Plaintext-Simulatability

Eiichiro Fujisaki
fujisaki@isl.ntt.co.jp

NTT Laboratories, Japan

Abstract. We propose a new security class, called plaintext-simulatability, defined over the public-key encryption schemes. The notion of plaintext simulatability (denoted PS) is similar to the notion of plaintext awareness (denoted PA) [2], but it is, "properly", a weaker security class for public-key encryption. It is known that PA implies the class of CCA2-secure encryption (denoted IND-CCA2) but not vice versa. In most cases, PA is "unnecessarily" strong — In such cases, PA is only used to prove that a public-key encryption scheme is CCA2-secure, because it looks much easier than to prove "directly" that the scheme meets IND-CCA2. We show that PS also implies IND-CCA2, while preserving a good view of the security proofs as well as PA. Recently, a couple of schemes [9, 15, 1] have been proposed, that have been proven to be CCA2 secure in the random oracle model but they lie outside PA. However, they have been "heuristically" proven and so there is no general observation that looks over those schemes and extracts the essence of those security proofs.
Not only does PS provide a good perspective of the security proof for an actual encryption scheme, but it is also more desirable from a theoretical viewpoint, because PA is obviously associated with non-interactive zero-knowledge proofs of knowledge [20] that is achieved under the existence of dense secure public-key cryptosystems [19] which seem to be stronger than the general assumptions, whereas PS is associated with non-interactive zero-knowledge proofs for membership of a language that can be achieved just under the general assumption that trap-door functions exist.
Finally, we suggest a few interesting encryption schemes. One is a random-oracle version of Dolev-Dwork-Naor's encryption scheme [6, 7]. Unlike the original scheme, this construction is efficient. The other is a public-key encryption scheme based on a strong pseudo-random permutation family [12] which provides the optimal ciphertext lengths for verifying the validity of ciphertexts, i.e., (ciphertext size) = (message size) + (randomness size). According to [15], such a construction remains open. Both schemes meet PS but not PA.

Plaintext-simulatability, Plaintext-awareness, Chosen-ciphertext security (CCA2-security), Dolev-Dwork-Naor's encryption scheme, CCA2-secure encryption scheme without overhead.

1 Introduction

Plaintext-awareness (denoted PA) is a security class for public-key encryption (mostly defined in the random oracle model), which implies the class of the CCA2-secure encryption, denoted IND-CCA2 [17, 2]. The notion of PA was suggested in [3] and later formalized in [2].

The key property of the notion of plaintext-awareness is, roughly said, that nobody can produce a *new* ciphertext without *knowing* the plaintext. Informally, we say that a public-key encryption scheme is *plaintext aware* if it is secure (indistinguishable) against chosen-plaintext attacks (IND-CPA), in addition to satisfying the above property. PA implies IND-CCA2 [2] — Intuitively, it comes from the following idea: If the target encryption scheme is plaintext aware, an adversary *is aware of* the decryption of the ciphertexts he has submitted to the decryption oracle. Hence, the adversary cannot get any additional information from

the decryption oracle because he already *knows* the corresponding plaintexts. Therefore, we can transform an adaptive chosen-ciphertext attack against the target encryption scheme into a chosen-plaintext attack against the same encryption scheme. Hence, if the scheme is secure in the IND-CPA sense, it would be also secure in the IND-CCA2 sense.

The opposite direction does not hold — IND-CCA2 does not imply PA. Paper [2] provides an "artificial" counter-example to prove this fact, whereas we can suggest a few "natural" counter-examples later.

In most cases, PA is merely a "means" rather than a "goal" — a tool to prove that a public-key encryption scheme is CCA2-secure, because it looks much easier to prove that a scheme meets PA than to prove "directly" that it meets IND-CCA2. So far, only an exception that requires the full power of PA is known in the literature [11], but the application makes sense only in a restricted computational model, so called the Dolev-Yao model [8]. In the ordinary adversary model for cryptographic protocols, no application using the full-advantage of PA has been published. Namely, PA is "unnecessarily" strong in the most cases. So far, a large number of generic methods for constructing CCA2-secure encryption schemes in the random oracle model have been proposed and it is true that most of them belong to the class of PA. It could be, however, explained by the fact that PA is the only criterion to "systematically" create a CCA2-secure encryption scheme in the random oracle model. Meanwhile, a few schemes proposed recently such as [9, 15, 1] are CCA2-secure (in the random oracle model) but not PA, though the authors do not mention that fact. Security for those schemes was proven "heuristically" in each proposal. In other words, there is no observation that looks over those schemes and extracts the essence of those security proofs.

The notion of PA might be possibly too strong from another viewpoint. Obviously PA has a strong analogy with the notion of CPA security (semantic security) coupled with a non-interactive zero-knowledge proof system of knowledge of the plaintext, suggested in [4, 20] [1]. The existence of a NIZKP system of knowledge for a NP relation is assured under the existence of a dense secure public-key cryptosystem [19] which has not been proven weaker than or equivalent to the general assumption (that trap-door functions exist). Therefore, the previous CCA2-secure encryption schemes such as [7, 18, 14], which stand in the standard model under the general assumption (that trap-door functions exist), do not depend on NIZKP systems of knowledge, but on NIZKP systems for membership of languages (because they exist if trap-door functions exist). Here the point is the following: Suppose that one suggests a CCA2-secure encryption scheme in the random oracle model under the general assumptions. He must hope that someday he or someone would find a transformation of his scheme into the one in the standard model, while preserving its security. However, it seems hard to modify a secure one if it is a PA encryption scheme, because of the close relation between PA and NIZKP systems of knowledge.

1.1 Our results

In this paper, we propose a weaker but still enough strong security class, called "plaintext-simulatability" (denoted PS). The difference of PS from PA is "subtle" in terms of definition, but PS is "properly" weaker than PA and stronger than IND-CCA2. As well as PA, PS can also give the designers of the schemes a better view of the security proofs: Namely, one can

[1] The subtle difference between them is that in PA the (knowledge) extractor cannot take (part of) a public key under its control to play with it.

treat "independently" or "orthogonally" the indistinguishability of the encryption and the simulation of decryption.

Technically speaking, if the simulation of decryption is perfect or statistically close to real decryption, the proof that PS implies IND-CCA2 is a natural extention of the proof that PA implies IND-CCA2 in [2]. However, if the simulation of decryption is only computationally indistinguishable from the real decryption, the proof is more involved.

We present a few "natural" examples that lie in the gap between PA and PS. It also means that they lie in the gap between PA and IND-CCA2, because PS implies IND-CCA2. In [2], they contrived an "artificial" encryption scheme to show that PA is "properly" stronger than IND-CCA2, whereas ours are more natural examples in the gap.

The PS encryption schemes in Sec. 6 are of independent interest. One is an encryption scheme "interpreted in the random oracle model" from Dolev-Dwork-Naor's encryption scheme [6, 7]. One is a public-key encryption scheme that provides the optimal ciphertext lengths for verifying validity of ciphertexts, i.e., (ciphertext size) = (message size) + (randomness size).

PS looks "properly" stronger than IND-CCA2. So far, however, it is not sure how to prove this, and so this remains open.

2 Preliminary

We begin with some notations.

We write $x := a$ to denote the operation of assigning the value of a to the variable x. Let X be a probability space on some finite set $\mathcal{S}(\subset \{0,1\}^*)$. We denote by $x \leftarrow X$ the operation of sampling an element of $\mathcal{S}$ according to the distribution of X, and assigning the result of this experiment to the variable x. We also write, for some finite set S', $x \leftarrow_R S'$ to denote the operation of sampling an element of S' uniformly, and assigning the result of this experiment to the variable x.

For probability spaces, $X_1, \ldots, X_k$, and k-ary predicate ϕ, we write $\Pr[x_1 \leftarrow X_1; x_2 \leftarrow X_2; \cdots : \phi(x_1, \ldots, x_k)]$ to denote the probability that the predicate $\phi(x_1, \ldots, x_k)$ is true after the experiments, "$x_1 \leftarrow X_1; x_2 \leftarrow X_2; \cdots$", are executed in that order. In this case, it is important that $x_1, \ldots, x_k$ are sampled in that order.

Let $\epsilon, \tau : \mathbb{N} \to [0,1](\subset \mathbb{R})$ be positive $[0,1]$-valued functions. We say that $\epsilon(k)$ is negligible in k if, for any constant c, there exists a constant, $k_0 \in \mathbb{N}$, such that $\epsilon(k) < (1/k)^c$ for any $k > k_0$. We say that $\tau(k)$ is overwhelming in k if $\epsilon(k) \triangleq 1 - \tau(k)$ is negligible in k. Let $f, g : \mathbb{N} \to \mathbb{R}$. We write $f(k) \in O(g(k))$ to denote that there is a constant $c > 0$ such that $f(k) \leq cg(k)$ for every sufficiently large k. We write $g(k) \in \Omega(f(k))$ to denote that there is a constant $c > 0$ such that $g(k) \geq cf(k)$ for every sufficiently large k.

2.1 Syntax of encryption schemes

A public-key encryption scheme is given by a triple of algorithm, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where, for every sufficiently large $k \in \mathbb{N}$,

- $\mathcal{K}$, the key-generation algorithm, is a probabilistic polynomial-time algorithm which on input 1^k outputs a pair of strings, $(pk, sk) \in \mathtt{PK} \times \mathtt{SK}$, where $\mathtt{PK} \times \mathtt{SK}$ is the product set of all possible corresponding public and secret keys generated by applying 1^k to $\mathcal{K}$.
  This experiment is written as $(pk, sk) \leftarrow \mathcal{K}(1^k)$.
  For given pk (and possibly k), the message and coin spaces, $\mathtt{MSP}$ and $\mathtt{COIN}$, of Π are uniquely determined.

– $\mathcal{E}$, the encryption algorithm, is a probabilistic polynomial-time algorithm that takes a public key $pk \in \mathtt{PK}$ and a message $x \in \mathtt{MSP}$, draws a string r uniformly from the coin space $\mathtt{COIN}$, and produces a string $y := \mathcal{E}_{pk}(x; r)$.
  This experiment is written as $y \leftarrow \mathcal{E}_{pk}(x)$.
– $\mathcal{D}$, the decryption algorithm, is a deterministic polynomial-time algorithm that takes a secret key $sk \in \mathtt{SK}$ and a string $y \in \{0,1\}^*$, and returns a string $x := \mathcal{D}_{sk}(y)$.

We further require that a public-key encryption scheme should be *complete* in the following sense: For every sufficiently large $k \in \mathbb{N}$ that Π can be defined on, it always holds that $\mathcal{D}_{sk}(\mathcal{E}_{pk}(x)) = x$, for every $(pk, sk) \in \mathtt{PK} \times \mathtt{SK}$ and every $x \in \mathtt{MSP}$. A string pk is called *valid* for Π with k, if $pk \in \mathtt{PK}$ for k.

2.2 CPA/CCA2-Security

We briefly recall the security notions for public-key encryption, called CPA-security (IND-CPA) and CCA2-security (IND-CCA2), following [17, 2].

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. For Π, we consider the following games, CPA and CCA2 games. In both games, adversary A takes two modes, "find" and "guess". A always starts with the find mode and then takes the guess mode. A start by taking pk and ends up the find mode by outputting two messages, $x_0, x_1 \in \mathtt{MSP}$. A takes y^* and enters the guess mode, where b is a random challenge bit and y^* is computed as $y^* \leftarrow \mathcal{E}_{pk}(x_b)$. In the CCA-2 game, at both modes, A can make access to the decryption oracle, $\mathcal{D}_{sk}(\cdot)$, at any time with any sequence of queries for decryption, whereas, in the CPA game, A cannot make access to the decryption oracle. The only restriction is that she cannot query the oracle on the challenge ciphertext y^* in the guess mode. Finally she ends up the guess mode by outputs b'. The advantage of A indicates how much better she can guess the value b than $\frac{1}{2}$, namely $2\Pr[b = b'] - 1$.

The random oracle version of this security notion is defined by allowing A to query a random oracle (s). We define by $\mathtt{Hash}$ a family of all the maps from an appropriate domain to an appropriate range. The domain and range depend on the underlying encryption scheme. For simplicity, even if we draw different random functions, G and H, from different function families, $\mathtt{Hash}_G$ and $\mathtt{Hash}_H$, we just write $G, H \leftarrow_R \mathtt{Hash}$ to denote the experiment.

In the following, we formally define these security notions in the random oracle model.

Definition 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let A be an adversary for Π. For $k \in \mathbb{N}$, denote the success event of A for Π by*

$$\mathrm{Succ}^{\mathrm{atk}}_{A,\Pi}(k) \triangleq H \leftarrow_R \mathtt{Hash};\ (pk, sk) \leftarrow \mathcal{K}(1^k); (x_0, x_1) \leftarrow A^{H,\mathcal{O}}(\mathrm{find}, pk);$$
$$b \leftarrow_R \{0,1\};\ y^* \leftarrow \mathcal{E}^H_{pk}(x_b) : A^{H,\mathcal{O}}(\mathrm{guess}, y^*) = b,$$

where $\mathcal{O} = \varepsilon$ (no oracle access) if $\mathrm{atk} = \mathrm{cpa}$; $\mathcal{O} = \mathcal{D}_{sk}(\cdot)$ if $\mathrm{atk} = \mathrm{cca2}$. We then define the advantage of A for Π as

$$\mathrm{Adv}^{\mathrm{atk}}_{A,\Pi}(k) \triangleq 2 \cdot \Pr[\mathrm{Succ}^{\mathrm{atk}}_{A,\Pi}(k)] - 1.$$

We say that Π is secure in the IND-ATK sense (or ATK-secure for short) if, for every polynomial-time (in k) adversary A, $\mathrm{Adv}^{\mathrm{atk}}_{A,\Pi}(k)$ is negligible in k, where $\mathrm{atk} = \{\mathrm{cpa}, \mathrm{cca2}\}$.

2.3 Plaintext Awareness

For completeness, we recall the definition of plaintext awareness, following [2]. Plaintext-awareness is defined in the random oracle model.

A knowledge extractor for a public-key encryption scheme is defined as follows. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let B and K be algorithms, called an adversary and a knowledge extractor, respectively. We describe their specifications here:

- B on input pk makes access to random oracle H and encryption oracle $\mathcal{E}_{pk}^{H}$, and, after that, finally outputs string $y \notin \mathcal{Y}$, where
  - $\mathcal{T}_H$ denotes the entire interaction between B and H, and
  - $\mathcal{Y}$ denotes the set of all "answers" made by $\mathcal{E}_{pk}^{H}(\cdot)$.

  The experiment that we get $(\mathcal{T}_H, \mathcal{Y}, y)$ by running B with pk is written as $(\mathcal{T}_H, \mathcal{Y}, y) \leftarrow \mathtt{run} B^{H, \mathcal{E}_{pk}^{H}}(pk)$. Here we insist the following:
  - The queries of B to $\mathcal{E}_{pk}^{H}$ are not included in $\mathcal{Y}$, and
  - The interaction between $\mathcal{E}_{pk}^{H}$ and H is not included in $\mathcal{T}_H$.
- K on input $(\mathcal{T}_H, \mathcal{Y}, y, pk)$ outputs string x. Here we insist that K is not allowed to access H nor $\mathcal{E}_{pk}^{H}$, i.e., K must output the decryption of y without any help of the oracles.

Definition 2. [Knowledge Extractor] [2] *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme, let B and K be algorithms. For $k \in \mathbb{N}$, define the success event of K for Π and B, $\mathrm{Succ}^{\mathrm{ke}}_{K,B,\Pi}(k)$, as*

$$H \leftarrow_R \mathtt{Hash}; (pk, sk) \leftarrow \mathcal{K}(1^k); (\mathcal{T}_H, \mathcal{Y}, y) \leftarrow \mathtt{run} B^{H, \mathcal{E}_{pk}^{H}}(pk) : K(\mathcal{T}_H, \mathcal{Y}, y, pk) = \mathcal{D}_{sk}^{H}(y).$$

We then denote the advantage of K for Π and B by

$$\mathrm{Adv}^{\mathrm{ke}}_{K,B,\Pi}(k) \triangleq \Pr[\mathrm{Succ}^{\mathrm{ke}}_{K,B,\Pi}(k)].$$

We say that K is a knowledge extractor for Π if, for every polynomial-time algorithm B, K runs in polynomial-time and $\mathrm{Adv}^{\mathrm{ke}}_{K,B,\Pi}(k)$ is overwhelming in k.

This formal definition of the knowledge extractor yields the security class of plaintext-awareness.

Definition 3. [Plaintext Awareness] [2] *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. We say that Π is secure in the sense of PA if Π is secure in the sense of IND-CPA and there is a knowledge extractor for Π.*

3 Plaintext Simulatability

We introduce the security notion of plaintext simulatability (denoted PS). First, we introduce an algorithm, called the knowledge simulator, which is similar to the knowledge-extractor. The notable differences from the knowledge extractor are twofold:

- The knowledge simulator is allowed to **make access to the random oracle** H to produce the decryption of given ciphertext.
- The output of the knowledge simulator is needed to be **computationally indistinguishable** from the real decryption of given ciphertext (over the random choice of H) against any distinguisher algorithm (even given the corresponding **secret key**).

Reminder: In the notion of PA, the knowledge extractor is not allowed to make access to the random oracle and its output must be the same as the real decryption of the given ciphertext.

A knowledge simulator for a public-key encryption scheme is defined as follows. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let B, K and C be algorithms, called an adversary, a knowledge simulator, and a distinguisher, respectively. We describe their specifications here:

- B on input pk *makes access to random oracle* H and encryption oracle $\mathcal{E}_{pk}^{H}$, and, after that, finally outputs string $y \notin \mathcal{Y}$, where
  - $\mathcal{T}_H$ denotes the entire interaction between B and H, and
  - $\mathcal{Y}$ denotes the set of all "answers" made by $\mathcal{E}_{pk}^{H}(\cdot)$.

  The experiment that we get $(\mathcal{T}_H, \mathcal{Y}, y)$ by running B with pk is written as

$$(\mathcal{T}_H, \mathcal{Y}, y) \leftarrow \mathtt{run} B^{H, \mathcal{E}_{pk}^{H}}(pk).$$

  Here we insist that the interaction between $\mathcal{E}_{pk}^{H}$ and H is not included in $\mathcal{T}_H$.
- K, on input $(\mathcal{T}_H, \mathcal{Y}, y, pk)$, *makes access to random oracle* H, and finally outputs string x.

  Informally, algorithm K is a knowledge simulator if the output of K as given above is (computationally) indistinguishable from the "real" decryption, over the random choice of H.
- C, on input $(\mathcal{T}_H, \mathcal{Y}, y, pk, sk, X)$, outputs just one bit, where X denotes a random string of the size of the plaintext, possibly representing $K^H(\mathcal{T}_H, \mathcal{Y}, y, pk)$ or $\mathcal{D}_{sk}^{H}(y)$.

We formalize this notion as follows.

Definition 4. [Knowledge Simulator] *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a public-key encryption scheme. Let B, K and C be algorithms specified above. For $k \in \mathbb{N}$, denote the success event of K for Π, B and C by*

$$\mathrm{Succ}^{\mathrm{ks}}_{K,B,C,\Pi}(k) \triangleq H \leftarrow_R \mathtt{Hash}; (pk, sk) \leftarrow \mathcal{K}(1^k); (\mathcal{T}_H, \mathcal{Y}, y) \leftarrow \mathtt{run} B^{H, \mathcal{E}_{pk}^{H}}(pk) : C\Big(\mathcal{T}_H, \mathcal{Y}, y, pk, sk, K^H(\mathcal{T}_H, \mathcal{Y}, y, pk)\Big) = C(\mathcal{T}_H, \mathcal{Y}, y, pk, sk, \mathcal{D}_{sk}^{H}(y)).$$

For $k \in \mathbb{N}$, denote the advantage of K for Π, B and C by

$$\mathrm{Adv}^{\mathrm{ks}}_{K,B,C,\Pi}(k) \triangleq \Pr[\mathrm{Succ}^{\mathrm{ks}}_{K,B,C,\Pi}(k)].$$

We say that K is a **knowledge simulator** *for Π if, for all polynomial-time (in k) algorithms, B and C, K runs in polynomial-time in k and $\mathrm{Adv}^{\mathrm{ks}}_{K,B,C,\Pi}(k)$ is overwhelming in k.*

Remark 1. The reader might think that to give the distinguisher C the private key sk looks a little bit tricky, but it is definitely necessary. If C is not given sk, C has no advantage over adversary A. Then for any query y whose plaintext is not open via the interaction between A and H, C cannot distinguish the decryption of y from any garbage message (if Π is at least IND-CPA secure). Indeed, sk is essential in the proof of Lemma 2, which states that PS implies IND-CCA2.

Definition 5. *We say that* $\epsilon^{\mathrm{ks}}(\cdot)$ *is the* **knowledge simulation error function** *for* Π *if there is a knowledge simulator* K *for* Π *and, for all polynomial time (in* k*) algorithms,* B *and* C*,* $\mathrm{Adv}^{\mathrm{ks}}_{K,B,C,\Pi}(k) \geq 1 - \epsilon^{\mathrm{ks}}(k)$.

The advantage of distinguisher C is upper-bounded by the knowledge simulation error function for Π. To prove this, we define some notations. For probability space X and distinguisher C, let us define $p^C_X(b) \triangleq \Pr[C(X) = b]$. For Π, B, C, and K mentioned above, we define $\mathrm{Dist}_{B,C,\Pi}(K^H, \mathcal{D}^H_{sk}) \triangleq |p^C_X(1) - p^C_Y(1)|$, where X denotes the probability space specified by the sequence of the random variables,

$$< \mathcal{T}_H, \mathcal{Y}, y, pk, sk, K^H(\mathcal{T}_H, \mathcal{Y}, y, pk) >,$$

and Y denotes the probability space specified by the sequence of the random variables,

$$< \mathcal{T}_H, \mathcal{Y}, y, pk, sk, D^H_{sk}(y) > .$$

Lemma 1. *Let* K *be a knowledge simulator for* Π*. Let* $\epsilon^{\mathrm{ks}}(\cdot)$ *be a knowledge simulation error function for* Π*. Then, for every polynomial-time algorithms* B *and* C*,*

$$\mathrm{Dist}_{B,D,\Pi}(K^H, \mathcal{D}^H_{sk}) \leq \epsilon^{\mathrm{ks}}(k). \tag{1}$$

Proof. Let us define $p^C_{X,Y}(b, b') \triangleq \Pr[C(X) = b \wedge C(Y) = b']$. Note that $\Pr[C(X) \neq C(Y)] = p^C_{X,Y}(1,0) + p^C_{X,Y}(0,1) \leq \epsilon^{\mathrm{ks}}(k)$. Hence,

$$\begin{aligned} p^C_X(1) - p^C_Y(1) &= p^C_{X,Y}(1,0) - p^C_{X,Y}(0,1) \\ &\leq \epsilon^{\mathrm{ks}}(k) - 2p^C_{X,Y}(0,1) \leq \epsilon^{\mathrm{ks}}(k). \end{aligned}$$

We define the security notion of plaintext simulatability as follows.

Definition 6. **[Plaintext Simulatability]** *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be a public-key encryption scheme. We say that* Π *is secure in the PS sense if* Π *is secure in the IND-CPA sense and there is a knowledge simulator for* Π*.*

4 PS implies IND-CCA2

In the following, we show that PS implies IND-CCA2.

Theorem 1. *PS implies IND-CCA2.*

The proof follows from the following lemma.

Lemma 2. *Let* Π *be a public-key encryption scheme that is secure in the PS sense. For every IND-CCA2 adversary* A *against* Π*, we can construct IND-CPA adversary* A' *against the same* Π *such that they satisfy the following relation:*

$$\mathrm{Adv}^{\mathtt{cca2}}_{A,\Pi}(k) \leq \mathrm{Adv}^{\mathtt{cpa}}_{A',\Pi}(k) + 2q\epsilon^{\mathrm{ks}}(k), \tag{2}$$

where q *denotes the total number of queries of* A *to the decryption oracle and* $\epsilon^{\mathrm{ks}}(\cdot)$ *denotes the knowledge simulation error function for* Π *defined in Def. 5.*

Proof. Suppose for contradiction that we have adversary A that breaks Π in the IND-CCA2 sense in the random oracle model. We then construct adversary A' that breaks Π in the IND-CPA sense in the random oracle model. Intuitively, the idea is that A' runs A and, for the queries of A to the decryption oracle, A' uses knowledge-simulator K and random oracle H, to simulate the answers.

Formally, to use knowledge simulator K, we construct a sequence of adversaries, $B_1, \ldots, B_q$, where q denotes the total number of queries of A to the decryption oracle. On each transcript of B_i, K produces the corresponding plaintext, making access to random oracle H. B_1 runs A until A outputs the first decryption query, where B_1 just hands queries of A to random oracle H and hands the answers of H to A. Each pair of query and answer is put on list $\mathcal{T}_H$. When A outputs the first decryption query, B_1 halts and outputs it as a ciphertext. B_i, $1 < i$, is a similar algorithm except that it uses K, too. It starts by running A. For j-th decryption query made by A, $1 \leq j < i$, B_i runs K on the corresponding transcript of B_j, to produce the decryption, which is sent to A. When K asks the random oracle H, B_i just hands the query to H and replies to K with the answer of H. Each interaction between A and H (via B_i) is put on list $\mathcal{T}_H$. B_i halts and output the i-th decryption query of A. *In addition, each interaction between K and H (via B_i) is also put on list $\mathcal{T}_H$, which is the significant difference from the knowledge extractor in the plaintext awareness setting.* B_i halts and outputs the i-th decryption query of A. For some i', $1 \leq i' \leq q$, A is given the challenge ciphertext y^*. Namely, $\mathcal{Y} = \{\}$ in the transcript of B_i, for $1 \leq i < i'$, and $\mathcal{Y} = \{y^*\}$ in the transcript of B_i, for $i' \leq i \leq q$. The presence of the adversaries, $B_1, \ldots, B_q$, is just "conceptual". In the following, we directly construct A' to break Π in the IND-CPA sense, where what we insist on is that the transcript given to K for the i-th decryption is identical to the transcript of B_i described above.

Construction. We construct A' by using A as a black box as follows.

1. A' is given pk.
2. A' runs A with pk in the find mode.
3. When A asks random oracle H, A' submits query h of A to random oracle H and replies A with the answer $H(h)$. Then A' puts $(h, H(h))$ in $\mathcal{T}_H$.
4. When A submits y to the decryption oracle, A' runs K with $(\mathcal{T}_H, \{\}, y, pk)$ and returns the answer x to A.
5. When A outputs a pair of plaintexts, (x_0, x_1), A' outputs the same pair (x_0, x_1).
6. A' is given $y^* := \mathcal{E}_{pk}(x_b)$ where bit b is randomly chosen.
7. A' hands y^* to A and runs A in the guess mode.
8. When A asks random oracle H, A' submits query h of A to random oracle H and replies A with the answer $H(h)$. A' puts $(h, H(h))$ in $\mathcal{T}_H$.
9. When A submits y to the decryption oracle, A' runs K on $(\mathcal{T}_H, \{y^*\}, y, pk)$ and returns the answer x to A.
10. Finally, A output bit b'.

First, to help the reader's comprehension, we analyze a simple case in which A' can use a knowledge simulator such that its output is statistically indistinguishable from the real decryption. We will consider then the general case that A' can use a knowledge simulator whose output is computationally indistinguishable from the real decryption.

Analysis of the "statistically close" case. As mentioned above, the transcript that A' makes for the i-th decryption is identical to the transcript of B_i for every i. So, from the definition, K can successfully produce the "real" decryption of every query y except for a negligible error probability, $\epsilon^{\mathrm{ks}}(k)$. Let F be the event that at least one of the replies of A' to A is not equal to the "real" decryption. Then it is obvious that $\Pr[F] \leq q\epsilon^{\mathrm{ks}}(k)$.

If A' succeeds in decrypting all the queries of A, it is obvious by construction that the success event of A' is identical to the success event of A, namely

$$\mathrm{Succ}^{\mathtt{cpa}}_{A',\Pi}(k) \cap \neg F = \mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k) \cap \neg F. \tag{3}$$

Some modification is necessary to make their probability spaces coincide, but we omit the description since it is somewhat trivial.

Claim. Let A, B, and F be events on some probability space Ω. Suppose that $A \cap \neg F = B \cap \neg F$ and $\Pr_\Omega[A] > \Pr_\Omega[B]$. Then we have $\Pr_\Omega[B] \geq \Pr_\Omega[A] - \Pr_\Omega[F]$.

Proof. $\Pr_\Omega[A] - \Pr_\Omega[B] \leq \Pr_\Omega[A \cap F] - \Pr_\Omega[B \cap F] \leq \Pr_\Omega[F]$.

By applying this claim to our case, we have

$$\Pr[\mathrm{Succ}^{\mathtt{cpa}}_{A',\Pi}(k)] \geq \Pr[\mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k)] - q\epsilon^{\mathrm{ks}}(k). \tag{4}$$

Hence, we complete the proof in this case. (Note: Since $\mathrm{Adv}^{\mathtt{atk}}_{A,\Pi}(k) = 2\Pr[\mathrm{Succ}^{\mathtt{atk}}_{A,\Pi}(k)] - 1$, the entire error probability becomes double.)

Analysis of the "computationally close" case. We now consider the "computationally close" case. As with the statistically close case, the transcript that A' makes for the i-th decryption is identical to the transcript of B_i for every i. However, each output of K is only computationally indistinguishable from the real decryption of each query (over the choice of H for almost all (pk, sk)). Suppose for contradiction that

$$\Pr[\mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k)] - \Pr[\mathrm{Succ}^{\mathtt{cpa}}_{A',\Pi}(k)] > q\epsilon^{\mathrm{ks}}(k). \tag{5}$$

Then we show that we can construct distinguisher C to distinguish, for some i, random variable

$$\mathcal{S}_i \triangleq < \mathcal{T}_{H,i}, \mathcal{Y}_i, y_i, pk, sk, K^H(\mathcal{T}_{H,i}, \mathcal{Y}_i, y_i, pk) >$$

from random variable

$$\mathcal{R}_i \triangleq < \mathcal{T}_{H,i}, \mathcal{Y}_i, y_i, pk, sk, D^H_{sk}(y_i) >$$

with probability more than $\epsilon^{\mathrm{ks}}(k)$, where $\mathcal{T}_{H,i}$ and $\mathcal{Y}_i$ denote, respectively, the lists, $\mathcal{T}_H$ and $\mathcal{Y}$, at the time A' submitted the i-th query to the decryption oracle and y_i denotes the i-th query of A' to the decryption oracle. If such a distinguisher exists, it contradicts the assumption that the output of K is computationally indistinguishable from the real decryption.

Such distinguisher C can be constructed by the hybrid argument method. Now let us denote by $\mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k, i)$ the success event of A for Π where the first i decryption queries come from the real decryption oracle but the latter answers of the queries come from the simulator. Then obviously $\mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k, q) = \mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k)$ and $\mathrm{Succ}^{\mathtt{cca2}}_{A,\Pi}(k, 0) = \mathrm{Succ}^{\mathtt{cpa}}_{A',\Pi}(k)$.

By (5),

$$\Pr[\mathrm{Succ}^{\mathsf{cca2}}_{A,\Pi}(k,q)] - \Pr[\mathrm{Succ}^{\mathsf{cca2}}_{A,\Pi}(k,0)] > q\epsilon^{\mathrm{ks}}(k).$$

Hence, for some i, by the hybrid argument,

$$\Pr[\mathrm{Succ}^{\mathsf{cca2}}_{A,\Pi}(k,i)] - \Pr[\mathrm{Succ}^{\mathsf{cca2}}_{A,\Pi}(k,i-1)] > \epsilon^{\mathrm{ks}}(k).$$

Note that C is given sk and picks up the challenge bit b by itself. Hence, for some i, C can distinguish, using A as above, random variable $\mathcal{S}_i$ from random variable $\mathcal{R}_i$ with at least probability $\epsilon^{\mathrm{ks}}(k)$. This contradicts the original assumption. Hence, (5) is not true. This completes the proof.

5 PA $\subsetneq$ PS

It is obvious by definition that PA still implies PS, namely PA $\subset$ PS. We will now show that PA $\subsetneq$ PS.

We propose a public-key encryption scheme, denoted Π_0, based on one-way permutation. The scheme is CCA2 secure in the random oracle model because it meets PS. At the same time, the scheme is evidence that PS is "properly" weaker than PA, because it does not meet PA. In addition, the scheme exemplifies a more natural example than the one shown in [2], namely PA is "properly" stronger than IND-CCA2.

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a trap-door permutation over $\{0,1\}^k$. Let $G : \{0,1\}^k \to \{0,1\}^k$ and $H : \{0,1\}^* \to \{0,1\}^{k'}$ be hash functions. Consider public-key encryption scheme Π_0 as follows:

Key generation. Simply run the generator for the one-way trapdoor permutation scheme, obtaining f and f^{-1}. The public-key is f and the private-key is f^{-1}.

Encryption. For plaintext $m \in \{0,1\}^k$, the encryption algorithm with public-key f randomly picks up inner coins $\sigma \in \{0,1\}^k$ and computes the ciphertext $\mathcal{E}^{\mathrm{hy}}_{pk}(m;\sigma)$ as follows.

$$\mathcal{E}^{\mathrm{hy}}_{pk}(m;\sigma) = f(\sigma) \mid\mid G(\sigma) \oplus m \mid\mid H(\sigma||e||c), \tag{6}$$

where $e = f(\sigma)$ and $c = G(\sigma) \oplus m$.

Decryption. Given ciphertext y, check whether the string can be parsed into the specified form. If not, just reject it, otherwise, appropriately parse it into $e \in \{0,1\}^k$, $c \in \{0,1\}^k$, and $h \in \{0,1\}^{k'}$. The decryption algorithm with f^{-1} then computes

$$\sigma = f^{-1}(e) \quad \text{and} \quad m = G(\sigma) \oplus c. \tag{7}$$

If $\sigma, m \in \{0,1\}^k$ and $h = H(\sigma||e||c)$, then the decryption algorithm outputs m; otherwise reject the ciphertext.

5.1 Π_0 does not meet PA

We prove in the next section that this scheme is CCA2-secure when G and H are modeled as random oracles. However this is not PA at the same time, which we can prove as follows: Consider an adversary B for Π_0 that takes public-key pk and outputs a ciphertext, after asking queries to the random oracles. The adversary can produce a *new* ciphertext $y = (e, c, h)$, *without knowing* the corresponding plaintext in the following way — The adversary just picks up any $\sigma, c \in \{0,1\}^k$ and computes $e = f(\sigma)$, using $pk = \{f\}$. The adversary submits (σ, e, c) to H to get $h = H(\sigma||e||c)$, and then submits, *not asking G with σ*, the ciphertext $y = (e, c, h)$ to the decryption oracle. The decryption for this ciphertext, $G(\sigma) \oplus c$, is unpredictable, because $G(\sigma)$ is unpredictable. Hence, the adversary can't be aware of the decryption for this ciphertext before getting it from the decryption oracle. Formally speaking, it is necessary to show that Π_0 is "plaintext-aware", i.e., that one can construct a knowledge-extractor that outputs the value $\mathcal{D}_{sk}^{\mathrm{hy},G,H}(e, c, h)$ on the transcript given above, i.e., $((\sigma, e, c, h), \{\}, y, pk)$. However, if such a knowledge extractor exists for Π_0, it contradicts the unpredictability of $G(\sigma)$. Finally, we have the following claim.

Theorem 2. *Let Π_0 be the above encryption scheme. There exists a polynomial-time bounded adversary B so that there is no knowledge extractor for Π_0 such that*

$$\mathrm{Adv}^{\mathrm{ke}}_{K,B,\Pi_0}(k) > 2^{-k}.$$

Proof. Suppose that K is a knowledge extractor for Π_0. We construct adversary B as mentioned above. We then run K on B's transcript $((\sigma, e, c, h), \{\}, y, pk)$. If K outputs m', we set $G(\sigma)$ as $m' \oplus c$ and output it. Since G is a random oracle and K is not allowed to access G, it is obvious by construction that $\mathrm{Adv}^{\mathrm{ke}}_{K,B,\Pi_0}(k) \leq 2^{-k}$, which is negligible in k.

By this theorem, Π_0 does not meet PA.

5.2 Π_0 meets PS

We prove that scheme Π_0 is "plaintext-simulatable".

Theorem 3. *Scheme Π_0 meets PS.*

The proof follows from the lemmas below.

Definition 7. *Let $\mathcal{F}$ be the generator for the one-way trapdoor permutation over $\{0,1\}^k$, obtaining (f, f^{-1}). Let A be an adversary that attacks f. For k, we denote the advantage of A by* $\mathrm{Adv}^{\mathrm{ow}}_{A,f}(k) \triangleq$

$$\Pr[(f, f^{-1}) \leftarrow \mathcal{F}; y \leftarrow \{0,1\}^k : A(f, y) = f^{-1}(y)].$$

We denote by $\epsilon^{\mathrm{ow}}(k)$ the maximum of $\mathrm{Adv}^{\mathrm{ow}}_{A,f}(k)$ over all adversaries A who run in polynomial time in k.

Lemma 3. *Let Π_0 be the scheme mentioned above. For every polynomial time adversary A, we have*

$$\mathrm{Adv}^{\mathrm{cpa}}_{A,\Pi_0}(k) \leq 2 \cdot \epsilon^{\mathrm{ow}}(k). \tag{8}$$

Lemma 4. *Let ϵ^{ks} be the knowledge simulation error function for Π_0 mentioned above. We then have*

$$\epsilon^{\mathrm{ks}}(k) \leq \epsilon^{\mathrm{ow}}(k) + 2^{-k'}. \tag{9}$$

We give the proofs below. By these two lemmas and theorem 1, it automatically turns out that Π_0 is plaintext-simulatable in the random oracle model. In particular, applying lemma 2 to Π_0, the following lemma holds.

Lemma 5. *Let Π_0 be the scheme mentioned above. For every polynomial time adversary A,*

$$\mathrm{Adv}^{\mathrm{cca2}}_{A,\Pi_0}(k) \leq 2(q+1)\epsilon^{\mathrm{ow}}(k) + q \cdot \left(\frac{1}{2}\right)^{k'-1}, \tag{10}$$

where q denotes the total number of the queries of A to the decryption oracle.

Proofs of Lemmas, 3 and 4 To save space, we first describe common operations in both proofs.

Simulation of oracle G. Let $\mathcal{T}_G$ be the list of pairs of the form (σ, g) $(\in \{0,1\}^k \times \{0,1\}^k)$. $\mathcal{T}_G$ is initially empty. For a fresh query $\sigma \in \{0,1\}^k$ to G, pick up $g \in_R \{0,1\}^k$ to reply with. Then add (σ, g) to $\mathcal{T}_G$.

Simulation of oracle H. Let $\mathcal{T}_H$ be the list of tuples of the form (σ, c, h) $(\in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{k'})$. $\mathcal{T}_H$ is initially empty. For a fresh query (σ, c) to H, pick up $h \in_R \{0,1\}^k$ to reply with. Then add (σ, c, h) to $\mathcal{H}$.

These simulations will be done by the constructed adversaries specified later, namely A' and K.

Proof of Lemma 3. Let A be a CPA-adversary for Π_0. We then construct adversary A' that attacks the one-wayness of f. A' takes e^*, picked up uniformly on $\{0,1\}^k$. A' runs A with $pk = \{f\}$. For every query of A to the random oracle(s), A' returns the answer to A, following the procedures above. When A outputs (m_0, m_1), A picks up at random $c^* \in \{0,1\}^k$, $h^* \in \{0,1\}^{k'}$ and returns (e^*, c^*, h^*) to A. A' outputs σ^* and halts if A submits a query to the random oracles with σ^* such that $e^* = f(\sigma^*)$.

Denote by AskG^* the event that A submits query $\sigma^* (\triangleq f^{-1}(e^*))$ to random oracle G. Similarly, denote by AskH^* the event that A submits query (σ^*, e, c) to random oracle H where $(e, c) \neq (e^*, c^*)$. Denote by Ask$G^* \vee H^*$ the event that either AskG^* or AskH^* occurs.

It is obvious by construction that the distribution of challenge ciphertext (e^*, c^*, h^*) is independent of the distribution of hidden bit b unless Ask$G^* \vee H^*$ occurs. Therefore, we have

$$\mathrm{Succ}^{\mathrm{cpa}}_{A,\Pi_0}(k) \leq \Pr[\mathrm{Ask}G^* \vee H^*] + \frac{1}{2}\Pr[\neg \mathrm{Ask}G^* \vee H^*].$$

Since $\Pr[\mathrm{Ask}G^* \vee H^*] \leq \epsilon^{\mathrm{ow}}(k)$, it leads that

$$\mathrm{Adv}^{\mathrm{cpa}}_{A,\Pi_0}(k) \leq 2\epsilon^{\mathrm{ow}}(k). \tag{11}$$

Proof of Lemma 4. We construct knowledge simulator K as follows. For given $(\mathcal{T}_G, \mathcal{T}_H, \mathcal{Y}, y, pk)$ to K, where $y = (e, c, h) \notin \mathcal{Y}$ and $\mathcal{Y} = \{(e^*, c^*, h^*)\}$, simulate the decryption of y as follows.

1. Search (σ, e, c, h) in $\mathcal{T}_H$ such that $e = f(\sigma)$. If not successful, reject the ciphertext and halts, otherwise
2. For the tuple (σ, e, c, h) in $\mathcal{T}_H$, search $\mathcal{T}_G$ to find (σ, g). If it exists, return $c \oplus g$, otherwise
3. Query oracle G on σ to get $g = G(\sigma)$ (where we simulate G as above) and finally return $c \oplus g$.

Analysis. Denote by AskG^* the event that B submits query $\sigma^* (\triangleq f^{-1}(e^*))$ to random oracle G. Similarly, denote by AskH^* the event that B submits query (σ^*, e, c) to random oracle H where $(e, c) \neq (e^*, c^*)$. Denote by Ask$G^* \vee H^*$ the event that either AskG^* or AskH^* occurs. Denote by Ask$H$ the event that B submits query (σ, e, c) to random oracle H such that $e = f(\sigma)$ and $y = (e, c, h)$.

Assume Ask$G^* \vee H^*$ does not occur. If Ask$H$ occurs, K always succeeds in decrypting y. On the other hand, if Ask$H$ does not occur, K always rejects y, whereas the real decryption oracle would reject it except with probability $2^{-k'}$, which means that the knowledge simulation error of K, conditional on $\neg$Ask$G^* \vee H^* \cap \neg$Ask$H$, is bounded by $2^{-k'}$.

To sum up, we have

$$\begin{aligned}\epsilon^{\mathrm{ks}}(k) &\leq \Pr[\mathrm{Ask}G^* \vee H^*] + \Pr[\neg(\mathrm{Ask}G^* \vee H^*) \cap \neg \mathrm{Ask}H] \cdot \left(\frac{1}{2}\right)^{k'} \\ &\leq \epsilon^{\mathrm{ow}}(k) + \left(\frac{1}{2}\right)^{k'}.\end{aligned}$$

6 Other PS Encryption Schemes

In this section, we present two public-key encryption schemes that lie in PS but not in PA. Why these schemes do not meet PA is essentially as the same as why Π_0 above does not meet PA. So we will omit these proofs.

6.1 A DDN-based construction of CCA2-secure encryption

The Dolev-Dwork-Naor encryption scheme [6, 7] is CCA2 secure. A concrete description of the scheme is given in the appendix. The key for the security proof is as follows: Let $< F^*, s^*, c^*, p^* >$ be the challenge ciphertext. Since F^* is the verification key of secure signature scheme Σ in the sense of [10], the adversary cannot produce signature s on (c, p) such that $(c, p) \neq (c^*, p^*)$. For $F \neq F^*$, the public key $e_1^{v_1}, \ldots, e_n^{v_n}$ happens to be different from $e_1^{v_1^*}, \ldots, e_n^{v_n^*}$ with overwhelming probability where $v_1 \cdots v_n = h(F)$ and $v_1^* \cdots v_n^* = h(F^*)$. Hence, the simulator can decrypt $< F, s, c, p >$ such that $F \neq F^*$, using a secret key $d_i^{v_i^*}$ such that $v_i^* \neq v_i$.

The following scheme is an interpretation of the Dolev-Dwork-Naor encryption scheme in the random oracle model. When $F \neq F^*$, the session key H happens to be different from H^* with overwhelming probability. This is a point that allows the knowledge simulator to simulate the decryption of the ciphertext in the scheme.

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a trap-door permutation over $\{0,1\}^k$. Let $H : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$ be a hash function modeled as a random oracle. We denote by $\Sigma = (G, S, V)$ a signature scheme secure against adaptive chosen message attacks [10] (For instance, if Σ is the Schnorr signature, the total encryption scheme is efficient and secure in the random oracle model).

Key generation. Simply run the generator for the one-way trapdoor permutation scheme, obtaining f and f^{-1}. The public-key is f and the secret-key is f^{-1}.

Encryption. Encrypt plaintext $m \in \{0,1\}^k$, using public-key f, as follows.

1. Run $G(1^k)$ to obtain (F, P), a pair of verification and signature keys.
2. Randomly pick up inner coins $\sigma \in \{0,1\}^k$ and compute $e := f(\sigma)$ and $c := m \oplus H(\sigma, F)$.
3. Generate signature s on (e, c), using the signing key P.
4. The encryption of m is (F, s, e, c).

Decryption. Given the ciphertext, check whether the string can be parsed into the specified form. If not, just reject it, otherwise appropriately parse it into (F, s, e, c). Then,

1. Verify that s is a signature on $(e, c) \in \{0,1\}^k \times \{0,1\}^k$, using the verification key F. If not, just reject it, otherwise,
2. Compute $\sigma = f^{-1}(e)$
3. Output $m = c \oplus H(\sigma, F)$.

Theorem 4. *The scheme meets PS.*

Proof. (Sketch) It is obvious by construction that the scheme meets IND-CPA. The construction of a knowledge simulator is as follows: Let (F, s, e, c) be a ciphertext that B submits to the knowledge simulator. Let $(F^*, s^*, e^*, c^*) \in \mathcal{Y}$ be the challenge ciphertext. If $F = F^*$, we just reject the ciphertext (implicitly assuming $(s^*, e^*, c^*) \neq (s, e, c)$). If $F \neq F^*$ and s is a valid signature on (e, c), we then simulate the decryption of (e, c) as in the proof of Lemma 4. Namely if there is the corresponding plaintext of (e, c) in list $\mathcal{T}_H$, then return it; otherwise register a random string as $H(\sigma, F)$ in $\mathcal{T}_H$ and return $c \oplus H(\sigma, F)$.

As with the proof Lemma 4, when $F \neq F^*$ and s is a valid signature on (e, c), each output of this simulation is identical to the real decryption of (e, c) except with negligible error 2^{-k} where k is the size of the output of H. When $F = F^*$, the probability that s is a valid signature on message (e, c) with the verification-key F is negligible because of unforgeability of Σ.

Remark 2. Since the Schnorr signature is known to be secure [10] in the random oracle model [16], this DDN encryption scheme by applying the Schnorr signature to Σ turns out IND-CCA2 in the random oracle model. To the best of our knowledge, there has been no IND-CCA2 encryption scheme (in the random oracle model) except this construction and [1] such that it is composed of a IND-CPA encryption plus a 3-move ZK-like signature. For instance, the signed El Gamal encryption scheme has not been proven IND-CCA2.

6.2 A CCA2-secure encryption scheme based on a strong pseudo-random permutation family

Let $\mathcal{P} = \{\{P_a : \{0,1\}^k \to \{0,1\}^k\}_{a \in \{0,1\}^k}\}_{k \in \mathbb{N}}$ be a strong pseudo-random permutation family [12]. Informally, a strong pseudo-random permutation has the property that the output of P_a^{-1}, not only P_a, with any input, looks random, i.e., pseudo-random. A well-known construction of strong pseudo-random permutations comsists of four rounds of Feistel permutations where each round takes a pseudo-random function with a different key [12]. Another construction is known in [13]. Using this family, we can naturally construct a PS

encryption scheme. This scheme provides the optimal ciphertext lengths for verifying the validity of ciphertexts, i.e., (ciphertext size) = (message size) + (randomness size). According to [15], such a construction remained open [2].

Let $f : \{0,1\}^k \to \{0,1\}^k$ be a trap-door permutation over $\{0,1\}^k$. Let $G : \{0,1\}^k \to \{0,1\}^k$ be a hash function modeled as a random oracle.

Key generation. Simply run the generator for the one-way trapdoor permutation scheme, obtaining f and f^{-1}. The public-key is f and the secret-key is f^{-1}.

Encryption. For plaintext $m \in \{0,1\}^k$, the encryption algorithm with public-key f randomly picks up inner coins $\sigma \in \{0,1\}^k$ and computes the ciphertext $\mathcal{E}_{pk}(m;\sigma)$ as follows.

$$\mathcal{E}_{pk}(m;\sigma) := f(\sigma) \mid\mid P_{G(\sigma)}(m). \tag{12}$$

Decryption. Given ciphertext (e,c), check whether the string can be parsed into the specified form. If not, just reject it, otherwise, appropriately parse it into $e \in \{0,1\}^k$, $c \in \{0,1\}^k$. The decryption algorithm with f^{-1} then computes

$$\sigma = f^{-1}(e) \quad \text{and} \quad m = P^{-1}_{G(\sigma)}(c). \tag{13}$$

If $\sigma, m \in \{0,1\}^k$, the decryption algorithm outputs m; otherwise reject the ciphertext.

Theorem 5. *The scheme meets PS.*

Proof. (Sketch) It is obvious by construction that the scheme is IND-CPA. The key for constructing a knowledge simulator is that if adversary B submits ciphertext (e,c) so that $\sigma \triangleq f^{-1}(e)$ is not in the list $\mathcal{T}_G$, we let the simulator reply with a truly random string m', which is computational indistinguishable from the real decryption $P^{-1}_{G(\sigma)}(c)$ because P is a strong pseudo-random permutation. Therefore, we can also construct a knowledge simulator.

7 Open problem

Although we believe that this new security class can help designers find a lot of new schemes, at least the current definition does not seems to cover the ideas of constructing secure hybrid encryption schemes such as the one proposed by Shoup [21] where the underlying symmetric encryption scheme is IND-CCA2 (in the symmetric encryption setting). We do not know how to prove Shoup's hybrid encryption schemes meet PS, except for the case that the underlying symmetric encryption scheme is a strong pseudo-random permutation (which implies IND-CCA2 symmetric encryption). We also can't prove that they do not meet PS. So far, we do not have any counter-example that PS is "properly" stronger than IND-CCA2.

References

1. M. Abe. Combining encryption and proof of knowledge in the random oracle model. *The Computer Journal*, 47(1):58–70, 2004. (Presented in CT-RSA 2002, LNCS 2271).
2. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.

[2] Independently, Choi, Kobara, and Imai recently presented another construction with optimal redundancy [5]. This scheme also lies in PS but not in PA.

3. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology — EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.
4. M. Blum, P. Feldman, and S. Micali. Proving security against chosen cyphertext attacks. pages 256–268. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.
5. Y. Cui, K. Kobara, and H. Imai. A generic conversion with optimal redundancy. In *RSA Conference 2005, Cryptographers' Track (CT-RSA 05)*, 2005. To appear.
6. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proceedings of the* 23*rd annual ACM Symposium on Theory of Computing*, pages 542–552, New York City, 1991.
7. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM. J. Computing*, 30(2):391–437, 2000. (Presented in STOC'91).
8. D. Dolev and A. Yao. On the security of public-key protocols. *Transactions on Information Theory*, 29, 1983.
9. P. A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen ciphertext attacks. In C. Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 351–368. Springer-Verlag, 2001.
10. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
11. J. Herzog, M. Liskov, and S. Micali. Plaintext awareness via key registration. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Advances in Cryptology — CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 548–564. Springer-Verlag, 2003.
12. M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM. J. Computing*, 17(2):373–386, 1988. (Presented in CRYPTO'85).
13. M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-rackoff revisited. *JOC*, 12(1):29–66, 1999. (Presented in STOC'97).
14. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the* 21*st annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.
15. D. H. Phan and D. Pointcheval. Chosen-ciphertext security without redundancy. In C. S. Laih, editor, *Advances in Cryptology – Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, December 2003.
16. D. Pointcheval and J. Stern. Security proofs for signature schemes. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.
17. C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1992.
18. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS99*, pages 543–553, 1999.
19. A. De Santis, G. Di Crescenzo, and G. Persiano. Necessary and sufficient assumptions for non-interactive zero-knowledge proofs of knowledge for all np relations. In *ICALP 2000*, volume 1853 of *LNCS*, pages 451–462. SV, 2000.
20. A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. In *FOCS92*, pages 427–436, 1992.
21. V. Shoup. A proposal for an ISO standard for public key encryption. Technical report, December 2001. Cryptology ePrint Archive, Report 2001/112 `http://eprint.iacr.org`.

A The DDN encryption scheme

For completeness, we recall the encryption scheme presented by Dolev, Dwork and Naor [6, 7], which is the first NM-CCA2 secure (CCA2-secure) encryption scheme under the generic assumption (that trap-door functions exists).

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a CPA-secure encryption scheme and let $\Sigma = (G, S, V)$ be a (possibly, one-time) signature scheme secure against adaptive chosen message attacks [10]. Note that a CPA secure encryption scheme can be obtained by using any trap-door encryption scheme and a secure signature scheme in the sense of [10] can be obtained by using any one-way function.

The DDN encryption scheme Π^{DDN} is constructed as follows.

Key generation.

1. Run $\mathcal{K}(1^k)$ $2n$ times. Denote the output by $(e_1^0, d_1^0), (e_1^1, d_1^1), \ldots, (e_n^0, d_n^0), (e_n^1, d_n^1)$.
2. Generate random U, the shared random string for non-interactive zero-knowledge proof.
3. Pick up a universal one-way hash function $h : \{0,1\}^* \to \{0,1\}^n$.
4. The public key for the target encryption scheme Π^{DDN} is $< h, e_1^0, e_1^1, \ldots, e_n^0, e_n^1, U >$. The corresponding secret key is $< d_1^0, d_1^1, \ldots, d_n^0, d_n^1 >$.

Encryption. For plaintext $m \in \{0,1\}^k$, the encryption algorithm works as follows.

1. Run $G(1^k)$ to obtain (F, P), a pair of verification and signature keys.
2. Compute $h(F)$. Denote the output by $v_1, \ldots, v_n$, where $v_i \in \{0,1\}$.
3. For each $1 \leq i \leq n$, pick up random string r_i and compute $c_i := \mathcal{E}_{e_i^{v_i}}(m; r_i)$.
4. Given $c :=< e_1^{v_1}, \ldots, e_n^{v_n}, c_1, \ldots, c_n >$, generate a non-interactive zero-knowledge proof p for language L, using witness $(r_1, \ldots, r_n)$ and string U, where $L := \{c | \mathcal{D}_{d_1^{v_1}}(c_1) = \cdots = \mathcal{D}_{d_n^{v_n}}(c_n)\}$.
5. Generate signature s on (c, p), using the signing key P.
6. The encryption of m is $< F, s, c, p >$.

Decryption. Given ciphertext $< F, s, c, p >$, check whether the string can be parsed into the specified form. If not, just reject it, otherwise, appropriately parse it. Then,

1. Verify that s is a signature on (c, p), using the verification key F.
2. Verify the validity of the non-interactive proof p that asserts $c \in L$, using the common string U.
3. Compute $h(F)$ and check $v_1 \cdots v_n = h(F)$.
4. Retrieve m using one of $d_1^{v_1}, \ldots, d_n^{v_n}$.

Claim. The encryption scheme Π^{DDN} is CCA2 secure if trap-door functions exist [7].