An extended abstract of this paper appears in the Proceedings of the 7th Theory of Cryptography Conference (TCC 2010), Lecture Notes in Computer Science 5978, D. Micciancio ed., Springer, 2010. This is the full version.

Robust Encryption

Michel Abdalla[1] Mihir Bellare[2] Gregory Neven[3]

Abstract

We provide a provable-security treatment of "robust" encryption. Robustness means it is hard to produce a ciphertext that is valid for two different users. Robustness makes explicit a property that has been implicitly assumed in the past. We argue that it is an essential conjunct of anonymous encryption. We show that natural anonymity-preserving ways to achieve it, such as adding recipient identification information before encrypting, fail. We provide transforms that do achieve it, efficiently and provably. We assess the robustness of specific encryption schemes in the literature, providing simple patches for some that lack the property. We explain that robustness of the underlying anonymous IBE scheme is essential for PEKS (Public Key Encryption with Keyword Search) to be consistent (meaning, not have false positives), and our work provides the first generic conversions of anonymous IBE schemes to consistent (and secure) PEKS schemes. Overall our work enables safer and simpler use of encryption.

[1] Departement d'Informatique, École normale supérieure, 45 Rue d'Ulm, 75005 Paris, France. Email: michel.abdalla@ens.fr. URL: `http://www.di.ens.fr/users/mabdalla`. Supported in part by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II, the French ANR-07-SESU-008-01 Project PAMPA, and the French ANR-14-CE28-0003 Project EnBiD.

[2] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: mihir@cs.ucsd.edu. URL: `http://cseweb.ucsd.edu/users/mihir/`. Supported in part by NSF grants CNS-0627779 and CCF-0915675.

[3] Department of Electrical Engineering, Katholieke Universiteit Leuven, and IBM Research – Zurich, Säumerstrasse 4, 8803 Rüschlikon, Switzerland. Email: nev@zurich.ibm.com. URL: `http://www.neven.org`. Supported in part by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II, a Postdoctoral Fellowship from the Research Foundation – Flanders (FWO – Vlaanderen) and by the European Community's Seventh Framework Programme project PrimeLife (grant agreement no. 216483).

Contents

1 Introduction

This paper provides a provable-security treatment of encryption "robustness." Robustness reflects the difficulty of producing a ciphertext valid under two different encryption keys. The value of robustness is conceptual, "naming" something that has been undefined yet at times implicitly (and incorrectly) assumed. Robustness helps make encryption more mis-use resistant. We provide formal definitions of several variants of the goal; consider and dismiss natural approaches to achieve it; provide two general robustness-adding transforms; test robustness of existing schemes and patch the ones that fail; and discuss some applications.

THE DEFINITIONS. Both the PKE and the IBE settings are of interest and the explication is simplified by unifying them as follows. Associate to each identity an *encryption key*, defined as the identity itself in the IBE case and its (honestly generated) public key in the PKE case. The adversary outputs a pair id_0, id_1 of distinct identities. For strong robustness it also outputs a ciphertext C^*; for weak, it outputs a message M^*, and C^* is defined as the encryption of M^* under the encryption key ek_1 of id_1. The adversary wins if the decryptions of C^* under the decryption keys dk_0, dk_1 corresponding to ek_0, ek_1 are *both* non-$\perp$. Both weak and strong robustness can be considered under chosen plaintext or chosen ciphertext attacks, resulting in four notions (for each of PKE and IBE) that we denote WROB-CPA, WROB-CCA, SROB-CPA, SROB-CCA.

WHY ROBUSTNESS? The primary security requirement for encryption is data-privacy, as captured by notions IND-CPA or IND-CCA [GM84, RS92, DDN00, BDPR98, BF03]. Increasingly, we are also seeing a market for *anonymity*, as captured by notions ANO-CPA and ANO-CCA [BBDP01, ABC+08]. Anonymity asks that a ciphertext does not reveal the encryption key under which it was created.

Where you need anonymity, there is a good chance you need robustness too. Indeed, we would go so far as to say that robustness is an essential companion of anonymous encryption. The reason is that without it we would have security without basic communication correctness, likely upsetting our application. This is best illustrated by the following canonical application of anonymous encryption, but shows up also, in less direct but no less important ways, in other applications. A sender wants to send a message to a *particular* target recipient, but, to hide the identity of this target recipient, anonymously encrypts it under her key and broadcasts the ciphertext to a larger group. But as a member of this group I need, upon receiving a ciphertext, to know whether or not I am the target recipient. (The latter typically needs to act on the message.) Of course I can't tell whether the ciphertext is for me just by looking at it since the encryption is anonymous, but decryption should divulge this information. It does, unambiguously, if the encryption is robust (the ciphertext is for me iff my decryption of it is not $\perp$) but otherwise I might accept a ciphertext (and some resulting message) of which I am not the target, creating mis-communication. Natural "solutions," such as including the encryption key or identity of the target recipient in the plaintext before encryption and checking it upon decryption, are, in hindsight, just attempts to add robustness without violating anonymity and, as we will see, don't work.

We were lead to formulate robustness upon revisiting Public key Encryption with Keyword Search (PEKS) [BDOP04]. In a clever usage of anonymity, Boneh, Di Crescenzo, Ostrovsky and Persiano (BDOP) [BDOP04] showed how this property in an IBE scheme allowed it to be turned into a privacy-respecting communications filter. But Abdalla et. al [ABC+08] noted that the BDOP filter could lack *consistency*, meaning turn up false positives. Their solution was to modify the construction. What we observe instead is that consistency would in fact be

$$
\begin{tikzcd}
\text{AI-CCA} \arrow[d] \arrow[dr, "/" marking] & \text{WROB-CCA} + \text{AI-CCA} \arrow[l] \arrow[d] \arrow[dr, "/" marking] & \text{SROB-CCA} + \text{AI-CCA} \arrow[l] \arrow[d] \\
\text{AI-CPA} & \text{WROB-CPA} + \text{AI-CPA} \arrow[l] & \text{SROB-CPA} + \text{AI-CPA} \arrow[l]
\end{tikzcd}
$$

Figure 1: **Relations between notions.** An arrow A $\rightarrow$ B is an implication, meaning every scheme that is A-secure is also B-secure, while a barred arrow A $\not\rightarrow$ B is a separation, meaning that there is a A-secure scheme that is not B-secure. (Assuming of course that there exists a A-secure scheme in the first place.)

provided by the *original* construct if the IBE scheme was robust. PEKS consistency turns out to correspond exactly to communication correctness of the anonymous IBE scheme in the sense discussed above. (Because the PEKS messages in the BDOP scheme are the recipients identities from the IBE perspective.) Besides resurrecting the BDOP construct, the robustness approach allows us to obtain the first consistent IND-CCA-secure PEKS without random oracles.

Sako's auction protocol [Sak00] uses anonymous PKE to hide the bids of losers. We present an attack on fairness whose cause is ultimately a lack of robustness in the anonymous encryption scheme.

All this underscores a number of the claims we are making about robustness: that it is of conceptual value; that it makes encryption more resistant to mis-use; that it has been implicitly (and incorrectly) assumed; and that there is value to making it explicit, formally defining and provably achieving it.

Weak versus strong. The above-mentioned auction protocol fails because an adversary can create a ciphertext that decrypts correctly under any decryption key. Strong robustness is needed to prevent this. Weak robustness (of the underlying IBE) will yield PEKS consistency for honestly-encrypted messages but may allow spammers to bypass all filters with a single ciphertext, something prevented by strong robustness. Strong robustness trumps weak for applications and goes farther towards making encryption mis-use resistant. We have defined and considered the weaker version because it can be more efficiently achieved, because some existing schemes achieve it and because attaining it is a crucial first step in our method for attaining strong robustness.

Achieving robustness. As the reader has surely already noted, robustness (even strong) is trivially achieved by appending the encryption key to the ciphertext and checking for it upon decryption. The problem is that the resulting scheme is not anonymous and, as we have seen above, it is exactly for anonymous schemes that robustness is important. Of course, data privacy is important too. Letting AI-ATK = ANO-ATK + IND-ATK for ATK $\in \{\text{CPA}, \text{CCA}\}$, the target notions of interest are AI-ATK + XROB-ATK for ATK $\in \{\text{CPA}, \text{CCA}\}$ and X $\in \{\text{W}, \text{S}\}$. Figure 1 shows the relations between these notions, which hold for both PKE and IBE. We note in particular that AI-CCA does not imply any form of robustness, refuting the possible impression that CCA-security automatically provides robustness.

Transforms. Towards achieving robustness, it is natural to begin by seeking a general transform that takes an arbitrary AI-ATK scheme and returns a AI-ATK + XROB-ATK one. This allows us to exploit known constructions of AI-ATK schemes, supports modular protocol design and also helps understand robustness divorced from the algebra of specific schemes. Furthermore, there is a natural and promising transform to consider. Namely, before encrypting, append to the message some redundancy, such as the recipient encryption key, a constant, or even a hash of the message, and check for its presence upon decryption. (Adding the redundancy before encrypting rather than after preserves AI-ATK.) Intuitively this should provide robustness because decryption with the "wrong" key will result, if not in rejection, then in recovery of a garbled plaintext, unlikely to possess the correct redundancy.

The truth is more complex. We consider two versions of the paradigm and summarize our findings in Figure 2. In encryption with *unkeyed redundancy*, the redundancy is a function RC of the message and encryption key alone. In this case we show that the method fails spectacularly, not providing even *weak* robustness *regardless of the choice of the function* RC. In encryption with *keyed redundancy*, we allow RC to depend on a key K that is placed in the public parameters of the transformed scheme, out of direct reach of the algorithms of the original scheme. In this form, the method can easily provide weak robustness, and that too with a very simple redundancy function, namely the one that simply returns K.

But we show that even encryption with keyed redundancy fails to provide *strong* robustness. To achieve the latter we have to step outside the encryption with redundancy paradigm. We present a strong robustness conferring transform that uses a (non-interactive) commitment scheme. For subtle reasons, for this transform to work the starting scheme needs to already be weakly robust. If it isn't already, we can make it so via our weak robustness transform.

In summary, on the positive side we provide a transform conferring weak robustness and another conferring strong robustness. Given any AI-ATK scheme the first transform returns a WROB-ATK + AI-ATK one. Given any AI-ATK + WROB-ATK scheme the second transform returns a SROB-ATK + AI-ATK one. In both cases it is for both ATK = CPA and ATK = CCA and in both cases the transform applies to what we call general encryption schemes, of which both PKE and IBE are special cases, so both are covered.

The Fujisaki-Okamoto (FO) transform [FO99] and the Canetti-Halevi-Katz (CHK) transform [CHK04, BCHK07] both confer IND-CCA, and a natural question is whether they confer robustness as well. It turns out that neither transform generically provides strong robustness (SROB-CCA) and CHK does not provide weak (WROB-CCA) either. We do not know whether or not FO provides WROB-CCA.

Robustness of specific schemes. The robustness of existing schemes is important because they might be in use. We ask which specific existing schemes are robust, and, for those that are not, whether they can be made so at a cost lower than that of applying one of our general transforms. The decryption algorithms of most AI-CPA schemes never reject, which means these schemes are not robust, so we focus on schemes that are known to be AI-CCA. This narrows the field quite a bit. The main findings and results we discuss next are summarized in Figure 2.

The Cramer-Shoup ($\mathcal{CS}$) PKE scheme is known to be AI-CCA in the standard model [CS03, BBDP01]. We show that it is WROB-CCA but not SROB-CCA, the latter because encryption with 0 randomness yields a ciphertext valid under any encryption key. We present a modified version $\mathcal{CS}^*$ of the scheme that disallows 0 randomness. It continues to be AI-CCA, and we show is SROB-CCA. Our proof that $\mathcal{CS}^*$ is SROB-CCA builds on the information-theoretic part of the proof of [CS03]. The result does not need to assume hardness of DDH. It relies instead on pre-image security of the underlying hash function for random range points, something not implied by collision-resistance but seemingly possessed by candidate functions. The same approach does not easily extend to variants of the $\mathcal{CS}$ scheme such as the hybrid Damgård-ElGamal scheme as proved secure by Kiltz et al. [KPSY09]. We leave their treatment to future work.

Transform	WROB-ATK	SROB-ATK
Encryption with unkeyed redundancy (EuR)	No	No
Encryption with keyed redundancy (EkR)	Yes	No

Scheme	**setting**	AI-CCA	WROB-CCA	SROB-CCA	**RO model**
$\mathcal{CS}$	PKE	Yes [CS03, BBDP01]	Yes	No	No
$\mathcal{CS}^*$	PKE	Yes	Yes	Yes	No
$\mathcal{DHIES}$	PKE	Yes [ABR01]	Yes	No	Yes
$\mathcal{DHIES}^*$	PKE	Yes	Yes	Yes	Yes
$\mathcal{BF}$	IBE	Yes [BF01, ABC+08]	Yes	Yes	Yes
$\mathcal{BW}$	IBE	Yes [BW06]	No	No	No

Figure 2: **Achieving Robustness.** The first table summarizes our findings on the encryption with redundancy transform. "No" means the method fails to achieve the indicated robustness for *all* redundancy functions, while "yes" means there exists a redundancy function for which it works. The second table summarizes robustness results about some specific AI-CCA schemes.

In the IBE setting, the CCA version $\mathcal{BF}$ of the RO model Boneh-Franklin scheme is AI-CCA [BF01, ABC+08], and we show it is SROB-CCA. The standard model Boyen-Waters scheme $\mathcal{BW}$ is AI-CCA [BW06], but we show it is neither WROB-CCA nor SROB-CCA. Of course it can be made robust via our transforms. We note that the $\mathcal{BF}$ scheme is obtained via the FO transform [FO99] and $\mathcal{BW}$ via the CHK transform [CHK04, BCHK07]. As indicated above, neither transform *generically* provides strong robustness. This doesn't say whether they do or not when applied to specific schemes, and indeed the first does for $\mathcal{BF}$ and the second does not for $\mathcal{BW}$.

$\mathcal{DHIES}$ is a standardized, in-use PKE scheme due to [ABR01], who show that it is AI-CCA. The situation for robustness is analogous to that for $\mathcal{CS}$ discussed above. Namely, we show $\mathcal{DHIES}$ is WROB-CCA but not SROB-CCA (due to the possibility of the randomness in the asymmetric component being 0) and present a modified version $\mathcal{DHIES}^*$ (it disallows 0 randomness and is still AI-CCA) that we show is SROB-CCA. This result assumes (only) a form of collision-resistance from the MAC.

Our coverage is intended to be illustrative rather than exhaustive. There are many more specific schemes about whose robustness one may ask, and we leave these as open questions.

Summary. Protocol design suggests that designers have the intuition that robustness is naturally present. This seems to be more often right than wrong when considering *weak* robustness of *specific* AI-CCA schemes. Prevailing intuition about *generic* ways to add even weak robustness is wrong, yet we show it can be done by an appropriate tweak of these ideas. Strong robustness is more likely to be absent than present in specific schemes, but important schemes can be patched. Strong robustness can also be added generically, but with more work.

Related work. There is growing recognition that robustness is important in applications and worth defining explicitly, supporting our own claims to this end. Thus, the strong correctness requirement for public-key encryption of [BBW06] and the correctness requirement for hidden-vector and predicate encryption of [BW07, KSW08] imply a form of weak robustness. In work that was concurrent to, and independent of, the preliminary version of our work [ABN10], Hofheinz and Weinreb [HW08] introduced a notion of *well-addressedness* of IBE schemes that is just like weak robustness except that the adversary gets the IBE master secret key. These works do not consider or achieves strong robustness, and the last does not treat PKE. Well-addressedness of IBE implies WROB-CCA but does not imply SROB-CCA and, on the other hand, SROB-CCA does not imply well-addressedness. Also in work that was concurrent to, and independent of, the preliminary version of our work [ABN10], Canetti, Kalai, Varia and Wichs [CKVW10] define wrong-key detection for symmetric encryption, which is a form of robustness. The term robustness is also used in multi-party computation to denote the property that corrupted parties cannot prevent honest parties from computing the correct protocol output [GMW87, BOGW88, HM01]. This meaning is unrelated to our use of the word robustness.

Subsequent work. Since the publication of a preliminary version of our work in [ABN08, ABN10], several extensions have appeared in the literature.

Mohassel [Moh10] observes that weak robustness is needed to ensure the chosen-ciphertext security of hybrid constructions and provides several new robustness-adding transforms providing different trade-offs between ciphertext size and computational overhead. He also proposes a new relaxation of robustness, known as *collision-freeness*, which may already be sufficient for certain applications. Informally, collision-freeness states that a ciphertext should not decrypt to the same message under two different decryption keys.

Other security notions related to robustness have also been proposed in [BD09, BDWY12]. While the notion of decryption verifiability in [BDWY12] can be interpreted as a weak form of robustness in the context of encryption schemes, the notion of unambiguity in [BD09] can be seen as an analogue of robustness for signatures.

Libert, Paterson, and Quaglia [LPQ12] show that robustness is important when building anonymous broadcast encryption generically from identity-based encryption. In their construction, the correctness of the broadcast encryption crucially depends on the weak robustness of the underlying identity-based encryption scheme. The relation between robustness and anonymous broadcast encryption was also observed in an earlier work by Barth, Boneh, and Waters [BBW06].

Farshim, Libert, Paterson and Quaglia [FLPQ13] introduce further notions of robustness including a strengthening and simplification of our strong robustness that they call complete robustness. They show that Sako's protocol [Sak00] is still vulnerable to attacks even if it uses a strongly robust encryption scheme, a gap addressed by complete robustness.

Boneh, Raghunathan, and Segev [BRS13] remark that our robustness conferring transforms also applies to function-private identity-based encryption schemes since they do not change the decryption keys and hence preserve function privacy.

Seurin and Treger [ST13] propose a variant of Schnorr-Signed ElGamal encryption [Jak98, TY98], and show that it is both AI-CCA and SROB-CCA. While the proof of AI-CCA relies on the hardness of DDH in the random-oracle model, the proof of SROB-CCA only assumes collision-resistance security of the underlying hash function.

Versions of this paper. A preliminary version of this paper appeared at the Theory of Cryptography Conference 2010 [ABN10]. This full version, apart from containing full proofs for all security statements, adds a discussion about the robustness of other schemes and transforms in Section 6 and Section 7, as well as more details about the application of our results to auctions and searchable encryption in Section 8 and Section 9.

2 Definitions

Notation and conventions. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. The empty string is denoted ε. By $a_1\|\ldots\|a_n$, we denote a string encoding of $a_1,\ldots,a_n$ from which $a_1,\ldots,a_n$ are uniquely recoverable. (Usually, concatenation

proc Initialize

$(pars, msk) \xleftarrow{\$} \mathsf{PG}$; $b \xleftarrow{\$} \{0,1\}$
$S, T, U, V \leftarrow \emptyset$
Return $pars$

proc GetEK(id)

$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

proc GetDK(id)

If $id \notin U$ then return $\perp$
If $id \in S$ then return $\perp$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

proc Dec(C, id)

If $id \notin U$ then return $\perp$
If $(id, C) \in T$ then return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
Return M

proc LR$(id_0^*, id_1^*, M_0^*, M_1^*)$

If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\perp$
If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\perp$
If $|M_0^*| \neq |M_1^*|$ then return $\perp$
$C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], M_b^*)$
$S \leftarrow S \cup \{id_0^*, id_1^*\}$
$T \leftarrow T \cup \{(id_0^*, C^*), (id_1^*, C^*)\}$
Return C^*

proc Finalize(b')

Return $(b' = b)$

Figure 3: Game $\mathrm{AI}_{\mathcal{GE}}$ defining AI-ATK security of general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

suffices.) By $a_1 \| \ldots \| a_n \leftarrow a$, we mean that a is parsed into its constituents $a_1, \ldots, a_n$. Similarly, if $a = (a_1, \ldots, a_n)$ then $(a_1, \ldots, a_n) \leftarrow a$ means we parse a as shown. Unless otherwise indicated, an algorithm may be randomized. By $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ we denote the operation of running A on inputs $x_1, x_2, \ldots$ and fresh coins and letting y denote the output. We denote by $[A(x_1, x_2, \ldots)]$ the set of all possible outputs of A on inputs $x_1, x_2, \ldots$. We assume that an algorithm returns $\perp$ if any of its inputs is $\perp$.

Games. Our definitions and proofs use code-based game-playing [BR06]. Recall that a game —look at Figure 3 for an example— has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A as follows. First, **Initialize** executes and its outputs are the inputs to A. Then A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted G^A, is called the output of the game, and we let "G^A" denote the event that this game output takes value true. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true. Our proofs will use the following.

Lemma 2.1 [**BR06**] *Let* $\mathrm{G}_i, \mathrm{G}_j$ *be identical until* bad *games, and* A *an adversary. Then*

$$\left|\Pr\left[\,\mathrm{G}_i^A\,\right] - \Pr\left[\,\mathrm{G}_j^A\,\right]\right| \;\leq\; \Pr\left[\,\mathrm{G}_j^A \text{ sets } \mathsf{bad}\,\right] . \blacksquare$$

The running time of an adversary is the worst case time of the execution of the adversary with the game defining its security, so that the execution time of the called game procedures is included.

General encryption. We introduce and use general encryption schemes, of which both PKE and IBE are special cases. This allows us to avoid repeating similar definitions and proofs. A *general encryption* (GE) scheme is a tuple $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms. The parameter generation algorithm PG takes no input and returns common parameter $pars$ and a master secret key msk. On input $pars, msk, id$, the key generation algorithm KG produces an encryption key ek and decryption key dk. On inputs $pars, ek, M$, the encryption algorithm Enc produces a ciphertext C encrypting plaintext M. On input $pars, ek, dk, C$, the deterministic

proc Initialize
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$; $U, V \leftarrow \emptyset$
Return $pars$

proc GetEK(id)
$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

proc GetDK(id)
If $id \notin U$ then return $\perp$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

proc Dec(C, id)
If $id \notin U$ then return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
Return M

proc Finalize(M, id_0, id_1) // $\mathrm{WROB}_{\mathcal{GE}}$
If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$M_0 \leftarrow M$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], M_0)$
$M_1 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
Return $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$

proc Finalize(C, id_0, id_1) // $\mathrm{SROB}_{\mathcal{GE}}$
If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$M_0 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_0], \mathsf{DK}[id_0], C)$
$M_1 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
Return $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$

Figure 4: Games $\mathrm{WROB}_{\mathcal{GE}}$ and $\mathrm{SROB}_{\mathcal{GE}}$ defining WROB-ATK and SROB-ATK security (respectively) of general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. The procedures on the left are common to both games, which differ only in their **Finalize** procedures.

decryption algorithm Dec returns either a plaintext message M or $\perp$ to indicate that it rejects. We say that $\mathcal{GE}$ is a public-key encryption (PKE) scheme if $msk = \varepsilon$ and KG ignores its id input. To recover the usual syntax we may in this case write the output of PG as $pars$ rather than $(pars, msk)$ and omit msk, id as inputs to KG. We say that $\mathcal{GE}$ is an identity-based encryption (IBE) scheme if the encryption key created by KG on inputs $pars, msk, id$ only depends on $pars$ and id. To recover the usual syntax we may in this case write the output of KG as dk rather than (ek, dk). It is easy to see that in this way we have recovered the usual primitives. But there are general encryption schemes that are neither PKE nor IBE schemes, meaning the primitive is indeed more general.

Correctness. Correctness of a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ requires that, for all $(pars, msk) \in [\mathsf{PG}]$, all plaintexts M in the underlying message space associated to $pars$, all identities id, and all $(ek, dk) \in [\mathsf{KG}(pars, msk, id)]$, we have $\mathsf{Dec}(pars, ek, dk, \mathsf{Enc}(pars, ek, M)) = M$ with probability one, where the probability is taken over the coins of Enc.

AI-ATK security. Historically, definitions of data privacy (IND) [GM84,RS92,DDN00,BDPR98, BF03] and anonymity (ANON) [BBDP01, ABC+08] have been separate. We are interested in schemes that achieve both, so rather than use separate definitions we follow [BGH07] and capture both simultaneously via game $\mathrm{AI}_{\mathcal{GE}}$ of Figure 3. A cpa adversary is one that makes no **Dec** queries, and a cca adversary is one that might make such queries. The ai-advantage of such an adversary, in either case, is

$$\mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(A) = 2 \cdot \Pr\left[\mathrm{AI}^{A}_{\mathcal{GE}}\right] - 1.$$

We will assume an ai-adversary makes only one **LR** query, since a hybrid argument shows that making q of them can increase its ai-advantage by a factor of at most q.

Oracle **GetDK** represents the IBE key-extraction oracle [BF03]. In the PKE case it is superfluous in the sense that removing it results in a definition that is equivalent up to a factor depending on the number of **GetDK** queries. That's probably why the usual definition has no such oracle. But conceptually, if it is there for IBE, it ought to be there for PKE, and it does

$$
\begin{tikzcd}
\text{AI-CCA} \arrow[d] \arrow[dr, "/" marking] & \text{WROB-CCA} + \text{AI-CCA} \arrow[l] \arrow[d] \arrow[dr, "/" marking] & \text{SROB-CCA} + \text{AI-CCA} \arrow[l] \arrow[d] \\
\text{AI-CPA} & \text{WROB-CPA} + \text{AI-CPA} \arrow[l] & \text{SROB-CPA} + \text{AI-CPA} \arrow[l]
\end{tikzcd}
$$

Figure 5: **Relations between notions.** An arrow A $\rightarrow$ B is an implication, meaning every scheme that is A-secure is also B-secure, while a barred arrow A $\not\rightarrow$ B is a separation, meaning that there is a A-secure scheme that is not B-secure. (Assuming of course that there exists a A-secure scheme in the first place.)

impact concrete security.

The traditional notions of data privacy (IND-ATK) and anonymity (ANO-ATK) are obtained by adding a restriction to the AI-ATK game in Figure 3 so that a **LR** query returns $\perp$ whenever $id_0^* \neq id_1^*$ or $M_0^* \neq M_1^*$, respectively. It is easy to see that ai security is implied by ind security and ano security, i.e., for each ai-atk adversary A, there exist an ind-atk adversary B_1 and an ano-atk adversary B_2 such that $\mathbf{Adv}^{\text{ai-atk}}_{\mathcal{GE}}(A) = \mathbf{Adv}^{\text{ind-atk}}_{\mathcal{GE}}(B_1) + \mathbf{Adv}^{\text{ano-atk}}_{\mathcal{GE}}(B_2)$.

Robustness. Associated to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are games WROB, SROB of Figure 4. As before, a cpa adversary is one that makes no **Dec** queries, and a cca adversary is one that might make such queries. The wrob and srob advantages of an adversary, in either case, are

$$\mathbf{Adv}^{\text{wrob}}_{\mathcal{GE}}(A) = \Pr\left[\text{WROB}^A_{\mathcal{GE}}\right] \quad \text{and} \quad \mathbf{Adv}^{\text{srob}}_{\mathcal{GE}}(A) = \Pr\left[\text{SROB}^A_{\mathcal{GE}}\right].$$

The difference between WROB and SROB is that in the former the adversary produces a message M, and C is its encryption under the encryption key of one of the given identities, while in the latter it produces C directly, and may not obtain it as an honest encryption. It is worth clarifying that in the PKE case the adversary does *not* get to choose the encryption (public) keys of the identities it is targeting. These are honestly and independently chosen, in real life by the identities themselves and in our formalization by the games.

Relations between notions. Figure 5 shows implications and separations in the style of [BDPR98]. We consider each robustness notion in conjunction with the corresponding AI one since robustness is interesting only in this case. The implications are all trivial. The first separation shows that the strongest notion of privacy fails to imply even the weakest type of robustness. The second separation shows that weak robustness, even under CCA, doesn't imply strong robustness. We stress that here an implication A $\rightarrow$ B means that any A-secure, *unaltered*, is B-secure. Correspondingly, a non-implication A $\not\rightarrow$ B means that there is an A-secure that, unaltered, is not B-secure. (It doesn't mean that an A-secure scheme can't be transformed into a B-secure one.) Only a minimal set of arrows and barred arrows is shown; others can be inferred. The picture is complete in the sense that it implies either an implication or a separation between any pair of notions.

RKG	$\mathsf{RC}(K, ek\|M)$	$\mathsf{RV}(K, ek\|M, r)$
Return $K \leftarrow \varepsilon$	Return ε	Return 1
Return $K \leftarrow \varepsilon$	Return 0^k	Return $(r = 0^k)$
Return $K \leftarrow \varepsilon$	Return ek	Return $(r = ek)$
Return $K \leftarrow \varepsilon$	$L \stackrel{\$}{\leftarrow} \{0,1\}^k$; Return $L\|H(L, ek\|M)$	$L\|h \leftarrow r$; Return $(h = H(L, ek\|M))$
Return $K \stackrel{\$}{\leftarrow} \{0,1\}^k$	Return K	Return $(r = K)$
Return $K \stackrel{\$}{\leftarrow} \{0,1\}^k$	Return $H(K, ek\|M)$	Return $(r = H(K, ek\|M))$

Figure 6: Examples of redundancy codes, where the data x is of the form $ek\|M$. The first four are unkeyed and the last two are keyed.

3 Robustness failures of encryption with redundancy

A natural privacy-and-anonymity-preserving approach to add robustness to an encryption scheme is to add redundancy before encrypting, and upon decryption reject if the redundancy is absent. Here we investigate the effectiveness of this encryption with redundancy approach, justifying the negative results discussed in Section 1 and summarized in the first table of Figure 2.

Redundancy codes and the transform. A redundancy code $\mathcal{RED} = (\mathsf{RKG}, \mathsf{RC}, \mathsf{RV})$ is a triple of algorithms. The redundancy key generation algorithm RKG generates a key K. On input K and data x the redundancy computation algorithm RC returns redundancy r. Given K, x, and claimed redundancy r, the deterministic redundancy verification algorithm RV returns 0 or 1. We say that $\mathcal{RED}$ is unkeyed if the key K output by RKG is always equal to ε, and keyed otherwise. The correctness condition is that for all x we have $\mathsf{RV}(K, x, \mathsf{RC}(K, x)) = 1$ with probability one, where the probability is taken over the coins of RKG and RC. (We stress that the latter is allowed to be randomized.)

Given a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a redundancy code $\mathcal{RED} = (\mathsf{RKG}, \mathsf{RC}, \mathsf{RV})$, the *encryption with redundancy transform* associates to them the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are shown on the left side of Figure 7. Note that the transform has the first of our desired properties, namely that it preserves AI-ATK. Also if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$, and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, which means the results we obtain here apply to both settings.

Figure 6 shows example redundancy codes for the transform. With the first, $\overline{\mathcal{GE}}$ is identical to $\mathcal{GE}$, so that the counterexample below shows that AI-CCA does not imply WROB-CPA, justifying the first separation of Figure 5. The second and third rows show redundancy equal to a constant or the encryption key as examples of (unkeyed) redundancy codes. The fourth row shows a code that is randomized but still unkeyed. The hash function H could be a MAC or a collision resistant function. The last two are keyed redundancy codes, the first the simple one that just always returns the key, and the second using a hash function. Obviously, there are many other examples.

SROB failure. We show that encryption with redundancy fails to provide strong robustness for *all* redundancy codes, whether keyed or not. More precisely, we show that for any redundancy code $\mathcal{RED}$ and both ATK $\in$ {CPA, CCA}, there is an AI-ATK encryption scheme $\mathcal{GE}$ such that the scheme $\overline{\mathcal{GE}}$ resulting from the encryption-with-redundancy transform applied to $\mathcal{GE}, \mathcal{RED}$ is not SROB-CPA. We build $\mathcal{GE}$ by modifying a given AI-ATK encryption scheme $\mathcal{GE}^* = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}^*, \mathsf{Dec}^*)$. Let l be the number of coins used by RC, and let $\mathsf{RC}(x; \omega)$ denote the result of executing RC on input x with coins $\omega \in \{0,1\}^l$. Let M^* be a function that given *pars* returns a point in the message space associated to *pars* in $\mathcal{GE}^*$. Then $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ where

Algorithm $\overline{\mathsf{PG}}$
$(pars, msk) \xleftarrow{\$} \mathsf{PG}\,;\ K \xleftarrow{\$} \mathsf{RKG}$
Return $((pars, K), msk)$

Algorithm $\overline{\mathsf{KG}}((pars, K), msk, id)$
$(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return ek

Algorithm $\overline{\mathsf{Enc}}((pars, K), ek, M)$
$r \xleftarrow{\$} \mathsf{RC}(K, ek\|M)$
$C \xleftarrow{\$} \mathsf{Enc}(pars, ek, M\|r)$
Return C

Algorithm $\overline{\mathsf{Dec}}((pars, K), ek, dk, C)$
$M\|r \leftarrow \mathsf{Dec}(pars, ek, dk, C)$
If $\mathsf{RV}(K, ek\|M, r) = 1$ then return M
Else return $\bot$

Algorithm $\mathsf{Enc}(pars, ek, M)$
$C \xleftarrow{\$} \mathsf{Enc}^*(pars, ek, M)$
Return C

Algorithm $\mathsf{Dec}(pars, ek, dk, C)$
$M \leftarrow \mathsf{Dec}^*(pars, ek, dk, C)$
If $M = \bot$ then
$\quad M \leftarrow M^*(pars)\|\mathsf{RC}(\varepsilon, ek\|M^*(pars); 0^l)$
Return M

Algorithm $\mathsf{Enc}(pars, ek, M)$
$C^* \xleftarrow{\$} \mathsf{Enc}^*(pars, ek, M)$
Return $1\|C^*$

Algorithm $\mathsf{Dec}(pars, ek, dk, C)$
$b\|C^* \leftarrow C$
If $b = 1$ then return $\mathsf{Dec}^*(pars, ek, dk, C^*)$
Else return $M^*(pars)\|\mathsf{RC}(C^*, ek\|M^*(pars); 0^l)$

Figure 7: **Left**: Transformed scheme for the encryption with redundancy paradigm. **Top Right**: Counterexample for WROB. **Bottom Right**: Counterexample for SROB.

the new algorithms are shown on the bottom right side of Figure 7. The reason we used 0^l as coins for RC here is that Dec is required to be deterministic.

Our first claim is that the assumption that $\mathcal{GE}^*$ is AI-ATK implies that $\mathcal{GE}$ is too. Our second claim, that $\overline{\mathcal{GE}}$ is not SROB-CPA, is demonstrated by the following attack. For a pair id_0, id_1 of distinct identities of its choice, the adversary A, on input $(pars, K)$, begins with queries $ek_0 \xleftarrow{\$} \mathbf{GetEK}(id_0)$ and $ek_1 \xleftarrow{\$} \mathbf{GetEK}(id_1)$. It then creates ciphertext $C \leftarrow 0 \,\|\, K$ and returns (id_0, id_1, C). We claim that $\mathbf{Adv}^{\mathrm{srob}}_{\overline{\mathcal{GE}}}(A) = 1$. Letting dk_0, dk_1 denote the decryption keys corresponding to ek_0, ek_1 respectively, the reason is the following. For both $b \in \{0,1\}$, the output of $\mathsf{Dec}(pars, ek_b, dk_b, C)$ is $M^*(pars)\|r_b(pars)$ where $r_b(pars) = \mathsf{RC}(K, ek_b\|M^*(pars); 0^l)$. But the correctness of $\mathcal{RED}$ implies that $\mathsf{RV}(K, ek_b\|M^*(pars), r_b(pars)) = 1$ and hence $\overline{\mathsf{Dec}}((pars, K), ek_b, dk_b, C)$ returns $M^*(pars)$ rather than $\bot$.

WROB FAILURE. We show that encryption with redundancy fails to provide even *weak* robustness for all *unkeyed* redundancy codes. This is still a powerful negative result because many forms of redundancy that might intuitively work, such as the first four of Figure 6, are included. More precisely, we claim that for any unkeyed redundancy code $\mathcal{RED}$ and both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$, there is an AI-ATK encryption scheme $\mathcal{GE}$ such that the scheme $\overline{\mathcal{GE}}$ resulting from the encryption-with-redundancy transform applied to $\mathcal{GE}$ and $\mathcal{RED}$ is not WROB-CPA. We build $\mathcal{GE}$ by modifying a given AI-ATK + WROB-CPA encryption scheme $\mathcal{GE}^* = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}^*, \mathsf{Dec}^*)$. With notation as above, the new algorithms for the scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are shown on the top right side of Figure 7.

Our first claim is that the assumption that $\mathcal{GE}^*$ is AI-ATK implies that $\mathcal{GE}$ is too. Our second claim, that $\overline{\mathcal{GE}}$ is not WROB-CPA, is demonstrated by the following attack. For a pair id_0, id_1 of distinct identities of its choice, the adversary A, on input $(pars, \varepsilon)$, makes queries $ek_0 \xleftarrow{\$} \mathbf{GetEK}(id_0)$ and $ek_1 \xleftarrow{\$} \mathbf{GetEK}(id_1)$ and returns (id_0, id_1, M), where M can be any message in the message space associated to pars. We claim that $\mathbf{Adv}^{\mathrm{wrob}}_{\overline{\mathcal{GE}}}(A)$ is high. Letting dk_1 denote the decryption key corresponding to ek_1, the reason is the following. Let

$r_0 \stackrel{\$}{\leftarrow} \mathsf{RC}(\varepsilon, ek_0\|M)$ and $C \stackrel{\$}{\leftarrow} \mathsf{Enc}(pars, ek_0, M\|r_0)$. The assumed WROB-CPA security of $\mathcal{GE}^*$ implies that $\mathsf{Dec}(pars, ek_1, dk_1, C)$ is most probably $M^*(pars)\|r_1(pars)$ where $r_1(pars) = \mathsf{RC}(\varepsilon, ek_1\|M^*(pars); 0^l)$. But the correctness of $\mathcal{RED}$ implies that $\mathsf{RV}(\varepsilon, ek_1\|M^*(pars), r_1(pars)) = 1$ and hence $\overline{\mathsf{Dec}}((pars, \varepsilon), ek_1, dk_1, C)$ returns $M^*(pars)$ rather than $\perp$.

4 Transforms that work

We present a transform that confers weak robustness and another that confers strong robustness. They preserve privacy and anonymity, work for PKE as well as IBE, and for CPA as well as CCA. In both cases the security proofs surface some delicate issues. Besides being useful in its own right, the weak robustness transform is a crucial step in obtaining strong robustness, so we begin there.

Weak robustness transform. We saw that encryption-with-redundancy fails to provide even weak robustness if the redundancy code is unkeyed. Here we show that if the redundancy code is keyed, even in the simplest possible way where the redundancy is just the key itself, the transform does provide weak robustness, turning any AI-ATK secure general encryption scheme into an AI-ATK + WROB-ATK one, for both ATK $\in$ {CPA, CCA}.

The transformed scheme encrypts with the message a key K placed in the public parameters. In more detail, the *weak robustness transform* associates to a given general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and integer parameter k, representing the length of K, the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are depicted in Figure 9. Note that if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$ and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, so that our results, captured by Theorem 4.1 below, cover both settings.

The intuition for the weak robustness of $\overline{\mathcal{GE}}$ is that the $\mathcal{GE}$ decryption under one key, of an encryption of $\overline{M}\|K$ created under another key, cannot, by the assumed AI-ATK security of $\mathcal{GE}$, reveal K, and hence the check will fail. This is pretty much right for PKE, but the delicate issue is that for IBE, information about K can enter via the identities, which in this case are the encryption keys and could be chosen by the adversary as a function of K. Indeed, the counterexample from Section 3 can be extended to work for any keyed redundancy code if the key can be encoded into the identity space. Namely, the adversary can encode the key K into the identity $id_1 = ek_1$ while the counterexample decryption algorithm could decode K from its input ek and output $M \leftarrow M^*(pars)\|\mathsf{RC}(K, ek\|M^*(pars); 0^l)$ as a default message. We show however that this can be dealt with by making K sufficiently longer than the identities.

Theorem 4.1 *Let $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme with identity space $\{0,1\}^n$, and let $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be the general encryption scheme resulting from applying the weak robustness transform to $\mathcal{GE}$ and integer parameter k. Then*

1. <u>AI-ATK</u>*: Let A be an ai-adversary against $\overline{\mathcal{GE}}$. Then there is an ai-adversary B against $\mathcal{GE}$ such that*

$$\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) \;=\; \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B)\;.$$

Adversary B inherits the query profile of A and has the same running time as A. If A is a cpa adversary then so is B.

2. <u>WROB-ATK</u>*: Let A be a wrob adversary against $\overline{\mathcal{GE}}$ with running time t, and let $\ell = 2n + \lceil \log_2(t) \rceil$. Then there is an ai-adversary B against $\mathcal{GE}$ such that*

$$\mathbf{Adv}^{\mathrm{wrob}}_{\overline{\mathcal{GE}}}(A) \;\le\; \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) + 2^{\ell-k}\;.$$

Adversary B inherits the query profile of A and has the same running time as A. If A is a cpa adversary then so is B. ∎

The first part of the theorem implies that if $\mathcal{GE}$ is AI-ATK then $\overline{\mathcal{GE}}$ is AI-ATK as well. The second part of the theorem implies that if $\mathcal{GE}$ is AI-ATK and k is chosen sufficiently larger than $2n + \lceil \log_2(t) \rceil$ then $\overline{\mathcal{GE}}$ is WROB-ATK. In both cases this is for both ATK $\in \{\text{CPA}, \text{CCA}\}$. The theorem says it directly for CCA, and for CPA by the fact that if A is a cpa adversary then so is B. When we say that B inherits the query profile of A we mean that for every oracle that B has, if A has an oracle of the same name and makes q queries to it, then this is also the number B makes.

Proof of Theorem 4.1. The proof of Part 1 of Theorem 4.1 is straightforward and is omitted. The proof of Part 2 of Theorem 4.1 relies on the following information-theoretic lemma.

Lemma 4.2 *Let $\ell \leq k$ be positive integers and let A_1, A_2 be arbitrary algorithms with the length of the output of A_1 always being ℓ. Let P denote the probability that $A_2(A_1(K)) = K$ where the probability is over K drawn at random from $\{0,1\}^k$ and the coins of A_1, A_2. Then $P \leq 2^{\ell-k}$.*

Proof of Lemma 4.2: We may assume A_1, A_2 are deterministic for, if not, we can hardwire a "best" choice of coins for each. For each ℓ-bit string L let $S_L = \{K \in \{0,1\}^k : A_1(K) = L\}$ and let $s(L) = |S_L|$. Let $\mathcal{L}$ be the set of all $L \in \{0,1\}^\ell$ such that $s(L) > 0$. Then

$$\begin{aligned} P &= \sum_{L \in \mathcal{L}} \Pr[\, A_2(L) = K \mid A_1(K) = L \,] \cdot \Pr[\, A_1(K) = L \,] \\ &= \sum_{L \in \mathcal{L}} \frac{1}{s(L)} \cdot \frac{s(L)}{2^k} \\ &= \sum_{L \in \mathcal{L}} \frac{1}{2^k} \end{aligned}$$

which is at most $2^{\ell-k}$ as claimed. ∎

Proof of Part 2 of Theorem 4.1: Games $\mathrm{G}_0, \mathrm{G}_1$ of Figure 8 differ only in their **Finalize** procedures, with the message encrypted at line 04 to create ciphertext C in G_1 being a constant rather than $\overline{M}_0$ in G_0. We have

$$\mathbf{Adv}^{\mathrm{wrob}}_{\overline{\mathcal{GE}}}(A) = \Pr\left[\, \mathrm{G}_0^A \,\right] = \left(\Pr\left[\, \mathrm{G}_0^A \,\right] - \Pr\left[\, \mathrm{G}_1^A \,\right]\right) + \Pr\left[\, \mathrm{G}_1^A \,\right] .$$

we design B so that

$$\Pr\left[\, \mathrm{G}_0^A \,\right] - \Pr\left[\, \mathrm{G}_1^A \,\right] \leq \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) .$$

On input *pars*, adversary B executes lines 02,03 of **Initialize** and runs A on input $(pars, K)$. It replies to **GetEK**, **GetDK** and **Dec** queries of A via its own oracles of the same name. When A halts with output M, id_0, id_1, adversary B queries its **LR** oracle with $id_0, id_0, 0^{|M|} \| 0^k, M \| K$ to get back a ciphertext C. It then makes query **GetDK**(id_1) to get back $\mathsf{DK}[id_1]$. Note this is a legal query for B because id_1 is not one of the challenge identities in its **LR** query, but it would not have been legal for A. Now B executes lines 01–09 of the code of **Finalize** of G_1, except that it sets the value C on line 04 to be its own challenge ciphertext. If $\overline{M}_1 \neq \perp$ it outputs 1, else 0.

To complete the proof we show that $\Pr[\mathrm{G}_1^A] \leq 2^{\ell-k}$. We observe that M as computed at line 05 of **Finalize** in G_1 depends only on $pars, \mathsf{EK}[id_1], \mathsf{EK}[id_0], \mathsf{DK}[id_1], |\overline{M}_0|, k$. We would have liked

proc Initialize // $\mathrm{G}_0, \mathrm{G}_1$

$(pars, msk) \xleftarrow{\$} \mathsf{PG}$
$K \xleftarrow{\$} \{0,1\}^k$
$U, V \leftarrow \emptyset$
Return $(pars, K)$

proc GetEK(id) // $\mathrm{G}_0, \mathrm{G}_1$

$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

proc GetDK(id) // $\mathrm{G}_0, \mathrm{G}_1$

If $id \notin U$ then return $\perp$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

proc Dec(C, id) // $\mathrm{G}_0, \mathrm{G}_1$

If $id \notin U$ then return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| K^* \leftarrow M$
If $(K = K^*)$ then return $\overline{M}$
Else Return $\perp$

proc Finalize$(\overline{M}, id_0, id_1)$ // G_0

If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$\overline{M}_0 \leftarrow \overline{M}$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], \overline{M}_0 \| K)$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
If $M = \perp$ then $\overline{M}_1 \leftarrow \perp$
Else
    $\overline{M}_1 \| K^* \leftarrow M$
    If $(K \neq K^*)$ then $\overline{M}_1 \leftarrow \perp$
Return $(\overline{M}_0 \neq \perp) \wedge (\overline{M}_1 \neq \perp)$

proc Finalize$(\overline{M}, id_0, id_1)$ // G_1

If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$\overline{M}_0 \leftarrow \overline{M}$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], 0^{|\overline{M}_0|} \| 0^k)$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
If $M = \perp$ then $\overline{M}_1 \leftarrow \perp$
Else
    $\overline{M}_1 \| K^* \leftarrow M$
    If $(K \neq K^*)$ then $\overline{M}_1 \leftarrow \perp$
Return $(\overline{M}_0 \neq \perp) \wedge (\overline{M}_1 \neq \perp)$

Figure 8: Games for the proof of Part 2 of Theorem 4.1.

to say that none of these depend on K. This would mean that the probability that $M \neq \perp$ and parses as $\overline{M}_1 \| K$ is at most 2^{-k}, making $\Pr[\mathrm{G}_1^A] \leq 2^{-k}$. In the PKE case, what we desire is almost true because the only item in our list that can depend on K is $|\overline{M}_0|$, which can carry at most $\log_2(t)$ bits of information about K. But id_0, id_1 could depend on K so in general, and in the IBE case in particular, $\mathsf{EK}[id_0], \mathsf{EK}[id_1], \mathsf{DK}[id_1]$ could depend on K. However we assumed that identities are n bits, so the total amount of information about K in the list $pars, \mathsf{EK}[id_1], \mathsf{EK}[id_0], \mathsf{DK}[id_1], |M_0|, k$ is at most $2n + \log_2(t)$ bits. We conclude by applying Lemma 4.2 with $\ell = 2n + \lceil \log_2(t) \rceil$. ∎

Arbitrary identities. Theorem 4.1 converts a scheme $\mathcal{GE}$ with identity space $\{0,1\}^n$ into a scheme $\overline{\mathcal{GE}}$ with the same identity space $\{0,1\}^n$. The condition that $\mathcal{GE}$ has identity space $\{0,1\}^n$ is not really a restriction, because any scheme with identity space $\{0,1\}^*$ can be easily converted by restricting the identities to n-bit strings. At the same time, by hashing the identities with a collision-resistant hash function, $\overline{\mathcal{GE}}$ can be made to handle arbitrary identities in $\{0,1\}^*$. It is well known that collision-resistant hashing of identities preserves AI-ATK [BB04] and it's also easy to see that it preserves WROB-ATK. Here, it is important that the transformed scheme calls the underlying encryption and decryption algorithms Enc and Dec of $\mathcal{GE}$ with the hashed identities, not the full identities. In practice we might hash with SHA256 so that $n = 256$, and, assuming $t \leq 2^{128}$, setting $k = 768$ would make $2^{\ell - k} = 2^{-128}$.

Commitment schemes. Our strong robustness transform will use commitments. A commitment scheme is a 3-tuple $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$. The parameter generation algorithm CPG returns public parameters $cpars$. The committal algorithm Com takes $cpars$ and data x as input and returns a commitment com to x along with a decommittal key dec. The deterministic verification algorithm Ver takes $cpars, x, com, dec$ as input and returns 1 to indicate that accepts or

Algorithm $\overline{\mathsf{PG}}$
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$
$K \xleftarrow{\$} \{0,1\}^k$
Return $((pars, K), msk)$

Algorithm $\overline{\mathsf{Enc}}((pars, K), ek, \overline{M})$
$C \xleftarrow{\$} \mathsf{Enc}(pars, ek, \overline{M} \| K))$
Return C

Algorithm $\overline{\mathsf{KG}}((pars, K), msk, id)$
$(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return (ek, dk)

Algorithm $\overline{\mathsf{Dec}}((pars, K), ek, dk, C)$
$M \leftarrow \mathsf{Dec}(pars, ek, dk, C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| K^* \leftarrow M$
If $(K = K^*)$ then return $\overline{M}$
Else Return $\perp$

Figure 9: General encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ resulting from applying our weak-robustness transform to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and integer parameter k.

proc Initialize
$cpars \xleftarrow{\$} \mathsf{CPG}$; $b \xleftarrow{\$} \{0,1\}$; Return $cpars$

proc LR(x_0, x_1)
$(com, dec) \xleftarrow{\$} \mathsf{Com}(cpars, x_b)$; Return com

proc Finalize(b')
Return $(b' = b)$

proc Initialize
$cpars \xleftarrow{\$} \mathsf{CPG}$; Return $cpars$

proc Finalize$(com, x_0, dec_0, x_1, dec_1)$
$d_0 \leftarrow \mathsf{Ver}(cpars, x_0, com, dec_0)$
$d_1 \leftarrow \mathsf{Ver}(cpars, x_1, com, dec_1)$
Return $(x_0 \neq x_1 \wedge d_0 = 1 \wedge d_1 = 1)$

Figure 10: Game $\mathrm{HIDE}_{\mathcal{CMT}}$ (left) captures the hiding property while Game $\mathrm{BIND}_{\mathcal{CMT}}$ (right) captures the binding property. The adversary may call **LR** only once.

0 to indicate that it rejects. Correctness requires that, for any $x \in \{0,1\}^*$, any $cpars \in [\mathsf{CPG}]$, and any $(com, dec) \in [\mathsf{Com}(cpars, x)]$, we have that $\mathsf{Ver}(cpars, x, com, dec) = 1$ with probability one, where the probability is taken over the coins of Com. We require the scheme to have the *uniqueness* property, which means that for any $x \in \{0,1\}^*$, any $cpars \in [\mathsf{CPG}]$, and any $(com, dec) \in [\mathsf{Com}(cpars, x)]$ it is the case that $\mathsf{Ver}(cpars, x, com^*, dec) = 0$ for all $com^* \neq com$. In most schemes the decommittal key is the randomness used by the committal algorithm and verification is by re-applying the committal function, which ensures uniqueness. The advantage measures

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) = 2 \cdot \Pr\left[\mathrm{HIDE}^A_{\mathcal{CMT}} \Rightarrow \mathsf{true}\right] - 1 \quad \text{and} \quad \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(A) = \Pr\left[\mathrm{BIND}^A_{\mathcal{CMT}} \Rightarrow \mathsf{true}\right],$$

which refer to the games of Figure 10, capture, respectively, the standard hiding and binding properties of a commitment scheme. We refer to the corresponding notions as HIDE and BIND. We refer to the corresponding notions as HIDE and BIND.

The strong robustness transform. The idea is for the ciphertext to include a commitment to the encryption key. The commitment is *not* encrypted, but the decommittal key is. In detail, given a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$ the *strong robustness transform* associates to them the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are depicted in Figure 11. Note that if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$ and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, so that our results, captured by the Theorem 4.3, cover both settings.

In this case the delicate issue is not the robustness but the AI-ATK security of $\overline{\mathcal{GE}}$ in

Algorithm $\overline{\mathsf{PG}}$ $(pars, msk) \xleftarrow{\$} \mathsf{PG}$ $cpars \xleftarrow{\$} \mathsf{CPG}$ Return $((pars, cpars), msk)$ **Algorithm** $\overline{\mathsf{Enc}}((pars, cpars), ek, \overline{M})$ $(com, dec) \xleftarrow{\$} \mathsf{Com}(cpars, ek)$ $C \xleftarrow{\$} \mathsf{Enc}(pars, ek, \overline{M} \| dec))$ Return (C, com)	**Algorithm** $\overline{\mathsf{KG}}((pars, cpars), msk, id)$ $(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$ Return (ek, dk) **Algorithm** $\overline{\mathsf{Dec}}((pars, cpars), ek, dk, (C, com))$ $M \leftarrow \mathsf{Dec}(pars, ek, dk, C)$ If $M = \perp$ then return $\perp$ $\overline{M} \| dec \leftarrow M$ If $(\mathsf{Ver}(cpars, ek, com, dec) = 1)$ then return $\overline{M}$ Else Return $\perp$

Figure 11: General encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ resulting from applying our strong robustness transform to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$.

the CCA case. Intuitively, the hiding security of the commitment scheme means that a $\overline{\mathcal{GE}}$ ciphertext does not reveal the encryption key. As a result, we would expect AI-ATK security of $\overline{\mathcal{GE}}$ to follow from the commitment hiding security and the assumed AI-ATK security of $\mathcal{GE}$. This turns out not to be true, and demonstrably so, meaning there is a counterexample to this claim. (See below.) What we show is that the claim is true if $\mathcal{GE}$ is additionally WROB-ATK. This property, if not already present, can be conferred by first applying our weak robustness transform.

Theorem 4.3 *Let $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme, and let $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be the general encryption scheme resulting from applying the strong robustness transform to $\mathcal{GE}$ and commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$. Then*

1. <u>AI-ATK</u>*: Let A be an ai-adversary against $\overline{\mathcal{GE}}$. Then there is a wrob adversary W against $\mathcal{GE}$, a hiding adversary H against $\mathcal{CMT}$ and an ai-adversary B against $\mathcal{GE}$ such that*

$$\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) \;\leq\; 2 \cdot \mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(W) + 2 \cdot \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(H) + 3 \cdot \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) \; .$$

Adversaries W, B inherit the query profile of A, and adversaries W, H, B have the same running time as A. If A is a cpa adversary then so are W, B.

2. <u>SROB-ATK</u>*: Let A be a srob adversary against $\overline{\mathcal{GE}}$ making q **GetEK** queries. Then there is a binding adversary B against $\mathcal{CMT}$ such that*

$$\mathbf{Adv}^{\mathrm{srob}}_{\overline{\mathcal{GE}}}(A) \;\leq\; \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(B) + \binom{q}{2} \cdot \mathbf{Coll}_{\mathcal{GE}} \; .$$

Adversary B has the same running time as A. ▌

The first part of the theorem implies that if $\mathcal{GE}$ is AI-ATK and WROB-ATK and $\mathcal{CMT}$ is HIDE then $\overline{\mathcal{GE}}$ is AI-ATK, and the second part of the theorem implies that if $\mathcal{CMT}$ is BIND secure and $\mathcal{GE}$ has low encryption key collision probability then $\overline{\mathcal{GE}}$ is SROB-ATK. In both cases this is for both ATK $\in$ {CPA, CCA}. We remark that the proof shows that in the CPA case the WROB-ATK assumption on $\mathcal{GE}$ in the first part is actually not needed. The encryption key collision probability $\mathbf{Coll}_{\mathcal{GE}}$ of $\mathcal{GE}$ is defined as the maximum probability that $ek_0 = ek_1$ in the experiment

$$(pars, msk) \xleftarrow{\$} \mathsf{PG}\; ; \; (ek_0, dk_0) \xleftarrow{\$} \mathsf{KG}(pars, msk, id_0)\; ; \; (ek_1, dk_1) \xleftarrow{\$} \mathsf{KG}(pars, msk, id_1) \; ,$$

where the maximum is over all distinct identities id_0, id_1. It is easy to see that $\mathcal{GE}$ being AI implies $\mathbf{Coll}_{\mathcal{GE}}$ is negligible, so asking for low encryption key collision probability is in fact not an extra assumption. (For a general encryption scheme the adversary needs to have hardwired the identities that achieve the maximum, but this is not necessary for PKE because here the probability being maximized is the same for all pairs of distinct identities.) The reason we made the encryption key collision probability explicit is that for most schemes it is unconditionally low. For example, when $\mathcal{GE}$ is the ElGamal PKE scheme, it is $1/|\mathbb{G}|$ where $\mathbb{G}$ is the group being used.

Proof of Part 1 of Theorem 4.3: Game G_0 of Figure 12 is game $\mathrm{AI}_{\overline{\mathcal{GE}}}$ tailored to the case that A makes only one **LR** query, an assumption we explained we can make. If we wish to exploit the assumed AI-ATK security of $\mathcal{GE}$, we need to be able to answer **Dec** queries of A using the **Dec** oracle in game $\mathrm{AI}_{\mathcal{GE}}$. Thus we would like to substitute the $\mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$ call in a $\mathbf{Dec}((C, com), id)$ query of G_0 with a $\mathbf{Dec}(C, id)$ call of an adversary B in $\mathrm{AI}_{\mathcal{GE}}$. The difficulty is that C might equal C^* but $com \neq com^*$, so that the call is not legal for B. To get around this, the first part of our proof will show that the decryption procedure of G_0 can be replaced by the alternative one of G_4, where this difficulty vanishes. This part exploits the uniqueness of the commitment scheme and the weak robustness of $\mathcal{GE}$. After that we will exploit the AI-ATK security of $\mathcal{GE}$ to remove dependence on dec^* in **LR**, allowing us to exploit the HIDE security of $\mathcal{CMT}$ to make the challenge commitment independent of $\mathsf{EK}[id_b^*]$. This allows us to conclude by again using the AI-ATK security of $\mathcal{GE}$. We proceed to the details.

In game G_0, if A makes a $\mathbf{Dec}((C^*, com), id_b^*)$ query with $com \neq com^*$ then the uniqueness of $\mathcal{CMT}$ implies that the procedure in question will return $\perp$. This means that line 02 of **Dec** in G_0 can be rewritten as line 02 of **Dec** in G_1 and the two procedures are equivalent. Procedure **Dec** of G_2 includes the boxed code and hence is equivalent to procedure **Dec** of G_1. Hence

$$\begin{aligned}\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) = \Pr\left[\,G_0^A\,\right] = \Pr\left[\,G_1^A\,\right] &= \Pr\left[\,G_2^A\,\right] \\ &= \Pr\left[\,G_3^A\,\right] + \Pr\left[\,G_2^A\,\right] - \Pr\left[\,G_3^A\,\right] \\ &\leq \Pr\left[\,G_3^A\,\right] + \Pr\left[\,G_3^A \text{ sets } \mathsf{bad}\,\right] .\end{aligned}$$

The inequality above is by Lemma 2.1 which applies because G_2, G_3 are identical until bad. We design W so that

$$\Pr\left[\,G_3^A \text{ sets } \mathsf{bad}\,\right] \leq \mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(W) .$$

On input $pars$, adversary W executes lines 02,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to **GetEK**, **GetDK**, **Dec** queries of A via its own oracles of the same name, as per the code of G_3. When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary W executes lines 01,02,03 of the code of **LR** of G_3. It then outputs $\overline{M}_b^* \| dec^*, id_b^*, id_{1-b}^*$ and halts.

Next we bound $\Pr[G_3^A]$. Procedure **Dec** of G_4 results from simplifying the code of procedure **Dec** of G_3, so

$$\Pr\left[\,G_3^A\,\right] = \Pr\left[\,G_4^A\,\right] = \left(\Pr\left[\,G_4^A\,\right] - \Pr\left[\,G_5^A\,\right]\right) + \Pr\left[\,G_5^A\,\right] .$$

The step from G_4 to G_5 modifies only **LR**, replacing dec^* with a constant. We are assuming here that any decommitment key output by Com, regardless of the inputs to the latter, has length d bits. We design B_1 so that

$$\Pr\left[\,G_4^A\,\right] - \Pr\left[\,G_5^A\,\right] = \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B_1) .$$

On input $pars$, adversary B_1 executes lines 02,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to **GetEK**, **GetDK**, **Dec** queries of A via its own oracles of the same

$\underline{\textbf{proc Initialize}}$ // G_0–G_6

$(pars, msk) \xleftarrow{\$} \mathsf{PG}$
$cpars \xleftarrow{\$} \mathsf{CPG}$
$b \xleftarrow{\$} \{0,1\}$
$S, U, V \leftarrow \emptyset$; $C^* \leftarrow \perp$; $com^* \leftarrow \perp$
$id_0^* \leftarrow \perp$; $id_1^* \leftarrow \perp$
Return $(pars, cpars)$

$\underline{\textbf{proc GetEK}(id)}$ // G_0–G_6

$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

$\underline{\textbf{proc GetDK}(id)}$ // G_0–G_6

If $id \notin U$ then return $\perp$
If $id \in \{id_0^*, id_1^*\}$ then return $\perp$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

$\underline{\textbf{proc Finalize}(b')}$ // G_0–G_6

Return $(b' = b)$

$\underline{\textbf{proc LR}(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)}$ // G_0–G_4

If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\perp$
If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\perp$
$(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, \mathsf{EK}[id_b^*])$
$C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| dec^*)$
Return (C^*, com^*)

$\underline{\textbf{proc LR}(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)}$ // G_5

If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\perp$
If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\perp$
$(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, \mathsf{EK}[id_b^*])$
$C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| 0^d)$
Return (C^*, com^*)

$\underline{\textbf{proc LR}(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)}$ // G_6

If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\perp$
If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\perp$
$(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, 0^e)$
$C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| 0^d)$
Return (C^*, com^*)

$\underline{\textbf{proc Dec}((C, com), id)}$ // G_0

If $id \notin U$ then return $\perp$
If $(id = id_b^*) \wedge (C, com) = (C^*, com^*)$ then return $\perp$
If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
    Return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| dec \leftarrow M$
If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
Else return $\perp$

$\underline{\textbf{proc Dec}((C, com), id)}$ // G_1

If $id \notin U$ then return $\perp$
If $(id = id_b^*) \wedge (C = C^*)$ then return $\perp$
If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
    Return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| dec \leftarrow M$
If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
Else return $\perp$

$\underline{\textbf{proc Dec}((C, com), id)}$ // $\boxed{\mathrm{G}_2}$, G_3

If $id \notin U$ then return $\perp$
If $(id = id_b^*) \wedge (C = C^*)$ then return $\perp$
If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
    Return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
If $(id = id_{1-b}^* \neq id_b^*) \wedge (C = C^*) \wedge (com \neq com^*)$ then
    $M^* \leftarrow M$
    If $M \neq \perp$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $M \leftarrow \perp$; $\boxed{M \leftarrow M^*}$
If $M = \perp$ then return $\perp$
$\overline{M} \| dec \leftarrow M$
If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
Else return $\perp$

$\underline{\textbf{proc Dec}((C, com), id)}$ // G_4–G_6

If $id \notin U$ then return $\perp$
If $(id = id_0^*) \wedge (C = C^*)$ then return $\perp$
If $(id = id_1^*) \wedge (C = C^*)$ then return $\perp$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| dec \leftarrow M$
If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
Else return $\perp$

Figure 12: Games for the proof of Part 1 of Theorem 4.3.

name, as per the code of G_4. Here we make crucial use of the fact that the alternative decryption rule of **Dec** of G_4 allows B_1 to respond to **Dec** queries of A without the need to query its own **Dec** oracle on (C^*, id_0^*) or (C^*, id_1^*). When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary B_1 executes lines 01,02,03 of the code of **LR** of G_4. It then queries $id_b^*, id_b^*, \overline{M}_b^* \| 0^d, \overline{M}_b^* \| dec^*$ to its own **LR** oracle to get back a ciphertext C^*, and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary B_1 outputs 1 if $b = b'$ and 0 otherwise.

Next we bound $\Pr[G_5^A]$. Procedure **LR** of G_6 uses a constant 0^e rather than $\mathsf{EK}[id_b^*]$ as data for Com at line 03. The value of e is arbitrary, and we can just let $e = 1$. Then

$$\Pr\left[\, G_5^A \,\right] = \left(\Pr\left[\, G_5^A \,\right] - \Pr\left[\, G_6^A \,\right]\right) + \Pr\left[\, G_6^A \,\right] .$$

We design H so that

$$\Pr\left[\, G_5^A \,\right] - \Pr\left[\, G_6^A \,\right] \leq \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(H) .$$

On input $cpars$, adversary H executes lines 01,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to **GetEK**, **GetDK**, **Dec** queries of A by direct execution of the code of these procedures in G_5, possible since it knows msk. When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary H executes lines 01,02 of the code of **LR** of G_5. It then queries $0^e, \mathsf{EK}[id_b^*]$ to its own **LR** oracle to get back a commitment com^*. It executes line 04 of **LR** of G_5 and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary H returns 1 if $b = b'$ and 0 otherwise.

Finally we design B_2 so that

$$2 \cdot \Pr\left[\, G_6^A \,\right] - 1 \leq \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B_2) .$$

On input $pars$, adversary B_2 executes lines 02,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to **GetEK**, **GetDK**, **Dec** queries of A via its own oracles of the same name, as per the code of G_6. Again we make crucial use of the fact that the alternative decryption rule of **Dec** of G_6 allows B_2 to respond to **Dec** queries of A without the need to query its own **Dec** oracle on (C^*, id_0^*) or (C^*, id_1^*). When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary B_2 executes lines 01,02,03 of the code of **LR** of G_6. It then queries $id_0^*, id_1^*, \overline{M}_0^* \| 0^d, \overline{M}_1^* \| dec^*$ to its own **LR** oracle to get back a ciphertext C^*, and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary B_2 outputs b'.

Adversary B of the theorem statement runs B_1 with probability 2/3 and B_2 with probability 1/3. ▮

Proof of Part 2 of Theorem 4.3: In the execution of A with game $\mathrm{SROB}_{\overline{\mathcal{GE}}}$ let COLL be the event that there exist distinct id_0, id_1 queried by A to its **GetEK** oracle such that the encryption keys returned in response are the same. Then

$$\begin{aligned} \mathbf{Adv}^{\mathrm{srob}}_{\overline{\mathcal{GE}}}(A) &= \Pr\left[\mathrm{SROB}^A_{\overline{\mathcal{GE}}} \wedge \mathrm{COLL}\right] + \Pr\left[\mathrm{SROB}^A_{\overline{\mathcal{GE}}} \wedge \overline{\mathrm{COLL}}\right] \\ &\leq \Pr\left[\,\mathrm{COLL}\,\right] + \Pr\left[\mathrm{SROB}^A_{\overline{\mathcal{GE}}} \wedge \overline{\mathrm{COLL}}\right] . \end{aligned}$$

But

$$\Pr\left[\,\mathrm{COLL}\,\right] \leq \binom{q}{2} \cdot \mathbf{Coll}_{\mathcal{GE}}$$

and we can design B such that

$$\Pr\left[\mathrm{SROB}^{A}_{\overline{\mathcal{GE}}} \wedge \overline{\mathrm{COLL}}\right] \le \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(B)\ .$$

We omit the details. ▌

THE NEED FOR WEAK-ROBUSTNESS. As we said above, the AI-ATK security of $\overline{\mathcal{GE}}$ won't be implied merely by that of $\mathcal{GE}$. (We had to additionally assume that $\mathcal{GE}$ is WROB-ATK.) Here we justify this somewhat counter-intuitive claim. This discussion is informal but can be turned into a formal counterexample. Imagine that the decryption algorithm of $\mathcal{GE}$ returns a fixed string of the form $(\hat{M}, \hat{dec})$ whenever the wrong key is used to decrypt. Moreover, imagine $\mathcal{CMT}$ is such that it is easy, given $cpars, x, dec$, to find com so that $\mathsf{Ver}(cpars, x, com, dec) = 1$. (This is true for any commitment scheme where dec is the coins used by the Com algorithm.) Consider then the AI-ATK adversary A against the transformed scheme that that receives a challenge ciphertext (C^*, com^*) where $C^* \leftarrow \mathsf{Enc}(pars, \mathsf{EK}[id_b], M^*\|dec^*)$ for hidden bit $b \in \{0,1\}$. It then creates a commitment $\hat{com}$ of $\mathsf{EK}[id_1]$ with opening information $\hat{dec}$, and queries $(C^*, \hat{com})$ to be decrypted under $\mathsf{DK}[id_0]$. If $b = 0$ this query will probably return $\perp$ because $\mathsf{Ver}(cpars, \mathsf{EK}[id_0], \hat{com}, dec^*)$ is unlikely to be 1, but if $b = 1$ it returns $\hat{M}$, allowing A to determine the value of b. The weak robustness of $\mathcal{GE}$ rules out such anomalies.

5 A SROB-CCA version of Cramer-Shoup

Let $\mathbb{G}$ be a group of prime order p, and $H\colon\ \mathsf{Keys}(H) \times \mathbb{G}^3 \to \mathbb{G}$ a family of functions. We assume $\mathbb{G}, p, H$ are fixed and known to all parties. Figure 13 shows the Cramer-Shoup (CS) scheme and the variant $\mathcal{CS}^*$ scheme where $\mathbf{1}$ denotes the identity element of $\mathbb{G}$. The differences are boxed. Recall that the CS scheme was shown to be IND-CCA in [CS03] and ANO-CCA in [BBDP01]. However, for any message $M \in \mathbb{G}$ the ciphertext $(\mathbf{1}, \mathbf{1}, M, \mathbf{1})$ in the CS scheme decrypts to M under *any* $pars, pk$, and sk, meaning in particular that the scheme is not even SROB-CPA. The modified scheme $\mathcal{CS}^*$ —which continues to be IND-CCA and ANO-CCA— removes this pathological case by having Enc choose the randomness u to be non-zero —Enc draws u from $\mathbb{Z}_p^*$ while the CS scheme draws it from $\mathbb{Z}_p$— and then having Dec reject (a_1, a_2, c, d) if $a_1 = \mathbf{1}$. This thwarts the attack, but is there any other attack? We show that there is not by proving that $\mathcal{CS}^*$ is actually SROB-CCA. Our proof of robustness relies only on the security —specifically, pre-image resistance— of the hash family H: it does not make the DDH assumption. Our proof uses ideas from the information-theoretic part of the proof of [CS03].

We say that a family $H\colon\ \mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Rng}(H)$ of functions is *pre-image resistant* if, given a key K and a *random* range element v^*, it is computationally infeasible to find a pre-image of v^* under $H(K, \cdot)$. The notion is captured formally by the following advantage measure for an adversary I:

$$\mathbf{Adv}^{\text{pre-img}}_{H}(I) = \Pr\left[\, H(K,x) = v^* \ \ : \ \ K \stackrel{\$}{\leftarrow} \mathsf{Keys}(H)\ ;\ v^* \stackrel{\$}{\leftarrow} \mathsf{Rng}(H)\ ;\ x \stackrel{\$}{\leftarrow} I(K, v^*) \,\right]\ .$$

Pre-image resistance is not implied by the standard notion of one-wayness, since in the latter the target v^* is the image under $H(K, \cdot)$ of a random domain point, which may not be a random range point. However, it seems like a fairly mild assumption on a practical cryptographic hash function and is implied by the notion of "everywhere pre-image resistance" of [RS04], the difference being that, for the latter, the advantage is the maximum probability over all $v^* \in \mathsf{Rng}(H)$. We now claim the following.

Algorithm PG $K \xleftarrow{\$} \mathsf{Keys}(H)$; $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$ $g_2 \leftarrow g_1^w$; Return (g_1, g_2, K) Algorithm $\mathsf{Enc}((g_1, g_2, K), (e, f, h), M)$ $u \xleftarrow{\$} \mathbb{Z}_p^{\boxed{*}}$ $a_1 \leftarrow g_1^u$; $a_2 \leftarrow g_2^u$; $b \leftarrow h^u$ $c \leftarrow b \cdot M$; $v \leftarrow H(K, (a_1, a_2, c))$ $d \leftarrow e^u f^{uv}$; Return (a_1, a_2, c, d)	Algorithm $\mathsf{KG}(g_1, g_2, K)$ $x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$ $e \leftarrow g_1^{x_1} g_2^{x_2}$; $f \leftarrow g_1^{y_1} g_2^{y_2}$; $h \leftarrow g_1^{z_1} g_2^{z_2}$ Return $((e, f, h), (x_1, x_2, y_1, y_2, z_1, z_2))$ Algorithm $\mathsf{Dec}((g_1, g_2, K), (e, f, h), (x_1, x_2, y_1, y_2, z_1, z_2), C)$ $(a_1, a_2, c, d) \leftarrow C$; $v \leftarrow H(K, (a_1, a_2, c))$; $M \leftarrow c \cdot a_1^{-z_1} a_2^{-z_2}$ If $d \neq a_1^{x_1+y_1 v} a_2^{x_2+y_2 v}$ Then $M \leftarrow \perp$ $\boxed{\text{If } a_1 = \mathbf{1} \text{ Then } M \leftarrow \perp}$ Return M

Figure 13: The original CS scheme [CS03] does not contain the boxed code while the variant $\mathcal{CS}^*$ does. Although not shown above, the decryption algorithm in both versions always checks to ensure that the ciphertext $C \in \mathbb{G}^4$. The message space is $\mathbb{G}$.

Theorem 5.1 *Let B be an adversary making two **GetEK** queries, no **GetDK** queries and at most $q-1$ **Dec** queries, and having running time t. Then we can construct an adversary I such that*

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathcal{CS}^*}(A) \;\leq\; \mathbf{Adv}^{\mathrm{pre\text{-}img}}_{H}(I) + \frac{2q+1}{p} \,. \tag{1}$$

Furthermore, the running time of I is $t + q \cdot O(t_{\mathrm{exp}})$ where t_{exp} denotes the time for one exponentiation in $\mathbb{G}$.

Since $\mathcal{CS}^*$ is a PKE scheme, the above automatically implies security even in the presence of multiple **GetEK** and **GetDK** queries as required by game $\mathrm{SROB}_{\mathcal{CS}^*}$. Thus the theorem implies that $\mathcal{CS}^*$ is SROB-CCA if H is pre-image resistant. A detailed proof of Theorem 5.1 is below. We begin by sketching some intuition.

We begin by conveniently modifying the game interface. We replace B with an adversary A that gets input $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$ representing the parameters that would be input to B and the public keys returned in response to B's two **GetEK** queries. Let $(x_{01}, x_{02}, y_{01}, y_{02}, z_{01}, z_{02})$ and $(x_{11}, x_{12}, y_{11}, y_{12}, z_{11}, z_{12})$ be the corresponding secret keys. The decryption oracle takes (only) a ciphertext and returns its decryption under *both* secret keys, setting a WIN flag if these are both non-$\perp$. Adversary A no longer needs an output, since it can win via a **Dec** query.

Suppose A makes a **Dec** query (a_1, a_2, c, d). Then the code of the decryption algorithm Dec from Figure 13 tells us that, for this to be a winning query, it must be that

$$d \;=\; a_1^{x_{01}+y_{01}v} a_2^{x_{02}+y_{02}v} \;=\; a_1^{x_{11}+y_{11}v} a_2^{x_{12}+y_{12}v}$$

where $v = H(K, (a_1, a_2, c))$. Letting $u_1 = \log_{g_1}(a_1), u_2 = \log_{g_2}(a_2)$ and $s = \log_{g_1}(d)$, we have

$$s \;=\; u_1(x_{01} + y_{01}v) + wu_2(x_{02} + y_{02}v) \;=\; u_1(x_{11} + y_{11}v) + wu_2(x_{12} + y_{12}v) \tag{2}$$

However, even acknowledging that A knows little about $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) through its **Dec** queries, it is unclear why Equation (2) is prevented by pre-image resistance —or in fact any property short of being a random oracle— of the hash function H. In particular, there seems no way to "plant" a target v^* as the value v of Equation (2) since the adversary controls u_1 and u_2. However, suppose now that $a_2 = a_1^w$. (We will discuss later why we can assume this.) This implies $wu_2 = wu_1$ or $u_2 = u_1$ since $w \neq 0$. Now from Equation (2) we have

$$u_1(x_{01} + y_{01}v) + wu_1(x_{02} + y_{02}v) - u_1(x_{11} + y_{11}v) - wu_1(x_{12} + y_{12}v) \;=\; 0 \,.$$

We now see the value of enforcing $a_1 \neq 1$, since this implies $u_1 \neq 0$. After canceling u_1 and re-arranging terms, we have

$$v(y_{01} + wy_{02} - y_{11} - wy_{12}) + (x_{01} + wx_{02} - x_{11} - wx_{12}) \;=\; 0\;. \tag{3}$$

Given that $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) and w are chosen by the game, there is at most one solution v (modulo p) to Equation (3). We would like now to design I so that on input K, v^* it chooses $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) so that the solution v to Equation (3) is v^*. Then (a_1, a_2, c) will be a pre-image of v^* which I can output.

To make all this work, we need to resolve two problems. The first is why we may assume $a_2 = a_1^w$ —which is what enables Equation (3)— given that a_1, a_2 are chosen by A. The second is to properly design I and show that it can simulate A correctly with high probability. To solve these problems, we consider, as in [CS03], a modified check under which decryption, rather than rejecting when $d \neq a_1^{x_1+y_1v} a_2^{x_2+y_2v}$, rejects when $a_2 \neq a_1^w$ or $d \neq a_1^{x+yv}$, where $x = x_1 + wx_2$, $y = y_1 + wy_2$, $v = H(K, (a_1, a_2, c))$ and (a_1, a_2, c, d) is the ciphertext being decrypted. In our proof below, games G_0–G_2 move us towards this perspective. Then, we fork off two game chains. Games G_3–G_6 are used to show that the modified decryption rule increases the adversary's advantage by at most $2q/p$. Games G_7–G_{11} show how to embed a target value v^* into the components of the secret key without significantly affecting the ability to answer **Dec** queries. Based on the latter, we then construct I as shown below.

Proof of Theorem 5.1: The proof relies on Games G_0–G_{11} of Figures 14–16 and the adversary I of Figure 17.

We begin by transforming B into an adversary A such that

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathcal{CS}^*}(B) \;\leq\; \Pr\left[\,G_0^A\,\right]\;. \tag{4}$$

On input $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$, adversary A runs B on input (g_1, g_2, K). Adversary A returns to B the public key (e_0, f_0, h_0) in response to B's first **GetEK** query id_0, and (e_1, f_1, h_1) in response to its second **GetEK** query id_1. When B makes a **Dec** query, which can be assumed to have the form $(a_1, a_2, c, d), id_b$ for some $b \in \{0,1\}$, adversary A queries (a_1, a_2, c, d) to its own **Dec** oracle to get back (M_0, M_1) and returns M_b to B. When B halts, with output that can be assumed to have the form $((a_1, a_2, c, d), id_0, id_1)$, adversary A makes a final query (a_1, a_2, c, d) to its **Dec** oracle and also halts.

We assume that every **Dec** query (a_1, a_2, c, d) of A satisfies $a_1 \neq \mathbf{1}$. This is without loss of generality because the decryption algorithm rejects otherwise. This will be crucial below. Similarly, we assume $(a_1, a_2, c, d) \in \mathbb{G}^4$. We now proceed to the analysis.

Games G_1, G_2 start to move us to the alternative decryption rule. In G_1, if $a_2 = a_1^w$ and $d = a_1^{x_b+y_bv}$ then $d = a_1^{x_{b1}+y_{b1}v} a_2^{x_{b2}+y_{b2}v}$, so **Dec** in G_1 returns the correct decryption, like in G_0. If $a_2 \neq a_1^w$ or $d \neq a_1^{x_b+y_bv}$ then, if $d \neq a_1^{x_{b1}+y_{b1}v} \cdot a_2^{x_{b2}+y_{b2}v}$, then **Dec** in G_1 returns $\perp$, else it returns $ca_1^{-z_{b1}} a_2^{-z_{b2}}$, so again is correct either way. Thus,

$$\begin{aligned}
\Pr\left[\,G_0^A\,\right] &= \Pr\left[\,G_1^A\,\right] \\
&= \Pr\left[\,G_2^A\,\right] + (\Pr\left[\,G_1^A\,\right] - \Pr\left[\,G_2^A\,\right]) \\
&\leq \Pr\left[\,G_2^A\,\right] + \Pr\left[\,G_2^A \text{ sets } \mathsf{bad}\,\right]\;,
\end{aligned} \tag{5}$$

where the last line is by Lemma 2.1 since G_1, G_2 are identical until bad. We now fork off two game chains, one to bound each term above.

First, we will bound the second term in the right-hand side of Inequality (5). Our goal is to

proc Initialize Game G_0

000 $g_1 \xleftarrow{\$} \mathbb{G}^* \,;\, w \xleftarrow{\$} \mathbb{Z}_p^* \,;\, g_2 \leftarrow g_1^w$
001 $K \xleftarrow{\$} \mathsf{Keys}(H)$
002 For $b = 0, 1$ do
003 $\quad x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
004 $\quad e_b \leftarrow g_1^{x_{b1}} g_2^{x_{b2}}$
005 $\quad f_b \leftarrow g_1^{y_{b1}} g_2^{y_{b2}}$
006 $\quad h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
007 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Games G_1,G_2,G_3,G_4

100 $g_1 \xleftarrow{\$} \mathbb{G}^* \,;\, w \xleftarrow{\$} \mathbb{Z}_p^* \,;\, g_2 \leftarrow g_1^w$
101 $K \xleftarrow{\$} \mathsf{Keys}(H)$
102 For $b = 0, 1$ do
103 $\quad x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
104 $\quad x_b \leftarrow x_{b1} + w x_{b2} \,;\, y_b \leftarrow y_{b_1} + w y_{b2}$
105 $\quad e_b \leftarrow g_1^{x_b} \,;\, f_b \leftarrow g_1^{y_b} \,;\, h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
106 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Finalize Games G_0,G_1,G_2

020 Return $\textsc{Win}$

proc Finalize Game G_3

320 Return true

proc Finalize Game G_4

420 For $b = 0, 1$ do
421 $\quad$ For all $(a_1, a_2, c, d, v) \in S$ do
422 $\quad\quad$ If $d = a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then
423 $\quad\quad\quad \mathsf{bad} \leftarrow \mathsf{true}$
424 Return true

proc Dec$((a_1, a_2, c, d))$ Game G_0

010 $v \leftarrow H(K, (a_1, a_2, c))$
011 For $b = 0, 1$ do
012 $\quad M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
013 $\quad$ If $d \neq a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then $M_b \leftarrow \perp$
014 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $\textsc{Win} \leftarrow \mathsf{true}$
015 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Games $\boxed{\mathrm{G}_1}$,G_2

110 $v \leftarrow H(K, (a_1, a_2, c))$
111 For $b = 0, 1$ do
112 $\quad M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
113 $\quad$ If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b + y_b v})$ Then
114 $\quad\quad M_b \leftarrow \perp$
115 $\quad\quad$ If $d = a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then
116 $\quad\quad\quad \mathsf{bad} \leftarrow \mathsf{true} \,;\, \boxed{M_b \leftarrow c a_1^{-z_{b1}} a_2^{-z_{b2}}}$
117 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $\textsc{Win} \leftarrow \mathsf{true}$
118 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Game G_3

310 $v \leftarrow H(K, (a_1, a_2, c))$
311 For $b = 0, 1$ do
312 $\quad M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
313 $\quad$ If $(a_2 \neq a_1^w)$ Then
314 $\quad\quad M_b \leftarrow \perp$
315 $\quad\quad$ If $d = a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then $\mathsf{bad} \leftarrow \mathsf{true}$
316 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Game G_4

410 $v \leftarrow H(K, (a_1, a_2, c))$
411 For $b = 0, 1$ do $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
412 If $(a_2 \neq a_1^w)$ Then
413 $\quad S \leftarrow S \cup \{(a_1, a_2, c, d, v)\} \,;\, M_0, M_1 \leftarrow \perp$
414 Return (M_0, M_1)

Figure 14: Games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$, and G_4 for proof of Theorem 5.1. G_1 includes the boxed code at line 116 but G_2 does not.

move the choices of $x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2}$ $(b = 0, 1)$ and the setting of bad into **Finalize** while still being able to answer **Dec** queries. We will then be able to bound the probability that bad is set by a static analysis. Consider Game G_3. If $a_2 \neq a_1^w$ and $d = a_1^{x_{b1}+y_{b1}v} a_2^{x_{b2}+y_{b2}v}$ then bad is set in G_2. But $a_2 = a_1^w$ and $d \neq a_1^{x_b+y_bv}$ implies $d \neq a_1^{x_{b1}+y_{b1}v} a_2^{x_{b2}+y_{b2}v}$, so bad is not set in G_2. So,

$$\Pr\left[\,\mathrm{G}_2^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_3^A \text{ sets } \mathsf{bad}\,\right] . \tag{6}$$

Since we are only interested in the probability that G_3 sets bad, we have it always return true. The flag bad may be set at line 315, but is not used, so we move the setting of bad into the **Finalize** procedure in G_4. This requires that G_4 do some bookkeeping. We have also done some restructuring, moving some loop invariants out of the loop in **Dec**. We have

$$\Pr\left[\,\mathrm{G}_3^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_4^A \text{ sets } \mathsf{bad}\,\right] . \tag{7}$$

The choice of x_{b1}, x_{b2}, x_b at lines 404, 405 can equivalently be written as first choosing x_b and x_{b2} at random and then setting $x_{b1} = x_b - wx_{b2}$. This is true because w is not equal to 0 modulo p. The same is true for y_{b1}, y_{b2}, y_b. Once this is done, $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ are not used until **Finalize**, so their choice can be delayed. Game G_5 makes these changes, so we have

$$\Pr\left[\,\mathrm{G}_4^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\right] . \tag{8}$$

Game G_6 simply writes the test of line 524 in terms of the exponents. Note that this game computes discrete logarithms, but it is only used in the analysis and does not have to be efficient. We have

$$\Pr\left[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}\,\right] . \tag{9}$$

We claim that

$$\Pr\left[\,\mathrm{G}_6^A \text{ sets } \mathsf{bad}\,\right] \;\leq\; \frac{2q}{p} , \tag{10}$$

(Recall q is the number of **Dec** queries made by A.) We now justify Equation (10). By the time we reach **Finalize** in G_6, we can consider the adversary coins, all random choices of **Initialize**, and all random choices of **Dec** to be fixed. We will take probability only over the choice of x_{b2}, y_{b2} made at line 621. Consider a particular $(a_1, a_2, c, d, v) \in S$. This is now fixed, and so are the quantities $u_1, u_2, s, t_0, t_1, \alpha$ and β as computed at lines 624–626. So we want to bound the probability that bad is set at line 627 when we regard t_b, α, β as fixed and take the probability over the random choices of x_{b2}, y_{b2}. The crucial fact is that $u_2 \neq u_1$ because $(a_1, a_2, c, d, v) \in S$, and lines 612, 613 only put a tuple in S if $a_2 \neq a_1^w$. So α and β are not 0 modulo p, and the probability that $t_b = \alpha x_{b2} + \beta y_{b2}$ is thus $1/p$. The size of S is at most q so line 627 is executed at most $2q$ times. Equation (10) follows from the union bound.

We now return to Equation (5) to bound the first term. Game G_7 removes from G_2 code that does not affect outcome of the game. Once this is done, $x_{b1}, y_{b1}, x_{b2}, y_{b2}$ are used only to define x_b, y_b, so G_7 picks only the latter. So we have

$$\Pr\left[\,\mathrm{G}_2^A\,\right] \;=\; \Pr\left[\,\mathrm{G}_7^A\,\right] . \tag{11}$$

proc Initialize Games G_5,G_6

$g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
$K \xleftarrow{\$} \mathsf{Keys}(H)$; $S \leftarrow \emptyset$
For $b = 0, 1$ do
    $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
    $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Dec$((a_1, a_2, c, d))$ Games G_5,G_6

$v \leftarrow H(K, (a_1, a_2, c))$
For $b = 0, 1$ do $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
If $(a_2 \neq a_1^w)$ Then
    $S \leftarrow S \cup \{(a_1, a_2, c, d, v)\}$; $M_0, M_1 \leftarrow \perp$
Return (M_0, M_1)

proc Finalize Game G_5

For $b = 0, 1$ do
    $x_{b2}, y_{b2} \xleftarrow{\$} \mathbb{Z}_p$
    $x_{b1} \leftarrow x_b - w x_{b2}$; $y_{b1} \leftarrow y_b - w y_{b2}$
    For all $(a_1, a_2, c, d, v) \in S$ do
        If $d = a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then $\mathsf{bad} \leftarrow \mathsf{true}$
Return true

proc Finalize Game G_6

For $b = 0, 1$ do
    $x_{b2}, y_{b2} \xleftarrow{\$} \mathbb{Z}_p$
    $x_{b1} \leftarrow x_b - w x_{b2}$; $y_{b1} \leftarrow y_b - w y_{b2}$
    For all $(a_1, a_2, c, d, v) \in S$ do
        $u_1 \leftarrow \log_{g_1}(a_1)$; $u_2 \leftarrow \log_{g_2}(a_2)$
        $s \leftarrow \log_{g_1}(d)$; $t_b \leftarrow s - u_1 x_b + u_1 y_b v$
        $\alpha \leftarrow w(u_2 - u_1)$; $\beta \leftarrow wv(u_2 - u_1)$
        If $t_b = \alpha x_{b2} + \beta y_{b2}$ Then $\mathsf{bad} \leftarrow \mathsf{true}$
Return true

Figure 15: Games G_5 and G_6 for proof of Theorem 5.1.

proc Initialize Game G_7

700 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
701 $K \xleftarrow{\$} \mathsf{Keys}(H)$
702 For $b = 0, 1$ do
703     $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
704     $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
705 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Dec$((a_1, a_2, c, d))$ Games G_7–G_{11}

710 $v \leftarrow H(K, (a_1, a_2, c))$
711 For $b = 0, 1$ do
712     $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
713   If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b + y_b v})$ Then $M_b \leftarrow \perp$
714 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $\mathrm{WIN} \leftarrow \mathsf{true}$
715 Return (M_0, M_1)

proc Finalize Games G_7–G_{11}

720 Return WIN

proc Initialize Game G_8/$\boxed{G_9}$

800 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$
801 For $b = 0, 1$ do
802     $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
803     $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
804 If $y_1 = y_0$ Then
805     $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}}$
806 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Game G_{10}

1000 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$
1001 $x_0, y_0, x_1 \xleftarrow{\$} \mathbb{Z}_q$; $y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}$
1002 For $b = 0, 1$ do
1003 $z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$
1004 $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
1005 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Game G_{11}

1100 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$; $v^* \xleftarrow{\$} \mathbb{Z}_q$
1101 $x_0, y_0 \xleftarrow{\$} \mathbb{Z}_q$; $y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}$; $x_1 \leftarrow x_0 - (y_1 - y_0)v^*$
1102 For $b = 0, 1$ do $z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
1103 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

Figure 16: Games G_7–G_{11} for proof of Theorem 5.1. G_9 includes the boxed code at line 805 but G_8 does not.

Game G_8 is the same as G_7 barring setting a flag that does not affect the game outcome, so

$$\begin{aligned}
\Pr\left[\,\mathrm{G}_7^A\,\right] &= \Pr\left[\,\mathrm{G}_8^A\,\right] \\
&= \Pr\left[\,\mathrm{G}_9^A\,\right] + \Pr\left[\,\mathrm{G}_8^A\,\right] - \Pr\left[\,\mathrm{G}_9^A\,\right] \\
&\leq \Pr\left[\,\mathrm{G}_9^A\,\right] + \Pr\left[\,\mathrm{G}_8^A \text{ sets } \mathsf{bad}\,\right] && (12)\\
&\leq \Pr\left[\,\mathrm{G}_9^A\,\right] + \frac{1}{p}\,. && (13)
\end{aligned}$$

Equation (12) is by Lemma 2.1 since $\mathrm{G}_8, \mathrm{G}_9$ are identical until bad. The probability that G_8 sets bad is the probability that $y_1 = y_0$ at line 805, and this is $1/p$ since y is chosen at random from $\mathbb{Z}_p$, justifying Equation (13). The distribution of y_1 in G_9 is always uniform over $\mathbb{Z}_q - \{y_0\}$, and the setting of bad at line 805 does not affect the game outcome, so

$$\Pr\left[\,\mathrm{G}_9^A\,\right] \;=\; \Pr\left[\,\mathrm{G}_{10}^A\,\right]\,. \tag{14}$$

Game G_{11} picks x_b, y_b differently from G_{10}, but since $y_1 - y_0 \neq 0$, the two ways induce the same distribution on x_0, x_1, y_0, y_1. Thus,

$$\Pr\left[\,\mathrm{G}_{10}^A\,\right] \;=\; \Pr\left[\,\mathrm{G}_{11}^A\,\right]\,. \tag{15}$$

We now claim that

$$\Pr\left[\,\mathrm{G}_{11}^A\,\right] \;\leq\; \mathbf{Adv}_H^{\text{pre-img}}(I) \tag{16}$$

where I is depicted in Figure 17. To justify this, say that the A makes a **Dec** query (a_1, a_2, c, d) which returns (M_0, M_1) with $M_0 \neq \perp$ and $M_1 \neq \perp$. This means we must have

$$d \;=\; a_1^{x_0 + y_0 v} \;=\; a_1^{x_1 + y_1 v}\,, \tag{17}$$

where $v = H(K, (a_1, a_2, c))$. Let $u_1 = \log_{g_1}(a_1)$ and $s = \log_{g_1}(d)$. Now, the above implies $u_1(x_0 + y_0 v) = u_1(x_1 + y_1 v)$. But (a_1, a_2, c, d) is a **Dec** query, and we know that $a_1 \neq \mathbf{1}$, so $u_1 \neq 0$. (This is a crucial point. Recall the reason we can without loss of generality assume $a_1 \neq \mathbf{1}$ is that the decryption algorithm of $\mathcal{CS}^*$ rejects otherwise.) Dividing u_1 out, we get $x_0 + y_0 v = x_1 + y_1 v$. Rearranging terms, we get $(y_1 - y_0)v = x_0 - x_1$. However, we know that $y_1 \neq y_0$, so $v = (y_1 - y_0)^{-1}(x_0 - x_1)$. However, this is exactly the value v^* due to the way I and Game G_{11} define x_0, y_0, x_1, y_1. Thus, we have $H(K, (a_1, a_2, c)) = v^*$, meaning I will be successful.

Putting together Equations (4)–(11), (13)–(16) concludes the proof of Theorem 5.1. ∎

Adversary $I(K, v^*)$

$g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $x_0, y_0 \xleftarrow{\$} \mathbb{Z}_p$; $y_1 \xleftarrow{\$} \mathbb{Z}_p - \{y_0\}$; $x_1 \leftarrow x_0 - (y_1 - y_0)v^*$

For $b = 0, 1$ do

$z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$

Run A on $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

On query $\mathbf{Dec}((a_1, a_2, c, d))$

$v \leftarrow H(K, (a_1, a_2, c))$

For $b = 0, 1$ do

$M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$

If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b + y_b v})$ Then $M_b \leftarrow \perp$

If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $(a_1^*, a_2^*, c^*) \leftarrow (a_1, a_2, c)$

Return (M_0, M_1) to A

Until A halts

Return (a_1^*, a_2^*, c^*)

Figure 17: Adversary I for proof of Theorem 5.1.

Algorithm PG $g \xleftarrow{\$} \mathbb{G}^*$ Return g	Algorithm $\mathsf{KG}(g)$ $x \xleftarrow{\$} \mathbb{Z}_p$; $y \leftarrow g^x$ Return (y, x)
Algorithm $\mathsf{Enc}(g, y, M)$ $r \xleftarrow{\$} \mathbb{Z}_p^{\boxed{*}}$; $R \leftarrow g^r$ $X \leftarrow y^r$; $SK \| MK \leftarrow H(X)$ $\gamma \leftarrow \mathsf{SEnc}(SK, M)$; $\tau \leftarrow \mathsf{Tag}(MK, \gamma)$ Return (R, γ, τ)	Algorithm $\mathsf{Dec}(g, y, x, (R, \gamma, \tau))$ $X \leftarrow R^x$; $SK \| MK \leftarrow H(X)$ If $\mathsf{Vf}(MK, \gamma, \tau) = 0$ Then $M \leftarrow \perp$ $\boxed{\text{If } R = \mathbf{1} \text{ Then } M \leftarrow \perp}$ $M \leftarrow \mathsf{SDec}(SK, \gamma)$; Return M

Figure 18: The original $\mathcal{DHIES}$ scheme [ABR01] does not contain the boxed code while the variant $\mathcal{DHIES}^*$ does.

6 A SROB-CCA version of DHIES

Let $\mathbb{G}$ be a group of prime order p, let $\mathcal{SE}$ and $\mathcal{MAC}$ be a symmetric encryption and message authentication code (MAC) scheme with key lenghts k_{SE} and k_{M}, respectively, and let $H : \mathbb{G} \mapsto \{0,1\}^{k_{\mathrm{SE}}+k_{\mathrm{M}}}$ be a hash function. The $\mathcal{DHIES}$ public-key encryption scheme depicted in Figure 18 was shown to be IND-CCA in [ABR98] and, in Section 6.1, we show it to be ANO-CCA as well. In terms of robustness, it suffers from a similar problem as the $\mathcal{CS}$ scheme: the ciphertext $(\mathbf{1}, \gamma^*, \tau^*)$ decrypts to M under any key sk for $SK^* \| MK^* \leftarrow H(\mathbf{1})$, $\gamma^* \xleftarrow{\$} \mathsf{SEnc}(SK^*, M)$, and $\tau^* \xleftarrow{\$} \mathsf{Tag}(MK^*, \gamma^*)$, meaning that it is not SROB-CPA. Similarly to the $\mathcal{CS}^*$ scheme, we show that a modified scheme $\mathcal{DHIES}^*$ that excludes the zero randomness and rejects ciphertexts with $\mathbf{1}$ as first component is SROB-CCA.

Symmetric encryption. A symmetric encryption scheme $\mathcal{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ consists of an encryption algorithm SEnc that, on input a k_{SE}-bit key SK and a message M, outputs a ciphertext γ; and a decryption algorithm SDec that, on input a key SK and ciphertext γ outputs a message M. Correctness requires that $\mathsf{SDec}(SK, \mathsf{SEnc}(SK, M)) = M$ with probability one for all $M \in \{0,1\}^*$ and all $SK \in \{0,1\}^{k_{\mathrm{SE}}}$. We require one-time encryption security (OTE) for $\mathcal{SE}$ as defined in Figure 19.

proc Initialize $MK \xleftarrow{\$} \{0,1\}^{k_{\mathrm{M}}}$; $Q \leftarrow \emptyset$ Return **proc Tag**(M) $\tau \xleftarrow{\$} \mathsf{Tag}(MK, M)$ $Q \leftarrow Q \cup \{(M, \tau)\}$ Return τ **proc Verify**(M, τ) $b \xleftarrow{\$} \mathsf{Vf}(MK, M, \tau)$ Return b **proc Finalize**(M^*, τ^*) $b \xleftarrow{\$} \mathsf{Vf}(MK, M^*, \tau^*)$ If $b = 1 \wedge (M^*, \tau^*) \notin Q$ Then return 1 else return 0	**proc Initialize** $SK \xleftarrow{\$} \{0,1\}^{k_{\mathrm{SE}}}$ $b \xleftarrow{\$} \{0,1\}$ Return **proc LR**(M_0^*, M_1^*) If $\lvert M_0^*\rvert \neq \lvert M_1^*\rvert$ then return $\perp$ $\gamma^* \xleftarrow{\$} \mathsf{SEnc}(SK, M_b^*)$ Return γ^* **proc Finalize**(b') Return $(b' = b)$

Figure 19: Games $\mathrm{SUF}_{\mathcal{MAC}}$ (left) and $\mathrm{OTE}_{\mathcal{SE}}$ (right) defining the strong unforgeability of MAC scheme $\mathcal{MAC} = (\mathsf{Tag}, \mathsf{Vf})$ with key length k_{M} and the one-time security of symmetric encryption scheme $\mathcal{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ with key length k_{SE}, respectively.

Message authentication codes. A message authentication code $\mathcal{MAC} = (\mathsf{Tag}, \mathsf{Vf})$ consists of a tagging algorithm Tag that, on input a k_{M}-bit key MK and a message M, outputs a tag τ; and a verification algorithm Vf that, on input a key MK, a message M, and a tag τ, outputs 0 or 1 indicating that the tag is invalid or valid, respectively. Correctness requires that $\mathsf{Vf}(MK, M, \mathsf{Tag}(MK, M, \tau)) = 1$ for all $M \in \{0,1\}^*$ and all $MK \in \{0,1\}^{k_{\mathrm{M}}}$.

Apart from the strong unforgeability (SUF) defined in Figure 19, for robustness we also require *collision resistance* of the MAC scheme, in the sense that it be hard for an adversary to come up with two keys MK_0, MK_1, a message M, and a tag τ that is valid under both keys, i.e., such that $\mathsf{Vf}(MK_0, M, \tau) = \mathsf{Vf}(MK_1, M, \tau) = 1$. Collision-resistant MAC schemes are easy to construct in the random-oracle model and the HMAC scheme [BCK96], where $\mathsf{Tag}(MK, M) = H(MK \oplus \mathtt{opad}, H(MK \oplus \mathtt{ipad}, M))$, naturally satisfies it if the underlying hash function H is collision resistant. We define the collision-finding advantage $\mathbf{Adv}^{\mathrm{coll}}_{\mathcal{MAC}}(A)$ of an adversary A for $\mathcal{MAC}$ as the probability that A outputs a collision as described above. Note that a proper definition of collision resistance would require MAC schemes to be chosen at random from a family, as is done when formally defining collision resistance for hash functions. We refrain from doing so to avoid overloading our notation.

Oracle Diffie-Hellman. We recall the oracle Diffie-Hellman (ODH) problem from [ABR98] in Figure 20. The adversary's goal is to distinguish the hash of a Diffie-Hellman solution from a random string when given access to an oracle that returns hash values of Diffie-Hellman solutions of any other group elements than the target group element. The advantage of an adversary A to solve the ODH problem is defined as

$$\mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G},H}(A) = 2 \cdot \Pr\left[\mathrm{ODH}^{A}_{\mathbb{G},H} \Rightarrow \mathsf{true}\right] - 1 \ .$$

The proof by [ABR98] that $\mathcal{DHIES}$ is IND-CCA relies on the assumption that ODH is hard; we use the same assumption here to prove that is also ANO-CCA.

We also introduce a double-challenge variant of ODH called ODH2 in Figure 20 and its associated advantage as $\mathbf{Adv}^{\mathrm{odh2}}_{\mathbb{G},H}(A) = 2 \cdot \Pr\left[\mathrm{ODH2}^{A}_{\mathbb{G},H} \Rightarrow \mathsf{true}\right] - 1$. The following lemma shows that the hardness of the ODH2 problem is implied by that of the ODH problem, but the

proc Initialize	**proc Initialize**
$b \xleftarrow{\$} \{0,1\}$	$b \xleftarrow{\$} \{0,1\}$
$g, Y \xleftarrow{\$} \mathbb{G}^* \; ; \; x \xleftarrow{\$} \mathbb{Z}_p^*$	$g, Y \xleftarrow{\$} \mathbb{G}^* \; ; \; x_0, x_1 \xleftarrow{\$} \mathbb{Z}_p^*$
$X \leftarrow g^x$	$X_0 \leftarrow g^{x_0} \; ; \; X_1 \leftarrow g^{x_1}$
If $b = 0$	If $b = 0$
Then $Z \leftarrow H(Y^x)$	Then $Z_0 \leftarrow H(Y^{x_0}) \; ; \; Z_1 \leftarrow H(Y^{x_1})$
Else $Z \xleftarrow{\$} \{0,1\}^\ell$	Else $Z_0, Z_1 \xleftarrow{\$} \{0,1\}^\ell$
Return (g, X, Y, Z)	Return $(g, X_0, X_1, Y, Z_0, Z_1)$
proc HDH(W)	**proc HDH2**(W)
If $W = Y$ or $W \notin \mathbb{G}$	If $W = Y$ or $W \notin \mathbb{G}$
Then return $\perp$	Then return $\perp$
Return $H(W^x)$	Return $(H(W^{x_0}), H(W^{x_1}))$
proc Finalize(b')	**proc Finalize**(b')
Return $(b' = b)$	Return $(b' = b)$

Figure 20: Games $\mathrm{ODH}_{\mathbb{G},H}$ (left) and $\mathrm{ODH2}_{\mathbb{G},H}$ (right) defining the oracle Diffie-Hellman (ODH) problem and the double ODH problem in $\mathbb{G}$ with respect to hash function $H : \mathbb{G} \mapsto \{0,1\}^\ell$, respectively.

proc Initialize Game G_0

$g, Y \xleftarrow{\$} \mathbb{G}^* \; ; \; x_0, x_1 \xleftarrow{\$} \mathbb{Z}_p^*$
$X_0 \leftarrow g^{x_0} \; ; \; X_1 \leftarrow g^{x_1}$
$Z_0 \leftarrow H(Y^{x_0}) \; ; \; Z_1 \leftarrow H(Y^{x_1})$
Return $(g, X_0, X_1, Y, Z_0, Z_1)$

proc HDH2(W) Games G_0, G_1, G_2

If $W = Y$ or $W \notin \mathbb{G}$
Then return $\perp$
Return $(H(W^{x_0}), H(W^{x_1}))$

proc Finalize(b') Game G_0, G_1, G_2

Return $(b' = 0)$

proc Initialize Game G_1

$g, Y \xleftarrow{\$} \mathbb{G}^* \; ; \; x_0, x_1 \xleftarrow{\$} \mathbb{Z}_p^*$
$X_0 \leftarrow g^{x_0} \; ; \; X_1 \leftarrow g^{x_1}$
$Z_0 \xleftarrow{\$} \{0,1\}^\ell \; ; \; Z_1 \leftarrow H(Y^{x_1})$
Return $(g, X_0, X_1, Y, Z_0, Z_1)$

proc Initialize Game G_2

$g, Y \xleftarrow{\$} \mathbb{G}^* \; ; \; x_0, x_1 \xleftarrow{\$} \mathbb{Z}_p^*$
$X_0 \leftarrow g^{x_0} \; ; \; X_1 \leftarrow g^{x_1}$
$Z_0, Z_1 \xleftarrow{\$} \{0,1\}^\ell$
Return $(g, X_0, X_1, Y, Z_0, Z_1)$

Figure 21: Games G_0, G_1, and G_2 for the proof of Lemma 6.1.

ODH2 problem is easier to work with in our proofs.

Lemma 6.1 *Let A be an adversary with advantage $\mathbf{Adv}^{\mathrm{odh2}}_{\mathbb{G},H}(A)$ in solving the* ODH2 *problem. Then there exists an adversary B such that $\mathbf{Adv}^{\mathrm{odh2}}_{\mathbb{G},H}(A) \leq 2 \cdot \mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G},H}(B)$.*

Proof: Consider the sequence of games G_0, G_1, and G_2 in Figure 21. Game G_0 is identical to the $\mathrm{ODH2}_{\mathbb{G},H}$ game in the case that $b = 0$. Game G_2 is almost identical to $\mathrm{ODH2}_{\mathbb{G},H}$ in the case that $b = 1$, except that it returns true when $\mathrm{ODH2}_{\mathbb{G},H}$ returns false and vice versa. We therefore have that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{odh2}}_{\mathbb{G},H}(A) &= 2 \cdot \Pr\left[\,\mathrm{ODH2}^A_{\mathbb{G},H} \Rightarrow \mathsf{true}\,\right] - 1 \\
&= \Pr\left[\,\mathrm{ODH2}^A_{\mathbb{G},H} \Rightarrow \mathsf{true} \;\middle|\; b = 0\,\right] + \Pr\left[\,\mathrm{ODH2}^A_{\mathbb{G},H} \Rightarrow \mathsf{true} \;\middle|\; b = 1\,\right] - 1 \\
&= \Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_2^A \Rightarrow \mathsf{true}\,\right] . \qquad (18)
\end{aligned}$$

Game G_1 differs from G_0 in that Z_0 is chosen at random from $\{0,1\}^\ell$, instead of computed as

$H(Y^{x_0})$. We claim that there exists an algorithm B_1 such that

$$\Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right] = \mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G},H}(B_1)\ . \tag{19}$$

Namely, on initial input (g, X, Y, Z), B_1 chooses $x_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $X_0 \leftarrow X$, $X_1 \leftarrow g^{x_1}$, $Z_0 \leftarrow Z$, and $Z_1 \leftarrow H(Y^{x_1})$. It then runs A on initial input $(g, X_0, X_1, Y, Z_0, Z_1)$, answering its $\mathbf{HDH2}(W)$ queries as $\mathbf{HDH}(W)\|H(W^{x_1})$. When A outputs b', B_1 also outputs b'.

It is clear that B_1 provides A with a perfect simulation of game G_0 if the challenge bit b in B_1's $\mathrm{ODH2}_{\mathbb{G},H}$ game is zero, and of game G_1 if $b = 1$. We therefore have that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G},H}(B_1) &= \Pr\left[\,\mathrm{ODH}^{B_1}_{\mathbb{G},H} \Rightarrow \mathsf{true} \mid b = 0\,\right] + \Pr\left[\,\mathrm{ODH}^{B_1}_{\mathbb{G},H} \Rightarrow \mathsf{true} \mid b = 1\,\right] - 1 \\
&= \Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] + (1 - \Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right]) - 1 \\
&= \Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right]\ .
\end{aligned}$$

By a similar reasoning, there exists an algorithm B_2 such that

$$\Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_2^A \Rightarrow \mathsf{true}\,\right] = \mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G},H}(B_2)\ . \tag{20}$$

Putting Equations (18), (19), and (20) together and letting B be the algorithm of B_1 or B_2 with the highest advantage yields the lemma statement. ▮

6.1 Anonymity of DHIES

The $\mathcal{DHIES}$ scheme was already proved to be IND-CCA secure [ABR98], so to prove AI-CCA security, we only have left to prove ANO-CCA security. As mentioned in Section 2, the ANO-CCA security game is the AI-CCA game in Figure 3 with an added restriction that two equal challenge messages $M_0^* = M_1^*$ must be submitted to the **LR** oracle.

Theorem 6.2 *Let $\mathcal{DHIES}$ be the general encryption scheme associated to group $\mathbb{G}$, symmetric encryption scheme $\mathcal{SE}$, message authentication code $\mathcal{MAC}$, and hash function $H : \mathbb{G} \mapsto \{0,1\}^{k_{\mathrm{SE}}+k_{\mathrm{M}}}$ as per Figure 18. Let A be an* ano-cca *adversary against $\mathcal{DHIES}$ that makes two **GetEK** queries, no **GetDK** queries and at most q **Dec** queries. Then there exist an* ODH2 *adversary B against $\mathbb{G}$ and an adversary C against the strong unforgeability of $\mathcal{MAC}$ such that*

$$\mathbf{Adv}^{\mathrm{ano\text{-}cca}}_{\mathcal{DHIES}}(A) \ \le\ 2 \cdot \mathbf{Adv}^{\mathrm{odh}}_{\mathbb{G}}(B) + \mathbf{Adv}^{\mathrm{suf}}_{\mathcal{MAC}}(C)\ .$$

*Adversaries B, C have the same running time as A, and adversary B makes q **Dec** queries.*

Since $\mathcal{DHIES}$ is a PKE scheme, the above implies security for multiple **GetEK** and **GetDK** queries as required by the ANO-CCA game. The above result easily extends to $\mathcal{DHIES}^*$ as well, because the exclusion of $r = 0$ from encryption and $R = \mathbf{1}$ from decryption only affect the ANO-CCA game if $R^* = \mathbf{1}$ in the challenge ciphertext C^*, which only happens with probability $1/p$.

Proof of Theorem 6.2: In Figure 22, we depict Games G_0 and G_1 used in the proof. Game G_0 differs from the original ANO-CCA game in that the challenge ciphertext uses symmetric encryption and MAC keys that are randomly chosen (in line 002) rather than computed as $SK^*\|MK^* \leftarrow H(R^*)$. The changes to **Dec** are purely cosmetic. We first show that for any ANO-CCA adversary A, there exists an ODH2 adversary B_2 such that

$$\mathbf{Adv}^{\mathrm{odh2}}_{\mathbb{G},H}(B_2) \ =\ \Pr\left[\,\mathrm{ANO\text{-}CCA}^A_{\mathcal{DHIES}} \Rightarrow \mathsf{true}\,\right] + \Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] - 1\ . \tag{21}$$

proc Initialize Game G_0, G_1

000 $g \xleftarrow{\$} \mathbb{G}^*$; $b \xleftarrow{\$} \{0,1\}$
001 $R^* \xleftarrow{\$} \mathbb{G}^*$
002 $SK^* \xleftarrow{\$} \{0,1\}^{k_{\mathrm{SE}}}$; $MK^* \xleftarrow{\$} \{0,1\}^{k_{\mathrm{M}}}$
003 $S, T, U, V \leftarrow \emptyset$
004 Return g

proc Dec$(C = (R, \gamma, \tau), id)$ Game $\boxed{G_0}$/G_1

005 If $id \notin U$ then return $\perp$
006 If $(id, C) \in T$ then return $\perp$
007 If $R = R^*$ then
008 $\quad M \leftarrow \perp$
009 $\quad$ If $\mathsf{Vf}(MK^*, \gamma, \tau) = 1$ then
010 $\qquad \mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{M \leftarrow \mathsf{SDec}(SK^*, \gamma)}$
011 Else $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
012 Return M

proc LR$(id_0^*, id_1^*, M_0^*, M_1^*)$ Game G_0, G_1

013 If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\perp$
014 If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\perp$
015 If $M_0^* \neq M_1^*$ then return $\perp$
016 $\gamma^* \leftarrow \mathsf{SEnc}(SK^*, M_b^*)$; $\tau^* \leftarrow \mathsf{Tag}(MK^*, \gamma^*)$
017 $C^* \leftarrow (R^*, \gamma^*, \tau^*)$
018 $S \leftarrow S \cup \{id_0^*, id_1^*\}$
019 $T \leftarrow T \cup \{(id_0^*, C^*), (id_1^*, C^*)\}$
020 Return C^*

proc Finalize(b') Game G_0, G_1

021 Return $(b' = b)$

Figure 22: Games G_0 and G_1 for the proof of Theorem 6.2. Game G_0 includes the boxed code at line 009 but G_1 does not.

Namely, on initial input $(g, X_0, X_1, Y, Z_0, Z_1)$, adversary B_2 chooses $b \xleftarrow{\$} \{0,1\}$ and runs A on initial input g and returns X_0 and X_1 as the two public encryption keys of A's **GetEK** queries id_0 and id_1.

To simulate A's **LR** query, B_2 sets $R^* \leftarrow Y$, parses Z_b as $SK^* \| MK^*$, and computes $\gamma^* \xleftarrow{\$} \mathsf{SEnc}(SK^*, M_b^*)$ and $\tau^* \xleftarrow{\$} \mathsf{Tag}(MK^*, \gamma^*)$. It returns $C^* = (R^*, \gamma^*, \tau^*)$ as the challenge ciphertext.

To answer A's **Dec**$((R, \gamma, \tau), id_d)$ queries, B_2 proceeds as follows. If $R \neq Y$, then B_2 queries its ODH2 oracle to obtain $SK \| MK \leftarrow \mathrm{ODH2}(R)$. If $R = Y$, it parses Z_d as $SK \| MK$. In both cases, it checks that $\mathsf{Vf}(MK, \gamma) = 1$, and, if so, returns $M \leftarrow \mathsf{SDec}(SK, \gamma)$. When A outputs its guess b', B_2 outputs $(b = b')$.

Let b_2 be the random bit chosen by B_2's challenger in the ODH2 game that B_2 has to guess. In the case that $b_2 = 0$, we have that $Z_0 = H(Y^{x_0})$ and $Z_1 = H(Y^{x_1})$, so that all symmetric encryption and MAC keys that B_2 used for the challenge ciphertext and to simulate A's decryption queries are exactly as in the real $\mathcal{DHIES}$ scheme. In the case that $b_2 = 1$, Z_0 and Z_1 are random strings, so that the challenge ciphertext and decryption responses are exactly as in Game G_0. We therefore have that

$$\Pr\left[\mathrm{ODH2}_{\mathbb{G},H}^{B_2} \Rightarrow \mathsf{true}\right] = \frac{1}{2} \cdot \Pr\left[\text{ANO-CCA}_{\mathcal{DHIES}}^{A} \Rightarrow \mathsf{true}\right] + \frac{1}{2} \cdot \Pr\left[G_0^A \Rightarrow \mathsf{true}\right] .$$

so that Equation (21) follows.

Games G_0 and G_1 are identical until bad on line 009 in Figure 22, so by Lemma 2.1, we have that

$$\left|\Pr\left[G_0^A \Rightarrow \mathsf{true}\right] - \Pr\left[G_1^A \Rightarrow \mathsf{true}\right]\right| \leq \Pr\left[G_1^A \text{ sets } \mathsf{bad}\right] . \tag{22}$$

For any adversary A that makes Game G_1 set bad, we construct an adversary C against the strong unforgeability of the MAC scheme so that

$$\Pr\left[G_1^A \text{ sets } \mathsf{bad}\right] = \mathbf{Adv}_{\mathsf{MAC}}^{\mathrm{suf}}(C) . \tag{23}$$

Namely, C chooses $g, R^* \xleftarrow{\$} \mathbb{G}^*$, $SK^* \xleftarrow{\$} \{0,1\}^{k_{\mathrm{SE}}}$, and $b \xleftarrow{\$} \{0,1\}$ as in Game G_1, but rather than choosing a MAC key MK^*, it uses its **Tag** and **Verify** oracles for all operations involving MK^*. More precisely, when answering A's **LR** query, it sets $\tau^* \leftarrow \mathbf{Tag}(\gamma^*)$. When A sets bad, i.e., makes a query $\mathbf{Dec}(C = (R, \gamma, \tau))$ with $R = R^*$ and $\mathbf{Verify}(\gamma, \tau) = 1$, then C returns its forgery (γ, τ). By line 006 we have that $C \neq (R^*, \gamma^*, \tau^*)$, so that $(\gamma, \tau) \neq (\gamma^*, \tau^*)$ and therefore (γ, τ) is a valid forgery.

Note that because $M_0^* = M_1^*$, A's view in Game G_1 is independent of the bit b, hence

$$\Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right] \;=\; \frac{1}{2}\,. \tag{24}$$

By the definition of ANO-CCA advantage we have

$$\begin{aligned}
\mathbf{Adv}^{\text{ano-cca}}_{\mathcal{DHIES}}(A) \;&=\; 2 \cdot \Pr\left[\,\text{ANO-CCA}^A_{\mathcal{DHIES}} \Rightarrow \mathsf{true}\,\right] - 1 \\
&=\; 2 \cdot \mathbf{Adv}^{\text{odh2}}_{\mathbb{G},H}(B_2) - 2 \cdot \Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] + 1 \\
&\leq\; 4 \cdot \mathbf{Adv}^{\text{odh}}_{\mathbb{G},H}(B) - 2 \cdot \left(\Pr\left[\,\mathrm{G}_0^A \Rightarrow \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right]\right) \\
&\qquad - 2 \cdot \Pr\left[\,\mathrm{G}_1^A \Rightarrow \mathsf{true}\,\right] + 1 \\
&\leq\; 4 \cdot \mathbf{Adv}^{\text{odh}}_{\mathbb{G},H}(B) + 2 \cdot \mathbf{Adv}^{\text{suf}}_{\mathsf{MAC}}(C)
\end{aligned}$$

where the first step is due to Equation (21), the second is by considering the ODH adversary B from Lemma 6.1, and the third is due to Equation (22), (23), and Equation (24). ▮

6.2 Robustness of DHIES

Theorem 6.3 *Let $\mathcal{DHIES}^*$ be the general encryption scheme associated to group $\mathbb{G}$, symmetric encryption scheme $\mathcal{SE}$, message authentication code $\mathcal{MAC}$, and hash function $H : \mathbb{G} \mapsto \{0,1\}^{k_{\mathrm{SE}}+k_{\mathrm{M}}}$ as per Figure 18. Let $H_{\mathrm{M}} : \mathbb{G} \mapsto \{0,1\}^{k_{\mathrm{M}}}$ be the function that outputs the last k_{M} bits of $H(x)$ on input $x \in \mathbb{G}$. Let A be an* srob-cca *adversary against $\mathcal{DHIES}$, making at most $q_{\textbf{GetEK}}$ queries to its* **GetEK** *oracle. Then there exist collision-finding adversaries B and C against H_{M} and $\mathcal{MAC}$, respectively, such that*

$$\mathbf{Adv}^{\text{srob}}_{\mathcal{DHIES}}(A) \;\leq\; \mathbf{Adv}^{\text{coll}}_{H_{\mathrm{M}}}(B) + \mathbf{Adv}^{\text{coll}}_{\mathcal{MAC}}(C) + \binom{q_{\textbf{GetEK}}}{2}/p\,.$$

Adversaries B and C have the same running time as A.

The proof intuition for the strong robustness of $\mathcal{DHIES}^*$ is quite straightforward. Let (C, id_0, id_1) be the output of a SROB-CCA adversary A where $C = (R, \gamma, \tau)$ and (id_0, id_1) are the identities associated with two different public keys $y_0 = g^{x_0}$ and $y_1 = g^{x_1}$. Let $SK_b \| MK_b \leftarrow H(y_b^r)$ for $b \in \{0,1\}$. First, $y_0^r \neq y_1^r$ since $R \neq \mathbf{1}$ and $y_0 \neq y_1 \neq \mathbf{1}$ with overwhelming probability. Second $MK_0 \neq MK_1$ with all but negligible probability since the probability that $H_{\mathrm{M}}(y_0^r) = H_{\mathrm{M}}(y_1^r)$ is negligible due to the collision resistance of H_{M}. Third, the probability that C is valid with respect to both y_0 and y_1 (i.e., $\mathsf{Dec}(g, y_0, x_0, C) \neq \perp$ and $\mathsf{Dec}(g, y_1, x_1, C) \neq \perp$) is negligible since the probability that $\mathsf{Vf}(MK_0, \gamma, \tau) = \mathsf{Vf}(MK_1, \gamma, \tau) = 1$ is negligible due to the collision resistance of $\mathcal{MAC}$. Finally, the latter is true even when A knows the corresponding secret keys x_0 and x_1 associated with y_0 and y_1 so **GetDK** and **Dec** are of no help to A.

Proof of Theorem 6.3: In order to prove the strong robustness of $\mathcal{DHIES}$, we consider a SROB-CCA adversary A that even knows the secret decryption key associated with each public key that it obtains via **GetEK** queries (and therefore can decrypt all the ciphertexts that it wants). That is, whenever A issues a **GetEK** query id, the challenger in the SROB-CCA game runs the key generation algorithm $\mathsf{KG}(g)$ to obtain a fresh pair of secret and public keys $(x, y = g^x)$ for id and returns both values to A. Hence, A can compute the answer to **GetDK** and **Dec** queries on its own.

Let (C, id_0, id_1) be the output of a SROB-CCA adversary A where $C = (R, \gamma, \tau)$ and (id_0, id_1) are the identities associated with two different public keys $y_0 = g^{x_0}$ and $y_1 = g^{x_1}$. Moreover, let $MK_b \leftarrow H_{\mathrm{M}}(y_b^r)$ for $b \in \{0,1\}$ denote the corresponding MAC keys. In order for A to be successful, one of the following cases need to occur:

(1) $y_0 = y_1$;

(2) $H_{\mathrm{M}}(y_0^r) = H_{\mathrm{M}}(y_1^r)$;

(3) $\mathsf{Vf}(MK_0, \gamma, \tau) = \mathsf{Vf}(MK_1, \gamma, \tau) = 1$.

Since public keys are generated honestly, the probability that $y_0 = y_1$ (i.e., Case (1)) can be upper-bounded by the probability that **GetEK** oracle generates the same public and secret keys for two different id values, which is at most $\binom{q_{\mathbf{GetEK}}}{2}/p$.

Assuming that $y_0 \neq y_1$ and since $R \neq \mathbf{1}$, it is easy to construct a collision-finding adversary B against H_{M} such that the probability that $H_{\mathrm{M}}(y_0^r) = H_{\mathrm{M}}(y_1^r)$ (i.e., Case (2)) is at most $\mathbf{Adv}^{\mathrm{coll}}_{H_{\mathrm{M}}}(B)$. Adversary B works as follows. B starts by running A, providing the latter with a generator g for the group $\mathbb{G}$. Whenever A issues a **GetEK** query id, B runs the key generation algorithm $\mathsf{KG}(g)$ to obtain a fresh pair of secret and public keys $(x, y = g^x)$ for id and returns both values to A. Finally, when A issues a **Finalize** query (C, id_0, id_1), where $C = (R, \gamma, \tau)$, let $(x_b, y_b = g^{x_b})$ be the secret and public key pair associated with id_b for $b \in \{0,1\}$. B simply outputs R^{x_0} and R^{x_1} as a collision for H_{M}. Clearly, B wins whenever $H_{\mathrm{M}}(y_0^r) = H_{\mathrm{M}}(y_1^r)$. Hence, the probability that $H_{\mathrm{M}}(y_0^r) = H_{\mathrm{M}}(y_1^r)$ is at most $\mathbf{Adv}^{\mathrm{coll}}_{H_{\mathrm{M}}}(B)$.

Finally, if we assume that $H_{\mathrm{M}}(y_0^r) \neq H_{\mathrm{M}}(y_1^r)$, then it is easy to construct a collision-finding adversary C against $\mathcal{MAC}$ such that the probability that $\mathsf{Vf}(MK_0, \gamma, \tau) = \mathsf{Vf}(MK_1, \gamma, \tau) = 1$ (i.e., Case (3)) is at most $\mathbf{Adv}^{\mathrm{coll}}_{\mathcal{MAC}}(C)$. Adversary C works as follows. C starts by running A, providing the latter with a generator g for the group $\mathbb{G}$. Whenever A issues a **GetEK** query id, C runs the key generation algorithm $\mathsf{KG}(g)$ to obtain a fresh pair of secret and public keys $(x, y = g^x)$ for id and returns both values to A. Finally, when A issues a **Finalize** query (C, id_0, id_1), where $C = (R, \gamma, \tau)$, let $(x_b, y_b = g^{x_b})$ be the secret and public key pair associated with id_b and let $MK_b \leftarrow H_{\mathrm{M}}(R^{x_b})$ for $b \in \{0,1\}$. C simply outputs (MK_0, MK_1) as the two MAC keys, γ as the message, and τ as the tag. Clearly, C wins whenever $\mathsf{Vf}(MK_0, \gamma, \tau) = \mathsf{Vf}(MK_1, \gamma, \tau) = 1$. Hence, the probability that $\mathsf{Vf}(MK_0, \gamma, \tau) = \mathsf{Vf}(MK_1, \gamma, \tau) = 1$ is at most $\mathbf{Adv}^{\mathrm{coll}}_{\mathcal{MAC}}(C)$. ∎

7 Other schemes and transforms

In this section we show that neither of two popular IND-CCA-providing transforms, the Fujisaki-Okamoto (FO) transform [FO99] in the random oracle model and the Canetti-Halevi-Katz (CHK) transform [CHK04, BCHK07] in the standard model, yield robustness. Since the FO transform even provides the stronger notion of plaintext awareness [BP04], the counterexample below is at the same time a proof that even plaintext awareness does not suffice for robustness. The fact that neither of the transforms confer robustness generically does not exclude that they may still do so for certain specific schemes. We show that this is actually the case for the Boneh-Franklin IBE [BF01], which uses the FO transform to obtain IND-CCA security, and that it is *not* the case for the Boyen-Waters IBE [BW06], which uses the CHK transform.

THE FO TRANSFORM. Given a public-key encryption scheme $\mathcal{PKE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ the FO transform yields a PKE scheme $\overline{\mathcal{PKE}} = (\mathsf{PG}, \mathsf{KG}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ where a message M is encrypted as

$$\big(\ \mathsf{Enc}(pars, pk, x; H(x, M))\ \ ,\ \ G(x) \oplus M\ \big)\ ,$$

where $x \xleftarrow{\$} \{0,1\}^k$, where $G(\cdot)$ and $H(\cdot)$ are random oracles, and where $H(x, M)$ is used as the random coins for the Enc algorithm. To decrypt a ciphertext (C_1, C_2), one recovers x by decrypting C_1, recovers $M \leftarrow C_2 \oplus G(x)$, and checks that $\mathsf{Enc}(pars, pk, x; H(x, M)) = C_1$. If this is the case then M is returned, otherwise $\perp$ is returned.

Given a scheme $\mathcal{PKE}^* = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}^*, \mathsf{Dec}^*)$, we show how to build a scheme $\mathcal{PKE} =$ $(\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}$ such that $\overline{\mathcal{PKE}}$ obtained by applying the FO transform to $\mathcal{PKE}$ is not SROB-CPA. Namely, for some fixed $x^* \in \{0,1\}^k$ and M^*, let encryption and decryption be given by

Algorithm $\mathsf{Enc}(pars, pk, x; \rho)$	Algorithm $\mathsf{Dec}(pars, pk, sk, b\|C^*)$
If $x = x^*$ and $\rho = H(x^*, M^*)$ then return 0	If $b = 0$ then return x^*
Else return $1\|\mathsf{Enc}^*(pars, pk, x; \rho)$	Else return $\mathsf{Dec}^*(pars, pk, sk, C^*)$.

It is easy to see that if $\mathcal{PKE}^*$ is one-way (the notion required by the FO transform), then so is $\mathcal{PKE}$, because for an honestly-generated ciphertext the random coins $H(x^*, M^*)$ will hardly ever occur. Moreover, it is also straightforward to show that, if $\mathcal{PKE}^*$ is γ-uniform, then $\mathcal{PKE}$ is γ'-uniform for $\gamma' = \max(\gamma, 1/2^\ell)$, where ℓ is the output length of H (please refer to [FO99] for the definition of γ-uniformity). It is also easy to see that the scheme $\overline{\mathcal{PKE}}$ obtained by applying the FO transform to $\mathcal{PKE}$ is not robust: the ciphertext $\overline{C} = (0\ ,\ G(x^*) \oplus M^*)$ decrypts correctly to M^* under any public key.

THE BONEH-FRANKLIN IBE. Boneh and Franklin proposed the first truly practical provably secure IBE scheme in [BF01]. They also propose a variant that uses the FO transform to obtain provable IND-CCA security in the random oracle model under the bilinear Diffie-Hellman (BDH) assumption; we refer to it as the BF-IBE scheme here. A straightforward modification of the proof can be used to show that BF-IBE is also ANO-CCA in the random oracle model under the same assumption. We now give a proof sketch that BF-IBE is also (unconditionally) SROB-CCA in the random oracle model.

Let e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a non-degenerate bilinear map, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative cyclic groups of prime order p [BF01]. Let g be a generator of $\mathbb{G}_1$. The master secret key of the BF-IBE scheme is an exponent $s \xleftarrow{\$} \mathbb{Z}_p^*$, the public parameters contain $S \leftarrow g^s$. For random oracles $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^k$, $H_3 : \{0,1\}^k \times \{0,1\}^\ell \to \mathbb{Z}_p^*$, and $H_4 : \{0,1\}^k \to \{0,1\}^\ell$, the encryption of a message M under identity id is a tuple

$$\big(g^r\ ,\ x \oplus H_2(e(S, H_1(id))^r)\ ,\ M \oplus H_4(x)\big)\ ,$$

where $x \xleftarrow{\$} \{0,1\}^k$ and $r \leftarrow H_3(x, M)$. To decrypt a ciphertext (C_1, C_2, C_3), the user with identity id and decryption key $dk = H_1(id)^s$ computes $x \leftarrow C_2 \oplus H_2(e(C_1, dk))$, $M \leftarrow C_3 \oplus H_4(x)$, and $r \leftarrow H_3(x, M)$. If $C_1 \neq g^r$ he rejects, otherwise he outputs M.

Let us now consider a SROB-CCA adversary A that even knows the master secret s (and therefore can derive all keys and decrypt all ciphertexts that it wants). Since H_1 maps into $\mathbb{G}_1^*$, all its outputs are of full order p. The probability that A finds two identities id_1 and id_2 such that $H_1(id) = H_1(id_2)$ is negligible. Since $S \in \mathbb{G}_1^*$ and the map is non-degenerate, we therefore have that $g_{id_1} = e(S, H_1(id_1))$ and $g_{id_2} = e(S, H_1(id_2))$ are different and of full order p. Since H_3 maps into $\mathbb{Z}_p^*$, we have that $r \neq 0$, and therefore that $g_{id_1}^r$ and $g_{id_2}^r$ are different. If the output of H_2 is large enough to prevent collisions from being found, that also means that $H_2(g_{id_1}^r)$ and $H_2(g_{id_2}^r)$ are different. Decryption under both identities therefore yields two different values $x_1 \neq x_2$, and possibly different messages M_1, M_2. In order for the ciphertext to be valid for both identities, we need that $r = H_3(x_1, M_1) = H_3(x_2, M_2)$, but the probability of this happening is again negligible in the random oracle model. As a result, it follows that the BF-IBE scheme is also SROB-CCA in the random oracle model.

The Canetti-Halevi-Katz transform. The CHK transform turns an IBE scheme and a one-time signature scheme [EGM96, Lam79] into a PKE scheme as follows. For each ciphertext a fresh signature key pair (spk, ssk) is generated. The ciphertext is a tuple (C, spk, σ) where C is the encryption of M to identity spk and σ is a signature of C under ssk. To decrypt, one verifies the signature σ, derives the decryption key for identity spk, and decrypts C.

Given a scheme $\mathcal{IBE}^* = (\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}^*, \mathsf{Dec}^*)$, consider the scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Enc}(pars, id, M) = 1 \| \mathsf{Enc}^*(pars, id, M)$ and where $\mathsf{Dec}(pars, id, dk, b \| C^*)$ returns $\mathsf{Dec}^*(pars, id, dk, C^*)$ if $b = 1$ and simply returns C^* if $b = 0$. This scheme clearly inherits the privacy and anonymity properties of $\mathcal{IBE}^*$. However, if $\mathcal{IBE}$ is used in the CHK transformation, then one can easily generate a ciphertext $(0 \| M, spk, \sigma)$ that validly decrypts to M under any parameters $pars$ (which in the CHK transform serve as the user's public key).

An extension of the CHK transform turns any IND-CPA secure $\ell + 1$-level hierarchical IBE (HIBE) into an IND-CCA secure ℓ-level HIBE. It is easy to see that this transform does not confer robustness either.

The Boyen-Waters IBE. Boyen and Waters [BW06] proposed a HIBE scheme which is IND-CPA and ANO-CPA in the standard model, and a variant that uses the CHK transform to achieve IND-CCA and ANO-CCA security. Decryption in the IND-CPA secure scheme never rejects, so it is definitely not WROB-CPA. Without going into details here, it is easy to see that the IND-CCA variant is not WROB-CPA either, because any ciphertext that is valid with respect to one identity will also be valid with respect to another identity, since the verification of the one-time signature does not depend on the identity of the recipient. (The natural fix to include the identity in the signed data may ruin anonymity.)

The IND-CCA-secure variant of Gentry's IBE scheme [Gen06] falls to a similar robustness attack as the original Cramer-Shoup scheme, by choosing a random exponent $r = 0$. We did not check whether explicitly forbidding this choice restores robustness, however.

Composite-order pairing-based schemes. As mentioned in the introduction, a number of encryption schemes based on composite-order bilinear maps satisfy a variant of our weak robustness notion [BW07, KSW08]. They achieve this by restricting the message space to a negligible fraction of the group and by proving that decryption of a ciphertext with an incorrect secret key yields a message with a random component in one of the subgroups. This message has a negligible probability of falling within the valid message space. It is unclear whether the same approach can be used to satisfy our robustness notions or whether it extends to other schemes.

There is growing recognition that robustness is important in applications and worth defining explicitly, supporting our own claims to this end. In particular, the strong correctness requirement for public-key encryption [BBW06] and the correctness requirement for hidden-vector and predicate encryption [BW07, KSW08] implies a form of weak robustness. In work concurrent to, and independent of, ours, Hofheinz and Weinreb [HW08] introduced a notion of *well-addressedness* of IBE schemes that is just like weak robustness except that the adversary gets the IBE master secret key.

Neither of these works considers or achieves strong robustness, and neither treats PKE. Well-addressedness of IBE implies WROB-CCA but does not imply SROB-CCA and, on the other hand, SROB-CCA does not imply well-addressedness. Note that the term robustness is also used in multi-party computation to denote the property that corrupted parties cannot prevent honest parties from computing the correct protocol output [GMW87, BOGW88, HM01]. This meaning is unrelated to our use of the word robustness.

8 Application to auctions

Robustness of ElGamal. The parameters of the ElGamal encryption scheme consist of the description of a group $\mathbb{G}$ of prime order p with generator g. The secret key of a user is $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, the corresponding public key is $X = g^x$. The encryption of a message M is the pair $(g^r, X^r \cdot M)$ for $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. A ciphertext (R, S) is decrypted as $M \leftarrow R/S^x$. Since the decryption algorithm never returns $\perp$, the ElGamal scheme is obviously not robust. Stronger even, the ciphertext $(1, M)$ decrypts to M under any secret key.

It is this strong failure of robustness that opens the way to attacks on applications like Sako's auction protocol [Sak00].

The protocol. Sako's auction protocol [Sak00] is important because it is the first truly practical one to hide the bids of losers. Let $1, \ldots, N$ be the range of possible bidding prices. In an initialization step, the auctioneer generates N ElGamal key pairs $(x_1, X_1), \ldots, (x_N, X_N)$, and publishes $g, X_1, \ldots, X_N$ and a fixed message $M \in \mathbb{G}$. A bidder places a bid of value $v \in \{1, \ldots, N\}$ by encrypting M under X_v and posting the ciphertext. Note that the privacy of the bids is guaranteed by the anonymity of ElGamal encryption. The authority opens bids $C_1 = (R_1, S_1), \ldots, C_n = (R_n, S_n)$ by decrypting all bids under secret keys $x_N, \ldots, x_1$, until the highest index w where one or more bids decrypt to M. The auctioneer announces the identity of the winner(s), the price of the item w, and the secret key x_w. All auctioneers can then check that $S_i/R_i^{x_w} = M$ for all winners i.

An attack. Our attack permits a dishonest bidder and a colluding auctioneer to break the fairness of the protocol. (Security against colluding auctioneers was not considered in [Sak00], so we do not disprove their results, but it is a property that one may expect the protocol to have.) Namely, a cheating bidder can place a bid $(1, M)$. If w is the highest honest bid, then the auctioneer can agree to open the corrupted bid to with x_{w+1}, thereby winning the auction for the cheating bidder at one dollar more than the second-highest bidder.

Sako came close to preventing this attack with an "incompatible encryption" property that avoids choosing $r = 0$ at encryption. A dishonest bidder however may deviate from this encryption rule; the problem is that the decryption algorithm does not reject ciphertexts (R, S) when $R = 1$. While such a ciphertext would surely look suspicious to a human observing the network traffic, it will most likely go unnoticed to the users if the software doesn't explicitly check for such ciphertexts. It is therefore up to the decryption algorithm to explicitly specify which cases need to be checked and up to the security proof to show that, if these cases are checked, the system indeed has the desired properties.

The attack above can easily be prevented by using any of our robust encryption schemes, so that decryption under any other secret key than the intended one results in $\perp$ being returned. Note that for this application we really need the strong robustness notion with adversarially generated ciphertexts.

Though necessary, our notion of strong robustness may not be sufficient to guarantee the fairness of the protocol in the case where a dishonest bidder has access the secret key held by the colluding auctioneer or when the public key of the scheme is not honestly generated, as our notion does not take these settings into account. Hence, to achieve fairness in Sako's auction protocol, it would be important to consider encryption schemes that achieve an even stronger notion of robustness in which public keys may be maliciously generated by the adversary [FLPQ13]. Interestingly, as pointed out in their paper, our strong robustness transform in Section 4 already achieves this stronger notion.

It is worth noting that, to enforce that all bids are independent of each other even in the presence of a colluding auctioneer, all bidders would also need to commit to their sealed bids (using a non-malleable commitment scheme) during a first round of communication and only open their commitments once all commitments made public.

9 Applications to searchable encryption

Public-key encryption with keyword search. A *public key encryption with keyword search* (PEKS) scheme [BDOP04] is a tuple $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ of algorithms. Via $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{KG}$, the key generation algorithm produces a pair of public and private keys. Via $C \stackrel{\$}{\leftarrow} \mathsf{PEKS}(pk, w)$, the encryption algorithm encrypts a keyword w to get a ciphertext under the public key pk. Via $t_w \stackrel{\$}{\leftarrow} \mathsf{Td}(sk, w)$, the trapdoor extraction algorithm computes a trapdoor t_w for keyword w. The deterministic test algorithm $\mathsf{Test}(t_w, C)$ returns 1 if C is an encryption of w and 0 otherwise.

Privacy and consistency of PEKS schemes. We formulate privacy notions for PEKS using the games of Figure 23. Let ATK $\in$ {CPA, CCA}. We define the advantage of an adversary A against the indistinguishability of $\mathcal{PEKS}$ as follows:

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{PEKS}}(A) = 2 \cdot \Pr\left[\,\text{IND-ATK}^{A}_{\mathcal{PEKS}} \Rightarrow \mathsf{true}\,\right] - 1\,.$$

We re-formulate the consistency definition of PEKS schemes of [ABC+08] using the game of Figure 23. We define the advantage of an adversary A against the consistency of $\mathcal{PEKS}$ as follows:

$$\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A) = \Pr\left[\,\text{CONSIST}^{A}_{\mathcal{PEKS}} \Rightarrow \mathsf{true}\,\right]\,.$$

Furthermore, we also recall the advantage measure $\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A)$, which captures the notion CONSIST of computational consistency of PEKS scheme $\mathcal{PEKS}$.

Transforming IBE to PEKS. The $\mathsf{bdop\text{-}ibe\text{-}2\text{-}peks}$ transform of [BDOP04] transforms an IBE scheme into a PEKS scheme. Given an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$, the transform associates to it the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$, where the key-generation algorithm KG returns $(pk, sk) \stackrel{\$}{\leftarrow} \mathsf{Setup}$; the encryption algorithm $\mathsf{PEKS}(pk, w)$ returns $C \leftarrow \mathsf{Enc}(pk, w, 0^k)$; the trapdoor extraction algorithm $\mathsf{Td}(sk, w)$ returns $t \stackrel{\$}{\leftarrow} \mathsf{Ext}(pk, sk, w)$; the test algorithm $\mathsf{Test}(t, C)$ returns 1 if and only if $\mathsf{Dec}(pk, t, C) = 0^k$. Abdalla et al. [ABC+08] showed that this transform generally does not provide consistency, and presented the consistency-providing $\mathsf{new\text{-}ibe\text{-}2\text{-}peks}$ transform as an alternative. We now show that the original $\mathsf{bdop\text{-}ibe\text{-}2\text{-}peks}$ transform does yield a consistent PEKS if the underlying IBE scheme is robust. We also show that if the base IBE scheme is ANO-CCA, then the PEKS scheme is IND-CCA, thereby yielding the first IND-CCA-secure PEKS schemes in the standard model, and the first consistent IND-CCA-secure PEKS schemes in the RO model. (Non-consistent IND-CCA-secure PEKS schemes in the RO model are easily derived from [FP07].)

proc Initialize $(pk, sk) \xleftarrow{\$} \mathsf{KG}$; $b \xleftarrow{\$} \{0,1\}$ $W \leftarrow \emptyset$; $C^* \leftarrow \perp$; Return pk **proc TD**(w) $\mathsf{TT}[w] \xleftarrow{\$} \mathsf{Td}(sk, w)$; $W \leftarrow W \cup \{w\}$; Return $\mathsf{TT}[w]$ **proc LR**(w_0^*, w_1^*) $C^* \xleftarrow{\$} \mathsf{PEKS}(pk, w_b^*)$; Return C^* **proc Test**(w, C) If $(C = C^*) \wedge (w \in \{w_0^*, w_1^*\})$ Then return $\perp$ If $\mathsf{TT}[w] = \perp$ Then $\mathsf{TT}[w] \xleftarrow{\$} \mathsf{Td}(sk, w)$ Return $\mathsf{Test}(\mathsf{TT}[w], C)$ **proc Finalize**(b') If $(\{w_0^*, w_1^*\} \cap W \neq \emptyset)$ Then return $(b = 0)$ Return $(b = b')$	**proc Initialize** $(pk, sk) \xleftarrow{\$} \mathsf{KG}(pars)$ Return pk **proc Finalize**(w, w') $C \xleftarrow{\$} \mathsf{PEKS}(pk, w)$ $t' \xleftarrow{\$} \mathsf{Td}(sk, w')$ Return $(w \neq w') \wedge (\mathsf{Test}(t', C))$

Figure 23: $\mathcal{PEKS} = (\mathsf{PG}, \mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ is a PEKS scheme. Games IND-CCA$_{\mathcal{PEKS}}$ and IND-CPA$_{\mathcal{PEKS}}$ are on the left, where the latter omits procedure **Test**. The **LR** procedure may be called only once. Game CONSIST$_{\mathcal{PEKS}}$ is on the right.

Proposition 9.1 *Let $\mathcal{IBE}$ be an IBE scheme, and let $\mathcal{PEKS}$ be the PEKS scheme associated to it per the* bdop-ibe-2-peks *transform. Given any adversary A running in time t, we can construct an adversary B running in time $t + O(t)$ executions of the algorithms of $\mathcal{IBE}$ such that*

$$\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A) \leq \mathbf{Adv}^{\text{wrob-cpa}}_{\mathcal{IBE}}(B) \qquad \textit{and} \qquad \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PEKS}}(A) \leq \mathbf{Adv}^{\text{ano-cca}}_{\mathcal{IBE}}(B) .$$

To see why the first inequality is true, it suffices to consider the adversary B that on input $pars$ runs $(w, w') \xleftarrow{\$} A(pars)$ and outputs these keywords along with the message 0^k. The proof of the second inequality is an easy adaptation of the proof of the new-ibe-2-peks transform in [ABC+08], where B answers A's **Test** queries using its own **Dec** oracle.

Securely combining PKE and PEKS. Searchable encryption by itself is only of limited use since it can only encrypt individual keywords, and since it does not allow decryption. Fuhr and Paillier [FP07] introduce a more flexible variant that allows decryption of the keyword. An even more powerful (and general) primitive can be obtained by combining PEKS with PKE to encrypt non-searchable but recoverable content. For example, one could encrypt the body of an email using a PKE scheme, and append a list of PEKS-encrypted keywords. The straightforward approach of concatenating ciphertexts works fine for CPA security, but is insufficient for a strong, combined IND-CCA security model where the adversary, in addition to the trapdoor oracle, has access to *both* a decryption oracle *and* a testing oracle. Earlier attempts to combine PKE and PEKS [BSS06, ZI07] do not give the adversary access to the latter. A full IND-CCA-secure PKE/PEKS scheme in the standard model can be obtained by combining the IND-CCA-secure PEKS schemes obtained through our transformation with the techniques of [DK05]. Namely, one can consider label-based [Sho01] variants of the PKE and PEKS primitives, tie the different components of a ciphertext together by using as a common label the verification key of a one-time signature scheme, and append to the ciphertext a signature of all components under the corresponding signing key. Though we omit the details, we note that the same techniques can be used to handle multiple encrypted keywords and avoid reordering attacks such as those mentioned by Boneh et al. [BDOP04].

Acknowledgments

We thank Chanathip Namprempre, who declined our invitation to be a co-author, for her participation and contributions in the early stage of this work.

References

[ABC+08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, July 2008. (Cited on pages 1, 4, 7, 36, and 37.)

[ABN08] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. Cryptology ePrint Archive, Report 2008/440, 2008. http://eprint.iacr.org/2008/440. (Cited on page 5.)

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, February 2010. (Cited on pages 4 and 5.)

[ABR98] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHIES: an encryption scheme based on the Diffie-Hellman problem. Contributions to IEEE P1363a, September 1998. (Cited on pages 26, 27, and 29.)

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, April 2001. (Cited on pages 4 and 26.)

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, May 2004. (Cited on page 13.)

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, December 2001. (Cited on pages 1, 3, 4, 7, and 19.)

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 52–64. Springer, February / March 2006. (Cited on pages 4, 5, and 34.)

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007. (Cited on pages 3, 4, and 32.)

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, August 1996. (Cited on page 27.)

[BD09] Mihir Bellare and Shanshan Duan. Partial signatures and their applications. Cryptology ePrint Archive, Report 2009/336, 2009. http://eprint.iacr.org/2009/336. (Cited on page 5.)

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, May 2004. (Cited on pages 1, 36, and 37.)

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, August 1998. (Cited on pages 1, 7, and 8.)

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662. Springer, April 2012. (Cited on page 5.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, August 2001. (Cited on pages 4 and 33.)

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. (Cited on pages 1 and 7.)

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th FOCS*, pages 647–657. IEEE Computer Society Press, October 2007. (Cited on page 7.)

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988. (Cited on pages 5 and 35.)

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, December 2004. (Cited on page 32.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, May / June 2006. (Cited on page 6.)

[BRS13] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 461–478. Springer, August 2013. (Cited on page 5.)

[BSS06] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart

Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 217–232. Springer, August / September 2006. (Cited on page 37.)

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 290–307. Springer, August 2006. (Cited on pages 4, 33, and 34.)

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535–554. Springer, February 2007. (Cited on pages 4 and 34.)

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, May 2004. (Cited on pages 3, 4, and 32.)

[CKVW10] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 52–71. Springer, February 2010. (Cited on page 5.)

[CS03] Ronald Cramer and Victor Shoup. 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 pages 3, 4, 19, 20, and 21.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Cited on pages 1 and 7.)

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 188–209. Springer, February 2005. (Cited on page 37.)

[EGM96] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996. (Cited on page 34.)

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 352–368. Springer, February / March 2013. (Cited on pages 5 and 36.)

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, August 1999. (Cited on pages 3, 4, 32, and 33.)

[FP07] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *LNCS*, pages 228–236, Wollongong, Australia, November 1–2, 2007. Springer. (Cited on pages 36 and 37.)

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 445–464. Springer, May / June 2006. (Cited on page 34.)

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on pages 1 and 7.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Cited on pages 5 and 35.)

[HM01] Martin Hirt and Ueli M. Maurer. Robustness for free in unconditional multi-party computation. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 101–118. Springer, August 2001. (Cited on pages 5 and 35.)

[HW08] Dennis Hofheinz and Enav Weinreb. Searchable encryption with decryption in the standard model. Cryptology ePrint Archive, Report 2008/423, 2008. `http://eprint.iacr.org/2008/423`. (Cited on pages 4 and 34.)

[Jak98] Markus Jakobsson. A practical mix. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 448–461. Springer, May / June 1998. (Cited on page 5.)

[KPSY09] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A new randomness extraction paradigm for hybrid encryption. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 590–609. Springer, April 2009. (Cited on page 3.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, April 2008. (Cited on pages 4 and 34.)

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979. (Cited on page 34.)

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 206–224. Springer, May 2012. (Cited on page 5.)

[Moh10] Payman Mohassel. A closer look at anonymity and robustness in encryption schemes. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 501–518. Springer, December 2010. (Cited on page 5.)

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, August 1992. (Cited on pages 1 and 7.)

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *LNCS*, pages 371–388. Springer, February 2004. (Cited on page 19.)

[Sak00] Kazue Sako. An auction protocol which hides bids of losers. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000*, volume 1751 of *LNCS*, pages 422–432. Springer, January 2000. (Cited on pages 2, 5, and 35.)

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/2001/112. (Cited on page 37.)

[ST13] Yannick Seurin and Joana Treger. A robust and plaintext-aware variant of signed ElGamal encryption. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 68–83. Springer, February / March 2013. (Cited on page 5.)

[TY98] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In Hideki Imai and Yuliang Zheng, editors, *PKC'98*, volume 1431 of *LNCS*, pages 117–134. Springer, February 1998. (Cited on page 5.)

[ZI07] Rui Zhang and Hideki Imai. Generic combination of public key encryption with keyword search and public key encryption. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *CANS 07*, volume 4856 of *LNCS*, pages 159–174. Springer, December 2007. (Cited on page 37.)