Combiners for Chosen-Ciphertext Security

Cong Zhang[1], David Cash[1], Xiuhua Wang[2], Xiaoqi Yu[3], and Sherman S.M. Chow[2](✉)

[1] Department of Computer Science, Rutgers University, New Brunswick, NJ, USA
[2] Department of Information Engineering, The Chinese University of Hong Kong, Sha Tin, N.T., Hong Kong
sherman@ie.cuhk.edu.hk
[3] Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong

Abstract. Security against adaptive chosen-ciphertext attack (CCA) is a *de facto* standard for encryption. While we know how to construct CCA-secure encryption, there could be pragmatic issues such as black-box design, software mis-implementation, and lack of security-oriented code review which may put the security in doubt. On the other hand, for double-layer encryption in which the two decryption keys are held by different parties, we expect the scheme remains secure even when one of them is compromised or became an adversary. It is thus desirable to combine two encryption schemes, where we cannot be assured that which one is really CCA-secure, to a new scheme that is CCA-secure. In this paper we propose new solutions to this problem for symmetric-key encryption and public-key encryption. One of our result can be seen as a new application of the detectable CCA notion recently proposed by Hohenberger *et al.* (Eurocrypt 2012).

Keywords: Encryption · Chosen-ciphertext security · Robust combiners

1 Introduction

Secure systems are usually complex and involve multiple components. If a component turns out to be problematic, the whole system may become totally insecure. For security-critical applications, a prudent practice is to have a robust design, such that the system remains secure even if a component is insecure. Of course, if one could identify which component is insecure, the designer can simply replace it with a secure one. Yet, it is notoriously difficult to ensure that a system component is secure in general. One example is that a component primitive is implemented as a black-box which the combiner cannot assert its security. On the other hand, even if the source code (of the software) or the circuit footprint (of the hardware) were available, asserting its security depends on the rigor and quality of the corresponding security-oriented review.

Sherman S.M. Chow is supported in part by the Early Career Award and the grants (CUHK 439713 & 14201914) of the Research Grants Council, Hong Kong.

In this paper, we look into a basic cryptographic tool which is encryption. We consider both public-key encryption (PKE) and symmetric-key encryption (SKE). The work of Herzberg [7] motivated the need of robust design of cryptosystem, which features a combination of multiple instantiation of the same primitives (*e.g.*, one may consider ElGamal encryption and RSA encryption as examples), such that as long as one of them ensures a certain level of security, the same security guarantee is preserved by the combined design, without knowing beforehand which one is that. *Robust combiner* achieving this property can ensure security even if there is doubt in the security of the component primitives [6]. It is also termed as tolerant cryptographic schemes [7] or cryptanalysis-tolerant schemes [5] in the literature.

Herzberg [7] proposed combiners that are secure against chosen-plaintext attack (CPA) or chosen-ciphertext attack. However, it is hard to achieve security against *adaptive* chosen-ciphertext attack (CCA in this paper[1]) if one of the component schemes turns out to be malleable. In the CCA attack, the adversary can query to a decryption oracle even after the adversary has obtained the challenge ciphertext, and the only disallowed query is the challenge ciphertext itself. Hence, if a part of the ciphertext is malleable, an adversary can simply maul it and obtain the plaintext from the decryption oracle. Dodis and Katz [5] proposed a cryptanalysis-tolerant CCA-secure encryption scheme, which remains secure when only an unknown one of the component schemes is CCA-secure.

Another usage of such a combiner is to achieve security for cryptosystems in which the decryption requires two private keys held by different parties. Security remains preserved when one of the parties is compromised by the adversary. An application is to support revocation via a security-mediator, a party whom needs to help the non-revoked users in every decryption request. Immediate revocation can be achieved once it is instructed to stop entertaining any further (partial) decryption request of the revoked user. For example, Chow *et al.* [2] proposed a CCA-secure security-mediated certificateless encryption scheme, combining an identity-based encryption with a public-key encryption generically. Without a combiner, a specific ad-hoc construction is probably needed [3].

Our Results. In this paper, we give two other cryptanalysis-tolerant CCA-secure encryption schemes, one for PKE and one for SKE. Our PKE combiner matches well with the notion of detectable chosen-ciphertext attack (DCCA) proposed by Hohenberger *et al.* [8] recently. Intuitively, DCCA is a weaker version of CCA, where "dangerous" ciphertexts are not allowed to be queried to the decryption oracle. Here, whether a ciphertext is dangerous can be checked by a polynomial-time function. Our combiner aims to achieve indistinguishability against DCCA attack, by detecting whether a query is originated from the challenge ciphertext of a component scheme. If so, such decryption query is disallowed. This gives a conceptually simple combiner with an elementary security proof. Furthermore, it illustrates yet another application of this DCCA notion.[2]

[1] We remark that it is called CCA2 in the literature when the adaptiveness matters.

[2] While the original paper has discussed the application of DCCA in ruling out some known implementation bug of a "sloppy" encryption scheme [8], our combiner does not assume the bug from the component scheme can be easily detected.

Yet, our combiner is downgrading the security of the component scheme since one of them is CCA-secure, but the resulting scheme is only DCCA-secure. For getting CCA-security, we resort back to the result of Hohenberger *et al.* [8]. Their work showed that we can construct a CCA-secure encryption scheme by a nested encryption approach, taking a DCCA-secure scheme, a CPA-secure scheme, and a 1-bounded CCA-secure scheme [4]. A q-bounded CCA-secure encryption system is secure against q chosen ciphertext queries, which can be constructed via a CPA-secure encryption primitive [4].

We then propose another combiner to directly obtain an SKE scheme with CCA security, by taking two SKE schemes in which only one of them is CCA-secure. This is different from our combiner for PKE. Note that an SKE scheme with security against chosen-plaintext attack and integrity of the ciphertext implies that this scheme is also CCA-secure [1]. For this combiner, our strategy is to work on these two properties instead, by taking two component schemes where an unknown one of them possesses of both properties.

Finally, we review in appendix the nested encryption technique of Hohenberger *et al.* [8] for obtaining CCA security.

2 Preliminaries

2.1 CCA Security for PKE

Definition 1 (Public-Key Encryption). *A public-key encryption scheme $\mathcal{PKE}$ consists of the following three probabilistic polynomial-time (PPT) algorithms* ($\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$)*.*

- $(EK, DK) \leftarrow \mathsf{KeyGen}(1^\lambda)$*: the algorithm outputs a pair of keys consisting of the public encryption key EK and the private decryption key DK, according to the input security parameter 1^λ.*
- $C \leftarrow \mathsf{Enc}(EK, m)$*: the algorithm takes a public key EK and a plaintext m as inputs, and outputs a ciphertext C.*
- $m \leftarrow \mathsf{Dec}(DK, C)$*: the algorithm uses the private key DK to decrypt a ciphertext C to recover the plaintext m, or to output $\perp$ denoting C is invalid.*

When the context is clear, we may put the input key as a subscript instead, or simply omit it.

We recall the definition of CCA security. Consider the following experiment $Exp^{\mathsf{cca}}_{\mathcal{A},\mathcal{PKE}}(1^\lambda)$ for $\mathcal{PKE}$:

- *Setup*: The challenger $\mathcal{C}$ takes a security parameter 1^λ and runs KeyGen to output keys (EK, DK). It gives $\mathcal{A}$ EK, and keeps DK to itself.
- *Query Phase 1*: $\mathcal{A}$ is given full access to the decryption oracle $\mathsf{Dec}(DK, \cdot)$. When the adversary $\mathcal{A}$ decides to terminate the query phase, it outputs a pair of messages m_0, m_1 of the same length.
- *Challenge*: The challenger $\mathcal{C}$ randomly picks a bit $b \leftarrow \{0, 1\}$, computes $C^* \leftarrow \mathsf{Enc}(EK, m_b)$ and sends C^* to $\mathcal{A}$.

– *Query Phase 2*: $\mathcal{A}$ continues to have access to $\mathsf{Dec}(DK, \cdot)$, but is not allowed to request for a decryption of C^*. Finally $\mathcal{A}$ outputs a bit b'.
– *Output*: The output of the experiment is defined to be 1 if $b' = b$, otherwise 0.

A PKE scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is CCA secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}()$ such that:

$$\Pr[Exp^{\mathsf{cca}}_{\mathcal{A},\mathcal{PKE}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

2.2 Detectable Chosen Ciphertext Security

Detectable chosen ciphertext attack (DCCA) is an attack mode against PKE introduced by Hohenberger *et al.* [8], which is weaker than the standard CCA notion. Considering a DCCA-secure PKE (or detectable encryption) suggests a new way to build CCA-secure encryption scheme. Their results show that one can construct a CCA-secure PKE scheme by applying nested encryption techniques on three primitives that are DCCA-secure, 1-bounded CCA-secure, and CPA-secure respectively.

A detectable encryption scheme is defined by $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, F)$, where KeyGen, Enc, and Dec behave as those in traditional encryption schemes, but with an additional efficient boolean function $F()$ available, which is designed to detect "dangerous" ciphertext. Specifically, $F()$ will be applied before any decryption query in Phase 2 of the original CCA game. When the queried ciphertext C "is related to" the challenge ciphertext C^*, meaning that adversary can infer "useful" information about C^* from the decryption query of C, $F()$ will return 1 and the query is rejected; else the decryption result of C will be returned to the adversary. Definition 2 formally describes the syntax of a detectable encryption scheme.

Definition 2 (Detectable Encryption). *A detectable encryption scheme consists of the following PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, F)$*.*

– $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ *are defined as those in a regular PKE scheme.*
– $\{0, 1\} \leftarrow F(EK, C, C^*)$*: The detecting function* F *takes as inputs a public key* EK *and two ciphertexts* C *and* C^**, and outputs* 1 *if* C *and* C^* *has some relations, else outputs* 0*.*

The definition of $F()$ above is at its full generality. We may omit the input of EK from $F()$ when the function $F()$ does not need it.

Correctness is defined as in a regular encryption scheme. A DCCA-secure scheme must satisfy unpredictability for F and indistinguishability under DCCA.

Unpredictability of the Detecting Function [8]. Intuitively, it is hard for the adversary to find a useful ciphertext C, given the detectable function $F()$ and a public key EK. This is formally defined via the game $Exp^{\mathsf{unp}}_{\mathcal{A},\Pi}(1^\lambda)$ for a detectable scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, F)$ played by an adversary $\mathcal{A}$.

– *Setup*: The challenger $\mathcal{C}$ takes a security parameter 1^λ and runs KeyGen to output keys (EK, DK). It gives EK to $\mathcal{A}$, and keeps DK to itself.
– *Query*: $\mathcal{A}$ can fully access the decryption oracle $\mathsf{Dec}(DK, \cdot)$. When $\mathcal{A}$ concludes the query phase, it outputs a message m and a ciphertext C.
– *Challenge*: The challenger $\mathcal{C}$ outputs a ciphertext $C^* \leftarrow \mathsf{Enc}(EK, m)$.
– *Output*: The experiment outputs $F(EK, C, C^*)$.

A detectable encryption scheme Π is said to have unpredictability for F if, for any PPT adversary $\mathcal{A}$, we have $\Pr[Exp^{\mathsf{unp}}_{\mathcal{A},\Pi} = 1] \leq \mathsf{negl}(\lambda)$.

One can formulate a stronger version of the above game, in which the adversary is given the decryption key instead of the oracle [8]. This implies the basic version of undetectability since the adversary can simulate the decryption oracle when given DK.

Indistinguishability under DCCA [8]. Now we formalize the confidentiality guarantee according to the following experiment $Exp^{\mathsf{dcca}}_{\mathcal{A},\Pi}(1^\lambda)$:

– *Setup*: The challenger $\mathcal{C}$ takes a security parameter 1^λ and runs KeyGen to output keys (EK, DK). It gives $\mathcal{A}$ EK, and keeps DK to itself.
– *Query Phase 1*: $\mathcal{A}$ is given full access to the decryption oracle $\mathsf{Dec}(DK, \cdot)$. When the adversary $\mathcal{A}$ decides that the query phase ends, it outputs messages m_0, m_1 of the same length.
– *Challenge*: The challenger $\mathcal{C}$ randomly picks a bit $b \leftarrow \{0, 1\}$, computes $C^* \leftarrow \mathsf{Enc}(EK, m_b)$ and sends C^* to $\mathcal{A}$.
– *Query Phase 2*: $\mathcal{A}$ continues to have access to $\mathsf{Dec}(DK, \cdot)$, but is not allowed to issue a decryption query such that $F(EK, C, C^*) = 1$.
– *Output*: $\mathcal{A}$ wins the game and the experiment outputs 1 if and only if $b' = b$.

A detectable encryption scheme Π is said to have indistinguishability under DCCA, if we have $\Pr[Exp^{\mathsf{dcca}}_{\mathcal{A},\Pi} = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$.

2.3 Authenticated (Symmetric-Key) Encryption

Definition 3 (Symmetric-Key Encryption). *A symmetric-key encryption scheme $\mathcal{SKE}$ consists of the following three probabilistic polynomial-time (PPT) algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

– $SK \leftarrow \mathsf{KeyGen}(1^\lambda)$*: the algorithm outputs a secret key SK according to the input security parameter 1^λ.*
– $C \leftarrow \mathsf{Enc}(SK, m)$*: the algorithm takes a secret key SK and a plaintext m as inputs, and outputs a ciphertext C.*
– $m \leftarrow \mathsf{Dec}(SK, C)$*: the algorithm decrypts a ciphertext C to the corresponding plaintext m, or outputs $\perp$, by using the secret key SK.*

Confidentiality. We recall the definition of CPA security and CCA security. Consider the following experiment, $Exp^{\mathsf{atk}}_{\mathcal{A},\mathcal{SKE}}(1^\lambda)$ for $\mathcal{SKE}$:

- *Setup*: The challenger $\mathcal{C}$ runs $\mathsf{KeyGen}(1^\lambda)$ and obtains the secret key SK.
- *Query Phase 1*: $\mathcal{A}$ is given full access to the encryption oracle $\mathsf{Enc}(SK,\cdot)$ for $\mathsf{atk}=\mathsf{cpa}$, and an additional decryption oracle $\mathsf{Dec}(SK,\cdot)$ for $\mathsf{atk}=\mathsf{cca}$. When the adversary $\mathcal{A}$ decides to terminate the query phase, it outputs a pair of messages m_0, m_1 of the same length.
- *Challenge*: The challenger $\mathcal{C}$ randomly picks a bit $b \leftarrow \{0,1\}$, computes $C^* \leftarrow \mathsf{Enc}(SK, m_b)$ and sends C^* to $\mathcal{A}$.
- *Query Phase 2*: $\mathcal{A}$ continues to have access to $\mathsf{Enc}(SK,\cdot)$ for $\mathsf{atk}=\mathsf{cpa}$. For $\mathsf{atk}=\mathsf{cca}$, the adversary also has access to the decryption oracle, but is not allowed to request for a decryption of C^*. Finally $\mathcal{A}$ outputs a bit b'.
- *Output*: The output of the experiment is defined to be 1 if $b'=b$, otherwise 0.

An SKE scheme $\mathcal{SKE}=(\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ is CPA/CCA-secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}()$ such that:

$$\Pr[Exp^{\mathsf{atk}}_{\mathcal{A},\mathcal{SKE}}(1^\lambda)=1] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

for $\mathsf{atk}=\mathsf{cpa}$ or $\mathsf{atk}=\mathsf{cca}$ respectively.

Integrity. Integrity of ciphertexts (INT) or *integrity of plaintexts* (INT-PTXT) is formally defined via the following experiment, $Exp^{\mathsf{atk}}_{\mathcal{A},\mathcal{SKE}}(1^\lambda)$, played by an adversary $\mathcal{A}$ for an SKE scheme $\mathcal{SKE}$, where $\mathsf{atk}=\mathsf{int}$ or $\mathsf{atk}=\mathsf{int\text{-}ptxt}$ respectively.

- *Setup*: The challenger $\mathcal{C}$ runs $\mathsf{KeyGen}(1^\lambda)$ and obtains the secret key SK.
- *Query*: $\mathcal{A}$ is given full access to the encryption oracle $\mathsf{Enc}(SK,\cdot)$ and the decryption oracle $\mathsf{Dec}(SK,\cdot)$.
- *Challenge*: When the adversary $\mathcal{A}$ decides to terminate the query phase, it outputs a forgery C^*.
- *Output*: The challenger $\mathcal{C}$ decrypts C^* to obtain M^*.
  - For $\mathsf{atk}=\mathsf{int}$, if $M^* \neq \perp$ and C^* has never appeared as a response by the challenger to any encryption oracle query of $\mathcal{A}$, $\mathcal{A}$ is considered to have won the game, and the experiment outputs 1; otherwise, outputs 0.
  - For $\mathsf{atk}=\mathsf{int\text{-}ptxt}$, if $M^* \neq \perp$ and M^* has never appeared in any encryption oracle query, $\mathcal{A}$ is considered to have won the game, and the experiment outputs 1; otherwise, outputs 0.

An SKE scheme $\mathcal{SKE}=(\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ is INT-secure/INT-PTXT-secure if for all PPT adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}()$ such that:

$$\Pr[Exp^{\mathsf{atk}}_{\mathcal{A},\mathcal{SKE}}(1^\lambda)=1] \le \mathsf{negl}(\lambda)$$

for $\mathsf{atk}=\mathsf{int}$ or $\mathsf{atk}=\mathsf{int\text{-}ptxt}$ respectively.

An INT-secure scheme is also INT-PTXT-secure [1].

3 Combiner from CCA Security to DCCA Security

Given two public-key encryption schemes, $\mathcal{PKE}_1$ and $\mathcal{PKE}_2$ where $\mathcal{PKE}_1 = (\mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$, $\mathcal{PKE}_2 = (\mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$ such that only one of them is CCA-secure, we can build a detectable public-key encryption scheme $\mathcal{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, F)$ that achieves DCCA security.

– $(EK, DK) \leftarrow \mathsf{KeyGen}(1^\lambda)$: $\mathsf{KeyGen}(1^\lambda)$ executes the key generation algorithm of $\mathcal{PKE}_1$ and $\mathcal{PKE}_2$: $(EK_1, DK_1) \leftarrow \mathcal{PKE}_1.\mathsf{KeyGen}_1(1^\lambda)$, $(EK_2, DK_2) \leftarrow \mathcal{PKE}_2.\mathsf{KeyGen}_2(1^\lambda)$; and outputs $EK = (EK_1, EK_2)$ and $DK = (DK_1, DK_2)$.
– $C \leftarrow \mathsf{Enc}(m)$: this algorithm chooses a random string r which is as long as the message m, and sets $C = (C_1, C_2) = (\mathcal{PKE}_1.\mathsf{Enc}_1(r), \mathcal{PKE}_2.\mathsf{Enc}_2(r \oplus m))$.
– $m \leftarrow \mathsf{Dec}(DK, C)$: this algorithm returns $m = \mathsf{Dec}_{DK_1}(C_1) \oplus \mathsf{Dec}_{DK_2}(C_2)$.
– $\{0,1\} \leftarrow F(EK, C, C^*)$: Let $C^* = (C_1^*, C_2^*)$ be the challenge ciphertext. Similarly, parse C as (C_1, C_2). We define the detecting function $F(EK, C, C^*)$ to output 1 if and only if:

$$C_1 = C_1^* \quad \text{or} \quad C_2 = C_2^*;$$

otherwise, *i.e.*, $C_1 \neq C_1^*$ and $C_2 \neq C_2^*$, it outputs 0.

In the following, we will show that this construction achieves DCCA security.

Lemma 4. *The detecting function F satisfies unpredictability.*

Proof. Since both $\mathcal{PKE}_1, \mathcal{PKE}_2$ are probabilistic schemes, without receiving the challenge ciphertext and with no decryption key, no adversary can output a ciphertext C such that $F(EK, C, C^*) = 1$ with non-negligible probability. Thus the unpredictability of the detecting function F for the combiner scheme is satisfied. Next we prove indistinguishability of encryptions, which will then complete the proof of DCCA security. □

Lemma 5. *If $\mathcal{PKE}_1$ is CCA-secure, then $\mathcal{PKE}$ is DCCA-secure.*

Proof. Since $\mathcal{PKE}$ satisfies unpredictability of the detecting function F, it suffices to show that $\mathcal{PKE}$ is indistinguishable. If there is an adversary $\mathcal{A}$ which can break the indistinguishability experiment of $\mathcal{PKE}$ with non-negligible probability ϵ, then we can construct a simulator $\mathcal{B}$ to break the CCA experiment of $\mathcal{PKE}_1$ with probability ϵ.

Given EK_1 of $\mathcal{PKE}_1$, $\mathcal{B}$ calls $\mathsf{KeyGen}_2()$ of $\mathcal{PKE}_2$, and sends $EK = (EK_1, EK_2)$ to $\mathcal{A}$. $\mathcal{B}$ can simulate the decryption oracle in Phase 1, by using that of $\mathcal{PKE}_1$ and the private decryption key DK_2.

During the challenge phase, $\mathcal{A}$ submits m_0, m_1 (of the same length) to $\mathcal{B}$. $\mathcal{B}$ chooses a randomness r_0, calculates $r_1 = r_0 \oplus m_0 \oplus m_1$, and sends r_0, r_1 to the challenger $\mathcal{C}$. Then the challenger $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$ and sends $\mathsf{Enc}_1(r_b)$ to $\mathcal{B}$. Receiving $\mathsf{Enc}_1(r_b)$, $\mathcal{B}$ computes $\mathsf{Enc}_2(r_0 \oplus m_0)$ and sends the challenge ciphertext $C^* = (C_1^*, C_2^*)$, where $C_1^* = \mathsf{Enc}_1(r_b)$, $C_2^* = \mathsf{Enc}_2(r_0 \oplus m_0)$ to $\mathcal{A}$.

If $b = 0$, this challenge ciphertext is correctly distributed for m_0. If $b = 1$, we view r_b as $r_1 = r_0 \oplus m_0 \oplus m_1$, then r_1 is randomly distributed, and $r_0 \oplus m_0 = r_1 \oplus m_1$, so the challenge ciphertext is also correctly distributed for m_1.

In Phase 2, the adversary is only allowed to submit C to the decryption oracle if $F(EK, C, C^*) = 0$. Parsing C as (C_1, C_2) and C^* as (C_1^*, C_2^*). Under this condition, we have $C_1 \neq C_1^*$. $\mathcal{B}$ can submit C_1 to the decryption oracle of its own challenger. Thus, $\mathcal{B}$ can also properly simulate the decryption oracle for $\mathcal{A}$ in Phase 2.

Finally, $\mathcal{A}$ outputs its guess b'. $\mathcal{B}$ simply forwards b' as its own guess. When $\mathcal{A}$ guesses it is $b' = 0$, it means $C_1^* = \mathsf{Enc}_1(r_0)$. When $\mathcal{A}$ guesses it is $b' = 1$, it means $C_1^* = \mathsf{Enc}_1(r_1)$. The messages just match with those chosen by $\mathcal{B}$ in its own game. Hence, the probability of $\mathcal{B}$ to win its game is also ϵ. $\square$

Since our encryption scheme does not look symmetric regarding the choice of $\mathcal{PKE}_1, \mathcal{PKE}_2$ (versus the ordering of $\mathcal{PKE}_2$ then $\mathcal{PKE}_1$), we will now consider the case that $\mathcal{PKE}_2$ is CCA-secure.

Lemma 6. *If $\mathcal{PKE}_2$ is CCA-secure, then $\mathcal{PKE}$ is DCCA-secure.*

Proof. In this case, the challenger $\mathcal{C}$ is for $\mathcal{PKE}_2$. Our goal is to construct a simulator $\mathcal{B}$ to win the CCA experiment of $\mathcal{PKE}_2$ with the help of adversary $\mathcal{A}$.

Given EK_2 of $\mathcal{PKE}_2$, $\mathcal{B}$ calls $\mathsf{KeyGen}_1()$ of $\mathcal{PKE}_1$, and sends $EK = (EK_1, EK_2)$ to $\mathcal{A}$. Simulation of the decryption oracle in Phase 1 is similar to the corresponding treatment in the previous proof.

In the challenge phase, $\mathcal{A}$ submits m_0, m_1 (of the same length) to $\mathcal{B}$. $\mathcal{B}$ choose a randomness r, calculates $r_0 = r \oplus m_0$, $r_1 = r \oplus m_1$, and sends r_0, r_1 to the challenger $\mathcal{C}$. Then the challenger $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0, 1\}$ and sends $\mathsf{Enc}_2(r_b)$ to $\mathcal{B}$. Receiving $\mathsf{Enc}_2(r_b)$, $\mathcal{B}$ computes $\mathsf{Enc}_1(r)$ and sends the challenge ciphertext $C^* = (C_1^*, C_2^*)$, where $C_1^* = \mathsf{Enc}_1(r)$, $C_2^* = \mathsf{Enc}_2(r_b)$ to the adversary $\mathcal{A}$. The challenge ciphertext is correctly distributed since $r_b = r \oplus m_b$ for $b \in \{0, 1\}$.

In Phase 2, the adversary is only allowed to query the decryption for C where $F(EK, C, C^*) = 0$. Parsing C as (C_1, C_2) and C^* as (C_1^*, C_2^*), we thus have $C_2 \neq C_2^*$. $\mathcal{B}$ can then submit C_2 to the decryption oracle in its own game.

Finally, $\mathcal{B}$ outputs what $\mathcal{A}$ outputs. Similar to our analysis of the distribution of the challenge ciphertext, the guess of $\mathcal{A}$ just matches with the guess of $\mathcal{B}$. Therefore $\mathcal{B}$ can win with probability ϵ. $\square$

Combining Lemmas 4, 5 and 6, we can see that $\mathcal{PKE}$ is DCCA-secure if one of $\mathcal{PKE}_1, \mathcal{PKE}_2$ is CCA-secure. We can then construct a CCA-secure scheme by using the nested encryption technique [8], which we review in Appendix.

4 Combiner for Secret Key Encryption

Similar to our analysis above, we can get a combiner for SKE if the nested encryption scheme of Hohenberger *et al.* [8] also applied on SKE. However, apart from the two component schemes, it also requires a 1-bounded CCA-secure scheme and another CPA-secure scheme. This appears to be complicated for an SKE scheme. We thus give a direct CCA-combiner for SKE below.

One of the advantages that SKE possesses over PKE is that the encryption needs the secret key, so that an adversary might not easily obtain by itself a ciphertext which decrypts to a valid message. This property is called integrity (INT). An SKE which is both CPA-secure and INT-secure is called authenticated encryption, and is CCA-secure [1]. Below we aim to propose a combiner for authenticated encryption, which achieves both CPA and INT security.

We require the ciphertext produced by $\mathsf{Enc}(SK, m; r)$ of the $\mathcal{SKE}$ schemes, is in the form of $C = (r, \mathsf{E}_{SK}(m, r))$ where r is the randomness used in the implicitly-defined deterministic key-ed function $\mathsf{E}_{SK}(m, r)$. We also assume that they are perfectly correct, *i.e.*, $\Pr[\mathsf{Dec}(SK, \mathsf{Enc}(SK, m)) = m] = 1$, $\forall SK \leftarrow \mathsf{KeyGen}(1^\lambda)$.

Given two encryption schemes $\mathcal{SKE}_1, \mathcal{SKE}_2$, one of which is CPA+INT-secure, and both have the stated ciphertext form and perfect correctness, our combiner for CPA+INT-secure SKE $\mathcal{SKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows.

- $\mathsf{KeyGen}(1^\lambda)$: takes the security parameter 1^λ, sets the $SK = (SK_1, SK_2)$, where SK_i is the secret key generated by $\mathcal{SKE}_i.\mathsf{KeyGen}_i(1^\lambda)$ for $i \in \{1, 2\}$.
- $\mathsf{Enc}(SK, m)$: chooses randomness R_1, R_2 such that $R_1 \oplus R_2 = m$ and sets

$$\begin{aligned} C_1 &= (r_1, \mathsf{E}_1(R_1, r_1)) \\ C_2 &= (r_2, \mathsf{E}_2(R_2, r_2)) \\ C_3 &= (r_3, \mathsf{E}_1(C_1||C_2||r_4, r_3)) \\ C_4 &= (r_4, \mathsf{E}_2(C_2||C_1||r_3, r_4)) \end{aligned}$$

  where r_1, r_2, r_3, r_4 are the randomness used in the corresponding encryption algorithm. Outputs $C = (C_1, C_2, C_3, C_4)$.
- $\mathsf{Dec}(SK, C)$: firstly parses C into (C_1, C_2, C_3, C_4), checks if both of $C_3 = (r_3, \mathsf{E}_1(C_1||C_2||r_4, r_3))$ and $C_4 = (r_4, \mathsf{E}_2(C_2||C_1||r_3, r_4))$ hold. If no, outputs $\perp$. Otherwise, gets $R'_1 = \mathsf{Dec}(SK_1, C_1)$ and $R'_2 = \mathsf{Dec}(SK_2, C_2)$. If none of them is $\perp$, returns $R'_1 \oplus R'_2$.

The two lemmas below assert the security of this combiner scheme $\mathcal{SKE}$.

Lemma 7. *$\mathcal{SKE}$ is CPA-secure.*

Proof. Since $\mathcal{SKE}$ is symmetric for $\mathcal{SKE}_1$ and $\mathcal{SKE}_2$, without loss of the generality, we suppose $\mathcal{SKE}_1$ is CPA+INT-secure and $\mathcal{SKE}_2$ is an arbitrary encryption scheme.

If $\mathcal{SKE}$ is not CPA-secure, we can construct a simulator $\mathcal{B}$ to win the CPA game of $\mathcal{SKE}_1$. Given the encryption oracle of $\mathcal{SKE}_1$, $\mathcal{B}$ generates the parameters of $\mathcal{SKE}_2$ and runs $\mathsf{Enc}_2()$ by itself. In Phase 1, when $\mathcal{A}$ issues an encryption query for message m, $\mathcal{B}$ randomly picks R_i calls the $\mathsf{Enc}_1()$ oracle to get $C_1 = (r_1, \mathsf{E}_1(m \oplus R_i, r_1))$, then $\mathcal{B}$ randomly chooses r_2 and r_4, computes $C_2 = (r_2, \mathsf{E}_2(R_i, r_2))$ and sends $C_1||C_2||r_4$ to the encryption oracle. Receiving

$C_3 = (r_3, \mathsf{E}_1(C_1||C_2||r_4, r_3))$, the simulator $\mathcal{B}$ also calculates C_4 with r_3 and r_4. Note that the messages encrypted in C_1 and C_2 follows the distribution of the real scheme, and C_3 and C_4 simply follow the construction as in the real scheme.

In the challenge phase, the adversary $\mathcal{A}$ submits m_0, m_1 to $\mathcal{B}$, and $\mathcal{B}$ randomly picks R and sends $m'_0 = m_0 \oplus R$ and $m'_1 = m_1 \oplus R$ to the challenger $\mathcal{C}$. Then the challenger $\mathcal{C}$ flips a coin $b \in \{0,1\}$, encrypts m'_b to obtain $C'_b = (r_1, \mathsf{E}_1(m'_b, r_1))$, and sends it to $\mathcal{B}$. $\mathcal{B}$ then randomly picks r_2, r_4, and computes $C^*_2 = (r_2, \mathsf{E}_2(R, r_2))$ and submits $C'_b||C^*_2||r_4$ to the encryption oracle. After receiving $C^*_3 = (r_3, \mathsf{E}_1(C'_b||C^*_2||r_4))$, the simulator calculates $C^*_4 = (r_4, \mathsf{E}_2(C'_b||C^*_2||r_3))$ and sends $(C'_b, C^*_2, C^*_3, C^*_4)$ as the challenge ciphertext for $\mathcal{A}$. It is a well-distributed challenge ciphertext, following the same analysis as that for the simulation of encryption oracle by $\mathcal{B}$.

In Phase 2, the simulator acts exactly as in Phase 1, and outputs what the adversary $\mathcal{A}$ outputs. We can see that the guess of $\mathcal{A}$ is correct if and only that of $\mathcal{B}$ is correct. Thus if $\mathcal{A}$ can guess correctly with non-negligible advantage over $\frac{1}{2}$, so does $\mathcal{B}$ in breaking the CPA security of $\mathcal{SKE}_1$. □

Lemma 8. *$\mathcal{SKE}$ is INT-secure.*

Proof. Without loss of the generality, we suppose that $\mathcal{SKE}_1$ is CPA+INT-secure. If $\mathcal{SKE}$ is not INT-secure, we construct a simulator $\mathcal{B}$ that breaks the INT-security of $\mathcal{SKE}_1$.

$\mathcal{B}$ is given the encryption oracle $\mathsf{Enc}_1()$ of $\mathcal{SKE}_1$, and runs $\mathsf{Enc}_2()$ normally by itself. In Phase 1, when the adversary $\mathcal{A}$ queries for an encryption of message m, $\mathcal{B}$ randomly picks an R and calls the $\mathsf{Enc}_1()$ oracle to gets $C'_1 = (r'_1, \mathsf{E}_1(R, r'_1))$, then $\mathcal{B}$ randomly picks r'_2 and r'_4, computes $C'_2 = (r'_2, \mathsf{E}_2(M \oplus R, r'_2))$ and sends $C'_1||C'_2||r'_4$ to $\mathsf{Enc}_1()$ oracle. Receiving $C'_3 = (r'_3, \mathsf{E}_1(C'_1||C'_2||r'_4, r'_3))$, the simulator $\mathcal{B}$ also calculates C'_4 with r'_3 and r'_4. Finally $\mathcal{B}$ returns $C' = (C'_1, C'_2, C'_3, C'_4)$ as the response. Note that C'_1 and C'_3 obtained by $\mathcal{B}$ from its own encryption oracle $\mathsf{Enc}_1()$ are always directly forwarded to $\mathcal{A}$ as is.

In the challenge phase, the adversary $\mathcal{A}$ returns the forgery C^*. $\mathcal{B}$ parses $C^* = (C^*_1, C^*_2, C^*_3, C^*_4)$. If C^*_1 or C^*_3 has not been forwarded by $\mathcal{B}$ to $\mathcal{A}$ before, $\mathcal{B}$ returns it to break the INT security of $\mathcal{SKE}_1$.

Consider to the contrary that both C^*_1 and C^*_3 directly came from the response of the encryption oracle due to some queries of $\mathcal{B}$. Note that for any valid forgery, $C^*_3 = (r^*_3, \mathsf{E}_1(C^*_1||C^*_2||r^*_4, r^*_3))$, which is ensured by the validity checking of the decryption algorithm. From the perfect correctness of $\mathcal{SKE}_1$, every valid ciphertext can only be decrypted to a single message. Since C^*_3 was created from $\mathsf{Enc}_1()$, the corresponding query supplied to $\mathsf{Enc}_1()$, and hence the decryption result of C^*_3, must be $C^*_1||C^*_2||r^*_4$, Also note that every $\mathsf{Enc}_1()$ oracle query $\mathcal{B}$ has ever made is for returning a ciphertext $C' = (C'_1, C'_2, C'_3, C'_4)$ to $\mathcal{A}$. When C^*_3 was returned by $\mathsf{Enc}_1()$, it must be triggered by an encryption oracle query by $\mathcal{A}$ which leads to the creation of $C' = (C^*_1, C^*_2, C^*_3, C'_4)$, where $C'_4 = (r^*_4, \mathsf{E}_2(C^*_2||C^*_1||r^*_3, r^*_4)) = C^*_4$. So the forgery $C^* = (C^*_1, C^*_2, C^*_3, C^*_4)$ is exactly the same as C' given by $\mathcal{B}$ to $\mathcal{A}$ before, violating the rule of the game played by $\mathcal{A}$. Contradiction occurs and this concludes the proof. □

5 Conclusion

We show two provably-secure robust combiners for ensuring security against chosen-ciphertext attack (CCA). Our robust combiner for public-key encryption (PKE) is inspired by the detectable chose-ciphertext attack (DCCA) notion proposed by Hohenberger *et al.* [8]. Instead of directly obtaining a combiner to get a CCA-secure PKE from two possibly CCA-secure PKE schemes, our goal is to devise a combiner for DCCA security, given that only one of the schemes is CCA-secure. A CCA-secure scheme can thus be obtained following the nested encryption approach proposed by Hohenberger *et al.* [8]. For our robust combiner for symmetric-key encryption (SKE), instead of directly working on CCA-security, we work on the CPA-security and the ciphertext integrity, in which a combination of both implies CCA-security of an SKE scheme [1].

It is our future work to build a more efficient SKE combiner.

A CCA Security from DCCA Security

CCA-secure PKE can be obtained by combining DCCA PKE, 1-bounded CCA PKE, and CPA PKE [8]. We remark that the same technique also works in identity-based encryption (IBE), attribute-based encryption (ABE), and threshold PKE/IBE. The same holds true for our combiner in Sect. 3.

We use Π_{DCCA}, Π_{CPA}, and Π_{qb} to denote the encryption primitives which are DCCA-secure, CPA-secure, and q-bounded-CCA-secure (where $q = 1$) respectively. For a probabilistic algorithm $\mathsf{Enc}(\cdot)$, we can transform it to a deterministic one $\mathsf{Enc}(\cdot; r)$ where r is a well-distributed random value.

We describe the CCA-secure encryption scheme in the context of IBE. It can easily degenerated to SKE/PKE, or extended into threshold PKE/IBE or ABE.

A.1 Syntax of IBE

In IBE, any user can request for a secret key SK_{ID} related to her identity ID from a trusted private key generator. The secret key SK_{ID} can decrypt the ciphertext encrypted for ID correctly. An IBE scheme is defined as follows.

- $(MPK, MSK) \leftarrow \mathsf{Setup}(1^\lambda)$: This algorithm takes as the security parameter 1^λ and returns a master public key MPK and a master secret key MSK. MPK is omitted from the input of the rest of the algorithms.
- $SK_{ID} \leftarrow \mathsf{Extract}(MSK, ID)$: This algorithm takes as inputs the master security key MSK and an user identity ID, and it returns a user secret key SK_{ID}.
- $C \leftarrow \mathsf{Enc}(ID, m)$: This algorithm takes as inputs a user identity ID, and a message m, it then returns a ciphertext C encrypting m for ID.
- $m \leftarrow \mathsf{Dec}(ID, SK_{ID}, C)$: It takes as inputs a secret key SK_{ID} corresponding to the identity ID, and a ciphertext C. It returns m or an invalid symbol $\perp$.

A.2 CCA-Secure Construction

- $(MPK, MSK) \leftarrow \mathsf{Setup}(1^\lambda)$: Run all the underlying IBE setup algorithms: $\mathsf{Setup}_{\mathrm{DCCA}}(1^\lambda)$ to get $(MPK_{\mathrm{DCCA}}, MSK_{\mathrm{DCCA}})$, then $\mathsf{Setup}_{\mathrm{CPA}}(1^\lambda)$ to obtain $(MPK_{\mathrm{CPA}}, MSK_{\mathrm{CPA}})$ and $\mathsf{Setup}_{\mathrm{qb}}(1^\lambda)$ to get $(MPK_{\mathrm{qb}}, SK_{\mathrm{qb}})$. Keep $MSK = (MSK_{\mathrm{DCCA}}, MSK_{\mathrm{CPA}}, MSK_{\mathrm{qb}})$ in secret and output the master public key as $MPK = (MPK_{\mathrm{DCCA}}, MPK_{\mathrm{CPA}}, MPK_{\mathrm{qb}})$.
- $SK_{ID} \leftarrow \mathsf{Extract}(ID)$: Run $\mathsf{Extract}_{\mathrm{DCCA}}(ID)$ to obtain $SK_{\mathrm{DCCA}.ID}$, then $\mathsf{Extract}_{\mathrm{CPA}}(ID)$ to obtain $SK_{\mathrm{CPA}.ID}$, and $\mathsf{Extract}_{\mathrm{qb}}(ID)$ to obtain $SK_{\mathrm{qb}.ID}$. Finally, output $SK_{ID} = (SK_{\mathrm{DCCA}.ID}, SK_{\mathrm{CPA}.ID}, SK_{\mathrm{qb}.ID})$.
- $C \leftarrow \mathsf{Enc}(ID, m)$: First pick three random values $r_{\mathrm{DCCA}}, r_{\mathrm{CPA}}, r_{\mathrm{qb}} \in \{0,1\}^\lambda$, encrypt two of them with the message m in C_{DCCA} using r_{DCCA} as the encryption randomness, *i.e.*, $\mathsf{Enc}_{\mathrm{DCCA}}(ID, (r_{\mathrm{CPA}}||r_{\mathrm{qb}}||m); r_{\mathrm{DCCA}})$; then compute two more encryption of it via $C_{\mathrm{qb}} = \mathsf{Enc}_{\mathrm{qb}}(ID, C_{\mathrm{DCCA}}; r_{\mathrm{qb}})$ and $C_{\mathrm{CPA}} = \mathsf{Enc}_{\mathrm{CPA}}(ID, C_{\mathrm{DCCA}}; r_{\mathrm{CPA}})$. Finally, we set $C = (C_{\mathrm{CPA}}, C_{\mathrm{qb}})$.
- $m \leftarrow \mathsf{Dec}(ID, SK_{ID}, C)$: Parse C into $(C_{\mathrm{CPA}}, C_{\mathrm{qb}})$. Decrypt the second ciphertext $\mathsf{Dec}_{\mathrm{qb}}(ID, SK_{ID}, C_{\mathrm{qb}})$ to obtain C_{DCCA}. Then decrypt it to obtain $(r_{\mathrm{CPA}}||r_{\mathrm{qb}}||m)$. Check that both $C_{\mathrm{qb}} = \mathsf{Enc}_{\mathrm{qb}}(ID, C_{\mathrm{DCCA}}; r_{\mathrm{qb}})$ and $C_{\mathrm{CPA}} = \mathsf{Enc}_{\mathrm{CPA}}(ID, C_{\mathrm{DCCA}}; r_{\mathrm{CPA}})$ holds. If so, output m; otherwise output $\perp$.

References

1. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000). doi:10.1007/3-540-44448-3_41
2. Chow, S.S.M., Boyd, C., Nieto, J.M.G.: Security-mediated certificateless cryptography. In: Public Key Cryptography (PKC), pp. 508–524 (2006). http://dx.doi.org/10.1007/11745853_33
3. Chow, S.S.M., Roth, V., Rieffel, E.G.: General certificateless encryption and timed-release encryption. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 126–143. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85855-3_9
4. Cramer, R., Hanaoka, G., Hofheinz, D., Imai, H., Kiltz, E., Pass, R., Shelat, A., Vaikuntanathan, V.: Bounded CCA2-secure encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 502–518. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76900-2_31
5. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30576-7_11
6. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (2005). doi:10.1007/11426639_6
7. Herzberg, A.: Folklore, practice and theory of robust combiners. J. Comput. Secur. **17**(2), 159–189 (2009). doi:10.3233/JCS-2009-0336
8. Hohenberger, S., Lewko, A., Waters, B.: Detecting dangerous queries: a new approach for chosen ciphertext security. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 663–681. Springer, Heidelberg (2012). doi:10.1007/978-3-642-29011-4_39