig. 1: Comparison of the algebraic instantiations of our three PKEBC constructions over some group $\mathbb{G}$, where n is the number of receivers, $|pk|$ denotes the size of public keys and $|c| - |m|$ is the ciphertext overhead. $\mathcal{P}$ is (half of) the codomain of a PRG, $\mathcal{X}$ is the codomain of a Random Oracle, $|\mathsf{SKE}|$ is the overhead introduced by secret key encryption and E is the number of encryption queries. $|\pi_n|$ is the size of a *simulation-sound* NIZK proof. The security loss refers to the loss introduced by the construction, not the loss between the primitives and underlying assumptions. In particular, obtaining simulation-sound NIZK proofs may incur some loss.

Efficient Consistent Off-The-Record Group Messaging. Our constructions, together with Maurer et al.'s $\mathsf{MDRS\text{-}PKE}$ compiler, allow for the first practical Off-The-Record Consistent messengers. This is because we already know how to efficiently instantiate Chakraborty et al.'s MDVS [10, Section 4.4.2]. In fact, if one is willing to move to the random oracle model, one can obtain instantiations

[5] We emphasize, however, that our construction requires perfect correctness.

that are efficient in practice. To be concrete, one can construct a suitable Σ protocol [57], for instance Schnorr's Σ protocol [59]—which is a proof of knowledge and allows for very efficient OR proofs [15]—and then apply the Fiat-Shamir heuristic [22] to it—which buys us simulation-soundness for free.

Our Secondary Contribution: Efficient Consistent Group Messaging in a PKI. In this paper, we also introduce Signcryption for Broadcast (SCBC), a variant of MDRS-PKE that provides all its guarantees except for Off-The-Record. This primitive allows more efficient instantiations of messengers when the Off-The-Record property is not needed.

We introduce two security models for SCBC: a game-based one and a composable one (in the Constructive Cryptography framework [53,49], to match the consistent communication channels that Liu-Zhang et al.'s messenger builds on [47]). We prove that a trivial variant of Maurer et al.'s MDRS-PKE compiler, where the MDVS scheme is replaced by a standard Digital Signature Scheme, allows to construct SCBC schemes whose efficiency is essentially the same as the underlying PKEBC's. In addition, we prove that our game-based security model for SCBC implies its composable security; this means that one can use SCBC to obtain a highly efficient, consistent messaging application that provides authenticity, confidentiality, and anonymity guarantees, by simply applying Liu-Zhang et al.'s messenger construction. An interesting feature of our composable protocol is that it is more efficient than the analogous one for MDRS-PKE [48]. This is because we can prove the protocol secure without having to add an extra receiver to ciphertexts. Liu-Zhang et al. rely on this additional receiver to enable a reduction from breaking the MDRS-PKE composable security notion to breaking one of its game-based notions [48]. We note that it was crucial to make a composable treatment of SCBC schemes, because without this treatment, we would not have been able to identify an additional security property that seems necessary from the underlying PKEBC, namely Adversarial Robustness. Finally, we believe that one can make an analogous improvement to the composable MDRS-PKE protocol given by Liu-Zhang et al. [48].

(Succinct) Broadcast Encryption. In our PKEBC construction (and thus also in the SCBC and MDRS-PKE schemes) ciphertexts grow still linear in the number of recipients. This is unavoidable because a recipient should be able to recover the list of recipients from the ciphertext. However, for PKEBC and SCBC schemes, this could be avoided by assuming that the recipient already knows the set of receivers and this is an input for Dec.[6]

Such broadcast encryption schemes with a succinct ciphertext were first studied in a setting with a central, trusted key generator [7,30,61,13,29] based on pairings, but such schemes can also be constructed in a decentralized setting without a trusted authority, see [26] for a comparison of different variants. Without

[6] For MDRS-PKE schemes, the deniability guarantee prevents us to achieve succinct ciphertexts even when the set of recipients is input to Dec.

a trusted setup, such schemes are currently only known from iO [8]. If one allows an initial trusted setup, such schemes can be built from witness encryption or registered ABE [23], pairings [65,43,38,27], or the novel [62] ℓ-succinct LWE assumption [12,11,63]. With the recent exception of [63], all of these schemes require a large common reference string that grows at least linearly in the maximum number of recipients of a ciphertext, which significantly impacts the efficiency gains they get from compact ciphertexts.

While these schemes are asymptotically more efficient than ours, they cannot outperform simply encrypting the message to each recipient separately for a small number of recipients, which frequently occurs in group messaging or e-mail communication.

Moreover, these schemes require communicating the list of recipients out-of-band. To achieve our security goals like consistency and anonymity, this would have to be done in a way that guarantees consistency and hides the set of recipients from outsiders. In other words, we would need a PKEBC scheme to achieve this anyway.

Thus, in this work, we focus on a setting where ciphertexts unavoidably grow linear in the number of users, and try to keep the per-recipient overhead as low as possible. This makes our scheme more flexible to use, because we do not need a trusted setup or out-of-band communication for the set of recipients.

2 Technical Overview

2.1 PKEBC in the standard model

Probably the most natural approach to achieve consistency for broadcast encryption is to encrypt the message to each receiver individually and to equip the ciphertext with a NIZK showing that the ciphertext was generated honestly, i.e., the ciphertext for each receiver was generated correctly and encrypts the same message. This was indeed the approach taken by the first PKEBC construction by Maurer et al. [51].

The main reason why this is very inefficient is that it results in ciphertexts that grow quadratic in the number of recipients. This is because, to achieve anonymity, the ciphertexts themselves cannot reveal the recipients, but the intended receiver should be able to recover the set of recipients, because otherwise, consistency clearly cannot be guaranteed. Thus, [51] include the list of recipients (or, more precisely, their public keys) in the message encrypted to everyone.

This drawback has been solved in Chakraborty et al. [10] by using hybrid encryption. In their scheme, each recipient only gets an encryption of a symmetric key, and the list of recipients, along with the actual message, gets encrypted only once in a symmetric ciphertext. The basic version of this idea still suffers from quadratic decryption time, because each (potential) recipient of the ciphertext has to try to decrypt the symmetric ciphertext for every key it obtains by decrypting one of the PKE ciphertexts. (Recall that, due to anonymity, it is not clear which of the PKE ciphertexts belong to the recipient, so they have to try all of them).

To solve this, they introduce a fast constant time check that allows each recipient to quickly assess if they decrypted the correct component of the ciphertext. However, their check comes with significant overhead (in particular, it incurs a per-user overhead) and we will describe later how we solve this problem more efficiently.

However, the main reason why the construction of [10] is not practical at all is that they also need to prove correctness of the secret key encryption in their NIZK. However, all practical SKE schemes, like AES, are very NIZK unfriendly, and proving the validity of a (long) SKE ciphertext would require using a NIZK for NP for a fairly big circuit.

Challenge 1: Excluding the SKE from the NIZK. It might seem, at first, that we can simply remove the SKE ciphertext from the NIZK relation. After all, if the NIZK would guarantee that every recipient obtains the same symmetric key, this is sufficient to guarantee that they also obtain the same plaintext when decrypting the symmetric ciphertext.

The difficulty here is that the NIZK also needs to guarantee that the list of recipient public keys, which was encrypted in the symmetric ciphertext, is the same list that was used to generate the asymmetric ciphertexts for each recipient. If we exclude the symmetric encryption from the NIZK relation, we lose that guarantee.

Our solution 1. We solve this problem by making the recipient list part of the statement of the NIZK (in [10], it was part of the NIZK). This allows a receiver to check the NIZK with the recipient list it got from the symmetric encryption, which restores the needed guarantee.

However, we are now facing a new problem: When the NIZK statement contains the set of recipients, the NIZK no longer hides the set of recipients. Thus, including the NIZK in the ciphertext would destroy anonymity. We resolve this issue by encrypting the NIZK with the symmetric encryption scheme.

Challenge 2: Efficient decryption checks. Even when we apply the modifications described above to the scheme of [10], we are still facing inefficiencies that come from their technique to achieve linear decryption time. They include, for each recipient, additional information that allows them to check efficiently if they obtained the correct symmetric key. This has two disadvantages.

- They incur a high per-receiver overhead. Concretely, they have to include 3 PKE ciphertexts per recipient (or 6 ciphertexts per recipient after applying the Naor–Yung transformation).
- The NIZK has to certify that all of these ciphertexts where generated correctly to guarantee consistency.

Our solution 2. We show that linear decryption time can also be achieved much more efficiently. Our check does not incur any per-recipient overhead, it does not need to be included in the NIZK relation, and it does not involve any public key operations.

Our new way to check a symmetric encryption scheme is very simple: First, we expand the key obtained by decrypting the PKE ciphertext using a PRG to twice its original length. The left half of this stretched key is included in the PKEBC ciphertext, and the right half is used as symmetric encryption key. An (alleged) recipient can then simply check if they got the correct symmetric key by checking if they get the same left half after applying the PRG.

Pseudorandomness guarantees that the right half is still suitable as an SKE secret key, even after publishing the left half.

To guarantee that this check works, we need a non-standard, but very plausible assumption from the PRG. Namely, we need that it is hard to find two different inputs that collide on the left half of their output. In particular, hash functions with partial collision resistance satisfy this.

Naor–Yung. Like [10], we use the Naor–Yung transformation [56] in our construction. That is, each user generates two public keys and we encrypt to both of them for each recipient. We do not need an extra NIZK for the Naor–Yung transformation, because the NIZK we are already using for consistency already provides all the guarantees we need for the Naor–Yung transformation.

The Naor–Yung transformation achieves two important goals. Firstly, we achieve CCA security even when we started only with a CPA secure PKE scheme. Secondly, it allows us to prove security tightly under adaptive corruptions. Note that a generic guessing argument would incur a security loss of $\Omega(2^n)$ for a system with n users (even in a simple single-challenge security game), because the reduction would have to guess for each user if it gets corrupted or included in the challenge ciphertext.

Multi-Recipient Public Key Encryption (mPKE*).* We further reduce the per-receiver overhead by using an mPKE [44] to efficiently bundle the ciphertexts. In our group-based instantiation, we use the mPKE by [44], which essentially consists of ElGamal ciphertexts with shared randomness.

Summary. We show a new PKEBC construction in the standard model that significantly improves upon [10]. In our scheme, the only per-recipient overhead are two (due to Naor–Yung) ciphertexts per recipient and a NIZK that proves that all of them encrypt the same message. The NP-relation for our NIZK is very simple, which allows us to use efficient NIZKs, such as the Groth-Sahai NIZK [33].

Challenge 3: Avoiding NIZKs. The main bottleneck in our PKEBC construction is, despite our optimizations and significant improvements the NIZK. Thus, to further improve efficiency, we need to replace them.

2.2 PKEBC in the random oracle model

Our Solution 3. We show that we can get rid of the NIZKs and obtain an even more efficient PKEBC in the random oracle model by leveraging the idea of the Fujisaki–Okamoto transformation [24,36].

The Fujisaki–Okamoto transformation first makes a PKE deterministic by generating the randomness for encryption by hashing the message with the random oracle, and then uses the deterministic PKE to encrypt a random key (KEM). This allows a receiver to check a ciphertext for validity by recomputing it, effectively replacing the need for a NIZK.

We show that this idea can also be used in the context of broadcast encryption to achieve consistency. Therefore, we apply the FO transformation to an mPKE, and use the trick we described in Solution 2 to achieve linear decryption time. The random oracle can be used here to instantiate the PRG.

The FO transform is used in this construction to achieve CCA security and consistency. Moreover, we show that it also preserves other desirable properties, such as anonymity.

Challenge 4: Adaptive corruptions for mPKE*.* This construction is tightly secure, but it requires the mPKE to satisfy a strong security notion, namely IND-CPA security with adaptive corruptions. This notion is generically implied by the standard IND-CPA security via a standard guessing argument, but the reduction is not tight. It incurs a fairly large loss that is exponential in the number of users registered in the system.

Our Solution 4. We show that we can achieve *tight* adaptive security with corruptions generically for any mPKE with a tamed-down version of the Naor–Yung transform, as follows. We generate two public keys per user and include both of them for each recipient of the ciphertext. Each recipient keeps one of the two secret keys at random and uses it for decryption.

This is sufficient to handle adaptive corruptions tightly, with a similar argument as in the standard model scheme. Importantly, we do not need to include a NIZK here, because we do not use the Naor–Yung transformation to achieve CCA security (which is already achieved by the FO transformation).

Challenge 5: Avoiding NY/key doubling. Now that we have eliminated the NIZKs, our PKEBC scheme is already very efficient. However, due to the tamed-down Naor–Yung transform, it still needs two public keys, and we need to produce two PKE ciphertexts for each of them (which in our group-based instantiations translates to two group elements in each public key and two group elements per recipient in the ciphertext). Thus, a natural question is whether we can avoid this.

Our solution 5. We answer this in the affirmative with our final construction, which builds an mPKE from a hash proof system (HPS) that natively supports tight adaptive corruptions. With the HPS, we get a group based instantiation where we have just one group element in each user's public key and include just one group element per user in the ciphertext.

In our HPS-based mPKE, each ciphertext consists of a statement of the hash-proof system language and, for each recipient, a copy of the symmetric key that is blinded by the hash proof of the statement under the recipient's public

key. This structure allows us to employ a proof strategy where we switch the statement (in one challenge ciphertext after the other) from a statement in the language to a statement not in the language. The pairwise universality of the hash proof system then guarantees that the symmetric key is hidden.

This proof strategy is tight in the single challenge setting and incurs a small security loss (linear in the number of challenge ciphertexts) in the more realistic multi-challenge setting. Note that this is still dramatically better than the generic guessing argument, which had a security loss exponential in the number of users registered in the system.

3 Preliminaries

We denote the arity of a vector $\vec{x}$ by $|\vec{x}|$ and for $i \in [|\vec{x}|]$ its i-th element by x_i. We write $\alpha \in \vec{x}$ to denote $\exists i \in \{1, \ldots, |\vec{x}|\}$ with $\alpha = x_i$. We write $\mathrm{Set}(\vec{x})$ to denote the set induced by vector $\vec{x}$, i.e. $\mathrm{Set}(\vec{x}) \coloneqq \{x_i \mid x_i \in \vec{x}\}$.

Vector Notation: Throughout the paper we frequently use vectors. We use upper case letters to denote vectors of parties, and lower case letters to denote vectors of artifacts such as public keys, sequences of random coins, etc. Moreover, we use the convention that if $\vec{V}$ is a vector of parties, then $\vec{v}$ denotes $\vec{V}$'s corresponding vector of public keys. For example, for a vector of parties $\vec{V} \coloneqq (\mathrm{Bob}, \mathrm{Charlie})$, $\vec{v} \coloneqq (\mathtt{pk}_{\mathrm{Bob}}, \mathtt{pk}_{\mathrm{Charlie}})$ is $\vec{V}$'s corresponding vector of public keys. In particular, V_1 is Bob and v_1 is Bob's public key $\mathtt{pk}_{\mathrm{Bob}}$, and V_2 is Charlie and v_2 is Charlie's public key $\mathtt{pk}_{\mathrm{Charlie}}$. More generally, for a vector of parties $\vec{V}$ with corresponding vector of public keys $\vec{v}$, V_i's public key is v_i, for $i \in \{1, \ldots, |\vec{V}|\}$.

Probability Distributions: For two probability distributions $\mathbf{X}$ and $\mathbf{Y}$, we write $\mathbf{X} \equiv \mathbf{Y}$ if they are the same probability distribution. Similarly, for two sequences of conditional probability distributions $\mathbf{X}$ and $\mathbf{Y}$, we write $\mathbf{X} \equiv \mathbf{Y}$ if they are the same sequences of conditional probability distributions. Consider a discrete random variable X with set of possible outcomes $\mathcal{X}$. The min-entropy of X is $\mathrm{H}_\infty(X) \coloneqq -\log\big(\max_{x_i \in \mathcal{X}} \Pr[X = x_i]\big)$. Consider a function $f : \mathcal{X} \to \mathcal{Y}$. The distribution induced by applying function f to X is denoted $f(X)$; for $y \in \mathcal{Y}$,

$$\Pr[f(X) = y] \coloneqq \sum_{x \in \mathcal{X}} b_x^y \cdot \Pr[X = x],$$

where b_x^y is an indicator variable with $b_x^y = 1 \Leftrightarrow f(x) = y$.

Sampleable Sets: A set S is sampleable if there is an explicit sampler Samp_S that outputs elements $s \in S$; we write $s \leftarrow S$ to denote $s \leftarrow \mathsf{Samp}_S$. We write $\mathrm{H}_\infty(\mathsf{Samp}_S)$ to denote the min-entropy associated with the discrete random variable sampled by Samp_S, and analogously for $\mathrm{H}_2(\mathsf{Samp}_S)$.

(Binary) Relations and Languages: A binary relation R on sets $\mathcal{X}$ (statements) and $\mathcal{W}$ (witnesses) is a subset of their Cartesian product, i.e. $R \subseteq \mathcal{X} \times \mathcal{W}$. We define the language induced by R as $L_R \coloneqq \{x \in \mathcal{X} \mid \exists w \in \mathcal{W} : (x, w) \in R\}$. Let Samp_R be a sampler for (binary) relation R; when clear from context, we write $(x, w) \leftarrow R$ to denote $(x, w) \leftarrow \mathsf{Samp}_R$. Consider function Stmt that maps statement-witness pairs to their respective statements—i.e. for $(x, w) \in R$, $\mathsf{Stmt}(x, w) \coloneqq x$; Samp_R is γ-Statement-Min-Entropic if $\mathrm{H}_\infty(\mathsf{Stmt}(\mathsf{Samp}_R)) \geq \gamma$.

3.1 Game-Based Security Definitions

We follow the style from [39] in defining game-based security notions. Concretely, for event-based security notions—which capture adversaries' ability of causing some "bad" event ξ defined by the notion to occur when interacting with a set of oracles (which define the security game), say $\mathcal{O}_1, \ldots, \mathcal{O}_n$—we denote an adversary $\mathbf{A}$'s probability of triggering ξ by $\Pr\left[\mathbf{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n} \Rightarrow \xi\right]$ and define $\mathbf{A}$'s advantage as $Adv^{\xi}_{\mathcal{O}_1,\ldots,\mathcal{O}_n}(\mathbf{A}) \coloneqq \Pr\left[\mathbf{A}^{\mathcal{O}_1,\ldots,\mathcal{O}_n} \Rightarrow \xi\right]$. When clear from context, we write instead $Adv^{\xi}(\mathbf{A}) \coloneqq \Pr\left[\mathbf{A} \Rightarrow \xi\right]$, omitting the oracles.

For distinction-based notions the goal of an adversary $\mathbf{A}$ is correctly guessing the challenge bit $\mathbf{b} \in \{0, 1\}$ by interacting with a set of oracles $\mathcal{O}^{\mathbf{b}}_1, \ldots, \mathcal{O}^{\mathbf{b}}_n$—whose behavior depends on the challenge bit $\mathbf{b}$. $\mathbf{A}$ wins if the bit $\mathbf{b} \in \{0, 1\}$ it outputs (its guess) matches $\mathbf{b}$; we denote this by $\mathbf{A}^{\mathcal{O}^{\mathbf{b}}_1,\ldots,\mathcal{O}^{\mathbf{b}}_n} \Rightarrow \mathtt{win}$. Some distinction-based notions specify disallowed oracle queries; if an adversary $\mathbf{A}$ makes such query, its output bit is replaced with one picked uniformly at random. For a security notion X defining oracles $\mathcal{O}^{\mathbf{b}}_1, \ldots, \mathcal{O}^{\mathbf{b}}_n$ (for $\mathbf{b} \in \{0, 1\}$), we denote $\mathbf{A}$'s winning probability by $\Pr\left[\mathbf{A}^{\mathcal{O}^{\mathbf{b}}_1,\ldots,\mathcal{O}^{\mathbf{b}}_n} \Rightarrow \mathtt{win}\right]$, and define $\mathbf{A}$'s advantage as

$$Adv^{\mathsf{X}}(\mathbf{A}) \coloneqq \left|\Pr\left[\mathbf{A}^{\mathcal{O}^{\mathbf{0}}_1,\ldots,\mathcal{O}^{\mathbf{0}}_n} \Rightarrow \mathtt{win}\right] + \Pr\left[\mathbf{A}^{\mathcal{O}^{\mathbf{1}}_1,\ldots,\mathcal{O}^{\mathbf{1}}_n} \Rightarrow \mathtt{win}\right] - 1\right|.$$

For a distinction-based security notion X, for $\mathbf{b} \in \{0, 1\}$, we write $\mathbf{G}^{\mathsf{X}}_{\mathbf{b}}$ as shorthand notation for the oracle set $\mathcal{O}^{\mathbf{b}}_1, \ldots, \mathcal{O}^{\mathbf{b}}_n$. Similarly, for an event-based security notion X, we write $\mathbf{G}^{\mathsf{X}}$ as shorthand notation for the oracles $\mathcal{O}_1, \ldots, \mathcal{O}_n$ defined by X.

3.2 Public Key Encryption for Broadcast

As introduced by Maurer et al. [51], a Public Key Encryption for Broadcast (PKEBC) scheme is a 4-tuple $\mathsf{PKEBC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of PPTs where:

$\mathsf{Setup}(1^k)$: outputs public parameters $\mathtt{pp}$;
$\mathsf{Gen}(\mathtt{pp})$: outputs a key-pair $(\mathtt{pk}, \mathtt{sk})$;
$\mathsf{Enc}_{\mathtt{pp}}(\vec{v} \coloneqq (\mathtt{pk}_1, \ldots, \mathtt{pk}_{|\vec{v}|}), m)$: outputs a ciphertext c;
$\mathsf{Dec}_{\mathtt{pp}}(\mathtt{sk}, c)$: outputs a vector/message pair $(\vec{v}, m)$ (or $\perp$ if c fails to decrypt).

We require that the public key vector $\vec{v}$ input to Enc contains no duplicates, i.e. $|\mathrm{Set}(\vec{v})| = |\vec{v}|$. This requirement is standard [44][7]—although not in existing

[7] In [44], Kurosawa implicitly makes the same requirement by defining the syntax of Multi-Recipient Public Key Encryption schemes as taking as input a set of parties rather than a vector.

PKEBC models—and enables more efficient instantiations of our construction's main building block: Multi-Recipient Public Key Encryption (mPKE).[8]

We now introduce the PKEBC security model from [10], but following the style of [39]. For $\mathsf{PKEBC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and security parameter k, the notions ahead provide adversaries with access to the following oracles:

$\mathcal{O}_{PP}$: on the first query compute $\mathtt{pp} \leftarrow \mathsf{Setup}(1^k)$; output $\mathtt{pp}$;
$\mathcal{O}_{SK}(B_j)$: if there was a prior query $\mathcal{O}_{SK}(B_j)$, simply look up and return the previously generated key for B_j; otherwise, store $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathsf{Gen}(\mathtt{pp})$ as B_j's key-pair, and output $(\mathtt{pk}_j, \mathtt{sk}_j)$;
$\mathcal{O}_{PK}(B_j)$: $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathcal{O}_{SK}(B_j)$; output $\mathtt{pk}_j$;
$\mathcal{O}_E(\vec{V}, m)$: let $\vec{v} \coloneqq (\mathcal{O}_{PK}(V_1), \ldots, \mathcal{O}_{PK}(V_{|\vec{V}|}))$; output $c \leftarrow \mathsf{Enc}_{\mathtt{pp}}(\vec{v}, m)$;
$\mathcal{O}_D(B_j, c)$: $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathcal{O}_{SK}(B_j)$; compute $(\vec{v}, m) \leftarrow \mathsf{Dec}_{\mathtt{pp}}(\mathtt{sk}_j, c)$; if, $(\vec{v}, m) = \perp$, or for some $l \in \{1, \ldots, |\vec{V}|\}$, there is no party B_j that was previously input to either $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ or $\mathcal{O}_D$ such that $v_l = \mathcal{O}_{RPK}(V_l)$, output $\perp$; else, output $(\vec{v}, m)$;
$\mathcal{O}_{Publish}(c)$: no output. (Only used to define Adversarial Replay-Correctness.)

As already mentioned, we adopt the style of [39] in defining game-based security notions. However, before stating a security notion as in [39], we show how one would write a security notion in the more standard style, when one is interested in concrete security:

Definition 1 (Correctness). *Correctness game* $\mathbf{G}^{\mathsf{Corr}}$ *provides adversaries with access to oracles* $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ *and* $\mathcal{O}_D$. *Define the following event* $\neg\mathsf{Corr}$*: An adversary* $\mathbf{A}$ *makes a query* $\mathcal{O}_E(\vec{V}, m)$ *that outputs a ciphertext* c*, and then makes a query* $\mathcal{O}_D(B_j, c)$*, where* $B_j \in \vec{V}$*, that outputs a pair* $(\vec{v'}, m')$ *such that* $(\vec{v'}, m') \neq (\vec{v}, m)$*,* $\vec{v}$ *being the vector of public keys corresponding to vector of parties* $\vec{V}$.

A PKEBC *scheme is* $(n, d_E, q_E, q_D, \varepsilon, t)$*-Correct if no adversary* $\mathbf{A}$ *running in time* t*, querying* $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ *and* $\mathcal{O}_D$ *on at most* n *different parties, making at most* q_E *and* q_D *queries to* $\mathcal{O}_E$ *and* $\mathcal{O}_D$*, respectively, with the sum of lengths of the party vectors input to* $\mathcal{O}_E$ *being at most* d_E*, satisfies*

$$Adv^{\mathsf{Corr}}(\mathbf{A}) \coloneqq \Pr\left[\mathbf{A}^{\mathcal{O}_{PP}, \mathcal{O}_{SK}, \mathcal{O}_{PK}, \mathcal{O}_E, \mathcal{O}_D} \Rightarrow \neg\mathsf{Corr}\right] \geq \varepsilon.$$

To avoid the unnecessary and distracting jargon above, we instead only define the security events that should not occur; for example, the incorrectness event ($\neg\mathsf{Corr}$) is defined as follows:

[8] This assumption is propagated to schemes built upon PKEBC, which means we end up with a weaker slightly ideal application semantics; however, it has no effect on (existing) protocols that build on these composable notions (in particular [47]). This is because, to the best of our knowledge, and as one may expect, there is no real application that could potentially benefit from include the same receiver more than once. (Even if there were, this information could still be carried in the underlying message instead.)

Definition 2 (Correctness: $\neg$Corr). *A query $\mathcal{O}_E(\vec{V}, m)$ outputs ciphertext c; a later query $\mathcal{O}_D(B_j \in \vec{V}, c)$ outputs either $\perp$ or some $(\vec{v}', m')$ with $(\vec{v}, m) \neq (\vec{v}', m')$, where $\vec{v}$ is $\vec{V}$'s corresponding vector of public keys.*

In Section 4.1 we state what it means for a PKEBC scheme to be secure with respect to concrete security parameters.

Definitions 3, 4 and 5 are only introduced to simplify the security proofs of the SCBC construction. In turn, the analogous SCBC notions are introduced to simplify the composable proof we give in appendix.

Definition 3 (Replay-Correctness: $\neg$RCorr). *An $\mathcal{O}_E$ query outputs c; a later $\mathcal{O}_E$ query outputs c' with $c = c'$.*

Definition 4 (Adversarial Replay-Correctness: $\neg$aRCorr). *Oracle $\mathcal{O}_{Publish}$ is queried on an input c; a later $\mathcal{O}_E$ query outputs c' with $c = c'$.*

Definition 5 (Public-Key Collision Resistance). $\mathsf{PKEBC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is n-Party ε-Public-Key Collision Resistant if*

$$\Pr\left[\left|\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}\right| < n \;\middle|\; \begin{array}{c} \mathtt{pp} \leftarrow \mathsf{Setup}(1^k) \\ (\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{Gen}(\mathtt{pp}), i \in [n] \end{array}\right] \leq \varepsilon.$$

Definition 6 (Self-Consistency: $\neg$Self-Cons). *A query $\mathcal{O}_D(B_i, c)$ outputs some pair $(\vec{v}, m) \neq \perp$ and $\mathtt{pk}_i \notin \vec{v}$.*

Definition 7 (Consistency: $\neg$Cons). *A query $\mathcal{O}_D(B_j, c)$ outputs some pair $(\vec{v}, m) \neq \perp$; a later query $\mathcal{O}_D(B_i, c)$, where $\mathtt{pk}_i \in \vec{v}$ outputs $\perp$ or a different pair $(\vec{v}', m') \neq (\vec{v}, m)$.*

Definition 8 (Robustness: $\neg$Rob). *A query $\mathcal{O}_E(\vec{V}, m)$ outputs a ciphertext c; a later query $\mathcal{O}_D(B_j \notin \vec{V}, c)$ does not output $\perp$.*

The games defined by the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ notion from [10] provide adversaries with access to oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$ and $\mathcal{O}_{Publish}$ from before, as well as to oracles $\mathcal{O}_E$ and $\mathcal{O}_D$ defined below:

$\mathcal{O}_E^{\mathbf{b}}\big((\vec{V}_0, m_0), (\vec{V}_1, m_1)\big)$: output encryption of $m_{\mathbf{b}}$ under $\vec{v}_{\mathbf{b}}$;
$\mathcal{O}_D(B_j, c)$: if c was output by $\mathcal{O}_E$, output $\mathtt{test}$; else, output $(\vec{v}, m) \leftarrow \mathsf{Dec}_{\mathtt{pp}}(\mathtt{sk}_j, c)$.

Definition 9 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ Security). $\mathcal{O}^{\mathbf{b}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}} \coloneqq (\mathcal{O}_{PP}, \mathcal{O}_{SK}, \mathcal{O}_{PK}, \mathcal{O}_E^{\mathbf{b}}, \mathcal{O}_D, \mathcal{O}_{Publish})$.
Disallowed: *1. queries $\mathcal{O}_E^{\mathbf{b}}((\vec{V}_0, m_0), (\vec{V}_1, m_1))$ with $(|\vec{V}_0|, |m_0|) \neq (|\vec{V}_1|, |m_1|)$; 2. making queries $\mathcal{O}_E^{\mathbf{b}}((\vec{V}_0, \cdot), (\vec{V}_1, \cdot))$ and $\mathcal{O}_{SK}(B_j \in \vec{V}_0 \cup \vec{V}_1)$.*

4 Efficient PKEBC Instantiation

We now give our constructions plus a new security notion: Adversarial Robustness.

4.1 New Game-Based Notion: Adversarial Robustness

We introduce Adversarial Robustness (aRob) because the analogous SCBC notion allows for more efficient (composable) reductions and our SCBC construction's aRob directly reduces to the aRob of the underlying PKEBC. Concretely, prior work [47] relied on an additional party in the public parameters—which simplifies the reduction but increases ciphertext sizes and the cost to encrypt and decrypt (due to the extra party).[9] Despite more involved (composable) security reductions, relying on Adversarial Robustness allows to avoid the shortcomings of prior work and obtain an overall more efficient instantiation.

Adversarial Robustness guarantees that if the decryption of a ciphertext c by an honest receiver B_j yields $(\vec{v}, m) \neq \perp$, then the decryption of c by any honest receiver B_i whose public key $\mathtt{pk}_i$ is not in $\vec{v}$ fails (i.e. outputs $\perp$). This notion implies the Robustness notions from [51,10] (Lemma 2): one can reduce breaking a PKEBC's Robustness to breaking its Adversarial Robustness or Correctness with similar advantage (Lemma 2).

Adversarial Robustness makes use of the oracles defined in Section 3.2.

Definition 10 (Adversarial Robustness: ¬aRob). *A query $\mathcal{O}_D(B_j, c)$ outputs some pair $(\vec{v}, m) \neq \perp$; a later query $\mathcal{O}_D(B_i, c)$, with $\mathtt{pk}_i \notin \vec{v}$, does not output $\perp$.*

An adversary $\mathbf{A}$ (ε, t)-breaks the (n, d_E, q_E, q_D, q_P)-Correctness (resp. -Replay-Correctness, -Adversarial Replay-Correctness, -Self-Consistency, -Consistency, -Adversarial Robustness, -Robustness and -$\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ security) of a scheme PKEBC if $\mathbf{A}$ runs in time at most t, queries $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most n different parties, makes at most q_E, q_D and q_P queries to $\mathcal{O}_E$, $\mathcal{O}_D$ and $\mathcal{O}_{Publish}$, respectively, with the sum of lengths of the party vectors input to $\mathcal{O}_E$ being at most d_E, and $Adv^{\neg\mathsf{Corr}}(\mathbf{A})$ (resp. $Adv^{\neg\mathsf{RCorr}}(\mathbf{A})$, $Adv^{\neg\mathsf{aRCorr}}(\mathbf{A})$, $Adv^{\neg\mathsf{Self\text{-}Cons}}(\mathbf{A})$, $Adv^{\neg\mathsf{Cons}}(\mathbf{A})$, $Adv^{\neg\mathsf{aRob}}(\mathbf{A})$, $Adv^{\neg\mathsf{Rob}}(\mathbf{A})$, $Adv^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}(\mathbf{A})$) is greater or equal to ε. Finally, we say that PKEBC is

$$\begin{aligned}(&\varepsilon_{\mathsf{Corr}}, \varepsilon_{\mathsf{RCorr}}, \varepsilon_{\mathsf{aRCorr}}, \varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}, \varepsilon_{\mathsf{Self\text{-}Cons}}, \varepsilon_{\mathsf{Cons}},\\ &\varepsilon_{\mathsf{aRob}}, \varepsilon_{\mathsf{Rob}}, \varepsilon_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}, t,\\ &n, d_E, q_E, q_D, q_P)\text{-secure},\end{aligned}$$

if there is no adversary $\mathbf{A}$ that breaks any of a PKEBC's security notions with the respective parameters, as above, and PKEBC is n-Party $\varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}$-Public-Key Collision Resistant.

4.2 Construction in the Standard Model

We now present our standard model construction, denoted $\mathsf{PKEBC}_{\mathsf{SM}}$.

[9] [47] studies MDRS-PKE schemes, not SCBC, yet one can use an analogous proof strategy for the case of SCBC.

Algorithm 1 Standard Model: Generic $\mathsf{PKEBC}_{\mathsf{SM}}$ construction.

$\mathsf{Setup}(1^k)$: **return** $\mathtt{pp} := (1^k, \mathtt{crs} \leftarrow \mathsf{NIZK.Setup}(1^k))$

$\mathsf{Gen}(\mathtt{pp})$
- **for** $i \in \{0,1\}$**:** $(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{mPKE.Gen}(1^k)$
- $b \xleftarrow{\$} \{0,1\}$
- **return** $(\mathtt{pk} := (\mathtt{pk}_0, \mathtt{pk}_1), \mathtt{sk} := (b, \mathtt{sk}_b))$

$\mathsf{Enc}(\vec{v} = ((\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}), \ldots, (\mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})), m)$:
- $R \xleftarrow{\$} \mathcal{R}$
- $\vec{u} := (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$
- $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{u}, R; \mathtt{r}_\$)$
- $\pi \leftarrow \mathsf{NIZK.Prv}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L_{\mathsf{Cons}}^{\mathsf{PKEBC}}, w := (\mathtt{r}_\$, R))$
- $(\tau, K) \leftarrow \mathsf{PRG.F}(R)$
- $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}, m, \pi))$
- **return** $c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}})$

$\mathsf{Dec}((b, \mathtt{sk}), c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$:
- $n := |c_{\mathsf{mPKE}}|/2$
- $K \leftarrow \perp$
- **for** $i = 1, \ldots, n$ **:**
  - $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i - 1 + b)$
  - $R_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
  - $(\tau_i, K_i) \leftarrow \mathsf{PRG.F}(R_i)$
  - **if** $\tau_i = \tau$**:** $K \leftarrow K_i$**; break**
- **if** $K = \perp$**: return** $\perp$
- $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
- **if** $(\vec{v}, m, \pi) = \perp \vee \mathtt{pk} \neq v_i$**: return** $\perp$
- $((\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}), \ldots, (\mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})) := \vec{v}$
- $\vec{u} := (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$
- **if** $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L_{\mathsf{Cons}}^{\mathsf{PKEBC}}, \pi) = 0$ **:** **return** $\perp$
- **return** $(\vec{v}, m)$

Algorithm 2 Standard Model: Algebraic instantiation $\mathsf{PKEBC}_{\mathsf{SM}}^{\mathsf{SXDH}}$.

$\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e, p)$

$\mathsf{Setup}(1^k)$
- $\mathtt{crs} \leftarrow \mathsf{GS.Setup}(1^k, \mathsf{par})$
- **return** $\mathtt{pp} := (1^k, \mathtt{crs})$

$\mathsf{Gen}(1^k)$: $\vec{x} \xleftarrow{\$} \mathbb{Z}_p^2$, $b \xleftarrow{\$} \{0,1\}$
- $\vec{X} := (g_1^{x_1}, g_1^{x_2})$
- **return** $(\mathtt{pk} := \vec{X}, \mathtt{sk} := (x_b, b))$

$\mathsf{Enc}(\vec{v} := (\vec{X}_1, \ldots, \vec{X}_n), m)$: $\vec{X}_i := (X_{i,0}, X_{i,1})$
- $Y \xleftarrow{\$} \mathbb{G}_1$, $r \xleftarrow{\$} \mathbb{Z}_p$
- $R := (g_1)^r$, $R' := (g_2)^r$
- **for** $i \in [n]$**:** $\vec{C}_i := (\vec{X}_i)^r \cdot Y$
- $\pi \leftarrow \mathsf{GS.Prv}(\mathtt{crs}, (g_1, R, \vec{X}, \vec{C}, g_2, R') \in R^{\mathsf{GS}}, Y)$
- $(\tau, K) \leftarrow \mathsf{PRG}_{\mathbb{G}_1}.\mathsf{F}(Y)$
- $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}, m, \pi))$
- **return** $c := (R, R', \vec{C}, \tau, c_{\mathsf{SKE}})$

$\mathsf{Dec}(\mathtt{sk} := (x_b, b), c := (R, R', \vec{C}, \tau, c_{\mathsf{SKE}}))$
- $n := |\vec{C}|/2$, $R_{\mathtt{sk}} := R^{-x_b}$
- $Y \leftarrow \perp$
- **for** $i = 1, \ldots, n$ **:**
  - $Y_i := R_{\mathtt{sk}} \cdot C_{i,b}$
  - $(\tau_i, K_i) \leftarrow \mathsf{PRG}_{\mathbb{G}_1}.\mathsf{F}(Y_i)$
  - **if** $\tau_i = \tau$**:** $Y \leftarrow Y_i$**; break**
- **if** $Y = \perp$**: return** $\perp$
- $(\tau, K) \leftarrow (\tau_i, K_i)$
- $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
- **if** $(\vec{v}, m, \pi) = \perp \vee |\vec{v}| \neq n$**: return** $\perp$
- **if** $\mathtt{pk} \neq v_i$**: return** $\perp$
- **if** $\mathsf{GS.Vrfy}((g_1, R, \vec{X}, \vec{C}, g_2, R') \in R^{\mathsf{GS}}, \pi) = 0$**:** **return** $\perp$
- **return** $(\vec{v}, m)$

Security Analysis For relation

$$R_{\mathsf{Cons}}^{\mathsf{PKEBC}} := \left\{ \left((\vec{u}, c_{\mathsf{mPKE}}), (\mathtt{r}_\$, R) \right) \mid c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{u}, R; \mathtt{r}_\$) \right\},$$

language $L_{\mathsf{Cons}}^{\mathsf{PKEBC}}$ is the natural language induced by $R_{\mathsf{Cons}}^{\mathsf{PKEBC}}$, i.e.

$$L_{\mathsf{Cons}}^{\mathsf{PKEBC}} := \left\{ (\vec{u}, c_{\mathsf{mPKE}}) \mid \exists (\mathtt{r}_\$, R) : \left((\vec{u}, c_{\mathsf{mPKE}}), (\mathtt{r}_\$, R) \right) \in R_{\mathsf{Cons}}^{\mathsf{PKEBC}} \right\}.$$

Theorem 1 (Informal). *If 1.* mPKE *is correct and tightly* IND-CPA *and* IK-CPA *secure under adaptive corruptions, 2.* NIZK *is tightly complete, sound, zero-knowledge and simulation-sound secure, 3.* PRG *is pseudorandom and left-collision resistant, 4.* SKE *is Correct, tightly* OT-IND-CCA *secure (under no corruptions), and its decryption algorithm is deterministic, then* $\mathsf{PKEBC}_{\mathsf{SM}}$ *tightly satisfies:*

1. *Correctness (Theorem 6);*
2. *Replay-Correctness (Theorem 7);*
3. *Adversarial Replay-Correctness (Theorem 8);*
4. *Public-Key Collision Resistance (Theorem 9);*
5. *Self-Consistency (Theorem 10);*
6. *Consistency (Theorem 11);*
7. *Adversarial Robustness (Theorem 12);*
8. *and* $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ *security (Theorem 13)*

all under adaptive corruptions.

Efficiency Analysis

Generic Efficiency: Let $\mathcal{X}$ be PRG's codomain. Public keys consist of two mPKE keys and the ciphertext overhead is $|c_{\mathsf{mPKE}}| + 2n|\mathtt{pk}_{\mathsf{mPKE}}| + |\pi_{\mathsf{NIZK}}| + |\mathcal{X}|/2 + |\mathsf{SKE}|$ where $|\mathtt{pk}_{\mathsf{mPKE}}|$ and $|c_{\mathsf{mPKE}}|$ are mPKE's public key and ciphertext sizes, $|\pi_{\mathsf{NIZK}}|$ is the NIZK's proof size, and $|\mathsf{SKE}|$ is the overhead introduced by SKE (τ contributes half of $|\mathcal{X}|$). Importantly, Enc and Dec each invoke SKE.Enc and SKE.Dec exactly once.

Algebraic Instantiation: Instantiating mPKE in Algorithm 1 with the ElGamal-style $\mathsf{mPKE}_{\mathbb{G}}$ [44] over a prime-order group $\mathbb{G}$ results in the following relation:

$$R_{\mathsf{Cons}}^{\mathsf{PKEBC}_{\mathsf{SM}}^{\mathbb{G}}} := \left\{ \left((\vec{X}, (\vec{C}, R)), (Y, r)\right) \mid g^r = R \wedge \bigwedge_{i \in [|\vec{C}|]} \vec{C}_i = (\vec{X}_i)^r \cdot Y \right\},$$

where $\vec{X}$ and $\vec{C}$ are *pairs* of vectors of group elements. To simplify notation, we understand operations and equations component-wise: For example $\vec{C}_i = (\vec{X}_i)^r \cdot Y$ stands for two equations $\vec{C}_{i,0} = (\vec{X}_{i,0})^r \cdot Y$ and $\vec{C}_{i,1} = (\vec{X}_{i,1})^r \cdot Y$.

We will instantiate the NIZK with a Groth-Sahai NIWI proof system [33] and apply the construction of [37] to obtain a simulation-sound NIZK. The Groth-Sahai NIWI-proof for the language induced by the above relation can be reduced to a single product pairing equation (PPE) with one variable by making two observations:
(1) eliminating Y: for any $i \in [|\vec{C}|]$ we have $Y = \vec{C}_i \cdot ((\vec{X}_i)^r)^{-1}$. We can replace Y by $\vec{C}_1 \cdot ((\vec{X}_1)^r)^{-1}$ in all but one of the equations $\vec{C}_i = (\vec{X}_i)^r \cdot Y$ and instead use the equivalent equation (for $i \geq 2$):

$$\vec{C}_1 \cdot ((\vec{X}_1)^r)^{-1} = \vec{C}_i \cdot ((\vec{X}_i)^r)^{-1}$$

That is, if the equation $Y = \vec{C}_1 \cdot ((\vec{X}_1)^r)^{-1}$ holds, the remaining equations can be expressed without witness Y.

(2) eliminating r: Given a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, we can remove r from all equations: embedding public inputs $\vec{X}, \vec{C}, R$ and witness Y into $\mathbb{G}_1$ and introducing a constant $R' \in \mathbb{G}_2$ such that $e(R, g_2) = e(g_1, R')$ allows (using non-degeneracy in the first step):

$$\begin{aligned} Y = \vec{C}_i \cdot ((\vec{X}_i)^r)^{-1} &\iff e(Y, g_2) = e(\vec{C}_i \cdot ((\vec{X}_i)^r)^{-1}, g_2) \\ &\iff e(Y, g_2) = e(\vec{C}_i, g_2) \cdot (e(\vec{X}_i, R')^{-1}). \end{aligned}$$

Combining both simplifications, we can express $R_{\mathsf{Cons}}^{\mathsf{PKEBC}_{\mathsf{SM}}^{\mathbb{G}}}$ by the equivalent relation

$$\begin{aligned} R^{\mathsf{GS}} := \Bigg\{ ((g_1, R, \vec{X}, \vec{C}, g_2, R'), Y) \;\Bigg|\; & \\ & e(Y, g_2) = e(\vec{C}_1, g_2) \cdot (e(\vec{X}_1, R')^{-1}) \;\wedge\; e(R, g_2) = e(g_1, R') \\ & \wedge \bigwedge_{2 \le i \le |\vec{C}|} e(\vec{C}_1, g_2) \cdot (e(\vec{X}_1, R')^{-1}) = e(\vec{C}_i, g_2) \cdot (e(\vec{X}_i, R')^{-1}) \Bigg\}. \end{aligned}$$

Note that the witness Y only appears in the first equation. This means we only need to prove a single product pairing equation (the first equation) inside the Groth-Sahai proof system and the remaining equations contain no variables and therefore contribute nothing to the proof size.

The instantiation of $\mathsf{PKEBC}_{\mathsf{SM}}$ using $\mathsf{mPKE}_{\mathbb{G}}$ and the optimized relation R^{GS} is described in Algorithm 2.

Concrete Instantiation. To obtain a simulation-sound NIZK for GS, we use the transformation of [37] instantiated with the (almost) tightly secure[10] structure-preserving signature scheme of [28] (based on the SXDH assumption). A conservative estimation[11] of the transformation's overhead results in proof size $|\pi_{\mathsf{GS}}|$[12] $= 104|\mathbb{G}_1| + 120|\mathbb{G}_2| + 2|\mathbb{Z}_p|$ and $|\mathtt{crs}| = 9|\mathbb{G}_1| + 15|\mathbb{G}_2|$.

For illustration, we give conservative estimates for ciphertext overhead and crs size when the scheme in Algorithm 2 is instantiated concretely using pairings over BLS12-381 and instantiating PRG with SHA3−512 and SKE with AES−GCM, targeting 128 bits of security. That is, $|\mathbb{G}_1| = 384$ bits, $|\mathbb{G}_2| = 768$ bits, $|\mathbb{Z}_p| = |\mathsf{SKE}| = 256$ bits, $|\mathcal{X}| = 512$ bits. We denote this instantiation by $\mathsf{PKEBC}_{\mathsf{BLS12-381}}^{\mathsf{SXDH}}$.

[10] The scheme has loss $6\log(q)$, where q is the number of signing queries.

[11] We assume full cost for each PPE, ignoring that linear PPEs admit shorter proofs in GS.

[12] π_{GS} is a Groth-Sahai NIWI proof over 21 PPEs and 27 variables (9 in $\mathbb{G}_1$ and 18 in $\mathbb{G}_2$) resulting from Groth's OR-encoding of the single PPE in GS and the verification equations of [28]'s signature scheme. Each variable costs 2 group elements and each equation 4 $\mathbb{G}_1$ and 4 $\mathbb{G}_2$ elements, see [32], Figure 3. The verification key and signature of the one-time signature scheme tots of [37] add 2 $\mathbb{G}_1$ and 2 $\mathbb{Z}_p$ elements.

For this instantiation we have $|\mathtt{crs}| \approx 1.87$ kB. We give ciphertext overhead estimations for different numbers of receivers in Figure 2, which also compares concrete instantiations of our ROM constructions.

	Ciphertext overhead in kB			
Construction	$n = 1$	$n = 10$	$n = 100$	$n = 1000$
$\mathsf{PKEBC}^{\mathsf{SXDH}}_{\mathsf{SM},\mathsf{BLS12-381}}$	15.25	16.98	34.25	207.06
$\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}_{\mathtt{secp256r1}}$	0.48	2.78	25.82	256.22
$\mathsf{PKEBC}^{\mathsf{FO[mPKE[HPS]]}}_{\mathtt{secp384r1}}$	0.54	2.27	19.55	192.35

Fig. 2: Comparison of the ciphertext overhead $|c| - |m|$ in kB for the concrete instantiations $\mathsf{PKEBC}^{\mathsf{SM},\mathsf{SXDH}}_{\mathsf{BLS12-381}}$, $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}_{\mathtt{secp256r1}}$, and $\mathsf{PKEBC}^{\mathsf{FO[mPKE[HPS]]}}_{\mathtt{secp384r1}}$ and different numbers of receivers n.

4.3 Construction from mPKE in the ROM

Our ROM-based PKEBC construction—$\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$ (Algorithm 3)—also follows the KEM-DEM paradigm [17,35]:

- the (Multi-Recipient) KEM component is constructed via the Fujisaki–Okamoto (FO) transform [25,24,36] applied to an adaptively secure mPKE scheme;
- the DEM component consists of a standard Symmetric Encryption scheme.

The main difference from the standard (single receiver) KEM-DEM approach is that we need to guarantee consistent decryption among all receivers. In both FO-based construction, this is guaranteed by having the decryption algorithm only output a valid $(\vec{v}, m)$ pair if, when it recomputes the KEM component it obtains the exact same (KEM) ciphertext—which includes all receivers' components. Once this is established—i.e. that the KEM component is in the range of an honest encryption/encapsulation for the given vector $\vec{v}$—then the Correctness of the KEM (essentially) guarantees that all receivers obtain the same.

$\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$'s building blocks are an adaptively secure Multi-Recipient Public Key Encryption Scheme $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$, a Symmetric Encryption Scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and hash functions H, T, V and K—which are modeled as random oracles in our security proofs.

Algorithm 3 $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ construction.

$\mathsf{Gen}(1^k)$: **return** $\mathsf{mPKE.Gen}(1^k)$

$\mathsf{Enc}(\vec{v} := (v_1, \ldots, v_n), m)$:
- $\tilde{m} \xleftarrow{\$} \mathcal{M}$
- $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}(\vec{v}, \tilde{m}))$
- $(\tau, \tau_{\vec{v}}, K) \leftarrow (\mathsf{T}(\tilde{m}), \mathsf{V}(\vec{v}, \tilde{m}), \mathsf{K}(\tilde{m}))$
- $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}, m))$
- **return** $c := (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$

$\mathsf{Dec}(\mathtt{sk}, c := (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$: $|c_{\mathsf{mPKE}}| = n$
- $\mathtt{ctr} \leftarrow \bot$
- **for** $i = 1, \ldots, n$ **:**
  - $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
  - $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
  - **if** $\mathsf{T}(\tilde{m}_i) = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
- **if** $\mathtt{ctr} = \bot$**: return** $\bot$
- $\tilde{m} \leftarrow \tilde{m}_{\mathtt{ctr}}, \quad K \leftarrow \mathsf{K}(\tilde{m})$
- $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
- **if** $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
- **if** $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}(\vec{v}, \tilde{m})$**: return** $\bot$
- **if** $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}(\vec{v}, \tilde{m}))$**:**
  - **return** $\bot$
- **return** $(\vec{v}, m)$

Algorithm 4 Algebraic $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{Adap}[\mathsf{mPKE}]]}$ instantiation.

$\mathsf{Gen}(1^k)$: $\vec{x} \xleftarrow{\$} \mathbb{Z}_p^2, \quad b \xleftarrow{\$} \{0,1\}$
- $\vec{X} = (g^{x_1}, g^{x_2})$
- **return** $(\mathtt{pk} := \vec{X}, \mathtt{sk} := (x_b, b))$

$\mathsf{Enc}(\vec{X} = ((X_1, X_2), \ldots, (X_{2n-1}, X_{2n})), m)$
- $h \xleftarrow{\$} \mathbb{G}, \quad r \leftarrow \mathsf{H}(\vec{X}, h)$ ▷ $r \in \mathbb{Z}_p$
- $\vec{X}' := (X_1, \ldots, X_{2n})$
- $(R, \vec{C}) \leftarrow (g, \vec{X}')^r \cdot (1, h, \ldots, h)$
- $(\tau, \tau_{\vec{X}}, K) \leftarrow (\mathsf{T}(h), \mathsf{V}(\vec{X}, h), \mathsf{K}(h))$
- $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{X}, m))$
- **return** $c := (R, \vec{C}, \tau, \tau_{\vec{X}}, c_{\mathsf{SKE}})$

$\mathsf{Dec}((x, b), c := (R, \vec{C}, \tau, \tau_{\vec{X}}, c_{\mathsf{SKE}}))$: $|\vec{C}| = 2n$
- $\mathtt{ctr} \leftarrow \bot$
- **for** $i = 1, \ldots, n$ **:**
  - $h_i \leftarrow C_{2i-1+b} \cdot R^{-x}$
  - **if** $\mathsf{T}(h_i) = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
- **if** $\mathtt{ctr} = \bot$**: return** $\bot$
- $h \leftarrow h_{\mathtt{ctr}}, \quad K \leftarrow \mathsf{K}(h)$
- $(\vec{X}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
- **if** $(\vec{X}, m) = \bot \vee |\vec{X}| \neq n$**: return** $\bot$
- **if** $\mathtt{pk} \notin \vec{X} \vee \tau_{\vec{X}} \neq \mathsf{V}(\vec{X}, h)$**: return** $\bot$
- $r \leftarrow \mathsf{H}(\vec{X}, h), \quad \vec{X}' := (X_1, \ldots, X_{2n})$
- **if** $(R, \vec{C}) \neq (g, \vec{X}')^r \cdot (1, h, \ldots, h)$**:**
  - **return** $\bot$
- **return** $(\vec{X}, m)$

Security Analysis. The informal theorem below gives an informal overview of the security properties of our $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ construction. The full theorem statements and their respective proofs are given in Appendix, Section D.

Theorem 2 (Informal). *We give tight reductions from breaking* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$*'s*

1. *Correctness (Theorem 14);*
2. *Adversarial Replay-Correctness (Theorem 15);*
3. *Replay-Correctness (Theorem 16);*
4. *Public-Key Collision Resistance (Theorem 17);*
5. *Self-Consistency (Theorem 18);*
6. *Consistency (Theorem 19);*
7. *Adversarial Robustness (Theorem 20);*
8. *and* $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}$ *security (Theorem 21)*

under adaptive corruptions to breaking 1. mPKE*'s* {IND, IK}-CPA$^{\text{adap}}$ *security (under adaptive corruptions) and/or correctness, and/or 2.* SKE*'s correctness and/or one-time* CCA *security (assuming its decryption algorithm is deterministic), where* H, T, K *and* V *are modeled as random oracles.*

The missing piece: constructing adaptively secure mPKE. We give two mPKE constructions that are proven secure under adaptive corruptions.

Lifting static to adaptive security. Our first construction is a simple transformation that lifts a statically secure mPKE scheme into an adaptively secure one. Our transformation, denoted Adap[mPKE], is defined in Algorithm 5. We note that there is only a factor of 2 efficiency loss relative to the original (statically secure) mPKE scheme: a public key of Adap[mPKE] consists of a pair of mPKE public keys, and ciphertexts are also double the original size (due to the doubling of public keys). On the other hand, we prove this scheme tightly secure with respect to the number of parties and the number of uses (i.e. encryption queries).

Algorithm 5 Transformation Adap[mPKE].

$\mathsf{Gen}(1^k)$: $b \xleftarrow{\$} \{0,1\}$, **for** $i \in \{0,1\}$: $(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{mPKE.Gen}(1^k)$
return $(\mathtt{pk} := (\mathtt{pk}_0, \mathtt{pk}_1), \mathtt{sk} := (b, \mathtt{sk}_b))$

$\mathsf{Enc}(\vec{v} := (v_1, \ldots, v_n), m)$: $\vec{u} := (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$
return $c \leftarrow \mathsf{mPKE.Enc}(\vec{u}, m; \mathtt{r}_\$)$

$\mathsf{Extract}(c, i)$: $|c| = 2n$, $i \in [n]$
return $(\mathsf{mPKE.Extract}(c, 2i-1), \mathsf{mPKE.Extract}(c, 2i))$

$\mathsf{Dec}(\mathtt{sk} := (b, \mathtt{sk}_b), c := (c_0, c_1))$: **return** $m \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}_b, c_b)$

The following (informal) theorem captures the security guarantees of Adap[mPKE]. (We *only* rely on these three properties to prove the security of PKEBC$^{\mathsf{FO[mPKE]}}$.)

Theorem 3 (Informal). *If* mPKE *is perfectly correct,* γ*-spread,* Extend*-extendable for an algorithm* Extend*, and tightly* {IND, IK}-CPA *secure, then* Adap[mPKE] *is 1. correct; 2.* γ*-spread; and 3. tightly* {IND, IK}-CPA$^{\text{adap}}$ *secure under adaptive corruptions (Theorem 22).*

Proof. Adap[mPKE]'s Correctness and γ-spreadness follow immediately from mPKE's Correctness and γ-spreadness.

The {IND, IK}-CPA$^{\text{adap}}$ security under adaptive corruptions crucially relies on the ability to extend ciphertexts. Concretely, one reduces to mPKE's {IND, IK}-CPA security (under static corruptions) by using the encryption oracle to obtain half of the mPKE ciphertext components—namely, for each party, the ones that correspond the "forgotten" secret key $\mathtt{sk}_{1-b}$—and then uses the ciphertext extension algorithm to obtain the entire ciphertext. This reduction is applied twice since there are two ciphertext components per party. (Refer to Theorem 22 for a detailed proof.)

Adaptively secure HPS*-based* mPKE. Our second scheme (Algorithm 6) is a direct mPKE construction based on Hash Proof Systems.

Algorithm 6 mPKE[HPS] construction.

$\mathsf{Gen}(\mathtt{pp})$: $\mathtt{sk} \leftarrow \mathsf{HPS.SK\text{-}Sample}(\mathtt{pp}_{\mathsf{HPS}})$
  return $(\mathtt{pk} \coloneqq \mathsf{HPS.PK\text{-}Of}(\mathtt{sk}), \mathtt{sk})$

$\mathsf{Extract}(i, c \coloneqq (x, \vec{d}))$: **return** $c_i \coloneqq (x, d_i)$

$\mathsf{Dec}(\mathtt{sk}, c_i)$: **return** $d_i \oplus \mathsf{HPS.Priv}(\mathtt{sk}, x)$

$\mathsf{Enc}(\vec{v} \coloneqq (v_1, \ldots, v_n), m)$:
  $(x, w) \leftarrow \mathsf{HPS.Samp}^{\in}(\mathtt{pp})$
  for $i \in [n]$ **:** $d_i \leftarrow m \oplus \mathsf{HPS.Pub}(v_i, x, w)$
  return $c \coloneqq (x, \vec{d})$

The following (informal) theorem captures the security guarantees of mPKE[HPS].

Theorem 4 (Informal). *If* HPS *is perfectly correct,* γ*-spread, pairwise universal and its language's subset membership problem is hard, then* mPKE[HPS] *is 1. correct; 2.* γ*-spread; and 3.* $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CPA}^{\mathsf{adap}}$ *secure under adaptive corruptions (Theorem 23).*

Proof. mPKE[HPS]'s Correctness, Public-Key Collision Resistance and γ-spreadness follow immediately from HPS's Perfect Correctness (Definition 26), $\gamma_{\mathtt{pk}}$-Public-Key Min-Entropy (Definition 27) and γ_x-Statement-Min-Entropy (Definition 30), respectively.

Proving mPKE[HPS]'s $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CPA}^{\mathsf{adap}}$ security (under adaptive corruptions) crucially relies on the fact that Pairwise Universal is a statistical notion, which allows us to handle secret key corruption queries without guessing. The security proof is tight in the number of corruption queries, but not in the number of challenge queries made by the adversary. (Refer to Theorem 23 for a detailed proof.)

Efficiency Analysis.

Generic Efficiency: Let $\mathcal{T}$ and $\mathcal{V}$ be T's and V's codomains, respectively. After applying Adap[mPKE], public keys consist of two mPKE keys and the ciphertext overhead is $2|c_{\mathsf{mPKE}}| + |\mathcal{T}| + |\mathcal{V}| + (2n)|\mathtt{pk}_{\mathsf{mPKE}}| + |\mathsf{SKE}|$ where $|\mathtt{pk}_{\mathsf{mPKE}}|$ and $|c_{\mathsf{mPKE}}|$ are mPKE's public key and ciphertext sizes and $|\mathsf{SKE}|$ is the overhead introduced by SKE. Each execution of Enc makes 4 RO-calls and executes both mPKE.Enc and SKE.Enc once. Dec makes $\mathcal{O}(n)$ mPKE.Dec and RO-calls and *one* SKE.Dec-call.

Algebraic Instantiations: Instantiating mPKE with the ElGamal-style $\mathsf{mPKE}_{\mathbb{G}}$ [44] over a cyclic group $\mathbb{G}$ as described in Algorithm 4 results in $|\mathtt{pk}| = 2|\mathbb{G}|$ and $|c| - |m| = (4n + 1)|\mathbb{G}| + |\mathcal{T}| + |\mathcal{V}| + |\mathsf{SKE}|$.

Alternatively, we can instantiate mPKE with our construction $\mathsf{mPKE}[\mathsf{HPS}]$ using a group-based hash-proof system HPS as described in algorithm 7. This results in $|\mathtt{pk}| = |\mathbb{G}|$ and $|c| - |m| = (n+2)|\mathbb{G}| + n|\mathcal{G}| + |\mathcal{T}| + |\mathcal{V}| + |\mathsf{SKE}|$, where $|\mathcal{G}|$ is G's codomain.

Algorithm 7 Alg. instantiation $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}[\mathsf{HPS}]]}$ from HPS-based mPKE.

Gen: $\vec{x} \leftarrow \mathbb{Z}_p^2$, $X = g_1^{x_1} g_2^{x_2}$; **return** $(\mathtt{pk} \coloneqq X, \mathtt{sk} \coloneqq \vec{x})$

$\mathsf{Extract}(\vec{X}, \mathtt{r}_\$)$:
$r \leftarrow \mathsf{H}(\vec{X}, \mathtt{r}_\$)$ ▷ $r \in \mathbb{Z}_p$
$(C_1, C_2) \leftarrow (g_1^r, g_2^r)$, $\vec{K} \leftarrow \vec{X}^r$
for $i \in |\vec{X}|$**:** $d_i = \mathsf{G}(K_i) \oplus \mathtt{r}_\$$
return $\left(\vec{C} \coloneqq (C_1, C_2), \vec{d}\right)$

$\mathsf{Enc}(\vec{X} \coloneqq (X_1, \ldots, X_n), m)$:
$\left(\vec{C}, \vec{d}\right) \leftarrow \mathsf{Encaps}(\vec{X}; \mathtt{r}_\$ \leftarrow \{0,1\}^k)$
$(\tau, \tau_{\vec{X}}, K) \leftarrow (\mathsf{T}(\mathtt{r}_\$), \mathsf{V}(\vec{X}, \mathtt{r}_\$), \mathsf{K}(\mathtt{r}_\$))$
$c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}, m))$
return $c \coloneqq (\vec{C}, \vec{d}, \tau, \tau_{\vec{X}}, c_{\mathsf{SKE}})$

$\mathsf{Dec}(\mathtt{sk}, c \coloneqq (\vec{C}, \vec{d}, \tau, \tau_{\vec{X}}, c_{\mathsf{SKE}}))$: $\vec{d} \coloneqq (d_1, \ldots, d_n)$
$\pi_{\mathtt{sk}} = \mathsf{G}(C_1^{x_1}.C_2^{x_2})$, $\mathtt{ctr} \leftarrow \bot$
for $i = 1, \ldots, n$ **:**
$\mathtt{r}_\$^i \leftarrow d_i \oplus \pi_{\mathtt{sk}}$
if $\mathsf{T}(\mathtt{r}_\$^i) = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
if $\mathtt{ctr} = \bot$**: return** $\bot$
$\mathtt{r}_\$ \leftarrow \mathtt{r}_\$^{\mathtt{ctr}}$, $K \leftarrow \mathsf{K}(\mathtt{r}_\$)$
$(\vec{X}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
if $(\vec{X}, m) = \bot \vee |\vec{X}| \neq n$**: return** $\bot$
if $(g_1^{x_1} g_2^{x_2}) \notin \vec{X} \vee \tau_{\vec{X}} \neq \mathsf{V}(\vec{X}, \mathtt{r}_\$)$**: return** $\bot$
if $\left(\vec{C}, \vec{d}\right) \neq \mathsf{Encaps}(\vec{X}; \mathtt{r}_\$)$**: return** $\bot$
return $(\vec{X}, m)$

Concrete Instantiations: We describe concrete instantiations for both algebraic instantiations, aimed at a security level of 128 bits.

For Algorithm 4, instantiating $\mathsf{mPKE}_{\mathbb{G}}$ over NIST P-256 (`secp256r1`), all random oracles with `SHA3-512` and SKE with `AES-GCM` results in $|\mathbb{G}| = 512$ bits, $|\mathcal{T}| = |\mathcal{V}| = 512$ bits, and $|\mathsf{SKE}| = 256$ bits. We denote this instantiation by $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}_{\mathtt{secp256r1}}$.

For Algorithm 7, as $\mathsf{mPKE}[\mathsf{HPS}]$ has a security loss of linear in the number of encryption queries, we instantiate all building blocks with at least 192 bits of security, such that allowing up to 2^{60} encryption queries still retains more than 128 bits of security. Instantiating $\mathsf{mPKE}_{\mathbb{G}}$ over NIST P-384 (`secp384r1`), all random oracles with `SHA3-512` and SKE with `AES-GCM` results in $|\mathbb{G}| = 1024$ bits, $|\mathcal{T}| = |\mathcal{G}| = 512$ bits, and $|\mathsf{SKE}| = 256$ bits. We denote this instantiation by $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}[\mathsf{HPS}]]}_{\mathtt{secp384r1}}$. Figure 2 gives a comparison of ciphertext sizes for these instantiations and the previous standard model instantiation.

5 (Anonymous) Signcryption for Broadcast

An (anonymous) Signcryption for Broadcast (SCBC) scheme is a 5-tuple of PPTs $\mathsf{SCBC} = (\mathsf{Setup}, \mathsf{Gen_S}, \mathsf{Gen_R}, \mathsf{Enc}, \mathsf{Dec})$ where:

$\mathsf{Setup}(1^k)$: outputs public parameters $\mathtt{pp}$;
$\mathsf{Gen_S}(\mathtt{pp})$: outputs a sender key-pair $(\mathtt{spk}, \mathtt{ssk})$;
$\mathsf{Gen_R}(\mathtt{pp})$: outputs a receiver key-pair $(\mathtt{rpk}, \mathtt{rsk})$;

$\mathsf{Enc}_{\mathsf{pp}}(\mathtt{ssk}, \vec{v}, m)$: outputs a ciphertext c, where $\mathtt{ssk}$ is a sender's secret key, $\vec{v}$ a vector of receiver public keys and m a message;

$\mathsf{Dec}_{\mathsf{pp}}(\mathtt{rsk}, c)$: outputs a triple $(\mathtt{spk}, \vec{v}, m)$—where $\mathtt{spk}$ a sender's public key, $\vec{v}$ a vector of receiver public keys and m a message—or $\perp$ if c's decryption with $\mathtt{rsk}$ fails.

As for PKEBC, we require that the public key vector $\vec{v}$ input to Enc contains no duplicates, i.e. $|\mathrm{Set}(\vec{v})| = |\vec{v}|$.

One can also define a two phase decryption variant for better efficiency: in the first phase, decryption would output a vector of public key identifiers—which can be more concise than normal public keys—and in the second phase, the full public keys would be input—i.e. the ones matching the identifiers obtained in the first decryption phase. With the extra information decryption could then proceed as if the full public keys of each party had been included in the PKEBC ciphertext. Although this could lead to efficiency gains, defining this more efficient variant would require additional formalisms—from having to re-define existing PKEBC game-based notions to extending existing composable treatments—that are mostly straightforward, yet cumbersome, and which ultimately are orthogonal to the main contributions of the paper.

A non-anonymous variant can be defined by having decryption take as input the public key of the sender and the vector of receivers' public keys.

5.1 (Game-Based) Security Notions

SCBC schemes are similar to MDRS-PKE but do not provide deniability guarantees. The notions below are similar to the MDRS-PKE and PKEBC notions from [51,10,47].

Consider a scheme $\mathsf{SCBC} = (\mathsf{Setup}, \mathsf{Gen_S}, \mathsf{Gen_R}, \mathsf{Enc}, \mathsf{Dec})$; the notions below have an implicitly defined security parameter k and provide adversaries with access to the following oracles:

$\mathcal{O}_{PP}$: on the first query compute $\mathtt{pp} \leftarrow \mathsf{Setup}(1^k)$; output $\mathtt{pp}$;
$\mathcal{O}_{SK}(A_i)$: For first A_i query, store $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathsf{Gen_S}(\mathtt{pp})$; output $(\mathtt{spk}_i, \mathtt{ssk}_i)$;
$\mathcal{O}_{SPK}(A_i)$: $(\mathtt{spk}_i, \cdot) \leftarrow \mathcal{O}_{SK}(A_i)$; output $\mathtt{spk}_i$;
$\mathcal{O}_{RK}(B_j)$: Analogous to $\mathcal{O}_{SK}$;
$\mathcal{O}_{RPK}(B_j)$: Analogous to $\mathcal{O}_{SPK}$;
$\mathcal{O}_E(A_i, \vec{V}, m)$: 1. $(\cdot, \mathtt{ssk}_i) \leftarrow \mathcal{O}_{SK}(A_i)$;
2. $\vec{v} := (\mathcal{O}_{RPK}(V_1), \ldots, \mathcal{O}_{RPK}(V_{|\vec{V}|}))$;
3. output $c \leftarrow \mathsf{Enc}(\mathtt{ssk}_i, \vec{v}, m)$;
$\mathcal{O}_D(B_j, c)$: 1. $(\cdot, \mathtt{rsk}_j) \leftarrow \mathcal{O}_{RK}(B_j)$; 2. $(\mathtt{spk}_i, \vec{v}, m) \leftarrow \mathsf{Dec}(\mathtt{rsk}_j, c)$; 3. if, for each party A_i previously input to either $\mathcal{O}_{SK}$, $\mathcal{O}_{SPK}$ or $\mathcal{O}_E$, $\mathtt{spk}_i \neq \mathcal{O}_{SPK}(A_i)$, output $\perp$; 4. if, for some $l \in \{1, \ldots, |\vec{V}|\}$, there is no party B_j that was previously input to either $\mathcal{O}_{RK}$, $\mathcal{O}_{RPK}$, $\mathcal{O}_E$ or $\mathcal{O}_D$ such that $v_l = \mathcal{O}_{RPK}(V_l)$, output $\perp$; 5. output $(\mathtt{spk}_i, \vec{v}, m)$;
$\mathcal{O}_{Publish}(c)$: no output.

Definition 11 (Correctness: $\neg$Corr). *A query $\mathcal{O}_E(A_i, \vec{V}, m)$ outputs c, and there is no prior $\mathcal{O}_{SK}(A_i)$ query; a later query $\mathcal{O}_D(B_j \in \vec{V}, c)$ outputs triple $(\mathtt{spk}'_i, \vec{v}', m') \neq (\mathtt{spk}_i, \vec{v}, m)$—where $\mathtt{spk}_i$, $\vec{v}$ are A_i's and vector $\vec{V}$'s associated public keys—and there is no prior query $\mathcal{O}_{RK}(B_j)$.*

Definition 12 (Replay-Correctness: $\neg$RCorr). *Two distinct $\mathcal{O}_E$ queries output the same ciphertext c.*

Definition 13 (Adversarial Replay-Correctness: $\neg$aRCorr). *There is a query $\mathcal{O}_{Publish}(c)$; a later $\mathcal{O}_E$ query also outputs c.*

Definition 14 (Public-Key Collision Resistance). $\mathsf{SCBC} = (\mathsf{Setup}, \mathsf{Gen_S}, \mathsf{Gen_R}, \mathsf{Enc}, \mathsf{Dec})$ *is (n, ℓ)-Party ε-Public-Key Collision Resistant if*

$$\Pr\left[\begin{array}{c|c} |\{\mathtt{spk}_1, \ldots, \mathtt{spk}_n\}| < n & \mathtt{pp} \leftarrow \mathsf{Setup}(1^k), \\ \vee & (\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathsf{Gen_S}(\mathtt{pp}), i \in [n] \\ |\{\mathtt{rpk}_1, \ldots, \mathtt{rpk}_\ell\}| < \ell & (\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow \mathsf{Gen_S}(\mathtt{pp}), j \in [\ell] \end{array}\right] \leq \varepsilon.$$

Definition 15 (Self-Consistency: $\neg$Self-Cons). *A query $\mathcal{O}_D(B_i, c)$ outputs some triple $(\mathtt{spk}, \vec{v}, m) \neq \perp$ and $\mathtt{pk}_i \notin \vec{v}$.*

Definition 16 (Consistency: $\neg$Cons). *A query $\mathcal{O}_D(B_j, c)$ outputs some triple $(\mathtt{spk}, \vec{v}, m) \neq \perp$; a later query $\mathcal{O}_D(B_i, c)$, where $\mathtt{pk}_i \in \vec{v}$ outputs $\perp$ or a different triple $(\mathtt{spk}', \vec{v}', m') \neq (\mathtt{spk}, \vec{v}, m)$.*

Definition 17 (Adversarial Robustness: $\neg$aRob). *A query $\mathcal{O}_D(B_j, c)$ outputs a triple $(\mathtt{spk}, \vec{v}, m) \neq \perp$; a later query $\mathcal{O}_D(B_i, c)$ with $\mathtt{pk}_i \notin \vec{v}$ does not output $\perp$.*

Definition 18 (Replay Unforgeability: $\neg$RUnforg). *A query $\mathcal{O}_D(B_j, c)$ outputs $(\mathtt{spk}_i, \vec{v}, m) \neq \perp$ such that $\mathtt{spk}_i$ and $\vec{v}$ are some sender A_i's and receiver-vector $\vec{V}$'s respective public keys, and no prior query $\mathcal{O}_E(A_i, \vec{V}, m)$ output c; there is no prior query $\mathcal{O}_{SK}(A_i)$; there is no prior query $\mathcal{O}_{RK}(B_j \in \vec{V})$.*

The $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ notion provides adversaries with access to the oracles from before, but where $\mathcal{O}_E$ and $\mathcal{O}_D$ are now as follows:

$\mathcal{O}_E^{\mathbf{b}}\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$: output $c \leftarrow \mathsf{Enc}_{\mathtt{pp}}(\mathtt{ssk}_{i,\mathbf{b}}, \vec{v}_{\mathbf{b}}, m_{\mathbf{b}})$;
$\mathcal{O}_D(B_j, c)$: if c was output by $\mathcal{O}_E$, output $\mathtt{test}$; else, output $(\mathtt{spk}, \vec{v}, m) \leftarrow \mathsf{Dec}_{\mathtt{pp},\mathtt{rsk}_j}(c)$.

Definition 19 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$). $\mathcal{O}^{\mathbf{b}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}} := (\mathcal{O}_{PP}, \mathcal{O}_{SK}, \mathcal{O}_{RK}, \mathcal{O}_{SPK}, \mathcal{O}_{RPK}, \mathcal{O}_E^{\mathbf{b}}, \mathcal{O}_D)$.
Disallowed: *1. queries $\mathcal{O}_E((\cdot, \vec{V}_0, m_0), (\cdot, \vec{V}_1, m_1))$ with $(|\vec{V}_0|, |m_0|) \neq (|\vec{V}_1|, |m_1|)$; 2. for query $\mathcal{O}_E((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1))$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$: there is a query $\mathcal{O}_{RK}(B_j \in \vec{V}_0 \cup \vec{V}_1)$, or there is a query $\mathcal{O}_{SK}(A_i \in \{A_{i,0}, A_{i,1}\})$.*

5.2 Application Semantics of the Game-Based Security Model

In [47], Liu-Zhang et al. show how to construct a messaging application providing consistency guarantees from repository-based communication channels. In particular, they show that if the underlying repositories provide additional security guarantees, such as authenticity, confidentiality and anonymity, then their messenger inherits each of these [47, Figure 1]. In Appendix, Section I, we prove that our SCBC game-based model suffices to construct [47]'s repositories with authenticity, confidentiality and anonymity guarantees.

5.3 Construction

Our SCBC construction—defined in Algorithm 8—is analogous to the MDRS-PKE construction from [51,52], but where the MDVS scheme is replaced with a DSS. Its building blocks are a PKEBC, an EUF-CMA secure DSS and an sEUF-CMA secure one; both DSS's signature verification is assumed to be deterministic.

Algorithm 8 Construction of $\mathsf{SCBC} = (\mathsf{Setup}, \mathsf{Gen_S}, \mathsf{Gen_R}, \mathsf{Enc}, \mathsf{Dec})$ from $\mathsf{PKEBC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathsf{EUF} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ and $\mathsf{sEUF} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

$\mathsf{Setup}(1^k)$: **return** $\mathtt{pp} := (1^k, \mathtt{pp}_{\mathsf{PKEBC}} \xleftarrow{\$} \mathsf{PKEBC.Setup}(1^k))$

$\mathsf{Gen_S}(\mathtt{pp})$
$(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathsf{EUF.Gen}(1^k)$
return $(\mathtt{spk} := \mathtt{vk}, \mathtt{ssk} := (\mathtt{spk}, \mathtt{sk}))$

$\mathsf{Gen_R}(\mathtt{pp})$
$(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{PKEBC.Gen}(\mathtt{pp}_{\mathsf{PKEBC}})$
return $(\mathtt{rpk} := \mathtt{pk}, \mathtt{rsk} := (\mathtt{rpk}, \mathtt{sk}))$

$\mathsf{Enc}(\mathtt{ssk}, \vec{v}, m)$
$(\mathtt{vk}_{\mathsf{sEUF}}, \mathtt{sk}_{\mathsf{sEUF}}) \leftarrow \mathsf{sEUF.Gen}(1^k)$
$\sigma_{\mathsf{EUF}} \leftarrow \mathsf{EUF.Sign}(\mathtt{sk}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$
$c \leftarrow \mathsf{PKEBC.Enc}(\vec{v}, (\mathtt{spk}, m, \sigma_{\mathsf{EUF}}))$
$\sigma_{\mathsf{sEUF}} \leftarrow \mathsf{sEUF.Sign}(\mathtt{sk}_{\mathsf{sEUF}}, c)$
return $(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$

$\mathsf{Dec}(\mathtt{rsk}, c_{\mathsf{SCBC}} := (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$
if $\mathsf{sEUF.Vrfy}(\mathtt{vk}_{\mathsf{sEUF}}, c, \sigma_{\mathsf{sEUF}}) = 1$ **:**
$(\vec{v}, (\mathtt{spk}, m, \sigma_{\mathsf{EUF}})) \leftarrow \mathsf{PKEBC.Dec}(\mathtt{sk}, c)$
if $((\vec{v}, (\mathtt{spk}, m, \sigma_{\mathsf{EUF}})) \neq \perp) \wedge (\mathsf{EUF.Vrfy}(\mathtt{vk}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}), \sigma_{\mathsf{EUF}}) = 1)$ **:**
return $(\mathtt{spk}, \vec{v}, m)$
return $\perp$

Security Analysis

Theorem 5 (Informal). *If 1.* PKEBC *is tightly Correct, Replay-Correct, Adversarially Replay-Correct, Public-Key Collision Resistant, Adversarially Robust, Self-Consistent, Consistent, Adversarially Robust and* $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ *secure (all under adaptive corruptions), and 2.* EUF *is tightly correct,* EUF-CMA *secure and has deterministic signature verification, all under adaptive corruptions, and 3.* sEUF *is tightly Correct, one-time* sEUF-CMA *secure and also has deterministic signature verification, then* SCBC *tightly satisfies:*

1. *Correctness (Theorem 24);*

2. *Replay-Correctness (Theorem 25);*
3. *Adversarial Replay-Correctness (Theorem 26);*
4. *Public-Key Collision Resistance (Corollary 1);*
5. *Self-Consistency (Theorem 27);*
6. *Consistency (Theorem 28);*
7. *Adversarial Robustness (Theorem 29);*
8. *Replay Unforgeability (Theorem 30); and*
9. $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}$ *security (Theorem 31)*

all under adaptive corruptions.

The formal theorem statements and their proofs are in the Appendix, Section H.

5.4 Concrete Instantiations

SCBC can be instantiated efficiently in the ROM using one of our PKEBC constructions in the ROM and the DSS of Diemert et. al. [21], used both for EUF and sEUF. Concretely, using the DDH-based instantiation over NIST-curve P-256 of [21]'s scheme (and instantiating random oracles with $\mathtt{SHA3{-}512}$) together with either $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}_{\mathtt{secp256r1}}$ or $\mathsf{PKEBC}^{\mathsf{FO[mPKE[HPS]]}}_{\mathtt{secp384r1}}$ (see section 4.3) results in an SCBC ciphertext overhead that is 0.7 kB larger compared to the sizes listed in fig. 2.

Acknowledgments. Eike Kiltz was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911). Roman Langrehr was supported by SNSF Postdoc.Mobility grant no. 235425. Guilherme Rito was supported by the European Union (ERC AdG REWORC - 101054911).

References

1. Alwen, J., Coretti, S., Jost, D., Mularczyk, M.: Continuous group key agreement with active security. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 261–290. Springer, Cham (Nov 2020). doi: 10.1007/978-3-030-64378-2_10 (Cited on page 2.)
2. Alwen, J., Jost, D., Mularczyk, M.: On the insider security of MLS. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 34–68. Springer, Cham (Aug 2022). doi: 10.1007/978-3-031-15979-4_2 (Cited on page 2.)
3. Alwen, J., Mularczyk, M., Tselekounis, Y.: Fork-resilient continuous group key agreement. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 396–429. Springer, Cham (Aug 2023). doi: 10.1007/978-3-031-38551-3_13 (Cited on page 2.)
4. Badertscher, C., Gazi, P., Kiayias, A., Russell, A., Zikas, V.: Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 913–930. ACM Press (Oct 2018). doi: 10.1145/3243734.3243848 (Cited on page 33.)
5. Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., Cohn-Gordon, K.: The Messaging Layer Security (MLS) Protocol. RFC 9420 (Jul 2023). doi: 10.17487/RFC9420, https://www.rfc-editor.org/info/rfc9420 (Cited on page 2.)
6. Bhargavan, K., Barnes, R., Rescorla, E.: TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS). Research report, Inria Paris (May 2018), https://inria.hal.science/hal-02425247 (Cited on page 2.)
7. 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, Berlin, Heidelberg (Aug 2005). doi: 10.1007/11535218_16 (Cited on page 6.)
8. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 480–499. Springer, Berlin, Heidelberg (Aug 2014). doi: 10.1007/978-3-662-44371-2_27 (Cited on page 7.)
9. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). doi: 10.1109/SFCS.2001.959888 (Cited on page 84.)
10. Chakraborty, S., Hofheinz, D., Maurer, U., Rito, G.: Deniable authentication when signing keys leak. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 69–100. Springer, Cham (Apr 2023). doi: 10.1007/978-3-031-30620-4_3 (Cited on pages 4, 5, 7, 8, 9, 13, 14, 15, 24, 67, 68, and 97.)
11. Champion, J., Hsieh, Y.C., Wu, D.J.: Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part III. LNCS, vol. 16002, pp. 3–34. Springer, Cham (Aug 2025). doi: 10.1007/978-3-032-01881-6_1 (Cited on page 7.)
12. Champion, J., Wu, D.J.: Distributed broadcast encryption from lattices. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024, Part III. LNCS, vol. 15366, pp. 156–189. Springer, Cham (Dec 2024). doi: 10.1007/978-3-031-78020-2_6 (Cited on page 7.)
13. Chase, M., Maller, M., Meiklejohn, S.: Déjà Q all over again: Tighter and broader reductions of q-type assumptions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part II. LNCS, vol. 10032, pp. 655–681. Springer, Berlin, Heidelberg (Dec 2016). doi: 10.1007/978-3-662-53890-6_22 (Cited on page 6.)

14. Coretti, S., Kiayias, A., Moore, C., Russell, A.: The generals' scuttlebutt: Byzantine-resilient gossip protocols. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 595–608. ACM Press (Nov 2022). doi: 10.1145/3548606.3560638 (Cited on page 33.)
15. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Berlin, Heidelberg (Aug 1994). doi: 10.1007/3-540-48658-5_19 (Cited on page 6.)
16. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Berlin, Heidelberg (Aug 1998). doi: 10.1007/BFb0055717 (Cited on page 35.)
17. 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) (Cited on page 19.)
18. Damgård, I., Haagh, H., Mercer, R., Nitulescu, A., Orlandi, C., Yakoubov, S.: Stronger security and constructions of multi-designated verifier signatures. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 229–260. Springer, Cham (Nov 2020). doi: 10.1007/978-3-030-64378-2_9 (Cited on page 4.)
19. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 66–98. Springer, Cham (Apr / May 2018). doi: 10.1007/978-3-319-78375-8_3 (Cited on page 33.)
20. Devigne, J., Duguey, C., Fouque, P.A.: MLS group messaging: How zero-knowledge can secure updates. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021, Part II. LNCS, vol. 12973, pp. 587–607. Springer, Cham (Oct 2021). doi: 10.1007/978-3-030-88428-4_29 (Cited on page 2.)
21. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 1–31. Springer, Cham (May 2021). doi: 10.1007/978-3-030-75248-4_1 (Cited on page 27.)
22. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). doi: 10.1007/3-540-47721-7_12 (Cited on page 6.)
23. Freitag, C., Waters, B., Wu, D.J.: How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 498–531. Springer, Cham (Aug 2023). doi: 10.1007/978-3-031-38551-3_16 (Cited on page 7.)
24. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Berlin, Heidelberg (Aug 1999). doi: 10.1007/3-540-48405-1_34 (Cited on pages 9 and 19.)
25. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (Jan 2013). doi: 10.1007/s00145-011-9114-1 (Cited on pages 4 and 19.)
26. Garg, R., Lu, G., Waters, B., Wu, D.J.: Realizing flexible broadcast encryption: How to broadcast to a public-key directory. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 1093–1107. ACM Press (Nov 2023). doi: 10.1145/3576915.3623168 (Cited on page 6.)

27. Garg, S., Kolonelos, D., Policharla, G.V., Wang, M.: Threshold encryption with silent setup. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 352–386. Springer, Cham (Aug 2024). doi: 10.1007/978-3-031-68394-7_12 (Cited on page 7.)
28. Gay, R., Hofheinz, D., Kohl, L., Pan, J.: More efficient (almost) tightly secure structure-preserving signatures. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 230–258. Springer, Cham (Apr / May 2018). doi: 10.1007/978-3-319-78375-8_8 (Cited on page 18.)
29. Gay, R., Kowalczyk, L., Wee, H.: Tight adaptively secure broadcast encryption with short ciphertexts and keys. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 123–139. Springer, Cham (Sep 2018). doi: 10.1007/978-3-319-98113-0_7 (Cited on page 6.)
30. 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, Berlin, Heidelberg (Apr 2009). doi: 10.1007/978-3-642-01001-9_10 (Cited on page 6.)
31. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454 (2017), https://eprint.iacr.org/2017/454 (Cited on page 33.)
32. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155 (2007), https://eprint.iacr.org/2007/155 (Cited on page 18.)
33. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Berlin, Heidelberg (Apr 2008). doi: 10.1007/978-3-540-78967-3_24 (Cited on pages 5, 9, and 17.)
34. Hashimoto, K., Katsumata, S., Postlethwaite, E., Prest, T., Westerbaan, B.: A concrete treatment of efficient continuous group key agreement via multi-recipient PKEs. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 1441–1462. ACM Press (Nov 2021). doi: 10.1145/3460120.3484817 (Cited on page 33.)
35. Herranz, J., Hofheinz, D., Kiltz, E.: Some (in)sufficient conditions for secure hybrid encryption. Inf. Comput. **208**(11), 1243–1257 (2010). doi: 10.1016/J.IC.2010.07.002, https://doi.org/10.1016/j.ic.2010.07.002 (Cited on page 19.)
36. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Cham (Nov 2017). doi: 10.1007/978-3-319-70500-2_12 (Cited on pages 9, 19, and 66.)
37. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Berlin, Heidelberg (Aug 2012). doi: 10.1007/978-3-642-32009-5_35 (Cited on pages 17 and 18.)
38. Hohenberger, S., Lu, G., Waters, B., Wu, D.J.: Registered attribute-based encryption. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 511–542. Springer, Cham (Apr 2023). doi: 10.1007/978-3-031-30620-4_17 (Cited on page 7.)
39. Janneck, J., Nishaburi, A., Rito, G.: Modeling emails: On the deniability of bccs. Cryptology ePrint Archive, Report 2025/1750 (2025), https://eprint.iacr.org/2025/1750 (Cited on pages 12 and 13.)
40. Jost, D., Maurer, U.: Overcoming impossibility results in composable security using interval-wise guarantees. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020,

Part I. LNCS, vol. 12170, pp. 33–62. Springer, Cham (Aug 2020). doi: 10.1007/978-3-030-56784-2_2 (Cited on page 85.)
41. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 357–388. Springer, Cham (Aug 2017). doi: 10.1007/978-3-319-63688-7_12 (Cited on page 33.)
42. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 590–609. Springer, Berlin, Heidelberg (Apr 2009). doi: 10.1007/978-3-642-01001-9_34 (Cited on page 35.)
43. Kolonelos, D., Malavolta, G., Wee, H.: Distributed broadcast encryption from bilinear groups. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part V. LNCS, vol. 14442, pp. 407–441. Springer, Singapore (Dec 2023). doi: 10.1007/978-981-99-8733-7_13 (Cited on page 7.)
44. Kurosawa, K.: Multi-recipient public-key encryption with shortened ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Berlin, Heidelberg (Feb 2002). doi: 10.1007/3-540-45664-3_4 (Cited on pages 9, 12, 17, 22, and 37.)
45. Liu-Zhang, C.D., Matt, C., Maurer, U., Rito, G., Thomsen, S.E.: Practical provably secure flooding for blockchains. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 774–805. Springer, Cham (Dec 2022). doi: 10.1007/978-3-031-22963-3_26 (Cited on page 33.)
46. Liu-Zhang, C.D., Maurer, U.: Synchronous constructive cryptography. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 439–472. Springer, Cham (Nov 2020). doi: 10.1007/978-3-030-64378-2_16 (Cited on page 85.)
47. Liu-Zhang, C.D., Portmann, C., Rito, G.: Stateful communication with malicious parties. Cryptology ePrint Archive, Report 2024/1593 (2024), https://eprint.iacr.org/2024/1593 (Cited on pages 2, 3, 6, 13, 15, 24, 26, 80, 84, 86, 87, 88, 89, and 90.)
48. Liu-Zhang, C.D., Portmann, C., Rito, G.: Simpler and stronger models for deniable authentication. Cryptology ePrint Archive, Report 2025/204 (2025), https://eprint.iacr.org/2025/204 (Cited on page 6.)
49. Maurer, U.: Constructive cryptography—a new paradigm for security definitions and proofs. In: Proceedings of Theory of Security and Applications, TOSCA 2011. Lecture Notes in Computer Science, vol. 6993, pp. 33–56. Springer (2012). doi: 10.1007/978-3-642-27375-9_3 (Cited on page 6.)
50. Maurer, U., Portmann, C., Rito, G.: Giving an adversary guarantees (or: How to model designated verifier signatures in a composable framework). In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 189–219. Springer, Cham (Dec 2021). doi: 10.1007/978-3-030-92078-4_7 (Cited on pages 86, 87, and 88.)
51. Maurer, U., Portmann, C., Rito, G.: Multi-designated receiver signed public key encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 644–673. Springer, Cham (May / Jun 2022). doi: 10.1007/978-3-031-07085-3_22 (Cited on pages 3, 4, 7, 12, 15, 24, 26, 80, and 97.)
52. Maurer, U., Portmann, C., Rito, G.: Multi-designated receiver signed public key encryption. Cryptology ePrint Archive, Report 2022/256 (2022), https://eprint.iacr.org/2022/256 (Cited on pages 26 and 80.)
53. Maurer, U., Renner, R.: Abstract cryptography. In: Chazelle, B. (ed.) ICS 2011. pp. 1–21. Tsinghua University Press (Jan 2011) (Cited on page 6.)

54. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Berlin, Heidelberg (Apr / May 2002). doi: 10.1007/3-540-46035-7_8 (Cited on page 84.)
55. Maurer, U.M., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Berlin, Heidelberg (Aug 2007). doi: 10.1007/978-3-540-74143-5_8 (Cited on page 84.)
56. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC. pp. 427–437. ACM Press (May 1990). doi: 10.1145/100216.100273 (Cited on pages 4 and 9.)
57. rd, I.D.: On sigma-protocols. https://cs.au.dk/ ivan/Sigma.pdf, [Accessed 30-09-2025] (Cited on page 6.)
58. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). doi: 10.1109/SFFCS.1999.814628 (Cited on page 41.)
59. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3), 161–174 (Jan 1991). doi: 10.1007/BF00196725 (Cited on page 6.)
60. Unger, N., Dechand, S., Bonneau, J., Fahl, S., Perl, H., Goldberg, I., Smith, M.: SoK: Secure messaging. In: 2015 IEEE Symposium on Security and Privacy. pp. 232–249. IEEE Computer Society Press (May 2015). doi: 10.1109/SP.2015.22 (Cited on page 2.)
61. Wee, H.: Déjà Q: Encore! Un petit IBE. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 237–258. Springer, Berlin, Heidelberg (Jan 2016). doi: 10.1007/978-3-662-49099-0_9 (Cited on page 6.)
62. Wee, H.: Circuit ABE with $\mathsf{poly}(\mathrm{depth}, \lambda)$-sized ciphertexts and keys from lattices. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part III. LNCS, vol. 14922, pp. 178–209. Springer, Cham (Aug 2024). doi: 10.1007/978-3-031-68382-4_6 (Cited on page 7.)
63. Wee, H., Wu, D.J.: Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In: Kalai, Y.T., Kamara, S.F. (eds.) CRYPTO 2025, Part III. LNCS, vol. 16002, pp. 204–235. Springer, Cham (Aug 2025). doi: 10.1007/978-3-032-01881-6_7 (Cited on page 7.)
64. Weidner, M., Kleppmann, M., Hugenroth, D., Beresford, A.R.: Key agreement for decentralized secure group messaging with strong security guarantees. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2024–2045. ACM Press (Nov 2021). doi: 10.1145/3460120.3484542 (Cited on page 2.)
65. Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J.: Ad hoc broadcast encryption (poster presentation). In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) ACM CCS 2010. pp. 741–743. ACM Press (Oct 2010). doi: 10.1145/1866307.1866416 (Cited on page 7.)

Appendix

A Clarifications

A.1 On Guaranteeing Network Delivery, in Practice

As one may note, the guarantee of decryption consistency that our schemes provide becomes meaningless if ciphertexts are not delivered by the network. In addition, as is well known, internet protocols (e.g. TCP/IP) do not (and cannot) ensure the delivery of messages.

Yet, as we now explain, the informal assumption that "the internet" provides a bulletin board—to which parties who are online (i.e. connected to the internet) can write to and read from—is a very mild one. (In turn, this bulletin board can be used to guarantee network delivery in the sense that online users can write and read ciphertexts from there.)

To see why this is the case, first note that it is common in the literature to make the informal assumption that as long as two parties are online at the same time—in the sense of being connected to the internet—they can communicate. (For example, the security of most blockchain protocols crucially relies on this assumption [31,41,4,19,45,14].)

Suppose a set of users agree on a set of n independent cloud service providers: $c_1, \ldots, c_n$. To implement a bulletin board, the users can follow the following informal protocol:

Write m**:** A user uploads m to each of the n cloud service providers.

Read: A user reads from each of the n cloud service providers; letting S be the set of all messages read, the user then uploads each of these messages back to each of the n cloud service providers. (Writing the messages back is to ensure that if the user reads a message that is injected by a malicious provider, then other users will also see this message.)

Note that as long as at least one of these n providers is honest and always available (which is the common goal for such services) the protocol above yields a bulletin board that in practice is good enough to ensure network delivery. This one out of n assumption is an *extremely* mild one, relative to what is necessary for byzantine broadcast and blockchain protocols (which additionally rely on honest majority type of assumptions). (If users would run their own storage servers, and ensured these would be online, then they would not even need to make the one out of n trust assumption.)

A.2 Committing Multi-Recipient Public-Key Encryption

In [34], Hashimoto et al. introduce Committing Multi-Recipient Public-Key Encryption (CmPKE) schemes: a variant of mPKE where ciphertexts include an additional commitment component which is used to enforce that each ciphertext can only be successfully decrypted into a single message. This property is weaker than the consistency guarantee of PKEBC schemes. This is because it does not

capture that if a ciphertext c is successfully decrypted by a recipient of c, then when any other recipient of c tries to decrypt c, their decryption must also succeed, and result in the same output as the first. Instead, it guarantees is that if a ciphertext c is successfully decrypted into a message m by a recipient of c, then when any other recipient of c tries to decrypt c, they either obtain the same message m or fail to decrypt.

B Game-Based Security Definitions

B.1 Pseudo-Random Generators

A Pseudo-Random Generator (PRG) is a PPT pair $\mathsf{PRG} = (\mathsf{Samp}, \mathsf{F})$ with an associated key-space $\mathcal{K}$ and (pseudo-)randomness-space $\mathcal{R}$, where Samp outputs elements from $\mathcal{K}$ and $\mathsf{F} : \mathcal{K} \to \mathcal{R}$. We assume $\mathcal{R} \coloneqq \mathcal{R}_1 \times \mathcal{R}_2$, meaning the output of $\mathsf{F}(k \in \mathcal{K})$ is a pair $(r_1, r_2) \in \mathcal{R}_1 \times \mathcal{R}_2$.

Consider a Pseudo-Random Generator scheme $\mathsf{PRG} = (\mathsf{Samp}, \mathsf{F})$. For $\mathbf{b} \in \{0, 1\}$, we define the following oracle(s):

$\mathcal{O}^{\mathbf{0}}_{\mathsf{F}}$: 1. $k \leftarrow \mathsf{Samp}$; 2. output $\mathsf{F}(k)$;
$\mathcal{O}^{\mathbf{1}}_{\mathsf{F}}$: 1. $r \leftarrow \mathcal{R}$; 2. output r.

Definition 20 (Pseudorandomness). $\mathcal{O}^{\mathbf{b}}_{\mathsf{PRG}} \coloneqq (\mathcal{O}^{\mathbf{b}}_{\mathsf{F}})$.

An adversary $\mathbf{A}$ (ε, t)-breaks the (n)-Pseudorandomness of PRG if it runs in time t, queries $\mathcal{O}_{\mathsf{F}}$ at most n times, and satisfies $Adv^{\mathsf{PRG}}(\mathbf{A}) \geq \varepsilon$.

The following security notion relies on a modified $\mathcal{O}_{\mathsf{F}}$ oracle:

$\mathcal{O}^{\mathbf{0}}_{\mathsf{F}}(k)$: 1. output $(r_1, r_2) \leftarrow \mathsf{F}(k)$.

Definition 21 (Left-Collision Resistance: $\neg$L-Coll-Res). *A query $\mathcal{O}_{\mathsf{F}}(k)$ outputs a pair $(r_1, \cdot)$. A later query $\mathcal{O}_{\mathsf{F}}(k')$ with $k \neq k'$ outputs a pair $(r_1, \cdot)$.*

An adversary $\mathbf{A}$ (ε, t)-breaks the Left-Collision Resistance of PRG if it runs in time t and satisfies $Adv^{\mathsf{L\text{-}Coll\text{-}Res}}(\mathbf{A}) \geq \varepsilon$. Finally, we say PRG is (ε, t, n)-secure if:

- no adversary $\mathbf{A}$ (ε, t)-breaks PRG's (n)-Pseudorandomness; and
- no adversary $\mathbf{A}$ (ε, t)-breaks PRG's Left-Collision Resistance.

B.2 Digital Signature Schemes

A Digital Signature Scheme (DSS) is a triple $\mathsf{DSS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ of PPTs. Below we state the definitions of Correctness, Existential Unforgeability under adaptive corruptions ($\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$) and (One-Time) Strong Existential Unforgeability ($\mathsf{OT\text{-}sEUF\text{-}CMA}$) for DSS schemes. The notions ahead make use of oracles $\mathcal{O}_{SK}$, $\mathcal{O}_{VK}$, $\mathcal{O}_S$ and $\mathcal{O}_V$, which, for a $\mathsf{DSS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, are defined as:

$\mathcal{O}_{SK}(i \in \mathbb{N})$: if there was a prior query on i, output the existing key-pair for i; otherwise, compute and store $(\mathtt{vk}_i, \mathtt{sk}_i) \leftarrow \mathsf{Gen}(1^k)$, and output the pair;

$\mathcal{O}_{VK}(i)$ $(\mathtt{vk}_i, \mathtt{sk}_i) \leftarrow \mathcal{O}_{SK}(i)$; output $\mathtt{vk}_i$;
$\mathcal{O}_S(i, m)$: output $\sigma \leftarrow \mathsf{Sign}_{\mathtt{sk}_i}(m)$;
$\mathcal{O}_V(i, m, \sigma)$: output $\mathsf{Vrfy}_{\mathtt{vk}_i}(m, \sigma)$.

Definition 22 (Correctness: $\neg$Corr). *A query $\mathcal{O}_S(i, m)$ outputs a signature σ; a later query $\mathcal{O}_V(i, m, \sigma)$ outputs 0.*

Definition 23 (Unforgeability: $\neg$EUF-CMA$^{\mathsf{adap}}$). *A query $\mathcal{O}_V(i^*, m^*, \sigma^*)$ outputs 1; there is no query $\mathcal{O}_S(i^*, m^*)$ or $\mathcal{O}_{SK}(i^*)$ prior to the $\mathcal{O}_V$ query above.*

Definition 24 (Replay Unforgeability: $\neg$OT-sEUF-CMA). *A query $\mathcal{O}_V(i^*, m^*, \sigma^*)$ outputs 1; no prior query $\mathcal{O}_S(i^*, m^*)$ output σ^*; for every $i \in \mathbb{N}$ there is at most one query $\mathcal{O}_S(i, \cdot)$.*

An adversary $\mathbf{A}$ (ε, t)-breaks the (n, q_S, q_V)-Correctness (resp. -EUF-CMA$^{\mathsf{adap}}$ and -OT-sEUF-CMA security) of DSS if it runs in time t, queries $\mathcal{O}_{SK}$, $\mathcal{O}_{VK}$, $\mathcal{O}_S$ and $\mathcal{O}_V$ on at most n different indices, makes at most q_S and q_V queries to oracles $\mathcal{O}_S$ and $\mathcal{O}_V$, respectively, and $Adv^{\neg\mathsf{Corr}}(\mathbf{A})$ (resp. $Adv^{\neg\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}(\mathbf{A})$ and $Adv^{\neg\mathsf{OT\text{-}sEUF\text{-}CMA}}(\mathbf{A})$) is greater or equal to ε.

Definition 25 (Public-Key Collision Resistance). $\mathsf{DSS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is n-Party ε-Public Key Collision-Resistant if*

$$\Pr\left[\left|\{\mathtt{vk}_1, \ldots, \mathtt{vk}_n\}\right| < n \;\middle|\; \begin{array}{c}(\mathtt{vk}_1, \mathtt{sk}_1) \leftarrow \mathsf{Gen}(1^k) \\ \ldots \\ (\mathtt{vk}_n, \mathtt{sk}_n) \leftarrow \mathsf{Gen}(1^k)\end{array}\right] \leq \varepsilon.$$

Finally, DSS is $(\varepsilon_{\mathsf{Corr}}, \varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, \varepsilon_{\mathsf{OT\text{-}sEUF\text{-}CMA}}, \varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}, t, n, q_S, q_V)$-secure if no adversary $\mathbf{A}$:

1. $(\varepsilon_{\mathsf{Corr}}, t)$-breaks the (n, q_S, q_V)-Correctness of DSS,
2. $(\varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, t)$-breaks the (n, q_S, q_V)-EUF-CMA$^{\mathsf{adap}}$ security of DSS, or
3. $(\varepsilon_{\mathsf{OT\text{-}sEUF\text{-}CMA}}, t)$-breaks the (n, q_S, q_V)-OT-sEUF-CMA security of DSS,

and DSS is n-Party $\varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}$-Public Key Collision-Resistant.

B.3 Hash Proof System

A Hash Proof System [16,42] for a relation $R \subseteq \mathcal{X} \times \mathcal{W}$ is a tuple of algorithms $\mathsf{HPS} = (\mathsf{SK\text{-}Sample}, \mathsf{PK\text{-}Of}, \mathsf{Samp}^{\in}, \mathsf{Samp}^{\notin}, \mathsf{Pub}, \mathsf{Priv})$ where:

$\mathsf{SK\text{-}Sample}(1^k)$: outputs a secret-key $\mathtt{sk}$;
$\mathsf{PK\text{-}Of}(\mathtt{sk})$: outputs a public-key $\mathtt{pk}$;
$\mathsf{Samp}^{\in}(1^k)$: outputs a pair $(x, w) \in R$;
$\mathsf{Samp}^{\notin}(1^k)$: outputs an element $x \in \mathcal{X} \setminus L_R$;
$\mathsf{Pub}(\mathtt{pk}, x, w)$: for $(x, w) \in R$, outputs π;
$\mathsf{Priv}(\mathtt{sk}, x)$: outputs π.

Algorithms $\mathsf{PK\text{-}Of}$, Pub and Priv are deterministic. The secret-key and public-key spaces are, respectively, $\mathcal{S} \coloneqq \mathrm{Supp}(\mathsf{SK\text{-}Sample}(1^k))$ and $\mathcal{P} \coloneqq \{\mathsf{PK\text{-}Of}(\mathtt{sk})\}_{\mathtt{sk}\in\mathcal{S}}$. We may alternatively write $\mathtt{sk} \leftarrow \mathcal{S}$ to mean $\mathtt{sk} \leftarrow \mathsf{SK\text{-}Sample}(1^k)$. We define $\mathcal{K} \coloneqq \{\mathsf{Priv}(\mathtt{sk}, x)\}_{(\mathtt{sk},x)\in\mathcal{S}\times\mathcal{X}}$ and require that it is efficiently sampleable.

Definition 26 (Perfect Correctness). *$\forall(x, w) \in R$, $\forall\mathtt{sk} \in \mathcal{S}$:*

$$\mathsf{Pub}(\mathsf{PK\text{-}Of}(\mathtt{sk}), x, w) = \mathsf{Priv}(\mathtt{sk}, x).$$

Definition 27 ($\gamma_{\mathtt{pk}}$-Spreadness).

$$\mathrm{H}_\infty(\mathsf{PK\text{-}Of}(\mathsf{SK\text{-}Sample}(1^k))) \geq \gamma_{\mathtt{pk}}.$$

Definition 28 ($\mathsf{Samp}^{\in}$-Correctness). $\mathrm{Supp}(\mathsf{Samp}^{\in}(1^k)) \subseteq R$.

Definition 29 ($\mathsf{Samp}^{\notin}$-Correctness). $\mathrm{Supp}(\mathsf{Samp}^{\notin}(1^k)) \cap L_R = \emptyset$.

Definition 30 (γ_x-Statement-Min-Entropy). $\mathrm{H}_\infty(\mathsf{Samp}^{\in}(1^k)) \geq \gamma_x$.

Definition 31 (Pairwise Universality). *For $x \in \mathcal{X}$, define*

$$\mathbf{D}_x^{\mathbf{b}} \coloneqq \left(\begin{array}{c|c} x, & \mathtt{sk} \leftarrow \mathsf{SK\text{-}Sample}(1^k), \\ \mathsf{PK\text{-}Of}(\mathtt{sk}), & K^{\mathbf{0}} \coloneqq \mathsf{Priv}(\mathtt{sk}, x), \\ K^{\mathbf{b}} & K^{\mathbf{1}} \leftarrow \mathcal{K} \end{array}\right).$$

HPS *is Pairwise Universal if* $\forall x \in \mathcal{X} \setminus L_R :\ \mathbf{D}_x^{\mathbf{0}} \equiv \mathbf{D}_x^{\mathbf{1}}$.

For scheme $\mathsf{HPS} = (\mathsf{SK\text{-}Sample}, \mathsf{PK\text{-}Of}, \mathsf{Samp}^{\in}, \mathsf{Samp}^{\notin}, \mathsf{Pub}, \mathsf{Priv})$, define oracles:

$\mathcal{O}_X^{\mathbf{0}}$: sample $(x, w) \leftarrow \mathsf{Samp}^{\in}(1^k)$; output x;
$\mathcal{O}_X^{\mathbf{1}}$: sample $x \leftarrow \mathsf{Samp}^{\notin}(1^k)$; output x.

Definition 32 (Subset Membership Problem: SM). $\vec{\mathcal{O}}^{\mathbf{b}} \coloneqq \mathcal{O}_X^{\mathbf{b}}$.

An adversary $\mathbf{A}$ (ε, t)-decides the (n)-Subset Membership Problem of HPS's relation if $\mathbf{A}$ runs in time at most t, queries $\mathcal{O}_X$ at most n times and $Adv^{\mathsf{SM}}(\mathbf{A}) \geq \varepsilon$. We say HPS is $\big(\varepsilon, t, n, \gamma_{\mathtt{pk}}, \gamma_x\big)$-secure if it is

- Perfectly Correct;
- $\gamma_{\mathtt{pk}}$-Public-Key Min-Entropic;
- $\mathsf{Samp}^{\in}$-Correct;
- $\mathsf{Samp}^{\notin}$-Correct;
- γ_x-Statement-Min-Entropic;
- Pairwise Universal; and
- no adversary (ε, t)-decides its (n)-Subset Membership Problem.

B.4 Multi-Recipient Public Key Encryption

Following [44], an Multi-Recipient Public Key Encryption scheme for a message space $\mathcal{M}$ is a tuple $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ of algorithms where:

- Gen: on input 1^k, generates a key-pair $(\mathtt{pk}, \mathtt{sk})$;
- Enc: on input $(\vec{v}, m)$, where $\vec{v} \coloneqq (\mathtt{pk}_1, \ldots, \mathtt{pk}_{|\vec{v}|})$ is a vector of public keys of the intended receivers and m is a message with, generates a ciphertext c;
- $\mathsf{Extract}$: on input (i, c), where $i \in \mathbb{N}$ is an index indicating the part of c to be extracted, $\mathsf{Extract}$ extracts the i-th part of c, say c_i, and outputs it;
- Dec: on input $(\mathtt{sk}, c_i)$, where $\mathtt{sk}$ is the receiver's secret key and c_i is as above, Dec decrypts c_i using $\mathtt{sk}$ and outputs the decrypted message m (or $\perp$ if the i-th part of c fails to decrypt).

It is required that the vector $\vec{v}$ of public keys input to Enc contains no duplicates, i.e. $|\mathrm{Set}(\vec{v})| = |\vec{v}|$. (In [44] this requirement is implicit in the syntax of the scheme: encryption takes a set of public keys instead of a vector.)

Below we state the multi-challenge variants of Correctness, $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ and $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}^{\mathsf{adap}}$ (or, more simply, CPA and $\mathsf{CPA}^{\mathsf{adap}}$, respectively) security for these schemes. Throughout the rest of this section, let $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ be an mPKE scheme. As before, we assume the game systems of the following definitions have (an implicitly defined) security parameter k.

The notions ahead provide adversaries with access to oracles $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$, $\mathcal{O}_{Ext}$ and $\mathcal{O}_D$, which, for a security parameter k work as follows:

$\mathcal{O}_{SK}(B_j)$: on the first query on input B_j, sample (and define) $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathsf{Gen}(1^k)$; output B_j's key-pair $(\mathtt{pk}_j, \mathtt{sk}_j)$;
$\mathcal{O}_{PK}(B_j)$: $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathcal{O}_{SK}(B_j)$; output $\mathtt{pk}_j$;
$\mathcal{O}_E(\vec{V}, m; r)$: output $c \leftarrow \mathsf{mPKE.Enc}(\vec{v}, m; r)$, where $\vec{v}$ is $\vec{V}$'s corresponding vector of public keys; if no r is input, sample r uniformly at random from $\mathsf{mPKE.Enc}$'s randomness space;
$\mathcal{O}_{Ext}(c, i \in \mathbb{N})$: output $c_i \leftarrow \mathsf{mPKE.Extract}(c, i)$;
$\mathcal{O}_D(B_j, c_i)$: output $\mathsf{mPKE.Dec}(\mathtt{sk}_j, c_i)$.

Definition 33 (Correctness: $\neg$Corr). *A query $\mathcal{O}_E(\vec{V}, m; \cdot)$ outputs c. A query $\mathcal{O}_{Ext}(c, i \in |\vec{V}|)$ outputs c_i. A query $\mathcal{O}_D(V_i, c_i)$ outputs m' with $m' \neq m$.*

A (computationally unbounded) adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{Corr}})$-breaks the (n)-Correctness of mPKE if $\mathbf{A}$ queries $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most n different parties and satisfies $Adv^{\mathsf{Corr}}(\mathbf{A}) \geq \varepsilon_{\mathsf{Corr}}$. We say that an mPKE is perfectly correct, if $\varepsilon_{\mathsf{Corr}} = 0$.

The $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ games provide adversaries with access to the oracles described above, but where oracle $\mathcal{O}_E$ is modified as follows:

$\mathcal{O}_{E^{\mathbf{b}}}((\vec{V}_0, m_0), (\vec{V}_1, m_1))$: Proceed as the original oracle would on input $\mathcal{O}_E(\vec{V}_{\mathbf{b}}, m_{\mathbf{b}})$.

Definition 34 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$). $\vec{\mathcal{O}}^{\mathbf{b}}_{\mathsf{CPA}} \coloneqq \vec{\mathcal{O}}[PK, E^{\mathbf{b}}, Ext]$.

Definition 35 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}^{\mathsf{adap}}$). $\vec{\mathcal{O}}^{\mathbf{b}}_{\mathsf{CPA}^{\mathsf{adap}}} \coloneqq \vec{\mathcal{O}}[SK, PK, E^{\mathbf{b}}, Ext]$.
Disallowed: *making queries* $\mathcal{O}_E((\vec{V}_0, \cdot), (\vec{V}_1, \cdot))$ *and* $\mathcal{O}_{SK}(B_j \in \vec{V}_0 \cup \vec{V}_1)$.

We say $\mathbf{A}$ (ε, t)-breaks the (n, q_E, d_E)-CPA (resp. $\mathsf{CPA}^{\mathsf{adap}}$) security of a scheme mPKE if $\mathbf{A}$ runs in time at most t, queries the oracles it has access to on at most n different parties, makes at most q_E queries to oracle $\mathcal{O}_E$ with the sum of the lengths of the party vectors input to $\mathcal{O}_E$ being at most d_E, and satisfies $Adv^{\mathsf{CPA}}(\mathbf{A}) \geq \varepsilon$ (resp. $Adv^{\mathsf{CPA}^{\mathsf{adap}}}(\mathbf{A}) \geq \varepsilon$).

Definition 36 (Public-Key Collision Resistance). *We say that* $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ *is* n*-Party* ε*-Public-Key Collision Resistant if*

$$\Pr\left[\left|\{\mathtt{pk}_1, \ldots, \mathtt{pk}_n\}\right| < n \middle| i \in [n] : (\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{Gen}(1^k)\right] \leq \varepsilon.$$

Definition 37 (γ-spread). *We say that* $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ *is* γ-spread *if for all* $n \in \mathbb{N}$, $0 \leq j < n$, $(\mathtt{pk}_j, \mathtt{sk}_j) \in \mathrm{Supp}(\mathsf{Gen}(1^k))$, $\vec{v} \coloneqq (\mathtt{pk}_0, \ldots, \mathtt{pk}_n)$, *all* $m \in \mathcal{M}$*:* $-\log \max_{c \in \mathcal{C}} \Pr_{\mathtt{r}_\$}[c = \mathsf{Enc}(\vec{v}, m; \mathtt{r}_\$)] \geq \gamma$. *A* γ*-spread* mPKE *satisfies that for any such* $\vec{v}, m$ *and any* $c \in \mathrm{Supp}(\mathsf{Enc}(\vec{v}, m))$

$$\Pr_{\mathtt{r}_\$}[c = \mathsf{Enc}(\vec{v}, m; \mathtt{r}_\$)] \leq 2^{-\gamma}.$$

The definition below relies on a function $\mathsf{Project}$ which, given n-long vectors $\vec{s}$ and $\vec{t}$ and a symbol α, outputs a vector $\vec{u} \coloneqq \mathsf{Project}(\vec{t}, \vec{s}, \alpha)$ where, for $i \in [n]$:

$$u_i \coloneqq \begin{cases} t_i & \text{if } s_i = \alpha \\ \perp & \text{otherwise} \end{cases}$$

Furthermore, let $\not\downarrow$ be a function that maps a given vector $\vec{u}$ into a new vector $\vec{v}$ s.t. $\vec{v}$'s i-th entry is the i-th non-$\perp$ entry of vector $\vec{u}$. The following notion involves an extra probabilistic algorithm Ext with the following syntax:

$\mathsf{Ext}(c, \vec{\mathtt{sk}}, m)$: for an n receiver ciphertext c, an n'-long vector $\vec{\mathtt{sk}}$—in which $n' - n$ entries are $\perp$ and the remainder are secret keys—and a message m, Ext outputs an n'-receiver ciphertext c'. Each entry of vector $\vec{\mathtt{sk}}$ is either a secret key or $\perp$; if the j-th entry of $\vec{\mathtt{sk}}$ is the i-th $\perp$ entry of $\vec{\mathtt{sk}}$, then the j-th component of the output ciphertext is to be set as the i-th component of input ciphertext c.

Definition 38 (Ext-ctxt extendable). $\mathsf{mPKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec})$ *is* Ext-*ctxt extendable if* $\forall k, n \in \mathbb{N}$, $\forall(\mathtt{pk}_1, \mathtt{sk}_1), \ldots, (\mathtt{pk}_n, \mathtt{sk}_n) \in \mathrm{Supp}(\mathsf{Gen}(1^k))$, $\forall \vec{s} \in \{0,1\}^n \setminus \{0^n, 1^n\}$, $\forall m_0, m_1 \in \mathcal{M}$, *letting* $\vec{\mathtt{pk}} \coloneqq (\mathtt{pk}_1, \ldots, \mathtt{pk}_n)$, $\vec{\mathtt{sk}} \coloneqq (\mathtt{sk}_1, \ldots, \mathtt{sk}_n)$, $\vec{u}^b \coloneqq \mathsf{Project}(\vec{u}, \vec{s}, b)$, *for* $u \in \{\mathtt{pk}, \mathtt{sk}\}$*:*

Correctness

$$\left(c \,|\, c \xleftarrow{\$} \mathsf{Enc}(\vec{\mathtt{pk}}, m_0)\right) \equiv \left(c \,|\, c \xleftarrow{\$} \mathsf{Ext}(\mathsf{Enc}(\not\downarrow(\vec{\mathtt{pk}}^1), m_0), \vec{\mathtt{sk}}^0, m_0)\right)$$

Commutativity

$$\big(c \,|\, c \xleftarrow{\$} \mathsf{Ext}(\mathsf{Enc}(\not\downarrow(\vec{\mathtt{pk}}^1), m_1), \vec{\mathtt{sk}}^0, m_0)\big) \equiv \big(c \,|\, c \xleftarrow{\$} \mathsf{Ext}(\mathsf{Enc}(\not\downarrow(\vec{\mathtt{pk}}^0), m_0), \vec{\mathtt{sk}}^1, m_1)\big)$$

We say mPKE is $(\varepsilon_{\mathsf{Corr}}, \varepsilon_{\mathsf{CPA}}, \varepsilon_{\mathsf{CPA}^{\mathsf{adap}}}, \varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}, \gamma, t, n, q_E, d_E, \mathsf{Ext})$-secure if

- no (possibly computationally unbounded) adversary $(\varepsilon_{\mathsf{Corr}})$-breaks mPKE's (n)-Correctness;
- if no adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{CPA}}, t)$-breaks mPKE's (n, q_E, d_E)-CPA security;
- no adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{CPA}^{\mathsf{adap}}}, t)$-breaks mPKE's (n, q_E, d_E)-$\mathsf{CPA}^{\mathsf{adap}}$ security;
- mPKE is γ-spread;
- mPKE is Ext-ciphertext extendable; and
- mPKE is n-Party $\varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}$-Public-Key Collision Resistant.

B.5 Symmetric Encryption

A Symmetric Encryption (SKE) scheme with message space $\mathcal{M}$ and key space $\mathcal{K}$ is a triple $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic algorithms. For security parameter k:

- $\mathsf{Gen}(1^k)$: outputs a key $K \in \mathcal{K}$;
- $\mathsf{Enc}(K, m \in \mathcal{M})$: outputs a ciphertext c;
- $\mathsf{Dec}(K, c)$: either outputs a message $m \in \mathcal{M}$ or special symbol $\perp$ indicating decryption failure.

Consider a scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with message space $\mathcal{M}$. (We assume an implicitly defined security parameter k.)

Definition 39 (Perfect Correctness).

$$\forall K \in \mathrm{Supp}(\mathsf{Gen}(1^k)), \forall m \in \mathcal{M} : \Pr[\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m] = 1.$$

For $\mathbf{b} \in \{0, 1\}$, the $\mathsf{OT\text{-}IND\text{-}CCA}$ security games give adversaries are given access to oracles $\vec{\mathcal{O}}[\mathsf{Enc}^{\mathbf{b}}, \mathsf{Dec}]$:

$\mathcal{O}_{\mathsf{Enc}^{\mathtt{b}}}(m)$: i-th query: 1. $K_i \leftarrow \mathsf{Gen}(1^k)$; 2. $m_0 \coloneqq m$; 3. $m_1 \leftarrow \mathcal{M}$ 4. output $c_i \leftarrow \mathsf{Enc}(K_i, m_{\mathbf{b}})$.
$\mathcal{O}_{\mathsf{Dec}}(i \in \mathbb{N}, c)$: if less than i $\mathcal{O}_{\mathsf{Enc}^{\mathtt{b}}}$ queries were made or $c = c_i$, output $\perp$; else, letting K_i be the key sampled in the i-th $\mathcal{O}_{\mathsf{Enc}^{\mathtt{b}}}$ query, output $\mathsf{Dec}(K_i, c)$.

Definition 40 ($\mathsf{OT\text{-}IND\text{-}CCA}$). $\vec{\mathcal{O}}^{\mathbf{b}}_{\mathsf{OT\text{-}IND\text{-}CCA}} \coloneqq \vec{\mathcal{O}}[\mathsf{Enc}^{\mathbf{b}}, \mathsf{Dec}]$.
Disallowed: *i-th $\mathcal{O}_{\mathsf{Enc}^{\mathtt{b}}}$ query outputs c and there is a later query $\mathcal{O}_{\mathsf{Dec}}(i, c)$.*

We say $\mathbf{A}$ (ε, t)-breaks the (q_E, q_D)-$\mathsf{OT\text{-}IND\text{-}CCA}$ security of SKE if $\mathbf{A}$ runs in time at most t, makes at most q_E queries to $\mathcal{O}_{\mathsf{Enc}}$ and q_D queries to $\mathcal{O}_{\mathsf{Dec}}$, and satisfies $Adv^{\mathsf{OT\text{-}IND\text{-}CCA}}(\mathbf{A}) \geq \varepsilon$. We say SKE is $(\varepsilon, t, q_E, q_D)$-secure if it is Perfectly Correctness (Definition 39) and no adversary $\mathbf{A}$ (ε, t)-breaks SKE's (q_E, q_D)-$\mathsf{OT\text{-}IND\text{-}CCA}$ security.

B.6 Non-Interactive Proof System

Let R be some binary relation and L_R be its induced language. A *Non-Interactive Proof System* (NIPS) for L_R is a triple of probabilistic algorithms $\mathsf{NIPS} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy})$ where, for security parameter k:

- $\mathsf{Setup}(1^k)$: outputs a common reference string $\mathtt{crs}$;
- $\mathsf{Prv}_{\mathtt{crs}}(x, w)$: given $\mathtt{crs}$ and a statement-witness pair $(x, w) \in R$, outputs a proof π;
- $\mathsf{Vrfy}_{\mathtt{crs}}(x, \pi)$: given $\mathtt{crs}$, statement x and proof π, outputs either 1 or 0 (respectively, accepts or rejects).

In the following definitions, consider a $\mathsf{NIPS} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy})$ for a relation R, and let k be the security parameter. The security notions below (Definitions 41 and 42) provide adversaries with access to the following oracles:

$\mathcal{O}_{\mathsf{Setup}}$: If $\mathtt{crs}$ is undefined, compute and store $\mathtt{crs} \leftarrow \mathsf{Setup}(1^k)$; output $\mathtt{crs}$.
$\mathcal{O}_{\mathsf{Prv}}(x, w)$: Output $\mathsf{Prv}_{\mathtt{crs}}(x, w)$.
$\mathcal{O}_{\mathsf{Vrfy}}(x, \pi)$: Output $\mathsf{Vrfy}_{\mathtt{crs}}(x, \pi)$.

Definition 41 (Completeness: $\neg\mathsf{Complete}$). *A query $\mathcal{O}_{\mathsf{Prv}}(x, w)$ with $(x, w) \in R$ outputs π. A query $\mathcal{O}_{\mathsf{Vrfy}}(x, \pi)$ outputs 0.*

Definition 42 (Soundness: $\neg\mathsf{Sound}$). *A query $\mathcal{O}_{\mathsf{Vrfy}}(x \notin L_R, \pi)$ outputs 1.*

We say that an adversary $\mathbf{A}$ (ε, t)-breaks the (q_P, q_V)-Completeness (resp. Soundness) of scheme NIPS if $\mathbf{A}$ runs in time at most t, makes at most q_P and q_V queries to oracles $\mathcal{O}_{\mathsf{Prv}}$ and $\mathcal{O}_{\mathsf{Vrfy}}$, respectively, and satisfies $Adv^{\mathsf{Complete}}(\mathbf{A}) \geq \varepsilon$ (resp. $Adv^{\mathsf{Sound}}(\mathbf{A}) \geq \varepsilon$). We say NIPS is $(\varepsilon_{\mathsf{Complete}}, \varepsilon_{\mathsf{Sound}}, t, q_P, q_V)$-secure if no adversary $(\varepsilon_{\mathsf{Complete}}, t)$-breaks NIPS's (q_P, q_V)-Completeness and no adversary $(\varepsilon_{\mathsf{Sound}}, t)$-breaks NIPS's (q_P, q_V)-Soundness.

B.7 Non-Interactive Zero-Knowledge

A Non-Interactive Zero-Knowledge scheme $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy}, \mathsf{Sim} = (\mathsf{Sim.Setup}, \mathsf{Sim.Prv}))$ for a relation R consists of a scheme $\mathsf{NIPS} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy})$ for R and a simulator $\mathsf{Sim} = (\mathsf{Sim.Setup}, \mathsf{Sim.Prv})$, where:

- $\mathsf{Sim.Setup}(1^k)$: outputs a Common Reference String/Trapdoor pair $(\mathtt{crs}, \varphi)$;
- $\mathsf{Sim.Prv}(\mathtt{crs}, \varphi, x)$: given $\mathtt{crs}$ and φ as above, and given a statement x, outputs a proof π.

Consider a scheme $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy}, \mathsf{Sim} = (\mathsf{Sim.Setup}, \mathsf{Sim.Prv}))$ for a relation R. We define the following oracles:

$\mathcal{O}_{\mathsf{Sim.Setup}}$: If $\mathtt{crs}$ is undefined, sample $(\mathtt{crs}, \varphi) \leftarrow \mathsf{Setup}(1^k)$; output $\mathtt{crs}$.
$\mathcal{O}_{\mathsf{Sim.Prv}}(x, w)$: Output $\mathsf{Sim.Prv}_{(\mathtt{crs}, \varphi)}(x)$.

Definition 43 (Zero Knowledge).

$$\vec{\mathcal{O}}^{\mathbf{0}}_{\mathsf{zk}} \coloneqq \vec{\mathcal{O}}[\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy}]$$
$$\vec{\mathcal{O}}^{\mathbf{1}}_{\mathsf{zk}} \coloneqq \vec{\mathcal{O}}[\mathsf{Sim.Setup}, \mathsf{Sim.Prv}, \mathsf{Vrfy}].$$

Disallowed: *queries to* $\mathcal{O}_{\mathsf{Prv}}$ *or* $\mathcal{O}_{\mathsf{Sim.Prv}}$ *on inputs* $(x, w) \notin R$.

We now introduce Simulation Soundness for NIZK [58]. The game system defined by this notion provides adversaries with access to oracles $\mathcal{O}_{\mathsf{Sim.Setup}}$ and $\mathcal{O}_{\mathsf{Sim.Prv}}$ as defined above, and to oracle $\mathcal{O}_{\mathsf{Vrfy}}$.

Definition 44 (Simulation Soundness: ¬SS). *A query* $\mathcal{O}_{\mathsf{Vrfy}}(x \notin L_R, \pi)$ *outputs* 1. *No prior query* $\mathcal{O}_{\mathsf{Sim.Prv}}(x)$ *output* π.

Adversary $\mathbf{A}$ (ε, t)-breaks the (q_P, q_V)-Zero Knowledge (resp. Simulation Soundness) of scheme NIZK if it makes at most q_P and q_V queries to $\mathcal{O}_{\mathsf{Prv}}/\mathcal{O}_{\mathsf{Sim.Prv}}$ and $\mathcal{O}_{\mathsf{Vrfy}}$, respectively, and satisfies $Adv^{\mathsf{zk}}(\mathbf{A}) \geq \varepsilon$ (resp. $Adv^{\mathsf{SS}}(\mathbf{A}) \geq \varepsilon$). Finally, NIZK is $(\varepsilon_{\mathsf{Complete}}, \varepsilon_{\mathsf{Sound}}, \varepsilon_{\mathsf{zk}}, \varepsilon_{\mathsf{SS}}, t, q_P, q_V)$-secure if NIPS $(\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrfy})$ is $(\varepsilon_{\mathsf{Complete}}, \varepsilon_{\mathsf{Sound}}, t, q_P, q_V)$-secure and no adversary $(\varepsilon_{\mathsf{zk}}, t)$-breaks NIZK's (q_P, q_V)-Zero Knowledge, or $(\varepsilon_{\mathsf{SS}}, t)$-breaks NIZK's (q_P, q_V)-Simulation Soundness.

C Security Analysis of $\mathsf{PKEBC}_{\mathsf{SM}}$

C.1 Correctness

Theorem 6. *If* mPKE *is* $(0, \cdot, t, 2n, \cdot)$*-secure,* NIZK *is* $(\varepsilon_{\mathsf{Complete}}, \cdot, \cdot, \cdot, t, q_P, q_V)$*-secure,* SKE *is Perfectly Correct and* PRG.F *is* $\varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}}$*-left collision resistant, then no adversary* $\mathbf{A}$ $(\varepsilon, \cdot)$*-breaks* $\mathsf{PKEBC}_{\mathsf{SM}}$*'s* $(n, \cdot, q_P, q_V, \cdot)$*-Correctness, with* $\varepsilon > \varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}} + \varepsilon_{\mathsf{Complete}}$.

Proof. The proof proceeds via a hybrid argument.

Algorithm 9 Description of the changes $\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{G}^1$.

$\mathcal{O}_E(\vec{V}, m)$
  $(1^k, \mathtt{crs}) \leftarrow \mathcal{O}_{PP}()$
  $\vec{v} \leftarrow \mathcal{O}_{PK}(\vec{V})$
  $R \xleftarrow{\$} \mathcal{R}$
  $\vec{u} := (v_{1,0}, v_{1,1}, \ldots, v_{n,0}, v_{n,1})$
  $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{u}, R; \mathtt{r}_{\$})$
  $\pi \leftarrow \mathsf{NIZK.Prv}_{\mathsf{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, w := (\mathtt{r}_{\$}, R))$
  $(\tau, K) \leftarrow \mathsf{PRG.F}(R)$
  $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}, m, \pi))$
  $c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}})$
  $\mathcal{V}[c] := \vec{V}, \quad \mathcal{K}[c] := K$
  $\mathcal{M}[c] := (\vec{v}, m, \pi), \quad \mathcal{R}[c] := R$
  return c

$\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$
  $(1^k, \mathtt{crs}) \leftarrow \mathcal{O}_{PP}()$
  $(\mathtt{pk}, \mathtt{sk}) := \mathcal{O}_{SK}(B_j)$
  if $\mathcal{V}[c] \neq \bot \wedge B_j \in \mathcal{V}[c]$ **:**
    $K \leftarrow \bot$
    for $i = 1, \ldots, n$ **:**
      $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i - 1 + b)$
      $R_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
      $\vec{V} := \mathcal{V}[c]$
      if $\vec{V}_i = B_j$**:** $R_i := \mathcal{R}[c]$
      $(\tau_i, K_i) \leftarrow \mathsf{PRG.F}(R_i)$
      if $\tau_i = \tau$**:** $K \leftarrow K_i$**; break**
    if $K = \bot$**: return** $\bot$
    $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
    if $(\vec{v}, m, \pi) = \bot \vee \mathtt{pk} \neq v_i$**: return** $\bot$
    $\vec{u} := (v_{1,0}, v_{1,1}, \ldots, v_{n,0}, v_{n,1})$
    if $\mathsf{NIZK.Vrfy}_{\mathsf{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi) = 0$ **:**
      return $\bot$
    return $(\vec{v}, m)$
  else: Proceed as original $\mathcal{O}_D$

Algorithm 10 Description of the changes $\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$.

$\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$
  $(1^k, \mathtt{crs}) \leftarrow \mathcal{O}_{PP}()$
  $(\mathtt{pk}, \mathtt{sk}) := \mathcal{O}_{SK}(B_j)$
  if $\mathcal{V}[c] \neq \bot \wedge B_j \in \mathcal{V}[c]$ **:**
    $K \leftarrow \bot$
    for $i = 1, \ldots, n$ **:**
      $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i - 1 + b)$
      $R_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
      $\vec{V} := \mathcal{V}[c]$
      if $\vec{V}_i = B_j$**:** $R_i := \mathcal{R}[c]$
      $(\tau_i, K_i) \leftarrow \mathsf{PRG.F}(R_i)$
      if $\tau_i = \tau$**:** $K \leftarrow K_i$**; break**
    if $K = \bot$**: return** $\bot$
    $K := \mathcal{K}[c]$
    $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
    if $(\vec{v}, m, \pi) = \bot \vee \mathtt{pk} \neq v_i$**: return** $\bot$
    $\vec{u} := (v_{1,0}, v_{1,1}, \ldots, v_{n,0}, v_{n,1})$
    if $\mathsf{NIZK.Vrfy}_{\mathsf{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi) = 0$ **:**
      return $\bot$
    return $(\vec{v}, m)$
  else: Proceed as original $\mathcal{O}_D$

$\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{G}^1$**:** These games are identical unless there is a query $\mathcal{O}_E(\vec{V}, m)$ outputting $c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}})$ and a party B_j with $\vec{V}_i = B_j$ such that the i-th part of c_{mPKE} decrypts to a value different than the R sampled during the $\mathcal{O}_E$-call for B_j's public key. This cannot happen due to the Perfect Correctness of mPKE therefore: $\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^{\mathsf{Corr}} \Rightarrow \neg\mathsf{Corr}\right]\right| = 0$.

$\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$**:** These games are identical, unless for a $c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}})$ output by $\mathcal{O}_E$, the **for**-loop in an $\mathcal{O}_D(\cdot, c)$-query returns a $K \neq \mathcal{K}[c]$. The **for**-loop only accepts a key K if decrypting c_{mPKE} yielded an R with $(\tau, K) \leftarrow \mathsf{PRG.F}(R)$. If this occurs, R and the R^* from which $K^* = \mathcal{K}[c]$ was obtained result in a left-collision

for PRG. Thus we have $\left|\Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}}$.

Algorithm 11 Description of the changes $\mathbf{G}^2 \rightsquigarrow \mathbf{G}^3$.

$\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$
  $(1^k, \mathtt{crs}) \leftarrow \mathcal{O}_{PP}()$
  $(\mathtt{pk}, \mathtt{sk}) := \mathcal{O}_{SK}(B_j)$
  if $\mathcal{V}[c] \neq \bot \wedge B_j \in \mathcal{V}[c]$ **:**
    $K \leftarrow \bot$
    for $i = 1, \ldots, n$ **:**
      $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i - 1 + b)$
      $R_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
      $\vec{V} := \mathcal{V}[c]$
      if $\vec{V}_i = B_j$**:** $R_i := \mathcal{R}[c]$
      $(\tau_i, K_i) \leftarrow \mathsf{PRG.F}(R_i)$
      if $\tau_i = \tau$**:** $K \leftarrow K_i$**; break**
    if $K = \bot$**: return** $\bot$
    $K := \mathcal{K}[c]$
    $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
    if $(\vec{v}, m, \pi) = \bot \vee \mathtt{pk} \neq v_i$**: return** $\bot$
    $\vec{v}, m, \pi := \mathcal{M}[c]$
    $\vec{u} := (v_{1,0}, v_{1,1}, \ldots, v_{n,0}, v_{n,1})$
    if $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi) = 0$ **:**
      return $\bot$
    return $(\vec{v}, m)$
  else: Proceed as original $\mathcal{O}_D$

Algorithm 12 Description of the changes $\mathbf{G}^3 \rightsquigarrow \mathbf{G}^4$.

$\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$
  $(1^k, \mathtt{crs}) \leftarrow \mathcal{O}_{PP}()$
  $(\mathtt{pk}, \mathtt{sk}) := \mathcal{O}_{SK}(B_j)$
  if $\mathcal{V}[c] \neq \bot \wedge B_j \in \mathcal{V}[c]$ **:**
    $K \leftarrow \bot$
    for $i = 1, \ldots, n$ **:**
      $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i - 1 + b)$
      $R_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
      $\vec{V} := \mathcal{V}[c]$
      if $\vec{V}_i = B_j$**:** $R_i := \mathcal{R}[c]$
      $(\tau_i, K_i) \leftarrow \mathsf{PRG.F}(R_i)$
      if $\tau_i = \tau$**:** $K \leftarrow K_i$**; break**
    if $K = \bot$**: return** $\bot$
    $K := \mathcal{K}[c]$
    $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
    if $(\vec{v}, m, \pi) = \bot \vee \mathtt{pk} \neq v_i$**: return** $\bot$
    $\vec{v}, m, \pi := \mathcal{M}[c]$
    $\vec{u} := (v_{1,0}, v_{1,1}, \ldots, v_{n,0}, v_{n,1})$
    if $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi) = 0$ **:**
      return $\bot$
    return $(\vec{v}, m)$
  else: Proceed as original $\mathcal{O}_D$

$\mathbf{G}^2 \rightsquigarrow \mathbf{G}^3$: The Perfect Correctness of SKE implies these games are identical: $\left|\Pr\left[\mathbf{AG}^3 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Corr}\right]\right| = 0$.

$\mathbf{G}^3 \rightsquigarrow \mathbf{G}^4$: These games are perfectly indistinguishable unless $\mathbf{A}$ queries $\mathcal{O}_D(B_j, c)$ for a $c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}})$ output by $\mathcal{O}_E$ with $\mathcal{M}[c] = (\vec{v}, m, \pi)$ such that $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi)$ returns 0. We can use an adversary distinguishing between games $\mathbf{G}^3$ and $\mathbf{G}^4$ to construct an adversary for the Soundness game of NIZK, implying $\left|\Pr\left[\mathbf{AG}^4 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^3 \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon_{\mathsf{Complete}}$. The reduction proceeds as game $\mathbf{G}^3$, but calls $\mathcal{O}_{\mathsf{Setup}}()$ instead of $\mathsf{NIZK.Setup}(1^k)$, calls $\mathcal{O}_{\mathsf{Prv}}((\vec{u}, c_{\mathsf{mPKE}}), (\mathtt{r}_{\$}, R))$ instead of $\mathsf{NIZK.Prv}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, w := (\mathtt{r}_{\$}, R))$, and calls $\mathcal{O}_{\mathsf{Vrfy}}((\vec{u}, c_{\mathsf{mPKE}}), \pi)$ instead of $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi)$. To conclude, note $\mathbf{G}^4$ cannot be won. □

C.2 Replay-Correctness

Theorem 7. *If* mPKE *is* γ*-spread then no adversary* $\mathbf{A}$ $(\varepsilon, \cdot)$*-breaks* $\mathsf{PKEBC_{SM}}$*'s*

$$(\cdot, \cdot, q_E, \cdot, \cdot, \cdot)\text{-}\textit{Replay-Correctness},$$

with $\varepsilon > (q_E)^2 2^{-(1+\gamma)}$.

Proof. Since each $\mathsf{PKEBC_{SM}}$-ciphertext contains a mPKE-ciphertext and mPKE is γ-spread, the probability that there is a collision among queries $c \leftarrow \mathcal{O}_E(\cdot, \cdot)$ is at most $(q_E)^2 2^{-(1+\gamma)}$ by a union bound.

C.3 Adversarial Replay-Correctness

Theorem 8. *If* mPKE *is γ-spread then no adversary* $\mathbf{A}$ *$(\varepsilon, \cdot)$-breaks* $\mathsf{PKEBC}_{\mathsf{SM}}$*'s*

$$(\cdot, \cdot, q_E, \cdot, \cdot, q_P)\text{-}\textit{Adversarial Replay-Correctness,}$$

with $\varepsilon > q_E q_P 2^{-\gamma}$.

Proof. If $\mathbf{A}$ wins the Adversarial Replay-Correctnessgame for $\mathsf{PKEBC}_{\mathsf{SM}}$ event $\neg\mathsf{aRCorr}$ occurs, that is, $\mathbf{A}$ must have made queries $\mathcal{O}_{Publish}(c)$ and later a query $c \leftarrow \mathcal{O}_E(\cdot,\cdot)$. Let $\mathcal{C}^*$ be the set of ciphertexts $\mathbf{A}$ has submitted to $\mathcal{O}_{Publish}(c)$ before this event occurred. Since each $\mathsf{PKEBC}_{\mathsf{SM}}$-ciphertext contains a mPKE-ciphertext and mPKE is γ-spread, the probability that for a query $c \leftarrow \mathcal{O}_E(\cdot,\cdot)$ we have $c \in \mathcal{C}^*$ is at most $|\mathcal{C}^*|2^{-\gamma} \leq q_P 2^{-\gamma}$. Since $\mathbf{A}$ makes q_E at most queries to $\mathcal{O}_E$, the probability that a query to $\mathcal{O}_E(\cdot,\cdot)$ returns a value in $\mathcal{C}^*$ is thus at most $q_E q_P 2^{-\gamma}$.

C.4 Public-Key Collision Resistance

Theorem 9. *If* mPKE *is $2n$-Party ε'-Public-Key Collision Resistant then* $\mathsf{PKEBC}_{\mathsf{SM}}$ *is n-Party ε-Public-Key Collision Resistant with $\varepsilon \leq \varepsilon'$.*

Proof. Since $\mathsf{PKEBC}_{\mathsf{SM}}$'s public keys consist of two mPKE keys, we have

$$\begin{aligned}
&\Pr\left[\left|\{\mathtt{pk}_1,\ldots,\mathtt{pk}_n\}\right| < n \;\middle|\; \begin{array}{c}\mathtt{pp} \leftarrow \mathsf{PKEBC}_{\mathsf{SM}}.\mathsf{Setup}(1^k) \\ (\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow \mathsf{PKEBC}_{\mathsf{SM}}.\mathsf{Gen}(\mathtt{pp}), i \in [n]\end{array}\right] \\
\leq &\Pr\left[\left|\{\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1}\}\right| < 2n \;\middle|\; \begin{array}{c}(\mathtt{pk}_{i,j}, \mathtt{sk}_{i,j}) \leftarrow \mathsf{mPKE}.\mathsf{Gen}(1^k) \\ i \in [n], j \in \{0,1\}\end{array}\right] \\
\leq &\varepsilon
\end{aligned}$$

□

C.5 Self-Consistency

Theorem 10. $\mathsf{PKEBC}_{\mathsf{SM}}$ *is Self-Consistent.*

Proof. $\mathsf{PKEBC}_{\mathsf{SM}}$ is Self-Consistent, because $\mathsf{PKEBC}_{\mathsf{SM}}$'s Dec algorithm returns $\perp$ if $\mathtt{pk} \neq v_i$ (see Algorithm 1). □

C.6 Consistency

Theorem 11. *If* mPKE *is perfectly correct,* PRG *is $(t, \varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}})$-left collision resistant and* NIZK *is $(\cdot, \varepsilon_{\mathsf{Sound}}, \cdot, \cdot, t, 0, q_V)$-secure then no adversary* $\mathbf{A}$ *(ε', t')-breaks* $\mathsf{PKEBC}_{\mathsf{SM}}$*'s*

$$(\cdot, \cdot, \cdot, q_V, \cdot)\text{-}\textit{Consistency,}$$

with $\varepsilon' > \varepsilon_{\mathsf{Sound}} + \varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}}$ and $t \approx t'$.

Proof. We first describe the reduction. The reduction simulates the oracle $\mathcal{O}_{PP}$ by calling the $\mathcal{O}_S$ oracle of the soundness game of NIZK to get $\mathtt{crs}$ and returning $(1^k, \mathtt{crs})$. The oracles $\mathcal{O}_{SK}$ and $\mathcal{O}_{PK}$, $\mathcal{O}_E$, and $\mathcal{O}_D$ are simulated as in the real game. In each $\mathcal{O}_D$ query, the reduction makes a $\mathcal{O}_V((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi)$ query for the soundness game of NIZK, where $\vec{u}$, c_{mPKE}, and π are obtained by running Dec.

If $\mathbf{A}$ wins the Consistency game, then it must have made a $\mathcal{O}_D(B_j, c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$ query that was answered with $(\vec{v}, m) \neq \perp$ and another $\mathcal{O}_D(B_i, c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$ query with $\mathtt{pk}_i \in \vec{v}$ that did output either $\perp$ or $(\vec{v}, m) \neq \perp$. If $(\vec{u}, c_{\mathsf{mPKE}}) \notin L^{\mathsf{PKEBC}}_{\mathsf{Cons}}$, the reduction wins the soundness game of NIZK.

For $(\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}$ and any $\hat{\mathtt{sk}}_1$ $\hat{\mathtt{sk}}_2$ (not necessarily belonging to a public key in $\vec{v}$), we can get a key $K \neq \perp$ in $\mathsf{Dec}(\hat{\mathtt{sk}}_1, c)$ and a different key $K' \neq K, K' \neq \perp$ during $\mathsf{Dec}(\hat{\mathtt{sk}}_2, c)$ with probability at most $\varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}}$. To see this, note that the for-loop in dec only accepts a key K when decrypting the mPKE ciphertext resulted in R with $(\tau, K) := \mathsf{PRG.F}(R)$. Thus, if this case occurs, we would find R_i and R_j with $(\tau, K) \leftarrow \mathsf{PRG.F}(R_i)$ and $(\tau, K') := \mathsf{PRG.F}(R_j)$ with $K \neq K'$, which breaks the left collision-resistance of PRG.

Finally, we argue that for $(\vec{u}, c_{\mathsf{mPKE}}) \notin L^{\mathsf{PKEBC}}_{\mathsf{Cons}}$ and $(\mathtt{pk}, \mathtt{sk})$ in the image of $\mathsf{mPKE.Gen}(1^k)$ and $\mathtt{pk} \in \vec{u}$, the for-loop in Dec will reach the break condition and thus $K \neq \perp$ after the for-loop. Let j be an index such that $\mathtt{pk} = \vec{u}_j$ By perfect correctness of mPKE, we have $R \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, j))$ and $(\tau, K) \leftarrow \mathsf{PRG.F}(R)$. Thus, at latest in the j-th iteration of the for-loop, $\mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$ returns a value different than $\perp$.

Everything in Dec after the for-loop is independent of the specific secret key $(b, \mathtt{sk}_b)$ provided as input and only depends on K. Since Dec is also deterministic, Dec will return in identical values. Thus, every adversary winning the consistency game can be reduced either to an adversary winning the soundness game for NIZK or the left collision resistance game for PRG.

C.7 Adversarial Robustness

Theorem 12. *If* PRG *is* $\varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}}$*-left collision resistant then no adversary* $\mathbf{A}$ $(\varepsilon_{\mathsf{L\text{-}Coll\text{-}Res}})$*-breaks* $\mathsf{PKEBC}_{\mathsf{SM}}$*'s Adversarial Robustness.*

Proof. If $\mathbf{A}$ wins the Adversarial Robustness game, it must have made queries

$$\begin{aligned} \perp \neq (\vec{v}, m) &\leftarrow \mathcal{O}_D(B_i, c \coloneqq (c_{\mathsf{mPKE}}, \tau)) \\ \perp \neq (\vec{v}', m') &\leftarrow \mathcal{O}_D(B_j, c), \text{ where } \mathcal{O}_{PK}(B_j) = \mathtt{pk}_j \notin \vec{v}. \end{aligned}$$

Let $(\mathtt{sk}_i, b_i)$ and $(\mathtt{sk}_j, b_j)$ be the secret keys belonging to B_i and B_j, respectively. By the definition of PKEBC's Dec, there must be i^* and j^* such that

$$\begin{aligned} c_{i^*} &\leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2i^* - 1 + b_{i^*}), \\ R_i &\leftarrow \mathsf{mPKE.Dec}(sk_i, c_{i^*}), \\ (\tau, K_i) &\leftarrow \mathsf{PRG.F}(R_i) \end{aligned}$$

and

$$\begin{aligned} c_{j^*} &\leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2j^* - 1 + b_{j^*}), \\ R_j &\leftarrow \mathsf{mPKE.Dec}(sk_j, c_{j^*}), \\ (\tau, K_j) &\leftarrow \mathsf{PRG.F}(R_j). \end{aligned}$$

As $\mathtt{pk}_j \notin \vec{v}$, we have $\vec{v} \neq \vec{v}'$ and $\mathsf{SKE.Dec}(c_{\mathsf{SKE}}, K_i) \neq \mathsf{SKE.Dec}(c_{\mathsf{SKE}}, K_j)$. Since both $\mathsf{SKE.Dec}$ and $\mathsf{PRG.F}$ are deterministic, we must have $K_i \neq K_j$ and, consequently, $R_i \neq R_j$. (R_i, R_j) is a left-collision for $\mathsf{PRG.F}$. □

C.8 {IND, IK}-CCA$^{\mathsf{adap}}$

Table 1: Overview of the hybrids for proving the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ security of $\mathsf{PKEBC}_{\mathsf{SM}}$ (Theorem 13). Shaded cells indicate that the corresponding aspect of the game is unchanged from the previous row. Columns: NIZK: whether the $\mathtt{crs}$ (from $\mathcal{O}_{PP}$) and the proofs π (in $\mathcal{O}_E$ queries) are real (Real) or simulated (Sim); $\mathcal{O}_E\colon c_{\mathsf{mPKE}}$: how the mPKE ciphertext is produced in an $\mathcal{O}_E$ query; $\mathcal{O}_E\colon c_{\mathsf{SKE}}$: the plaintext that is symmetrically encrypted under the key K; $\mathcal{O}_D$ (fresh): how decryption queries whose c_{mPKE} was *not* output by a previous $\mathcal{O}_E$ query are answered; $\mathcal{O}_D$ (patched): how decryption queries whose c_{mPKE} *was* output by a previous $\mathcal{O}_E$ query are answered; $\mathcal{O}_{SK}$: the secret key returned for a corruption query $\mathcal{O}_{SK}(B_j)$. Transition: the assumption used for the transition from the previous row and the resulting difference in advantage.

Hybrid	NIZK	$\mathcal{O}_E\colon c_{\mathsf{mPKE}}$	$\mathcal{O}_E\colon c_{\mathsf{SKE}}$	$\mathcal{O}_D$ (fresh)	$\mathcal{O}_D$ (patched)	$\mathcal{O}_{SK}$	Transition
$\mathbf{G}_\beta^{\mathsf{CCA}}$	Real	$c_{\mathsf{mPKE}} \leftarrow \mathsf{Enc}(\vec{u}; R)$ (one encryption to all $2n$ keys)	$\mathsf{Enc}(K, (\vec{v}_\beta, m_\beta, \pi))$	dec. with $\mathtt{sk}_b$ (extract slot $2j{-}1{+}b$)	dec. with $\mathtt{sk}_b$	$(b_j, \mathtt{sk}_{j,b_j})$	Initial
$\mathbf{G}_\beta^1$	Sim						$\leq \varepsilon_{\mathsf{ZK}}$ NIZK ZK
$\mathbf{G}_\beta^2$		$c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{i,1-b_i})_i; R)$, then Extend to $(\mathtt{pk}_{i,b_i})_i$ with message R					$= 0$ Extend correctness
$\mathbf{G}_\beta^3$		$c' \leftarrow \mathsf{Enc}((\mathtt{pk}'_i)_i; R')$, then Extend to $(\mathtt{pk}_{i,b_i})_i$ with message R					$\leq \varepsilon_{\mathsf{CPA}}$ mPKE $\{\mathsf{IND}, \mathsf{IK}\}$-CPA
$\mathbf{G}_\beta^4$		$c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{i,b_i})_i; R)$, then Extend to $(\mathtt{pk}'_i)_i$ with message R'					$= 0$ Extend commutativity
$\mathbf{G}_\beta^5$				dec. with $\mathtt{sk}_{1-b}$ (extract slot $2j{-}b$)			$\leq \varepsilon_{\mathsf{SS}}$ NIZK SS
$\mathbf{G}_\beta^6$					look up stored R; $(\tau, K) \leftarrow \mathsf{PRG.F}(R)$		$= 0$ mPKE correctness
$\mathbf{G}_\beta^7$						$(1 - b_j, \mathtt{sk}_{j,1-b_j})$	$= 0$ b_j independent of view
$\mathbf{G}_\beta^8$		$c' \leftarrow \mathsf{Enc}((\mathtt{pk}''_i)_i; R')$, then Extend to $(\mathtt{pk}'_i)_i$ with message R'					$\leq \varepsilon_{\mathsf{CPA}}$ mPKE $\{\mathsf{IND}, \mathsf{IK}\}$-CPA
$\mathbf{G}_\beta^9$		$c_{\mathsf{mPKE}} \leftarrow \mathsf{Enc}((\tilde{\mathtt{pk}}_i)_{i\in[2n]}; R')$					$= 0$ Extend correctness
$\mathbf{G}_\beta^{10}$					look up stored R; (τ, K) uniform per R		$\leq \varepsilon_{\mathsf{PRG}}$ PRG pseudorandomness
$\mathbf{G}_\beta^{11}$			$\mathsf{Enc}(K, m)$, $m \xleftarrow{\$} \mathcal{M}$ (random)				$\leq \varepsilon_{\mathsf{OT\text{-}IND\text{-}CCA}}$ SKE OT-IND-CCA

Theorem 13. *If* mPKE *is* $(0, \varepsilon_{\mathsf{CPA}}, t, 2n, q_E, \mathsf{Extend})$*-secure,* NIZK *is* $(\cdot, \cdot, \varepsilon_{\mathsf{ZK}}, \varepsilon_{\mathsf{SS}}, t, q_E, q_D)$*-secure, and* SKE *is* $(\varepsilon, t, q_E, q_D)$*-secure, and* PRG *is* $(\varepsilon_{\mathsf{PRG}}, t, n)$*-secure, then no adversary* $\mathbf{A}$ (ε', t')*- breaks* $\mathsf{PKEBC}_{\mathsf{SM}}$*'s*

$$(n, d_E, q_E, q_D, \cdot)\text{-}\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}} \text{ security,}$$

with $\varepsilon' > \varepsilon_{\mathsf{ZK}} + \varepsilon_{\mathsf{SS}} + 2\varepsilon_{\mathsf{CPA}} + \varepsilon_{\mathsf{PRG}} + \varepsilon_{\mathsf{OT\text{-}IND\text{-}CCA}}$ *and* $t' \approx t + q_E \cdot t_{\mathsf{Sim}} + q_E \cdot t_{\mathsf{Extend}}$*, where* t_{Sim} *is the runtime of* $\mathsf{NIZK.Sim.Prv}$ *and* t_{Extend} *is the runtime of* Extend*.*

Proof. The proof proceeds via a hybrid argument. The first game $\mathbf{G}^0_{\mathbf{b}}$ is identical to the $\mathbf{G}^{\mathsf{CCA}}_{\mathbf{b}}$ game. The next hybrid game $\mathbf{G}^1_{\mathbf{b}}$ generates $\mathtt{crs}$ via $(\mathtt{crs}, \varphi) \leftarrow \mathsf{NIZK.Sim.Setup}(1^k)$ (instead of $\mathtt{crs} \leftarrow \mathsf{NIZK.Setup}(1^k)$) and in each $\mathcal{O}_E(\vec{V}_0)$ query the proof π is generated as $\pi \leftarrow \mathsf{NIZK.Sim.Prv}_{(\mathtt{crs},\varphi)}((\vec{v}', c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}})$ (instead of $\pi \leftarrow \mathsf{NIZK.Prv}_{\mathtt{crs}}((\vec{v}', c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, w \coloneqq (\mathtt{r}_{\$}, R))$).

$\mathbf{G}^0_{\mathbf{b}} \rightsquigarrow \mathbf{G}^1_{\mathbf{b}}$: Indistinguishability between $\mathbf{G}^0_{\mathbf{b}}$ and $\mathbf{G}^1_{\mathbf{b}}$ can be argued with a straightforward reduction to the zero-knowledge property of NIZK. The reduction simulates the game $\mathbf{G}^1_{\mathbf{b}}$ but uses the oracle $\mathcal{O}_S$ instead of calling $\mathsf{Sim_{zk}.Setup}$ and the oracle $\mathcal{O}_P((\vec{v}, c_{\mathsf{mKEM}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, w \coloneqq (\mathtt{r}_{\$}, R))$ instead of calling $\mathsf{Sim_{zk}.Prv}$. This reduction simulates the game $\mathbf{G}^{\mathbf{b}}_{\mathbf{b}}$ if it is playing the $\mathbf{G}^{\mathsf{ZK}}_{\mathbf{b}}$ game for NIZK and thus

$$\left|\Pr\left[\mathbf{AG}^0_{\mathbf{b}} \Rightarrow 1\right] - \Pr\left[\mathbf{AG}^1_{\mathbf{b}} \Rightarrow 1\right]\right| \leq \varepsilon_{\mathsf{ZK}}.$$

The next hybrid $\mathbf{G}^2_{\mathbf{b}}$ is identical to $\mathbf{G}^1_{\mathbf{b}}$, except that the oracle $\mathcal{O}_E$ is generating the ciphertext c_j as follows. Instead of running $c_{\mathsf{mPKE}} \leftarrow \mathsf{Enc}(\vec{v_j}', R)$ with $((\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}), \ldots, (\mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})) \coloneqq \vec{v_j}$, $\vec{v_j}' \coloneqq (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$, and corresponding secret keys $((b_1, \mathtt{sk}_{1,b_1}), \ldots, (b, \mathtt{sk}_{n,b_n}))$, the game generates $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,1-b_1}, \ldots, \mathtt{pk}_{n,1-b_n}), R)$ and then runs $c_{\mathsf{mPKE}} \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}^{\mathsf{Extend}}, R)$, where $\vec{\mathtt{sk}}^{\mathsf{Extend}}$ is a vector with $2n$ entries, where for each $i \in [n]$ entries $\mathtt{sk}^{\mathsf{Extend}}_{2i}$ and $\mathtt{sk}^{\mathsf{Extend}}_{2i+1}$ are either $\mathtt{sk}_{i,b_i}$ and $\perp$, respectively, if $b_i = 0$, or are $\perp$ and $\mathtt{sk}_{i,b_i}$, respectively, if $b_i = 1$.

$\mathbf{G}^1_{\mathbf{b}} \rightsquigarrow \mathbf{G}^2_{\mathbf{b}}$: The games are identical by definition 38's correctness and thus

$$\Pr\left[\mathbf{AG}^1_{\mathbf{b}} \Rightarrow 1\right] = \Pr\left[\mathbf{AG}^2_{\mathbf{b}} \Rightarrow 1\right].$$

The next hybrid $\mathbf{G}^3_{\mathbf{b}}$ is identical to $\mathbf{G}^2_{\mathbf{b}}$, except that the game samples in each $\mathcal{O}_E$ query a second key $R' \xleftarrow{\$} \mathcal{K}$ and key pairs $(\mathtt{pk}'_i, \mathtt{sk}'_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [n]$. The step $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,1-b_1}, \ldots, \mathtt{pk}_{n,1-b_n}), R)$ is replaced with $c' \leftarrow \mathsf{Enc}((\mathtt{pk}'_1, \ldots, \mathtt{pk}'_n), R')$.

$\mathbf{G}^2_{\mathbf{b}} \rightsquigarrow \mathbf{G}^3_{\mathbf{b}}$: We can use an adversary distinguishing between $\mathbf{G}^2_{\mathbf{b}}$ and $\mathbf{G}^3_{\mathbf{b}}$ to an adversary breaking the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ security of mPKE. The reduction proceeds as the game $\mathbf{G}^2_{\mathbf{b}}$, but generates the ciphertext c_{mPKE} by calling $\mathcal{O}_E((\vec{V}_b, R), (\vec{V}', R'))$ in the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ game for mPKE. Here $\vec{V}'$ is a vector of dummy users that the reduction creates for each $\mathcal{O}_E$ query. Note that the reduction can answer all $\mathcal{O}_D$ queries, because it knows the secret keys $\mathtt{sk}_{i,b_i}$ for $i \in [n]$. Thus

$$\left|\Pr\left[\mathbf{AG}^2_{\mathbf{b}} \Rightarrow 1\right] - \Pr\left[\mathbf{AG}^3_{\mathbf{b}} \Rightarrow 1\right]\right| \leq \varepsilon_{\mathsf{CPA}}.$$

Recall that in $\mathbf{G}^3_{\mathbf{b}}$ the game generates in each $\mathcal{O}_E$ query c_{mPKE} by sampling $c' \leftarrow \mathsf{Enc}((\mathtt{pk}'_1, \ldots, \mathtt{pk}'_n), R')$ and then extending it by running $c_{\mathsf{mPKE}} \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}^{\mathsf{Extend}}, R)$. Game $\mathbf{G}^4_{\mathbf{b}}$ instead samples $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,b_1}, \ldots, \mathtt{pk}_{n,b_n}), R)$ and then extends it by running $c_{\mathsf{mPKE}} \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}'}^{\mathsf{Extend}}, R')$, where $\vec{\mathtt{sk}'}^{\mathsf{Extend}}$ is also a vector with $2n$ entries, where for each $i \in [n]$ entries $\mathtt{sk}'^{\mathsf{Extend}}_{2i}$ and $\mathtt{sk}'^{\mathsf{Extend}}_{2i+1}$ are either $\bot$ and $\mathtt{sk}'_i$, respectively, if $b_i = 0$, or are $\mathtt{sk}'_i$ and $\bot$, respectively, if $b_i = 1$.

$\mathbf{G}^3_{\mathbf{b}} \rightsquigarrow \mathbf{G}^4_{\mathbf{b}}$: The games are identical by definition 38's commutativity and thus

$$\Pr\big[\mathbf{AG}^3_{\mathbf{b}} \Rightarrow 1\big] = \Pr\big[\mathbf{AG}^4_{\mathbf{b}} \Rightarrow 1\big].$$

The next hybrid $\mathbf{G}^5_{\mathbf{b}}$ is identical to $\mathbf{G}^4_{\mathbf{b}}$, except that the game answers $\mathcal{O}_D(B_j, c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$ queries, where c_{mPKE} is not the output of a previous $\mathcal{O}_E$ query, differently. Namely, instead of computing $\mathsf{mPKE.Dec}(\mathtt{sk}_b, c_j)$ (for all $j \in [n]$, $c_j \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2j-1+b)$), the reduction computes $\mathsf{mPKE.Dec}(\mathtt{sk}_{1-b}, c_j)$ (for all $j \in [n]$, $c_j \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, 2j-b)$).

$\mathbf{G}^4_{\mathbf{b}} \rightsquigarrow \mathbf{G}^5_{\mathbf{b}}$: Indistinguishability between $\mathbf{G}^4_{\mathbf{b}}$ and $\mathbf{G}^5_{\mathbf{b}}$ can be argued with a reduction to the simulation soundness of NIZK. More precisely,

$$\Big|\Pr\big[\mathbf{AG}^4_{\mathbf{b}} \Rightarrow 1\big] - \Pr\big[\mathbf{AG}^5_{\mathbf{b}} \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{SS}}.$$

The reduction calls $\mathcal{O}_S$ instead of calling $\mathsf{Sim_{zk}.Setup}$ and the oracle $\mathcal{O}_P((\vec{v}, c_{\mathsf{mKEM}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, w := (\mathtt{r}_\$, R))$ instead of calling $\mathsf{Sim_{zk}.Prv}$. The reduction simulates each $\mathcal{O}_D(B_j, c, \mathsf{Ext}, c_{\mathsf{Ext}})$ query first as in $\mathbf{G}^5_{\mathbf{b}}$ and then as in $\mathbf{G}^4_{\mathbf{b}}$. If the outputs are identical, the reduction answers the query with this output. If the outputs are different, the reduction aborts. In this case, at least for one secret keys $\mathtt{sk}_{j,0}$ or $\mathtt{sk}_{j,1}$ of the user B_j decrypted the submitted mPKE ciphertext to $(K, \bot) \neq \bot$ and $\mathsf{Ext}(\mathtt{pp}, c, c_{\mathsf{Ext}}, K, \bot)$ returned $(\vec{v}, \pi_1, m)$ with $\vec{v}_i = \mathtt{pk}_j$ and $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{v}', c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi_1) = 1$ (where $\vec{v}' := (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$ for $((\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}), \ldots, (\mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})) := \vec{v}$). If $(\vec{v}', c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}$, decrypting the mPKE ciphertext with the other secret key ($\mathtt{sk}_{j,1}$ or $\mathtt{sk}_{j,0}$) would lead to the same result $(K, \bot) \neq \bot$ by perfect correctness of mPKE. Thus, π_1 is in this case a valid proof for a wrong statement, which allows the reduction to win the simulation-soundness game.

The next hybrid $\mathbf{G}^6_{\mathbf{b}}$ is identical to $\mathbf{G}^5_{\mathbf{b}}$, except that the game answers $\mathcal{O}_D(B_j, c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$ queries, where c_{mPKE} is the output of a previous $\mathcal{O}_E$ query, differently. We call such $\mathcal{O}_D$ queries *patched*. In each $\mathcal{O}_E$ query, the game stores the message R and the corresponding resulting ciphertext c_{mPKE}. In $\mathcal{O}_D(B_j, c = (c_{\mathsf{mPKE}}, \tau, c_{\mathsf{SKE}}))$ queries, where c_{mPKE} is the output of a previous $\mathcal{O}_E$ query, the game looks up the key R that was encrypted in c_{mPKE}, instead of trying to decrypt c_{mPKE} with B_j's secret key. This key is then used to compute $(\tau', K) \leftarrow \mathsf{PRG.F}(R)$, $(\vec{v}, m, \pi) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$. If this decryption of c_{SKE} fails, $\tau \neq \tau'$, B_j's public key is not in $\vec{v}$, or $\mathsf{NIZK.Vrfy}_{\mathtt{crs}}((\vec{u}, c_{\mathsf{mPKE}}) \in L^{\mathsf{PKEBC}}_{\mathsf{Cons}}, \pi) = 0$ the oracle returns $\bot$. Otherwise, it returns $(\vec{v}, m)$.

$\mathbf{G}^5_{\mathbf{b}} \rightsquigarrow \mathbf{G}^6_{\mathbf{b}}$: Due to the perfect correctness of mPKE, the games are identical

$$\Pr\big[\mathbf{AG}^5_{\mathbf{b}} \Rightarrow 1\big] = \Pr\big[\mathbf{AG}^6_{\mathbf{b}} \Rightarrow 1\big].$$

In the next hybrid $\mathbf{G}^7_{\mathbf{b}}$, all $\mathcal{O}_{SK}(B_j)$ queries are answered by returning the secret key $(1-b_j, \mathtt{sk}_{j,1-b_j})$, instead of $(b_j, \mathtt{sk}_{j,b_j})$. $\mathbf{G}^6_{\mathbf{b}} \rightsquigarrow \mathbf{G}^7_{\mathbf{b}}$**:**

$$\Pr\big[\mathbf{AG}^6_{\mathbf{b}} \Rightarrow 1\big] = \Pr\big[\mathbf{AG}^7_{\mathbf{b}} \Rightarrow 1\big].$$

Since the adversary is not allowed to make an $\mathcal{O}_E$ query with the user B_j when he queries $\mathcal{O}_{SK}(B_j)$, the bit b_j is independent of all queries except for the $\mathcal{O}_{SK}(B_j)$ query. Thus, the games are distributed identically.

The next hybrid $\mathbf{G}^8_{\mathbf{b}}$ is identical to $\mathbf{G}^7_{\mathbf{b}}$, except that in each $\mathcal{O}_E$ query the step $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,b_1}, \ldots, \mathtt{pk}_{n,b_n}), R)$ is replaced with $c' \leftarrow \mathsf{Enc}((\mathtt{pk}''_1, \ldots, \mathtt{pk}''_n), R')$ for $(\mathtt{pk}''_i, \mathtt{sk}''_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [n]$.

$\mathbf{G}^7_{\mathbf{b}} \rightsquigarrow \mathbf{G}^8_{\mathbf{b}}$**:** We can argue, identically to the transition $\mathbf{G}^2_{\mathbf{b}} \rightsquigarrow \mathbf{G}^3_{\mathbf{b}}$, that

$$\Big|\Pr\big[\mathbf{AG}^7_{\mathbf{b}} \Rightarrow 1\big] - \Pr\big[\mathbf{AG}^8_{\mathbf{b}} \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{CPA}}.$$

Recall that in $\mathbf{G}^8_{\mathbf{b}}$ the game generates in each $\mathcal{O}_E$ query c_{mPKE} by sampling $c' \leftarrow \mathsf{Enc}((\mathtt{pk}''_1, \ldots, \mathtt{pk}''_n), R')$ and then extending it by running $c_{\mathsf{mPKE}} \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}', R')$. In final hybrid $\mathbf{G}^9_{\mathbf{b}}$, the game instead samples directly via $c_{\mathsf{mPKE}} \leftarrow \mathsf{Enc}((\tilde{\mathtt{pk}}_1, \ldots, \tilde{\mathtt{pk}}_{2n}), R')$ for $(\tilde{\mathtt{pk}}_i, \tilde{\mathtt{sk}}_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [2n]$.

$\mathbf{G}^8_{\mathbf{b}} \rightsquigarrow \mathbf{G}^9_{\mathbf{b}}$**:** The games are identical by definition 38's correctness and thus

$$\Pr\big[\mathbf{AG}^8_{\mathbf{b}} \Rightarrow 1\big] = \Pr\big[\mathbf{AG}^9_{\mathbf{b}} \Rightarrow 1\big].$$

In the next hybrid $\mathbf{G}^{10}_{\mathbf{b}}$ we change patched $\mathcal{O}_D$ queries. Recall that in these queries the game $\mathbf{G}^9_{\mathbf{b}}$ computes $(\tau', K) \leftarrow \mathsf{PRG.F}(R)$. In $\mathbf{G}^{10}_{\mathbf{b}}$ we replace this with uniformly random (τ', K) that are sampled once for each R.

$\mathbf{G}^9_{\mathbf{b}} \rightsquigarrow \mathbf{G}^{10}_{\mathbf{b}}$**:** We argue via a reduction to the (n)-pseudorandomness of PRG that

$$\Big|\Pr\big[\mathbf{AG}^9_{\mathbf{b}} \Rightarrow 1\big] - \Pr\big[\mathbf{AG}^{10}_{\mathbf{b}} \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{PRG}}.$$

The reduction is straightforward since each PRG input R is uniformly random and not used anywhere else.

In the final hybrid $\mathbf{G}^{11}_{\mathbf{b}}$, the game generates in $\mathcal{O}_E$ queries the symmetric ciphertext as $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, m)$ for a random message $m \leftarrow \mathcal{M}$ (instead of $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}_b, m_b, \pi))$).

$\mathbf{G}^{10}_{\mathbf{b}} \rightsquigarrow \mathbf{G}^{11}_{\mathbf{b}}$**:** We argue via a reduction to the $\mathsf{OT\text{-}IND\text{-}CCA}$ security of SKE that

$$\Big|\Pr\big[\mathbf{AG}^{10}_{\mathbf{b}} \Rightarrow 1\big] - \Pr\big[\mathbf{AG}^{11}_{\mathbf{b}} \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{OT\text{-}IND\text{-}CCA}}.$$

With probability $1/2$ we have $\mathbf{b} = 0$ and thus the games are identical. If $\mathbf{b} = 1$, we use a reduction to the $\mathsf{OT\text{-}IND\text{-}CCA}$ security of SKE that generates in each $\mathcal{O}_E$ query $c_{\mathsf{SKE}} \leftarrow \mathcal{O}_{\mathsf{Enc}^{\mathbf{b}}}(i, (\vec{v}_b, m_b, \pi))$, where i is the index of the query. In all $\mathcal{O}_D$ queries, where c_{mPKE} is the mPKE ciphertext returned by the i-th $\mathcal{O}_E$ query, c_{SKE} is decrypted by a $\mathcal{O}_D(i, c_{\mathsf{SKE}})$ call to the $\mathsf{OT\text{-}IND\text{-}CCA}$ game.

In the game $\mathbf{G}^{11}_{\mathbf{b}}$ the adversary's view is statistically independent of the challenge bit $\mathbf{b}$ and thus wins this game with probability $1/2$.

D Security Analysis of $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$

D.1 Correctness

Theorem 14. *If* mPKE *is*

$$(\varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, \cdot, \cdot, n^{\mathsf{mPKE}}, \cdot, \cdot, \cdot)\text{-}secure, \tag{D.1}$$

and SKE *is Perfectly Correct then no adversary* $\mathbf{A}$ $(\varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}})$*-breaks* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$*'s*

$$(n \coloneqq n^{\mathsf{mPKE}})\text{-}Correctness,$$

with $\varepsilon > \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}} + q_D \cdot d_E / |\mathcal{T}|$*, where* T *is a random oracles and* $\mathcal{T}$ *is its codomain.*

Proof. $\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{G}^1$**:** In $\mathbf{G}^1$ it is assumed that mPKE's event $\neg\mathsf{Corr}$ (Definition 33, adapted for mPKE schemes) does not occur. A simple reduction to mPKE's oracles (together with assumption D.1) implies

$$\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^{\mathsf{Corr}} \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}.$$

It remains to bound the probability of triggering $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$'s $\neg\mathsf{Corr}$ event in $\mathbf{G}^1$. Recall the definition of event $\neg\mathsf{Corr}$:

1. A query $\mathcal{O}_E(\vec{V}, m)$ outputs ciphertext c;
2. a later query $\mathcal{O}_D(B_j \in \vec{V}, c)$ outputs either $\perp$ or some $(\vec{v}', m')$ with $(\vec{v}, m) \neq (\vec{v}', m')$, where $\vec{v}$ is $\vec{V}$'s corresponding vector of public keys.

Fix a query $\mathcal{O}_E(\vec{V}, m)$ and let $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$ be the ciphertext it outputs. Furthermore, let $\vec{v}$ be the vector of public keys of the parties in vector $\vec{V}$, and let $\tilde{m}_E$ be the plaintext sampled from mPKE's message space during encryption. Consider a decryption query for ciphertext c by a party $B_j \in \vec{V}$, i.e. $\mathcal{O}_D(B_j \in \vec{V}, c)$, and let $\mathtt{pk}_j$ and $\mathtt{sk}_j$ be B_j's public and secret keys, respectively. For each index i that is reached in decryption's **for**-loop, let c_i be the ciphertext resulting from $\mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$ and $\tilde{m}_i$ be the plaintext resulting from $\mathsf{mPKE.Dec}(\mathtt{sk}_j, c_i)$. Also, let $l \in [|\vec{V}|]$ be the least index such that $B_j = V_l$. If c's decryption reaches **for**-loop index l, then the correctness of mPKE implies $\tilde{m}_l = \tilde{m}_E$, since in $\mathbf{G}^1$ mPKE's event $\neg\mathsf{Corr}$ is assumed not to occur.

Proposition 1. *If* $\tilde{m}_i = \tilde{m}_E$ *then event* $\neg\mathsf{Corr}$ *does not occur.*

Proof. Running T or K two (or more) times on the same input always yields the same output. Since $\tilde{m}_i = \tilde{m}_E$, this implies $\mathsf{T}(\tilde{m}_i) = \mathsf{T}(\tilde{m}_E) = \tau$ — and therefore variable $\mathtt{ctr}$ gets set to i — and also that $K = \mathsf{K}(\tilde{m}_i) = \mathsf{K}(\tilde{m}_E)$, where K was the SKE key used by the $\mathcal{O}_E$ query to encrypt $(\vec{v}, m)$, which resulted in c_{SKE}. By assumption SKE is Perfectly Correct (Definition 39), and since we know decryption obtained the correct key K, then B_j's decryption of c_{SKE} results in the same pair $(\vec{v}, m) \neq \perp$ that was encrypted by the $\mathcal{O}_E$ query. By definition of

the encryption algorithm we also have that $|c_{\mathsf{mPKE}}| = n = |\vec{v}|$. Furthermore, event $\neg\mathsf{Corr}$ can only occur if B_j is in the vector $\vec{V}$ input to encryption (i.e. $B_j \in \vec{V}$). Let $\mathtt{pk}_j$ be B_j's public key. In this case we have that $\mathtt{pk}_j \in \vec{v}$, and therefore $\mathcal{O}_D$ does not output $\perp$ due to any of the checks $(\vec{v}, m) \stackrel{?}{=} \perp$, $|\vec{v}| \stackrel{?}{\neq} n$, or $\mathtt{pk}_j \stackrel{?}{\notin} \vec{v}$. It remains to argue that the two remaining checks also pass. Note that for a fixed public-key vector $\vec{v}$ and a fixed message $\tilde{m}$, both V and mPKE's encryption with randomness $\mathsf{H}(\vec{v}, \tilde{m})$ are deterministic, in the sense that two evaluations on the same input yield exactly the same output. We have already established that c_{SKE}'s decryption yields the vector $\vec{v}$ of public keys that corresponds to the vector of parties $\vec{V}$, and that decryption recovers the message $\tilde{m} = \tilde{m}_i = \tilde{m}_E$ that was used during encryption. Since these are the same vector and message that were input to V and to mPKE's encryption during encryption, it follows that $\mathsf{V}(\vec{v}, \tilde{m}) = \mathsf{V}(\vec{v}, \tilde{m}_E) = \tau_{\vec{v}}$ and $\mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}(\vec{v}, \tilde{m})) = c_{\mathsf{mPKE}}$. Therefore, both the $\tau_{\vec{v}}$ check and the c_{mPKE} recomputation check pass. Putting things together, it follows that decryption outputs the correct pair $(\vec{v}, m)$, and therefore so does the $\mathcal{O}_D$ query. □

Note that $B_j \in \vec{V}$ implies that an index l exists with $\tilde{m}_i = \tilde{m}_E$. From Proposition 1 and from the assumption that mPKE's event $\neg\mathsf{Corr}$ does not occur, we have that decryption may only fail if it reaches some **for**-loop index $i < l$ such that $\tilde{m}_i \neq \tilde{m}_E$ but yet $\mathsf{T}(\tilde{m}_i) = \tau = \mathsf{T}(\tilde{m}_E)$. To conclude the proof it suffices to bound the probability for this to occur for any pair of $\mathcal{O}_E$ and $\mathcal{O}_D$ queries.

Recall that T is (modeled as) a random oracle and that $\mathcal{T}$ is T's co-domain. This means that for each input x in T's domain, T samples a value τ_x uniformly at random from $\mathcal{T}$ (and independently of all other assignments) and assigns x to map to this value τ_x. For two fixed values x and $x' \neq x$, the probability T samples the same image for both is

$$\begin{aligned}\Pr[\mathsf{T}(x) = \mathsf{T}(x')] &= \sum_{\tau_x \in \mathcal{T}} \Pr[\mathsf{T}(x) = \mathsf{T}(x') \mid \mathsf{T}(x) = \tau_x] \cdot \Pr[\mathsf{T}(x) = \tau_x] \\ &= \sum_{\tau_x \in \mathcal{T}} \Pr[\mathsf{T}(x') = \tau_x] \cdot \frac{1}{|\mathcal{T}|} \\ &= \frac{1}{|\mathcal{T}|}.\end{aligned}$$

Now consider any query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ such that c was the output of some query $\mathcal{O}_E(\vec{V}, m)$ with $B_j \in \vec{V}$, and let $l \in [|\vec{V}|]$ be the first index such that $B_j = V_l$. For a fixed **for**-loop index $i \in [l-1]$ and $\tilde{m}_i \neq \tilde{m}_E$, the probability that $\tau = \tau_i \coloneqq \mathsf{T}(\tilde{m}_i)$ is $1/|\mathcal{T}|$. By union bound, the probability that for some $i \in [l-1]$ with $\tilde{m}_i \neq \tilde{m}_E$ we have $\tau \in \{\tau_i\}_{i \in [l-1]}$ is at most $l-1/|\mathcal{T}| < |\vec{V}|/|\mathcal{T}|$. Now consider all $\mathcal{O}_E$ queries such that, together with the fixed $\mathcal{O}_D$ query, event $\neg\mathsf{Corr}$ could occur, and recall the assumption that the sum of the lengths of the vectors input to $\mathcal{O}_E$ is at most d_E. By the union bound, the probability that for at least one of these $\mathcal{O}_E$ queries there is a **for**-loop index i such that $\tilde{m}_i \neq \tilde{m}_E$ and $\tau = \tau_i$ is at most $d_E/|\mathcal{T}|$. Conversely, the probability that for none of these

queries there is a **for**-loop index i such that $\tilde{m}_i \neq \tilde{m}_E$ and $\tau = \tau_i$ is at least $1 - {}^{d_E}/_{|\mathcal{T}|}$. Finally, the probability it does not occur for any $\mathcal{O}_D$ query, which implies $\neg\mathsf{Corr}$ event does not occur, is at least $(1 - {}^{d_E}/_{|\mathcal{T}|})^{q_D} \geq 1 - {}^{q_D \cdot d_E}/_{|\mathcal{T}|}$. □

D.2 Adversarial Replay-Correctness

Theorem 15. *Suppose* mPKE *is* γ*-spread. Then no adversary* $\mathbf{A}$ (ε)*-breaks the* $(\cdot, \cdot, q_E, \cdot, \cdot, q_P)$*-Adversarial Replay-Correctness of* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ *with* $\varepsilon \geq {}^{q_E \cdot q_P}/_{2^\gamma}$*, where* H *is a random oracle.*

Proof. Each ciphertext output by $\mathcal{O}_E$ includes an mPKE component $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}(\vec{v}, \tilde{m}))$, where $\tilde{m}$ is freshly and uniformly sampled from $\mathcal{M}$. Since H is a random oracle, the random coins $\mathsf{H}(\vec{v}, \tilde{m})$ used by mPKE's encryption are uniformly distributed. Since mPKE is γ-spread (Definition 37), the resulting c_{mPKE} equals any fixed ciphertext with probability at most $2^{-\gamma}$. As two PKEBC ciphertexts can only be equal only if their mPKE components are equal, it suffices to bound the probability that a c_{mPKE} component produced by an $\mathcal{O}_E$ query collides with the mPKE component of a ciphertext published by the adversary via the $\mathcal{O}_{Publish}$ oracle.

For a set S of mPKE ciphertexts with $|S| \coloneqq s$, the probability that mPKE's encryption produces a ciphertext c_{mPKE} that is not in S is at least $1 - {}^{s}/_{2^\gamma}$, by applying a union bound to the argument from above. For any subset $S' \subseteq S$, the probability that mPKE's encryption produces a ciphertext that is not in S' is at least $1 - {}^{s}/_{2^\gamma}$. For any $n \in \mathbb{N}$, consider any n subsets $S_1, \ldots, S_n$ of S, and suppose one produces n ciphertexts $c^1_{\mathsf{mPKE}}, \ldots, c^n_{\mathsf{mPKE}}$ independently using mPKE's encryption (each with a freshly sampled $\tilde{m}$ and random coins $\mathsf{H}(\vec{v}, \tilde{m})$); the probability that $\forall i \in [n] : c^i_{\mathsf{mPKE}} \notin S_i$ is at least $(1 - {}^{s}/_{2^\gamma})^n \geq 1 - {}^{n \cdot s}/_{2^\gamma}$.

For the i-th $\mathcal{O}_E$ query, we define S_i as the set of ciphertexts that were previously input to $\mathcal{O}_{Publish}$. By assumption, $\mathbf{A}$ makes at most q_E and q_P queries to $\mathcal{O}_E$ and $\mathcal{O}_{Publish}$, respectively. So we can take $s = q_P$ and $n = q_E$, which yields the claimed bound. □

D.3 Replay-Correctness

Theorem 16. *Suppose* mPKE *is* γ*-spread. Then no adversary* $\mathbf{A}$ (ε)*-breaks the* $(\cdot, \cdot, q_E, \cdot, \cdot, \cdot)$*-Replay-Correctness of* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ *with* $\varepsilon \geq {}^{{q_E}^2}/_{2^\gamma}$*, where* H *is a random oracle.*

Proof. The argument establishing Theorem 15 applies, but we pick S_i now as the set of mPKE components of the ciphertexts previously output by the i-th $\mathcal{O}_E$ query. Since $\mathbf{A}$ makes at most q_E queries to $\mathcal{O}_E$, we can take $s = q_E$ and $n = q_E$, which yields the claimed bound. □

D.4 Public-Key Collision Resistance

Theorem 17. *If* mPKE *is* n*-Party* $\varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}$*-Public-Key Collision Resistant then so is* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ *(with the same parameters).*

Proof. Follows directly from mPKE's n-Party $\varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}$-Public-Key Collision Resistance. □

D.5 Self-Consistency

Theorem 18. $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$ *is Self-Consistent.*

Proof. $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$'s decryption algorithm only outputs a valid pair $(\vec{v}, m)$ if the public key $\mathtt{pk}$ of the party decrypting is in vector $\vec{v}$. If this is not the case, decryption always outputs $\perp$. □

D.6 Consistency

Theorem 19. *If* mPKE *is* $(\varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, \cdot, \cdot, n^{\mathsf{mPKE}}, \cdot, \cdot, \cdot)$*-secure and* SKE*'s decryption is a deterministic algorithm then no adversary* $\mathbf{A}$ *that causes random oracle* T *to be queried on at most* q_T *distinct inputs* (ε)*-breaks* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$*'s*

$$(n \coloneqq n^{\mathsf{mPKE}})\text{-}\textit{Consistency},$$

with $\varepsilon > \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}} + q_T^2/|\mathcal{T}|$*, where* T *is a random oracle and* $\mathcal{T}$ *is its codomain.*

Proof. $\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{G}^1$**:** In $\mathbf{G}^1$ it is assumed that adversary $\mathbf{A}$ cannot find a message m, random coins $\mathtt{r}_\$$ and a party B_i — whose public/secret key-pair is $(\mathtt{pk}_i, \mathtt{sk}_i)$ — such that, letting $c = \mathsf{mPKE.Enc}(\mathtt{pk}_i, m; \mathtt{r}_\$)$, we have $m \neq \mathsf{mPKE.Dec}(\mathtt{sk}_i, c)$.[13] Note that if $\mathbf{A}$ could break this assumption, then it could trigger mPKE's event $\neg\mathsf{Corr}$ (see Definition 33, adapted for mPKE schemes). A simple reduction to mPKE's oracles (together with this theorem's assumption) implies

$$\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Cons}\right] - \Pr\left[\mathbf{AG}^{\mathsf{Corr}} \Rightarrow \neg\mathsf{Cons}\right]\right| \leq \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}.$$

It remains to bound the probability of triggering $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$'s $\neg\mathsf{Cons}$ event in $\mathbf{G}^1$. By Definition 7, event $\neg\mathsf{Cons}$ occurs if there is a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ that outputs some pair $(\vec{v}_j, m_j) \neq \perp$ and there is a later query $\mathcal{O}_D(B_i, c)$, with $\mathtt{pk}_i \in \vec{v}_j$, which outputs $\perp$ or some different pair $(\vec{v}_i, m_i) \neq (\vec{v}_j, m_j)$.

Let $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ be the first decryption query made with the ciphertext c that produces a valid output $(\vec{v}_j, m_j) \neq \perp$. Let $\tilde{m}_j$ be the message recovered during B_j's decryption of c (i.e. the value $\tilde{m}_{\mathtt{ctr}}$ on which the **for**-loop stopped) This message satisfies: 1. $\mathsf{T}(\tilde{m}_j) = \tau$; 2. $\tau_{\vec{v}} = \mathsf{V}(\vec{v}_j, \tilde{m}_j)$; and 3. $c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}_j, \tilde{m}_j; \mathsf{H}(\vec{v}_j, \tilde{m}_j))$. Otherwise, B_j's decryption would have output $\perp$. Condition 3 implies that c_{mPKE} is a well-formed mPKE encryption of the message $\tilde{m}_j$ for the public key vector $\vec{v}_j$.

[13] For mPKE schemes with probabilistic decryption it suffices to require that all fresh decryptions of c by B_i result in m.

Suppose there is a later $\mathcal{O}_D(B_i, c)$ query with $\mathtt{pk}_i \in \vec{v}_j$. Let $l \in [|\vec{v}_j|]$ be the smallest index with $\mathtt{pk}_i = (\vec{v}_j)_l$. Since c_{mPKE} is a well-formed mPKE encryption of $\tilde{m}_j$ for $\vec{v}_j$, and no mPKE correctness error occurs in $\mathbf{G}^1$, it follows that if during B_i's decryption of c the **for**-loop reaches index l, then B_i recovers the same message that was used to form c_{mPKE}, because for $c_l \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, l)$ and $\tilde{m}_l \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}_i, c_l)$, we have $\tilde{m}_l = \tilde{m}_j$.

Since the pair $(\vec{v}_j, m_j)$ output by B_j's query is such that $\mathtt{pk}_i \in \vec{v}_j$, we know such an index exists (namely, l). To conclude this proof we consider two possible cases:

- B_i's decryption reaches an index $k \in [|\vec{v}_j|]$ such that $\tilde{m}_k = \tilde{m}_j$;
- B_i's decryption reaches an index $k \in [|\vec{v}_j|]$ with $\tilde{m}_k \neq \tilde{m}_j$ and $\mathsf{T}(\tilde{m}_k) = \mathsf{T}(\tilde{m}_j)$.

The following implies event $\neg\mathsf{Cons}$ does not occur in the first case:

Proposition 2. *If $\tilde{m}_k = \tilde{m}_j$ then event $\neg\mathsf{Cons}$ does not occur.*

Proof. This follows from a variant of Proposition 1's proof. The main difference is that this proof cannot rely on the correctness of the SKE scheme, because c_{SKE} may not be a well-formed ciphertext (i.e. it may not be in the support of SKE's Enc); instead it relies on SKE's deterministic decryption. Since the message $\tilde{m}_k$ recovered by B_i's decryption equals $\tilde{m}_j$, the SKE key computed by B_i is $K = \mathsf{K}(\tilde{m}_k) = \mathsf{K}(\tilde{m}_j)$, which is the same key that B_j's decryption used. Since SKE's decryption is deterministic, B_i's decryption of c_{SKE} yields the same pair $(\vec{v}_j, m_j)$ that B_j's decryption obtained. Finally, the remaining checks in B_i's decryption all pass: $\mathtt{pk}_i \in \vec{v}_j$ by assumption, $\tau_{\vec{v}} = \mathsf{V}(\vec{v}_j, \tilde{m}_k) = \mathsf{V}(\vec{v}_j, \tilde{m}_j)$, and $\mathsf{mPKE.Enc}(\vec{v}_j, \tilde{m}_k; \mathsf{H}(\vec{v}_j, \tilde{m}_k)) = \mathsf{mPKE.Enc}(\vec{v}_j, \tilde{m}_j; \mathsf{H}(\vec{v}_j, \tilde{m}_j)) = c_{\mathsf{mPKE}}$. Here the last two equalities hold because B_j's successful decryption already checked $\tau_{\vec{v}} = \mathsf{V}(\vec{v}_j, \tilde{m}_j)$ and $c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}_j, \tilde{m}_j; \mathsf{H}(\vec{v}_j, \tilde{m}_j))$. To conclude, B_i's decryption also outputs $(\vec{v}_j, m_j)$, and thus $\neg\mathsf{Cons}$ does not occur. □

It only remains to consider the second case: that B_i's decryption query does not reach any index k with $\tilde{m}_k = \tilde{m}_j$ in the decryption's **for**-loop. This can only happen if B_i's decryption reaches an index $k \in [|\vec{v}_j|]$ such that $\mathsf{T}(\tilde{m}_j) = \mathsf{T}(\tilde{m}_k)$ (but yet $\tilde{m}_j \neq \tilde{m}_k$). But this only occurs if the adversary makes two T queries (or causes two T queries by querying another oracle that makes these queries), on distinct inputs which still result in the same output. Since for each input, T samples the respective image uniformly at random from $\mathcal{T}$, this can happen for each pair of queries only with probability $1/|\mathcal{T}|$. A union bound over all (pairs of) distinct T queries yields $q_T^2/|\mathcal{T}|$. □

D.7 Adversarial Robustness

Theorem 20. *No adversary $\mathbf{A}$ that (directly or indirectly) causes oracle V to be queried on at most q_V distinct inputs (ε)-breaks $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$'s Adversarial Robustness with $\varepsilon > q_V^2/|\mathcal{V}|$, where V is a random oracle and $\mathcal{V}$ is its codomain.*

Proof. By Definition 10, event $\neg\mathsf{aRob}$ occurs if there is a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ that outputs some pair $(\vec{v}, m) \neq \perp$ and there is a later query $\mathcal{O}_D(B_i, c)$, with $\mathtt{pk}_i \notin \vec{v}$, which does not output $\perp$. Suppose B_j's decryption of c results in a pair $(\vec{v}_j, m_j) \neq \perp$. We bound the probability that B_i's decryption of c results in some pair $(\vec{v}_i, m_i) \neq \perp$.

Step 1: Show $\vec{v}_i \neq \vec{v}_j$: Note that $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$'s decryption checks if the public key of the decrypting party is in the vector of public keys it will output. In particular, if this check fails decryption outputs $\perp$. This means that if B_i's decryption of c were to succeed, then the resulting vector $\vec{v}_i$ would include B_i's public key, i.e. $\mathtt{pk}_i \in \vec{v}_i$. If there were a prior decryption query by B_j of c and the output of this query were a pair $(\vec{v}_j, m_j) \neq \perp$ such that $\mathtt{pk}_i \notin \vec{v}_j$, then this would mean $\vec{v}_j \neq \vec{v}_i$.

Step 2: Show $\mathsf{V}(\vec{v}_j, \tilde{m}_j) = \mathsf{V}(\vec{v}_i, \tilde{m}_i)$: $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$'s decryption of a ciphertext $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$ only outputs a valid pair $(\vec{v}, m)$ if $\tau_{\vec{v}} = \mathsf{V}(\vec{v}, \tilde{m})$, where $\tilde{m}$ is the plaintext that decryption obtained from c_{mPKE}. Consider B_j's decryption query of ciphertext c, and let $\tilde{m}_j$ be the plaintext it obtained from it c_{mPKE} component. The decryption query $\mathcal{O}_D(B_j, c)$ only outputs a pair $(\vec{v}_j, m_j) \neq \perp$ if $\tau_{\vec{v}} = \mathsf{V}(\vec{v}_j, \tilde{m}_j)$. The same argument can also be applied to B_i's decryption query to show that if B_i's decryption query would output a pair $(\vec{v}_i, m_i) \neq \perp$ then, $\tau_{\vec{v}} = \mathsf{V}(\vec{v}_i, \tilde{m}_i)$, where $\tilde{m}_i$ is the plaintext obtained from B_i's c_{mPKE} component. Hence, $\mathsf{V}(\vec{v}_j, \tilde{m}_j) = \tau_{\vec{v}} = \mathsf{V}(\vec{v}_i, \tilde{m}_i)$.

Putting things together: $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$'s $\neg\mathsf{aRob}$ event only occurs if there are two V queries on inputs $(\vec{v}_j, \tilde{m}_j)$ and $(\vec{v}_i, \tilde{m}_i)$ which are distinct — since $\vec{v}_j \neq \vec{v}_i$ — which satisfy $\mathsf{V}(\vec{v}_j, \tilde{m}_j) = \mathsf{V}(\vec{v}_i, \tilde{m}_i)$.

Since V is a random oracle, it samples the image of each input uniformly at random from $\mathcal{V}$. Any two fixed distinct inputs collide only with probability $1/|\mathcal{V}|$. Considering adversaries that only cause at most q_V queries to random oracle V — either by directly querying V or by querying another oracle that causes these queries — and then using the union bound over all (pairs of) distinct V queries yields the final bound $q_V^2/|\mathcal{V}|$. □

D.8 $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$

Theorem 21. *If* mPKE *has (efficiently sampleable) message space* $\mathcal{M}$ *and is*

$$(\varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}, \cdot, \varepsilon^{\mathsf{mPKE}}_{\mathsf{CPA}^{\mathsf{adap}}}, \varepsilon^{\mathsf{mPKE}}_{\mathsf{PK\text{-}Coll\text{-}Res}}, \gamma^{\mathsf{mPKE}}, t^{\mathsf{mPKE}}, n^{\mathsf{mPKE}}, q_E^{\mathsf{mPKE}}, d_E^{\mathsf{mPKE}}, \cdot)\text{-}\textit{secure,} \tag{D.2}$$

SKE *is*

$$(\varepsilon^{\mathsf{SKE}}_{\mathsf{OT\text{-}IND\text{-}CCA}}, t^{\mathsf{SKE}}, q_E^{\mathsf{SKE}}, q_D^{\mathsf{SKE}})\text{-}\textit{secure,} \tag{D.3}$$

and H, T, V *and* K *are modeled as random oracles — with respective co-domains* $\mathcal{H}$, $\mathcal{T}$, $\mathcal{V}$ *and* $\mathcal{K}$ *— then no adversary* $\mathbf{A}$ (ε, t)*-breaks* $\mathsf{PKEBC}^{\mathsf{FO[mPKE]}}$*'s*

$$\begin{aligned}(&n \coloneqq n^{\mathsf{mPKE}}, d_E \coloneqq d_E^{\mathsf{mPKE}}, q_E \coloneqq \min(q_E^{\mathsf{mPKE}}, q_E^{\mathsf{SKE}}),\\ &q_D \coloneqq \min(q_D^{\mathsf{mPKE}}, q_D^{\mathsf{SKE}}), \cdot)\text{-}\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}\ \textit{security,}\end{aligned}$$

with $\varepsilon > q_D \cdot 2^{-\gamma^{\mathsf{mPKE}}} + {q_H}^2 \cdot 2^{-\gamma^{\mathsf{mPKE}}} + {q_D \cdot q_T}/{|\mathcal{T}|} + \varepsilon^{\mathsf{mPKE}}_{\mathsf{PK\text{-}Coll\text{-}Res}} + {q_T}^2/|\mathcal{T}| + \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}} + \varepsilon^{\mathsf{mPKE}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CPA}^{\mathsf{adap}}} + 2 \cdot q_E \cdot \left(q_T + q_H + q_K + q_V/|\mathcal{M}|\right) + \varepsilon^{\mathsf{SKE}}_{\mathsf{OT\text{-}IND\text{-}CCA}}$, *and* $t^{\mathsf{mPKE}}, t^{\mathsf{SKE}} \approx t + t_{\mathsf{CCA}} + (q_H + q_V) \cdot t_{\mathsf{mPKE.Enc}}$, *where* t_{CCA} *is the time to run* $\mathsf{PKEBC}^{\mathsf{FO}[\mathsf{mPKE}]}$*'s* $\mathbf{G}^{\mathsf{CCA}}$ *games (i.e. the oracles), and* $t_{\mathsf{mPKE.Enc}}$ *is* mPKE*'s encryption time.*

Algorithm 13 Game $\mathbf{G}^{\mathsf{CCA}}_{\mathbf{b}}$.

INITIALIZATION
  $\mathsf{Keys} \leftarrow \emptyset$, $\mathsf{Challs} \leftarrow \emptyset$
  $\mathsf{H} \leftarrow \emptyset$, $\mathsf{T} \leftarrow \emptyset$, $\mathsf{V} \leftarrow \emptyset$, $\mathsf{K} \leftarrow \emptyset$

$\mathcal{O}_{Publish}(c)$ ▷ Nothing happens

$\mathcal{O}_{PK}(B_j)$: $(\mathtt{pk}_j, \cdot) \leftarrow \mathcal{O}_{SK}(B_j)$; **return** $\mathtt{pk}_j$

$\mathcal{O}_{SK}(B_j)$: **if** $B_j \notin \mathrm{Keys}$**:** $\mathrm{Keys}[B_j] \leftarrow \mathsf{Gen}(\mathsf{pp})$
  return $\mathrm{Keys}[B_j]$

$\mathcal{O}_{\mathsf{T}}(m)$: **if** $\tilde{m} \notin \mathsf{T}$**:** $\mathsf{T}[\tilde{m}] \xleftarrow{\$} \mathcal{T}$; **return** $\mathsf{T}[\tilde{m}]$
$\mathcal{O}_{\mathsf{K}}(m)$: **if** $m \notin \mathsf{K}$**:** $\mathsf{K}[m] \xleftarrow{\$} \mathcal{K}$; **return** $\mathsf{K}[\tilde{m}]$

$\mathcal{O}_{\mathsf{V}}(\vec{v}, \tilde{m})$: **if** $(\vec{v}, \tilde{m}) \notin \mathsf{V}$**:** $\mathsf{V}[\vec{v}, \tilde{m}] \xleftarrow{\$} \mathcal{V}$
  return $\mathsf{V}[\vec{v}, \tilde{m}]$

$\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$: **if** $(\vec{v}, \tilde{m}) \notin \mathsf{H}$**:** $\mathsf{H}[\vec{v}, \tilde{m}] \xleftarrow{\$} \mathcal{H}$
  return $\mathsf{H}[\vec{v}, \tilde{m}]$

$\mathcal{O}^{\mathbf{b}}_E((\vec{V}_0, m_0), (\vec{V}_1, m_1))$
  $\tilde{m} \xleftarrow{\$} \mathcal{M}$
  $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}_{\mathbf{b}}, \tilde{m}; \mathcal{O}_{\mathsf{H}}(\vec{v}_{\mathbf{b}}, \tilde{m}))$
  $(\tau, \tau_{\vec{v}}, K) \leftarrow (\mathcal{O}_{\mathsf{T}}(\tilde{m}), \mathcal{O}_{\mathsf{V}}(\vec{v}_{\mathbf{b}}, \tilde{m}), \mathcal{O}_{\mathsf{K}}(\tilde{m}))$
  $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}_{\mathbf{b}}, m_{\mathbf{b}}))$
  $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$
  $\mathrm{Challs} \leftarrow \mathrm{Challs} \cup \{c\}$
  return c

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$: $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in \mathrm{Challs}$**: return** $\mathtt{test}$
  $(\cdot, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
  $\mathtt{ctr} \leftarrow \bot$
  for $i = 1, \ldots, n$ **:**
    $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
    $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
    if $\mathcal{O}_{\mathsf{T}}[\tilde{m}_i] = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
  if $\mathtt{ctr} = \bot$**: return** $\bot$
  $\tilde{m} \leftarrow \tilde{m}_{\mathtt{ctr}}$, $K \leftarrow \mathcal{O}_{\mathsf{K}}(\tilde{m})$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathcal{O}_{\mathsf{V}}(\vec{v}, \tilde{m})$**: return** $\bot$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m}))$**:**
    return $\bot$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\bot$

Algorithm 14 Hybrid $\mathbf{G}^1_{\mathbf{b}}$. We use symbol $\diamond$ to denote helper procedures.

INITIALIZATION
$\mathsf{Keys} \leftarrow \emptyset$, $\mathsf{Challs} \leftarrow \emptyset$
$\mathsf{H} \leftarrow \emptyset$, $\mathsf{T} \leftarrow \emptyset$, $\mathsf{V} \leftarrow \emptyset$, $\mathsf{K} \leftarrow \emptyset$
$\mathsf{mPKE\text{-}Ctxts} \leftarrow \emptyset$

$\mathcal{O}_{Publish}(B_j)$ ▷ Nothing happens

$\mathcal{O}_{PK}(B_j)$: $(\mathtt{pk}_j, \cdot) \leftarrow \mathcal{O}_{SK}(B_j)$
return $\mathtt{pk}_j$

$\mathcal{O}_{SK}(B_j)$: **if** $B_j \notin \mathrm{Keys}$: $\mathrm{Keys}[B_j] \leftarrow \mathsf{Gen}(\mathsf{pp})$
return $\mathrm{Keys}[B_j]$

$\mathcal{O}_{\mathsf{T}}(\tilde{m})$: $\mathsf{Samp}_{\tilde{m}}(\tilde{m})$; **return** $\mathsf{T}[\tilde{m}]$
$\mathcal{O}_{\mathsf{K}}(\tilde{m})$: $\mathsf{Samp}_{\tilde{m}}(\tilde{m})$; **return** $\mathsf{K}[\tilde{m}]$
$\mathcal{O}_{\mathsf{V}}(\vec{v}, \tilde{m})$: $\mathsf{Samp}_{\vec{v},\tilde{m}}(\vec{v}, \tilde{m})$; **return** $\mathsf{V}[\vec{v}, \tilde{m}]$
$\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$: $\mathsf{Samp}_{\vec{v},\tilde{m}}(\vec{v}, \tilde{m})$; **return** $\mathsf{H}[\vec{v}, \tilde{m}]$

$\diamond\ \mathsf{Samp}_{\tilde{m}}(\tilde{m})$: **if** $\tilde{m} \notin \mathsf{T}$: $(\mathsf{T}[\tilde{m}], \mathsf{K}[m]) \xleftarrow{\$} \mathcal{T} \times \mathcal{K}$

$\diamond\ \mathsf{Samp}_{\vec{v},\tilde{m}}(\vec{v}, \tilde{m})$
if $(\vec{v}, \tilde{m}) \notin \mathsf{H}$:
$(\mathsf{H}[\vec{v}, \tilde{m}], \mathsf{V}[\vec{v}, \tilde{m}]) \xleftarrow{\$} \mathcal{H} \times \mathcal{V}$
$c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$
$\mathsf{mPKE\text{-}Ctxts}[c_{\mathsf{mPKE}}] \leftarrow (\vec{v}, \tilde{m})$
$\mathsf{Samp}_{\tilde{m}}(\tilde{m})$

$\mathcal{O}^{\mathbf{b}}_E((\vec{V}_0, m_0), (\vec{V}_1, m_1))$
$\tilde{m} \xleftarrow{\$} \mathcal{M}$
$\mathcal{O}_{\mathsf{H}}(\vec{v}_{\mathbf{b}}, \tilde{m})$
$c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}_{\mathbf{b}}, \tilde{m}; \mathsf{H}[\vec{v}_{\mathbf{b}}, \tilde{m}])$
$(\tau, \tau_{\vec{v}}, K) \leftarrow (\mathsf{T}[\tilde{m}], \mathsf{V}[\vec{v}_{\mathbf{b}}, \tilde{m}], \mathsf{K}[\tilde{m}])$
$c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}_{\mathbf{b}}, m_{\mathbf{b}}))$
$c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$
$\mathsf{Challs} \leftarrow \mathsf{Challs} \cup \{c\}$
return c

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$: $n \coloneqq |c_{\mathsf{mPKE}}|$
if $c \in \mathrm{Challs}$: **return** `test`
$(\cdot, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
$\mathtt{ctr} \leftarrow \bot$
for $i = 1, \ldots, n$:
$c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
$\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
$\mathcal{O}_{\mathsf{T}}[\tilde{m}_i]$
if $\mathsf{T}[\tilde{m}_i] = \tau$: $\mathtt{ctr} \leftarrow i$; **break**
if $\mathtt{ctr} = \bot$: **return** $\bot$
$\tilde{m} \leftarrow \tilde{m}_{\mathtt{ctr}}$, $K \leftarrow \mathsf{K}[\tilde{m}]$
$(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$: **return** $\bot$
$\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$: **return** $\bot$
if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$:
return $\bot$
if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$:
return $(\vec{v}, m)$
return $\bot$

Algorithm 15 Hybrid $\mathbf{G}^2_{\mathbf{b}}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** test
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\perp$
  $(\cdot, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
  $\mathtt{ctr} \leftarrow \perp$
  for $i = 1, \ldots, n$ **:**
    $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
    $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
    $\mathcal{O}_{\mathsf{T}}[\tilde{m}_i]$
    if $\mathsf{T}[\tilde{m}_i] = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
  if $\mathtt{ctr} = \perp$**: return** $\perp$
  $\tilde{m} \leftarrow \tilde{m}_{\mathtt{ctr}}, \quad K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \perp \vee |\vec{v}| \neq n$**: return** $\perp$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$**: return** $\perp$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$**:**
    return $\perp$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\perp$

Algorithm 16 Hybrid $\mathbf{G}^3_{\mathbf{b}}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** test
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\perp$
  $(\cdot, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
  $\mathtt{ctr} \leftarrow \perp$
  for $i = 1, \ldots, n$ **:**
    $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
    $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
    if $\tilde{m}_i \notin \mathsf{T}$**:**
      if $\mathsf{T}[\tilde{m}_i] = \tau$**:** $\mathtt{ctr} \leftarrow i$**; break**
  if $\mathtt{ctr} = \perp$**: return** $\perp$
  $\tilde{m} \leftarrow \tilde{m}_{\mathtt{ctr}}, \quad K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \perp \vee |\vec{v}| \neq n$**: return** $\perp$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$**: return** $\perp$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$**:**
    return $\perp$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\perp$

Algorithm 17 Hybrid $\mathbf{G}^5_{\mathbf{b}}$.

$\diamond\ \mathsf{Samp}_{\tilde{m}}(\tilde{m})$
  if $\tilde{m} \notin \mathsf{T}$**:** $(\mathsf{T}[\tilde{m}], \mathsf{K}[\tilde{m}]) \xleftarrow{\$} \mathcal{T} \times \mathcal{K}$
  if $\exists \tilde{m}' \neq \tilde{m} : \mathsf{T}[\tilde{m}'] = \mathsf{T}[\tilde{m}]$**:** Abort

Algorithm 18 Hybrid $\mathbf{G}^6_{\mathbf{b}}$.

$\diamond\ \mathsf{Samp}_{\vec{v}, \tilde{m}}(\vec{v}, \tilde{m})$
  if $(\vec{v}, \tilde{m}) \notin \mathsf{H}$**:**
    $(\mathsf{H}[\vec{v}, \tilde{m}], \mathsf{V}[\vec{v}, \tilde{m}]) \xleftarrow{\$} \mathcal{H} \times \mathcal{V}$
    $c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$
    if $c_{\mathsf{mPKE}} \in$ mPKE-Ctxts**:** Abort
    $\mathsf{mPKE\text{-}Ctxts}[c_{\mathsf{mPKE}}] \leftarrow (\vec{v}, \tilde{m})$
  $\mathsf{Samp}_{\tilde{m}}(\tilde{m})$

Algorithm 19 Hybrid $\mathbf{G}_{\mathbf{b}}^{7}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** `test`
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\bot$
  $(\vec{v'}, \tilde{m}) \leftarrow$ mPKE-Ctxts$[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau$**: return** $\bot$
  $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\cdot, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
  `ctr` $\leftarrow \bot$
  for $i = 1, \ldots, n$ **:**
    $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
    $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
    if $\tilde{m}_i = \tilde{m}$**:** `ctr` $\leftarrow i$**; break**
  if `ctr` $= \bot$**: return** $\bot$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$**: return** $\bot$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$**:**
    return $\bot$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\bot$

Algorithm 20 Hybrid $\mathbf{G}_{\mathbf{b}}^{8}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** `test`
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\bot$
  $(\vec{v'}, \tilde{m}) \leftarrow$ mPKE-Ctxts$[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau$**: return** $\bot$
  $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathrm{Keys}[B_j]$
  `ctr` $\leftarrow \bot$
  if $\mathtt{pk}_j = v'_i$ for some $i \in [n]$**:** `ctr` $\leftarrow 1$
  else ▷ $\mathtt{pk}_j \notin \vec{v'}$
    for $i = 1, \ldots, n$ **:**
      $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$
      $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$
      if $\tilde{m}_i = \tilde{m}$**:** `ctr` $\leftarrow i$**; break**
  if `ctr` $= \bot$**: return** $\bot$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$**: return** $\bot$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$**:**
    return $\bot$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\bot$

Algorithm 21 Hybrid $\mathbf{G}_{\mathbf{b}}^{9}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** `test`
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\bot$
  $(\vec{v'}, \tilde{m}) \leftarrow$ mPKE-Ctxts$[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau$**: return** $\bot$
  $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\mathtt{pk}_j, \cancel{\mathtt{sk}_j}) \leftarrow \mathrm{Keys}[B_j]$
  if $\mathtt{pk}_j \notin \vec{v'}$**: return** $\bot$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$
  if $\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$**: return** $\bot$
  if $c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$**:**
    return $\bot$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\bot$

Algorithm 22 Hybrid $\mathbf{G}_{\mathbf{b}}^{10}$.

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in$ Challs**: return** `test`
  if $c_{\mathsf{mPKE}} \notin$ mPKE-Ctxts**: return** $\bot$
  $(\vec{v'}, \tilde{m}) \leftarrow$ mPKE-Ctxts$[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v'}, \tilde{m}]$**: return** $\bot$
  $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\mathtt{pk}_j, \cdot) \leftarrow \mathrm{Keys}[B_j]$
  if $\mathtt{pk}_j \notin \vec{v'}$**: return** $\bot$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \bot \vee |\vec{v}| \neq n$**: return** $\bot$
  if $\vec{v} \neq \vec{v'}$**: return** $\bot$
  ~~$\mathcal{O}_{\mathsf{H}}(\vec{v}, \tilde{m})$~~
  if ~~$\mathtt{pk} \notin \vec{v} \vee \tau_{\vec{v}} \neq \mathsf{V}[\vec{v}, \tilde{m}]$~~**: return** $\bot$
  if ~~$c_{\mathsf{mPKE}} \neq \mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$~~**:**
    ~~**return** $\bot$~~
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\bot$

Algorithm 23 Hybrid $\mathbf{G}_{\mathbf{b}}^{10}$.

$\mathcal{O}_E^{\mathbf{b}}((\vec{V}_0, m_0), (\vec{V}_1, m_1))$ ▷ Unchanged
  $\tilde{m} \xleftarrow{\$} \mathcal{M}$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}_{\mathbf{b}}, \tilde{m})$
  $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}_{\mathbf{b}}, \tilde{m}; \mathsf{H}[\vec{v}_{\mathbf{b}}, \tilde{m}])$
  $(\tau, \tau_{\vec{v}}, K) \leftarrow (\mathsf{T}[\tilde{m}], \mathsf{V}[\vec{v}_{\mathbf{b}}, \tilde{m}], \mathsf{K}[\tilde{m}])$
  $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}_{\mathbf{b}}, m_{\mathbf{b}}))$
  $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$
  $\mathrm{Challs} \leftarrow \mathrm{Challs} \cup \{c\}$
  return c

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in \mathrm{Challs}$**: return** `test`
  if $c_{\mathsf{mPKE}} \notin \mathsf{mPKE}\text{-Ctxts}$**: return** $\perp$
  $(\vec{v'}, \tilde{m}) \leftarrow \mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau \ \vee \ \tau_{\vec{v}} \neq \mathsf{V}[\vec{v'}, \tilde{m}]$**: return** $\perp$
  $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\mathtt{pk}_j, \cdot) \leftarrow \mathrm{Keys}[B_j]$
  if $\mathtt{pk}_j \notin \vec{v'}$**: return** $\perp$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \perp \vee |\vec{v}| \neq n$**: return** $\perp$
  if $\vec{v} \neq \vec{v'}$**: return** $\perp$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\perp$

Algorithm 24 Hybrid $\mathbf{G}_{\mathbf{b}}^{11}$.

Initialization
  $\mathsf{Keys} \leftarrow \emptyset, \quad \mathrm{Challs} \leftarrow \emptyset$
  $\mathsf{H} \leftarrow \emptyset, \quad \mathsf{T} \leftarrow \emptyset, \quad \mathsf{V} \leftarrow \emptyset, \quad \mathsf{K} \leftarrow \emptyset$
  $\mathsf{mPKE}\text{-Ctxts} \leftarrow \emptyset, \quad \mathsf{SKE}\text{-Keys} \leftarrow \emptyset$

$\mathcal{O}_E^{\mathbf{b}}((\vec{V}_0, m_0), (\vec{V}_1, m_1))$
  $\tilde{m_1}, \tilde{m_2} \xleftarrow{\$} \mathcal{M}$
  $\mathcal{O}_{\mathsf{H}}(\vec{v}_{\mathbf{b}}, \tilde{m_1})$
  $c_{\mathsf{mPKE}} \leftarrow \mathsf{mPKE.Enc}(\vec{v}_{\mathbf{0}}, \tilde{m_2}; \mathbf{r}_{\$})$
  $(\tau, \tau_{\vec{v}}, \cancel{K}) \leftarrow (\mathsf{T}[\tilde{m_1}], \mathsf{V}[\vec{v}_{\mathbf{b}}, \tilde{m_1}], \cancel{\mathsf{K}[\tilde{m_1}]})$
  $\mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}] \coloneqq (\vec{v}_{\mathbf{b}}, \tilde{m_1})$
  $\mathsf{SKE}\text{-Keys}[c_{\mathsf{mPKE}}] \coloneqq K \xleftarrow{\$} \mathsf{SKE.Gen}(1^k)$
  $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}(K, (\vec{v}_{\mathbf{b}}, m_{\mathbf{b}}))$
  $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$
  $\mathrm{Challs} \leftarrow \mathrm{Challs} \cup \{c\}$
  return c

$\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$
  $n \coloneqq |c_{\mathsf{mPKE}}|$
  if $c \in \mathrm{Challs}$**: return** `test`
  if $c_{\mathsf{mPKE}} \notin \mathsf{mPKE}\text{-Ctxts}$**: return** $\perp$
  $(\vec{v'}, \tilde{m}) \leftarrow \mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$
  if $\mathsf{T}[\tilde{m}] \neq \tau \ \vee \ \tau_{\vec{v}} \neq \mathsf{V}[\vec{v'}, \tilde{m}]$**: return** $\perp$
  if $c_{\mathsf{mPKE}} \in \mathsf{SKE}\text{-Keys}$**:**
    $K \leftarrow \mathsf{SKE}\text{-Keys}[c_{\mathsf{mPKE}}]$
  else ▷ $c_{\mathsf{mPKE}} \notin \mathsf{SKE}\text{-Keys}$
    $K \leftarrow \mathsf{K}[\tilde{m}]$
  $(\mathtt{pk}_j, \cdot) \leftarrow \mathrm{Keys}[B_j]$
  if $\mathtt{pk}_j \notin \vec{v'}$**: return** $\perp$
  $(\vec{v}, m) \leftarrow \mathsf{SKE.Dec}(K, c_{\mathsf{SKE}})$
  if $(\vec{v}, m) = \perp \vee |\vec{v}| \neq n$**: return** $\perp$
  if $\vec{v} \neq \vec{v'}$**: return** $\perp$
  if $\forall l \in [n], \exists B_i : \mathtt{pk}_i = v_l$**:**
    return $(\vec{v}, m)$
  return $\perp$

Proof. We proceed by a sequence of hybrids.

$\mathbf{G}_{\mathbf{b}}^{\mathsf{CCA}} \rightsquigarrow \mathbf{G}_{\mathbf{b}}^{1}$: Game $\mathbf{G}_{\mathbf{b}}^{1}$ (Algorithm 14) and the original $\mathbf{G}_{\mathbf{b}}^{\mathsf{CCA}}$ game are perfectly indistinguishable. The only difference is that $\mathbf{G}_{\mathbf{b}}^{1}$ relies on two helper functions for sampling random oracle images; these helper procedures essentially sample images for all of the scheme's random oracles as soon as there is a query for a particular input. It follows

$$\left|\Pr\left[\mathbf{AG}_{\mathbf{b}}^{1} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}_{\mathbf{b}}^{\mathsf{CCA}} \Rightarrow \mathtt{win}\right]\right| = 0.$$

$\mathbf{G}_{\mathbf{b}}^{1} \rightsquigarrow \mathbf{G}_{\mathbf{b}}^{2}$: Game $\mathbf{G}^{2}$, defined in Algorithm 15, is just like $\mathbf{G}_{\mathbf{b}}^{1}$ except that now some decryption queries are handled differently. Concretely, in $\mathbf{G}^{2}$ a decryption query for a ciphertext $c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$ whose mPKE component c_{mPKE} has not yet been defined simply output $\perp$. Note that $\mathbf{G}_{\mathbf{b}}^{1}$ and $\mathbf{G}_{\mathbf{b}}^{2}$ are perfectly indistinguishable unless:

1. there is a decryption query for a ciphertext $c := (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$ such that c_{mPKE} is not in mPKE-Ctxts (and therefore $\mathcal{O}_D$ outputs $\perp$);
2. then there is a query $\mathcal{O}_{\mathsf{H}}(\vec{v}, m)$ that outputs some randomness $\mathtt{r}_{\$}$ such that $c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}, m; \mathtt{r}_{\$})$; and
3. later there is a decryption query for ciphertext $c' := (c_{\mathsf{mPKE}}, \tau', \tau'_{\vec{v}}, c'_{\mathsf{SKE}})$ that outputs some non-$\perp$ pair.

From assumption D.2 we know mPKE is γ^{mPKE}-spread (Definition 37). So for a fixed ciphertext c_{mPKE}, the probability that $\mathsf{H}[\vec{v}, m]$ is sampled so that $c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v}, m; \mathsf{H}[\vec{v}, m])$ is at most $2^{-\gamma^{\mathsf{mPKE}}}$. Since by assumption adversary $\mathbf{A}$ makes at most q_D queries to $\mathcal{O}_D$, a union bound then implies

$$\left|\Pr\left[\mathbf{AG}^2_{\mathtt{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^1_{\mathtt{b}} \Rightarrow \mathtt{win}\right]\right| \leq q_D \cdot 2^{-\gamma^{\mathsf{mPKE}}}.$$

$\mathbf{G}^2_{\mathtt{b}} \rightsquigarrow \mathbf{G}^3_{\mathtt{b}}$: Game $\mathbf{G}^3_{\mathtt{b}}$ (Algorithm 16) is just like $\mathbf{G}^2_{\mathtt{b}}$ except that some decryption queries are handled differently. Concretely, for a decryption query $\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$, $\mathcal{O}_D$ no longer checks, for each index $i \in |c_{\mathsf{mPKE}}|$, whether $\tau = \mathsf{T}[\tilde{m}_i]$. Instead, it only makes this check for indices for which $\mathsf{T}[\tilde{m}_i]$ had previously been sampled. Note that the two games are perfectly indistinguishable unless:

1. there is a decryption query $\mathcal{O}_D(B_j, c := (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ such that for some **for**-loop index $i \in |c_{\mathsf{mPKE}}|$, letting $c_i \leftarrow \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i)$ and $\tilde{m}_i \leftarrow \mathsf{mPKE.Dec}(\mathtt{sk}, c_i)$, we have $\tilde{m}_i \in \mathsf{T}$; (and therefore $\mathcal{O}_D$ skips this index); and
2. there is a query $\mathcal{O}_{\mathsf{T}}(\tilde{m}_i)$ that outputs τ.

Since T is a random oracle, the probability this happens for any fixed τ and m for which $\mathsf{T}[m]$ has not yet been sampled is at most $1/|\mathcal{T}|$. By assumption an adversary only makes (or indirectly causes) at most q_T queries to $\mathcal{O}_T$ and makes at most q_D queries to $\mathcal{O}_D$. By applying the union bound over all pairs of $\mathcal{O}_T$ and $\mathcal{O}_D$ queries it then follows $\left|\Pr\left[\mathbf{AG}^3_{\mathtt{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^2_{\mathtt{b}} \Rightarrow \mathtt{win}\right]\right| \leq q_D \cdot q_T / |\mathcal{T}|$.

$\mathbf{G}^3_{\mathtt{b}} \rightsquigarrow \mathbf{G}^4_{\mathtt{b}}$: Game $\mathbf{G}^4_{\mathtt{b}}$ is just like $\mathbf{G}^3_{\mathtt{b}}$ except that it is assumed all parties have distinct public keys. It is easy to see that the two games are perfectly indistinguishable unless there are two parties with the same public key. By assumption $\mathbf{A}$ only queries the oracles it has access to on at most n^{mPKE} different parties and mPKE is n^{mPKE}-Party $\varepsilon^{\mathsf{mPKE}}_{\mathsf{PK\text{-}Coll\text{-}Res}}$-Public-Key Collision Resistant. So: $\left|\Pr\left[\mathbf{AG}^4_{\mathtt{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^3_{\mathtt{b}} \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon^{\mathsf{mPKE}}_{\mathsf{PK\text{-}Coll\text{-}Res}}$.

$\mathbf{G}^4_{\mathtt{b}} \rightsquigarrow \mathbf{G}^5_{\mathtt{b}}$: Game $\mathbf{G}^5_{\mathtt{b}}$ (Algorithm 17) is just like $\mathbf{G}^4_{\mathtt{b}}$ except that now it is assumed that T is an injection, i.e. for every pair of distinct messages m and m', $\mathsf{T}(m) \neq \mathsf{T}(m')$. The games are perfectly indistinguishable unless the adversary causes two T queries on distinct messages m and m' such that $\mathsf{T}(m) \neq \mathsf{T}(m')$. Since by assumption $\mathbf{A}$ only causes at most q_T queries to $\mathcal{O}_{\mathsf{T}}$, a union bound over all pairs of inputs to $\mathcal{O}_{\mathsf{T}}$ implies: $\left|\Pr\left[\mathbf{AG}^5_{\mathtt{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^4_{\mathtt{b}} \Rightarrow \mathtt{win}\right]\right| \leq {q_T}^2 / |\mathcal{T}|$.

$\mathbf{G}^5_{\mathbf{b}} \rightsquigarrow \mathbf{G}^6_{\mathbf{b}}$**:** Game $\mathbf{G}^6_{\mathbf{b}}$ (Algorithm 18) is just like $\mathbf{G}^5_{\mathbf{b}}$ except that now it is assumed that there are no two queries to H that define the same mPKE ciphertext. In other words, it is assumed that for every pair of queries $\mathcal{O}_{\mathsf{H}}(\vec{v}, m)$ and $\mathcal{O}_{\mathsf{H}}(\vec{v'}, m')$ such that $(\vec{v}, m) \neq (\vec{v'}, m')$, the ciphertexts these two queries define are distinct, i.e. $\mathsf{mPKE.Enc}(\vec{v}, m; \mathsf{H}[\vec{v}, m]) \neq \mathsf{mPKE.Enc}(\vec{v'}, m'; \mathsf{H}[\vec{v'}, m'])$. The games are perfectly indistinguishable unless the adversary causes two H queries on distinct pairs $(\vec{v}, m) \neq (\vec{v'}, m')$ such that $\mathsf{mPKE.Enc}(\vec{v}, m; \mathsf{H}[\vec{v}, m]) \neq \mathsf{mPKE.Enc}(\vec{v'}, m'; \mathsf{H}[\vec{v'}, m'])$. Since by assumption $\mathbf{A}$ only causes at most q_H queries to $\mathcal{O}_{\mathsf{H}}$, mPKE's γ^{mPKE}-spreadness plus a union bound over all pairs of inputs to $\mathcal{O}_{\mathsf{H}}$ then implies: $\left|\Pr\left[\mathbf{AG}^5_{\mathbf{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^6_{\mathbf{b}} \Rightarrow \mathtt{win}\right]\right| \leq {q_H}^2/2^{\gamma^{\mathsf{mPKE}}}$.

$\mathbf{G}^6_{\mathbf{b}} \rightsquigarrow \mathbf{G}^7_{\mathbf{b}}$**:** Game $\mathbf{G}^7_{\mathbf{b}}$ (Algorithm 19) is just like $\mathbf{G}^6_{\mathbf{b}}$ except that some decryption queries are handled differently. Consider a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ with $c_{\mathsf{mPKE}} \in \mathsf{mPKE}\text{-Ctxts}$ and let $(\vec{v'}, \tilde{m}) \coloneqq \mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$. Before proceeding to decryption's **for**-loop, in $\mathbf{G}^7_{\mathbf{b}}$ $\mathcal{O}_D$ first checks whether $\mathsf{T}[\tilde{m}]$ matches the ciphertext's τ, and if it does not decryption simply outputs $\perp$. Furthermore, decryption sets the symmetric key K to $\mathsf{K}[\tilde{m}]$ (recall that in $\mathbf{G}^6_{\mathbf{b}}$, K was computed using the $\tilde{m_{\mathtt{ctr}}}$ obtained from decryption's **for**-loop). Then, decryption proceeds to the **for**-loop; consider any index i that decryption's **for**-loop reaches. While $\mathbf{G}^6_{\mathbf{b}}$'s $\mathcal{O}_D$ would check if for that index $\tilde{m}_i \in \mathsf{T}$ and $\mathsf{T}[\tilde{m}_i] = \tau$ — to decide whether to set variable $\mathtt{ctr}$ to i and quit the loop — $\mathbf{G}^7_{\mathbf{b}}$'s $\mathcal{O}_D$ instead simply checks $\tilde{m}_i = \tilde{m}$ — and if this equality hold, it sets variable $\mathtt{ctr}$ to i and quits the loop. It is easy to see that $\mathbf{G}^6_{\mathbf{b}}$ and $\mathbf{G}^7_{\mathbf{b}}$ are perfectly indistinguishable as long as there are no two distinct messages m and m' such that $\mathsf{T}[m] = \mathsf{T}[m']$. This is because if there are no such two messages then $\mathcal{O}_D$ only quits decryption's **for**-loop if it obtains the correct message — i.e. the one to which $\mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$ maps to. However, since game $\mathbf{G}^5_{\mathbf{b}}$ we are assuming there are no two distinct messages with the same T image — which means we are assuming this for both $\mathbf{G}^6_{\mathbf{b}}$ and $\mathbf{G}^7_{\mathbf{b}}$. It follows: $\left|\Pr\left[\mathbf{AG}^7_{\mathbf{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^6_{\mathbf{b}} \Rightarrow \mathtt{win}\right]\right| = 0$.

$\mathbf{G}^7_{\mathbf{b}} \rightsquigarrow \mathbf{G}^8_{\mathbf{b}}$**:** Game $\mathbf{G}^8_{\mathbf{b}}$ (Algorithm 20) is just like $\mathbf{G}^7_{\mathbf{b}}$ except that some decryption queries are handled differently. Consider a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ with $c_{\mathsf{mPKE}} \in \mathsf{mPKE}\text{-Ctxts}$ and let $(\vec{v'}, \tilde{m}) \coloneqq \mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$. $\mathbf{G}^8_{\mathbf{b}}$'s $\mathcal{O}_D$ checks if $\mathtt{pk}_j$ is in this vector $\vec{v'}$. If it is not (i.e. if $\mathtt{pk}_j \notin \vec{v'}$), then it proceeds as before. But if it is, then instead of going through the **for**-loop it simply sets $\mathtt{ctr}$ to 1,[14] which allows decryption to proceed.

The only case in which the two hybrids may differ is if there is a decryption query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ for which $c_{\mathsf{mPKE}} \in \mathsf{mPKE}\text{-Ctxts}$, and for the pair $(\vec{v'}, \tilde{m})$ mapped by c_{mPKE} (i.e. $(\vec{v'}, \tilde{m}) \coloneqq \mathsf{mPKE}\text{-Ctxts}[c_{\mathsf{mPKE}}]$), letting

[14] Value 1 was picked arbitrarily. All that matters is that $\mathtt{ctr}$ is set to a non-$\perp$ value so that decryption can succeed. In particular, note that it does not matter which value $\mathtt{ctr}$ is set to as long as it is not $\perp$.

$n \coloneqq |c_{\mathsf{mPKE}}|$, we have $\mathtt{pk}_j \in \vec{v'}$ but yet there is no index $i \in [n]$ for which

$$\mathsf{mPKE.Dec}(\mathtt{sk}, \mathsf{mPKE.Extract}(c_{\mathsf{mPKE}}, i))$$

output $\tilde{m}$. With a (carefully crafted) reduction we can reduce this event to triggering mPKE's $\neg$Corr event. Such reduction relies on the oracles provided by the underlying mPKE to generate parties' key-pairs and generate ciphertexts (for queries to PKEBC's $\mathcal{O}_E$ oracle). Consider each time that a new ciphertext c_{mPKE} is defined by helper procedure $\mathsf{Samp}_{\vec{v},m}$, and let $(\vec{v'}, \tilde{m})$ be the input given to $\mathsf{Samp}_{\vec{v},m}$. Define $n \coloneqq |\vec{v'}|$. We distinguish two cases:

Case 1: $\forall i \in [n]$ there is a party B_j with $\mathtt{pk}_j = v'_i$. Letting $\vec{V'}$ be the party vector corresponding to public key vector $\vec{v'}$ (note $\vec{V'}$ is unique since we are assuming all parties have distinct public keys), the reduction makes a query $\mathcal{O}_E(\vec{V'}, \tilde{m}; \mathtt{r}_\$)$, where $\mathtt{r}_\$$ is the (mPKE) encryption randomness sampled by $\mathcal{O}_{\mathsf{H}}$.

Case 2: Otherwise, the reduction records, for each such $\mathsf{Samp}_{\vec{v},m}$ call, which public keys of $\vec{v'}$ are no party's public key, together with $\vec{v'}$ and $\tilde{m}$. Then, each time the reduction has to generate a public key for a party B_j that was not queried before:

1. the reduction uses mPKE's underlying $\mathcal{O}_{SK}$ to obtain a key-pair for B_j, say $(\mathtt{pk}_j, \mathtt{sk}_j)$;
2. for each call $\mathsf{Samp}_{\vec{v},m}(\vec{v'}, \tilde{m})$ such that $\mathtt{pk}_j$ matches some public key in $\vec{v'}$ — which was undefined when $\mathsf{Samp}_{\vec{v},m}$ was called — the reduction checks if now, $\forall i \in [n]$ there is a party B_l with $\mathtt{pk}_l = v'_i$. If this holds, the reduction makes a query $\mathcal{O}_E(\vec{V'}, \tilde{m}; \mathsf{H}[\vec{v'}, \tilde{m}])$.

Then, on each decryption query, the reduction proceeds as in $\mathbf{G}^7_{\mathbf{b}}$, but relies on mPKE's underlying $\mathcal{O}_{Ext}$ and $\mathcal{O}_D$ oracles to extract and decrypt the i-th component of the c_{mPKE} component of the ciphertext input to PKEBC's $\mathcal{O}_D$. Since in $\mathbf{G}^7_{\mathbf{b}}$ we are assuming all parties have unique public keys, this means for each such public key vector $\vec{v'}$ there is at most one corresponding party vector $\vec{V'}$. Furthermore, also note that the only event (described above) for which the two hybrids are not perfectly indistinguishable can only occur if the mPKE Correctness reduction made a query $\mathcal{O}_E(\vec{V'}, \tilde{m}; \mathtt{r}_\$ \coloneqq \mathsf{H}[\vec{v'}, \tilde{m}])$ which output some ciphertext c_{mPKE}, and then for each index $i \in [n]$ there was a query $\mathcal{O}_{Ext}(c_{\mathsf{mPKE}}, i)$ that output some c_i plus a query $\mathcal{O}_D(B_j, c_i)$ which did not output $\tilde{m}$. Since $\mathtt{pk}_j \in \vec{v'}$, then $\mathtt{pk}_j = v'_i$ for one such index $i \in [n]$, which means mPKE's $\neg$Corr event occurred. Finally, by assumption Equation D.2 it then follows

$$\left|\Pr\left[\mathbf{AG}^8_{\mathbf{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^7_{\mathbf{b}} \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon^{\mathsf{mPKE}}_{\mathsf{Corr}}.$$

$\mathbf{G}^8_{\mathbf{b}} \rightsquigarrow \mathbf{G}^9_{\mathbf{b}}$: The only difference between the games is that in $\mathbf{G}^9_{\mathbf{b}}$ (Algorithm 21) once again some decryption queries are handled differently. Consider a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ with $c_{\mathsf{mPKE}} \in$ mPKE-Ctxts and let $(\vec{v'}, \tilde{m}) \coloneqq$

mPKE-Ctxts$[c_{\mathsf{mPKE}}]$. In $\mathbf{G}^9_{\mathbf{b}}$, whenever oracle $\mathcal{O}_D$ checks if $\mathtt{pk}_j$ is in vector $\vec{v'}$ but it is not (i.e. if $\mathtt{pk}_j \notin \vec{v'}$), the oracle simply outputs $\perp$. These two hybrids are perfectly indistinguishable. To see why, first note that the only case in which one could possibly distinguish them is if in $\mathbf{G}^8_{\mathbf{b}}$ there were a decryption query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ with $c_{\mathsf{mPKE}} \in \mathsf{mPKE}$-Ctxts and for which, letting $(\vec{v'}, \tilde{m}) \coloneqq \mathsf{mPKE}$-Ctxts$[c_{\mathsf{mPKE}}]$, we had $\mathtt{pk}_j \notin \vec{v'}$ (in which case $\mathbf{G}^9_{\mathbf{b}}$ would always output $\perp$), and yet in $\mathbf{G}^8_{\mathbf{b}}$ the output of this decryption query was not $\perp$; let us say the output was some pair $(\vec{v}, m)$. On one hand, if this were to happen then B_j's decryption must have obtained some symmetric key K with which it successfully decrypted c_{SKE} to the pair $(\vec{v}, m)$ that decryption output. But we also know that this $\vec{v}$ must include B_j's public key, $\mathtt{pk}_j$ because otherwise the decryption query would output $\perp$. But this also means that $\vec{v}$ differs from the vector $\vec{v'}$ that was mapped to by c_{mPKE} (mPKE-Ctxts being the mapping), because $\mathtt{pk}_j \in \vec{v}$ but $\mathtt{pk}_j \notin \vec{v'}$. Now note that if B_j's decryption outputs pair $(\vec{v}, m)$ then it must have successfully recomputed c_{mPKE} as $\mathsf{mPKE.Enc}(\vec{v}, \tilde{m}; \mathsf{H}[\vec{v}, \tilde{m}])$. Finally, note this actually can never happen: starting from hybrid $\mathbf{G}^6_{\mathbf{b}}$ — and including both $\mathbf{G}^8_{\mathbf{b}}$ and $\mathbf{G}^9_{\mathbf{b}}$ — we have been assuming that there are no two pairs $(\vec{v_1}, \tilde{m_1})$ and $(\vec{v_2}, \tilde{m_2})$ such that

$$\mathsf{mPKE.Enc}(\vec{v_1}, \tilde{m_1}; \mathsf{H}[\vec{v_1}, \tilde{m_1}]) = c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v_2}, \tilde{m_2}; \mathsf{H}[\vec{v_2}, \tilde{m_2}]) :$$

if there were, then $\mathsf{Samp}_{\vec{v},m}$ would have already aborted. It then follows

$$\left|\Pr\left[\mathbf{AG}^9_{\mathbf{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^8_{\mathbf{b}} \Rightarrow \mathtt{win}\right]\right| = 0.$$

$\mathbf{G}^9_{\mathbf{b}} \rightsquigarrow \mathbf{G}^{10}_{\mathbf{b}}$: Game $\mathbf{G}^{10}_{\mathbf{b}}$ (Algorithm 22) is just like $\mathbf{G}^9_{\mathbf{b}}$ except that some decryption queries are handled differently. Consider a query $\mathcal{O}_D(B_j, c \coloneqq (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}}))$ with $c_{\mathsf{mPKE}} \in \mathsf{mPKE}$-Ctxts and let $(\vec{v'}, \tilde{m}) \coloneqq \mathsf{mPKE}$-Ctxts$[c_{\mathsf{mPKE}}]$. In $\mathbf{G}^{10}_{\mathbf{b}}$, if the vector $\vec{v}$ obtained from decrypting c_{SKE} differs from the vector $\vec{v'}$ mapped by c_{mPKE} then the game simply outputs $\perp$. ($\tau_{\vec{v}}$ is now also compared against $\mathsf{V}[\vec{v'}, \tilde{m}]$, but assuming $\vec{v'} = \vec{v}$ this check does exactly the same as before.) Once again, note that since hybrid $\mathbf{G}^6_{\mathbf{b}}$ — and including both $\mathbf{G}^9_{\mathbf{b}}$ and $\mathbf{G}^{10}_{\mathbf{b}}$ — we have been assuming that there are no two pairs $(\vec{v_1}, \tilde{m_1})$ and $(\vec{v_2}, \tilde{m_2})$ such that

$$\mathsf{mPKE.Enc}(\vec{v_1}, \tilde{m_1}; \mathsf{H}[\vec{v_1}, \tilde{m_1}]) = c_{\mathsf{mPKE}} = \mathsf{mPKE.Enc}(\vec{v_2}, \tilde{m_2}; \mathsf{H}[\vec{v_2}, \tilde{m_2}]).$$

This implies that if c_{SKE}'s decryption results in some vector $\vec{v} \neq \vec{v'}$ then decryption would output $\perp$ because it would not reconstruct the same c_{mPKE} that is in the ciphertext. This in particular implies that the games are perfectly indistinguishable. We then have

$$\left|\Pr\left[\mathbf{AG}^{10}_{\mathbf{b}} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^9_{\mathbf{b}} \Rightarrow \mathtt{win}\right]\right| = 0.$$

$\mathbf{G}^{10}_{\mathbf{b}} \rightsquigarrow \mathbf{G}^{11}_{\mathbf{b}}$: The only difference between $\mathbf{G}^{10}_{\mathbf{b}}$ and $\mathbf{G}^{11}_{\mathbf{b}}$ — defined in Algorithm 24 — is that in $\mathbf{G}^{11}_{\mathbf{b}}$ for an encryption query $\mathcal{O}^{\mathbf{b}}_E((\vec{V_0}, m_0), (\vec{V_1}, m_1))$, the output

ciphertext's c_{mPKE} component is an encryption of a message $\tilde{m}_2$ — which is independent from both the SKE key K that is used to generate c_{SKE}, from τ and $\tau_{\vec{v}}$ — under the vector of public keys $\vec{v_0}$ corresponding to vector of parties $\vec{V_0}$.

Distinguishing these two hybrids can be reduced to breaking the underlying mPKE's $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}^{\mathsf{adap}}$ security. (The reduction borrows an idea from [36, $\mathsf{IND\text{-}CPA}$ adversary $\mathsf{D} = (\mathsf{D1}, \mathsf{D2})$ Theorem 3.2's proof, Figure 8].) Concretely, a reduction can handle adaptive corruption queries by relying on the mPKE's underlying oracles — because we are assuming an adaptively secure mPKE — and can handle encryption queries $\mathcal{O}_E^{\mathbf{b}}((\vec{V_0}, m_0), (\vec{V_1}, m_1))$ by sampling two messages $\tilde{m}_1$ and $\tilde{m}_2$ uniformly at random from mPKE's message space, and then obtaining c_{mPKE} by making an mPKE challenge query $\mathcal{O}_E^{\mathbf{b}}((\vec{V_{\mathbf{b}}}, \tilde{m}_1), (\vec{V_{\mathbf{0}}}, \tilde{m}_2))$. Both τ would be obtained from $\tilde{m}_1$ — i.e. $\tau = \mathsf{T}[\tilde{m}_1]$ — and $\tau_{\vec{v}}$ would be obtained from $\vec{v_{\mathbf{b}}}$ and $\tilde{m}_1$ — i.e. $\tau_{\vec{v}} = \mathsf{V}[\vec{v_{\mathbf{b}}}, \tilde{m}_1]$. Regarding the c_{SKE} component, the game would sampled a fresh new SKE key K and use it to encrypt $(\vec{v_{\mathbf{b}}}, m_{\mathbf{b}})$. As for decryption queries, note that neither $\mathbf{G}_{\mathbf{b}}^{10}$ or $\mathbf{G}_{\mathbf{b}}^{11}$ makes use of parties' secret keys to handle decryption queries. Then, the reduction tries to use random oracle queries the adversary makes to win the underlying $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}^{\mathsf{adap}}$ game; in case the adversary makes no "useful" random oracle query, the reduction uses the guess bit that the adversary outputs to choose its own guess bit. To be concrete, suppose the adversary queries random oracles T or K on some message m, or queries H or V on an input of the form $(\cdot, m)$: if for some prior $\mathcal{O}_E^{\mathbf{b}}$ query the reduction had sampled two messages $\tilde{m}_1$ and $\tilde{m}_2$, and m matches one of them, say $m = \tilde{m}_l$ for $l \in \{1, 2\}$, then the reduction outputs guess bit $l - 1$. (If there is more than one such prior query, then pick the first one satisfying the condition; if both messages match m, then pick $\tilde{m}_1$.) Letting $b_{\mathbf{A}}$ be the guess bit that the adversary outputs, for $\mathbf{b} \in \{0, 1\}$, the reduction between games $\mathbf{G}_{\mathbf{b}}^{10}$ and $\mathbf{G}_{\mathbf{b}}^{11}$ outputs guess bit $b_{\mathbf{A}} \oplus \mathbf{b}$, i.e. it guesses that it is interacting with $\vec{\mathcal{O}}^{(b_{\mathbf{A}} \oplus \mathbf{b})}_{\mathsf{CPA}^{\mathsf{adap}}}$.

For $\mathbf{b} \in \{0, 1\}$, consider the reduction between games $\mathbf{G}_{\mathbf{b}}^{10}$ and $\mathbf{G}_{\mathbf{b}}^{11}$. Note that if the reduction is interacting with mPKE game $\vec{\mathcal{O}}^{\mathbf{0}}_{\mathsf{CPA}^{\mathsf{adap}}}$ then it perfectly emulates $\mathbf{G}_{\mathbf{b}}^{10}$. On the other hand, if the reduction is instead interacting with $\vec{\mathcal{O}}^{\mathbf{1}}_{\mathsf{CPA}^{\mathsf{adap}}}$ then it perfectly emulates $\mathbf{G}_{\mathbf{b}}^{11}$. Therefore, if the adversary does not make a query to one of the random oracles that makes the reduction immediately outputs a guess bit, the difference of an adversary's winning probabilities against $\mathbf{G}_{\mathbf{b}}^{10}$ and $\mathbf{G}_{\mathbf{b}}^{11}$ is bounded by the advantage of the reduced adversary in distinguishing the two mPKE $\mathsf{CPA}^{\mathsf{adap}}$ games. Finally, suppose that the reduction is interacting with game $\vec{\mathcal{O}}^{\mathbf{b}'}_{\mathsf{CPA}^{\mathsf{adap}}}$; fix some PKEBC $\mathcal{O}_E$ query and let m_1 and m_2 be the messages that the reduction sampled to handle this query. The probability that the adversary makes a query to one of random oracles on the wrong message $m_{2-\mathbf{b}'}$ is at most $q_T + q_H + q_K + q_V / |\mathcal{M}|$ because all the adversary sees is independent from $m_{2-\mathbf{b}'}$. By taking a union bound over all $\mathcal{O}_E$ queries, it follows

$$\left| \Pr\left[\mathbf{AG}_{\mathbf{b}}^{11} \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}_{\mathbf{b}}^{10} \Rightarrow \mathtt{win}\right] \right| \leq \varepsilon^{\mathsf{mPKE}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CPA}^{\mathsf{adap}}} + 2 \cdot q_E \cdot (q_T + q_H + q_K + q_V / |\mathcal{M}|).$$

Finally, we bound the distinguishing advantage of an adversary against games $\mathbf{G}_{\mathbf{0}}^{11}$ and $\mathbf{G}_{\mathbf{1}}^{11}$ via a reduction to the $\mathsf{OT\text{-}IND\text{-}CCA}$ security of the underlying SKE

scheme. The only difference between the two games is how encryption queries are handled. Consider one such query: $\mathcal{O}_E^{\mathbf{b}}((\vec{V}_0, m_0), (\vec{V}_1, m_1))$. For $\mathbf{b} \in \{0,1\}$, $\mathbf{G}_{\mathbf{b}}^{11}$ generates c_{SKE} as an encryption of $(\vec{v}_{\mathbf{b}}, m_{\mathbf{b}})$ under a key K that is sampled freshly and which is information theoretically hidden from the adversary (apart from c_{SKE}). Decryption queries for ciphertexts $c = (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, c_{\mathsf{SKE}})$ such that there was a query to $\mathcal{O}_E$ that output a ciphertext of the form $c = (c_{\mathsf{mPKE}}, \tau, \tau_{\vec{v}}, \cdot)$ can be handled by relying on the decryption oracle provided by the OT-IND-CCA game of SKE. Considering this straightforward reduction $\mathbf{R}$ to the OT-IND-CCA security of SKE, we then have:

$$\begin{aligned}
&\Big|\Pr\big[\mathbf{AG}_{\mathbf{0}}^{11} \Rightarrow \mathtt{win}\big] + \Pr\big[\mathbf{AG}_{\mathbf{1}}^{11} \Rightarrow \mathtt{win}\big] - 1\Big| \\
&= \Big|\Pr\big[(\mathbf{AR})\mathbf{G}_{\mathbf{0}}^{\mathsf{OT\text{-}IND\text{-}CCA[SKE]}} \Rightarrow \mathtt{win}\big] + \Pr\big[(\mathbf{AR})\mathbf{G}_{\mathbf{1}}^{\mathsf{OT\text{-}IND\text{-}CCA[SKE]}} \Rightarrow \mathtt{win}\big] - 1\Big| \\
&= Adv_{\mathsf{SKE}}^{\mathsf{OT\text{-}IND\text{-}CCA}}(\mathbf{AR}) \leq \varepsilon_{\mathsf{OT\text{-}IND\text{-}CCA}}^{\mathsf{SKE}}.
\end{aligned}$$

□

E [10]'s PKEBC Construction

In Algorithm 25 we present the adaptively secure PKEBC construction from [10]. The building blocks are a Non-Interactive Zero-Knowledge NIZK, a Public Key Encryption PKE and a Symmetric Encryption SKE.

In the following, vectors are assumed to have matching lengths; furthermore, to simplify the definition of the relations below, we introduce the following predicate:

$$\begin{aligned}
&\mathsf{CtxtMatch}(\mathtt{pk}_{\mathtt{pp}}, \mathtt{pk}, r_0, r_1, r_2, \alpha, k, c_0, c_1, c_2) \coloneqq \big((c_0, c_1, c_2) = \\
&\qquad \big(\mathsf{PKE.Enc}(\mathtt{pk}_{\mathtt{pp}}, \alpha; r_0), \mathsf{PKE.Enc}(\mathtt{pk}, r_0; r_1), \mathsf{PKE.Enc}(\mathtt{pk}, k; r_2)\big)\big).
\end{aligned} \tag{E.1}$$

Consider relation $R_{\mathsf{PKEBC}}^{\text{Prior-Work}}$ defined as

$$\begin{aligned}
R_{\mathsf{PKEBC}}^{\text{Prior-Work}} \coloneqq \Big\{ &\big((1^k, \mathtt{pk}_{\mathtt{pp}}, c_{\mathtt{pp}}, \vec{c}, c_{\mathsf{SKE}}), (\vec{v}, m, r_{\mathtt{pp}}, \vec{r}, r_{\mathsf{SKE}}, {r_{\mathsf{SKE}}}')\big) \mid \\
&\big(k_{\mathsf{SKE}} = \mathsf{SKE.Gen}(1^k; r_{\mathsf{SKE}})\big) \wedge \big(c_{\mathsf{SKE}} = \mathsf{SKE.Enc}\big(k_{\mathsf{SKE}}, (\vec{v}, m); {r_{\mathsf{SKE}}}'\big)\big) \\
&\wedge \big(c_{\mathtt{pp}} = \mathsf{PKE.Enc}(\mathtt{pk}_{\mathtt{pp}}, m; r_{\mathtt{pp}})\big) \wedge \Big[\bigwedge_{j \in \{1,\ldots,|\vec{c}|\},\, b \in \{0,1\}} \\
&\mathsf{CtxtMatch}(\mathtt{pk}_{\mathtt{pp}}, v_j.\mathtt{pk}_b, r_{j,0}, r_{j,b,1}, r_{j,b,2}, v_j, k_{\mathsf{SKE}}, c_{j,0}, c_{j,b,1}, c_{j,b,2})\Big]\Big\},
\end{aligned} \tag{E.2}$$

where CtxtMatch is as above. In Algorithm 25, we consider the following language:

$$\begin{aligned}
L_{\mathsf{PKEBC}}^{\text{Prior-Work}} \coloneqq \Big\{ &(1^k, \mathtt{pk}_{\mathtt{pp}}, c_{\mathtt{pp}}, \vec{c}, c_{\mathsf{SKE}}) \mid \exists (\vec{v}, m, r_{\mathtt{pp}}, \vec{r}, r_{\mathsf{SKE}}, {r_{\mathsf{SKE}}}') : \\
&\big((1^k, \mathtt{pk}_{\mathtt{pp}}, c_{\mathtt{pp}}, \vec{c}, c_{\mathsf{SKE}}), (\vec{v}, m, r_{\mathtt{pp}}, \vec{r}, r_{\mathsf{SKE}}, {r_{\mathsf{SKE}}}')\big) \in R_{\mathsf{PKEBC}}^{\text{Prior-Work}}\Big\}.
\end{aligned}$$

Algorithm 25 [10]'s PKEBC Construction.

$\mathsf{Setup}(1^k)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{PKE.Gen}(1^k)$
  return $\mathtt{pp} := (1^k, \mathtt{crs} \leftarrow \mathsf{NIZK.Setup}(1^k), \mathtt{pk})$

$\mathsf{Gen}(\mathtt{pp})$
  $(\mathtt{pk}_0, \mathtt{sk}_0) \leftarrow \mathsf{PKE.Gen}(1^k)$
  $(\mathtt{pk}_1, \mathtt{sk}_1) \leftarrow \mathsf{PKE.Gen}(1^k)$
  $b \xleftarrow{\$} \{0,1\}$
  return $\big(\mathtt{pk} := (\mathtt{pk}_0, \mathtt{pk}_1), \mathtt{sk} := (\mathtt{pk}, b, \mathtt{sk}_b)\big)$

$\mathsf{Enc}(\mathtt{pp}, \vec{v} := \big(\mathtt{pk}_1, \ldots, \mathtt{pk}_{|\vec{v}|}\big), m)$
  $c_{\mathsf{pp}} \leftarrow \mathsf{PKE.Enc}\big(\mathtt{pp.pk}, m; \mathtt{r}_{\$}^{\mathsf{pp}}\big)$
  $k_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Gen}\big(1^k; \mathtt{r}_{\$}^{\mathsf{SKE}}\big)$
  $c_{\mathsf{SKE}} \leftarrow \mathsf{SKE.Enc}\big(k_{\mathsf{SKE}}, (\vec{v}, m); \mathtt{r}_{\$}^{\mathsf{SKE}'}\big)$
  for $j \in \{1, \ldots, |\vec{v}|\}$ **:**
    $c_{j,0} \leftarrow \mathsf{PKE.Enc}\big(\mathtt{pp.pk}, v_j; \mathtt{r}_{\$}^{j,0}\big)$
    for $b \in \{0,1\}$ **:**
      $(c_{j,b,1}, c_{j,b,2}) \leftarrow \big(\mathsf{PKE.Enc}(v_j.\mathtt{pk}_b, \mathtt{r}_{\$}^{j,0}; \mathtt{r}_{\$}^{j,b,1}), \mathsf{PKE.Enc}(v_j.\mathtt{pk}_b, k_{\mathsf{SKE}}; \mathtt{r}_{\$}^{j,b,2})\big)$
    $(r_j, c_j) \leftarrow \big((\mathtt{r}_{\$}^{j,0}, \mathtt{r}_{\$}^{j,0,1}, \mathtt{r}_{\$}^{j,0,2}, \mathtt{r}_{\$}^{j,1,1}, \mathtt{r}_{\$}^{j,1,2}), (c_{j,0}, c_{j,0,1}, c_{j,0,2}, c_{j,1,1}, c_{j,1,2})\big)$
  $(\vec{r}, \vec{c}) \coloneqq \big((r_1, \ldots, r_{|\vec{v}|}), (c_1, \ldots, c_{|\vec{v}|})\big)$
  $p \leftarrow \mathsf{NIZK.Prv}\big(\mathtt{crs}, (1^k, \mathtt{pp.pk}, c_{\mathsf{pp}}, \vec{c}, c_{\mathsf{SKE}}) \in L_{\mathsf{PKEBC}}^{\text{Prior-Work}}, (\vec{v}, m, r_{\mathsf{pp}}, \vec{r}, \mathtt{r}_{\$}^{\mathsf{SKE}}, \mathtt{r}_{\$}^{\mathsf{SKE}'})\big)$
  return $(p, c_{\mathsf{pp}}, \vec{c}, c_{\mathsf{SKE}})$

$\mathsf{Dec}(\mathtt{pp}, \mathtt{sk}, c := (p, c_{\mathsf{pp}}, \vec{c}, c_{\mathsf{SKE}}))$
  if $\mathsf{NIZK.Vrfy}\big(\mathtt{crs}, (1^k, \mathtt{pp.pk}, c_{\mathsf{pp}}, \vec{c}, c_{\mathsf{SKE}}) \in L_{\mathsf{PKEBC}}^{\text{Prior-Work}}, p\big) = 1$ **:**
    for $j = 1, \ldots, |\vec{c}|$ **:**
      $r \leftarrow \mathsf{PKE.Dec}(\mathtt{sk.sk}, c_{j,\mathtt{sk}.b,1})$
      if $r \neq \bot \wedge \ \mathsf{PKE.Enc}(\mathtt{pp.pk}, \mathtt{sk.pk}; r) = c_{j,0}$ **:**
        $k_{\mathsf{SKE}} \leftarrow \mathsf{PKE.Dec}(\mathtt{sk.sk}, c_{j,\mathtt{sk}.b,2})$
        return $\mathsf{SKE.Dec}(k_{\mathsf{SKE}}, c_{\mathsf{SKE}})$
  return $\bot$

F Adap[mPKE]'s {IND, IK}-CPA$^{\mathsf{adap}}$ Security

Theorem 22. *If* mPKE *is*

$$(0, \varepsilon_{\mathsf{CPA}}, \cdot, \cdot, \cdot, t, 2n, q_E, 2d_E, \mathsf{Extend})\text{-}\textit{secure} \tag{F.1}$$

then no adversary $\mathbf{A}$ *breaks* Adap[mPKE]*'s*

$$(0, \cdot, \varepsilon_{\mathsf{CPA}^{\mathsf{adap}}}, \cdot, \cdot, t, n, q_E, d_E, \mathsf{Extend})\ \textit{security}$$

with $\varepsilon_{\mathsf{CPA}^{\mathsf{adap}}} > 2\varepsilon_{\mathsf{CPA}}$.

Proof. The proof proceeds via a hybrid argument. The first game $\mathbf{G}_{\mathbf{b}}^0$ is the $\mathbf{G}_{\mathbf{b}}^{\mathsf{CPA}^{\mathsf{adap}}}$ game. In the next hybrid, $\mathbf{G}_{\mathbf{b}}^1$, each $\mathcal{O}_{E^{\mathtt{b}}}((\vec{U_0}, m_0), (\vec{U_1}, m_1))$ query generates ciphertext c as follows: instead of running $c \leftarrow \mathsf{Enc}(\vec{v_b}, m_{\mathbf{b}})$ with $((\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}), \ldots, (\mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})) := \vec{u_b}$, $\vec{v_b} := (\mathtt{pk}_{1,0}, \mathtt{pk}_{1,1}, \ldots, \mathtt{pk}_{n,0}, \mathtt{pk}_{n,1})$, and corresponding secret keys $((b_1, \mathtt{sk}_{1,b_1}), \ldots, (b, \mathtt{sk}_{n,b_n}))$, the game generates $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,1-b_1}, \ldots, \mathtt{pk}_{n,1-b_n}), m_{\mathbf{b}})$ and then runs $c \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}^{\mathsf{Extend}}, m_{\mathbf{b}})$

where $\vec{\mathtt{sk}}^{\mathsf{Extend}}$ is a vector with $2n$ entries, where for each $i \in [n]$ entries $\mathtt{sk}_{2i}^{\mathsf{Extend}}$ and $\mathtt{sk}_{2i+1}^{\mathsf{Extend}}$ are either $\mathtt{sk}_{i,b_i}$ and $\perp$, respectively, if $b_i = 0$, or are $\perp$ and $\mathtt{sk}_{i,b_i}$, respectively, if $b_i = 1$.

$\mathbf{G}_{\mathbf{b}}^0 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^1$**:** The games are identical by definition 38's correctness and thus

$$\Pr\big[\mathbf{AG}_{\mathbf{b}}^0 \Rightarrow 1\big] = \Pr\big[\mathbf{AG}_{\mathbf{b}}^1 \Rightarrow 1\big].$$

The next hybrid $\mathbf{G}_{\mathbf{b}}^2$ is identical to $\mathbf{G}_{\mathbf{b}}^1$, except that the game samples in each $\mathcal{O}_E$ query key pairs $(\mathtt{pk}'_i, \mathtt{sk}'_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [n]$, and the step $c \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,1-b_1}, \ldots, \mathtt{pk}_{n,1-b_n}), m_{\mathbf{b}})$ is replaced with $c \leftarrow \mathsf{Enc}((\mathtt{pk}'_1, \ldots, \mathtt{pk}'_n), m_{\mathbf{1}})$.

$\mathbf{G}_{\mathbf{b}}^1 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^2$**:** We can use an adversary distinguishing between $\mathbf{G}_{\mathbf{b}}^1$ and $\mathbf{G}_{\mathbf{b}}^2$ to an adversary breaking the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ security of mPKE. The reduction proceeds as the game $\mathbf{G}_{\mathbf{b}}^1$, but the ciphertext c is obtained by calling $\mathcal{O}_E((\vec{V}_b, m_{\mathbf{b}}), (\vec{V}', m_{\mathbf{1}}))$ in the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CPA}$ game for mPKE. Here $\vec{V}'$ is a vector of dummy users that the reduction creates for each $\mathcal{O}_E$ query. Note that the reduction can answer all $\mathcal{O}_D$ queries, because it knows the secret keys $\mathtt{sk}_{i,b_i}$ for $i \in [n]$. Thus

$$\Big|\Pr\big[\mathbf{AG}_{\mathbf{b}}^1 \Rightarrow 1\big] - \Pr\big[\mathbf{AG}_{\mathbf{b}}^2 \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{CPA}}.$$

Recall that in $\mathbf{G}_{\mathbf{b}}^2$ the game answers each $\mathcal{O}_E$ query by first computing $c' \leftarrow \mathsf{Enc}((\mathtt{pk}'_1, \ldots, \mathtt{pk}'_n), m_{\mathbf{1}})$ and then extending c' by running $c \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}^{\mathsf{Extend}}, m_{\mathbf{b}})$ where $\vec{\mathtt{sk}}^{\mathsf{Extend}}$ is a vector with $2n$ entries, where for each $i \in [n]$ entries $\mathtt{sk}_{2i}^{\mathsf{Extend}}$ and $\mathtt{sk}_{2i+1}^{\mathsf{Extend}}$ are either $\mathtt{sk}_{i,b_i}$ and $\perp$, respectively, if $b_i = 0$, or are $\perp$ and $\mathtt{sk}_{i,b_i}$, respectively, if $b_i = 1$. In game $\mathbf{G}_{\mathbf{b}}^3$, the game instead samples $c' \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,b_1}, \ldots, \mathtt{pk}_{n,b_n}), m_{\mathbf{b}})$ and then extends it by running $c \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}'}^{\mathsf{Extend}}, m_{\mathbf{1}})$ where $\vec{\mathtt{sk}'}^{\mathsf{Extend}}$ is also a vector with $2n$ entries, where for each $i \in [n]$ entries $\mathtt{sk'}_{2i}^{\mathsf{Extend}}$ and $\mathtt{sk'}_{2i+1}^{\mathsf{Extend}}$ are either $\perp$ and $\mathtt{sk}'_i$, respectively, if $b_i = 0$, or are $\mathtt{sk}'_i$ and $\perp$, respectively, if $b_i = 1$.

$\mathbf{G}_{\mathbf{b}}^2 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^3$**:** The games are identical by definition 38's commutativity; thus

$$\Pr\big[\mathbf{AG}_{\mathbf{b}}^2 \Rightarrow 1\big] = \Pr\big[\mathbf{AG}_{\mathbf{b}}^3 \Rightarrow 1\big].$$

In the next hybrid $\mathbf{G}_{\mathbf{b}}^4$, all $\mathcal{O}_{SK}(B_j)$ queries are answered by returning the secret key $(1 - b_j, \mathtt{sk}_{j,1-b_j})$, instead of $(b_j, \mathtt{sk}_{j,b_j})$. $\mathbf{G}_{\mathbf{b}}^3 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^4$**:**

$$\Pr\big[\mathbf{AG}_{\mathbf{b}}^3 \Rightarrow 1\big] = \Pr\big[\mathbf{AG}_{\mathbf{b}}^4 \Rightarrow 1\big].$$

Since the adversary is not allowed to make an $\mathcal{O}_E$ query with the user B_j when he queries $\mathcal{O}_{SK}(B_j)$, the bit b_j is independent of all queries except for the $\mathcal{O}_{SK}(B_j)$ query. Thus, the games are distributed identically.

The next hybrid $\mathbf{G}_{\mathbf{b}}^5$ is identical to $\mathbf{G}_{\mathbf{b}}^4$, except that in each $\mathcal{O}_E$ query the step $c \leftarrow \mathsf{Enc}((\mathtt{pk}_{1,b_1}, \ldots, \mathtt{pk}_{n,b_n}), m_{\mathbf{b}})$ is replaced with $c \leftarrow \mathsf{Enc}((\mathtt{pk}''_1, \ldots, \mathtt{pk}''_n), m_{\mathbf{1}})$ for $(\mathtt{pk}''_i, \mathtt{sk}''_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [n]$.

$\mathbf{G}_{\mathbf{b}}^4 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^5$**:** We can argue, identically to the transition $\mathbf{G}_{\mathbf{b}}^1 \rightsquigarrow \mathbf{G}_{\mathbf{b}}^2$, that

$$\Big|\Pr\big[\mathbf{AG}_{\mathbf{b}}^4 \Rightarrow 1\big] - \Pr\big[\mathbf{AG}_{\mathbf{b}}^5 \Rightarrow 1\big]\Big| \leq \varepsilon_{\mathsf{CPA}}.$$

Recall that in $\mathbf{G}^5_{\mathbf{b}}$ the game generates in each $\mathcal{O}_E$ query c by sampling $c' \leftarrow \mathsf{Enc}((\mathtt{pk}''_1, \ldots, \mathtt{pk}''_n), m_{\mathbf{1}})$ and then extending it by running $c \leftarrow \mathsf{Extend}(c', \vec{\mathtt{sk}}'^{\mathsf{Extend}}, m_{\mathbf{1}})$. In final hybrid $\mathbf{G}^6_{\mathbf{b}}$, the game instead samples directly via $c \leftarrow \mathsf{Enc}((\tilde{\mathtt{pk}}_1, \ldots, \tilde{\mathtt{pk}}_{2n}), m_{\mathbf{1}})$ for $(\tilde{\mathtt{pk}}_i, \tilde{\mathtt{sk}}_i) \xleftarrow{\$} \mathsf{Gen}(1^k)$ for $i \in [2n]$.

$\mathbf{G}^5_{\mathbf{b}} \rightsquigarrow \mathbf{G}^6_{\mathbf{b}}$: The games are identical by definition 38's correctness and thus

$$\Pr\big[\mathbf{AG}^5_{\mathbf{b}} \Rightarrow 1\big] = \Pr\big[\mathbf{AG}^6_{\mathbf{b}} \Rightarrow 1\big].$$

□

G Security Analysis of mPKE[HPS]

G.1 {IND, IK}-CPA$^{\mathsf{adap}}$

Theorem 23. *If* HPS *is Perfectly Correct (Definition 26) and Pairwise Universal (Definition 31), and no adversary $(\varepsilon_{\mathsf{SM}}, t)$-decides* HPS*'s* (1)*-Subset Membership Problem (Definition 32), then no adversary* $\mathbf{A}$ *(ε, t')-breaks* mPKE[HPS]*'s (n, q_E, d_E)-*{IND, IK}-CPA$^{\mathsf{adap}}$ *security with $\varepsilon > q_E \cdot \varepsilon_{\mathsf{SM}}$, where $t' \approx t$.*

Proof. We assume that every challenge query $\mathcal{O}^{\mathbf{b}}_E((\vec{V}_0, m_0), (\vec{V}_1, m_1))$ satisfies $|\vec{V}_0| = |\vec{V}_1|$ and $|m_0| = |m_1|$. We proceed by a sequence of (hybrid) games, starting with the real game $\mathbf{G}^{\mathsf{CPA}}_{\mathbf{b}}$.

$\mathbf{G}^{\mathsf{CPA}}_{\mathbf{b}} \rightsquigarrow \mathbf{G}^1_{\mathbf{b}}$: Game $\mathbf{G}^1_{\mathbf{b}}$ is just like $\mathbf{G}^{\mathsf{CPA}}_{\mathbf{b}}$, except that each challenge query now computes the hash proofs with the secret keys instead of the public keys and the witness. In more detail, it sets $d_i = m_{\mathbf{b}} \oplus \mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x)$, where $\mathtt{sk}_{\mathbf{b}i}$ is the secret key of the i-th party in $\vec{V}_{\mathbf{b}}$.

The two games are perfectly indistinguishable, because the correctness of the hash proof system implies $\mathsf{HPS.Pub}(v_{\mathbf{b}i}, x, w) = \mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x)$ for every i.

$$\big|\Pr\big[\mathbf{AG}^1_{\mathbf{b}} \Rightarrow \mathtt{win}\big] - \Pr\big[\mathbf{AG}^{\mathsf{CPA}}_{\mathbf{b}} \Rightarrow \mathtt{win}\big]\big| = 0.$$

Next, we introduce a series of hybrids $\mathbf{G}^2_{\mathbf{b}}[l]$, $l \in [0, q_E]$. The game $\mathbf{G}^2_{\mathbf{b}}[l]$ is defined like $\mathbf{G}^1_{\mathbf{b}}$, except that the first l challenge queries are answered differently: Instead of sampling $(x, w) \leftarrow \mathsf{HPS.Samp}^{\in}(\mathtt{pp})$ the game samples $x \leftarrow \mathsf{HPS.Samp}^{\notin}(\mathtt{pp})$ and samples $\vec{d}$ uniformly at random.

Trivially, we have $\mathbf{G}^2_{\mathbf{b}}[0] = \mathbf{G}^1_{\mathbf{b}}$.

$\mathbf{G}^2_{\mathbf{b}}[l] \rightsquigarrow \mathbf{G}^2_{\mathbf{b}}[l+1]$, $l \in [0, q_E - 1]$: We describe a reduction $\mathbf{R}$ to HPS's subset membership problem (Definition 32). $\mathbf{R}$ samples the challenge bit $\mathbf{b}$ itself and generates every party's key pair honestly. It answers $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$ and $\mathcal{O}_{Ext}$ queries honestly. $\mathbf{R}$ answers the challenge queries as follows:

- the first l challenge queries are answered by sampling $x \leftarrow \mathsf{HPS.Samp}^{\notin}(\mathtt{pp})$ and a uniformly random $\vec{d}$;

– the last $q_E - l - 1$ challenge queries are answered by sampling $(x, w) \leftarrow \mathsf{HPS.Samp}^{\in}(\mathtt{pp})$, and setting $d_i = m_{\mathbf{b}} \oplus \mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x)$;
– the $(l+1)$-st challenge query is answered using the subset membership problem challenge: $\mathbf{R}$ queries its oracle $\mathcal{O}_X^{\mathbf{b}'}$ to obtain a statement x, and sets $d_i = m_{\mathbf{b}} \oplus \mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x)$ for each i.

Finally, $\mathbf{R}$ outputs $\mathbf{1}$ if and only if $\mathbf{A}$ correctly guesses $\mathbf{b}$.

If $\mathbf{b}' = \mathbf{0}$ then $(x, w) \leftarrow \mathsf{HPS.Samp}^{\in}(\mathtt{pp})$, so by HPS's Perfect Correctness the $(l+1)$-st query is answered exactly as in $\mathbf{G}^1_{\mathbf{b}}$, and $\mathbf{R}$ perfectly simulates $\mathbf{G}^2_{\mathbf{b}}[l]$. If $\mathbf{b}' = \mathbf{1}$ then $x \leftarrow \mathsf{HPS.Samp}^{\notin}(\mathtt{pp})$. By HPS's Pairwise Universal (Definition 31) $\vec{d}$ is uniformly random in the adversaries view. Note here that $\vec{V}_{\mathbf{b}}$ only contains uncorrupted users and each secret key appears at most once. Moreover, all other challenge queries reveal no information about the secret keys that is not contained in the public key. This holds because the other challenge query either sample $\vec{d}$ uniformly at random or compute it via $d_i = m_{\mathbf{b}} \oplus \mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x)$ for a statement x in HPS's language. In the latter case $\mathsf{HPS.Priv}(\mathtt{sk}_{\mathbf{b}i}, x) = \mathsf{HPS.Pub}(v_{\mathbf{b}i}, x, w)$, and therefore d_i is conditionally independent of $(\mathtt{sk}_{\mathbf{b}})_i$, given $(\mathtt{pk}_{\mathbf{b}})_i$. Consequently, when $\mathbf{b}' = \mathbf{0}$, $\mathbf{R}$ perfectly simulates $\mathbf{G}^2_{\mathbf{b}}[l]$, and when $\mathbf{b}' = \mathbf{1}$, $\mathbf{R}$ perfectly simulates $\mathbf{G}^2_{\mathbf{b}}[l+1]$. Therefore,

$$\left|\Pr\left[\mathbf{AG}^2_{\mathbf{b}}[l] \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}^2_{\mathbf{b}}[l+1] \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon_{\mathsf{SM}}.$$

In the final hybrid $\mathbf{G}^2_{\mathbf{b}}[q_E]$ every challenge query is answered by sampling $x \leftarrow \mathsf{HPS.Samp}^{\notin}(\mathtt{pp})$ and a uniformly random $\vec{d}$ of length $|\vec{V}_{\mathbf{b}}| = |\vec{V}_0| = |\vec{V}_1|$. Thus, $\mathbf{A}$'s view is independent of the challenge bit $\mathbf{b}$, and we have

$$\Pr\left[\mathbf{AG}^2_{\mathbf{b}}[q_E] \Rightarrow \mathtt{win}\right] = 1/2.$$

Putting the hybrid transitions together yields the theorem. □

H Security Analysis of SCBC Instantiation

H.1 Correctness

Theorem 24. *If* PKEBC *is*

$$(\varepsilon^{\mathsf{PKEBC}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, t^{\mathsf{PKEBC}}, n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}secure, \tag{H.1}$$

EUF *is*

$$(\varepsilon^{\mathsf{EUF}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, t^{\mathsf{EUF}}, n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})\text{-}secure, \tag{H.2}$$

and sEUF *is*

$$(\varepsilon^{\mathsf{sEUF}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, t^{\mathsf{sEUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})\text{-}secure, \tag{H.3}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}&(n_S \coloneqq n^{\mathsf{EUF}}, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}},\\ &\ q_E \coloneqq \min(q_E^{\mathsf{PKEBC}}, q_S^{\mathsf{EUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}),\\ &\ q_D \coloneqq \min(q_D^{\mathsf{PKEBC}}, q_V^{\mathsf{EUF}}, q_V^{\mathsf{sEUF}}), q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\mathit{Correctness},\end{aligned}$$

with $\varepsilon > \varepsilon_{\mathsf{Corr}}^{\mathsf{PKEBC}} + \varepsilon_{\mathsf{Corr}}^{\mathsf{EUF}} + \varepsilon_{\mathsf{Corr}}^{\mathsf{sEUF}}$*, and* $t^{\mathsf{PKEBC}}, t^{\mathsf{EUF}}, t^{\mathsf{sEUF}} \approx t + t_{\mathsf{Corr}}$*, where* t_{Corr} *is the time to run* SCBC*'s* $\mathbf{G}^{\mathsf{Corr}}$ *game.*

Proof. We prove this result via game hopping.

$\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{G}^1$: In $\mathbf{G}^1$, for query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$—where c_{SCBC} was output by a query $\mathcal{O}_E(A_i, \vec{V}, m)$—$\mathcal{O}_D$ no longer verifies if σ_{sEUF} is a valid signature on c with respect to $\mathtt{vk}_{\mathsf{sEUF}}$, and instead simply assumes it is.

Distinguishing the two games can be reduced to breaking the correctness of sEUF: since the reduction holds all EUF and PKEBC secret keys and can sign ciphertexts using the $\mathcal{O}_S$ oracle from sEUF's $\mathbf{G}^{\mathsf{Corr}}$ game, it can handle any oracle queries. Since $\mathbf{A}$ only makes at most $q_E \leq \min(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}})$ $\mathcal{O}_E$ queries and $q_D \leq q_V^{\mathsf{sEUF}}$ $\mathcal{O}_D$ queries, it follows from Equation H.3 that no adversary $(t^{\mathsf{sEUF}}, \varepsilon_{\mathsf{Corr}}^{\mathsf{sEUF}})$-breaks the $(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})$-Correctness of sEUF, implying $\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^{\mathsf{Corr}} \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon_{\mathsf{Corr}}^{\mathsf{sEUF}}$.

$\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$: In $\mathbf{G}^2$, for query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$, where c_{SCBC} was output by a query $\mathcal{O}_E(A_i, \vec{V}, m)$ with $B_j \in \vec{V}$, $\mathcal{O}_D$ works as follows: let $(\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})$ be the plaintext encrypted by $\mathsf{PKEBC.Enc}$ under $\vec{v}$ (which resulted in ciphertext c), where $\mathtt{spk}_i$ is A_i's public key and $\vec{v}$ is the vector of PKEBC public keys corresponding to $\vec{V}$, and where $\sigma_{\mathsf{EUF}} \leftarrow \mathsf{EUF.Sign}(\mathtt{ssk}_i.\mathtt{sk}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$ is an EUF signature with $\mathtt{ssk}_i$ being A_i's secret signing key; oracle $\mathcal{O}_D$ no longer decrypts c using $\mathsf{PKEBC.Dec}$ with B_j's PKEBC secret key, and instead simply assumes decryption outputs $(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}}))$.

One can reduce distinguishing the two games to breaking the correctness of PKEBC: since the reduction holds all EUF and sEUF secret keys and also has access to all PKEBC secret keys it needs through the $\mathcal{O}_{SK}$ oracle of PKEBC's Correctness game, it can handle any oracle queries. If $\mathbf{A}$ only queries for at most $n_R \leq n^{\mathsf{PKEBC}}$ different receivers, the sum of lengths of the vectors input to $\mathcal{O}_E$ is at most $d_E \leq d_E^{\mathsf{PKEBC}}$, and $\mathbf{A}$ makes at most $q_E \leq q_E^{\mathsf{PKEBC}}$ and $q_D \leq q_D^{\mathsf{PKEBC}}$ queries to oracles $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, since from Equation (H.1) no adversary $(t^{\mathsf{PKEBC}}, \varepsilon_{\mathsf{Corr}}^{\mathsf{PKEBC}})$-breaks the $(n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})$-Correctness of PKEBC, we have $\left|\Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon_{\mathsf{Corr}}^{\mathsf{PKEBC}}$.

$\mathbf{G}^2 \rightsquigarrow \mathbf{G}^3$: Game $\mathbf{G}^3$ is just like $\mathbf{G}^2$, except that once again some decryption queries work differently. Essentially, in the same way that $\mathbf{G}^2$ differed from $\mathbf{G}^1$—assuming that the ciphertext c in each ciphertext $c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$ output by a query to $\mathcal{O}_E$ decrypts correctly when $\mathcal{O}_D$ is queried on (B_j, c_{SCBC}),

B_j being one of the parties in the vector input to $\mathcal{O}_E$—$\mathbf{G}^3$ differs from $\mathbf{G}^2$ in that it assumes that also each signature σ_{EUF} generated by $\mathcal{O}_E$ using $\mathsf{EUF.Sign}$ also verifies as being valid when $\mathcal{O}_D$ is queried on a matching input. To be more precise, for a query $\mathcal{O}_E(A_i, \vec{V}, m)$, let $(\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})$ be the plaintext that was signed by $\mathsf{EUF.Sign}$ using $\mathtt{ssk}_i$—A_i's secret key—$\vec{v}$ is the vector of PKEBC public keys corresponding to $\vec{V}$, let σ_{EUF} be the resulting signature ($\sigma_{\mathsf{EUF}} \leftarrow \mathsf{EUF.Sign}(\mathtt{ssk}_i, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$), and let c_{SCBC} be the ciphertext output by the $\mathcal{O}_E$ query. Then, when queried on input (B_j, c_{SCBC}) such that $B_j \in \vec{V}$, $\mathcal{O}_D$ no longer verifies if σ_{EUF} is valid by running $\mathsf{EUF.Vrfy}(\mathtt{spk}_i, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$ and instead simply assumes the EUF signature verification outputs 1.

If $\mathbf{A}$ only queries for at most $n_S \leq n^{\mathsf{EUF}}$ different sender keys, makes up to $q_E \leq q_S^{\mathsf{EUF}}$ queries to $\mathcal{O}_E$ and up to $q_D \leq q_V^{\mathsf{EUF}}$ queries to $\mathcal{O}_D$, since from Equation H.2 no adversary $(\varepsilon_{\neg\mathsf{Corr}}^{\mathsf{EUF}}, t^{\mathsf{EUF}})$-breaks the $(n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})$-Correctness of EUF, it follows $\left|\Pr\left[\mathbf{AG}^3 \Rightarrow \neg\mathsf{Corr}\right] - \Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Corr}\right]\right| \leq \varepsilon_{\mathsf{Corr}}^{\mathsf{EUF}}$.

To conclude the proof, note that no adversary can win $\mathbf{G}^3$. □

H.2 Replay-Correctness

Theorem 25. *If* PKEBC *is*

$$\begin{aligned}(\cdot, &\varepsilon_{\mathsf{RCorr}}^{\mathsf{PKEBC}}, \cdot, \cdot, \cdot, \cdot, \varepsilon_{\mathsf{aRob}}^{\mathsf{PKEBC}}, \varepsilon_{\mathsf{Rob}}^{\mathsf{PKEBC}}, \varepsilon_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}^{\mathsf{PKEBC}}, t^{\mathsf{PKEBC}},\\ &n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}\textit{secure},\end{aligned} \tag{H.4}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S \coloneqq \infty, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}}, q_E \coloneqq q_E^{\mathsf{PKEBC}},\\ q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\textit{Replay-Correctness},\end{aligned}$$

with $\varepsilon > \varepsilon_{\mathsf{RCorr}}^{\mathsf{PKEBC}}$ *and* $t \approx t_{\mathsf{PKEBC}}$.

Proof. Since every SCBC ciphertext includes a PKEBC ciphertext, the claim follows directly from Equation H.4. □

H.3 Adversarial Replay-Correctness

Theorem 26. *If* PKEBC *is*

$$\begin{aligned}(\cdot, \cdot, &\varepsilon_{\mathsf{aRCorr}}^{\mathsf{PKEBC}}, \cdot, \cdot, \cdot, \cdot, \varepsilon_{\mathsf{aRob}}^{\mathsf{PKEBC}}, \varepsilon_{\mathsf{Rob}}^{\mathsf{PKEBC}}, \varepsilon_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}^{\mathsf{PKEBC}}, t^{\mathsf{PKEBC}},\\ &n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}\textit{secure},\end{aligned} \tag{H.5}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S \coloneqq \infty, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}}, q_E \coloneqq q_E^{\mathsf{PKEBC}},\\ q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\textit{Adversarial Replay-Correctness},\end{aligned}$$

with $\varepsilon > \varepsilon_{\mathsf{RCorr}}^{\mathsf{PKEBC}}$ *and* $t \approx t_{\mathsf{PKEBC}}$.

Proof. Analogous to Theorem 25. □

H.4 Public-Key Collision Resistance

Corollary 1. *If* EUF *is* n^{EUF}*-Party* $\varepsilon^{\mathsf{EUF}}_{\mathsf{PK\text{-}Coll\text{-}Res}}$*-Public Key Collision Resistant and* PKEBC *is* n^{PKEBC}*-Party* $\varepsilon^{\mathsf{PKEBC}}_{\mathsf{PK\text{-}Coll\text{-}Res}}$*-Public Key Collision Resistant then* SCBC *is* (n_S, n_R)*-Party* ε*-Public Key Collision Resistant with* $(n_S \coloneqq n^{\mathsf{EUF}}, n_R \coloneqq n^{\mathsf{PKEBC}})$ *and* $\varepsilon \coloneqq \varepsilon^{\mathsf{EUF}}_{\mathsf{PK\text{-}Coll\text{-}Res}} + \varepsilon^{\mathsf{PKEBC}}_{\mathsf{PK\text{-}Coll\text{-}Res}}$.

Proof. Follows from the definition of SCBC (Algorithm 8) and the assumptions on EUF and PKEBC.

H.5 Self-Consistency

Theorem 27. *If* PKEBC *is*

$$\begin{aligned}&(\cdot,\cdot,\cdot,\cdot,\varepsilon^{\mathsf{PKEBC}}_{\mathsf{Self\text{-}Cons}},\cdot,\cdot,\cdot,\cdot,t^{\mathsf{PKEBC}},\\&\quad n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}\textit{secure},\end{aligned} \tag{H.6}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S \coloneqq \infty, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}},\\ q_E \coloneqq q_E^{\mathsf{PKEBC}}, q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\textit{Self-Consistency},\end{aligned}$$

with $\varepsilon > \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Self\text{-}Cons}}$*, and* $t_{\mathsf{PKEBC}} \approx t$.

Proof. Follows directly from PKEBC's Self-Consistency. □

H.6 Consistency

Theorem 28. *If* PKEBC *is*

$$\begin{aligned}&(\cdot,\cdot,\cdot,\cdot,\varepsilon^{\mathsf{PKEBC}}_{\mathsf{Self\text{-}Cons}},\varepsilon^{\mathsf{PKEBC}}_{\mathsf{Cons}},\cdot,\cdot,\cdot,t^{\mathsf{PKEBC}},\\&\quad n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}\textit{secure},\end{aligned} \tag{H.7}$$

and both EUF *and* sEUF *have deterministic signature verification, then no adversary* $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S \coloneqq \infty, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}},\\ q_E \coloneqq q_E^{\mathsf{PKEBC}}, q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\textit{Consistency},\end{aligned}$$

with $\varepsilon > \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Cons}} + \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Self\text{-}Cons}}$*, and* $t_{\mathsf{PKEBC}} \approx t$.

Proof. As we will now show, since both $\mathsf{EUF.Vrfy}$ and $\mathsf{sEUF.Vrfy}$ are deterministic, we can reduce an adversary breaking SCBC's consistency directly into one breaking PKEBC's consistency.

$\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{G}^1$**:** The only difference between games $\mathbf{G}^1$ and $\mathbf{G}^{\mathsf{Cons}}$ is that in $\mathbf{G}^1$ decryption queries are handled according to Algorithm 26. Since $\mathsf{sEUF.Vrfy}$ is a deterministic algorithm, $\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Cons}\right] - \Pr\left[\mathbf{AG}^{\mathsf{Cons}} \Rightarrow \neg\mathsf{Cons}\right]\right| = 0$.

Algorithm 26 Description of $\mathcal{O}_D$ for game system $\mathbf{G}^1$ in the proof of Theorem 28.

INITIALIZATION
  $\mathtt{vfy}_{\mathsf{sEUF}} \leftarrow \emptyset$

$\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$
  if $(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c) \notin \mathtt{vfy}_{\mathsf{sEUF}}$ **:**
    CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  if $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] = 1$ **:**
    $(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})) \leftarrow \mathsf{PKEBC.Dec}(\mathtt{rsk}_j.\mathtt{sk}, c)$
    if $(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})) \neq \perp$ **:**
      if $\mathsf{EUF.Vrfy}(\mathtt{spk}_i, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}), \sigma_{\mathsf{EUF}}) = 1$ **:**
        return $(\mathtt{spk}_i, \vec{v}, m)$
  return $\perp$

CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] \leftarrow \mathsf{sEUF.Vrfy}_{\mathtt{vk}_{\mathsf{sEUF}}}(c, \sigma_{\mathsf{sEUF}})$

Algorithm 27 Description of $\mathcal{O}_D$ for game system $\mathbf{G}^2$ in the proof of Theorem 28.

INITIALIZATION
  $\mathtt{vfy}_{\mathsf{sEUF}} \leftarrow \emptyset$
  $\mathtt{vfy}_{\mathsf{EUF}} \leftarrow \emptyset$

$\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$
  if $(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c) \notin \mathtt{vfy}_{\mathsf{sEUF}}$ **:**
    CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  if $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] = 1$ **:**
    $(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})) \leftarrow \mathsf{PKEBC.Dec}(\mathtt{rsk}_j.\mathtt{sk}, c)$
    if $(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})) \neq \perp$ **:**
      if $(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})) \notin \mathtt{vfy}_{\mathsf{EUF}}$ **:**
        CONSISTENT-EUF-VERIFICATION$(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$
      if $\mathtt{vfy}_{\mathsf{EUF}}[\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})] = 1$ **:**
        return $(\mathtt{spk}_i, \vec{v}, m)$
  return $\perp$

CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] \leftarrow \mathsf{sEUF.Vrfy}(\mathtt{vk}_{\mathsf{sEUF}}, c, \sigma_{\mathsf{sEUF}})$

CONSISTENT-EUF-VERIFICATION$(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, m' \coloneqq (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$
  $\mathtt{vfy}_{\mathsf{EUF}}[\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, m'] \leftarrow \mathsf{EUF.Vrfy}(\mathtt{spk}_i, m', \sigma_{\mathsf{EUF}})$

Algorithm 28 Description of $\mathcal{O}_D$ for game system $\mathbf{G}^3$ in the proof of Theorem 28.

INITIALIZATION
  $\mathtt{dec} \leftarrow \emptyset$ ▷ Immutable map: for each x, once $\mathtt{dec}[x]$ is set, it cannot be modified.
  $\mathtt{vfy}_{\mathsf{sEUF}} \leftarrow \emptyset$
  $\mathtt{vfy}_{\mathsf{EUF}} \leftarrow \emptyset$

$\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} := (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$
  if $(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c) \notin \mathtt{vfy}_{\mathsf{sEUF}}$ **:**
    CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  if $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] = 1$ **:**
    if $(\mathtt{rpk}_j, c) \notin \mathtt{dec}$ **:**
      CONSISTENT-PKEBC-DECRYPTION(B_j, c)
    $\big(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})\big) \leftarrow \mathtt{dec}[\mathtt{rpk}_j, c]$
    if $\big(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})\big) \neq \perp$ **:**
      if $(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})) \notin \mathtt{vfy}_{\mathsf{EUF}}$ **:**
        CONSISTENT-EUF-VERIFICATION$(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$
      if $\mathtt{vfy}_{\mathsf{EUF}}[\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})] = 1$ **:**
        return $(\mathtt{spk}_i, \vec{v}, m)$
  return $\perp$

CONSISTENTDECRYPTION(B_j, c)
  $\big(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})\big) \leftarrow \mathsf{PKEBC.Dec}(\mathtt{rsk}_j.\mathtt{sk}, c)$
  if $\big(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})\big) = \perp$ **:** ▷ Decryption failed.
    $\mathtt{dec}[\mathtt{rpk}_j, c] \leftarrow \perp$
  else if $(\mathtt{rpk}_j \notin \vec{v}) \vee (\exists \mathtt{pk} \in \vec{v}$ with $(\mathtt{pk}, c) \in \mathtt{dec})$ **:** ▷ $\neg$Self-Cons $\vee$ $\neg$Cons occurred.
    Abort
  else
    for $\mathtt{pk} \in \vec{v}$**:** $\mathtt{dec}[\mathtt{pk}, c] \leftarrow \big(\vec{v}, (\mathtt{spk}_i, m, \sigma_{\mathsf{EUF}})\big)$

CONSISTENT-sEUF-VERIFICATION$(\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$
  $\mathtt{vfy}_{\mathsf{sEUF}}[\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c] \leftarrow \mathsf{sEUF.Vrfy}(\mathtt{vk}_{\mathsf{sEUF}}, c, \sigma_{\mathsf{sEUF}})$

CONSISTENT-EUF-VERIFICATION$(\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, m' := (\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}}))$
  $\mathtt{vfy}_{\mathsf{EUF}}[\mathtt{spk}_i, \sigma_{\mathsf{EUF}}, m'] \leftarrow \mathsf{EUF.Vrfy}(\mathtt{spk}_i, m', \sigma_{\mathsf{EUF}})$

$\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$**:** The only difference between games $\mathbf{G}^2$ and $\mathbf{G}^1$ is that in $\mathbf{G}^2$ decryption queries are handled according to Algorithm 27. Since $\mathsf{EUF.Vrfy}$ is a deterministic algorithm, it follows $\left|\Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Cons}\right] - \Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{Cons}\right]\right| = 0$.

$\mathbf{G}^2 \rightsquigarrow \mathbf{G}^3$**:** The only difference between games $\mathbf{G}^3$ and $\mathbf{G}^2$ is that in $\mathbf{G}^3$ decryption queries are handled according to Algorithm 28. Distinguishing these games can be reduced to breaking PKEBC's Consistency or Self-Consistency: a reduction can handle any receiver key-related queries by relying on PKEBC's underlying $\mathcal{O}_{PK}$ and $\mathcal{O}_{SK}$ oracles and can handle decryption queries using the PKEBC's $\mathcal{O}_D$ oracle. If $\mathbf{A}$ only queries $\mathcal{O}_{PK}$, $\mathcal{O}_{SK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most $n_R \leq n^{\mathsf{PKEBC}}$ different receivers, makes at most $q_E \leq q_E^{\mathsf{PKEBC}}$ and $q_D \leq q_D^{\mathsf{PKEBC}}$ queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, and the sum of lengths of the party vectors $\mathbf{A}$ inputs to $\mathcal{O}_E$ is at most $d_E \leq d_E^{\mathsf{PKEBC}}$, then it follows from Equation (H.7)

$$\left|\Pr\left[\mathbf{AG}^3 \Rightarrow \neg\mathsf{Cons}\right] - \Pr\left[\mathbf{AG}^2 \Rightarrow \neg\mathsf{Cons}\right]\right| \leq \varepsilon_{\mathsf{Cons}}^{\mathsf{PKEBC}} + \varepsilon_{\mathsf{Self\text{-}Cons}}^{\mathsf{PKEBC}}.$$

To conclude the proof, note that $\mathbf{G}^3$ cannot be won. □

H.7 Adversarial Robustness

Theorem 29. *If* PKEBC *is*

$$\begin{aligned}(\cdot,\cdot,\cdot,\cdot,\cdot,\cdot,\varepsilon_{\mathsf{aRob}}^{\mathsf{PKEBC}},\cdot,\cdot,t^{\mathsf{PKEBC}},\\ n^{\mathsf{PKEBC}},d_E^{\mathsf{PKEBC}},q_E^{\mathsf{PKEBC}},q_D^{\mathsf{PKEBC}},q_P^{\mathsf{PKEBC}})\text{-}\textit{secure,}\end{aligned} \tag{H.8}$$

then no adversary $\mathbf{A}$ (ε,t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S \coloneqq \infty, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}}, q_E \coloneqq q_E^{\mathsf{PKEBC}},\\ q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_D \coloneqq q_D^{\mathsf{PKEBC}}, q_P \coloneqq q_P^{\mathsf{PKEBC}})\text{-}\textit{Adversarial Robustness,}\end{aligned}$$

with $\varepsilon > \varepsilon_{\mathsf{aRob}}^{\mathsf{PKEBC}}$*, and* $t_{\mathsf{PKEBC}} \approx t$*.*

Proof. First recall that $\mathbf{A}$ wins if it makes two queries $\mathcal{O}_D(B_i, c_{\mathsf{SCBC}})$ and $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}})$, the output of the first query is some triple $(\mathtt{spk}, \vec{v}, m) \neq \perp$ where $\mathtt{rpk}_j \notin \vec{v}$—$\mathtt{rpk}_j$ being B_j's public key—and the output of the second query is not $\perp$. We can then bound $\mathbf{A}$'s advantage by reducing directly to the adversarial robustness of the underlying PKEBC.

Algorithm 29 Description of $\mathcal{O}_D$ for game system $\mathbf{G}^1$ in the proof of Theorem 29.

```
INITIALIZATION
  dec ← ∅

O_D(B_j, c_SCBC := (vk_sEUF, σ_sEUF, c))
  if sEUF.Vrfy(vk_sEUF, σ_sEUF, c) = 1 :
    if (c ∉ dec) ∨ ((c ∈ dec) ∧ (rpk_j ∈ dec[c])) :
      (v⃗, (spk_i, m, σ_EUF)) ← PKEBC.Dec(rsk_j.sk, c)
      if (v⃗, (spk_i, m, σ_EUF)) ≠ ⊥ :
        if c ∉ dec: dec[c] ← v⃗
        if EUF.Vrfy(spk_i, (v⃗, m, vk_sEUF), σ_EUF) = 1 :
          return (spk_i, v⃗, m)
  return ⊥
```

$\mathbf{G}^{\mathsf{aRob}} \rightsquigarrow \mathbf{G}^1$: In $\mathbf{G}^1$ decryption queries are handled differently (see Algorithm 29); concretely, $\mathbf{G}^1$ is defined so that an adversary cannot win the adversarial robustness game. One can reduce distinguishing $\mathbf{G}^1$ and $\mathbf{G}^{\mathsf{aRob}}$ to breaking the adversarial robustness of the underlying PKEBC: the reduction holds all EUF and sEUF secret keys, and can use the oracles provided by the adversarial robustness game to handle any queries that involve PKEBC keys, so it can handle any oracle queries; in particular it can handle decryption queries using the decryption oracle provided by PKEBC's adversarial robustness game. If $\mathbf{A}$ only queries $\mathcal{O}_{RPK}$, $\mathcal{O}_{RK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most $n_R \leq n^{\mathsf{PKEBC}}$ different receivers, makes at most $q_E \leq q_E^{\mathsf{PKEBC}}$ and $q_D \leq q_D^{\mathsf{PKEBC}}$ queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, and the sum of lengths of the party vectors input to $\mathcal{O}_E$ is at most $d_E \leq d_E^{\mathsf{PKEBC}}$, then it follows from Equation (H.8) that $\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{aRob}\right] - \Pr\left[\mathbf{AG}^{\mathsf{aRob}} \Rightarrow \neg\mathsf{aRob}\right]\right| \leq \varepsilon_{\mathsf{aRob}}^{\mathsf{PKEBC}}$.

Finally, note that $\mathbf{G}^1$ cannot be won. □

H.8 Replay Unforgeability

Theorem 30. *If* EUF *is*

$$(\cdot, \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, \cdot, \cdot, t^{\mathsf{EUF}}, n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})\text{-}\textit{secure,} \tag{H.9}$$

and sEUF *is*

$$(\cdot, \cdot, \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}}, \cdot, t^{\mathsf{sEUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})\text{-}\textit{secure,} \tag{H.10}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\begin{aligned}(n_S &:= n_S^{\mathsf{EUF}}, n_R := \infty, q_E := \min(q_S^{\mathsf{EUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}),\\ &q_D := \min(q_V^{\mathsf{EUF}}, q_V^{\mathsf{sEUF}}), q_P := \infty)\text{-}\textit{Replay Unforgeability,}\end{aligned}$$

with $\varepsilon > \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}} + \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$*, and* $t^{\mathsf{sEUF}}, t^{\mathsf{EUF}} \approx t + t_{\mathsf{RUnforg}}$*, where* t_{RUnforg} *is the time to run* SCBC*'s* $\mathbf{G}^{\mathsf{RUnforg}}$ *game.*

Proof. This proof proceeds via a sequence of games.

$\mathbf{G}^{\mathsf{RUnforg}} \rightsquigarrow \mathbf{G}^1$**:** The difference between $\mathbf{G}^1$ and $\mathbf{G}^{\mathsf{RUnforg}}$ is that in $\mathbf{G}^1$, when $\mathcal{O}_D$ is queried on an input $(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that there is a query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}{}^*, \sigma_{\mathsf{sEUF}}{}^*, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$ and $\mathtt{vk}_{\mathsf{sEUF}} = \mathtt{vk}_{\mathsf{sEUF}}{}^*$, $\mathcal{O}_D$ simply outputs $\perp$.

One can reduce distinguishing the two games to breaking the $\mathsf{OT\text{-}sEUF\text{-}CMA}$ security of sEUF: the reduction holds all EUF and PKEBC secret keys and can sign ciphertexts using sEUF's $\mathcal{O}_S$ oracle so it can handle any oracle queries. If $\mathbf{A}$ only makes at most $q_E \leq \min(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}})$ and $q_D \leq q_V^{\mathsf{sEUF}}$ queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, since from Equation (H.10) no adversary $(t^{\mathsf{sEUF}}, \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}})$-breaks the $(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})$-$\mathsf{OT\text{-}sEUF\text{-}CMA}$ of sEUF, it follows

$$\left|\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{RUnforg}\right] - \Pr\left[\mathbf{AG}^{\mathsf{RUnforg}} \Rightarrow \neg\mathsf{RUnforg}\right]\right| \leq \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}}.$$

We now reduce to the $\mathsf{EUF\text{-}CMA}$ game of EUF: $\mathbf{G}^1$ already outputs $\perp$ for any query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that there was a query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}{}^*, \sigma_{\mathsf{sEUF}}{}^*, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$ and $\mathtt{vk}_{\mathsf{sEUF}} = \mathtt{vk}_{\mathsf{sEUF}}{}^*$, so we only have to make sure that no decryption query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}, \sigma', c))$ such that there was no query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output such c^*_{SCBC} allows the adversary to win the game. If $c_{\mathsf{SCBC}} = c^*_{\mathsf{SCBC}}$ the adversary does not win the game (see Definition 18); on the other hand, if some query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ (where $\mathtt{vk}$ was not output as part of any challenge ciphertext) outputs something other than $\perp$, then the EUF signature encrypted by c actually verified as being a valid signature on a triple $(\vec{v}, m, \mathtt{vk}_{\mathsf{sEUF}})$ which was never signed (since $\mathtt{vk}_{\mathsf{sEUF}}$ was not output as part of any $\mathcal{O}_E$ ciphertext). Since from Equation (H.9) no adversary $(\varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, t^{\mathsf{EUF}})$-breaks the

$$(n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})\text{-}\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$$

of EUF, if $\mathbf{A}$ only queries for at most $n_S \leq n^{\mathsf{EUF}}$ different sender keys, makes up to $q_E \leq q_S^{\mathsf{EUF}}$ queries to $\mathcal{O}_E$ and up to $q_D \leq q_V^{\mathsf{EUF}}$ queries to $\mathcal{O}_D$, it follows $\Pr\left[\mathbf{AG}^1 \Rightarrow \neg\mathsf{RUnforg}\right] \leq \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$. □

H.9 $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}^{\mathsf{adap}}$ Security

Theorem 31. *If* PKEBC *is*

$$\begin{aligned}(\varepsilon^{\mathsf{PKEBC}}_{\mathsf{Corr}}, \cdot, \cdot, \cdot, \cdot, \cdot, \cdot, \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Rob}}, \varepsilon^{\mathsf{PKEBC}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}, t^{\mathsf{PKEBC}},\\ n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}}, q_P^{\mathsf{PKEBC}})\text{-}\textit{secure},\end{aligned} \tag{H.11}$$

EUF *is*

$$(\cdot, \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, \cdot, \cdot, t^{\mathsf{EUF}}, n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})\text{-}\textit{secure}, \tag{H.12}$$

and sEUF *is*

$$(\cdot, \cdot, \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}}, \cdot, t^{\mathsf{sEUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})\text{-}\textit{secure}, \tag{H.13}$$

then no adversary $\mathbf{A}$ (ε, t)*-breaks* SCBC*'s*

$$\Big(n_S \coloneqq n^{\mathsf{EUF}}, n_R \coloneqq n^{\mathsf{PKEBC}}, d_E \coloneqq d_E^{\mathsf{PKEBC}},$$
$$q_E \coloneqq \min(q_E^{\mathsf{PKEBC}}, q_S^{\mathsf{EUF}}, n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}),$$
$$q_D \coloneqq \min(\max(q_D^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, 0), q_V^{\mathsf{EUF}}, q_V^{\mathsf{sEUF}})\Big)$$
$$\text{-}\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}\text{-}2^{\mathsf{adap}} \textit{ security,}$$

with $\varepsilon > 2 \cdot \left(\varepsilon^{\mathsf{sEUF}}_{\mathsf{OT}\text{-}\mathsf{sEUF}\text{-}\mathsf{CMA}} + \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF}\text{-}\mathsf{CMA}^{\mathsf{adap}}} + \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Rob}} + 2 \cdot \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Corr}}\right) + \varepsilon^{\mathsf{PKEBC}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}$ *and with* $t^{\mathsf{sEUF}}, t^{\mathsf{EUF}}, t^{\mathsf{PKEBC}} \approx t + t_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}$*, where* $t_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}$ *is the time to run* SCBC*'s* $\mathbf{G}^{\mathsf{CCA}}$ *games.*

Proof. This proof is analogous to the one given in [51,52, Theorem 13] for their MDRS-PKE scheme, and to [47, Theorem 9], which explains how to handle some additional (non-challenge) encryption queries. We only present it for completeness.

Our scheme's ciphertexts are triples $c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c)$ where $\mathtt{vk}_{\mathsf{sEUF}}$ and σ_{sEUF} are, respectively, a verification key and a signature of the underlying sEUF and c is a PKEBC ciphertext. Our goal is reducing an adversary distinguishing SCBC's $\{\mathsf{IND}, \mathsf{IK}\}$-CCA-$2^{\mathsf{adap}}$ games to one distinguishing the analogous games for (the underlying) PKEBC.

First, consider queries $\mathcal{O}_E((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1))$ with $(A_{i,0}, \vec{V}_0, m_0) = (A_{i,1}, \vec{V}_1, m_1)$. Reductions to the security of either sEUF or PKEBC can handle such queries easily because they have access to all sender's EUF secret keys: so they can simply generate SCBC ciphertexts locally—without relying on the oracles provided by the sEUF or PKEBC games. This allows reductions to PKEBC's $\{\mathsf{IND}, \mathsf{IK}\}$-CCA-$2^{\mathsf{adap}}$ security to rely on the provided $\mathcal{O}_D$ oracle even if the PKEBC component of the ciphertext had been output as part of such a challenge ciphertext. A reduction to EUF's EUF-$\mathsf{CMA}^{\mathsf{adap}}$ security can use the signing $\mathcal{O}_S$ for signing on behalf of honest senders without querying for their secret keys.

Now consider queries $\mathcal{O}_E((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1))$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$. Reductions to the security of either sEUF or EUF can handle such challenge queries by relying on the signing oracles provided by sEUF's and EUF's games; decryption queries can be handled as well since these reductions have access to all PKEBC secret keys. Reductions to the security of PKEBC have to generate all EUF and sEUF keys, so they can handle senders' corruption queries (using EUF's secret keys) as well as encryption queries (using both EUF's and sEUF's keys); they can rely on the underlying encryption oracles to generate PKEBC components of SCBC ciphertexts. If the adversary makes a query

$$\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$$

for a c_{SCBC} that was output by a challenge $\mathcal{O}_E$ query, the reduction can refuse to decrypt; if c_{SCBC} was not output by a query to $\mathcal{O}_E$ and the PKEBC ciphertext component c of c_{SCBC} was also not output (as part of any ciphertext output by $\mathcal{O}_E$) then we can use PKEBC's $\{\mathsf{IND}, \mathsf{IK}\}$-CCA-$2^{\mathsf{adap}}$ games $\mathcal{O}_D$ oracle. However

there is a problem when c_{SCBC} was not output by any challenge query but its c component was: this disallows us from relying on PKEBC's oracle $\mathcal{O}_D$. To get around this we will show—via a sequence of hybrids starting from $\mathbf{G}^{\mathsf{CCA}}_{\beta}$ and ending in $\mathbf{G}^{5}_{\beta}$, for $\beta \in \{0,1\}$—that such queries can be handled by simply outputting $\perp$, thus enabling a reduction to PKEBC's $\{\mathsf{IND},\mathsf{IK}\}$-CCA-$2^{\mathsf{adap}}$ security. Consider any query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that c_{SCBC} was not output by a challenge query to $\mathcal{O}_E$ but c was (in the sense above); recall we only need to consider challenge queries $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$; the following hybrids highlight the main steps of the proof:

- $\mathbf{G}^{1}_{\beta}$**:** if any $\mathcal{O}_E$ query output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $\mathtt{vk}_{\mathsf{sEUF}} = \mathtt{vk}^*_{\mathsf{sEUF}}$, $\mathcal{O}_D$ outputs $\perp$;
- $\mathbf{G}^{2}_{\beta}$**:** if any query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $B_j \notin \vec{V}_{\beta}$ and $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$, $\mathcal{O}_D$ outputs $\perp$;
- $\mathbf{G}^{5}_{\beta}$**:** if any query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $B_j \in \vec{V}_{\beta}$ and $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$, $\mathcal{O}_D$ outputs $\perp$.

In $\mathbf{G}^{5}_{\beta}$ we know how to handle any decryption queries for ciphertexts that were not output by $\mathcal{O}_E$ but whose PKEBC component was, and hence we are (essentially) set to make the final reduction to the $\{\mathsf{IND},\mathsf{IK}\}$-CCA-$2^{\mathsf{adap}}$ security of PKEBC. In the following, let $\beta \in \{0,1\}$.

$\mathbf{G}^{\mathsf{CCA}}_{\beta} \rightsquigarrow \mathbf{G}^{1}_{\beta}$: In $\mathbf{G}^{1}_{\beta}$ some decryption queries are handled differently: when $\mathcal{O}_D$ is queried on an input $(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that there is a query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ where $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$ and $(c, \mathtt{vk}_{\mathsf{sEUF}}) = (c^*, \mathtt{vk}^*_{\mathsf{sEUF}})$, $\mathcal{O}_D$ outputs $\perp$.

One can reduce distinguishing the two games to breaking the OT-sEUF-CMA security of sEUF: since the reduction holds all EUF and PKEBC secret keys and can sign ciphertexts using oracle $\mathcal{O}_S$ from sEUF's $\mathbf{G}^{\mathsf{OT\text{-}sEUF\text{-}CMA}}$ game, it can handle any oracle queries. If $\mathbf{A}$ only makes at most $q_E \leq \min(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}})$ and $q_D \leq q_V^{\mathsf{sEUF}}$ queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, since no adversary $(t^{\mathsf{sEUF}}, \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}})$-breaks sEUF's $(n^{\mathsf{sEUF}}, q_S^{\mathsf{sEUF}}, q_V^{\mathsf{sEUF}})$-OT-sEUF-CMA security (Equation H.13):

$$\Big|\Pr\big[\mathbf{AG}^{1}_{\beta} \Rightarrow \mathtt{win}\big] - \Pr\big[\mathbf{AG}^{\mathsf{CCA}}_{\beta} \Rightarrow \mathtt{win}\big]\Big| \leq \varepsilon^{\mathsf{sEUF}}_{\mathsf{OT\text{-}sEUF\text{-}CMA}}.$$

$\mathbf{G}^{1}_{\beta} \rightsquigarrow \mathbf{G}^{2}_{\beta}$: In $\mathbf{G}^{2}_{\beta}$ some decryption queries are once again handled differently; when $\mathcal{O}_D$ is queried on an input $(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ and there is a query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ where $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$, and $B_j \notin \vec{V}_{\beta}$, $\mathcal{O}_D$ outputs $\perp$.

One can reduce distinguishing $\mathbf{G}^{2}_{\beta}$ and $\mathbf{G}^{1}_{\beta}$ to breaking PKEBC's robustness (Definition 8). The main things to note are:

1. a reduction to PKEBC's robustness can rely on PKEBC's $\mathcal{O}_{SK}$ to access every PKEBC secret key that an adversary can access (via SCBC's oracle $\mathcal{O}_{RK}$), can use PKEBC's $\mathcal{O}_D$ to decrypt any ciphertexts on behalf of receivers without knowing their PKEBC secret keys, and has access to all EUF and sEUF keys (since it has to generate them), so it can handle any oracle queries that an adversary may make;
2. for a query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$, with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$ the reduction can make a query $\mathcal{O}_E(A_{i,\beta}, \vec{V}_\beta, m_\beta)$ to PKEBC's robustness game; and
3. if a query $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ does not output $\perp$, then the decryption by B_j of the PKEBC ciphertext c (that is part of c_{SCBC}) did not result in $\perp$ either.

If $\mathbf{A}$ only queries for at most $n_R \leq n^{\mathsf{PKEBC}}$ different receivers, the sum of lengths of the vectors input to $\mathcal{O}_E$ is at most $d_E \leq d_E^{\mathsf{PKEBC}}$, and $\mathbf{A}$ makes at most $q_E \leq q_E^{\mathsf{PKEBC}}$ and $q_D \leq q_D^{\mathsf{PKEBC}}$ queries to oracles $\mathcal{O}_E$ and $\mathcal{O}_D$, it follows from Equation (H.11)

$$\Big|\Pr\big[\mathbf{AG}_\beta^2 \Rightarrow \mathtt{win}\big] - \Pr\big[\mathbf{AG}_\beta^1 \Rightarrow \mathtt{win}\big]\Big| \leq \varepsilon_{\mathsf{Rob}}^{\mathsf{PKEBC}}.$$

$\mathbf{G}_\beta^2 \rightsquigarrow \mathbf{G}_\beta^3$: In $\mathbf{G}_\beta^3$ some decryption queries are handled differently: when $\mathcal{O}_D$ is queried on an input $(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that there is a query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ with $(A_{i,0}, \vec{V}_0, m_0) \neq (A_{i,1}, \vec{V}_1, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$, $c = c^*$, and $B_j \in \vec{V}_\beta$, $\mathcal{O}_D$ works as follows: let $(\mathtt{spk}_{i,\beta}, m_\beta, \sigma_{\mathsf{EUF}})$ be the plaintext encrypted by PKEBC.Enc under $\vec{v_\beta}_{\mathsf{PKEBC}}$ (which resulted in ciphertext c), where $\mathtt{spk}_{i,\beta}$ is $A_{i,\beta}$'s public key, $\vec{v_\beta}_{\mathsf{PKEBC}} \coloneqq (\mathtt{pk}_{\mathsf{PKEBC}1,\beta}, \ldots, \mathtt{pk}_{\mathsf{PKEBC}|\vec{v}|,\beta})$ is the vector of PKEBC public keys corresponding to $\vec{V}_\beta$, and where

$$\sigma_{\mathsf{EUF}} \leftarrow \mathsf{EUF.Sign}(\mathtt{ssk}_{i,\beta}.\mathtt{sk}, (\vec{v_\beta}_{\mathsf{PKEBC}}, m_\beta, \mathtt{vk}_{\mathsf{sEUF}})),$$

is an EUF signature on $(\vec{v_\beta}_{\mathsf{PKEBC}}, m_\beta, \mathtt{vk}_{\mathsf{sEUF}})$, with $\mathtt{ssk}_{i,\beta}$ being $A_{i,\beta}$'s secret key and $\mathtt{vk}_{\mathsf{sEUF}}$ being the sEUF verification key in c_{SCBC}; if the signature σ_{sEUF} verifies as valid for c under verification key $\mathtt{vk}_{\mathsf{sEUF}}$, oracle $\mathcal{O}_D$ no longer decrypts c using PKEBC.Dec with B_j's PKEBC secret key, and instead simply assumes decryption outputs $(\vec{v_\beta}_{\mathsf{PKEBC}}, (\mathtt{spk}_{i,\beta}, m_\beta, \sigma_{\mathsf{EUF}}))$. By considering a reduction and an argument similar to the one in the proof of Theorem 24 for the step $\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$ (For query $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ the reduction queries $\mathcal{O}_E(A_{i,\beta}, \vec{V}_\beta, m_\beta)$ to PKEBC's Correctness game), we have

$$\Big|\Pr\big[\mathbf{AG}_\beta^3 \Rightarrow \mathtt{win}\big] - \Pr\big[\mathbf{AG}_\beta^2 \Rightarrow \mathtt{win}\big]\Big| \leq \varepsilon_{\mathsf{Corr}}^{\mathsf{PKEBC}}.$$

$\mathbf{G}^3_\beta \rightsquigarrow \mathbf{G}^4_\beta$**:** In $\mathbf{G}^4_\beta$ some decryption queries are handled differently: when $\mathcal{O}_D$ is queried on an input $(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that there is a query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ where $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$, and $B_j \in \vec{V_\beta}$, $\mathcal{O}_D$ simply outputs $\perp$. Before moving to showing that

$$\Big|\Pr\big[\mathbf{AG}^4_\beta \Rightarrow \mathtt{win}\big] - \Pr\big[\mathbf{AG}^3_\beta \Rightarrow \mathtt{win}\big]\Big| \leq \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}.$$

(by explaining why one can make a reduction to the $\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ notion of EUF), we want to note that, at this point, for any $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ query the adversary makes such that there is a query to $\mathcal{O}_E$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$ and $c = c^*$, $\mathcal{O}_D$ simply outputs $\perp$. We are essentially set for PKEBC's $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA\text{-}2}^{\mathsf{adap}}$ reduction).

$\mathbf{G}^3_\beta$ already outputs $\perp$ for any $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that:

- there was a query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ where $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$ and $\mathtt{vk}_{\mathsf{sEUF}} = \mathtt{vk}^*_{\mathsf{sEUF}}$ (see $\mathbf{G}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA\text{-}2}}_\beta \rightsquigarrow \mathbf{G}^1_\beta$); or
- there was a query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ where $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$ and $B_j \notin \vec{V_\beta}$ (see $\mathbf{G}^1_\beta \rightsquigarrow \mathbf{G}^2_\beta$).

Thus we only have to make sure that we can use the adversary to break EUF's $\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ security for the remainder of the queries, i.e. queries to $\mathcal{O}_D(B_j, c_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}_{\mathsf{sEUF}}, \sigma_{\mathsf{sEUF}}, c))$ such that:

1. there is no query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$ and $\mathtt{vk}_{\mathsf{sEUF}} = \mathtt{vk}^*_{\mathsf{sEUF}}$;
2. there is no query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$ and $B_j \notin \vec{V_\beta}$; and
3. there exists a query $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$ with $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$ that output $c^*_{\mathsf{SCBC}} \coloneqq (\mathtt{vk}^*_{\mathsf{sEUF}}, \sigma^*_{\mathsf{sEUF}}, c^*)$ with $c_{\mathsf{SCBC}} \neq c^*_{\mathsf{SCBC}}$, $c = c^*$, $\mathtt{vk}_{\mathsf{sEUF}} \neq \mathtt{vk}^*_{\mathsf{sEUF}}$ and $B_j \in \vec{V_\beta}$.

Note that since $B_j \in \vec{V_\beta}$ and because we are assuming the correctness of PKEBC the reduction need not attempt to decrypt c, and instead can simply assume that the decryption is $(\vec{v_\beta}_{\mathsf{PKEBC}}, (\mathtt{spk}_{i,\beta}, m_\beta, \sigma_{\mathsf{EUF}}))$, as explained in step $\mathbf{G}^2_\beta \rightsquigarrow \mathbf{G}^3_\beta$—this is necessary because we need the EUF keys the reduction obtains from the decryption of c to match the ones from the underlying $\mathbf{G}^{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$, as otherwise we cannot win EUF's $\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ game;[15] on the other hand we are

[15] For $\neg\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ to occur, an adversary has to make a query $\mathcal{O}_V(i, m, \sigma)$ where $i \in \mathbb{N}$.

also assuming there was no query to $\mathcal{O}_E$ that output the same verification key $\mathtt{vk}_{\mathsf{sEUF}}$. Since the sEUF verification key is part of the messages that are signed and/or verified using EUF then if σ_{EUF} verifies as being a valid signature on $(\vec{v_\beta}_{\mathsf{PKEBC}}, m_\beta, \mathtt{vk}_{\mathsf{sEUF}})$ with respect to verifying public key $\mathtt{spk}_{i,\beta}.\mathtt{vk}$, the reduction wins the $\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ game of EUF (see Definition 23). Finally, if $\mathbf{A}$ only queries for at most $n_S \leq n^{\mathsf{EUF}}$ different sender keys, makes up to $q_E \leq q_S^{\mathsf{EUF}}$ queries to $\mathcal{O}_E$ and up to $q_D \leq q_V^{\mathsf{EUF}}$ queries to $\mathcal{O}_D$, since from Equation (H.12) no adversary $(\varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, t_{\mathsf{EUF}})$-breaks EUF's $(n^{\mathsf{EUF}}, q_S^{\mathsf{EUF}}, q_V^{\mathsf{EUF}})$-$\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}$ security, $\left|\Pr\left[\mathbf{AG}_\beta^4 \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}_\beta^3 \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon^{\mathsf{EUF}}_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$.

$\mathbf{G}_\beta^4 \rightsquigarrow \mathbf{G}_\beta^5$**:** The only difference between these games is that in $\mathbf{G}_\beta^5$ the distribution of PKEBC key-pairs returns to being the distribution induced by PKEBC's Gen algorithm. (This step is necessary to allow us to reduce an adversary distinguishing $\mathbf{G}_\mathbf{0}^5$ and $\mathbf{G}_\mathbf{1}^5$ to one distinguishing PKEBC's $\mathbf{G}_\mathbf{0}^{\mathsf{CCA}}$ and $\mathbf{G}_\mathbf{1}^{\mathsf{CCA}}$ games.) It then follows $\left|\Pr\left[\mathbf{AG}_\beta^5 \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}_\beta^4 \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Corr}}$.

Finally, we can now reduce an adversary distinguishing $\mathbf{G}_\mathbf{0}^5$ and $\mathbf{G}_\mathbf{1}^5$ to one distinguishing PKEBC's $\mathbf{G}_\mathbf{0}^{\mathsf{CCA}}$ and $\mathbf{G}_\mathbf{1}^{\mathsf{CCA}}$ games because we can handle all decryption queries—either by using the $\mathcal{O}_D$ oracle of PKEBC's $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA\text{-}2}^{\mathsf{adap}}$ games, or by outputting $\perp$. So, if $\mathbf{A}$ only queries for at most $n_R \leq n^{\mathsf{PKEBC}}$ different receivers, the sum of lengths of the vectors input to $\mathcal{O}_E$ is at most $d_E \leq d_E^{\mathsf{PKEBC}}$, and $\mathbf{A}$ makes at most $q_E \leq q_E^{\mathsf{PKEBC}}$ and $q_D \leq q_D^{\mathsf{PKEBC}}$ queries to oracles $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, since from Equation (H.11) that no adversary $(t^{\mathsf{PKEBC}}, \varepsilon^{\mathsf{PKEBC}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}})$-breaks the

$$(n^{\mathsf{PKEBC}}, d_E^{\mathsf{PKEBC}}, q_E^{\mathsf{PKEBC}}, q_D^{\mathsf{PKEBC}})\text{-}\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}} \text{ security}$$

of PKEBC, then $\left|\Pr\left[\mathbf{AG}_\mathbf{0}^5 \Rightarrow \mathtt{win}\right] - \Pr\left[\mathbf{AG}_\mathbf{1}^5 \Rightarrow \mathtt{win}\right]\right| \leq \varepsilon^{\mathsf{PKEBC}}_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}^{\mathsf{adap}}}$. □

I Composable Semantics of SCBC

I.1 Constructive Cryptography

Most of this section is transcribed verbatim from [47].

Resources. Akin to functionalities in UC [9], resources are interactive systems. Just like a mathematical function $f : X \to Y$, a resource also has input and output domains; if a resource $\mathbf{R}$ has input domain $\mathcal{X}$ and output (co-)domain $\mathcal{Y}$ (both being non-empty), we say $\mathbf{R}$ is an $(\mathcal{X}, \mathcal{Y})$ resource. Similarly to functions, one can provide inputs $x \in \mathcal{X}$ to an $(\mathcal{X}, \mathcal{Y})$-resource, which then provides some output $y \in \mathcal{Y}$. Formally, resources are random systems [54,55]; in turn, a random system is defined as a sequence of conditional probability distributions [55, Definition 2]. If two $(\mathcal{X}, \mathcal{Y})$-resources $\mathbf{R}$ and $\mathbf{S}$ are the same sequence of conditional probability distributions, we say they are equivalent and write $\mathbf{R} \equiv \mathbf{S}$ [55, Definition 3]. For simplicity, we usually describe resources by pseudo-code.

We often attach resources together; for (compatible) resources $\mathbf{R}$ and $\mathbf{S}$, we denote by $\mathbf{R} \cdot \mathbf{S}$ the resource resulting from attaching $\mathbf{R}$ and $\mathbf{S}$.[16]. For n resources $\{\mathbf{R}_i\}_{i=1}^n$, where each $\mathbf{R}_i$ is an $(\mathcal{X}_i, \mathcal{Y}_i)$-resource, if for all distinct $i, j \in [n]$, both $\mathcal{X}_i$ and $\mathcal{Y}_i$ are disjoint from $\mathcal{Y}_j$, then we denote the combined resource—corresponding to attaching $\mathbf{R}_1, \ldots, \mathbf{R}_n$ together—by $\mathbf{R} \coloneqq [\mathbf{R}_1, \ldots, \mathbf{R}_n]$, and call $\mathbf{R}$ the parallel composition of $\{\mathbf{R}_i\}_{i=1}^n$.

Interfaces. For an $(\mathcal{X}, \mathcal{Y})$-resource $\mathbf{R}$, an interface $I = (I_\mathcal{X}, I_\mathcal{Y})$ is a pair of subsets of $\mathbf{R}$'s input and output domains, i.e. $I_\mathcal{X} \subseteq \mathcal{X}$ and $I_\mathcal{Y} \subseteq \mathcal{Y}$. We call $I_\mathcal{X}$ (respectively, $I_\mathcal{Y}$) an input (respectively, output) interface of $\mathbf{R}$. For two interfaces $I_1 = (I_{1,\mathcal{X}}, I_{1,\mathcal{Y}})$ and $I_2 = (I_{2,\mathcal{X}}, I_{2,\mathcal{Y}})$, we say that I_1 is a subset of I_2—or write $I_1 \subseteq I_2$—to mean $I_{1,\mathcal{X}} \subseteq I_{2,\mathcal{X}}$ and $I_{1,\mathcal{Y}} \subseteq I_{2,\mathcal{Y}}$ (i.e. the input and output interfaces of I_1 are, respectively, a subset of the input and output interfaces of I_2). Similarly, we say I_1 and I_2 are disjoint—or write $I_1 \cap I_2 = \emptyset$—to mean $I_{1,\mathcal{X}} \cap I_{2,\mathcal{X}} = \emptyset$ and $I_{1,\mathcal{Y}} \cap I_{2,\mathcal{Y}} = \emptyset$ (i.e. the input and output interfaces of I_1 are, respectively, disjoint with the input and output interfaces of I_2). We define the union of interfaces I_1 and I_2, denoted $I_1 \cup I_2$, as $I_1 \cup I_2 \coloneqq (I_{1,\mathcal{X}} \cup I_{2,\mathcal{X}}, I_{1,\mathcal{Y}} \cup I_{2,\mathcal{Y}})$.

A set of interfaces $\mathcal{I}$ of an $(\mathcal{X}, \mathcal{Y})$-resource $\mathbf{R}$ is one such that any distinct interfaces $I_1, I_2 \in \mathcal{I}$ are disjoint, and the union of all interfaces in $\mathcal{I}$ is $\mathbf{R}$'s input and output domains, i.e. $(\mathcal{X}, \mathcal{Y}) = \bigcup_{I \in \mathcal{I}} I$.

When considering (simulator-based) security notions it is often helpful to have the notion of a party. For a set of n parties $\mathcal{P} \coloneqq (P_1, \ldots, P_n)$, one considers a set of interfaces $\mathcal{I}$ where for each party $P \in \mathcal{P}$ there is an interface $I_P = (I_{P,\mathcal{X}} \coloneqq (\{P\} \times \mathcal{X}_P), I_{P,\mathcal{Y}} \coloneqq (\{P\} \times \mathcal{Y}_P))$. We say that $I_{P,\mathcal{X}}$ and $I_{P,\mathcal{Y}}$ are P's input and output interfaces for $\mathbf{R}$, respectively.

Converters. A *converter* is an $(\mathcal{X}, \mathcal{Y})$-resource that is executed either locally by a single party or cooperatively by multiple parties. The inside interface connects to (a subset of those parties' interfaces of) the available resources, resulting in a new resource. For instance, connecting a converter α to (the entirety of) Alice's interface A of a resource $\mathbf{R}$ results in a new resource, which we denote by $\alpha^A \mathbf{R}$; we denote the inside interface of α by $\alpha.in$. The outside interface of the converter α is now the new A-interface of $\alpha^A \mathbf{R}$; we denote it by $\alpha.out$. Thus, a converter may be seen as a map between resources. Converters applied at different interfaces commute [40, Proposition 1]: $\beta^B \alpha^A \mathbf{R} \equiv \alpha^A \beta^B \mathbf{R}$.

A protocol is given by a tuple of converters $\pi = (\pi_{P_i})_{P_i \in \mathcal{P}}$, one for each party $P_i \in \mathcal{P}$. Simulators are also given by converters. For any set $\mathcal{S}$ we denote $(\pi_{P_i})_{P_i \in \mathcal{S}} \mathbf{R}$ by $\pi^\mathcal{S} \mathbf{R}$. We also often drop the interface superscript and write just $\pi \mathbf{R}$ when it is clear from the context to which interfaces π connects.

Distinguishers. To measure the distance between two resources we use the standard notion of a distinguisher, an interactive system $\mathbf{D}$ which interacts with

[16] Resources $\mathbf{R}$ and $\mathbf{S}$ can only be attached together if their composition results in a well-defined sequence of conditional probability distributions (see, e.g. [46, Definition 7]); this is not the case for all pairs of resources.

a resource at all its interfaces, and outputs a bit 0 or 1. The distinguishing advantage for distinguisher $\mathbf{D}$ is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \coloneqq |\Pr[\mathbf{DS} = 1] - \Pr[\mathbf{DR} = 1]|$$

where $\mathbf{DR}$ and $\mathbf{DS}$ are the random variables over the output of $\mathbf{D}$ when it interacts with $\mathbf{R}$ and $\mathbf{S}$, respectively.

Reductions. Typically one proves that the ability to distinguish between two resources is bounded by some function of the distinguisher, e.g. for any $\mathbf{D}$, $\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \leq \varepsilon(\mathbf{D})$ where $\varepsilon(\mathbf{D})$ might be the probability that $\mathbf{D}$ can win a game or solves some problem believed to be hard.

Security Statements. We now define a cryptographic construction.

Definition 45 (Simulation-based construction). *Let $\mathbf{R}$ and $\mathbf{S}$ be two resources and π a protocol for $\mathbf{R}$. We say π ε-constructs $\mathbf{S}$ from $\mathbf{R}$ if there is a simulator sim such that for any distinguisher $\mathbf{D}$, $\Delta^{\mathbf{D}}(\pi\mathbf{R}, \mathsf{sim}\mathbf{S}) \leq \varepsilon(\mathbf{D})$, and the interfaces of sim and π are pairwise disjoint.*

I.2 (Asynchronous) Repositories

We now introduce the repository model from [50]. Most of the following is taken verbatim from [47].

Similarly to [50,47], we model access control via repositories. A repository contains a set of registers and a (corresponding) set of register identifiers IdSet; a register is a pair $\mathtt{reg} = (\mathtt{id}, m)$, where $\mathtt{id}$ is the register's identifier (uniquely identifying it among all repositories), and m is a message. We consider two types of repository access rights: *read access* and *write access*. We denote by $\mathcal{W}$ and $\mathcal{R}$ the sets of parties with write and read access to a repository $\mathbf{rep}$, respectively; to make the access permissions explicit we write $\mathbf{rep}^{\mathcal{W}}_{\mathcal{R}}$, but otherwise simply write $\mathbf{rep}$. For example, consider a three party setting with a sender Alice, a receiver Bob and a dishonest third-party Eve—so $\mathcal{P} = \{A, B, E\}$. An insecure repository—which allows everyone to read and write—is given by $\mathbf{INS}^{\mathcal{P}}_{\mathcal{P}}$; a (replay-protected) authentic repository from Alice to Bob is given by

$\mathbf{AUT}^{\{A\}}_{\{B,E\}}$. The semantics of atomic repositories is defined in Algorithm 30.[17]

Algorithm 30 Atomic repository $\mathbf{rep}^{\mathcal{W}}_{\mathcal{R}}$.

◇ Initialization
  IdSet ← ∅

▷ ($P \in \mathcal{W}$)-Write(m)
  id ← NewRegister(m)
  IdSet.Add(id)
  Output(id)

▷ ($P \in \mathcal{R}$)-Read
  list ← ∅
  for id ∈ IdSet **:**
    list.Add(id, GetMessage(id))
  Output(list)

Algorithm 31 Repository $\mathbf{REP} = [\mathbf{rep_1}^{\mathcal{W}_1}_{\mathcal{R}_1}, \ldots, \mathbf{rep_n}^{\mathcal{W}_n}_{\mathcal{R}_n}]$.

▷ ($P \in \mathcal{P}$)-Write($\mathbf{rep_i}^{\mathcal{W}_i}_{\mathcal{R}_i}, m$)
Require: ($P \in \mathcal{W}_i$)
  Output($\mathbf{rep_i}$-Write(m))

▷ ($P \in \mathcal{P}$)-Read
  list ← ∅
  for $\mathbf{rep_i} \in \mathbf{REP}$ **with** $P \in \mathcal{R}_i$ **:**
    for $(\mathtt{id}, m) \in \mathbf{rep_i}$-Read **:**
      list.Add(id, ($\mathbf{rep_i}, m$))
  Output(list)

Following [50], to model that parties may have access to multiple repositories—say $\mathbf{rep_1}^{\mathcal{W}_1}_{\mathcal{R}_1}, \ldots, \mathbf{rep_n}^{\mathcal{W}_n}_{\mathcal{R}_n}$—we define a new type of repository denoted $\mathbf{REP} = [\mathbf{rep_1}^{\mathcal{W}_1}_{\mathcal{R}_1}, \ldots, \mathbf{rep_n}^{\mathcal{W}_n}_{\mathcal{R}_n}]$, which is essentially a parallel composition of atomic repositories equipped with a single read operation that allows parties to (efficiently) read all their incoming messages at once (instead of having to read from each atomic repository $\mathbf{rep_i}$ they have access to). The exact semantics of **REP** is defined in Algorithm 31.

Asynchronous Repositories. We follow [47] to model asynchronous repositories, i.e. we attach the converter Net introduced in [47] (Algorithm 32) that has an interface for message delivery and only allows honest receivers to read the messages that were already delivered to them.

Algorithm 32 The following is transcribed verbatim from [47]. Semantics of Net for a repository $\mathbf{REP} = [\mathbf{rep_1}, \ldots, \mathbf{rep_n}]$.

◇ Initialization
  for $P_i \in \mathcal{P}$ **:** Received[P_i] ← ∅

▷ ($P \in \overline{\mathcal{P}^H}$)-Read: Output(Read)
▷ Deliver($P \in \mathcal{P}^H$, id): Received[P].Add(id)
▷ ($P \in \mathcal{P}$)-Write($\mathbf{rep_i}, m$):
  Output(Write($\mathbf{rep_i}, m$))

▷ ($P \in \mathcal{P}^H$)-Read
  list ← ∅
  for $(\mathtt{id}, (\mathbf{rep_i}, m)) \in$ Read **:**
    if id ∈ Received[P] **:**
      list.Add(id, ($\mathbf{rep_i}, m$))
  Output(list)

I.3 Composable Security Notions for SCBC

The application semantics for SCBC schemes is very similar to the one for MDRS-PKE schemes introduced in [47]: the main difference is that SCBC schemes

[17] As needed to capture the Off-The-Record guarantee, the repository semantics capture sender anonymity: for a repository $\mathbf{rep}^{\mathcal{W}}_{\mathcal{R}}$, readers do not learn who wrote (among the parties in $\mathcal{W}$) each of the repository's messages.

do not provide Off-The-Record guarantees. Even though in [47] Liu-Zhang et al. focused on MDRS-PKE schemes, thanks to the modularity of their composable notions, the application semantics of SCBC schemes are captured naturally as a special case of the application semantics they introduced, as we now explain. Most of this section is transcribed verbatim from [47], with only minor modifications.

Real World.

Algorithm 33 The **KGA** resource for $\mathsf{SCBC} = (\mathsf{Setup}, \mathsf{Gen_S}, \mathsf{Gen_R}, \mathsf{Enc}, \mathsf{Dec})$.

◇ Initialization
  $\mathtt{pp} \leftarrow \mathsf{Setup}(1^k)$
  for $A_i \in \mathcal{S}$**:** $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathsf{Gen_S}(\mathtt{pp})$
  for $B_j \in \mathcal{R}$**:** $(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow \mathsf{Gen_R}(\mathtt{pp})$

▷ $(P \in \mathcal{P})$-PublicParams Output($\mathtt{pp}$)
▷ $(P \in \mathcal{P})$-SenderPublicKey($A_i \in \mathcal{S}$) Output($\mathtt{spk}_i$)
▷ $(P \in \mathcal{P})$-ReceiverPublicKey($B_j \in \mathcal{R}$) Output($\mathtt{rpk}_j$)
▷ $(A_i \in \mathcal{S}^H)$-SenderKeyPair Output($\mathtt{spk}_i, \mathtt{ssk}_i$)
▷ $(B_j \in \mathcal{R}^H)$-ReceiverKeyPair Output($\mathtt{rpk}_j, \mathtt{rsk}_j$)
▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$) Output($\mathtt{spk}_i, \mathtt{ssk}_i$)
▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$) Output($\mathtt{rpk}_j, \mathtt{rsk}_j$)

▷ $(B_j \in \mathcal{R}^H)$-GetLabel($\mathtt{spk}, \vec{v}$)
  $\mathcal{S}_{\mathtt{spk}} \coloneqq \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
  if $|\mathcal{S}_{\mathtt{spk}}| \neq 1$**:** Output($\perp$)
  for $l = 1, \ldots, |\vec{v}|$ **:**
    $\mathcal{R}_{v_l} \coloneqq \{B_k \mid v_l = \mathtt{rpk}_k\}$
    if $|\mathcal{R}_{v_l}| \neq 1$**:** Output($\perp$)
    else: Let V_l be the element of $\mathcal{R}_{v_l}$
  Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
  Output($\langle A_i \to \vec{V} \coloneqq (V_1, \ldots, V_{|\vec{v}|})\rangle$)

Assumed Resources. Parties have access to an asynchronous and anonymous insecure repository Net·**INS**—to which everyone can write to and read from—and to a *Key Generation Authority* (**KGA**) resource [50], which generates and stores parties' key pairs (see Algorithm 33).

The **KGA** resource—which is implicitly parameterized by a security parameter k—initially generates key-pairs for all senders and receivers—using G_S and G_R, respectively. Every honest party can query their own key-pair and everyone's public keys at their own interface. Dishonest parties can additionally obtain the key-pairs of any dishonest senders or receivers. [18]

Protocol. Each honest sender and each honest receiver locally runs a converter Snd and Rcv, respectively (see Algorithm 34); these converters connect to both

[18] Resource **KGA** supports an additional helper operation GetLabel which, given an sender's public key and a vector of receiver public keys, outputs the unique corresponding label $\langle A_i \to \vec{V}\rangle$, or $\perp$ if the label does not exist or is not unique.

Algorithm 34 Converters Snd and Rcv (left and right columns, respectively).

$\triangleright$ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $\mathtt{pp} \leftarrow$ PUBLICPARAMS
  $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow$ SENDERKEYPAIR
  for $l = 1, \ldots, |\vec{V}|$ **:**
    $\mathtt{rpk}_l \leftarrow$ RECEIVERPUBLICKEY(V_l)
  $c \leftarrow \mathsf{Enc}_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v} \coloneqq (\mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{V}|}), m)$
  OUTPUT(WRITE(c))

$\triangleright$ $(B_j \in \mathcal{R}^H)$-READ
  $\mathtt{pp} \leftarrow$ PUBLICPARAMS
  $(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow$ RECEIVERKEYPAIR
  list, ctxtSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ctxtSet **:**
    ctxtSet.ADD(c)
    $(\mathtt{spk}_i, \vec{v}, m) \leftarrow \mathsf{Dec}_{\mathtt{pp}}(\mathtt{rsk}_j, c)$
    if $(\mathtt{spk}_i, \vec{v}, m) \neq \perp$ **:**
      $\langle A_i \to \vec{V}\rangle \leftarrow$ GETLABEL$(\mathtt{spk}_i, \vec{v})$
      if $\langle A_i \to \vec{V}\rangle \neq \perp$ **:**
        list.ADD$(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))$
  OUTPUT(list)

the **KGA** and to $\mathsf{Net} \cdot \mathbf{INS}$ and provide an outer interface that is identical to the interface of a repository for a party who is a writer and a reader, respectively. A party A_i's Snd converter provides a procedure WRITE which takes as input a label $\langle A_i \to \vec{V}\rangle$ (defining the vector of receivers $\vec{V} = (V_1, \ldots, V_{|\vec{V}|})$) and a message m; upon such input, Snd encrypts the input message, writes the resulting ciphertext to $\mathsf{Net} \cdot \mathbf{INS}$, and outputs the $\mathtt{id}$ output by writing to $\mathsf{Net} \cdot \mathbf{INS}$. A party B_j's Rcv converter reads all (received) ciphertexts from $\mathsf{Net} \cdot \mathbf{INS}$—filtering out duplicated ones—and tries decrypting each of them. For each ciphertexts that decrypt correctly, Rcv uses the **KGA**'s GETLABEL operation to obtain a label—i.e. looks up the sender/vector of receivers with public keys matching the ones obtained from decryption—and then outputs a set of triples, each triple corresponding to a ciphertext that decrypted correctly and for which the GETLABEL operation returned a valid label (i.e. not $\perp$).

Real World System. The real world system $\mathbf{R}$ is given by

$$\mathsf{Snd}^{\mathcal{S}^H}\, \mathsf{Rcv}^{\mathcal{R}^H}\, [\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}]. \tag{I.1}$$

Ideal World. The ideal system $\mathbf{T}$ includes (atomic) repositories

$$\left[\langle A_i \to \vec{V}\rangle^{\{A_i\} \cup \overline{\mathcal{P}^H}}_{\vec{V} \cup \overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+}$$

to which A_i and each dishonest party $P \in \overline{\mathcal{P}^H}$ can write to, and from which each receiver in $\vec{V}$ and each dishonest party $P \in \overline{\mathcal{P}^H}$ can read from. Following [47], to capture authenticity we attach converter $\perp$ to the WRITE sub-interfaces of dishonest parties' for which the sender A_i in the input label $\langle A_i \to \vec{V}\rangle$ is honest (i.e. $A_i \in \mathcal{S}^H$). By having converter $\perp$ cover such interfaces, dishonest parties cannot issue WRITE operations impersonating honest ones: $\perp$ removes such inputs from dishonest parties' input domains. Denoting these sub-interfaces by $\mathsf{Auth\text{-}Intf} \coloneqq \overline{\mathcal{P}^H}$-WRITE$(\langle \mathcal{S}^H \to \mathcal{R}^+\rangle, \cdot)$, this means $\mathbf{T}$ includes

$$\perp^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle^{\{A_i\} \cup \overline{\mathcal{P}^H}}_{\vec{V} \cup \overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+}. \tag{I.2}$$

Following [47], confidentiality and anonymity are modeled similarly, by attaching a converter $\mathsf{ConfAnon}$ (defined in Algorithm 35) to the READ interfaces of dishonest parties that similarly limits their reading capabilities: if an honest sender $A_i \in \mathcal{S}^H$ sends a message m to a vector of receivers $\vec{V}$ all of whom are honest (i.e. $\vec{V} \in {\mathcal{R}^H}^+$), then $\mathsf{ConfAnon}$ only leaks $|\vec{V}|$ and $|m|$ to dishonest parties (instead of $\langle A_i \to \vec{V}\rangle$ and m). By also attaching the network converter Net we then obtain the ideal world system $\mathbf{T}$, which is formally defined as

$$\mathbf{T} \coloneqq \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \cdot \mathsf{Net} \cdot \bot^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle_{\vec{V}\cup\overline{\mathcal{P}^H}}^{\{A_i\}\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}. \tag{I.3}$$

Algorithm 35 Converter $\mathsf{ConfAnon}$.

▷ $(P \in \overline{\mathcal{P}^H})$-READ
  list $\leftarrow \emptyset$
  for $(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m)) \in$ READ **:**
    if $(A_i, \vec{V}) \in \mathcal{S}^H \times (\mathcal{R}^H)^+$**:** list $\leftarrow$ list.ADD$(\mathtt{id}, (|\vec{V}|, |m|))$
    else: list.ADD$(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))$
  OUTPUT(list)

I.4 Application Semantics of SCBC Game-Based Notions

Theorem 32. *Consider simulator* sim *defined in Algorithms 36 and 37, reductions* $\mathbf{C}^{\mathsf{RCorr}}$, $\mathbf{C}^{\mathsf{Cons}}$, $\mathbf{C}^{\mathsf{aRob}}$, $\mathbf{C}^{\mathsf{Corr}}$, $\mathbf{C}^{\mathsf{RUnforg}}$ *and* $\mathbf{C}^{\mathsf{CCA}}$ *(defined, respectively, in Algorithms 38 and 39, Algorithms 38 and 40, Algorithms 38 and 41, Algorithms 38 and 42, Algorithms 38 and 43, and Algorithms 38 and 44). If the* SCBC *scheme is* (m, n)*-Party* ε*-Public Key Collision Resistant, then for any distinguisher* $\mathbf{D}$*,*

$$\begin{aligned}
&\Delta^{\mathbf{D}}\Bigg(\mathsf{Snd}^{\mathcal{S}^H}\mathsf{Rcv}^{\mathcal{R}^H}[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}],\\
&\mathsf{sim}^{\overline{\mathcal{P}^H}} \cdot \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \cdot \mathsf{Net} \cdot \bot^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle_{\vec{V}\cup\overline{\mathcal{P}^H}}^{\{A_i\}\cup\overline{\mathcal{P}^H}}\right]_{\substack{A_i\in\mathcal{S}\\ \vec{V}\in\mathcal{R}^+}}\Bigg)\\
&\leq Adv^{\neg\mathsf{RCorr}\vee\neg\mathsf{aRCorr}}(\mathbf{DC}^{\mathsf{RCorr}}) + Adv^{\neg\mathsf{Cons}\vee\neg\mathsf{Self\text{-}Cons}}(\mathbf{DC}^{\mathsf{Cons}}) + Adv^{\mathsf{aRob}}(\mathbf{DC}^{\mathsf{aRob}})\\
&\quad + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}}) + Adv^{\mathsf{RUnforg}}(\mathbf{DC}^{\mathsf{RUnforg}}) + Adv^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}\text{-}2^{\mathsf{adap}}}(\mathbf{DC}^{\mathsf{CCA}}) + \varepsilon.
\end{aligned}$$

In Algorithm 36 we describe the behavior of part of the simulator we consider in the proof below (i.e. of Theorem 32) for the (sub-)interfaces of dishonest parties that correspond to an interface of the $\mathbf{KGA}$ resource in the real world system. Similarly, we describe the behavior of the reductions we consider in the proof below for the same (sub-)interfaces in Algorithm 38.

Proof. Let $\mathbf{R}$ be the real world system $\mathbf{R} \coloneqq \mathsf{Snd}^{\mathcal{S}^H}\mathsf{Rcv}^{\mathcal{R}^H}[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}]$, $\mathbf{T}$ be the ideal repository defined in Equation I.3, i.e.

$$\mathbf{T} \coloneqq \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \cdot \mathsf{Net} \cdot \bot^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle_{\vec{V}\cup\overline{\mathcal{P}^H}}^{\{A_i\}\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}, \tag{I.4}$$

Algorithm 36 Description of Theorem 32's simulator (more in Algorithm 37).

◇ Initialization
  INS-Initialization
  $(\text{CtxtOfId}, \text{IdOfCtxt}) \leftarrow (\emptyset, \emptyset)$
  $\mathtt{pp} \leftarrow \mathsf{Setup}(1^k)$
  for $A_i \in \mathcal{S}$**:** $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathsf{Gen}_{\mathsf{S}}(\mathtt{pp})$
  for $B_j \in \mathcal{R}$**:** $(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow \mathsf{Gen}_{\mathsf{R}}(\mathtt{pp})$
  if $(|\{\mathtt{spk}_i\}_{i\in\mathcal{S}}|, |\{\mathtt{rpk}_i\}_{i\in\mathcal{R}}|) \neq (|\mathcal{S}|, |\mathcal{R}|)$ **:**
    Abort

▷ $(P \in \overline{\mathcal{P}^H})$-PublicParams: Output($\mathtt{pp}$)
▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$): Output($\mathtt{spk}_i, \mathtt{ssk}_i$)
▷ $(P \in \overline{\mathcal{P}^H})$-SenderPublicKey($A_i \in \mathcal{S}$): Output($\mathtt{spk}_i$)
▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$): Output($\mathtt{rpk}_j, \mathtt{rsk}_j$)
▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverPublicKey($B_j \in \mathcal{R}$): Output($\mathtt{rpk}_j$)

◇ Decryption(B_j, c)
  $(\mathtt{spk}, \vec{v}, m) \leftarrow \mathsf{Dec}_{\mathtt{pp}}(\mathtt{rsk}_j, c)$
  if $(\mathtt{spk}, \vec{v}, m) \neq \perp$ **:**
    $\langle A_i \to \vec{V}\rangle \leftarrow$ GetLabel($\mathtt{spk}, \vec{v}$)
    if $\langle A_i \to \vec{V}\rangle \neq \perp$ **:**
      return $(\langle A_i \to \vec{V}\rangle, m)$
  return $\perp$

◇ GetLabel($\mathtt{spk}, \vec{v}$)
  $\mathcal{S}_{\mathtt{spk}} \coloneqq \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
  if $|\mathcal{S}_{\mathtt{spk}}| \neq 1$**: return** $\perp$
  for $l \in \{1, \ldots, |\vec{v}|\}$ **:**
    $\mathcal{R}_{v_l} \coloneqq \{B_k \mid v_l = \mathtt{rpk}_k\}$
    if $|\mathcal{R}_{v_l}| \neq 1$**: return** $\perp$
    else: Let V_l be the element of $\mathcal{R}_{v_l}$
  Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
  return $\langle A_i \to \vec{V} \coloneqq (V_1, \ldots, V_{|\vec{v}|})\rangle$

Algorithm 37 Remainder of Theorem 32's simulator description.

▷ $(P \in \overline{\mathcal{P}^H})$-Write($c$)
  if $c \in$ IdOfCtxt**:** Output(IdOfCtxt[c])
  else
    for $j = 1, \ldots, |\mathcal{R}|$ **with** $B_j \in \mathcal{R}^H$ **:**
      $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ Decryption(B_j, c)
      if $((\langle A_i \to \vec{V}\rangle, m) \neq \perp) \wedge (A_i \in \overline{\mathcal{S}^H})$ **:**
        $\mathtt{id} \leftarrow$ **T**-Write($\langle A_i \to \vec{V}\rangle, m$)
        $(\text{IdOfCtxt}[c], \text{CtxtOfId}[\mathtt{id}]) \leftarrow (\mathtt{id}, c)$
        Output($\mathtt{id}$)
    $\mathtt{id} \leftarrow$ **INS**-Write(c) ▷ No $B_j \in \mathcal{R}^H$ could decrypt c
    $(\text{IdOfCtxt}[c], \text{CtxtOfId}[\mathtt{id}]) \leftarrow (\mathtt{id}, c)$
    Output($\mathtt{id}$)

▷ $(P \in \overline{\mathcal{P}^H})$-Read
  for $(\langle A_i \to \vec{V}\rangle, \mathtt{id}, m) \in$ **T**-Read **:**
    if $\mathtt{id} \notin$ CtxtOfId **:**
      $c \leftarrow \mathsf{Enc}_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v} \coloneqq (\mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{V}|}), m)$
      if $c \in$ IdOfCtxt**:** Abort
      $(\text{IdOfCtxt}[c], \text{CtxtOfId}[\mathtt{id}]) \leftarrow (\mathtt{id}, c)$
  for $(\mathtt{id}, (l, l')) \in$ **T**-Read **:** ▷ $(l, l') \in \mathbb{N} \times \mathbb{N}$
    if $\mathtt{id} \notin$ Ctxts **:**
      $(A_i, \vec{V}) \xleftarrow{\$} (\mathcal{S}^H, (\mathcal{R}^H)^l)$ ▷ $\vec{V}$'s sampling is conditioned on not having duplicate entries, i.e. $|\text{Set}(\vec{V})| = |\vec{V}| = l$.
      $c \leftarrow \mathsf{Enc}_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v} \coloneqq (v_1, \ldots, v_l), 0^{l'})$
      if $c \in$ IdOfCtxt**:** Abort
      $(\text{IdOfCtxt}[c], \text{CtxtOfId}[\mathtt{id}]) \leftarrow (\mathtt{id}, c)$
  Output(CtxtOfId) ▷ CtxtOfId is a set of pairs $(\mathtt{id}, c)$

Algorithm 38 Description of the common part of Theorem 32's reductions.

◇ Initialization
  INS-Initialization
  $(\text{CtxtDec}, \text{RcvCtxtDec}) \leftarrow (\emptyset, \emptyset)$
  $(\text{CtxtHon}, \text{CtxtDis}) \leftarrow (\emptyset, \emptyset)$
  for $A_i \in \mathcal{S}$**:** $\mathtt{spk}_i \leftarrow \mathcal{O}_{SPK}(A_i)$
  for $B_j \in \mathcal{R}$**:** $\mathtt{rpk}_j \leftarrow \mathcal{O}_{RPK}(B_j)$
  for $B_j \in \mathcal{R}$**:** $\text{Received}[B_j] \leftarrow \emptyset$
  if $(|\{\mathtt{spk}_i\}_{i\in\mathcal{S}}|, |\{\mathtt{rpk}_i\}_{i\in\mathcal{R}}|) \neq (|\mathcal{S}|, |\mathcal{R}|)$ **:**
    Abort

▷ Deliver(P, id): Received[P].Add(id)
▷ ($P \in \overline{\mathcal{P}^H}$)-PublicParams: Output($\mathcal{O}_{PP}$)
▷ ($P \in \overline{\mathcal{P}^H}$)-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$): Output($\mathcal{O}_{SK}(A_i)$)
▷ ($P \in \overline{\mathcal{P}^H}$)-SenderPublicKey($A_i \in \mathcal{S}$): Output($\mathcal{O}_{SPK}(A_i)$)
▷ ($P \in \overline{\mathcal{P}^H}$)-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$): Output($\mathcal{O}_{RK}(B_j)$)
▷ ($P \in \overline{\mathcal{P}^H}$)-ReceiverPublicKey($B_j \in \mathcal{R}$): Output($\mathcal{O}_{RPK}(B_j)$)
▷ ($P \in \overline{\mathcal{P}^H}$)-Read: Output(Read)

◇ Decryption(B, c)
  $(\mathtt{spk}, \vec{v}, m) \leftarrow \mathcal{O}_D(B, c)$
  if $(\mathtt{spk}, \vec{v}, m) \neq \perp$ **:**
    $\langle A_i \to \vec{V}\rangle \leftarrow$ GetLabel($\mathtt{spk}, \vec{v}$)
    if $\langle A_i \to \vec{V}\rangle \neq \perp$ **:**
      return $(\langle A_i \to \vec{V}\rangle, m)$
  return $\perp$

◇ GetDelivered(P, list)
  filteredList $\leftarrow \emptyset$
  for $(\mathtt{id}, x) \in$ list **with** $\mathtt{id} \in \text{Received}[P]$ **:**
    filteredList.Add($\mathtt{id}, x$)
  return filteredList

◇ GetLabel($\mathtt{spk}, \vec{v}$)
  $\mathcal{S}_{\mathtt{spk}} := \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
  if $|\mathcal{S}_{\mathtt{spk}}| \neq 1$**:** **return** $\perp$
  for $l \in \{1, \ldots, |\vec{v}|\}$ **:**
    $\mathcal{R}_{v_l} := \{B_k \mid v_l = \mathtt{rpk}_k\}$
    if $|\mathcal{R}_{v_l}| \neq 1$**:** **return** $\perp$
    else: Let V_l be the element of $\mathcal{R}_{v_l}$
  Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
  return $\langle A_i \to \vec{V} := (V_1, \ldots, V_{|\vec{v}|})\rangle$

Algorithm 39 Reduction $\mathbf{C}^{\mathsf{RCorr}}$ for Theorem 32.

▷ ($A_i \in \mathcal{S}^H$)-Write($\langle A_i \to \vec{V}\rangle, m$)
  $c \leftarrow \mathcal{O}_E(A_i, \vec{V}, m)$
  if $c \in \text{CtxtHon} \cup \text{CtxtDis}$ **:** $\neg\mathsf{RCorr} \vee \neg\mathsf{aRCorr}$
  CtxtHon.Add(c)
  Output(Write(c))

▷ ($P \in \overline{\mathcal{P}^H}$)-Write($c$)
  if $c \notin \text{CtxtHon}$ **:**
    CtxtDis.Add(c)
    $\mathcal{O}_{Publish}(c)$
  Output(Write(c))

▷ ($B_j \in \mathcal{R}^H$)-Read
  list, ctxtSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ Read **with** $c \notin$ ctxtSet **:**
    ctxtSet.Add(c)
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ Decryption(B_j, c)
    if $(\langle A_i \to \vec{V}\rangle, m) \neq \perp$ **:**
      list.Add($\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m)$)
  Output(GetDelivered(B_j, list))

Algorithm 40 Reduction $\mathbf{C}^{\mathsf{Cons}}$ for Theorem 32. (We only show the differences from the previous hybrid.)

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E(A_i, \vec{V}, m)$
  if $c \in$ CtxtHon $\cup$ CtxtDis **:** ABORT
  CtxtHon.ADD(c)
  SETDECRYPTION(c)
  OUTPUT(WRITE(c))

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  if $c \notin$ CtxtHon **:**
    if $c \notin$ CtxtDis **:** SETDECRYPTION(c)
    CtxtDis.ADD(c)
  OUTPUT(WRITE(c))

◇ SETDECRYPTION(c)
  for $j = 1, \ldots, |\mathcal{R}|$ **with** $(B_j \in \mathcal{R}^H) \wedge (B_j \notin$ RcvCtxtDec$[c])$ **:**
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ DECRYPTION(B_j, c)
    if $(\langle A_i \to \vec{V}\rangle, m) = \bot$ **:** RcvCtxtDec$[c][B_j] \leftarrow \bot$
    else if $\exists B_l \in \vec{V}^H$**:** $(l < j) \wedge$ (RcvCtxtDec$[c][B_l] \neq (\langle A_i \to \vec{V}\rangle, m)$)**:**
      RcvCtxtDec$[c][B_j] \leftarrow \bot$
    else
      for $B_l \in \vec{V}^H$**:** RcvCtxtDec$[c][B_l] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$ ▷ Set Dec for each $B_l \in \vec{V}^H$

Algorithm 41 Reduction $\mathbf{C}^{\mathsf{aRob}}$ for Theorem 32. (We only show the differences from the previous hybrid.)

▷ $(B_j \in \mathcal{R}^H)$-READ
  list, ctxtSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ctxtSet **:**
    ctxtSet.ADD(c)
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ RcvCtxtDec$[c][B_j]$
    if $((\langle A_i \to \vec{V}\rangle, m) \neq \bot) \wedge (B_j \in \vec{V})$ **:** list.ADD$(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))$
  OUTPUT(GETDELIVERED$(B_j$, list$)$)

◇ SETDECRYPTION(c)
  for $j = 1, \ldots, |\mathcal{R}|$ **with** $(B_j \in \mathcal{R}^H) \wedge (B_j \notin$ RcvCtxtDec$[c])$ **:**
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ DECRYPTION(B_j, c)
    if $(\langle A_i \to \vec{V}\rangle, m) = \bot$ **:** RcvCtxtDec$[c][B_j] \leftarrow \bot$
    else if $B_j \notin \vec{V}$ **:** ABORT
    else if $\exists B_l \in \vec{V}^H$**:** $(l < j) \wedge$ (RcvCtxtDec$[c][B_l] \neq (\langle A_i \to \vec{V}\rangle, m)$) **:**
      RcvCtxtDec$[c][B_j] \leftarrow \bot$
    else ▷ $B_j \in \vec{V}^H$
      for $B_l \in \vec{V}^H$**:** RcvCtxtDec$[c][B_l] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$ ▷ Set Dec for each $B_l \in \vec{V}^H$
    if $(c \notin$ CtxtDec$) \wedge ((\langle A_i \to \vec{V}\rangle, m) \neq \bot)$ **:** CtxtDec$[c] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  if $c \notin$ CtxtDec **:** CtxtDec$[c] \leftarrow \bot$

Algorithm 42 Reduction $\mathbf{C}^{\mathsf{Corr}}$ for Theorem 32. (We only show the differences from the previous hybrid.)

▷ $(B_j \in \mathcal{R}^H)$-READ
  list, ctxtSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ctxtSet **:**
    ctxtSet.ADD(c)
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ CtxtDec$[c]$
    if $((\langle A_i \to \vec{V}\rangle, m) \neq \bot) \wedge (B_j \in \vec{V})$ **:** list.ADD$(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))$
  OUTPUT(GETDELIVERED$(B_j,$ list))

◇ SETDECRYPTION(c)
  for $j = 1, \ldots, |\mathcal{R}|$ **with** $(B_j \in \mathcal{R}^H)$ [redacted] **:**
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ DECRYPTION(B_j, c)
    if $((\langle A_i \to \vec{V}\rangle, m) \neq \bot) \wedge (B_j \notin \vec{V})$ **:** ABORT
    if $(c \notin$ CtxtDec$) \wedge ((\langle A_i \to \vec{V}\rangle, m) \neq \bot)$ **:** CtxtDec$[c] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  if $c \notin$ CtxtDec **:** CtxtDec$[c] \leftarrow \bot$

Algorithm 43 Reduction $\mathbf{C}^{\mathsf{RUnforg}}$ for Theorem 32. (We only show the differences from the previous hybrid.)

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E(A_i, \vec{V}, m)$
  if $c \in$ CtxtHon $\cup$ CtxtDis **:** ABORT
  CtxtHon.ADD(c)
  CtxtDec$[c] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  OUTPUT(WRITE(c))

Algorithm 44 Reduction $\mathbf{C}^{\mathsf{CCA}}$ for Theorem 32. (We only show the differences from the previous hybrid.)

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $(A, \vec{B}) \xleftarrow{\$} (\mathcal{S}^H, (\mathcal{R}^H)^{|\vec{V}|})$ ▷ $\vec{B}$'s sampling is conditioned on not having duplicate entries, i.e. $|\mathrm{Set}(\vec{B})| = |\vec{B}|$.
  $\alpha_{\mathbf{0}} := (A_i, \vec{V}, m)$, $\alpha_{\mathbf{1}} := (A, \vec{B}, 0^{|m|})$
  if $(A_i \notin \mathcal{S}^H) \vee (\vec{V} \notin (\mathcal{R}^H)^+)$ **:** $\alpha_{\mathbf{1}} := \alpha_{\mathbf{0}}$
  $c \leftarrow \mathcal{O}_E(\alpha_{\mathbf{0}}, \alpha_{\mathbf{1}})$
  if $c \in$ CtxtHon $\cup$ CtxtDis **:** ABORT
  CtxtHon.ADD(c)
  CtxtDec$[c] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  OUTPUT(WRITE(c))

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  if $c \notin$ CtxtHon **:**
    if $c \notin$ CtxtDis **:**
      SETDECRYPTION(c)
      $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ CtxtDec$[c]$
      if $((\langle A_i \to \vec{V}\rangle, m) \neq \bot) \wedge (A_i \in \mathcal{S}^H)$ **:** ABORT
    CtxtDis.ADD(c)
  OUTPUT(WRITE(c))

and let sim be the simulator specified in Algorithms 36 and 37. The remainder of the proof bounds $\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T})$ by proceeding in a sequence of hybrids. In the following, we consider the reduction systems defined in the theorem's statement.

$\mathbf{R} \rightsquigarrow \mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}$**:** $\mathbf{R}$ and $\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}$ are the same sequence of conditional probability distributions conditioned on no two parties having the same public key by considering, on one hand, the definition of $\mathbf{R}$—i.e. the definitions of converters Snd and Rcv (Algorithm 34), the definition of the $\mathbf{KGA}$ resource (Algorithm 33), and the definitions of $\mathbf{INS}$ (Algorithm 30) and of Net (Algorithm 32)—and, on the other hand, the definition of $\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}$—i.e. the definition of $\mathbf{G}^{\mathsf{RCorr}}$ (Definition 12) and the definition of $\mathbf{C}^{\mathsf{RCorr}}$ (Algorithms 38 and 39). Since by assumption the SCBC scheme is (m, n)-Party ε-Public Key Collision Resistant and there are m senders and n receivers, it follows

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}) \leq \varepsilon.$$

$\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$**:** $\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}$ and $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ are the same sequence of conditional probability distributions conditioned on $\neg\mathsf{RCorr} \vee \neg\mathsf{aRCorr}$ not occurring: in particular note that $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ only aborts if a query $\mathcal{O}_E$ outputs a previously defined ciphertext c, matching the condition for $\neg\mathsf{RCorr} \vee \neg\mathsf{aRCorr}$ to occur. It follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \leq Adv^{\neg\mathsf{RCorr}\vee\neg\mathsf{aRCorr}}(\mathbf{D}\mathbf{C}^{\mathsf{RCorr}}).$$

$\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$**:** The only difference between $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ and $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ is that in $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ the decryption of every ciphertext—either generated by a WRITE operation at the interface of an honest sender $A_i \in \mathcal{S}^H$ or input to a WRITE operation at the interface of a dishonest party $P \in \overline{\mathcal{P}^H}$—is guaranteed to be Consistency and Self-Consistency. (For more details see Algorithms 40, 41 and 38). Conditioned on $\neg\mathsf{Self\text{-}Cons} \vee \neg\mathsf{Cons}$ not occurring, $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ are the same; the advantage of a distinguisher $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ is bounded by the advantage of adversary $\mathbf{D}\mathbf{C}^{\mathsf{Cons}}$ in causing $\neg\mathsf{Self\text{-}Cons} \vee \neg\mathsf{Cons}$ to occur: note that $\mathbf{C}^{\mathsf{Cons}}$ makes a query to $\mathcal{O}_D$ on each honest party $B \in \mathcal{R}^H$ when queried for any (honest or dishonest) WRITE operation, and when queried for a READ operation at the interface of a receiver $B_j \in \mathcal{R}^H$ makes a query to $\mathcal{O}_D$ for each $(\mathtt{id}, c)$ in the reduction's internal repository $\mathbf{INS}$. It follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}) \leq Adv^{\neg\mathsf{Self\text{-}Cons}\vee\neg\mathsf{Cons}}(\mathbf{D}\mathbf{C}^{\mathsf{Cons}}).$$

$\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}} \rightsquigarrow \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$**:** The only difference between $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ is that in $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ once the decryption of each ciphertext c succeeds, and letting $\vec{V}$ be the corresponding vector of receivers (obtained from decrypting c), it is assumed that the decryption of c by any party $B_j \in \mathcal{R}^H$ who is not in $\vec{V}$—so, $B_j \in \mathcal{R}^H \setminus \vec{V}$—fails. (See Algorithm 42). Noting $\mathbf{C}^{\mathsf{aRob}}$ does not query for the secret key of any receiver $B_j \in \mathcal{R}^H$, the advantage of a distinguisher $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}$ and $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ is bounded by the advantage of adversary

$\mathbf{DC}^{\mathsf{aRob}}$ in winning $\mathbf{G}^{\mathsf{aRob}}$: note that $\mathbf{C}^{\mathsf{aRob}}$, upon any WRITE operation, makes a query to $\mathcal{O}_D$ on each honest party $B \in \mathcal{R}^H$ whose public key was not output by a prior $\mathcal{O}_D$ query (on the same ciphertext). It follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \leq Adv^{\mathsf{aRob}}(\mathbf{DC}^{\mathsf{aRob}}).$$

$\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}$**:** System $\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}$ differs from $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ in that ciphertexts generated by WRITE operations issued at the interface of honest senders are no longer decrypted and instead the result of their decryption is simply assumed to be the correct label-message pair. Since at this point we are assuming no two parties have the same public key, $\mathbf{D}$'s advantage in distinguishing $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}$ is upper bounded by the advantage of $\mathbf{DC}^{\mathsf{Corr}}$ in winning the correctness game $\mathbf{G}^{\mathsf{Corr}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}) \leq Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}}).$$

$\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}} \rightsquigarrow \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}$**:** The main things to note for this step are that

1. $\mathbf{D}$ has no access to the secret key of any honest sender $A_i \in \mathcal{S}^H$ nor any honest receiver $B_j \in \mathcal{R}^H$;
2. the only case in which $\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}$ may differ from $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}$ is if $\mathbf{D}$ makes a query for a WRITE operation at the interface of a dishonest party $P \in \overline{\mathcal{S}^H} \cup \overline{\mathcal{R}^H}$ with input ciphertext c whose decryption results in a label $\langle A_i \rightarrow \vec{V} \rangle$ where $A_i \in \mathcal{S}^H$ and yet there was no WRITE operation at the interface of A_i that resulted in ciphertext c.

We can then bound the advantage of $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}$ and $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}$ by the advantage of $\mathbf{DC}^{\mathsf{RUnforg}}$ in winning $\mathbf{G}^{\mathsf{RUnforg}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}) \leq Adv^{\mathsf{RUnforg}}(\mathbf{DC}^{\mathsf{RUnforg}}).$$

$\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\mathsf{CCA}} \rightsquigarrow \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}$**:** It is easy to see that $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\mathsf{CCA}}$ and $\mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}$ are perfectly indistinguishable, and so $\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\mathsf{CCA}}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) = 0$.

To conclude the proof we use triangle inequality:

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) &\leq \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}) + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{RCorr}}\mathbf{G}^{\mathsf{RCorr}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}) + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{aRob}}\mathbf{G}^{\mathsf{aRob}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}) + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{RUnforg}}\mathbf{G}^{\mathsf{RUnforg}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\mathsf{CCA}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\mathsf{CCA}}) + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\mathsf{CCA}}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) \\
&\leq \varepsilon + Adv^{\neg\mathsf{RCorr}\vee\neg\mathsf{aRCorr}}(\mathbf{DC}^{\mathsf{RCorr}}) + Adv^{\neg\mathsf{Self\text{-}Cons}\vee\neg\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}}) + Adv^{\mathsf{aRob}}(\mathbf{DC}^{\mathsf{aRob}}) \\
&\quad Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}}) + Adv^{\mathsf{RUnforg}}(\mathbf{DC}^{\mathsf{RUnforg}}) + Adv^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}\text{-}2^{\mathsf{adap}}}(\mathbf{DC}^{\mathsf{CCA}}).
\end{aligned}$$

J Useful Facts

J.1 Public-Key Collision Resistance of EUF-CMA$^{\text{adap}}$ Secure DSS

Lemma 1. *If* $\mathsf{DSS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is*

$$(\varepsilon_{\mathsf{Corr}}, \varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}, \varepsilon_{\mathsf{OT\text{-}sEUF\text{-}CMA}}, \varepsilon_{\mathsf{PK\text{-}Coll\text{-}Res}}, t, n, q_S, q_V)\text{-}\textit{secure},$$

with $t \gtrapprox n \cdot t_{\mathsf{Gen}} + t_{\mathsf{Vrfy}}$ *—where* t_{Gen} *and* t_{Vrfy} *are, respectively, the times to run* $\mathsf{DSS.Gen}$ *and* $\mathsf{DSS.Vrfy}$ *—then* DSS *is* n*-Party* ε*-Public Key Collision-Resistant, with* $\varepsilon \leq \varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}} + \varepsilon_{\mathsf{Corr}}$.

Proof. We give two adversaries $\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$ and $\mathbf{A}^n_{\mathsf{Corr}}$ such that

$$Adv^{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}(\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}) + Adv^{\mathsf{Corr}}(\mathbf{A}^n_{\mathsf{Corr}}) \geq \varepsilon.$$

Since $Adv^{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}(\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}) \leq \varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$ and $Adv^{\mathsf{Corr}}(\mathbf{A}^n_{\mathsf{Corr}}) \leq \varepsilon_{\mathsf{Corr}}$, it follows $\varepsilon \leq \varepsilon_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}} + \varepsilon_{\mathsf{Corr}}$.

$\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$ keeps track of a mapping IndexOfPK that maps public keys to an index $i \in [n]$; this map is initially empty; for each $i \in [n]$ (in increasing order), $\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$ queries $\mathcal{O}_{VK}(i)$, obtaining some public key $\mathtt{vk}_i$; if $\mathtt{vk}_i \notin$ IndexOfPK (i.e. if there is no entry in the IndexOfPK map for $\mathtt{vk}_i$) the adversary adds a mapping from $\mathtt{vk}_i$ to i; otherwise the adversary queries $\mathcal{O}_S(i, 0)$ to obtain a signature σ a zero-bitstring, and then makes a query $\mathcal{O}_V(\text{IndexOfPK}[\mathtt{vk}_i], 0, \sigma)$ and finishes.

$\mathbf{A}^n_{\mathsf{Corr}}$ is identical, but instead of making query $\mathcal{O}_V(\text{IndexOfPK}[\mathtt{vk}_i], 0, \sigma)$, makes query $\mathcal{O}_V(i, 0, \sigma)$.

It is easy to see that if there are two indices with the same public key, then the only case $\mathbf{A}^n_{\mathsf{EUF\text{-}CMA}^{\mathsf{adap}}}$ may not win is if the DSS is not correct. □

J.2 Corr + aRob ⇒ Rob

The $\{\mathsf{IND}, \mathsf{IK}\}$-CCA$^{\mathsf{adap}}$ security proof of both our SCBC construction and Maurer et al.'s black-box MDRS-PKE construction assume a PKEBC scheme satisfying Robustness (Definition 8, [51,10]). However, we have only proved the Adversarial Robustness of our PKEBC construction. To close the gap, we now show that any scheme $\mathsf{PKEBC} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfying Adversarial Robustness and Correctness also satisfies Robustness.

Lemma 2. *If no adversary* $(\varepsilon_{\mathsf{Corr}}, t_{\mathsf{Corr}})$*-breaks* PKEBC*'s* $(n_{\mathsf{Corr}}, d_{E\mathsf{Corr}}, q_{E\mathsf{Corr}}, q_{D\mathsf{Corr}}, q_{P\mathsf{Corr}})$*-Correctness with* $q_{E\mathsf{Corr}} < q_{D\mathsf{Corr}}$ *nor* $(\varepsilon_{\mathsf{aRob}}, t_{\mathsf{aRob}})$*-breaks* PKEBC*'s* $(n_{\mathsf{aRob}}, d_{E\mathsf{aRob}}, q_{E\mathsf{aRob}}, q_{D\mathsf{aRob}}, q_{P\mathsf{aRob}})$*-Adversarial Robustness with* $q_{E\mathsf{aRob}} < q_{D\mathsf{aRob}}$ *then no adversary* (ε, t)*-breaks* PKEBC*'s*

$(n \coloneqq \min(n_{\mathsf{Corr}}, n_{\mathsf{aRob}}), d_E \coloneqq \min(d_{E\mathsf{Corr}}, d_{E\mathsf{aRob}}), q_E \coloneqq \min(q_{E\mathsf{Corr}}, q_{E\mathsf{aRob}}),$
$q_D \coloneqq \min(q_{D\mathsf{Corr}} - q_{E\mathsf{Corr}}, q_{D\mathsf{aRob}} - q_{E\mathsf{aRob}}), q_P \coloneqq \min(q_{P\mathsf{Corr}}, q_{P\mathsf{aRob}}))$*-Robustness,*

with $\varepsilon > \varepsilon^{\mathsf{PKEBC}}_{\mathsf{Corr}} + \varepsilon^{\mathsf{PKEBC}}_{\mathsf{aRob}}$*, and* $t_{\mathsf{Corr}}, t_{\mathsf{aRob}} \approx t$.

Proof. For an adversary $\mathbf{A}$ (ε, t)-breaking PKEBC's (n, d_E, q_E, q_D, q_P)-Robustness, we give adversaries $\mathbf{A}_{\mathsf{Corr}}$ and $\mathbf{A}_{\mathsf{aRob}}$ such that either $\mathbf{A}_{\mathsf{Corr}}$ $(\varepsilon_{\mathsf{Corr}}, t_{\mathsf{Corr}})$-breaks PKEBC's $(n_{\mathsf{Corr}}, d_{E\,\mathsf{Corr}}, q_{E\,\mathsf{Corr}}, q_{D\,\mathsf{Corr}}, q_{P\,\mathsf{Corr}})$-Correctness or $\mathbf{A}_{\mathsf{aRob}}$ $(\varepsilon_{\mathsf{aRob}}, t_{\mathsf{aRob}})$-breaks PKEBC's $(n_{\mathsf{aRob}}, d_{E\,\mathsf{aRob}}, q_{E\,\mathsf{aRob}}, q_{D\,\mathsf{aRob}}, q_{P\,\mathsf{aRob}})$-Adversarial Robustness. Both $\mathbf{A}_{\mathsf{Corr}}$ and $\mathbf{A}_{\mathsf{aRob}}$ simply run $\mathbf{A}$ internally, forwarding every oracle queries $\mathbf{A}$ makes to the oracles provided by the games they are interacting with; for each query $\mathcal{O}_E(\vec{V}, m)$ that $\mathbf{A}$ makes, $\mathbf{A}_{\mathsf{Corr}}$ and $\mathbf{A}_{\mathsf{aRob}}$ make an additional query $\mathcal{O}_D(V_1, c)$, where c is the ciphertext output by the $\mathcal{O}_E$ query.

By Definition 8, $\mathbf{A}$ only wins $\mathbf{G}^{\mathsf{Rob}}$ if it makes a query $\mathcal{O}_E(\vec{V}, m)$ that outputs a ciphertext c and makes a query $\mathcal{O}_D(B, c)$—where $B \notin \vec{V}$—that outputs some pair $(\vec{v}', m') \neq \perp$. But then, because $\mathbf{A}_{\mathsf{Corr}}$ and $\mathbf{A}_{\mathsf{aRob}}$ make the additional $\mathcal{O}_D$ query, either $\mathbf{A}_{\mathsf{Corr}}$ wins—if the additional $\mathcal{O}_D$ query outputs something other than $(\vec{v}, m)$—or $\mathbf{A}_{\mathsf{aRob}}$ wins—because the additional query $\mathcal{O}_D$ output the correct $(\vec{v}, m)$ pair, implying B's public key $\mathtt{pk}$ is not in the output vector $\vec{v}$ (i.e. $\mathtt{pk} \notin \vec{v}$), and there is another query to $\mathcal{O}_D$ on the same ciphertext and on party B that did output some pair $(\vec{v}', m') \neq \perp$. □