Public-Key Encryption with Lazy Parties

Kenji Yasunaga

Institute of Systems, Information Technologies and Nanotechnologies
Fukuoka SRP Center Building 7F, 2-1-22, Momochihama
Sawara-ku, Fukuoka 814-0001, Japan

yasunaga@isit.or.jp

Abstract

In a public-key encryption scheme, if a sender is not concerned about the security of a message and is unwilling to generate costly randomness, the security of the encrypted message can be compromised. This is caused by the *laziness* of the sender. In this work, we characterize *lazy parties* in cryptography. Lazy parties are regarded as honest parties in a protocol, but they are not concerned about the security of the protocol in a certain situation. In such a situation, they behave in an honest-looking way, and are unwilling to do a costly task. We study, in particular, public-key encryption with lazy parties. Specifically, as the first step toward understanding the behavior of lazy parties in public-key encryption, we consider a rather simple setting in which the costly task is to generate randomness used in algorithms, and parties can choose either costly good randomness or cheap bad randomness. We model lazy parties as rational players who behaves rationally to maximize their utilities, and define a security game between lazy parties and an adversary. A secure encryption scheme requires that the game is conducted by lazy parties in a secure way if they follow a prescribed strategy, and the prescribed strategy is a good equilibrium solution for the game. Since a standard secure encryption scheme does not work for lazy parties, we present some public-key encryption schemes that are secure for lazy parties.

1 Introduction

Consider the following situation. Alice is a teacher of a course "Introduction to Cryptography." She promised to inform the students of their grades by using public-key encryption. Each student prepared his/her public key, and sent it to Alice. Since there are many students taking the course, it is very costly to encrypt the grades of all the student. However, since she promised to use public-key encryption, she decided to encrypt the grades. To encrypt messages, she needs to generate randomness. Generating good randomness is also a costly task. While the grades are personal information for the students and thus they want them to be securely transmitted, the grades are not personal information for Alice. The security of the grades is not her concern. She noticed that, even if she used bad randomness for encryption, no one may detect it. Consequently, she used cheap bad randomness for encryption instead of costly good randomness.

The above situation resulted in an undesirable consequence. This example demonstrates that, if some party in a cryptographic protocol is not concerned about the security and is unwilling to do a costly task, then the security of the protocol may be compromised. The insecurity is caused by the *laziness* of the party. A traditional cryptography did not consider the laziness of players who are regarded as honest. However, the security should be preserved even if such lazy parties exist.

1.1 This Work

We introduce the notion of *lazy parties*, who may compromise the security of cryptographic protocols. We characterize lazy parties such that (1) they are not concerned about the security of the protocol in a certain situation, and (2) they behave in an honest-looking way and are unwilling to do a costly task. We study, in particular, public-key encryption schemes with lazy parties. As the first step toward understanding the behavior of lazy parties in public-key encryption, we consider the following rather simple setting. The sender and the receiver have their own valuable messages. They want to transmit a message securely if it is valuable for them. However, since both the sender and the receiver are lazy, the sender is not willing to do a costly task if a message is not valuable for him, and the receiver vice versa. The costly task we consider is to generate randomness used in algorithms. For simplicity, we assume that players can choose either costly good randomness or cheap bad randomness. While the costly good randomness is a truly random string, the cheap bad randomness is some fixed string in our setting. Our goal is to design public-key encryption schemes in which valuable messages of the sender or the receiver can be transmitted securely by the lazy sender and receiver who may use bad randomness in algorithms.

Formalizing the Problem. We formalize the security of public-key encryption for lazy parties as follows. First we define a security game between a sender, a receiver, and an adversary. The game is a variant of the usual chosen plaintext attack (CPA) game of public-key encryption. In this game we see the sender and the receiver as rational players. The sender and the receiver have their utility functions, the values of which are determined by the outcome of the game, and they play the game to maximize their utilities. Roughly speaking, we say that an encryption scheme is secure for lazy parties if there is a pair of prescribed strategies of the sender and the receiver for the game, the game is conducted in a secure way if they follow the prescribed pair of strategies, and the pair of strategies is a good equilibrium solution. The solution concepts we consider in this work are Nash equilibrium and strict Nash equilibrium, which is stronger than Nash equilibrium.

Impossibility Results. As impossibility results, we show that to achieve the security for lazy parties with a Nash equilibrium solution in our setting, the sender must generate a secret key, and the encryption phase requires at least two rounds. Neither of them is satisfied in the usual public-key encryption. Therefore, we need to consider encryption schemes in which the sender generates a secret key in the key generation phase, and the sender and the receiver interacts at least two times in encrypting a message.

Constructions. The security for lazy parties varies according to what information each player knows. We consider several situations according to the information each player knows, and present a secure encryption scheme for lazy parties in each situation.

First we consider a basic situation in which the receiver does not know whether a message to be encrypted is valuable for him or not, and the sender knows the value of the message for him. We propose a two-round encryption scheme that is secure for lazy parties with a strict Nash equilibrium solution. The idea is simple. First the receiver generates a random string, encrypts it by the public key of the sender, and sends it to the sender. Next the sender recovers the random string from the ciphertext and uses it to encrypt a message by the one-time pad. Since the receiver does not know whether a message to be encrypted is valuable for him or not, the receiver will generate good randomness.

Next, we consider a situation in which the receiver may know whether a message to be encrypted is valuable for him or not. This captures a real-life situation; If we use encryption, in many cases, it is realized not only by the sender but also the receiver that what kind of message will be sent. Under this situation, the above two-round scheme seems no longer secure since the receiver would not generate good randomness if a message to be encrypted is not valuable for him. We show that for any pair of strategies the above two-round scheme cannot achieve the security for lazy parties with a Nash equilibrium solution. Thus we propose a three-round encryption scheme that is secure in this situation. The encryption phase is conducted as follows. First the sender and the receiver perform a key-agreement protocol to share a random string between them so that the shared string will be good randomness if at least one of them uses good randomness in the key-agreement protocol. Then, the sender uses the shared string as randomness in the encryption algorithm. Finally, after recovering a message, the receiver encrypts the message by the sender's public key and makes it public. At first glance, the final step of making the encrypted message public seems redundant, but our scheme does not achieve the security without this step. Our three-round scheme is secure for lazy parties with a strict Nash equilibrium solution.

We generalize the above situation such that both the sender and the receiver may know that a message to be encrypted is valuable for them. The difference from the previous situation is that the sender may be able to know the value of the message for the receiver, and the receiver vice versa. In this situation, we realized that the above three-round scheme has two different pairs of strategies that achieve the security with a strict Nash equilibrium. There is a situation such that one pair yields a higher utility to the sender, and the other pair yields a higher utility to the receiver. Moreover, if the sender follows a strategy that yields a higher utility to him and the receiver also does so, they will conduct an encryption protocol in an insecure way, which is worse for both of them. Thus, we propose a simple way to avoid such a consequence.

Finally, we consider constructing a non-interactive encryption scheme that is secure for lazy parties. We avoid the impossibility result of existing non-interactive schemes by adding some reasonable assumption to lazy parties. The assumption is that players do not want to reveal their secret key to adversaries. Then we employ a *signcryption* scheme for an encryption scheme. A signcryption scheme is a cryptographic primitive that achieves both public-key encryption and signature simultaneously, and thus the sender also has the secret key. Some of signcryption schemes (e.g., [24]) have the *key-exposure property*, which means that the sender's secret key can be efficiently recovered from a ciphertext and its random string. This property seems to be undesirable in a standard setting. However, we show that if a signcryption scheme with the key-exposure property is employed as a public-key encryption scheme, it is secure for lazy parties with a strict Nash equilibrium solution.

1.2 Related Work

Halpern and Pass [16] have introduced *Bayesian machine games* in which players' utilities can depend on the computational costs of their strategies. We could use the framework of Halpern and Pass to define a security of public-key encryption schemes for lazy parties since the utilities of lazy parties depend on their computational cost. We did not use their framework in this work since their framework seems too general for our purpose.

There have been many studies on *rational cryptography* [18, 11, 15], in which players in cryptographic protocols are considered rational players. Much study has been devoted to designing *rational secret sharing* [17, 1, 13, 19, 20, 21, 22, 3]. Recently, the problem of fair two-party computation with rational players was considered [2, 14]. The work in this paper also can be seen as a study of rational cryptography. As far as we know, this is the first study of rational behavior in public-key encryption schemes.

In cryptography, there are several characterizations of parties who are neither honest nor malicious [8, 9, 4]. In particular, the deviations of honest-looking parties were studied in [8, 9]. All types of honest-looking parties defined in [8, 9] deviate from the protocol in a way that is computationally indistinguishable from the view of external or internal parties. This means that any efficient statistical test cannot tell the difference between honest parties and honest-looking parties. In this study, we consider honest-looking parties who may deviate from the protocol by using a fixed string instead of a truly random string. Since the difference between fixed strings and truly random strings can be told by a simple statistical test, the deviations of lazy parties in this study are bolder than honest-looking parties in [8, 9]. Note that all the characterization in [8, 9] appeared in the context of general multiparty computation, not in public-key encryption.

A main problem of public-key encryption with lazy parties is that lazy parties might not use good randomness in algorithms. There are many studies on the security of cryptographic tasks when only weak randomness is available. If there are only high min-entropy sources, not including truly random one, many impossibility results are known [10, 6]. Bellare et al. [5] introduced hedged public-key encryption, which achieves the usual CPA security if good randomness is used, and achieves a weaker security if bad randomness is used. In this work, we consider only two types of randomness sources, truly random ones and fixed ones. We achieve the security by a mechanism such that lazy parties choose to use good randomness for their purpose.

1.3 Future Work

Possible future work is extending the framework of this work to more general settings. For example, in this work, lazy players can choose either truly random (full entropy) strings or fixed (zero entropy) strings as the randomness in algorithms. Since it seems more realistic for players to be able to choose random strings from general entropy sources, extending the framework to such a general setting and defining a reasonable security on that setting are interesting for future work.

Another possible future work is to explore cryptographic protocols that may be compromised in the presence of lazy parties. Although we consider only generating good randomness as a costly task, it is possible to consider another thing as cost, such as time for computation and delay in the protocol.

1.4 Organization

In Section 2, we introduce the CPA game for lazy parties, define utility functions of lazy parties, and provide a definition of the CPA security for lazy parties. Our secure encryption schemes in various situations are presented in Section 3. The definition of strict Nash equilibrium and all the proofs of propositions and theorems are presented in the appendix.

1.5 Notations

A function $\epsilon(\cdot)$ is called *negligible* if for any constant c, $\epsilon(n) < 1/n^c$ for every sufficiently large n. For two families of random variables $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$, we say that X and Y are *computationally indistinguishable*, denoted by $X \approx_c Y$, if for every probabilistic polynomial-time (PPT) distinguisher D, there is a negligible function $\epsilon(\cdot)$ such that $|\Pr[D(X_n) = 1] - \Pr[D(Y_n) = 1]| \leq \epsilon(n)$ for every sufficiently large n. For a probabilistic algorithm A, the output of A when the input is x is denoted by $A(x)$, and denoted by $A(x; r)$ when the random string r used in A is represented explicitly.

2 Lazy Parties in Public-Key Encryption

We consider the following setting of public-key encryption between a lazy sender and a lazy receiver. Each lazy party has a set of valuable messages, and wants a message to be sent securely if it is valuable for that party. If a message to be encrypted is not valuable for a party, he is not concerned about the security of the message, and does not want to use good randomness in the computation. In this paper, we consider only two types of randomness, good randomness and bad randomness. Good randomness is a truly random string but costly. Bad randomness is generated with zero cost, but is some fixed string.

We formalize the security as follows. Lazy parties are considered as rational players who have some utility functions and behave rationally to maximize their utilities. We define a security game between a lazy sender, a lazy receiver, and an adversary. Then, we say that an encryption scheme is secure if there is a pair of prescribed strategies of the sender and the receiver for the game, the game is conducted in a secure way if they follow the strategies, and the pair of strategies is a good equilibrium solution.

We define public-key encryption as an *interactive* protocol between a sender and a receiver. The reason is that we cannot achieve the security if the sender does not have a secret key or the encryption phase is conducted in one round, which will be described in the last of this section. In the key generation phase, both the sender and the receiver generate their own public key and secret key, then each public key is distributed to the other player. In the encryption phase, the players conduct an interactive protocol in which the sender has a message as an input. After the encryption phase, the receiver can recover the message by running the decryption algorithm. This definition is much more general than the usual public-key encryption, in which only the receiver generates a public key and a secret key, and the encryption phase is just sending a ciphertext from the sender to the receiver.

Definition 1 (Public-key encryption scheme)**.** An n-round public-key encryption scheme Π is the tuple $(\{\textsc{Gen}_w\}_{w\in\{S,R\}}, \{\textsc{Enc}_i\}_{i\in\{1,\dots,n\}}, \textsc{Dec})$ such that

- **Key generation:** For each $w \in \{S, R\}$, on input 1^k, $\textsc{Gen}_w$ outputs (pk_w, sk_w). Let $\mathcal{M}$ denote the message space.
- **Encryption:** For a message $m \in \mathcal{M}$, set $st_S = (pk_S, pk_R, sk_S, m)$, $st_R = (pk_S, pk_R, sk_R)$, and $c_0 = \perp$. Let $w \in \{S, R\}$ be the first sender, and $\bar{w} \in \{S, R\} \setminus \{w\}$ the second sender. For each round $i \in \{1, \dots, n\}$, when i is odd, $\textsc{Enc}_i(c_{i-1}, st_w)$ outputs (c_i, st'_w), and st_w is updated to st'_w, and when i is even, $\textsc{Enc}_i(c_{i-1}, st_{\bar{w}})$ outputs $(c_i, st'_{\bar{w}})$, and $st_{\bar{w}}$ is updated to $st'_{\bar{w}}$.

- **Decryption:** After the encryption phase, on input st_R, $\textsc{Dec}$ outputs $\hat{m}$.
- **Correctness:** For any message $m \in \mathcal{M}$, after the encryption phase, $\textsc{Dec}(st_R) = m$.

We provide a definition of the chosen plaintext attack (CPA) game for lazy parties. The game is a variant of the usual CPA game for public-key encryption. The game is conducted as follows. The sender S (and the receiver R) has his valuable message space $\mathcal{M}_S$ (and $\mathcal{M}_R$), which is a subset of $\{0,1\}^*$. First, each player $w \in \{S, R\}$ are asked to choose good randomness or bad randomness for the key generation algorithm. If player w chooses good randomness, a random string r_w^g for key generation is sampled as a truly random string. Otherwise, r_w^g is generated by the adversary of this game. Then, pairs of public and secret keys for the two parties are generated using r_w^g as a random string, and the public keys are distributed to the sender, the receiver, and the adversary. Next, the adversary generates two sequences $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$ of challenge messages, where $\boldsymbol{m}_b = (m_{b,1}, \ldots, m_{b,\ell})$ for $b \in \{0,1\}$ and some polynomial ℓ. After that, the challenger chooses $b \in \{0,1\}$ uniformly at random. The sender receives $\boldsymbol{m}_b$ and asked to choose good or bad randomness for the encryption protocol. If he chooses good randomness, random strings $r_{i,j}^e$ for encryption is sampled as truly random strings, where $r_{i,j}^e$ represents a random string used in the j-th round of the encryption for the i-th message $m_{b,i}$. Otherwise, strings $r_{i,j}^e$'s are generated by the adversary. Similarly, the receiver also asked to choose good or bad randomness for the encryption protocol without seeing the challenge messages $\boldsymbol{m}_b$, and random strings $r_{i,j}^e$'s are generated in the same way as the sender. Then, a sequence of challenge messages are encrypted using $r_{i,j}^e$'s as random strings. Finally, the adversary receives a sequence of challenge ciphertexts, and outputs a guess $b' \in \{0,1\}$. The outcome of the game consists of five values $\mathsf{Win}, \mathsf{Val}_S, \mathsf{Val}_R, \mathsf{Num}_S$, and Num_R. The value Win takes 1 if the guess of the adversary is correct, namely $b = b'$, and 0 otherwise. The value Val_w for player $w \in \{S, R\}$ takes 1 if there is at least one valuable message for player w in the sequence $\boldsymbol{m}_b$ of challenge messages, and 0 otherwise. The value Num_w for player $w \in \{S, R\}$ represents the number of times that player w chose good randomness in the game, which is between 0 and 2.

In the following, we provide a formal definition of the CPA game for lazy parties. For a probabilistic algorithm A, we denote by $\ell(A)$ the length of random bits required in running A. We define $\textsc{Samp}(\cdot)$ to be an algorithm such that $\textsc{Samp}(A)$ samples a random string from $\{0,1\}^{\ell(A)}$.

Definition 2 (CPA game for lazy parties)**.** Let $\Pi = (\{\textsc{Gen}_w\}_{w \in \{S,R\}}, \{\textsc{Enc}_i\}_{i \in \{1,\ldots,n\}}, \textsc{Dec})$ be a public-key encryption scheme. For an adversary A, the security parameter k, valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$, and a pair of strategies (σ_S, σ_R), we define the following game.

Game$^{\mathrm{cpa}}(\Pi, k, A, \mathcal{M}_S, \mathcal{M}_R, \sigma_S, \sigma_R)$:

1. **Choice of randomness for key generation:** For each $w \in \{S, R\}$, compute $x_w^g \leftarrow \sigma_w(1^k, \mathcal{M}_w)$, where $x_w^g \in \{\mathsf{Good}, \mathsf{Bad}\}$. If $x_w^g = \mathsf{Bad}$, then given $(1^k, w)$, A outputs $r_w^g \in \{0,1\}^{\ell(\textsc{Gen}_w(1^k))}$. Otherwise sample $r_w^g \leftarrow \textsc{Samp}(\textsc{Gen}_w(1^k))$.
2. **Key generation:** For each $w \in \{S, R\}$, generate $(pk_w, sk_w) \leftarrow \textsc{Gen}_w(1^k; r_w^g)$. Let $\mathcal{M}$ be the corresponding message space.
3. **Challenge generation:** Given (pk_S, pk_R), A outputs $\boldsymbol{m}_0 = (m_{0,1}, \ldots, m_{0,\ell})$ and $\boldsymbol{m}_1 = (m_{1,1}, \ldots, m_{1,\ell})$, where $\ell \in \mathbb{N}$ is a polynomial in k and $m_{i,j} \in \mathcal{M}$ for each $i \in \{0,1\}$ and $j \in \{1, \ldots, \ell\}$. Then sample $b \in \{0,1\}$ uniformly at random.
4. **Choice of randomness for encryption:** For each $w \in \{S, R\}$, compute $x_w^e \leftarrow \sigma_w(pk_S, pk_R, sk_w, aux_w)$, where $x_w^e \in \{\mathsf{Good}, \mathsf{Bad}\}$, $aux_S = \boldsymbol{m}_b$, and $aux_R = \perp$. If $x_w^e = \mathsf{Bad}$,

then given w, A outputs $r^e_{i,j} \in \{0,1\}^{\ell(\textsc{Enc}_j(\cdot))}$ for each $i \in \{1,\dots,\ell\}$ and $j \in \{1,\dots,n\}$. Otherwise sample $r^e_{i,j} \leftarrow \textsc{Samp}(\textsc{Enc}_j(\cdot))$ for each $i \in \{1,\dots,\ell\}$ and $j \in \{1,\dots,n\}$. Let w be the first sender, and $\bar{w}$ the second sender, which are determined by Π.

5. **Encryption:** For $i \in \{1,\dots,\ell\}$, do the following. Set $st_S = (pk_S, pk_R, sk_S, m_{b,i})$, $st_R = (pk_S, pk_R, sk_R)$, and $c_{i,0} = \perp$. For $j \in \{1,\dots,n\}$, when j is odd, compute $(c_{i,j}, st'_w) \leftarrow \textsc{Enc}_j(c_{i,j-1}, st_w; r^e_{i,j})$ and st_w is updated to st'_w, and when j is even, compute $s(c_{i,j}, st'_{\bar{w}}) \leftarrow \textsc{Enc}_j(c_{i,j-1}, st_{\bar{w}}; r^e_{i,j})$ and $st_{\bar{w}}$ is updated to $st'_{\bar{w}}$.
6. **Guess:** Given $\{c_{i,j} : i \in \{1,\dots,\ell\}, j \in \{1,\dots,n\}\}$, A outputs $b' \in \{0,1\}$.
7. Output $(\mathsf{Win}, \mathsf{Val}_S, \mathsf{Val}_R, \mathsf{Num}_S, \mathsf{Num}_R)$, where Win takes 1 if $b' = b$, and 0 otherwise, Val_w takes 1 if $m_{b,i} \in \mathcal{M}_w$ for some $i \in \{1,\dots,\ell\}$, and 0 otherwise, and Num_w represents the number of times that σ_w output Good in the game.

Next, we define the utility functions of lazy sender and receiver for this game.

Definition 3 (Utility function for CPA game)**.** Let (σ_S, σ_R) be a pair of strategies of the game $\mathbf{Game}^{\mathrm{cpa}}$. The utility of player $w \in \{S, R\}$ when the outcome $\mathsf{Out} = (\mathsf{Win}, \mathsf{Val}_S, \mathsf{Val}_R, \mathsf{Num}_S, \mathsf{Num}_R)$ happens is defined by

$$u_w(\mathsf{Out}) = (-\alpha_w) \cdot \mathsf{Win} \cdot \mathsf{Val}_w + (-\beta_w) \cdot \mathsf{Num}_w,$$

where $\alpha_w, \beta_w \in \mathbb{R}$ are some non-negative constant. Let q_w be the maximum number that Num_w can take. (q_w is either 0, 1, or 2, depending on the scheme Π.) We say that the utility is *valid* if $\alpha_w/2 > q_w \cdot \beta_w$ for each $w \in \{S, R\}$.

The utility when the players follow a pair of strategies (σ_S, σ_R) is defined by

$$U_w(\sigma_S, \sigma_R) = \min_{A, \mathcal{M}_S, \mathcal{M}_R} \{\mathbf{E}[u_w(\mathsf{Out})]\},$$

where Out is the outcome of the game $\mathbf{Game}^{\mathrm{cpa}}(\Pi, k, A, \mathcal{M}_S, \mathcal{M}_R, \sigma_S, \sigma_R)$, and the minimum is taken over all PPT adversaries A and valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$ for every sufficiently large k.

Note that, in the above definition, we take the minimum over all possible adversaries (and valuable message spaces) to define the utility when players follow a pair of strategies (σ_S, σ_R). This is because we would like to evaluate a pair of strategies (σ_S, σ_R) by considering the worst-case for possible adversaries and valuable message spaces. In other words, we would like to say that a pair of strategies is good if it is guaranteed to yield high utility for any adversary and players, who are associated with valuable message spaces.

Note that the validity condition of the utility guarantees that players have an incentive to use good randomness for achieving the security. If players do not use good randomness, then there is an adversary such that $\mathsf{Win} \cdot \mathsf{Val}_w$ is always 1. The best we can hope for is that the expected value of $\mathsf{Win} \cdot \mathsf{Val}_w$ is $1/2$ (plus some negligible value), which increases the utility by $\alpha_w/2$. Since Num_w takes at most q_w in the game, the inequality $\alpha_w/2 > q_w \cdot \beta_w$ means that achieving the security is worth paying the cost of good randomness. Hereafter, we assume that the utility functions are valid.

As game theoretic solution concepts, we define Nash equilibrium and strict Nash equilibrium. Since any strategy that a player can follow should be computable in a polynomial time and a negligible difference of the outcome of the game should be ignored for PPT algorithms, we consider a computational Nash equilibrium.

Definition 4 (Computational Nash equilibrium)**.** A pair of PPT strategies (σ_S, σ_R) of the game $\mathbf{Game}^{\rm cpa}$ is called a *computational Nash equilibrium* if for each player $w \in \{S, R\}$, it holds that

$$U_w(\sigma_S^*, \sigma_R^*) \leq U_w(\sigma_S, \sigma_R) + \epsilon(k)$$

for every PPT strategy σ'_w of player w, where $(\sigma_S^*, \sigma_R^*) = (\sigma'_S, \sigma_R)$ if $w = S$, $(\sigma_S^*, \sigma_R^*) = (\sigma_S, \sigma'_R)$ otherwise, and $\epsilon(\cdot)$ is a negligible function.

Strict Nash equilibrium guarantees that if a player deviates from the strategy, then the utility of the player decreases by a non-negligible amount. A definition of strict Nash equilibrium is presented in Appendix A.

We define the security of encryption schemes for lazy parties.

Definition 5 (CPA security for lazy parties)**.** Let $\Pi = (\{\textsc{Gen}_w\}_{w\in\{S,R\}}, \{\textsc{Enc}_i\}_{i\in\{1,\ldots,n\}}, \textsc{Dec})$ be a public-key encryption scheme, and (σ_S, σ_R) a pair of strategies of the game $\mathbf{Game}^{\rm cpa}$. We say that $(\Pi, \sigma_S, \sigma_R)$ is *CPA secure with a (strict) Nash equilibrium for* $\mathbf{Game}^{\rm cpa}$ if

1. For any PPT adversary A, valuable message spaces $\mathcal{M}_S, \mathcal{M}_R$, and every sufficiently large k, it holds that $\Pr[\mathsf{Win}\cdot(\mathsf{Val}_S + \mathsf{Val}_R) \neq 0] \leq 1/2 + \epsilon(k)$, where $\mathsf{Win}, \mathsf{Val}_S, \mathsf{Val}_R$ are components of the outcome of the game $\mathbf{Game}^{\rm cpa}(\Pi, k, A, \mathcal{M}_S, \mathcal{M}_R, \sigma_S, \sigma_R)$, and $\epsilon(\cdot)$ is a negligible function;
2. The pair of strategies (σ_S, σ_R) is a computational (strict) Nash equilibrium.

In the first condition, we evaluate the value of $\mathsf{Win} \cdot (\mathsf{Val}_S + \mathsf{Val}_R)$ since if $\mathsf{Val}_S + \mathsf{Val}_R = 0$, all the messages chosen by the adversary are not valuable for both the sender and the receiver.

Note that the usual CPA security of usual (non-interactive) public-key encryption is a special case of the above definition. If the scheme Π consists of $(\textsc{Gen}_R, \textsc{Enc}_1, \textsc{Dec})$, a pair of strategies (σ_S, σ_R) is such that both σ_S and σ_R always output Good, and the second condition of the security is not considered, then the above security is equivalent to the usual CPA security of public-key encryption. For a usual encryption scheme $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$, we say that Π is *CPA secure* if it is CPA secure in this sense.

Impossibility Results. The first observation for achieving the security for lazy parties is that the sender must generate a secret key and the encryption phase requires at least two rounds, neither of them is satisfied in the usual public-key encryption. Roughly speaking, the reason why secure schemes require to generate a secret key for a sender is that if the messages to be encrypted are valuable for the receiver but not for the sender, the sender does not use good randomness and thus the adversary can correctly guess which of the challenge messages was encrypted because she known all the input to the sender. Furthermore, even if the sender has his secret key, if the encryption phase is 1-round, there is an adversary who can guess the challenge correctly. Consider an adversary who submits challenge messages such that one consists of the same two messages and the other consists of different two messages, and all the messages are valuable for the receiver but not for the sender. Then the sender does not use good randomness, and thus the adversary can choose randomness for encryption. If she choose the same random strings for two challenge messages, then although the adversary does not know the secret key of the sender, since the encryption is 1-round, she can correctly guess which of the challenges was encrypted by checking whether given two challenge ciphertexts are the same or not. See Appendix B for the formal statements and proofs.

3 Secure Encryption Schemes for Lazy Parties

3.1 Two-Round Encryption Scheme

We present a two-round public-key encryption scheme that is CPA secure with a strict Nash equilibrium. The encryption phase is conducted as follows. First the receiver generates a random string, encrypts it by the public key of the sender, and sends it to the sender. Next the sender encrypt a messages by the one-time pad, in which the sender uses the random string received from the receiver. The receiver can recover the message since he knows the random string. Our scheme is based on any CPA-secure public-key encryption scheme $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ in which the message space is $\{0,1\}^\mu$ and the length of random bits required in $\textsc{Enc}$ is μ.

The description of our two-round scheme $\Pi_{\mathsf{two}} = (\textsc{Gen}_S, \{\textsc{Enc}_i\}_{i\in\{1,2\}}, \textsc{Dec}_R)$ is the following.

- $\textsc{Gen}_S(1^k)$: Generate $(pk_S, sk_S) \leftarrow \textsc{Gen}(1^k)$, and output (pk_S, sk_S).
  Let $\mathcal{M} = \{0,1\}^\mu$ be the message space, where μ is a polynomial in k. Set $st_S = (pk_S, sk_S)$ and $st_R = pk_S$.
- $\textsc{Enc}_1(st_R)$: Sample $r \in \{0,1\}^\mu$ uniformly at random, compute $c_1 \leftarrow \textsc{Enc}(pk_S, r)$, and output $(c_1, (sk_R, r))$.
  $\textsc{Enc}_2(c_1, st_S)$: Compute $\hat{r} \leftarrow \textsc{Dec}(sk_S, c_1)$ and $c_2 = m \oplus \hat{r}$, and output c_2.
- $\textsc{Dec}_R(c_2, (sk_R, r))$: Compute $\hat{m} = c_2 \oplus r$ and output $\hat{m}$.

We define a pair of strategies (σ_S, σ_R) such that

- $\sigma_S(1^k, \mathcal{M}_S)$ outputs Good with probability 1. $\sigma_S(pk_S, sk_S, aux_S)$ is not defined.
- $\sigma_R(1^k, \mathcal{M}_R)$ is not defined. $\sigma_R(pk_S, aux_R)$ outputs Good with probability 1.

Theorem 1. *If Π is CPA secure, $(\Pi_{\mathsf{two}}, \sigma_S, \sigma_R)$ is CPA secure with a strict Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}$*.*

3.2 Additional Information to the Receiver

In this section, we consider a situation in which the receiver may know whether a message to be encrypted is valuable for the receiver or not. This situation can be reflected by changing the game $\mathbf{Game}^{\mathrm{cpa}}$ such that the adversary can choose either "$aux_R = \perp$" or "$aux_R = \mathsf{Val}_R$" in the challenge generation phase. Let $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{R}}$ denote the modified game.

In this situation, the scheme presented in Section 3.1 is no longer secure. Intuitively, this is because the receiver does not generate good randomness if a message to be encrypted is not valuable for him.

Proposition 1. *For any pair of strategies (σ_S, σ_R), $(\Pi_{\mathsf{two}}, \sigma_S, \sigma_R)$ is not CPA secure with a Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{R}}$*.*

We present a three-round encryption scheme that is secure for $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{R}}$. In the encryption phase, first the sender and the receiver perform a key-agreement protocol that generates a random string shared between them. The shared string is good randomness if one of the sender and the receiver uses good randomness in the key-agreement protocol. Then, the sender uses the shared string as randomness to encrypt a message. Finally, after recovering a message, the receiver encrypt the message by the sender's public key and makes it public. As described later, the final step is necessary to achieve the security. Our scheme is based on any CPA-secure public-key encryption scheme $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ in which the message space is $\{0,1\}^{2\mu}$ and the length of random bits required in $\textsc{Enc}$ is μ.

The description of the encryption scheme $\Pi_{\mathsf{three}} = (\{\textsc{Gen}_w\}_{w\in\{S,R\}}, \{\textsc{Enc}_i\}_{i\in\{1,2,3\}})$ is the following. The decryption algorithm does not exist in Π_{three} since the receiver decrypts a message in computing $\textsc{Enc}_3$.

- $\textsc{Gen}_w(1^k)$: Generate $(pk_w, sk_w) \leftarrow \textsc{Gen}(1^k)$, and output (pk_w, sk_w).
  Let $\mathcal{M} = \{0,1\}^{2\mu}$ be the message space, where μ is a polynomial in k. Set $st_S = (pk_S, pk_R, sk_S)$ and $st_R = (pk_S, pk_R, sk_R)$.
- $\textsc{Enc}_1(st_R)$: Sample $r_1 \in \{0,1\}^\mu$ uniformly at random, compute $c_1 \leftarrow \textsc{Enc}(pk_S, r_1)$, and output $(c_1, (sk_R, r_1))$.
  $\textsc{Enc}_2(c_1, st_S)$: Sample $r_2 \in \{0,1\}^\mu$ uniformly at random and compute $c_2 \leftarrow \textsc{Enc}(pk_R, r_2)$ and $\hat{r}_1 \leftarrow \textsc{Dec}(sk_S, c_1)$. Then set $r_L \circ r_R = \hat{r}_1 \oplus r_2$ such that $|r_L| = |r_R| = \mu$, compute $c_3 \leftarrow \textsc{Enc}(pk_R, m; r_L)$, and output $((c_2, c_3), sk_S)$, where $x \circ y$ denote the concatenation of strings x and y.
  $\textsc{Enc}_3((c_2, c_3), st_R)$: Compute $\hat{r}_2 \leftarrow \textsc{Dec}(sk_R, c_2)$, set $\hat{r}_L \circ \hat{r}_R = r_1 \oplus \hat{r}_2$, compute $\hat{m} \leftarrow \textsc{Dec}(sk_S, c_3)$ and $c_4 \leftarrow \textsc{Enc}(pk_S, \hat{m}; \hat{r}_R)$, and make c_4 public. The decrypted message is $\hat{m}$.

We define a pair of strategies (σ_S, σ_R) such that

- $\sigma_S(1^k, \mathcal{M}_S)$ outputs Good with probability 1. $\sigma_S(pk_S, pk_R, sk_S, aux_S)$ outputs Good if $m_{b,i} \in \mathcal{M}_S$ for some $i \in \{1, \dots, \ell\}$, and Bad otherwise.
- $\sigma_R(1^k, \mathcal{M}_R)$ outputs Good with probability 1. $\sigma_R(pk_S, pk_R, sk_R, aux_R)$ outputs Good if $aux_R = \perp$ or $\mathsf{Val}_R = 1$, and Bad otherwise.

At first glance, it does not seem necessary to make c_4 public at the third round of the encryption phase. However, it is necessary to do so because if not, the sender can achieve the security without using good randomness in the key generation phase.

Theorem 2. *If Π is CPA secure, $(\Pi_{\mathsf{three}}, \sigma_S, \sigma_R)$ is CPA secure with a strict Nash equilibrium for* $\mathbf{Game}_{\mathrm{R}}^{\mathrm{cpa}}$.

3.3 Additional Information to the Sender and the Receiver

In this section, we consider a situation in which both the sender and the receiver may know that a message to be encrypted is valuable for them. The situation is different from that of the previous section because the sender may be able to know the value of a message for the receiver, and the receiver vice versa. This situation can be reflected by changing the game $\mathbf{Game}_{\mathrm{R}}^{\mathrm{cpa}}$ such that the adversary can choose either "$aux_S = \boldsymbol{m}_b$" or "$aux_S = (\boldsymbol{m}_b, \mathsf{Val}_R)$", and either "$aux_R = \perp$", "$aux_R = \mathsf{Val}_R$", "$aux_R = \mathsf{Val}_S$", or "$aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$" in the challenge generation phase. Let $\mathbf{Game}_{\mathrm{S,R}}^{\mathrm{cpa}}$ denote the modified game.

In this game, the scheme Π_{three} has two different strict Nash equilibria.

Proposition 2. *There are two pairs of strategies (σ_S, σ_R) and (ρ_S, ρ_R) such that $\sigma_S \not\approx \rho_S$, $\sigma_R \not\approx \rho_R$, and both $(\Pi_{\mathsf{three}}, \sigma_S, \sigma_R)$ and $(\Pi_{\mathsf{three}}, \rho_S, \rho_R)$ are CPA secure with strict Nash equilibrium for $\mathbf{Game}_{\mathrm{S,R}}^{\mathrm{cpa}}$, where the notation $\approx$ is defined in Appendix A. Furthermore, there is a PPT adversary*

A and valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$ such that $\mathbf{E}[u_S(\mathsf{Out}_\rho)] - \mathbf{E}[u_S(\mathsf{Out}_\sigma)] \geq \beta_S - \epsilon(k)$ and $\mathbf{E}[u_R(\mathsf{Out}_\sigma)] - \mathbf{E}[u_R(\mathsf{Out}_\rho)] \geq \beta_R - \epsilon(k)$ for every sufficiently large k, where Out_σ is the outcome of the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{S,R}}$ in which players follow (σ_S, σ_R), Out_ρ is the outcome of the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{S,R}}$ in which players follow (ρ_S, ρ_R), and $\epsilon(\cdot)$ is a negligible function.

As shown in the proof, the difference between outputs of (σ_S, σ_R) and (ρ_S, ρ_R) is only in the case that $aux_S = (\boldsymbol{m}_b, \mathsf{Val}_R)$, $aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$, and $\mathsf{Val}_S = \mathsf{Val}_R = 1$. In this case, the sender uses good randomness and the receiver uses bad randomness in (σ_S, σ_R), while the sender uses bad randomness and the receiver uses good randomness in (ρ_S, ρ_R). Therefore, the sender prefers to following (ρ_S, ρ_R), while the receiver prefers to following (σ_S, σ_R). It is difficult to determine which pair of strategies the players follow. If the protocol have started, but the sender and the receiver have not agreed on which pair of strategies they follow, the outcome can be worse for both of them. If the sender follows (ρ_S, ρ_R) and the receiver follows (σ_S, σ_R) when $\mathsf{Val}_S = \mathsf{Val}_R = 1$, in this case both players are to use bad randomness in the encryption, thus the adversary can correctly guess b with probability 1. Such an outcome should be avoided for both players.

There is a simple way of avoiding that outcome. In the encryption phase, if $x_R^{\mathrm{e}} \neq \mathsf{Good}$, the receiver uses the all-zero string as a random string. Since the sender can verify if the random string chosen by the receiver is all-zero or not, if so, the sender will use good randomness if a message is valuable. The all-zero string is a signal that the receiver did not used good randomness.

3.4 Signcryption with an Additional Assumption

A signcryption scheme is one of cryptographic primitives that achieves both public-key encryption and signature simultaneously. In particular, a secret key for encryption and a signing key for signature is common, and a public key for encryption and a verification key for signature is also common.

We show that signcryption schemes with some property can achieve the CPA security for lazy parties if we add an assumption for players. The assumption is that players do not want to reveal their secret keys. This is plausible since, if the secret key of some player is revealed, it is equivalent to that the encrypted messages to the player are revealed and the signatures of the player are forged.

Formally, a signcryption scheme Π_{sigenc} consists of three PPT algorithms $(\{\textsc{Gen}_w\}_{w\in\{S,R\}}, \textsc{SigEnc}, \textsc{VerDec})$ such that

- $\textsc{Gen}_w(1^k)$: Output a signing/decryption key (secret key) sk_w and a verification/encryption key (public key) pk_w; Let $\mathcal{M}$ denote the message space.
- $\textsc{SigEnc}(pk_R, sk_S, m)$: For a message $m \in \mathcal{M}$, output the ciphertext c;
- $\textsc{VerDec}(pk_S, sk_R, c)$: For a ciphertext c, output $\perp$ if the verification fails, and the decrypted message $\hat{m}$ otherwise.

Some of signcryption schemes (e.g., [24]) have the *key-exposure* property that, if the randomness used in $\textsc{SigEnc}$ is revealed, then the secret key of the sender is efficiently computed from the randomness. This property seems to be undesirable in a standard setting. However, if a signcryption scheme with key-exposure property is used as a public-key encryption scheme, it can achieve the CPA security for lazy parties.

We modify the game $\mathbf{Game}^{\mathrm{cpa}}$ such that the adversary outputs (b', sk'_S) in the guess phase, and Secret is included in the output of the game, where Secret takes 1 if $sk_S = sk'_S$ and 0 otherwise. Let $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{secret}}$ denote the modified game.

The utility function for the sender when the outcome $\mathsf{Out} = (\mathsf{Win}, \mathsf{Val}_S, \mathsf{Val}_R, \mathsf{Num}_S, \mathsf{Num}_R, \mathsf{Secret})$ happens is defined by

$$u_S(\mathsf{Out}) = (-\alpha_S) \cdot \mathsf{Win} \cdot \mathsf{Val}_S + (-\beta_S) \cdot \mathsf{Num}_S + (-\gamma_S) \cdot \mathsf{Secret},$$

where $\gamma_S \in \mathbb{R}$ is a non-negative constant such that $\gamma_S > \alpha_S/2 + q_S \cdot \beta_S$. The condition on γ_S implies that achieving $\mathsf{Secret} = 0$ is the most valuable for the sender.

We define a pair of strategies (σ_S, σ_R) for the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{secret}}$ such that

- $\sigma_S(1^k, \mathcal{M}_S)$ outputs Good with probability 1. $\sigma_S(pk_S, sk_S, aux_S)$ outputs Good with probability 1.
- $\sigma_R(1^k, \mathcal{M}_R)$ outputs Good with probability 1. $\sigma_R(pk_S, aux_R)$ is not defined.

Theorem 3. *Let $\Pi_{\mathsf{sigenc}} = (\{\textsc{Gen}_w\}_{w \in \{S,R\}}, \textsc{SigEnc}, \textsc{VerDec})$ be a signcryption scheme with CPA security and key-exposure property. Then $(\Pi_{\mathsf{sigenc}}, \sigma_S, \sigma_R)$ is CPA secure with a strict Nash equilibrium for the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{secret}}$.*

References

[1] I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In E. Ruppert and D. Malkhi, editors, *PODC*, pages 53–62. ACM, 2006.

[2] G. Asharov, R. Canetti, and C. Hazay. Towards a game theoretic view of secure computation. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 426–445. Springer, 2011.

[3] G. Asharov and Y. Lindell. Utility dependence in correct and fair rational secret sharing. *J. Cryptology*, 24(1):157–202, 2011.

[4] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Journal of Cryptology*, 23(2):281–343, 2010.

[5] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT*, pages 232–249, 2009.

[6] C. Bosley and Y. Dodis. Does privacy require true randomness? In *TCC*, pages 1–20, 2007.

[7] R. Canetti, editor. *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*. Springer, 2008.

[8] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996.

[9] R. Canetti and R. Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? (extended abstract). In *STOC*, pages 255–264, 1999.

[10] Y. Dodis, S. J. Ong, M. Prabhakaran, and A. Sahai. On the (im)possibility of cryptography with imperfect randomness. In *FOCS*, pages 196–205, 2004.

[11] Y. Dodis and T. Rabin. Cryptography and game theory. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, pages 181–207. Cambridge University Press, 2007.

[12] G. Fuchsbauer, J. Katz, and D. Naccache. Efficient rational secret sharing in standard communication networks. In *TCC*, pages 419–436, 2010.

[13] S. D. Gordon and J. Katz. Rational secret sharing, revisited. In R. D. Prisco and M. Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 229–241. Springer, 2006.

[14] A. Groce and J. Katz. Fair computation with rational players. *IACR Cryptology ePrint Archive*, 2011:396, 2011.

[15] J. Y. Halpern. Computer science and game theory. In S. N. Durlauf and L. E. Blume, editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, 2008.

[16] J. Y. Halpern and R. Pass. Game theory with costly computation. In *Innovations in Computer Science*, pages 120–142, 2010.

[17] J. Y. Halpern and V. Teague. Rational secret sharing and multiparty computation: extended abstract. In L. Babai, editor, *STOC*, pages 623–632. ACM, 2004.

[18] J. Katz. Bridging game theory and cryptography: Recent results and future directions, 2008.

[19] G. Kol and M. Naor. Cryptography and game theory: Designing protocols for exchanging information. In Canetti [7], pages 320–339.

[20] G. Kol and M. Naor. Games for exchanging information. In C. Dwork, editor, *STOC*, pages 423–432. ACM, 2008.

[21] S. Micali and A. Shelat. Purely rational secret sharing (extended abstract). In Reingold [23], pages 54–71.

[22] S. J. Ong, D. C. Parkes, A. Rosen, and S. P. Vadhan. Fairness with an honest minority and a rational majority. In Reingold [23], pages 36–53.

[23] O. Reingold, editor. *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*. Springer, 2009.

[24] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In *CRYPTO*, pages 165–179, 1997.

A Strict Nash Equilibrium

We define a computational strict Nash equilibrium. The definition is based on that of [12], which appeared in the context of rational secret sharing.

Definition 6 (Equivalent strategy)**.** Let (σ_S, σ_R) be a pair of strategies of the game $\mathbf{Game}^{\mathrm{cpa}}$, and σ'_w any strategy of player $w \in \{S, R\}$. We say σ'_w is *equivalent to* σ_w, denoted by $\sigma'_w \approx \sigma_w$, if for any PPT adversary A and valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$,

$$\{\mathsf{Trans}(1^k, \sigma_w)\} \approx_c \{\mathsf{Trans}(1^k, \sigma'_w)\},$$

where $\mathsf{Trans}(1^k, \rho_w)$ represents the transcript of the game $\mathbf{Game}^{\mathrm{cpa}}(\Pi, k, A, \mathcal{M}_S, \mathcal{M}_R, \sigma^*_S, \sigma^*_R)$, which includes all values generated in the game except the internal random coin of σ'_w, and $(\sigma^*_S, \sigma^*_R) = (\sigma'_S, \sigma_R)$ if $w = S$, $(\sigma^*_S, \sigma^*_R) = (\sigma_S, \sigma'_R)$ otherwise.

Definition 7 (Computational strict Nash equilibrium)**.** A pair of strategies (σ_S, σ_R) of the game $\mathbf{Game}^{\mathrm{cpa}}$ is called a *computational strict Nash equilibrium* if

1. (σ_S, σ_R) is a Nash equilibrium;
2. For any $w \in \{S, R\}$ and any $\sigma'_w \not\approx \sigma_w$, there is a constant $c > 0$ such that $U_w(\sigma^*_S, \sigma^*_R) \leq U_w(\sigma_S, \sigma_R) - 1/k^c$ for infinitely many k, where $(\sigma^*_S, \sigma^*_R) = (\sigma'_S, \sigma_R)$ if $w = S$, $(\sigma^*_S, \sigma^*_R) = (\sigma_S, \sigma'_R)$ otherwise.

B Impossibility of Secure Schemes without Sender's Secret Key or with Non-interactive Encryption

Proposition 3. *For any public-key encryption scheme* $\Pi = (\{\textsc{Gen}_w\}_{w \in \{S,R\}}, \{\textsc{Enc}_i\}_{i \in \{1,\ldots,n\}}, \textsc{Dec})$ *and any pair of strategies* (σ_S, σ_R)*, if* $\textsc{Gen}_S$ *does not output* sk_S*, then* $(\Pi, \sigma_S, \sigma_R)$ *is not CPA secure with a Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}$*.*

Proof. Suppose that $(\Pi, \sigma_S, \sigma_R)$ is CPA secure with a Nash equilibrium. Consider an adversary A who submits challenge messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$ such that $\boldsymbol{m}_0 = m_0$, $\boldsymbol{m}_1 = m_1$, $m_0 \neq m_1$, and $m_0, m_1 \in \mathcal{M}_R \setminus \mathcal{M}_S$. Since any challenge message is not in $\mathcal{M}_S$, the best strategy of the sender for A in the encryption phase is to choose $x^{\mathrm{e}}_S = \mathsf{Bad}$ regardless of the receiver's strategy. Therefore, $\sigma_S(pk_S, pk_R, sk_S, aux_S) = \mathsf{Bad}$ with probability at least $1 - \epsilon(k)$, where $\epsilon(\cdot)$ is a negligible function. Then, since A knows all the input to the sender in the encryption phase, which consists of $st_S = (pk_S, pk_R, aux_S, \boldsymbol{m}_b)$ and the random strings for encryption, A can correctly guess b from $c_{1,1}, \ldots, c_{1,n}$. This implies that the first condition of the CPA security does not hold. □

Proposition 4. *For any* 1*-round public-key encryption scheme* $\Pi = (\{\textsc{Gen}_w\}_{w \in \{S,R\}}, \textsc{Enc}, \textsc{Dec})$ *and any pair of strategies* (σ_S, σ_R)*,* $(\Pi, \sigma_S, \sigma_R)$ *is not CPA secure with a Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}$*.*

Proof. Suppose that $(\Pi, \sigma_S, \sigma_R)$ is CPA secure with a Nash equilibrium. Consider an adversary A who submits challenge messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$ such that $\boldsymbol{m}_0 = (m, m)$, $\boldsymbol{m}_1 = (m, m')$, $m \neq m'$, and $m, m' \in \mathcal{M}_R \setminus \mathcal{M}_S$. Since any challenge message is not in $\mathcal{M}_S$, the best strategy of the sender for A in the encryption phase is to choose $x^{\mathrm{e}}_S = \mathsf{Bad}$ regardless of the receiver's strategy, which implies that $\sigma_S(pk_S, pk_R, sk_S, aux_S) = \mathsf{Bad}$ with probability at least $1 - \epsilon(k)$ for a negligible function

$\epsilon(\cdot)$. Then, A receives the pair of ciphertexts (c_1, c_2) such that $c_1 = \textsc{Enc}(pk_S, pk_R, sk_S, m; r_1^e)$ and $c_2 = \textsc{Enc}(pk_S, pk_R, sk_S, m^*; r_2^e)$, where m^* is either m or m'. Since A knows $pk_S, pk_R, m, m', r_1^e, r_2^e$, the only information A does not know in c_1 and c_2 is sk_S. It follows from the correctness condition of the encryption scheme that $c_1 = c_2$ if $m^* = m$, and $c_1 \neq c_2$ otherwise. Therefore, A can correctly guess b from c_1 and c_2. This implies that the first condition of the CPA security does not hold. □

C Proof of Theorem 1

Theorem 1. *If Π is CPA secure, $(\Pi_{\mathsf{two}}, \sigma_S, \sigma_R)$ is CPA secure with a strict Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}$*, where (σ_S, σ_R) is defined right after the description of Π_{two}.*

Proof. First we show the correctness of the scheme Π_{two}. Note that $c_1 = \textsc{Enc}(pk_S, r), c_2 = m \oplus \textsc{Dec}(sk_S, c_1)$, and the output of $\textsc{Dec}_R$ is $\hat{m} = c_2 \oplus r$. It follows from the correctness of the underlying scheme Π that $\hat{m} = (m \oplus \textsc{Dec}(sk_S, c_1)) \oplus r = m \oplus r \oplus r = m$.

Next we show that for any PPT adversary A, valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$, after running the game $\mathbf{Game}^{\mathrm{cpa}}$ with a pair of strategies (σ_S, σ_R), we have $\Pr[\mathsf{Win} \cdot (\mathsf{Val}_S + \mathsf{Val}_R) \neq 0] \leq 1/2 + \epsilon(k)$ for some negligible function $\epsilon(\cdot)$. It is sufficient to show that $\Pr[\mathsf{Win} = 1] \leq 1/2 + \epsilon(k)$. In the game $\mathbf{Game}^{\mathrm{cpa}}$ with (σ_S, σ_R), the adversary A needs to guess b from $(pk_S, c_1, c_2, \boldsymbol{m}_0, \boldsymbol{m}_1)$. For any $m_0 \in \boldsymbol{m}_0, m_1 \in \boldsymbol{m}_1$, it follows from the security of the underling scheme $\Pi = (\textsc{Gen}, \textsc{Enc}, \textsc{Dec})$ that $\{pk_S, \textsc{Enc}(pk_S, r), r \oplus m_0, \boldsymbol{m}_0, \boldsymbol{m}_1\} \approx_c \{pk_S, \textsc{Enc}(pk_S, r'), r \oplus m_0, \boldsymbol{m}_0, \boldsymbol{m}_1\} = \{pk_S, \textsc{Enc}(pk_S, r'), r'', \boldsymbol{m}_0, \boldsymbol{m}_1\} = \{pk_S, \textsc{Enc}(pk_S, r'), r \oplus m_1, \boldsymbol{m}_0, \boldsymbol{m}_1\} \approx_c \{pk_S, \textsc{Enc}(pk_S, r), r \oplus m_1, \boldsymbol{m}_0, \boldsymbol{m}_1\}$, where r, r', r'' are independently and uniformly sampled from the message space $\{0,1\}^{\mu}$. This implies that $\Pr[\mathsf{Win} = 1] \leq 1/2 + \epsilon(k)$ for some negligible function $\epsilon(\cdot)$.

Finally we show that (σ_S, σ_R) is a strict Nash equilibrium. It is required to show that (σ_S, σ_R) is a Nash equilibrium. Suppose that the receiver follows σ_R. If $\sigma_S(1^k, \mathcal{M}_R)$ outputs Bad, which increases the utility of the sender by β_S, there is an adversary who can compute sk_S correctly, and thus guess b correctly. If all the challenge messages are in $\mathcal{M}_S$, this reduces the utility of the sender by $\alpha_S/2$. Thus, when the receiver follows σ_R, since any deviation from σ_S reduces the utility by $\alpha_S/2 - \beta_S > 0$, the strategy σ_S maximizes the utility of the sender. Suppose that the sender follows σ_S. If $\sigma_R(pk_S, aux_R)$ outputs Bad, which increases the utility of the receiver by β_R, there is an adversary who computes $m_b = r \oplus c_2$ by using c_2 and $r = r_R^e$. If all the challenge messages are in $\mathcal{M}_R$, this reduces the utility of the receiver by $\alpha_R/2$. Hence, when the sender follows σ_S, since any deviation from σ_R reduces the utility by $\alpha_R/2 - \beta_R > 0$, the strategy σ_R maximizes the utility of the receiver. Therefore, the pair (σ_S, σ_R) is a Nash equilibrium.

To show the second condition of strict Nash equilibrium, consider a strategy σ'_S of the sender such that $\sigma'_S \not\approx \sigma_S$. This implies that $\sigma'_S(1^k, \mathcal{M}_R)$ outputs Bad with probability at least $1/k^c$ for a constant c. By the same argument as above, this reduces the utility of the sender by $(1/k^c) \cdot (\alpha_S/2 - \beta_S)$, namely $U_S(\sigma'_S, \sigma_R) \leq U_S(\sigma_S, \sigma_R) - (\alpha_S/2 - \beta_S)/k^c$. Consider a strategy σ'_R such that $\sigma'_R \not\approx \sigma_R$, which implies that $\sigma'_R(pk_S, aux_R)$ outputs Bad with probability at least $1/k^c$ for a constant c. As above, this reduces the utility of the receiver by $(1/k^c) \cdot (\alpha_R/2 - \beta_R)$, namely $U_R(\sigma_S, \sigma'_R) \leq U_R(\sigma_S, \sigma_R) - (\alpha_R/2 - \beta_R)/k^c$. Therefore (σ_S, σ_R) is a strict Nash equilibrium. □

D Proof of Theorem 2

Theorem 2. *If Π is CPA secure, $(\Pi_{\mathsf{three}}, \sigma_S, \sigma_R)$ is CPA secure with a strict Nash equilibrium for* $\mathbf{Game}_{\mathrm{R}}^{\mathrm{cpa}}$*, where (σ_S, σ_R) is defined right after the description of Π_{three}.*

Proof. First we show the correctness of the scheme Π_{three}. Note that $c_1 = \textsc{Enc}(pk_S, r_1), c_2 = \textsc{Enc}(pk_S, r_2), c_3 = \textsc{Enc}(pk_R, m; r_L)$, and the decrypted message is $\hat{m} = \textsc{Dec}(sk_S, c_3)$. It follows from the correctness of the underlying scheme Π that $\hat{m} = m$.

Next we show that for any PPT adversary A, valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$, after running the game $\mathbf{Game}_{\mathrm{R}}^{\mathrm{cpa}}$ with a pair of strategies (σ_S, σ_R), we have $\Pr[\mathsf{Win} \cdot (\mathsf{Val}_S + \mathsf{Val}_R) \neq 0] \leq 1/2 + \epsilon(k)$ for some negligible function $\epsilon(\cdot)$. Without loss of generality, we assume that $\mathsf{Val}_S + \mathsf{Val}_R \neq 0$. We will show that $\Pr[\mathsf{Win} = 1] \leq 1/2 + \epsilon(k)$. Since $\mathsf{Val}_S + \mathsf{Val}_R \neq 0$ and the players follow (σ_S, σ_R), at least one of x_S^e and x_R^e will be Good. Suppose that $x_S^e = \mathsf{Good}$ and $x_R^e = \mathsf{Bad}$. When A chose m_0, m_1 as the challenge messages, the view of A is

$$\begin{aligned}
&\{pk_S, pk_R, (r_1, r_e), c_1, c_2, c_3, c_4\}\\
&= \{pk_S, pk_R, (r_1, r_e), \textsc{Enc}(pk_S, r_1; r_e), \textsc{Enc}(pk_R, r_2), \textsc{Enc}(pk_R, m_b; r_L), \textsc{Enc}(pk_S, m_b; r_R)\}\\
&\approx_c \{pk_S, pk_R, (r_1, r_e), \textsc{Enc}(pk_S, r_1; r_e), \textsc{Enc}(pk_R, r_2'), \textsc{Enc}(pk_R, m_b; r_L), \textsc{Enc}(pk_S, m_b; r_R)\}\\
&\approx_c \{pk_S, pk_R, (r_1, r_e), \textsc{Enc}(pk_S, r_1; r_e), \textsc{Enc}(pk_R, r_2'), \textsc{Enc}(pk_R, m_{1-b}; r_L), \textsc{Enc}(pk_S, m_{1-b}; r_R)\}\\
&\approx_c \{pk_S, pk_R, (r_1, r_e), \textsc{Enc}(pk_S, r_1; r_e), \textsc{Enc}(pk_R, r_2), \textsc{Enc}(pk_R, m_{1-b}; r_L), \textsc{Enc}(pk_S, m_{1-b}; r_R)\},
\end{aligned}$$

where r_e is the randomness used in computing $c_1 \leftarrow \textsc{Enc}_1(pk_S, r_1)$, and r_1, r_e, r_2, r_2' are uniformly random strings and $r_L \circ r_R = r_1 \oplus r_2$. The above relations follow from the security of the underlying scheme Π. Therefore, in this case, we have that $\Pr[\mathsf{Win} = 1] \leq 1/2 + \epsilon(k)$. The proof of the case that $x_S^e = \mathsf{Bad}$ and $x_R^e = \mathsf{Good}$ can be done in a similar way.

Finally we show that the pair of strategies (σ_S, σ_R) is a strict Nash equilibrium. Suppose that the receiver follows σ_R. Consider any strategy σ_S' of the sender, and an adversary who set $aux_R = \mathsf{Val}_R$ and submits the challenge messages such that all of them are in $\mathcal{M}_S \setminus \mathcal{M}_R$. If $\sigma_S'(1^k, \mathcal{M}_S)$ outputs Bad, which increases the utility of the sender by β_S, the adversary can compute sk_S correctly, and thus can guess b correctly from $c_4 = \textsc{Enc}(pk_S, m)$. If $\sigma_S'(pk_S, pk_R, sk_S, aux_S)$ outputs Bad, which also increases the utility of the sender by β_S, since the receiver chooses Bad in the encryption phase, the adversary can compute $r_1 \oplus r_2$ correctly, and guess b correctly from $c_3 = \textsc{Enc}(pk_R, m; r_L)$, where $r_1 \oplus r_2 = r_L \circ r_R$. Since the adversary can guess b correctly in both cases, the utility of the sender decreases by at least $\alpha_S/2 - 2\beta_S > 0$ if the sender deviated from σ_S. This implies that the strategy σ_S maximizes the utility of the sender if the receiver follows σ_R. Next suppose that the receiver follows σ_S. Consider any strategy σ_R' of the receiver, and an adversary who submits the challenge messages such that all of them are in $\mathcal{M}_R \setminus \mathcal{M}_S$. If $\sigma_R'(1^k, \mathcal{M}_R)$ outputs Bad, which increases the utility of the receiver by β_R, the adversary can compute sk_R correctly, and thus can guess b correctly from $c_3 = \textsc{Enc}(pk_R, m)$. If $\sigma_R'(pk_S, pk_R, sk_R, aux_R)$ outputs Bad, which increases the utility of the receiver by β_R, since the sender chooses Bad in the encryption phase, the adversary can compute $r_1 \oplus r_2$ correctly, and guess b correctly from $c_4 = \textsc{Enc}(pk_R, m; r_R)$, where $r_1 \oplus r_2 = r_L \circ r_R$. Since the adversary can guess b correctly in both cases, the utility of the receiver decreases by at least $\alpha_R/2 - 2\beta_R > 0$ if the receiver deviated from σ_S. This implies that the strategy σ_R maximizes the utility of the receiver if the sender follows σ_S. Therefore, (σ_S, σ_R) is a Nash equilibrium.

To show the second condition of strict Nash equilibrium, consider any strategy σ_S' of the sender such that $\sigma_S' \not\approx \sigma_S$. This implies that, if $aux_R = \mathsf{Val}_R$ and all the challenge messages are in

$\mathcal{M}_S \setminus \mathcal{M}_R$, either $\sigma'_S(1^k, \mathcal{M}_S)$ or $\sigma'_S(pk_S, pk_R, sk_S, aux_S)$ outputs Bad with probability at least $1/k^c$ for a constant c. By the same argument as above, this reduces the utility of the sender by $(1/k^c) \cdot (\alpha_S/2 - 2\beta_S)$, namely $U_S(\sigma'_S, \sigma_R) \leq U_S(\sigma_S, \sigma_R) - (\alpha_S/2 - 2\beta_S)/k^c$. Consider any strategy σ'_R of the receiver such that $\sigma'_R \not\approx \sigma_S$, which implies that, if all the challenge messages are in $\mathcal{M}_R \setminus \mathcal{M}_S$, either $\sigma'_R(1^k, \mathcal{M}_R)$ or $\sigma'_R(pk_S, pk_R, sk_R, aux_R)$ outputs Bad with probability at least $1/k^c$ for a constant c. As above, this implies that $U_R(\sigma_S, \sigma'_R) \leq U_R(\sigma_S, \sigma_R) - (\alpha_R/2 - 2\beta_R)/k^c$. Therefore, the pair of strategy (σ_S, σ_R) is a strict Nash equilibrium. □

E Proof of Theorem 3

Theorem 3. *Let* $\Pi_{\mathsf{sigenc}} = (\{\textsc{Gen}_w\}_{w \in \{S,R\}}, \textsc{SigEnc}, \textsc{VerDec})$ *be a signcryption scheme with CPA security and key-exposure property. Then* $(\Pi_{\mathsf{sigenc}}, \sigma_S, \sigma_R)$ *is CPA secure with a strict Nash equilibrium for the game* $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{secret}}$.

Proof. The first condition of the CPA security follows from the CPA security of Π_{sigenc}. Hence we show the second condition, that is (σ_S, σ_R) is a strict Nash equilibrium for $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{secret}}$.

Suppose that the receiver follows σ_R. Consider any strategy σ'_S of the sender and an adversary. If $\sigma'_S(1^k, \mathcal{M}_S)$ outputs Bad, which increases the utility of the sender by β_S, the adversary can compute sk_S correctly. If $\sigma'_S(pk_S, pk_R, sk_S, aux_S)$ outputs Bad, which increases the utility of the sender by β_S, since the adversary knows the random string r^e of the ciphertext $c = \textsc{SigEnc}(pk_R, sk_S, m_b; r^e)$, she can compute sk_S by the key-exposure property of Π_{sigenc}. Thus the strategy σ_S maximizes the utility of the sender if the receiver follows σ_R. Suppose that the sender follows σ_S. Consider any strategy σ'_R of the receiver and an adversary who submits the challenge messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$ such that $\boldsymbol{m}_0 = (m, m), \boldsymbol{m}_1 = (m, m'), m \neq m'$, and $m, m' \in \mathcal{M}_R$. If $\sigma'_R(1^k, \mathcal{M}_S)$ outputs Bad, which increases the utility of the sender by β_R, the adversary can compute sk_R, and thus guess b correctly by computing $\textsc{Dec}(pk_S, sk_R, c)$. Thus the strategy σ_R maximizes the utility of the receiver if the sender follows σ_S. Therefore, (σ_S, σ_R) is a Nash equilibrium. To show the second condition of strict Nash equilibrium, consider any strategy σ'_S of the sender such that $\sigma'_S \not\approx \sigma_S$. This implies that either $\sigma'_S(1^k, \mathcal{M}_S)$ or $\sigma'_S(pk_S, pk_R, sk_S, aux_S)$ outputs Bad with probability at least $1/k^c$ for a constant c. By the same argument above, this reduces the utility of the sender by at least $(1/k^c) \cdot (\gamma_S - \alpha_S/2 - 2\beta_S)$. Next consider any σ'_R of the receiver such that $\sigma'_R \not\approx \sigma_R$. This implies that $\sigma'_R(1^k, \mathcal{M}_S)$ outputs Bad with probability at least $1/k^c$ for a constant c. As above, this reduces the utility of the receiver by at least $(1/k^c) \cdot (\alpha_R/2 - \beta_R)$. Therefore, (σ_S, σ_R) is a strict Nash equilibrium. □

F Proofs of Propositions

Proposition 1. *For any pair of strategies* (σ_S, σ_R)*,* $(\Pi_{\mathsf{two}}, \sigma_S, \sigma_R)$ *is not CPA secure with a Nash equilibrium for* $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{R}}$.

Proof. Suppose that $(\Pi_{\mathsf{two}}, \sigma_S, \sigma_R)$ is CPAsecure with a Nash equilibrium. Consider an adversary A who sets $aux_R = \mathsf{Val}_R$ and submits challenge messages $(\boldsymbol{m}_0, \boldsymbol{m}_1)$ such that $\boldsymbol{m}_0 = m_0$, $\boldsymbol{m}_1 = m_1$, $m_0 \neq m_1$, and $m_0, m_1 \in \mathcal{M}_S \setminus \mathcal{M}_R$. The best strategy of the receiver for A is to choose $x^{\mathrm{e}}_R = \mathsf{Bad}$ regardless of the sender's strategy. Therefore, $\sigma_S(pk_S, aux_R) = \mathsf{Bad}$ with probability at least $1 - \epsilon(k)$, where $\epsilon(\cdot)$ is a negligible function. Since A knows the random string r for encryption, she can correctly guess b by computing $m_b = c_2 \oplus r$. This implies that the first condition of the CPA security does not hold. $\square$

Proposition 2. *There are two pairs of strategies (σ_S, σ_R) and (ρ_S, ρ_R) such that $\sigma_S \not\approx \rho_S$, $\sigma_R \not\approx \rho_R$, and both $(\Pi_{\mathsf{three}}, \sigma_S, \sigma_R)$ and $(\Pi_{\mathsf{three}}, \rho_S, \rho_R)$ are CPA secure with strict Nash equilibrium for $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{S,R}}$. Furthermore, there is a PPT adversary A and valuable message spaces $\mathcal{M}_S$ and $\mathcal{M}_R$ such that $\mathbf{E}[u_S(\mathsf{Out}_\rho)] - \mathbf{E}[u_S(\mathsf{Out}_\sigma)] \geq \beta_S - \epsilon(k)$ and $\mathbf{E}[u_R(\mathsf{Out}_\sigma)] - \mathbf{E}[u_R(\mathsf{Out}_\rho)] \geq \beta_R - \epsilon(k)$ for every sufficiently large k, where Out_σ is the outcome of the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{S,R}}$ in which players follow (σ_S, σ_R), Out_ρ is the outcome of the game $\mathbf{Game}^{\mathrm{cpa}}_{\mathrm{S,R}}$ in which players follow (ρ_S, ρ_R), and $\epsilon(\cdot)$ is a negligible function.*

Proof. We define (σ_S, σ_R) and (ρ_S, ρ_R) as follows.

- $\sigma_S(1^k, \mathcal{M}_S)$ outputs Good with probability 1. $\sigma_S(pk_S, pk_R, sk_S, aux_S)$ outputs Good if $m_{b,i} \in \mathcal{M}_S$ for some $i \in \{1, \ldots, \ell\}$, and Bad otherwise.
- $\sigma_R(1^k, \mathcal{M}_R)$ outputs Good with probability 1. $\sigma_R(pk_S, pk_R, sk_R, aux_R)$ outputs Good if
  - $aux_R = \perp$,
  - $aux_R = \mathsf{Val}_S$ and $\mathsf{Val}_S = 0$,
  - $aux_R = \mathsf{Val}_R$ and $\mathsf{Val}_R = 1$, or
  - $aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$, $\mathsf{Val}_S = 0$, and $\mathsf{Val}_R = 1$,

  and Bad otherwise.
- $\rho_S(1^k, \mathcal{M}_S)$ outputs Good with probability 1. $\rho_S(pk_S, pk_R, sk_S, aux_S)$ outputs Good if
  - $aux_S = \boldsymbol{m}_b$ and $m_{b,i} \in \mathcal{M}_S$ for some $i \in \{1, \ldots, \ell\}$, or
  - $aux_S = (\boldsymbol{m}_b, \mathsf{Val}_R)$, $m_{b,i} \in \mathcal{M}_S$ for some $i \in \{1, \ldots, \ell\}$, and $\mathsf{Val}_R = 0$,

  and Bad otherwise.
- $\rho_R(1^k, \mathcal{M}_R)$ outputs Good with probability 1. $\rho_R(pk_S, pk_R, sk_R, aux_R)$ outputs Good if
  - $aux_R = \perp$,
  - $aux_R = \mathsf{Val}_S$,
  - $aux_R = \mathsf{Val}_R$ and $\mathsf{Val}_R = 1$, or
  - $aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$ and $\mathsf{Val}_R = 1$,

  and Bad otherwise.

The difference between outputs of (σ_S, σ_R) and (ρ_S, ρ_R) is only in the case that $aux_S = (\boldsymbol{m}_b, \mathsf{Val}_R)$, $aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$, and $\mathsf{Val}_S = \mathsf{Val}_R = 1$. In this case, the sender uses good randomness and the receiver uses bad randomness in (σ_S, σ_R), while the sender uses bad randomness and the receiver uses good randomness in (ρ_S, ρ_R). Hence we have that $\sigma_S \not\approx \rho_S$ and $\sigma_R \not\approx \rho_R$. In the proof of Theorem 2, we show that, if at least one of x^{e}_S and x^{e}_R is Good, Π_{three} satisfies the first condition of the CPA security. Thus, we can verify that both $(\Pi_{\mathsf{three}}, \sigma_S, \sigma_R)$ and $(\Pi_{\mathsf{three}}, \rho_S, \rho_R)$ satisfy the first condition of the CPA security.

Consider an adversary who sets $aux_S = (\boldsymbol{m}_b, \mathsf{Val}_R)$ and $aux_R = (\mathsf{Val}_S, \mathsf{Val}_R)$, and submits the challenge messages such that all of them are in $\mathcal{M}_S \cap \mathcal{M}_R$. For this adversary, $\sigma_S(pk_S, pk_R, sk_S, aux_S)$ outputs Good and $\sigma_R(pk_S, pk_R, sk_R, aux_R)$ outputs Bad, while

$\rho_S(pk_S, pk_R, sk_S, aux_S)$ outputs Bad and $\rho_R(pk_S, pk_R, sk_R, aux_R)$ outputs Good. Since it follows from the above argument that the expected value of $\mathsf{Win} \cdot \mathsf{Val}_w$ is at most $1/2 + \epsilon(k)$ for a negligible function $\epsilon(\cdot)$, we have that $\mathbf{E}[u_S(\mathsf{Out}_\rho)] - \mathbf{E}[u_S(\mathsf{Out}_\sigma)] \geq \beta_S - \epsilon'(k)$ and $\mathbf{E}[u_R(\mathsf{Out}_\sigma)] - \mathbf{E}[u_R(\mathsf{Out}_\rho)] \geq \beta_R - \epsilon'(k)$ for a negligible function $\epsilon'(\cdot)$.

We show that (σ_S, σ_R) is a strict Nash equilibria. We follow the same reasoning as the proof of Theorem 2. It is sufficient to show that, for each $w \in \{S, R\}$, if player w follows a different strategy σ'_w from σ_w, then the utility of player w decreases by some constant value. We show that if σ'_w outputs Bad in the case that σ_w outputs Good, there exists an adversary who can guess b correctly, which decreases the utility of player w by at least $\alpha_w/2 - 2\beta_w > 0$. First note that, for each $w \in \{S, R\}$, if $\sigma'_w(1^k, \mathcal{M}_w)$ outputs Bad, the adversary can guess b correctly by the same argument as the proof of Theorem 2. Suppose that $\sigma'_S(pk_S, pk_R, sk_S, aux_S)$ outputs Bad in the case that $\sigma_S(pk_S, pk_R, sk_S, aux_S) = \mathsf{Good}$. Consider an adversary who sets $aux_R = \mathsf{Val}_R$ and submits the challenge messages such that all of them are in $\mathcal{M}_S \setminus \mathcal{M}_R$. Since the receiver chooses $x^e_R = \mathsf{Bad}$ for this adversary, the adversary can guess b correctly from $r_1 \oplus r_2$ and $c_3 = \textsc{Enc}(pk_R, m; r_L)$, where $r_1 \oplus r_2 = r_L \circ r_R$. Suppose that $\sigma'_R(pk_S, pk_R, sk_S, aux_R)$ outputs Bad in the case that $\sigma_R(pk_S, pk_R, sk_S, aux_R)$ outputs Good. Consider an adversary who submits the challenge messages such that all of them are in $\mathcal{M}_R \setminus \mathcal{M}_S$. Since the sender chooses $x^e_S = \mathsf{Bad}$ for this adversary, the adversary can guess b correctly from $r_1 \oplus r_2$ and $c_4 = \textsc{Enc}(pk_R, m; r_R)$, where $r_1 \oplus r_2 = r_L \circ r_R$. Therefore, by the same reasoning as the proof of Theorem 2, (σ_S, σ_R) is a strict Nash equilibrium. By the same argument, we can show that (ρ_S, ρ_R) is also a strict Nash equilibrium. $\square$