On Plaintext-Aware Public-Key Encryption Schemes

James Birkett

Technical Report
RHUL–MA–2001–11
15th April 2010

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

http://www.rhul.ac.uk/mathematics/techreports

ON PLAINTEXT-AWARE PUBLIC-KEY ENCRYPTION SCHEMES

James Birkett

Royal Holloway and Bedford New College,
University of London

Thesis submitted to
The University of London
for the degree of
Doctor of Philosophy
2009.

I, James Birkett, declare that this thesis is my own work.

Abstract

Plaintext awareness is a property of a public-key encryption scheme intended to capture the idea that the only way to produce a valid ciphertext is to take a message and encrypt it. The idea is compelling, but the devil, as always, is in the details. The established definition of plaintext awareness in the standard model is known as PA2 plaintext awareness and was introduced by Bellare and Palacio. We propose a modified definition of plaintext awareness, which we call 2PA2, in which the arbitrary stateful plaintext creators of the PA2 definition are replaced with a choice of two fixed stateless plaintext creators. We show that under reasonable conditions our new definition is equivalent to the standard one. We also adapt techniques used by Teranishi and Ogata to show that no encryption scheme which allows arbitrarily long messages can be PA2 plaintext aware, a disadvantage which our new definition does not appear to share.

Dent has shown that a variant of the Cramer-Shoup encryption scheme based on the Diffie-Hellman problem is PA2 plaintext aware under the Diffie-Hellman Knowledge (DHK) assumption. We present a generalisation of this assumption to arbitrary subset membership problems, which we call the Subset Witness Knowledge (SWK) assumption, and use it to show that the generic Cramer-Shoup and Kurosawa-Desmedt encryption schemes based on hash proof systems are plaintext aware. In the case of the Diffie-Hellman problem, the SWK assumption is exactly the Diffie-Hellman Knowledge assumption, but we also discuss several other possible instantiations of this assumption.

Acknowledgements

First and foremost, I am indebted to my supervisor Dr Alexander Dent. Without his encouragement, enthusiasm and relentless dedication to accuracy, clarity and precision, this thesis would not have been possible. I would also like to thank my advisor, Dr Steven Galbraith, and Prof Kenneth Paterson for their unique perspectives, Prof Keith Martin, who introduced me to the terrifying reality of teaching, and Prof Nigel Smart, who encouraged me to pursue an academic career in the first place.

This research was funded by the Engineering and Physical Sciences Research Council, and I am grateful for its financial support. I would also like to thank the European Commission for funding the ECRYPT project, which provided me with generous funding to attend conferences and workshops abroad.

Part of the writing of this thesis was done while working at the Information Security Institute at the Queensland University of Technology. I would like to express my gratitude to Prof Colin Boyd and Dr Juanma González Nieto for their help and understanding during this time.

I must also extend my thanks to all my colleagues and friends at Royal Holloway, who have never failed to provide stimulating conversation, discussion and entertainment.

Finally, I must thank my parents, John and Christine Birkett, for their love and support, my sister, Rowan, for her endless enthusiasm and pride, my brother, Alex, for his belief in my abilities and encouragement to use them. To Luke, I simply say thank you.

Contents

Chapter 1

Introduction and Definitions

Plaintext awareness was first defined by Bellare and Rogaway [4] as a device to prove the OAEP scheme was IND-CCA2 secure. The idea is that an encryption scheme is plaintext aware if the only way to generate a valid ciphertext is to take a message and encrypt it.

The original form of the definition relied heavily on the random oracle model and could not easily be generalised to the standard model. The first definition of plaintext awareness in the standard model was proposed by Bellare and Palacio [3]. In this formulation, the plaintext extractor is given the random tape used by the plaintext creator instead of the oracle queries. We will only consider this and related standard-model definitions in this work.

In either the random oracle or standard model, the plaintext extractor is given enough information to "follow" the execution of the ciphertext creator. One may think of the ciphertext creator as trying to construct a ciphertext with the extractor watching over its shoulder: if the ciphertext creator takes a message and encrypts it, the extractor sees this and can simply output this message. On the other hand, plaintext awareness means that there is no way

to produce a valid ciphertext without "knowing" the underlying plaintext, so if the ciphertext creator does not encrypt some known message, then the plaintext extractor may confidently assert that the ciphertext is invalid.

Bellare *et al.* [2] later studied plaintext awareness in as a property in its own right, but in order to prove that an encryption scheme which is IND-CPA secure and plaintext aware is IND-CCA2 secure, they had to capture the adversary's ability to obtain ciphertexts that were produced by a third party. To this end, they augmented the model with a "plaintext creator" which creates messages which the adversary does not know, but may have some influence over. These messages are then encrypted before being returned to the adversary. This stronger definition is called PA2 plaintext awareness. Despite this, the original proof of security for OAEP went unchallenged in the published literature until four papers relating to OAEP appeared in CRYPTO 2001. Manger demonstrated a chosen ciphertext attack against the PKCS standard version of RSA OAEP [23]. This attack relied on a side-channel, specifically the ability to distinguish two different types of decryption error. Shoup demonstrated that the there is no black box proof that an encryption scheme which is both IND-CPA secure and PA1 plaintext aware is IND-CCA2 secure. He also presented a modified version of OAEP, called OAEP+ which is IND-CCA2 secure in the random oracle model. Fujisaki *et al.* [16] used algebraic properties of the RSA function to give a proof of security for RSA-OAEP, despite the weaknesses in the generic OAEP construction. Boneh [6] used algebraic properties of both the RSA and Rabin trapdoor functions to show that a construction much simpler than the full version of OAEP achieves IND-CCA2 security when instantiated with either of those trapdoor functions.

When Bellare and Palacio introduced the standard model definitions of

plaintext awareness [3], they showed that a simplified form of the Cramer-Shoup encryption scheme is PA1 plaintext aware under a non-standard assumption called the Diffie-Hellman knowledge (DHK) assumption, also known as the knowledge of exponent assumption, which was first introduced by Damgård [13]. Dent later showed that the full Cramer-Shoup scheme is PA2 plaintext aware under the same assumption. Since these schemes were already known to be IND-CCA2 secure, plaintext awareness does not serve its original function as a method to prove that an encryption scheme is IND-CCA2 secure in these cases, but it has been studied as a property of independent interest.

Raimondo *et al.* gave a construction of an authentication and key exchange protocol which relies on the plaintext awareness of an underlying encryption scheme [30]. Apart from this and the original OAEP papers, plaintext awareness has remained mostly of theoretical interest. Of the theoretical literature, one of the more interesting results is by Teranishi and Ogata [34], who proved that any encryption scheme which is one way and PA2 plaintext aware must be IND-CPA (and hence IND-CCA2) secure. We will return to and extend this result in Chapter2.

1.1 Probability and Statistics

1.1.1 Statistical Distance

Statistical distance is a metric for comparing two probability distributions. It is sometimes useful in provable security because substituting one random variable with another in any experiment will change the success probability by at most the statistical distance between the two variables, a property we prove as Lemma 1.2.1.

Definition 1.1.1 (Statistical Distance)**.** *Let x and y be random variables taking values on a finite set S. We define the statistical distance between x and y as*

$$\Delta[x, y] = \frac{1}{2} \sum_{s \in S} |\Pr[x = s] - \Pr[y = s]| .$$

Lemma 1.1.2. *Let x and y be random variables taking values on a finite set S, and let $T \subseteq S$. Then*

$$\Delta[x, y] \geq |\Pr[x \in T] - \Pr[y \in T]| .$$

Additionally, there exists a set T' such that

$$\Delta[x, y] = |\Pr[x \in T'] - \Pr[y \in T']| .$$

The proof of this is straight-forward and is given as Theorem 6.15 of [33].

Lemma 1.1.3. *If f is a function on the set S, then the following inequality holds:*

$$\Delta[\Pr[f(x)], \Pr[f(y)]] \leq \Delta[x, y] .$$

The proof of this is straight-forward and is given as Theorem 6.16 of [33].

Lemma 1.1.4. *Let $x_1, \ldots, x_\ell, y_1, \ldots, y_\ell$ be mutually independent random variables. Then*

$$\Delta[(x_1, \ldots, x_\ell), (y_1, \ldots, y_\ell)] \leq \sum_{i=1}^{\ell} \Delta[x_i, y_i] .$$

The proof of this theorem is given as Theorem 6.18 of [33].

1.1.2 Conditional Probability and Experiments

We will use the following notation for *experiments*. For an event A and a sequence of statements s, $\Pr[A : s]$ denotes the probability that event A occurs after running s. For example, $P = \Pr[x = 0 : y \xleftarrow{R} \{0,1\}^n; x \leftarrow M(y)]$ means: let $y \xleftarrow{R} \{0,1\}^n$, let $x \leftarrow M(y)$, and let $P = \Pr[x = 0]$. This notation should not be confused with the conditional probability $\Pr[A|B]$ which is the probability that event A occurs given that event B occurs.

1.2 Algorithms and Complexity

Formally, we define an algorithm as a Turing machine which halts on all input. However, for practical reasons we note that any reasonable modern programming language is Turing complete, so we will specify algorithms in pseudo-code that resembles high-level compiled or interpreted languages. For simplicity, we will assume that all computations are done in binary, and all inputs and outputs of algorithms are represented as binary strings. We will also assume that tuples of binary strings can be encoded in an unambiguous way, so that, for example, $(001, 00)$ is distinct from $(00, 100)$.

An algorithm may be deterministic or probabilistic. A deterministic algorithm uses no randomness, while a probabilistic algorithm requires access to a source of randomness. We will model this as a Turing machine with, in addition to its working tape(s), a read-only tape which is initialised with an infinite sequence of bits chosen uniformly and independently at random, which we will call a *random tape*. We present notation used for algorithms in Table 1.2.

Notation	Meaning
$x \leftarrow value$	Variable x is assigned the value $value$.
$x \xleftarrow{R} S$	Variable x is assigned a value chosen uniformly at random from the finite set S.
$y \leftarrow \mathcal{A}(x)$	Variable y is assigned the output of probabilistic algorithm $\mathcal{A}$ when run on input x with a random tape $R[\mathcal{A}]$, an infinite string of bits chosen independently at random from $\{0,1\}$. We may consider deterministic algorithms to be probabilistic algorithms which simply do not use their random tape.
$y \leftarrow \mathcal{A}^{\mathcal{O}}(x)$	Variable y is assigned the output of algorithm $\mathcal{A}$ when run on input x and with access to the oracle $\mathcal{O}$.
$y \leftarrow \mathcal{A}(x;R)$	Variable y is assigned the output of probabilistic algorithm $\mathcal{A}$ when run on input x with the specific random tape R. When using this notation we must ensure that R is sufficiently long.
$y \leftarrow \mathcal{A}^{\mathcal{O}}(x;R)$	Variable y is assigned the output of probabilistic algorithm $\mathcal{A}$ when run on input x with the specific random tape R and with access to the oracle $\mathcal{O}$. When using this notation we must ensure that R is sufficiently long.

Table 1.1: Notation for Algorithms

Algorithms may run other algorithms as *subroutines*. The controlling algorithm may "suspend" an algorithm that is running as a subroutine, by saving the subroutine's working tapes and head state into a variable. In principle the controlling algorithm could do this in response to any event, such as after a fixed number of operations, but typically the event in question will be a particular oracle query. We write this as follows:

function $\mathcal{B}(x)$
  Run $x \leftarrow \mathcal{A}(y)$
    if Trigger Event Occurs **then**
      $\mathsf{state} \leftarrow$ **Suspend**$(\mathcal{A})$
  $x \leftarrow$ **Resume**$(\mathcal{A}, \mathsf{state})$

This ability may be used to "rewind" an algorithm but in this work it will be mainly useful to pass the working state of a subroutine between algorithms. For ease of notation, we will sometimes set a state variable, for example $\mathsf{state} \leftarrow (x, y, z; R)$, and call $\mathbf{Resume}(\mathcal{A}, \mathsf{state})$. When we do this, we mean the same as running $\mathcal{A}$ on inputs (x, y, z) with random tape R.

Sometimes we will consider oracle-algorithms, i.e. algorithms which have access to one or more *oracles*. The algorithm may "query" an oracle, by writing an input to a dedicated "oracle tape", then changing to a particular head state. In a single step, the oracle will then compute its output value and write that to the oracle tape. Sometimes the value returned by the oracle could be computed from the inputs given to the algorithm itself, but there may be no known way for the algorithm to do so in polynomial-time. On the other hand, we will sometimes consider oracles that return values that are independent of all other inputs to the algorithm. When writing an algorithm that has access to an oracle $\mathcal{O}$, we will write "$y \leftarrow$ Query $\mathcal{O}(x)$" to denote the action of calling the oracle on input x and storing the result in variable y. We use this notation to distinguish an oracle call from running an algorithm as a subroutine. Algorithms which run oracle-algorithms as subroutines must respond to all oracle queries. We write this as follows:

Run $x \leftarrow \mathcal{A}^{\mathcal{O}}(y)$
  if $\mathcal{A}$ queries $\mathcal{O}(z)$ **then**
    Do Something
    return w

We will sometimes consider stateful algorithms. This means that the algorithm takes a value state as one of its inputs, and returns an updated value of state as part of its output. This should not be confused with suspending and resuming an algorithm that is being run as a subroutine.

Lemma 1.2.1. *Let $\mathcal{A}$ be an algorithm which takes inputs $(x, y_1, \ldots, y_n)$, and random tape R where x is independent of $(y_1, \ldots, y_n, R)$. Then for any random variables $x, x', y_1, \ldots, y_n$, $\Delta[\mathcal{A}(x, y_1, \ldots, y_n; R), \mathcal{A}(x', y_1, \ldots, y_n; R)] \leq \Delta[x, x']$.*

Proof. Since R is fixed, $\mathcal{A}(\cdot; R)$ defines a function. By Lemma 1.1.3,

$$\Delta[\mathcal{A}(x, y_1, \ldots, y_n; R), \mathcal{A}(x', y_1, \ldots, y_n; R)] \leq \Delta[(x, y_1, \ldots, y_n, R), (x', y_1, \ldots, y_n, R)] .$$

By Lemma 1.1.4,

$$\Delta[(x, y_1, \ldots, y_n, R), (x', y_1, \ldots, y_n, R)] \leq \Delta[x, x'] ,$$

since

$$\Delta[(y_1, \ldots, y_n, R), (y_1, \ldots, y_n, R)] = 0 .$$

Putting it together we see that:

$$\Delta[\mathcal{A}(x, y_1, \ldots, y_n; R), \mathcal{A}(x', y_1, \ldots, y_n; R)] \leq \Delta[x, x'] .$$

□

Variable Types

All variables are either integers, strings or lists. We do not explicitly specify variable types, but employ common sense to determine what type a variable is[1]. Unless otherwise specified, integers are all initialised to 0, strings initialised to the empty string ε, and lists initialised to the empty list ().

[1]Computer scientists sometimes call this "duck typing", presumably on the basis that if it walks like a duck and quacks like a duck, it's safe to cast it to an integer.

Lists

We write $\mathsf{List} = (x_1, x_2, \ldots, x_n)$ to specify a list consisting of the values $x_1, \ldots, x_n$. We write "Append x to List" to append the value x to the end of the list List. $\mathsf{List}[k]$ denotes the k^{th} entry of the list. The first element of the list is $\mathsf{List}[1]$ – there is no $\mathsf{List}[0]$ – and $|\mathsf{List}|$ to denotes the number of entries of the list or, equivalently, the index of the final entry.

1.2.1 Notation for computational complexity

Given functions $f, g : S \to \mathbb{R}$, where S is either $\mathbb{R}$ or $\mathbb{Z}$, and $g(x)$ is non-negative for all $x \in S$ we define the following:

Definition 1.2.2 (Big and Little O-notation)**.** *$f = O(g)$ if there exists $c, d \in \mathbb{R}$ such that $|f(x)| \leq cg(x)$ for all $x \geq d$. $f = o(g)$ if $f(x)/g(x) \to 0$ as $x \to \infty$*

Definition 1.2.3 (Negligible)**.** *A function f is negligible if for all $n \in \mathbb{N}$, $f = O\left(\frac{1}{x^n}\right)$. We say that a function $f(x, y)$ is negligible in x if the function $g_y(x) = f(x, y)$ is negligible. Note that f may be negligible in x for some values of y but not others.*

Definition 1.2.4 (Running Time)**.** *We say that an algorithm $\mathcal{A}$ has running time $T_{\mathcal{A}}(\lambda)$ if for all $x \in \{0,1\}^\lambda$ and for all random tapes $R \in \{0,1\}^\infty$, $\mathcal{A}(x; R)$ terminates within $T_{\mathcal{A}}(\lambda)$ steps, and there exists $x \in \{0,1\}^\lambda$ and $R \in \{0,1\}^\infty$ such that $\mathcal{A}(x; R)$ takes $T_{\mathcal{A}}(\lambda)$ steps to complete. This latter condition is to ensure that the running time is uniquely defined.*

The number of random bits used by a probabilistic algorithm is bounded by its running time, since reading each random bit uses one operation. This means that if we run a probabilistic algorithm $\mathcal{A}$ on input x, we may use a

finite random string $R \in \{0,1\}^m$ as the random tape, as long as we ensure that it is sufficiently long; typically we will chose m to be the running time $T_{\mathcal{A}}(|x|)$.

Definition 1.2.5 (Polynomial-Time)**.** *An algorithm $\mathcal{A}$ is* polynomial-time *if $T_{\mathcal{A}}(\lambda) = O(\lambda^n)$ for some $n \in \mathbb{N}$.*

Throughout this work, we will describe the running time of algorithms in terms of the security parameter, λ, rather than the length of the input to the algorithms. In practise, this does not make any difference, since in all our experiments at least one of the algorithms takes as input a string of length λ, and all subsequent algorithms take outputs from previous algorithms as their inputs. Since we only distinguish between polynomial-time and super polynomial-time algorithms, and the set of polynomials is closed under composition, each algorithm is polynomial-time in the input length if and only if it is polynomial-time in the security parameter.

1.3 Public-Key Encryption Schemes

1.3.1 Definitions and Notation

A public-key encryption scheme, originally defined by Diffie and Hellman [15], consists of three algorithms:

- KeyGen: A probabilistic polynomial-time algorithm which takes a security parameter 1^λ as input and returns a pair of keys (pk, sk). The public key pk is used by the sender to encrypt, and the private key sk is used by the recipient to decrypt. Each public key is associated with a message space $\mathbf{MsgSp}(pk)$ and a ciphertext space $\mathbf{CiphSp}(pk)$.

- Encrypt: A probabilistic polynomial-time algorithm which takes a pair (pk, m) as input, where pk is a public key and $m \in \mathbf{MsgSp}(pk)$ is a message, and returns a ciphertext $C \in \mathbf{CiphSp}(pk)$.

- Decrypt: A probabilistic polynomial-time algorithm which takes a pair (sk, C) as input, where sk is a private key and $C \in \mathbf{CiphSp}(pk)$ is a ciphertext, and returns a message $m \in \mathbf{MsgSp}(pk)$ or the distinguished "reject" symbol $\perp$, indicating the ciphertext is invalid.

Since $|pk|$ and $|sk|$ are both polynomially bounded in λ, and the set of polynomials is closed under composition, it follows that the running time of Encrypt and Decrypt is bounded by a polynomial in λ. Of course, there is no lower bound on $|pk|$, but for any algorithm KeyGen, we may trivially modify it to pad pk and sk with the value 1^λ. In some sense, this means it does not matter whether we consider Encrypt to be polynomially bounded in $|pk|$ or λ. We also note that while the running time of Encrypt may be polynomial in $|m|$, for any practical encryption scheme the running time of Encrypt will be at most linear in λ.

$\mathbf{MsgSp}(pk)$ and $\mathbf{CiphSp}(pk)$ should be polynomial-time decidable, i.e. there is a polynomial-time algorithm which takes a public key pk and bitstring m as input, and returns 1 if $m \in \mathbf{MsgSp}(pk)$ and 0 otherwise, and similarly for $\mathbf{CiphSp}(pk)$. Note that we did not require that for all $C \in \mathbf{CiphSp}(pk)$, there exists $m \in \mathbf{MsgSp}(pk)$ and $r \in \{0,1\}^*$ such that $\mathsf{Encrypt}(pk, m; R) = C$. In other words, we think of $\mathbf{CiphSp}(pk)$ more like the co-domain of the encryption algorithm than the range.

A public-key encryption scheme must satisfy the following soundness property: For all $\lambda \in \mathbb{N}$, for all $(pk, sk) \in (\{0,1\}^*)^2$ such that $\Pr[\mathsf{KeyGen}(1^\lambda) =$

$(pk, sk)] > 0$ and for all $m \in \mathbf{MsgSp}(pk)$:

$$\Pr[\mathsf{Decrypt}(sk, \mathsf{Encrypt}(pk, m)) = m] = 1\,.$$

1.3.2 Security of Public-Key Encryption Schemes

We will define notions of security in terms of experiments performed on adversaries. An adversary is modelled as a probabilistic polynomial-time algorithm, or in some cases a tuple of such algorithms. We say that $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_n)$ is polynomial-time if $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are all polynomial-time.

Indistinguishability of Ciphertexts

The IND-CPA definition was proposed by Goldwasser and Micali [18], who called it polynomial security. Naor and Yung [25] proposed the IND-CCA1 model, and the IND-CCA2 model was proposed by Rackoff and Simon [29]. An adversary against the IND security of a public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a pair of probabilistic polynomial-time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ takes a public key pk as input, and returns two messages $m_0, m_1 \in \mathbf{MsgSp}(pk)$ such that $|m_0| = |m_1|$, and some state information state and $\mathcal{A}_2$ takes a ciphertext $C^* \in \mathbf{CiphSp}(pk)$ and the value state that was returned by $\mathcal{A}_1$ as input and returns a single bit b'. We now describe the IND experiment:

$\mathbf{Expt}^{\mathrm{IND-ATK-b}}_{\Pi,\mathcal{A}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk, \cdot)}(pk)$
  $C^* \leftarrow \mathsf{Encrypt}(pk, m_b)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk, \cdot)}(C^*, \mathsf{state})$
  return b'

$\mathcal{A}_1$ and $\mathcal{A}_2$ have access to an oracle $\mathsf{Decrypt}$, which takes a ciphertext $C \in \mathbf{CiphSp}(pk)$ and returns $\mathsf{Decrypt}(sk, C)$. We consider three attack models. In the chosen plaintext attack (CPA) model, neither $\mathcal{A}_1$ nor $\mathcal{A}_2$ may make any decryption oracle queries. In the (non-adaptive) chosen ciphertext attack model (CCA1 or lunch-time attack), $\mathcal{A}_1$ may query the decryption oracle $\mathsf{Decrypt}$ but $\mathcal{A}_2$ may not query the decryption oracle. In the adaptive chosen ciphertext attack (CCA2) model, $\mathcal{A}_1$ and $\mathcal{A}_2$ may both query the decryption oracle, but $\mathcal{A}_2$ may not query the decryption oracle on C^*.

We define the IND-ATK advantage of $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{IND-ATK}}(\lambda) = |\Pr[\mathbf{Expt}_{\Pi,\mathcal{A}}^{\mathrm{IND-ATK-1}}(\lambda) = 1] - \Pr[\mathbf{Expt}_{\Pi,\mathcal{A}}^{\mathrm{IND-ATK-0}}(\lambda) = 1]|$$

where $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$.

Definition 1.3.1 (IND-ATK)**.** *A public key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is IND-ATK secure if for any polynomial-time IND-ATK adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{IND-ATK}}(\lambda)$ *is negligible in* λ*.*

One-Wayness

This definition of a one-way encryption scheme was originally proposed by Rabin [28]. An adversary against the one-wayness of an encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with a finite message space is a probabilistic polynomial-time algorithm $\mathcal{A}$ which takes a public key pk and ciphertext $C \in \mathbf{CiphSp}(pk)$ as input, and returns a message $m' \in \mathbf{MsgSp}(pk)$. We now describe the OW experiment for an adversary $\mathcal{A}$:

$\mathbf{Expt}_{\Pi,\mathcal{A}}^{\mathrm{OW-CPA}}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $m \xleftarrow{\mathrm{R}} \mathbf{MsgSp}(pk)$

$C^* \leftarrow \mathsf{Encrypt}(pk, m)$
$m' \leftarrow \mathcal{A}(pk, C^*)$
if $m' = m$ **then**
    return 1
else
    return 0

We define the advantage of $\mathcal{A}$ as:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathrm{OW-CPA}}(\lambda) = \Pr[\mathbf{Expt}_{\Pi,\mathcal{A}}^{\mathrm{OW-CPA}}(\lambda) = 1]\,.$$

Definition 1.3.2 (OW-CPA)**.** *A public key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is OW-CPA secure if for any polynomial-time OW-CPA adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}_{\mathcal{A}}^{\mathrm{OW-CPA}}(\lambda)$ *is negligible in* λ*.*

One could give corresponding definitions for OW-CCA1 and OW-CCA2 but we will not need them in this work.

1.3.3 Plaintext Awareness

Loosely speaking, Bellare and Rogaway defined an encryption scheme to be plaintext aware in the random oracle model if for any algorithm $\mathcal{A}$ (known as a ciphertext creator) that outputs a ciphertext C, there is another algorithm $\mathcal{A}^*$ (known as a plaintext extractor) that takes C and the list of random oracle queries made by $\mathcal{A}$ as input and returns the underlying message, if there is one. This definition could be adapted to the standard model, since the plaintext extractor relies on the random oracle queries, so in Bellare and Palacio's standard model definition [3], they supply the random tape of $\mathcal{A}$ to the ciphertext creator. Their definitions are as follows:

PA1 Plaintext Awareness

Formally, we consider two experiments. In both cases, the ciphertext creator $\mathcal{A}$ is given a public key pk and outputs a bitstring x. The games are distinguished by the "decryption oracle" to which $\mathcal{A}$ has access. $\mathcal{A}$ may query the decryption oracle on any ciphertext $C \in \mathbf{MsgSp}(pk)$. In the PA1-Real experiment, the oracle will return $\mathsf{Decrypt}(sk, C)$. In the PA1-Fake experiment, the oracle will compute $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{state}_{\mathcal{A}^*})$, where $\mathsf{state}_{\mathcal{A}^*}$ is $\mathcal{A}^*$'s state variable and $R[\mathcal{A}]$ is the random tape of $\mathcal{A}$.[2] The oracle then returns m to $\mathcal{A}$.

The encryption scheme is plaintext aware if the string x_{real} returned by $\mathcal{A}$ in the real experiment is computationally indistinguishable from the output x_{fake} of $\mathcal{A}$ in the fake experiment, or in other words, no polynomial time distinguishing algorithm D can distinguish x_{real} from x_{fake}. We present these two experiments below:

$\mathbf{Expt}^{\text{PA1-Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  Run $x_{\text{real}} \leftarrow \mathcal{A}^{\mathsf{Decrypt}}(pk)$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      $m \leftarrow \mathsf{Decrypt}(sk, C)$
      return m
  $b \leftarrow D(x_{\text{real}})$
  return b

$\mathbf{Expt}^{\text{PA1-Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  Run $x_{\text{fake}} \leftarrow \mathcal{A}^{\mathsf{Decrypt}}(pk)$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{state}_{\mathcal{A}^*})$
      return m
  $b \leftarrow D(x_{\text{fake}})$
  return b

[2]Strictly speaking, we define $R[\mathcal{A}]$ as the first $T_{\mathcal{A}}(\lambda)$ bits of the random tape, where $T_{\mathcal{A}}$ is the running time of $\mathcal{A}$. However, we will refer to this as simply the random tape of $\mathcal{A}$ for clarity.

Definition 1.3.3 (PA1 Plaintext Awareness)**.** *A public-key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is PA1 plaintext aware if for all PA1 ciphertext creators* $\mathcal{A}$ *there exists a polynomial-time plaintext extractor* $\mathcal{A}^*$ *such that for all polynomial-time distinguishing algorithms* D*, the advantage*

$$\mathbf{Adv}^{\mathrm{PA1}}_{\mathcal{A},\mathcal{A}^*,D}(\lambda) = |\Pr[\mathbf{Expt}^{\mathrm{PA1-Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathrm{PA1-Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda) = 1]|$$

is negligible in λ*.*

PA2 Plaintext Awareness

Bellare and Palacio [3] addressed the need to consider ciphertexts obtained from third parties in their standard model formulation of plaintext awareness by defining a a standard model version of PA2 plaintext awareness. Like the random oracle model definition, the model is augmented with an $\mathsf{Encrypt}$ oracle and an algorithm $\mathcal{P}$ called a plaintext creator. $\mathcal{P}$ takes a public key pk, a bitstring s and a state variable $\mathsf{state}_{\mathcal{P}}$ as input, and returns a message $m \in \mathbf{MsgSp}(pk)$ and a new state $\mathsf{state}_{\mathcal{P}}$. The $\mathsf{Encrypt}$ oracle takes a string s as input, and computes $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$. It then computes $C \leftarrow \mathsf{Encrypt}(pk, m)$, appends C to a list Clist and returns C. This ensures that $\mathcal{A}$ may influence the distribution of messages (by choosing inputs for $\mathcal{P}$) without necessarily knowing the exact message that was encrypted. $\mathcal{A}$ may also make $\mathsf{Decrypt}$ queries as in the PA1 definition, but it may not query $\mathsf{Decrypt}(C)$ for any $C \in \mathsf{Clist}$. In the PA2-Fake experiment, the plaintext extractor $\mathcal{A}^*$ is also given the list Clist to enable it to follow the execution of $\mathcal{A}$ as before.

We present the PA2 experiments below:

$\mathbf{Expt}^{\text{PA2-Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$m \leftarrow \mathsf{Decrypt}(sk, C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
return C
$b \leftarrow D(x)$
return b

$\mathbf{Expt}^{\text{PA2-Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
return C
$b \leftarrow D(x)$
return b

Definition 1.3.4 (PA2 Plaintext Awareness)**.** *A public key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is PA2 plaintext aware if for all polynomial-time ciphertext creators* $\mathcal{A}$*, there exists a polynomial-time plaintext extractor* $\mathcal{A}^*$ *such that for all polynomial-time plaintext creators* $\mathcal{P}$ *and polynomial-time distinguishing algorithms* D*, the advantage*

$$\mathbf{Adv}^{\text{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda) = |\Pr[\mathbf{Expt}^{\text{PA2-Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\text{PA2-Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda) = 1]|$$

is negligible in λ*.*

The following theorem was proved by Bellare and Palacio [3]. We call this theorem the fundamental theorem of plaintext awareness, because it provided the original motivation for the study of plaintext awareness[3].

Theorem 1.3.5 (Fundamental Theorem of Plaintext Awareness)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme. If* Π *is IND-CPA secure and PA2 plaintext aware then it is IND-CCA2 secure.*

Remark 1.3.6. The definitions above are sometimes collectively known as *computational* plaintext awareness, in contrast with stronger notions where the output of the ciphertext creator must be statistically or perfectly indistinguishable in the real and fake experiments. In this work, we focus only on computational notions of plaintext awareness, as this is the weakest (and thus easiest to achieve) of these three definitions, but it is sufficient for the fundamental theorem to hold.

Dent [14] introduced the following modified definition which he used to prove the PA2 plaintext awareness of the Cramer-Shoup encryption scheme.

Definition 1.3.7 (PA+)**.** *For any plaintext awareness definition PA (PA1, PA2), we define a new definition PA+ (PA1+, PA2+) by adding a randomness oracle* Randomness*, which takes no input and returns a random bit. The plaintext extractor is altered so that it takes a list* Rlist *of all bits returned by the randomness oracle as one of its inputs, i.e.* $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$.

Note that any PA+ definition is at least as strong as the corresponding PA definition, since a PA plaintext creator may be regarded as a PA+ plaintext creator which does not use the randomness oracle.

[3]More accurately, the corresponding "fundamental theorem of random oracle model plaintext awareness" motivated the study of plaintext awareness, but we do not study the random oracle model plaintext awareness here.

The idea behind this definition is that in the standard PA2 definition the plaintext extractor, which knows the code of the adversary and its randomness, can determine the past and present states of the adversary. Interestingly, because it knows the entire random tape of the adversary it can predict what the adversary would do after being given a particular message by the plaintext creator. In principle, this allows the plaintext creator to chose its responses based on what the adversary will do with them. The randomness oracle prevents the plaintext extractor from using this sort of strategy, because the adversary can get new randomness that the plaintext extractor has not yet seen.

These definitions parallel two possible interpretations of probabilistic algorithms. We defined probabilistic algorithms as Turing machines that have access to a tape containing randomly selected bits, but alternatively, we could have defined them as Turing machines with a randomness oracle, or an operation that writes a random bit into the current tape square. For the purposes of algorithms operating in isolation, it makes no difference which interpretation we choose; in the setting of plaintext awareness where the plaintext extractor gets access to the randomness used by the ciphertext creator, the difference emerges. We will relate PA2 with PA2+ in Chapter 2.

1.3.4 Special-Form: Notational Convenience for Plaintext Awareness

When constructing a plaintext extractor $\mathcal{A}^*$ for a plaintext creator $\mathcal{A}$, it will occasionally be useful to give standard names to the following variables:

- Elist is the list of encryption queries $\mathcal{A}$ makes.

- Clist is the corresponding list of ciphertexts returned by the encryption oracle.
- Dlist is the list of decryption queries $\mathcal{A}$ makes.
- Mlist is the corresponding list of messages returned by the decryption oracle.

Of these, only Clist is given as an input to the plaintext extractor in the PA2 experiment. However, all of these values can be computed from the information available to the plaintext extractor in a standard way. In order to show that an encryption scheme is PA2 plaintext aware, it suffices to construct a plaintext extractor $\hat{\mathcal{A}}^*$ of a special-form[4] which takes as input $pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{Elist}, \mathsf{Dlist}, \mathsf{Mlist}$ and a state variable $\mathsf{state}_{\hat{\mathcal{A}}^*}$, and returns a message $m \in \mathbf{MsgSp}(pk)$ and a new state $\mathsf{state}_{\hat{\mathcal{A}}^*}$. In some cases, passing these values as input can make describing the plaintext extractor substantially clearer, so although we only use this technique once, we believe it is potentially useful for other plaintext awareness proofs so we have included it on its own merit.

Given a special-form plaintext extractor $\hat{\mathcal{A}}^*$, we construct a normal-form plaintext extractor $\mathcal{A}^*$ as follows:

function $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}^*_{\mathcal{A}})$
  if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
    $\mathsf{state}_{\mathcal{A}} \leftarrow (pk; R)$
  else
    Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(n_r, n_e, n_d, \mathsf{Elist}, \mathsf{Dlist}, \mathsf{Mlist}, \mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\hat{\mathcal{A}}^*})$
  Resume $\mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt},\mathsf{Randomness}}(\mathsf{state}_{\mathcal{A}})$
  if $|\mathsf{Mlist}| > 0$ **then**
    Return $\mathsf{Mlist}[|\mathsf{Mlist}|]$ to $\mathcal{A}$'s previous query.

[4]We distinguish special-form plaintext extractors by making them wear dunce hats, because they are too lazy to compute these values for themselves.

if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$n_r \leftarrow n_r + 1$
return $\mathsf{Rlist}[n_r]$
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$n_e \leftarrow n_e + 1$
Append s to Elist
return $\mathsf{Clist}[n_e]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Append C to Dlist
$(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \hat{\mathcal{A}}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{Elist}, \mathsf{Dlist}, \mathsf{Mlist}, \mathsf{state}_{\hat{\mathcal{A}}^*})$
Append m to Mlist
$\mathsf{state}_{\mathcal{A}} \leftarrow \mathbf{Suspend}(\mathcal{A})$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow (n_r, n_e, n_d, \mathsf{Elist}, \mathsf{Dlist}, \mathsf{Mlist}, \mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\hat{\mathcal{A}}^*})$
return $(m, \mathsf{state}_{\mathcal{A}^*})$

We will now show by induction that the values of Elist, Dlist, and Mlist computed above are correct: i.e, they are the same as the encryption queries, decryption queries, and the corresponding responses to the decryption queries that were made by $\mathcal{A}$ in the $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\hat{\mathcal{A}}^*,\mathcal{P},D}$.

Suppose that the values computed by $\mathcal{A}^*$ prior to the the i^{th} decryption query are correct. $\mathcal{A}^*$ simulates a run of $\mathcal{A}$ using the same inputs and the same randomness, so it behaves the same as the instance of $\mathcal{A}$ ran by the challenger. In particular, the values of Elist and Dlist collected by $\mathcal{A}^*$ are identical to their real values in $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}$. This means that the inputs to $\hat{\mathcal{A}}^*$ are correct. Mlist is correct by definition, since it consists of the messages returned to $\mathcal{A}$ by the plaintext extractor.

We emphasise that a special-form plaintext extractor receives $R[\mathcal{A}]$ as usual, so it could in principle compute $\mathsf{Elist}, \mathsf{Dlist}$ and Mlist for itself. In particular, when using a special-form extractor in a proof, one may not substitute these values for computationally indistinguishable values, since $\hat{\mathcal{A}}^*$ may always verify their correctness using $R[\mathcal{A}]$.

1.4 Symmetric Encryption Schemes

We will need to consider symmetric encryption schemes when we study the plaintext-awareness properties of the Kurosawa-Desmedt scheme in Chapter 5. The definitions we use are based on those of Cramer and Shoup [12]. Cramer and Shoup called the primitive we describe *one-time symmetric-key encryption*, though it is now more commonly known as a *Data Encapsulation Mechanism* or DEM.

1.4.1 Definitions and Notation

A DEM Σ parametrised by a security parameter λ consists of two algorithms:

- A deterministic polynomial-time algorithm enc which takes a key $\kappa \in \{0,1\}^{\ell(\lambda)}$ and a message $m \in \mathbf{MsgSp}(\lambda)$ as input, and returns a ciphertext $\chi \in \mathbf{CiphSp}(\lambda)$.

- A deterministic polynomial-time algorithm dec which takes a key $\kappa \in \{0,1\}^{\ell(\lambda)}$ and a ciphertext $\chi \in \mathbf{CiphSp}(\lambda)$ as input, and returns a message $m \in \mathbf{CiphSp}(\lambda)$ or the distinguished reject symbol $\perp$, which indicates that the ciphertext is invalid.

where $\ell : \mathbb{N} \to \mathbb{N}$ is called a *key-length function*, $\mathbf{MsgSp}(\lambda)$ is the message space and $\mathbf{CiphSp}(\lambda)$ is the ciphertext space. Note that the message and ciphertext spaces depend only on the security parameter λ, unlike the case of public-key encryption where the message and ciphertext spaces may depend on the public key pk. As in the public-key case, $\mathbf{MsgSp}(\lambda)$ and $\mathbf{CiphSp}(\lambda)$ should be polynomial-time decidable.

We require that for all $\lambda \in \mathbb{N}$, for all $\kappa \in \{0,1\}^{\ell(\lambda)}$ and $m \in \mathbf{MsgSp}(\lambda)$,

$$\mathsf{dec}(\kappa, \mathsf{enc}(\kappa, m)) = m\,.$$

1.4.2 Security of DEMs

We will use two security properties of DEMs:

One-time IND-CCA2 Security of DEMs

An adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the one-time IND security of a DEM $\Sigma = (\mathsf{enc}, \mathsf{dec})$ is a pair of probabilistic polynomial-time algorithms where $\mathcal{A}_1$ takes a security parameter 1^λ as input and returns two messages $(m_0, m_1) \in \mathbf{MsgSp}(\lambda)$ such that $|m_0| = |m_1|$ and some state information state, and $\mathcal{A}_2$ takes input a ciphertext $\chi^* \in \mathbf{CiphSp}(\lambda)$ and the value state that was returned by $\mathcal{A}_1$ as input and returns a single bit b'. We now describe the OT-IND experiment:

$\mathbf{Expt}^{\mathrm{OT-IND-CCA2-b}}_{\Sigma,\mathcal{A}}(\lambda)$
  $\kappa \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
  $(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1()$
  $\chi^* \leftarrow \mathsf{enc}(\kappa, m_b)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{dec}(\kappa,\cdot)}(\chi^*, \mathsf{state})$
  return b'

$\mathcal{A}_2$ has access to an oracle dec, which takes a ciphertext $\chi \in \mathbf{CiphSp}(\lambda)$ as input and returns $\mathsf{dec}(\kappa, \chi)$. $\mathcal{A}_2$ may not query the decryption oracle on χ^*. Note that $\mathcal{A}_1$ does not have access to any oracles.

We define the one-time IND-CCA2 advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{OT-IND-CCA2}}_{\Sigma,\mathcal{A}}(\lambda) = |\Pr[\mathbf{Expt}^{\mathrm{OT-IND-CCA2-1}}_{\Sigma,\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathrm{OT-IND-CCA2-0}}_{\Sigma,\mathcal{A}}(\lambda) = 1]|\,.$$

Definition 1.4.1 (IND-CCA2)**.** *A DEM* $\Sigma = (\mathsf{enc}, \mathsf{dec})$ *is one-time IND-CCA2 secure if for any polynomial-time IND-CCA2 adversary* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{OT-IND-CCA2}}_{\Sigma,\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

Rejection Security

The Kurosawa-Desmedt [22] scheme also requires that the DEM satisfies the following condition:

Definition 1.4.2 (Rejection Security)**.** *A DEM* $\Sigma = (\mathsf{enc}, \mathsf{dec})$ *is* ϵ*-rejection secure if for all* $\lambda \in \mathbb{N}$*, and for all* $\chi \in \mathbf{CiphSp}(\lambda)$*,*

$$\Pr[\mathsf{dec}(\kappa, \chi) \neq \perp : \kappa \xleftarrow{R} \{0,1\}^{\ell(\lambda)}] \leq \epsilon(\lambda)\,.$$

This definition models the idea that if you take a ciphertext and attempt to decrypt it with a randomly-chosen key, you will obtain the output $\perp$ with overwhelming probability. This should not be confused with an integrity check; it may be possible for an adversary to take a ciphertext $C = \mathsf{enc}(\kappa, m)$ and compute a modified ciphertext C' which decrypts to some message $m' \neq \perp$ under the key κ.

1.5 Smooth Hash Functions

A *hash function* is a polynomial-time algorithm Hash which takes as input a security parameter λ and an element $x \in X_\lambda$ (which may depend on λ) and returns a bitstring $h \in \{0,1\}^{\ell(\lambda)}$ for some polynomial ℓ known as the output-length function. There are many properties that are expected of hash functions, including preimage-resistance, collision resistance and second preimage-resistance, but here we are only concerned with *smooth* hash functions.

Intuitively, a hash function is *smooth* if, when given a random input, the hash value is distributed uniformly at random on $\{0,1\}^{\ell(\lambda)}$.

Definition 1.5.1 (Smooth Hash Function)**.** *Let* Hash *be a hash function, and fix a security parameter* λ*. Let* $x \xleftarrow{R} X_\lambda$ *and* $s \xleftarrow{R} \{0,1\}^{\ell(\lambda)}$*. Then* Hash *is* δ*-smooth if* $\Delta(\mathsf{Hash}(\lambda, x), s) \leq \delta(\lambda)$*.*

Lemma 1.5.2. *Let* $\Sigma = (\mathsf{enc}, \mathsf{dec})$ *be an* ϵ*-rejection secure DEM and* Hash *be a* δ*-smooth hash function such that the key-length function of* Σ *equals the output-length function of* Hash*. Then for all* $\chi \in \mathbf{CiphSp}(\lambda)$*,*

$$\Pr[\mathsf{dec}(\kappa, \chi) \neq \perp \ : \ a \xleftarrow{R} X_\lambda; \kappa \leftarrow \mathsf{Hash}(a)] \leq \epsilon(\lambda) + \delta(\lambda)\,.$$

Proof. Fix a security parameter λ and a ciphertext $\chi \in \mathbf{CiphSp}(\lambda)$ and let $S = \{s \in \{0,1\}^{\ell(\lambda)} \mid \mathsf{dec}(s, \chi) \neq \perp\}$. Let $a \xleftarrow{\mathrm{R}} X_\lambda$, $\kappa \leftarrow \mathsf{Hash}(a)$ and $\kappa' \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$. Then by the smoothness of Hash and Lemma 1.1.2

$$\begin{aligned} \Pr[\mathsf{dec}(\kappa, \chi) \neq \perp] &= \Pr[\kappa \in S] \\ &\leq \Pr[\kappa' \in S] + \Delta(\kappa', \kappa) \\ &\leq \epsilon(\lambda) + \delta(\lambda)\,. \quad \square \end{aligned}$$

Chapter 2

Relations Among Notions of Plaintext Awareness

In this chapter we will propose several new definitions of PA2 plaintext awareness and will relate these new definitions to the existing ones.

2.1 Outline

Our aim for this chapter is to investigate the definition of PA2 plaintext awareness in the standard model. We will present a naive approach to achieving PA2 plaintext awareness without the full power of a plaintext creator, which we call PA2E, motivated by the belief that the arbitrary stateful plaintext creator makes full PA2 definition more complicated than it needs to be. This naive approach is inadequate, as Lemma 2.2.2 shows, because the combination of PA2E plaintext awareness and IND-CPA security does not imply IND-CCA2 security.

We will then introduce a new definition called 2PA2 plaintext awareness, which restricts the PA2 definition to the two plaintext creators used in Bellare

and Palacio's proof of the fundamental theorem of standard model plaintext awareness. Since this theorem motivates the study of plaintext awareness, it was crucial that our new definition is sufficient to give the same result.

Together, these sections show that our definition 2PA2 is in some sense close to minimal; it is adequate to prove the fundamental theorem of plaintext awareness, but Theorem 2.2.2 shows that if we remove the plaintext creator entirely, it no longer fulfils this requirement.

We would have liked to show that 2PA2 is in fact equivalent to PA2 plaintext awareness, at least for schemes which are IND-CPA secure. This idea seems reasonable, since if the encryption scheme is IND-CPA secure, then it shouldn't matter what message is encrypted. In Section 2.4, we will give this idea formal treatment, but are not able to prove this. Instead, we introduced the idea of length hiding; informally this captures the idea that an encryption scheme hides the length of a message as well as its contents. We will show that any scheme which is both length-hiding IND-CPA secure and 2PA2 is in fact PA2 plaintext aware.

We will investigate the idea of length-hiding and how it relates to plaintext awareness further in Section 2.6. Specifically we use the ideas of Teranishi and Ogata to show that any scheme which is IND-CPA, has a suitably large message space, and is PA2 plaintext aware is also length-hiding. This shows why we need length-hiding to prove our result in Section 2.4: any encryption scheme which is IND-CPA secure and PA2 plaintext awareness but not length-hiding is not PA2 plaintext aware by Theorem 2.6.2! 2PA2 does not appear to share this characteristic, since the stateful nature of the plaintext creator is key to proving Theorem 2.6.2.

Finally, we will use our definition of 2PA2 plaintext awareness to give a

rigorous proof of Dent's conjecture that PA2 plaintext awareness is stronger than PA1+ plaintext awareness. This works by showing that any scheme which is 2PA2 plaintext aware and IND-CPA secure is 2PA2+ plaintext aware, and we may use our previous theorems to complete the proof. Using 2PA2+ plaintext awareness simplified this proof, because we were able to construct an extractor $\mathcal{A}^*$ which relies on the particular behaviour of $\mathcal{P}_0$ and $\mathcal{P}_1$. In contrast, an extractor for PA2 plaintext awareness must work for all stateful plaintext creators $\mathcal{P}$.

2.2 Introducing PA2E

One of the more complex aspects of PA2 plaintext awareness is the fact that the encryption oracle returns an encryption of a message that has been chosen from some arbitrary distribution defined by $\mathcal{P}$. The order of the quantifiers in the definition of PA2 plaintext awareness means that neither the ciphertext creator $\mathcal{A}$, nor the plaintext extractor $\mathcal{A}^*$, know the distribution from which messages are chosen, although the ciphertext creator does have the ability to affect this distribution via its input s to the encryption oracle.

One approach to simplifying PA2 plaintext awareness might be to remove the plaintext creator from the definition entirely, and instead give the ciphertext creator access to a straightforward encryption oracle which computes $\mathsf{Encrypt}(pk, \cdot)$. Equivalently, we could consider the fixed plaintext creator $\mathcal{P}_E$ defined as follows:

function $\mathcal{P}_E(s)$
  return s

Definition 2.2.1 (PA2E)**.** *A public key encryption scheme* $\Pi = (\mathsf{KeyGen},$

$\mathsf{Encrypt}, \mathsf{Decrypt}$) *is PA2E plaintext aware if for all polynomial-time ciphertext creators* $\mathcal{A}$*, there exists a polynomial-time plaintext extractor* $\mathcal{A}^*$ *such that for all polynomial-time distinguishing algorithms* D*, the advantage*

$$\mathbf{Adv}^{\mathrm{PA2E}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda) = \mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_E,D}(\lambda)$$

is negligible in λ*.* $\mathbf{Expt}^{\mathrm{PA2E-Real}}_{\Pi,\mathcal{A},D}(\lambda)$ *and* $\mathbf{Expt}^{\mathrm{PA2E-Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda)$ *are defined in the natural way.*

Unfortunately, this definition is not strong enough. The minimum we require from any definition of plaintext awareness is for it to satisfy the fundamental theorem of plaintext awareness, Theorem 1.3.5. In other words any encryption scheme which is IND-CPA secure and plaintext aware should be IND-CCA2 secure. We will now show that PA2E does not meet this requirement.

Theorem 2.2.2. *Suppose that* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is an encryption scheme which is IND-CPA secure and PA2E plaintext aware. Then there exists an encryption scheme* Π' *which is IND-CPA secure and PA2E plaintext aware, but not IND-CCA2 secure.*

Proof. Let $\Pi' = (\mathsf{KeyGen}, \mathsf{Encrypt}', \mathsf{Decrypt}')$, where $\mathsf{Encrypt}'$ and $\mathsf{Decrypt}'$ are defined as follows:

function $\mathsf{Encrypt}'(pk, m)$
  $C' \leftarrow \mathsf{Encrypt}(pk, m)$
  $C \leftarrow C'||0$
  return C

function $\mathsf{Decrypt}'(sk, C)$
  Parse C as $C'||a$ where $a \in \{0, 1\}$
  $m \leftarrow \mathsf{Decrypt}(sk, C')$
  return m

Π' is not IND-CCA2, as an adversary may change the final bit of the challenge ciphertext C^* to a 1 and call the decryption oracle on the resulting ciphertext to obtain the underlying message, and recover b.

On the other hand, if $\mathcal{A}$ is an adversary against the IND-CPA security of Π' then there is an adversary $\mathcal{B}$ against the IND-CPA security of Π, which simply runs $\mathcal{A}$ and appends a 0 bit to the challenge ciphertext. $\mathcal{B}$ wins if and only if $\mathcal{A}$ does, so $\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{B}}(\lambda) = \mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi',\mathcal{A}}(\lambda)$, which is negligible in λ since Π was assumed to be IND-CPA secure.

We must now show that Π' is PA2E plaintext aware. Let $\mathcal{A}$ be an arbitrary PA2E ciphertext creator for Π'. We construct a PA2E ciphertext creator $\mathcal{B}$ for Π as follows.

function $\mathcal{B}(pk)$
  Run $x \leftarrow \mathcal{A}^{\mathsf{Encrypt}',\mathsf{Decrypt}'}(pk)$
    if $\mathcal{A}$ queries $\mathsf{Encrypt}'(m)$ **then**
      Query $C \leftarrow \mathsf{Encrypt}(m)$
      $C' \leftarrow C||0$
      Append m to Mlist
      Append C' to Clist
      return C'
    if $\mathcal{A}$ queries $\mathsf{Decrypt}'(C')$ **then**
      Parse C' as $C||a$ for some bit a
      if $a = 1$ and $C||0 = \mathsf{Clist}[i]$ for some i **then**
        return $\mathsf{Mlist}[i]$
      else
        Query $m \leftarrow \mathsf{Decrypt}(C)$
        return m
  return x

By the PA2E property of Π, there exists a plaintext extractor $\mathcal{B}^*$ for $\mathcal{B}$ such that $\mathbf{Adv}^{\mathrm{PA2E}}_{\Pi,\mathcal{B},\mathcal{B}^*,D}(\lambda)$ is negligible in λ for all distinguishing algorithms D. We use $\mathcal{B}^*$ to construct a special-form PA2E plaintext extractor $\hat{\mathcal{A}}^*$ for $\mathcal{A}$ as follows:

function $\hat{\mathcal{A}}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}, \mathsf{Elist}, \mathsf{Mlist}, \mathsf{Dlist}, \mathsf{state})$
  Parse C' as $C||a$ for some bit a
  for $i \leftarrow 1$ to $|\mathsf{Clist}|$ **do**
    Parse $\mathsf{Clist}[i]$ as $C_i||0$
    Append C_i to $\mathsf{Clist}_{\mathcal{B}}$

if $a = 1$ and $C||0 = \mathsf{Clist}[i]$ for some i **then**
  $m \leftarrow \mathsf{Mlist}[i]$
else
  $(m, \mathsf{state}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
return (m, state)

Let $\mathcal{A}^*$ be the corresponding normal-form plaintext creator as described in Section 1.3.4.

If $\mathcal{A}$ makes a decryption query $C'||1$, $\mathcal{B}$ avoids calling the $\mathsf{Decrypt}$ oracle on ciphertexts that been returned by the $\mathsf{Encrypt}$ oracle by checking Clist for the ciphertext $C'||0$. Since $\mathsf{Decrypt}'$ ignores the final bit, the corresponding plaintext must be the correct decryption for both $C'||0$ and $C'||1$. Note that all ciphertexts in Clist must end in 0 so we do not need to check the case $C'||0$ since $\mathcal{A}$ may not make such queries anyway. The plaintext extractor $\mathcal{A}^*$ mimics this behaviour to ensure the queries to $\mathcal{B}^*$ remain the same.

We must now show that $\mathbf{Adv}^{\mathrm{PA2E}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda)$ is negligible in λ for all distinguishing algorithms D, where $\mathcal{A}^*$ is the normal-form plaintext extractor constructed from $\hat{\mathcal{A}}^*$. The proof is structured as a sequence of four games, in the style of Shoup [32]. Throughout the proof, we use the symbol C to represent ciphertexts of the basic scheme Π and C' to represent the augmented ciphertexts belonging to the scheme Π'. We also name the oracles $\mathsf{Encrypt}'$ and $\mathsf{Decrypt}'$ to emphasise that they implement the encryption and decryption algorithms of the modified scheme Π' rather than the original. We fix a distinguishing algorithm D and let S_i be the event that D outputs 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\mathrm{PA2-Fake}}_{\Pi',\mathcal{A},\mathcal{A}^*,\mathcal{P}_E,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_0 \leftarrow \mathcal{A}^{\mathsf{Decrypt}',\mathsf{Encrypt}'}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Encrypt}'(m)$ **then**

$C \leftarrow \mathsf{Encrypt}(pk, m)$
$C' \leftarrow C||0$
return C'
if $\mathcal{A}$ queries $\mathsf{Decrypt}'(C')$ **then**
Parse C' as $C||a$ for some bit a
if $a = 1$ and $C||0 = \mathsf{Clist}[i]$ for some i **then**
return $\mathsf{Mlist}[i]$
else
$m \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist})$
return m
$b'_0 \leftarrow D(x_0)$
return b'_0

Game 1: Let Game 1 be $\mathbf{Expt}^{\mathrm{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_E,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_1 \leftarrow \mathcal{A}^{\mathsf{Decrypt}',\mathsf{Encrypt}'}(pk)$
if $\mathcal{A}$ queries $\mathsf{Encrypt}'(m)$ **then**
$C \leftarrow \mathsf{Encrypt}(pk, m)$
$C' \leftarrow C||0$
Append m to Mlist
Append C' to Clist
return C'
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C' as $C||a$ for some bit a
if $a = 1$ and $C||0 = \mathsf{Clist}[i]$ for some i **then**
return $\mathsf{Mlist}[i]$
else
$m \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist})$
return m
$b'_1 \leftarrow D(x_1)$
return b'_1

This is a bridging step; instead of Mlist and Clist being recorded by $\mathcal{A}^*$, it is recorded by $\mathcal{B}$, but otherwise there is no change, so

$$\Pr[S_1] = \Pr[S_0]\,.$$

Game 2: Let Game 2 be $\mathbf{Expt}^{\mathrm{PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P}_E,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_2 \leftarrow \mathcal{A}^{\mathsf{Decrypt}',\mathsf{Encrypt}'}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Encrypt}'(m)$ **then**
    $C \leftarrow \mathsf{Encrypt}(pk, m)$
    $C' \leftarrow C||0$
    Append m to Mlist
    Append C' to Clist
    return C'
  if $\mathcal{A}$ queries $\mathsf{Decrypt}'(C)$ **then**
    Parse C' as $C||a$ for some bit a
    if $a = 1$ and $C||0 = \mathsf{Clist}[i]$ for some i **then**
      return $\mathsf{Mlist}[i]$
    else
      $m \leftarrow \mathsf{Decrypt}(sk, C)$
      return m
$b'_2 \leftarrow D(x_2)$
return b'_2

$$\Pr[S_2] - \Pr[S_1] \leq \mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_E,D}(\lambda)$$

which is negligible in λ by the PA2E property of Π.

Game 3: Let Game 3 be $\mathbf{Expt}^{\mathrm{PA2-Real}}_{\Pi',\mathcal{A},\mathcal{P}_E,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_3 \leftarrow \mathcal{A}^{\mathsf{Decrypt}',\mathsf{Encrypt}'}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Encrypt}'(m)$ **then**
    $C \leftarrow \mathsf{Encrypt}(pk, m)$
    $C' \leftarrow C||0$
    return C'
  if $\mathcal{A}$ queries $\mathsf{Decrypt}'(C')$ **then**
    Parse C' as $C||a$ where a is a single bit.
    $m \leftarrow \mathsf{Decrypt}(sk, C)$
    return m
$b'_3 \leftarrow D(x_3)$
return b'_3

This is a bridging step; if the condition $a = 1$ and $C||0 = \mathsf{Clist}[i]$ ever

occurs, then $C||0 = \mathsf{Encrypt}'(pk, \mathsf{Mlist}[i])$ by definition. By the soundness property of Π, this implies that $\mathsf{Decrypt}(sk, C) = \mathsf{Mlist}[i]$ with probability 1. Thus the responses to $\mathcal{A}$'s queries are unchanged and so

$$\Pr[S_3] = \Pr[S_2]\,.$$

Putting it all together, we see that $\mathbf{Adv}^{\mathrm{PA2}}_{\Pi',\mathcal{A},\mathcal{A}^*,\mathcal{P}_E,D}(\lambda) = \mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_E,D}(\lambda)$, which is negligible in λ by assumption, hence we see that Π' is PA2E plaintext aware. □

This proof works precisely because in PA2E, the plaintext creator $\mathcal{A}$ simply submits a message to the encryption oracle for decryption. This allows $\mathcal{A}^*$ to determine the message underlying the returned ciphertext by looking at the queries $\mathcal{A}$ makes. In contrast, in the full PA2 model, $\mathcal{A}^*$ cannot determine the message based on the inputs to the encryption oracle because the same plaintext extractor must work for all plaintext creators $\mathcal{P}$.

Canetti *et al.* gave a modified IND-CCA2 definition [9], which they call relaxed chosen ciphertext security (IND-RCCA2), which improves upon earlier definitions by Shoup [31], Krawczyk *et al.* [21] and An *et al.* [1]. The goal of all of these models is to include a variety of schemes where the adversary can modify a ciphertext to produce another valid ciphertext which decrypts to the same value, such as the scheme Π' which we defined above. Shoup calls these schemes "benignly malleable", because they seem to be secure for most practical purposes, but would nevertheless be ruled out by the stricter IND-CCA2 definition.

We suspect that it is possible to adapt the results of Section 2.3 to the PA2E notion of plaintext awareness by using the IND-RCCA2 definition of

security, but we will focus on IND-CCA2 security in this work.

2.3 Introducing 2PA2

Let us now consider a slightly stronger definition of plaintext awareness, inspired by Bellare and Palacio's proof [3] that a scheme which is PA2 and IND-CPA secure must be IND-CCA2 secure. They make use of two plaintext creators, $\mathcal{P}_0$ and $\mathcal{P}_1$, as defined below:

function $\mathcal{P}_i(pk, s)$
    Parse s as (m_0, m_1)
    if parsing fails **then**
        return 0
    else
        return m_i

We have modified their definition slightly, so that behaviour on input s which is not of the form (m_0, m_1) is defined. Notice that $\mathcal{P}_i$ does not check whether $|m_0| = |m_1|$.

Definition 2.3.1 (2PA2)**.** *A public key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is 2PA2 plaintext aware if for all polynomial-time ciphertext creators* $\mathcal{A}$*, there exists a polynomial-time plaintext extractor* $\mathcal{A}^*$ *such that for all polynomial-time distinguishing algorithms* D*, the advantage*

$$\mathbf{Adv}^{\mathrm{2PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda) = \max\{\mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_0,D}(\lambda), \mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_1,D}(\lambda)\}$$

is negligible in λ*.*

Since the proof [3] of the fundamental theorem of plaintext awareness (Theorem 1.3.5) only makes use of the plaintext creators $\mathcal{P}_0$ and $\mathcal{P}_1$, the following modification to the fundamental theorem holds:

Theorem 2.3.2 (Bellare-Palacio)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme. If* Π *is IND-CPA secure and 2PA2 plaintext aware then it is IND-CCA2 secure.*

Definition 2.3.3 (2PA2+)**.** *We define 2PA2+ in the same way as we did for PA1+ and PA2+, by adding a randomness oracle, which takes no input and returns a random bit. The plaintext extractor is altered so that it takes a list* Rlist *of all such bits queried so far as one of its inputs, i.e.* $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist})$.

2.4 Length Hiding

We now introduce a generalisation of IND security, which we call Length Hiding, or LH-IND. This will be used when we come to relate 2PA2 with PA2 plaintext awareness.

An adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the LH-IND security of an encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a pair of probabilistic polynomial-time algorithms such that $\mathcal{A}_1$ takes a public key pk as input and returns two messages $m_0, m_1 \in \mathbf{MsgSp}(pk)$ and some state information state, and $\mathcal{A}_2$ takes a ciphertext $C^* \in \mathbf{CiphSp}(pk)$ and the value state that was output by $\mathcal{A}_1$ as input, and returns a single bit b'. This differs from the definition of an IND adversary only in that m_0 and m_1 may have different lengths. We define $\mathbf{Expt}^{\mathrm{LH-IND-ATK-b}}_{\Pi,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathrm{LH-IND-ATK}}_{\Pi,\mathcal{A}}(\lambda)$ for $b \in \{0,1\}$ and $ATK \in \{CPA, CCA2\}$ exactly as we did for their IND counterparts, except without the restriction that $|m_0| = |m_1|$.

Definition 2.4.1 (Length-Hiding IND-CPA security)**.** *An encryption scheme*

$\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is LH-IND-CPA secure if for any polynomial-time adversary LH-IND-CPA* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

Definition 2.4.2 (Length-Hiding IND-CCA2 security)**.** *An encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is LH-IND-CCA2 secure if for any polynomial-time adversary LH-IND-CCA2* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible in* λ*.*

This is a natural extension of the IND definition, because it captures the idea that it should be impossible to distinguish between encryptions of any two messages, not just those of equal length.

Remark 2.4.3. If an encryption scheme has the property that for all public keys pk, $\mathbf{MsgSp}(pk) = \{0,1\}^{\ell}$ for some $\ell = \ell(pk)$, then LH-IND-ATK (for ATK $\in$ {CPA, CCA2})is trivially equivalent to IND-ATK because the fact that $m_0, m_1 \in \mathbf{MsgSp}(pk)$ implies that $|m_0| = \ell(pk) = |m_1|$.

Remark 2.4.4. If an encryption scheme is LH-IND-ATK (for ATK $\in$ {CPA, CCA2}) secure then it is IND-ATK secure, because IND-ATK adversaries are a special case of LH-IND-ATK adversaries.

Lemma 2.4.5. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme with the infinite message space* $\{0,1\}^*$*. Then* Π *is not LH-IND-CPA secure.*

Proof. Let PK_λ be the set of public keys, i.e $PK_\lambda = \{pk \in \{0,1\}^* | \Pr[\mathsf{KeyGen}(1^\lambda) = (pk, sk) \text{ for some } sk] > 0\}$, let $t(\lambda)$ be an upper bound on the running time of $\mathsf{Encrypt}(pk, 0)$ for all $pk \in PK_\lambda$, and let $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$. We construct an LH-IND-CPA adversary as follows:

function $\mathcal{A}_1(pk)$
  $m_0 \leftarrow 0$
  $m_1 \xleftarrow{R} \{0,1\}^{t(\lambda)+\lambda+1}$
  return (m_0, m_1, ε)

function $\mathcal{A}_2(C^*, \mathsf{state})$
  if $|C^* > t(\lambda)|$ **then**
    return 1
  else
    return 0

$\Pr[\mathbf{Expt}^{\text{LH-IND-CPA-0}}_{\Pi,\mathcal{A}}(\lambda) = 0] = 1$, because $|C^*|$ is bounded by the running time $t(\lambda)$. On the other hand, there are at most $2^{t(\lambda)+1} - 1$ ciphertexts of length less than or equal to $t(\lambda)$, so if $m_1 \xleftarrow{R} \{0,1\}^{t(\lambda)+\lambda+1}$ then by the pigeonhole principle we see that $|\Pr[\mathsf{Encrypt}(pk, m_1) \leq t]| \leq 2^{t(\lambda)+1}/2^{t(\lambda)+\lambda+1} = 2^{-\lambda}$. So $\Pr[\mathbf{Expt}^{\text{LH-IND-CPA-1}}_{\Pi,\mathcal{A}}(\lambda) = 0] \leq 2^{-\lambda}$. Thus

$$\begin{aligned}\mathbf{Adv}^{\text{LH-IND-CPA}}_{\Pi,\mathcal{A}}(\lambda) &= |\Pr[\mathbf{Expt}^{\text{LH-IND-CPA-0}}_{\Pi,\mathcal{A}}(\lambda) = 0] \\ &\qquad - \Pr[\mathbf{Expt}^{\text{LH-IND-CPA-1}}_{\Pi,\mathcal{A}}(\lambda) = 0]| \\ &\geq 1 - 2^{\lambda}\end{aligned}$$

which is non-negligible in λ, so Π is not LH-IND-CPA secure. □

This fairly obvious result shows that any encryption scheme which allows arbitrarily long plaintexts cannot be LH-IND-CPA secure. It is of course not the goal of such systems to hide the message length, but we include this result to show that LH-IND security excludes an important class of encryption schemes, notably including any hybrid scheme which uses symmetric encryption to encrypt arbitrarily long messages.

Using the methodology of Bellare and Palacio [3], we now prove the analogue of Theorem 2.3.2 for LH-IND security. In Theorem 2.5.2 below we will

show that any encryption scheme which is 2PA2+ plaintext aware and LH-IND-CCA2 secure must be PA2+ plaintext aware; by combining these results we see that any encryption scheme which is 2PA2+ plaintext aware and LH-IND-CPA secure is PA2+ plaintext aware.

Theorem 2.4.6. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme. If* Π *is LH-IND-CPA secure and 2PA2 plaintext aware then it is LH-IND-CCA2 secure.*

Proof. Let $\mathcal{A}$ be an LH-IND-CCA2 adversary. We construct a 2PA2 ciphertext creator $\mathcal{B}$ and use $\mathcal{B}$ to construct an LH-IND-CPA adversary $\mathcal{C}$:

function $\mathcal{B}(pk)$
  $R_1 \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
  $R_2 \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
  Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
    if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
  Query $C^* \leftarrow \mathsf{Encrypt}(m_0, m_1)$
  Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
    if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
  return b'

By the 2PA2 property of Π there exists a plaintext extractors $\mathcal{B}^*$ such that for all distinguishing algorithms D, and for $i \in \{0,1\}$, $\mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_i D}(\lambda)$ is negligible in λ.

We use $\mathcal{B}$ and $\mathcal{B}^*$ to construct an LH-IND-CPA adversary $\mathcal{C}$:

function $\mathcal{C}_1(pk)$
  $R_1 \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
  $R_2 \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
  Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
    if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**

$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Cliststate}_{\mathcal{B}^*})$
return m
$\mathsf{state}_{\mathcal{C}} \leftarrow (\mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\mathcal{B}^*}, R_1, R_2)$
return $(m_0, m_1, \mathsf{state}_{\mathcal{C}})$

function $\mathcal{C}_2(C^*, \mathsf{state}_{\mathcal{C}})$
Parse $\mathsf{state}_{\mathcal{C}}$ as $(\mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\mathcal{B}^*}, R_1, R_2)$
$\mathsf{Clist} \leftarrow (C^*)$
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
return m
return b'

Note that $\mathcal{C}_2$ does use a random tape of its own, the randomness for $\mathcal{A}_2$ is supplied as part of $\mathsf{state}_{\mathcal{C}}$. This is because $\mathcal{B}^*$ takes the random tape of $\mathcal{B}$ as one of its inputs, which consists of $R_1||R_2$, so we must generate these values in advance. We define D to be the distinguishing algorithm which takes a single bit b as input and returns b.

We now use $\mathcal{B}$, $\mathcal{B}^*$ and $\mathcal{C}$ to prove that $\mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible in λ. The proof is structured as a sequence of games. We let S_i be the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\mathrm{LH-IND-CCA2-0}}_{\Pi,\mathcal{A}}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$C^* \leftarrow \mathsf{Encrypt}(pk, m_0)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

Game 1: Let Game 1 be $\mathbf{Expt}^{\mathrm{2PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P}_0,D}(\lambda)$. By writing $\mathcal{B}$ out in full, and recalling that D simply returns the bit it is given as input, we see that it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    $m \leftarrow \mathsf{Decrypt}(sk, C)$
    return m
$C^* \leftarrow \mathsf{Encrypt}(pk, \mathcal{P}_0(pk, m_0, m_1))$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    $m \leftarrow \mathsf{Decrypt}(sk, C)$
    return m
return b'

Since the operations performed are essentially unchanged, this is a bridging step, and we see that

$$\Pr[S_1] = \Pr[S_0]\,.$$

Game 2: Let Game 2 be $\mathbf{Expt}^{\text{2PA2}-\text{Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_0,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    $\boxed{(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})}$
    return m
$C^* \leftarrow \mathsf{Encrypt}(pk, \mathcal{P}_0(pk, m_0, m_1))$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    $\boxed{(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})}$
    return m
return b'

By the 2PA2 property of Π,

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\mathrm{2PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,D}(\lambda)\,.$$

Game 3: Let Game 3 be $\mathbf{Expt}^{\mathrm{LH-IND-CPA-0}}_{\Pi,\mathcal{C}}(\lambda)$. By writing $\mathcal{C}$ out in full, we see that it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
    return m
$\boxed{C^* \leftarrow \mathsf{Encrypt}(pk, m_0)}$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
    return m
return b'

Since $\mathcal{P}_0(pk, m_0, m_1) = m_0$the operations performed are essentially unchanged, this is a bridging step, and we see that

$$\Pr[S_3] = \Pr[S_2]\,.$$

Game 4: Let Game 4 be $\mathbf{Expt}^{\mathrm{LH-IND-CPA-1}}_{\Pi,\mathcal{C}}(\lambda)$. By writing $\mathcal{C}$ out in full, we see that it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$

return m
$\boxed{C^* \leftarrow \mathsf{Encrypt}(pk, m_1)}$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
return m
return b'

$$|\Pr[S_4] - \Pr[S_3]| \leq \mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)\,.$$

Game 5: Let Game 5 be $\mathbf{Expt}^{\mathrm{2PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_1,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\mathrm{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
return m
$\boxed{C^* \leftarrow \mathsf{Encrypt}(pk, \mathcal{P}_1(pk, m_0, m_1))}$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R_1||R_2, \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
return m
return b'

Since $\mathcal{P}_1(pk, m_0, m_1) = m_1$, the operations performed are essentially unchanged, this is a bridging step, and we see that

$$\Pr[S_5] = \Pr[S_4]\,.$$

Game 6: Let Game 6 be $\mathbf{Expt}^{\mathrm{2PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P}_1,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$R_1 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_1}(\lambda)}$
$R_2 \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}_2}(\lambda)}$
Run $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk; R_1)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    $\boxed{m \leftarrow \mathsf{Decrypt}(sk, C)}$
    return m
$C^* \leftarrow \mathsf{Encrypt}(pk, \mathcal{P}_1(pk, m_0, m_1))$
Append C^* to Clist
Run $b' \leftarrow \mathcal{A}_2(C^*, \mathsf{state}_{\mathcal{A}}; R_2)$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    $\boxed{m \leftarrow \mathsf{Decrypt}(sk, C)}$
    return m
return b'

By the 2PA2 property of Π,

$$|\Pr[S_6] - \Pr[S_5]| \leq \mathbf{Adv}^{\text{2PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,D}(\lambda)\,.$$

Game 7: Let Game 7 be $\mathbf{Expt}^{\text{LH}-\text{IND}-\text{CCA2}-1}_{\Pi,\mathcal{A}}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$C^* \leftarrow \mathsf{Encrypt}(pk, m_1)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

Since the operations performed are essentially unchanged, this is a bridging step, and we see that

$$\Pr[S_7] = \Pr[S_6]\,.$$

Putting it all together, we see

$$\begin{aligned}
\mathbf{Adv}^{\text{LH}-\text{IND}-\text{CCA2}}_{\Pi,\mathcal{A}}(\lambda) &= |\Pr[\mathbf{Expt}^{\text{LH}-\text{IND}-\text{CCA2}-1}_{\Pi,\mathcal{A}}(\lambda)] - \\
&\qquad \Pr[\mathbf{Expt}^{\text{LH}-\text{IND}-\text{CCA2}-0}_{\Pi,\mathcal{A}}(\lambda)]| \\
&= |\Pr[S_7] - \Pr[S_0]| \\
&\leq |\Pr[S_7] - \Pr[S_6]| + \ldots + |\Pr[S_1] - \Pr[S_0]| \\
&\leq 2\mathbf{Adv}^{\text{2PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,D}(\lambda) + \mathbf{Adv}^{\text{LH}-\text{IND}-\text{CPA}}_{\Pi,\mathcal{C}}(\lambda)
\end{aligned}$$

which is negligible in λ by the LH-IND-CPA and 2PA2 properties of Π. □

2.5 Relation between 2PA2 and PA2

Firstly, we note that 2PA2 is a special case of PA2.

Theorem 2.5.1. *If an encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is PA2 plaintext aware then it is 2PA2 plaintext aware.*

Proof. Let $\mathcal{A}$ be a polynomial-time ciphertext creator and let $\mathcal{P}_i$ (where $i \in \{0,1\}$) be as in the definition of 2PA2. Then by the PA2 property of Π there exists a polynomial-time plaintext extractor $\mathcal{A}^*$ such that for all polynomial-time distinguishing algorithms D, the advantage

$$\mathbf{Adv}^{\text{2PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda) = \max\{\mathbf{Adv}^{\text{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_0,D}(\lambda), \mathbf{Adv}^{\text{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_1,D}(\lambda)\}$$

is negligible in λ as required. □

It is more interesting to consider the question of whether 2PA2 plaintext awareness implies PA2 plaintext awareness. We will soon show that with certain extra conditions, this is the case. First we show that if an LH-IND-CCA2

encryption scheme is plaintext aware with respect to some fixed, stateless plaintext creator, then it is plaintext aware with respect to all plaintext creators.

Theorem 2.5.2. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be an encryption scheme which is LH-IND-CCA2 secure and suppose that there is some polynomial-time stateless plaintext creator* $\hat{\mathcal{P}}$ *such that for all polynomial-time ciphertext creators* $\mathcal{A}$ *there exists a ciphertext extractor* $\mathcal{A}^*$ *such that for all distinguishing algorithms* D, $\mathbf{Adv}^{\mathrm{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\hat{\mathcal{P}},D}(\lambda)$ *negligible in* λ. *Then* Π *is PA2+ plaintext aware.*

Remark 2.5.3. The conditions of this theorem are similar to the definition of PA2+ plaintext awareness but with two important relaxations. Firstly, $\hat{\mathcal{P}}$ is required to be stateless, and more importantly, the order of the quantifiers is reversed: $\mathcal{A}^*$ is only assumed to be a valid plaintext extractor for $\mathcal{A}$ with respect to some fixed plaintext creator $\hat{\mathcal{P}}$, instead of requiring that $\mathcal{A}^*$ works for all plaintext creators.

Proof. Consider an arbitrary PA2+ ciphertext creator $\mathcal{A}$. Let $\mathcal{A}^*$ be the plaintext extractor for $\mathcal{A}$ with respect to $\hat{\mathcal{P}}$, which exists by the assumption above. Let q_e be an upper bound for the number of encryption queries made by $\mathcal{A}$ and let q_d be an upper bound for the number of decryption queries made by $\mathcal{A}$. We will show that for any plaintext creator $\mathcal{P}$, and for any distinguishing algorithm D:

$$\mathbf{Adv}^{\mathrm{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda)$$

is negligible in λ.

We structure the proof as a sequence of games.

Let x_i be the output of $\mathcal{A}$ in Game i. Fix a distinguishing algorithm D and let S_i be the event that $D(x_i) = 1$.

Game 0: Let Game 0 be $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda)$, and let $\mathcal{A}^*$ be the plaintext extractor for $\mathcal{A}$ with respect to $\hat{\mathcal{P}}$, which exists by the assumption above. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_0 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    Append ρ to Rlist
    return ρ
$b'_0 \leftarrow D(x_0)$
return b'_0

Game 1: Let Game 1 be $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\hat{\mathcal{P}},D}(\lambda)$, i.e. we change the plaintext creator to $\hat{\mathcal{P}}$. Written out in full it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_1 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $\boxed{m \leftarrow \hat{\mathcal{P}}(pk, s)}$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    Append ρ to Rlist

return ρ
$b'_1 \leftarrow D(x_1)$
return b'_1

To prove that $|\Pr[S_1] - \Pr[S_0]|$ is negligible, we will consider a sequence of games, Game $(0,0), \ldots,$ Game $(0, q_e)$, where Game $(0,0)$ equals Game 0, Game $(0, q_e)$ equals Game 1, and $|\Pr[S_{0,i}] - \Pr[S_{0,i-1}]|$ is negligible for $i = 1, \ldots, q_e$.

Game $(0, i)$: Let Game $(0, i)$ be the same as $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\hat{\mathcal{P}},D}(\lambda)$ except that the first $q_e - i$ queries to the encryption oracle are answered using $\mathcal{P}$, and all encryption queries after the $(q_e - i)^{th}$ are handled using $\hat{\mathcal{P}}$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_{0,i} \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $\boxed{n_e \leftarrow n_e + 1}$
    if $\boxed{n_e > q_e - i}$ **then**
      $\boxed{m \leftarrow \hat{\mathcal{P}}(pk, s)}$
    else
      $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    Append ρ to Rlist
    return ρ
$b'_{0,i} \leftarrow D(x_{0,i})$
return $b'_{0,i}$

We construct a sequence of adversaries $\mathcal{B}_1, \mathcal{B}_2, \ldots$ against the LH-IND-CPA security of the scheme as follows, where $\mathcal{B}_i = (\mathcal{B}_{i,1}, \mathcal{B}_{i,2})$ for all $i \in \mathbb{N}$:

function $\mathcal{B}_{i,1}(pk)$

$R_{\mathcal{A}} \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R_{\mathcal{A}^*} \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}^*}(\lambda)}$
Run $x_{0,i} \leftarrow \mathcal{A}(pk; R_{\mathcal{A}})$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    Append ρ to Rlist
    Return ρ
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R_{\mathcal{A}}, \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*}; R_{\mathcal{A}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $n_e \leftarrow n_e + 1$
    if $n_e \leq q_e - i$ **then**
      $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
      $C \leftarrow \mathsf{Encrypt}(pk, m)$
      Append C to Clist
      return C_{n_e}
    else if $n_e = q_e - i + 1$ **then**
      $(m'_0, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
      $m'_1 \leftarrow \hat{\mathcal{P}}(pk, s)$
      $\mathsf{state}_{\mathcal{A}} \leftarrow \mathbf{Suspend}(\mathcal{A})$
      $\mathsf{state}_{\mathcal{B}} \leftarrow (pk, \mathsf{Clist}, \mathsf{Rlist}, R_{\mathcal{A}^*}, \mathsf{state}_{\mathcal{A}}, n_e, \mathsf{state}_{\mathcal{A}^*})$
      Terminate and output $(m'_0, m'_1, \mathsf{state}_{\mathcal{B}})$

function $\mathcal{B}_{i,2}(C^*, \mathsf{state}_{\mathcal{B}})$
  Parse $\mathsf{state}_{\mathcal{B}}$ as $(pk, \mathsf{Clist}, \mathsf{Rlist}, R_{\mathcal{A}^*}, , \mathsf{state}_{\mathcal{A}}, n_e, \mathsf{state}_{\mathcal{A}^*})$
  Run $x \leftarrow \mathbf{Resume}(\mathcal{A}, \mathsf{state}_{\mathcal{A}})$
    Return C^* in response to $\mathcal{A}$'s previous encryption query
    if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
      $\rho \xleftarrow{R} \{0,1\}$
      Append ρ to Rlist
      Return ρ
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      $(m, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(pk, C, R_{\mathcal{A}}, \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*}; R_{\mathcal{A}^*})$
      return m
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
      $n_e \leftarrow n_e + 1$
      $m \leftarrow \hat{\mathcal{P}}(pk, s)$
      $C_{n_e} \leftarrow \mathsf{Encrypt}(pk, m)$
      Append C to Clist
      return C_{n_e}
  return $D(x)$

$\mathcal{B}_{i,1}$ takes as input the public key pk and runs $\mathcal{A}$ with oracle access to $\mathcal{A}^*$ exactly as described in the Game 0. $\mathcal{B}_{i,1}$ responds to the first $q_e - i - 1$ encryption oracle queries as in Game 0 (i.e. by computing $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$ and returning $C \leftarrow \mathsf{Encrypt}(pk, m)$). For the $(q_e - i)^{th}$ query to the encryption oracle, $\mathcal{B}_{i,1}$ suspends $\mathcal{A}$, generates both $(m_0, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$ and $m_1 \leftarrow \hat{\mathcal{P}}(pk, s)$ and returns $(m_0, m_1, \mathsf{state})$ to the challenger.

The challenger chooses $b \xleftarrow{R} \{0,1\}$, computes $C^* \leftarrow \mathsf{Encrypt}(pk, m_b)$ and runs $\mathcal{B}_{i,2}(C^*, \mathsf{state})$. $\mathcal{B}_{i,2}$ resumes $\mathcal{A}$ from where it left off when $\mathcal{B}_{i,1}$ terminated, i.e, by responding to $\mathcal{A}$'s encryption query using C^*. It then responds to all subsequent encryption queries using $\hat{\mathcal{P}}$. Eventually $\mathcal{A}$ terminates and outputs a bitstring x. $\mathcal{B}_{i,2}$ terminates by outputting the bit $D(x)$.

Since Π is LH-IND-CPA, $\mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{B}_i}(\lambda)$ is negligible in λ for all $1 \leq i \leq q_e$. Since Game $(0, i) = \mathbf{Expt}^{\mathrm{IND-CPA-0}}_{\Pi,\mathcal{B}_i}(\lambda)$, and Game $(0, i+1) = \mathbf{Expt}^{\mathrm{IND-CPA-1}}_{\Pi,\mathcal{B}_i}(\lambda)$, this implies that

$$\Pr[S_{0,i+1}] - \Pr[S_{0,i}]| \leq \mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{B}_i} .$$

Therefore:

$$\begin{aligned} |\Pr[S_1] - \Pr[S_0]| &= |\Pr[S_{0,0}] - \Pr[S_{0,q_e}]| \\ &\leq |\Pr[S_{0,0}] - \Pr[S_{0,1}]| + \ldots + |\Pr[S_{0,q_e-1}] - \Pr[S_{0,q_e}]| \\ &\leq \sum_{i=1}^{q_e} \mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{B}_i}(\lambda) . \end{aligned}$$

Game 2: Let Game 2 be $\mathbf{Expt}^{\mathrm{PA2+Real}}_{\Pi,\mathcal{A},\hat{\mathcal{P}},D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_2 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**

$\boxed{m \leftarrow \mathsf{Decrypt}(sk, C)}$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$m \leftarrow \hat{\mathcal{P}}(pk, s)$
$C \leftarrow \mathsf{Encrypt}(m)$
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\rho \xleftarrow{R} \{0,1\}$
return ρ
$b'_2 \leftarrow D(x_2)$
return b'_2

By the assumption that Π is 2PA2+ plaintext aware, we have that:

$$|\Pr[S_2] - \Pr[S_1]| \leq \mathbf{Adv}^{\mathrm{2PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}(\lambda)\,.$$

Game 3: Let Game 3 be $\mathbf{Expt}^{\mathrm{PA2+Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda)$. In other words, we modify the encryption oracle to use $\mathcal{P}$ instead of $\hat{\mathcal{P}}$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_3 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$m \leftarrow \mathsf{Decrypt}(sk, C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$\boxed{(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})}$
$C \leftarrow \mathsf{Encrypt}(m)$
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\rho \xleftarrow{R} \{0,1\}$
return ρ
$b'_3 \leftarrow D(x_3)$
return b'_3

To prove that $|\Pr[S_3] - \Pr[S_2]|$ is negligible, we will consider a sequence of games, Game $(2,0), \ldots,$ Game $(2, q_e)$, where Game $(2,0)$ equals Game 2, Game $(2, q_e)$ equals Game 3, and $|\Pr[S_{2,i}] - \Pr[S_{2,i-1}]$ is negligible for $i = 1, \ldots, q_e$.

Game $(2, i)$: Let Game $(2, i)$ be the same as $\mathbf{Expt}^{\mathrm{PA2+Real}}_{\Pi,\mathcal{A},\mathcal{P},D}(\lambda)$ except that

all encryption queries after the i^{th} are handled as in $\mathbf{Expt}^{\text{PA2+Real}}_{\Pi,\mathcal{A},\hat{\mathcal{P}},D}(\lambda)$, i.e. the encryption oracle computes $m \leftarrow \hat{\mathcal{P}}(pk, s)$ and returns $C \leftarrow \mathsf{Encrypt}(pk, m)$. Only the first i queries to the encryption oracle are answered using $\mathcal{P}$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
Run $x_{2,i} \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $m \leftarrow \mathsf{Decrypt}(sk, C)$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $n_e \leftarrow n_e + 1$
    if $n_e \leq i$ **then**
      $(m, \mathsf{state}_\mathcal{P}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_\mathcal{P})$
    else
      $m \leftarrow \hat{\mathcal{P}}(pk, s)$
    $C \leftarrow \mathsf{Encrypt}(m)$
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    return ρ
$b'_{2,i} \leftarrow D(x_{2,i})$
return $b'_{2,i}$

We construct a sequence of adversaries $\mathcal{C}_1, \mathcal{C}_2, \ldots$ against the LH-IND-CCA2 security of the scheme as follows, where $\mathcal{C}_i = (\mathcal{C}_{i,1}, \mathcal{C}_{i,2})$ for all $1 \leq i \leq q_e$:

function $\mathcal{C}_{i,1}(pk)$
  $R_\mathcal{A} \xleftarrow{R} \{0,1\}^{T_\mathcal{A}(\lambda)}$
  Run $\mathcal{A}(pk; R_\mathcal{A})$
    if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
      $\rho \xleftarrow{R} \{0,1\}$
      return ρ
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
      $n_e \leftarrow n_e + 1$
      if $n_e < i$ **then**
        $(m, \mathsf{state}_\mathcal{P}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_\mathcal{P})$

$C \leftarrow \mathsf{Encrypt}(pk, m)$
return C
else if $n_e = i$ **then**
$(m_0, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$
$m_1 \leftarrow \hat{\mathcal{P}}(pk, s)$
$\mathsf{state}_{\mathcal{A}} \leftarrow \mathbf{Suspend}(\mathcal{A})$
$\mathsf{state}_{\mathcal{C}} \leftarrow (\mathsf{state}_{\mathcal{A}}, n_e)$
Terminate and output $(m_0, m_1, \mathsf{state}_{\mathcal{C}})$

function $\mathcal{C}_{i,2}(C^*, \mathsf{state}_{\mathcal{C}})$
Parse $\mathsf{state}_{\mathcal{C}}$ as $(\mathsf{state}_{\mathcal{A}}, n_e)$
Run $x \leftarrow \mathbf{Resume}(\mathcal{A}, \mathsf{state}_{\mathcal{A}})$
Return C^* in response to $\mathcal{A}$'s previous encryption query
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\rho \xleftarrow{R} \{0,1\}$
return ρ
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
Query $m \leftarrow \mathsf{Decrypt}(C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$n_e \leftarrow n_e + 1$
$m \leftarrow \hat{\mathcal{P}}(pk, s)$
$C \leftarrow \mathsf{Encrypt}(pk, m)$ **return** C
return $D(x)$

$\mathcal{C}_{i,1}$ takes as input the public key pk and runs $\mathcal{A}$ with oracle access to $\mathcal{A}^*$ exactly as described in the Game 2. $\mathcal{C}_i$ responds to the first $i-1$ encryption oracle queries as in Game 2 (i.e. by computing $(m, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s, \mathsf{state}_{\mathcal{P}})$ and returning $C \leftarrow \mathsf{Encrypt}(pk, m)$ to $\mathcal{A}$). For the i^{th} query to the encryption oracle, $\mathcal{C}_{i,1}$ generates both $(m_0, \mathsf{state}_{\mathcal{P}}) \leftarrow \mathcal{P}(pk, s)$ and $m_1 \leftarrow \hat{\mathcal{P}}(s)$ and returns $(m_0, m_1, \mathsf{state})$ to the challenger.

The challenger chooses $b \xleftarrow{R} \{0,1\}$, computes $C^* \leftarrow \mathsf{Encrypt}(pk, m_b)$ and runs $\mathcal{C}_{i_2}(C^*, \mathsf{state})$. $\mathcal{C}_{i,2}$ resumes $\mathcal{A}$ so that it is in the same state as when $\mathcal{C}_{i,1}$ terminated. It responds to the i^{th} encryption query using C^*. It then responds to all subsequent encryption queries using $\hat{\mathcal{P}}$. Eventually $\mathcal{A}$ terminates and outputs a bitstring x. $\mathcal{C}_{i,2}$ terminates by outputting the bit $D(x)$.

Since Π is LH-IND-CCA2, $\mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{C}_i}(\lambda)$ is negligible in λ. Since Game $(2, i-1) = \mathbf{Expt}^{\mathrm{IND-CCA2-0}}_{\Pi,\mathcal{C}}(\lambda)$, and Game $(2, i) = \mathbf{Expt}^{\mathrm{IND-CCA2-1}}_{\Pi,\mathcal{C}_i}(\lambda)$, this implies that $|\Pr[S_{2,i}] - \Pr[S_{2,i-1}]| \leq \mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{C}_i}(\lambda)$, for all $i \in \{1, \ldots, q_e\}$. Therefore

$$\begin{aligned} |\Pr[S_3] - \Pr[S_2]| &= |\Pr[S_{2,0}] - \Pr[S_{2,q_e}]| \\ &\leq |\Pr[S_{2,0}] - \Pr[S_{2,1}]| + \ldots + |\Pr[S_{2,q_e-1}] - \Pr[S_{2,q_e}]| \\ &\leq \sum_{i=1}^{q_e} \mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{C}_i}(\lambda)\,. \end{aligned}$$

Putting it all together, we see that:

$$\begin{aligned} \mathbf{Adv}^{\mathrm{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda) &= |\Pr[S_0] - \Pr[S_3]| \\ &\leq |\Pr[S_0] - \Pr[S_1]| + |\Pr[S_2] - \Pr[S_2]| + |\Pr[S_2] - \Pr[S_3] \\ &= \sum_{i=1}^{q_e} \mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{B}_i}(\lambda) + \mathbf{Adv}^{\mathrm{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\hat{\mathcal{P}},D}(\lambda) + \\ &\qquad \sum_{i=1}^{q_e} \mathbf{Adv}^{\mathrm{LH-IND-CCA2}}_{\Pi,\mathcal{C}_i}(\lambda)\,. \end{aligned}$$

which is negligible in λ as required. □

Theorem 2.5.4. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be an encryption scheme which is LH-IND-CCA2 secure and suppose that there is some polynomial-time stateless plaintext creator* $\hat{\mathcal{P}}$ *such that for all polynomial-time ciphertext creators* $\mathcal{A}$ *there exists a ciphertext extractor* $\mathcal{A}^*$ *such that for all distinguishing algorithms* D, $\mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{A},\mathcal{A}^*,\hat{\mathcal{P}},D}(\lambda)$ *negligible in* λ. *Then* Π *is PA2 plaintext aware.*

The proof of this theorem is nearly identical, but without the need to consider the randomness oracle.

Corollary 2.5.5. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme which is LH-IND-CPA secure and 2PA2+ plaintext aware. Then* Π *is PA2+ plaintext aware.*

Proof. By Theorem 2.4.6, Π is LH-IND-CCA2 secure. Since $\mathcal{P}_0$ is a stateless plaintext creator, Theorem 2.5.2 shows that Π is PA2+ secure. □

The fact that we may be substitute an arbitrary plaintext creator $\mathcal{P}$ with the specific plaintext creator $\hat{\mathcal{P}}$ will be crucial in proving the relationship between PA2 and PA2+ in Section 2.7.

2.6 PA2, One-Wayness and Length Hiding

In the previous section, our theorem required the encryption scheme to be length hiding, rather than just IND-CPA. This may seem like an unreasonable restriction. Far from being unreasonable, we will show in this section that length hiding is necessary in order for an IND-CPA scheme to achieve PA2 plaintext awareness. Our proof will not preclude the possibility that a scheme is 2PA2 plaintext aware, IND-CPA secure, but not length hiding.

In particular, the combination of Theorem 2.6.2 and Lemma 2.4.5 shows that there are no schemes with infinite message spaces which are IND-CPA and PA2 plaintext aware.

We base our proof on the following result, proved by Teranishi and Ogata [34].

Theorem 2.6.1 (Teranishi and Ogata)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme and suppose that* Π *is OW-CPA secure and PA2 plaintext aware. Then* Π *is IND-CPA secure, and hence IND-CCA2 secure.*

Teranishi and Ogata assume that Π is a public-key encryption scheme which is not IND-CPA secure, so there is an IND-CPA adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{A}}(\lambda)$ is non-negligible in λ. They construct a ciphertext creator $\mathcal{B}$ and a plaintext creator $\mathcal{P}$ which collude to "smuggle" a ciphertext generated by $\mathcal{P}$ back to $\mathcal{B}$ in a bit by bit fashion, by using $\mathcal{A}$ to exploit the insecurity of Π. Since it was not returned by the encryption oracle, C does not appear in Clist, and $\mathcal{A}$ may legally decrypt it using the decryption oracle.

The observation underlying this entire section is simply that this technique works identically if $\mathcal{A}$ is an LH-IND-CPA adversary instead of an IND-CPA adversary, as the restriction to equal length messages is unused in the proof. We would like to include the case where $\mathbf{MsgSp}(pk)$ is infinite, for example $\{0,1\}^*$, but OW-CPA security is not defined for infinite message spaces. To account for this, we will adapt their proof to show that a scheme which is IND-CPA and PA2 plaintext aware is LH-IND-CPA. In the case where $\mathbf{MsgSp}(pk)$ is finite, we may combine this with Teranishi and Ogata's original result to see that if Π is OW-CPA then it is LH-IND-CPA. The precise details of this construction are presented in Theorem 2.6.2.

Theorem 2.6.2. *Suppose that* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is an encryption scheme which is PA2 plaintext aware and IND-CPA secure, and suppose that* $|\mathbf{MsgSp}(pk)|$ *is super-polynomial in* λ. *Then* Π *is LH-IND-CPA secure.*

Proof. Suppose that Π is PA2 plaintext aware, but not LH-IND-CPA secure. Then there exists an LH-IND-CPA adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathbf{Adv}^{\mathrm{LH-IND-CPA}}_{\Pi,\mathcal{A}}(\lambda)$ is non-negligible in λ. Teranishi and Ogata construct a probabilistic polynomial-time algorithm $\mathsf{GuessNu}$ which takes a public key pk, a bit b and an integer N as input. $\mathsf{GuessNu}$ runs $\mathbf{Expt}^{\mathrm{IND-CPA-b}}_{\Pi,\mathcal{A}}$ N times and returns an estimate for $\Pr[\mathbf{Expt}^{\mathrm{IND-CPA-b}}_{\Pi,\mathcal{A}}(\lambda) = 1]$ by simply computing

the fraction ℓ/N where ℓ is the number of times $\mathcal{A}$ correctly guessed b[1]. The following algorithms make use of $\mathsf{GuessNu}$ to transmit a single bit:

function $\mathsf{Setup}(pk, N)$
  $v_0 \leftarrow \mathsf{GuessNu}(pk, 0, N)$
  $v_1 \leftarrow \mathsf{GuessNu}(pk, 1, N)$
  for $i = 1$ to N **do**
    $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathcal{A}_1(pk)$
    Append $(m_0, m_1, \mathsf{state}_{\mathcal{A}})$ to $\mathsf{FindList}$
  $\mathsf{Param} \leftarrow (v_0, v_1, \mathsf{FindList})$
  return Param

function $\mathsf{Send}(pk, \mathsf{Param}, b, j)$
  Parse Param as $(v_0, v_1, \mathsf{FindList})$
  $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathsf{FindList}[j]$
  return m_b

function $\mathsf{Receive}(pk, \mathsf{Param}, N, C_1, \ldots, C_N)$
  Parse Param as $(v_0, v_1, \mathsf{FindList})$
  for $j = 1$ to N **do**
    $(m_0, m_1, \mathsf{state}_{\mathcal{A}}) \leftarrow \mathsf{FindList}[j]$
    $b_j \leftarrow \mathcal{A}_2(C_j, \mathsf{state}_{\mathcal{A}})$
    $\ell \leftarrow \ell + b_j$
  if $\ell/N \geq (v_0 + v_1)/2$ **then**
    return 1
  else
    return 0

The algorithms $(\mathsf{Setup}, \mathsf{Send}, \mathsf{Receive})$ may be used to send a single bit b as follows:

function $\mathsf{Channel}(b)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  $N \leftarrow \lceil 32n\lambda^\alpha / p(\lambda)^2 \rceil$
  $\mathsf{Param} \leftarrow \mathsf{Setup}(pk, N)$
  for $i = 1$ to N **do**
    $m \leftarrow \mathsf{Send}(pk, \mathsf{Param}, b, i)$
    $C_i \leftarrow \mathsf{Encrypt}(pk, m)$

[1] By extension, for LH-IND-CPA adversaries $\mathsf{GuessNu}$ estimates $\Pr[\mathbf{Expt}^{\mathrm{LH-IND-CPA-b}}_{\Pi,\mathcal{A}}(\lambda) = 1]$.

$b' \leftarrow \mathsf{Receive}(pk, \mathsf{Param}, C_1, \ldots, C_N)$
return b'

In the above, α is an arbitrary constant which represents a trade-off between running time and success probability. For our purposes any value $\alpha > 0$ suffices.

Lemma 2.6.3. *There is an infinite set $\Lambda \subset \mathbb{N}$ of security parameters, a family of sets $\{\Omega_\lambda\}_{\lambda \in \Lambda}$ and a polynomial p such that for all $\lambda \in \Lambda$, the following two properties hold:*

$$\Pr[pk \in \Omega_\lambda : (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)] \geq \frac{1}{p(\lambda)}$$

and for all $\lambda > 0$, for all $n > 0$, for all $\alpha > 0$, for all $pk \in \Omega_\lambda$ and for $b \in \{0, 1\}$,

$$\Pr[\mathsf{Channel}(b) = b] \geq 1 - \frac{1}{n\lambda^\alpha}$$

The proof of this is given in Appendix B.1 of Teranishi and Ogata's paper [34]. Since neither GuessNu, Setup, Send nor Receive ever make use of the fact that $|m_0| = |m_1|$ in $\mathbf{Expt}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{A}}$, the proof also covers the case of LH-IND-CPA adversaries with no alteration.

We present a ciphertext creator and a plaintext creator which exploit this channel to break the IND-CPA security of Π. The interaction between this ciphertext creator and plaintext creator is quite complex, so we will first describe it in general terms, then present the algorithms below. Recall that when $\mathcal{B}$ makes an encryption query the challenger runs $\mathcal{P}$ on this input, $\mathcal{P}$ returns a message, which the challenger encrypts and returns to $\mathcal{B}$. The objective

here is for $\mathcal{P}$ to produce a ciphertext C to which $\mathcal{B}$ does not "know" the underlying message, and return it to $\mathcal{B}$. $\mathcal{B}$ may then use its decryption oracle to recover the message, since it was not one of the ciphertexts created by the challenger. The complication arises because while $\mathcal{B}$ may pass any information it likes to $\mathcal{P}$, $\mathcal{P}$'s responses will all be encrypted by the challenger. Instead, $\mathcal{B}$ runs Setup to generate a set of parameters for each bit of the ciphertext to be transmitted, which it passes to $\mathcal{P}$ via an $\mathsf{Encrypt}$ query. $\mathcal{P}$ uses these along along with the Send algorithm to send the bits of C^* to $\mathcal{B}$, which may recover the bits of the ciphertext individually using $\mathsf{Receive}$. $\mathcal{B}$ may then decrypt the ciphertext C^* using its decrypt oracle (since C^* was not returned by the encryption oracle and therefore does not appear in Clist). Finally, $\mathcal{B}$ checks that the message returned by the decrypt oracle is correct by sending it back to the plaintext creator in another encrypt query, which uses Send to tell $\mathcal{B}$ whether the message is correct.

Let M be a probabilistic polynomial-time algorithm which takes a public key pk as input and returns an arbitrary pair of messages $(m_0, m_1) \in \mathbf{MsgSp}(pk)^2$ such that $|m_0| = |m_1|$ but $m_0 \neq m_1$. If C is any IND-CPA adversary, then $\mathcal{C}_1$ satisfies these requirements, so the assumption that M exists is not unusual. Let $\ell(pk)$ be an upper bound on the length of a ciphertext produced by encrypting a message of length $|m_0|$ under the key pk, let $\alpha > 0$, and let p be the polynomial as in Lemma 2.6.3. We present a ciphertext creator $\mathcal{B}$ and plaintext creator $\mathcal{P}$ below. The symbols $\texttt{SendParams}, \texttt{AskBit}, \texttt{SendM}$ and $\texttt{AskT}$ represent arbitrary constants: all that we require is that they are distinct.

function $\mathcal{B}(pk)$
  $n \leftarrow \ell(pk)$
  $N \leftarrow \lceil 32(n+1)\lambda^\alpha / p(\lambda)^2 \rceil$

for $i = 1$ to n **do**
  $\mathsf{Param}_i \leftarrow \mathsf{Setup}(pk, N)$
Query $\mathsf{Encrypt}(\texttt{SendParams}, (pk, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n))$
for $i = 1$ to n **do**
  for $j = 1$ to N **do**
    Query $C_j \leftarrow \mathsf{Encrypt}(\texttt{AskBit}, (i, j))$
  $b'_i \leftarrow \mathsf{Receive}(pk, \mathsf{Param}_i, N, C_1, \ldots, C_N)$
$C' \leftarrow b'_1 || \ldots || b'_n$
Query $m' \leftarrow \mathsf{Decrypt}(C')$
$\mathsf{Param}' \leftarrow \mathsf{Setup}(pk, N)$
Query $\mathsf{Encrypt}(\texttt{SendM}, (C', m', \mathsf{Param}'))$
for $j = 1$ to N **do**
  Query $C_j \leftarrow \mathsf{Encrypt}(\texttt{AskT}, j)$
$T' \leftarrow \mathsf{Receive}(pk, \mathsf{Param}', N, C_1, \ldots, C_N)$
if $T' = 0$ **then**
  $m' \leftarrow \perp$
return (pk, C', m')

function $\mathcal{P}((Q_1, Q_2), \mathsf{state}_\mathcal{P})$
  if $Q_1 = \texttt{SendParams}$ **then**
    Parse Q_2 as $(pk, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
    $(m_0, m_1) \leftarrow M(pk)$
    $b \xleftarrow{\text{R}} \{0, 1\}$
    $C^* \leftarrow \mathsf{Encrypt}(pk, m_b)$
    $\mathsf{state}_\mathcal{P} \leftarrow (pk, m_b, C^*, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
    return $(0, \mathsf{state}_\mathcal{P})$
  else if $Q_1 = \texttt{AskBit}$ **then**
    Parse Q_2 as (i, j)
    Parse $\mathsf{state}_\mathcal{P}$ as $(pk, m_b, C^*, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
    $b_i \leftarrow i^{th}$ bit of C^*
    $m \leftarrow \mathsf{Send}(pk, \mathsf{Param}_i, b_i, j)$
    return $(m, \mathsf{state}_\mathcal{P})$
  else if $Q_1 = \texttt{SendM}$ **then**
    Parse Q_2 as $(C', m', \mathsf{Param}'))$
    Parse $\mathsf{state}_\mathcal{P}$ as $(pk, m_b, C^*, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
    if $m_b = m'$ **then**
      $T \leftarrow 1$
    else
      $T \leftarrow 0$
    $\mathsf{state}_\mathcal{P} \leftarrow (pk, \mathsf{Param}', N, T)$
    return $(0, \mathsf{state}_\mathcal{P})$

else if $Q_1 = \texttt{AskT}$ **then**
$j \leftarrow Q_2$
Parse $\mathsf{state}_{\mathcal{P}} \leftarrow (pk, \mathsf{Param}', N, T)$
$m \leftarrow \mathsf{Send}(pk, \mathsf{Param}', T, j)$
return $(m, \mathsf{state}_{\mathcal{P}})$

By the PA2 property of Π, there is an extractor $\mathcal{B}^*$ such that for all distinguishing algorithms D,

$$\mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda) = |\Pr[\mathbf{Expt}^{\mathrm{PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P},D}(\lambda) - \Pr[\mathbf{Expt}^{\mathrm{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda)]|\,,$$

is negligible in λ. Let us now analyse the behaviour of $\mathcal{B}$ in the $\mathbf{Expt}^{\mathrm{PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P},D}(\lambda)$ experiment. Lemma 2.6.3 shows that for all $\lambda \in \Lambda$,

$$\Pr[pk \in \Omega_\lambda] \geq \frac{1}{p(\lambda)}\,,$$

which is non-negligible since p is a polynomial. Suppose that $pk \in \Omega_\lambda$. Then Lemma 2.6.3 also shows that each bit of C' is received correctly by $\mathcal{B}$ with probability at least $1 - \frac{1}{(n+1)\lambda^\alpha}$. Since each bit is recovered independently, it follows that $C' = C$ with probability at least

$$\left(1 - \frac{1}{(n+1)\lambda^\alpha}\right)^n > 1 - \frac{n}{(n+1)\lambda^\alpha}\,.$$

Thus

$$\begin{aligned}\Pr[C' = C] &\geq \Pr[C' = C | pk \in \Omega_\lambda]\Pr[pk \in \Omega_\lambda] \\ &\geq \frac{1}{p(\lambda)}\left(1 - \frac{n}{(n+1)\lambda^\alpha}\right)\,.\end{aligned}$$

which is non-negligible. Since decryption queries are answered using the real decryption algorithm, this implies that the event $m' = m_b$ occurs with the

same probability. If $m' = m_b$ then $T = 1$, and $T' = 1$ with probability at least $1 - \frac{1}{(n+1)\lambda^\alpha}$.

Putting it all together we see that

$$\begin{aligned}\Pr[\mathcal{B} \text{ outputs } m] &\geq \frac{1}{p(\lambda)}\left(1 - \frac{n}{(n+1)\lambda^\alpha}\right)\left(1 - \frac{1}{(n+1)\lambda^\alpha}\right) \\ &> \frac{1}{p(\lambda)}\left(1 - \frac{n}{(n+1)\lambda^\alpha} - \frac{1}{(n+1)\lambda^\alpha}\right) \\ &= \frac{1}{p(\lambda)}\left(1 - \frac{1}{\lambda^\alpha}\right)\end{aligned}$$

On the other hand, if $T' = 0$, $\mathcal{B}$ returns $\perp$ by definition. Now consider the $\mathbf{Expt}^{\text{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda)$ experiment with the distinguishing algorithm D which simply returns 1 if $\mathcal{B}$ outputs $\perp$ and 0 otherwise. This shows that that in the $\mathbf{Expt}^{\text{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda)$ experiment, the probability that $\mathcal{B}$ returns $m' \neq \perp$, or equivalently,

$$\Pr[m' = m_b] \geq \frac{1}{p(\lambda)}\left(1 - \frac{1}{\lambda^\alpha}\right) - \mathbf{Adv}^{\text{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda)\,.$$

We construct an IND-CPA adversary $\mathcal{C}$ based on $\mathcal{B}$, $\mathcal{B}^*$ and $\mathcal{P}$ as follows:

function $\mathcal{C}_1(pk)$
  $R[\mathcal{B}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{B}}(\lambda)}$
  Run $(pk, C', m') \leftarrow \mathcal{B}(pk; R[\mathcal{B}])$
    if $\mathcal{B}$ queries $\mathsf{Encrypt}(Q_1, Q_2)$ **then**
      if $Q_1 =$ `SendParams` **then**
        $\mathsf{state}_{\mathcal{B}} \leftarrow$ **Suspend**$(\mathcal{B})$
        $(m_0, m_1) \leftarrow M(pk)$
        Parse Q_2 as $(pk, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
      else
        return $\perp$ ▷ This will never occur, since the first query
        ▷ made by $\mathcal{B}$ is always `SendParams`.
  $\mathsf{state}_{\mathcal{C}} \leftarrow (\mathsf{state}_{\mathcal{B}}, R[\mathcal{B}], pk, m_0, m_1, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$

return $(m_0, m_1, \mathsf{state}_{\mathcal{C}})$

function $\mathcal{C}_2(C^*, \mathsf{state}_{\mathcal{C}})$
Parse $\mathsf{state}_{\mathcal{C}}$ as $(\mathsf{state}_{\mathcal{B}}, R[\mathcal{B}], pk, m_0, m_1, N, \mathsf{Param}_1, \ldots, \mathsf{Param}_n)$
$C \leftarrow \mathsf{Encrypt}(pk, 0)$
Append C to Clist
Run $(pk, C', m') \leftarrow \mathbf{Resume}(\mathcal{B}, \mathsf{state}_{\mathcal{B}})$
return C to $\mathcal{B}$'s previous query.
if $\mathcal{B}$ queries $\mathsf{Encrypt}(Q_1, Q_2)$ **then**
if $Q_1 = \texttt{AskBit}$ **then**
Parse Q_2 as (i, j)
$b_i \leftarrow i^{th}$ bit of C^*
$m \leftarrow \mathsf{Send}(pk, \mathsf{Param}_i, b_i, j)$
else
Terminate $\mathcal{B}$
$C \leftarrow \mathsf{Encrypt}(pk, m)$
Append C to Clist
return C
if $\mathcal{B}$ queries $\mathsf{Decrypt}(C')$ **then**
$m' \leftarrow \mathcal{B}^*(pk, C', R[\mathcal{B}], \mathsf{Clist}, \varepsilon)$
Terminate $\mathcal{B}$
if $m' = m_1$ **then**
return 1
else
return 0

By definition of $\mathcal{C}$, $\mathbf{Expt}^{\mathrm{IND-CPA-b}}_{\Pi,\mathcal{C}}(\lambda)$ is identical to $\mathbf{Expt}^{\mathrm{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}',D}(\lambda)$ until the point where $\mathcal{B}$ makes the $\mathsf{Decrypt}$ query. The queries that occur after this point in the PA2-Fake experiment were needed to show that $\mathcal{B}^*$ returns the correct message, but once we have obtained the value m' it is not necessary for the adversary to continue the simulation. In particular, it follows that:

$$\Pr[m' = m_b] \geq \frac{1}{p(\lambda)}\left(1 - \frac{1}{\lambda^{\alpha}}\right) - \mathbf{Adv}^{\mathrm{PA2}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}(\lambda)\,.$$

We must now bound $\Pr[\mathbf{Expt}^{\mathrm{IND-CPA-0}}_{\Pi,\mathcal{C}}(\lambda) = 1]$. By definition of PA2 plaintext awareness, $\mathcal{B}^*$ is independent of the choice of $\mathcal{P}$ and does not receive

the random coins of $\mathcal{P}$. Since the challenger does not ever use the value m_1 in $\mathbf{Expt}_{\Pi,\mathcal{C}}^{\mathrm{IND-CPA-0}}(\lambda)$, it follows that m_1 is independent of the inputs to $\mathcal{B}^*$, and thus

$$\Pr[\mathbf{Expt}_{\Pi,\mathcal{C}}^{\mathrm{IND-CPA-0}}(\lambda) = 1] \leq \frac{1}{|\mathbf{MsgSp}(pk)|}\,.$$

Putting it all together we see that:

$$\begin{aligned}
\mathbf{Adv}_{\Pi,\mathcal{C}}^{\mathrm{IND-CPA}}(\lambda) &= |\Pr[\mathbf{Expt}_{\Pi,\mathcal{C}}^{\mathrm{IND-CPA-1}}(\lambda) = 1] - \Pr[\mathbf{Expt}_{\Pi,\mathcal{C}}^{\mathrm{IND-CPA-0}}(\lambda) = 1]| \\
&\geq \frac{1}{p(\lambda)}\left(1 - \frac{1}{\lambda^{\alpha}}\right) - \mathbf{Adv}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P},D}^{\mathrm{PA2}}(\lambda) - \frac{1}{|\mathbf{MsgSp}(pk)|}
\end{aligned}$$

which is negligible as required, since $|\mathbf{MsgSp}(pk)|$ is super-polynomial.

□

2.7 Connection Between PA2 And PA2+

Clearly, a scheme which is PA2+ must necessarily be PA2, since an adversary may simply not use its randomness oracle, but the converse is not obviously true. We now show that it is true for any IND-CPA secure encryption scheme, since an adversary may use randomness inherent in a ciphertext generated by the encryption oracle to simulate a randomness oracle. This in turn implies that a suitably random PA2 encryption scheme is PA1+, thus giving a formal proof to the conjecture of Dent [14].

The proof essentially involves constructing a randomness oracle by taking ciphertexts created by the encryption algorithm and hashing them onto a single bit using a randomly-chosen universal hash function. The resulting

distribution on $\{0,1\}$ is only a small statistical distance from the uniform distribution on $\{0,1\}$ and the result follows from the Leftover Hash Lemma [20]. One subtlety of the proof is that we will require the ciphertext creator $\mathcal{A}^*$ that we construct to know the functionality of the plaintext creator $\mathcal{P}$. Hence, we actually prove that an IND-CPA and 2PA2 plaintext aware encryption scheme is 2PA2+, and appeal to Theorem 2.5.2 and Theorem 2.6.2 to finish the proof.

Definition 2.7.1 (Universal Hash Family)**.** *A family* $\mathbf{H} = (H, K, A, B)$ *of functions* $(H_k)_{k\in K}$ *where each* H_k *maps* A *to* B *is universal if for all* $x \neq y$ *in* A, $\Pr[H_k(x) = H_k(y) : k \xleftarrow{R} K] \leq 1/|B|$.

This definition was originally given by Carter and Wegman [35]. We will use a universal function family $\mathbf{H} = (H_k)_{k\in K}$ where H_k is a function from $\{0,1\}^* \to \{0,1\}$ for all $k \in K$. For simplicity, we will assume $K = \{0,1\}^n$. Such families are known to exist without any computational assumptions [35].

Definition 2.7.2 (Collision Probability)**.** *For any random variable* x *which takes values on a set* X*, we define the collision probability*

$$\kappa(x) := \sum_{y\in X} \Pr[x = y]^2 .$$

Remark 2.7.3. $\kappa(x)$ is called the collision probability because if x and x' are

random variables independent and identically distributed on X,

$$\begin{aligned}
\Pr[x = x'] &= \sum_{y \in X} \Pr[x = x' = y] \\
&= \sum_{y \in X} \Pr[x = y \wedge x' = y] \\
&= \sum_{y \in X} \Pr[x = y] \Pr[x' = y] \\
&= \sum_{y \in X} \Pr[x = y]^2 \\
&= \kappa(x)
\end{aligned}$$

Lemma 2.7.4. *Let x be a random variable on a set X. Then*

$$\max_{z \in X} \Pr[x = z] \leq \sqrt{\kappa(x)}\,.$$

Proof. Let $z \in X$ be an arbitrary element of X. Then

$$\Pr[x = z]^2 \leq \sum_{y \in X} \Pr[x = y]^2 = \kappa(x)$$

So $\Pr[x = z] \leq \sqrt{\kappa(x)}$. □

The Leftover Hash Lemma was originally proved by Håstard *et al.* [19]. We present the version of Leftover Hash Lemma given in Theorem 6.21 of [33].

Theorem 2.7.5 (Leftover Hash Lemma)**.** *Let $\mathbf{H} = (H, K, A, B)$ be a family of universal hash functions where $|B| = \beta$. Let $k \xleftarrow{R} K$, let $x_1 \ldots x_n$ denote any mutually independent random variables on the set A which are independent of the choice of k, let $(y_1, \ldots, y_n) \xleftarrow{R} B^n$, and let*

$$\kappa = \max_{i=1}^{\ell} \{\kappa(x_i)\}\,.$$

Then $\Delta[(k, H_k(x_1), \ldots, H_k(x_\ell)), (k, y_1, \ldots, y_\ell)] \leq \ell\sqrt{\beta\kappa}/2$.

Lemma 2.7.6. *Let* Π *be an IND-CPA encryption scheme, let* M *be a deterministic polynomial-time algorithm which takes a public key* pk *as input and returns a message* $m \in \mathbf{MsgSp}(pk)$*, and let* X *be a random variable distributed according to* $\mathsf{Encrypt}(pk, M(pk))$*. Then there is an IND-CPA adversary* $\mathcal{C}$ *such that the collision probability*

$$\kappa(X) = \mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)\,.$$

Proof. Let $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2)$ be the following polynomial-time IND-CPA adversary:

function $\mathcal{C}_1(pk)$
  $m_0 \leftarrow M(pk)$
  Let m_1 be an arbitrary element of $\mathbf{MsgSp}(pk) \setminus \{m_0\}$.
  $\mathsf{state} \leftarrow pk$
  return $(m_0, m_1, \mathsf{state})$

function $\mathcal{C}_2(C^*, \mathsf{state})$
  $pk \leftarrow \mathsf{state}$
  $C \leftarrow \mathsf{Encrypt}(pk, M(pk))$
  if $C = C^*$ **then**
    return 0
  else
    return 1

Then

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{A}}(\lambda) &= |\Pr[\mathbf{Expt}^{\mathrm{IND-CPA-0}}_{\Pi,\mathcal{A}}(\lambda) = 0] \\
&\qquad - \Pr[\mathbf{Expt}^{\mathrm{IND-CPA-1}}_{\Pi,\mathcal{A}}(\lambda) = 0]| \\
&= |\Pr[C = C^*] - 0| \\
&= \Pr[C = C^*]
\end{aligned}$$

because in $\mathbf{Expt}^{\mathrm{IND-CPA-1}}_{\Pi,\mathcal{A}}(\lambda)$, $\mathsf{Decrypt}(sk, C^*) = m_1 \neq m_0 = \mathsf{Decrypt}(sk, C)$, so $C^* \neq C$. In $\mathbf{Expt}^{\mathrm{IND-CPA-0}}_{\Pi,\mathcal{A}}(\lambda)$, C^* and C are independent and identically distributed, as they are both generated by $\mathsf{Encrypt}(pk, M(pk))$. Thus $\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[C = C^*] = \kappa(C)$ by Remark 2.7.3.

□

Theorem 2.7.7. *Suppose a public key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is IND-CPA secure and 2PA2 plaintext aware. Then it is 2PA2+ plaintext aware.*

Proof. Fix a security parameter λ, and a key pair $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$. Let $\ell(\lambda)$ be an upper bound on $|\mathsf{Encrypt}(pk, 0)|$. Carter and Wegman give constructions of universal hash functions mapping $\{0, 1, \ldots, \alpha - 1\}^n \to \{0,1\}^m$ using keys in the set $\{0,1\}^{\alpha n m}$ [35] without computational assumptions. However this only works for fixed length strings. In this case, we would like to map an element $x \in \{0,1\}^{\leq \ell(\lambda)}$ onto $\{0,1\}$. To achieve this, we may first encode x as a bitstring of length $\ell(\lambda)$ by encoding x as the $\ell(\lambda)+1$ bit string $0^{\ell(\lambda)-|x|}||1||x$, and then hashing using the construction described above, using keys of length $2\ell(\lambda)+2$ bits.

Let $\mathbf{H} = (H, \{0,1\}^{2\ell(\lambda)+2}, \{0,1\}^{\leq \ell(\lambda)}, \{0,1\})$ be a family of universal hash functions mapping $\{0,1\}^{\leq \ell(\lambda)}$ onto $\{0,1\}$.

Let $\mathcal{A}$ be a 2PA2+ ciphertext creator for Π. We construct a 2PA2 ciphertext creator $\mathcal{B}$ as follows:

function $\mathcal{B}(pk)$
  $k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
  $R[\mathcal{A}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
  $x \leftarrow \mathcal{A}(pk; R[\mathcal{A}])$
  if $\mathcal{A}$ Queries $\mathsf{Encrypt}(m_0, m_1)$ **then**
    $C \leftarrow$ Query $\mathsf{Encrypt}(m_0, m_1)$
    return C

if $\mathcal{A}$ Queries $\mathsf{Decrypt}(C)$ **then**
  $m \leftarrow$ Query $\mathsf{Decrypt}(C)$
  return m
if $\mathcal{A}$ Queries $\mathsf{Randomness}$ **then**
  $C \leftarrow$ Query $\mathsf{Encrypt}(0,0)$
  $\rho \leftarrow H_k(C)$
  return ρ
return x

Since $\mathcal{B}$ is a valid 2PA2 ciphertext creator, there exists a plaintext extractor $\mathcal{B}^*$. We use $\mathcal{B}^*$ to construct a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$.

Recall that $\mathcal{A}^*$ takes input $(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$. $\mathcal{A}^*$ works by choosing a hash key k then using the random coins $R[\mathcal{A}]$ to simulate the execution of $\mathcal{A}$. This allows it to construct a fake ciphertext list $\mathsf{Clist}_{\mathcal{B}}$ for $\mathcal{B}^*$ which interleaves ciphertexts that come from encryption queries, and ciphertexts that $\mathcal{B}$ would use to obtain random bits. Specifically it works as follows:

function $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{Clist}, \mathsf{state}_{\mathcal{A}^*})$
  if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
    $k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    $\mathsf{state}_{\mathcal{A}} \leftarrow (pk; R[\mathcal{A}])$
  else
    Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(k, \mathsf{Clist}_{\mathcal{B}}, \mathsf{Mlist}, \mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\mathcal{B}^*}, n_c, n_r)$
  Run **Resume**$(\mathcal{A}, \mathsf{state}_{\mathcal{A}})$
    if $|\mathsf{Mlist}| > 0$ **then**
      Return $\mathsf{Mlist}[|\mathsf{Mlist}|]$ to $\mathcal{A}$'s previous query.
    if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
      $n_r \leftarrow n_r + 1$ ▷ Randomness query counter
      $j \leftarrow 0$
      repeat
        $j \leftarrow j + 1$
        $C \leftarrow \mathsf{Encrypt}(pk, 0)$
        if $j = \lambda$ **then**
          Abort ▷ $\mathcal{A}^*$ Fails
      until $H_k(C) = \mathsf{Rlist}[n_r]$
      Append C to $\mathsf{Clist}_{\mathcal{B}}$
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**

$n_c \leftarrow n_c + 1$ ▷ Encryption query counter
Append $\mathsf{Clist}[n_c]$ to $\mathsf{Clist}_{\mathcal{B}}$
return $\mathsf{Clist}[n_c]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, k||R[\mathcal{A}], \mathsf{Clist}_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
Append m to Mlist
$\mathsf{state}_{\mathcal{A}} \leftarrow \mathbf{Suspend}(\mathcal{A})$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow (k, \mathsf{Clist}_{\mathcal{B}}, \mathsf{Mlist}, \mathsf{state}_{\mathcal{A}}, \mathsf{state}_{\mathcal{B}^*}, n_c, n_r)$
return $(m, \mathsf{state}_{\mathcal{A}^*})$

We now show that $\mathbf{Adv}^{\text{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_\beta,D}(\lambda)$ is negligible in λ for any distinguishing algorithm D and all $\beta \in \{0,1\}$. Choose $\beta \in \{0,1\}$, fix a distinguishing algorithm D, let x_i be the output of $\mathcal{A}$ in Game i and let S_i be the event that $D(x_i) = 1$.

Game 0: Let Game 0 be $\mathbf{Expt}^{\text{PA2+Fake}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P}_\beta,D}(\lambda)$.

Game 1: We modify the game so that the lists Clist and Rlist that were computed by $\mathcal{A}^*$ in Game 0 are instead computed by the challenger. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{\text{R}} \{0,1\}^{2\ell(\lambda)+2}$
$R[\mathcal{A}] \xleftarrow{\text{R}} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R[\mathcal{B}] \leftarrow k||R[\mathcal{A}]$
Run $x_1 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist}_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$m \leftarrow \mathcal{P}_\beta(pk, s)$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
Append C to $\mathsf{Clist}_{\mathcal{B}}$
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\rho \xleftarrow{\text{R}} \{0,1\}$
repeat
$j \leftarrow j + 1$

if $j = \lambda$ **then**
Abort
$C \leftarrow \mathsf{Encrypt}(pk, 0)$
until $H_k(C) = \rho$
Append C to $\mathsf{Clist}_{\mathcal{B}}$
return ρ
$b'_1 \leftarrow D(x_1)$
return b'_1

Since the challenger computes Clist and Rlist using the same inputs and randomness that $\mathcal{A}^*$ did in game 0, their values in game 1 are unchanged, i.e. this is a bridging step. So $\Pr[S_1] = \Pr[S_0]$.

Game 2: We modify the $\mathsf{Randomness}$ oracle so that it puts the ciphertexts it generates onto Clist. Since this makes Clist and $\mathsf{Clist}_{\mathcal{B}}$ identical, we also remove $\mathsf{Clist}_{\mathcal{B}}$ and just use Clist instead. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
$R[\mathcal{A}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R[\mathcal{B}] \leftarrow k||R[\mathcal{B}]$
Run $x_2 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$\boxed{(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})}$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$m \leftarrow \mathcal{P}_\beta(pk, s)$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\rho \xleftarrow{R} \{0,1\}$
repeat
$j \leftarrow j + 1$
if $j = \lambda$ **then**
Abort
$C \leftarrow \mathsf{Encrypt}(pk, 0)$
until $H_k(C) = \rho$
$\boxed{\text{Append } C \text{ to } \mathsf{Clist}}$
return ρ

$b_2' \leftarrow D(x_2)$
return b_2'

In Game 2, the ciphertexts that were generated by the randomness oracle are added to Clist, and $\mathcal{A}$ may not query $\mathsf{Decrypt}(C)$ for any ciphertext $C \in \mathsf{Clist}$. These ciphertexts are not given to $\mathcal{A}$, but $\mathcal{A}$ could by chance generate such a ciphertext independently. Let F be the event that $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ for some ciphertext C which was added to Clist in response to a randomness query.

For a given element $C \in \mathsf{Clist}$ which was computed as part of a randomness query, and for a given $\mathsf{Decrypt}$ query C',

$$\begin{aligned} \Pr[C = C'] &\leq \max_{y \in \mathbf{CiphSp}(pk)} \Pr[C = y] \\ &\leq \sqrt{\kappa(C)} \end{aligned}$$

by Lemma 2.7.4. But $\kappa(C) \leq \mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)$ by Lemma 2.7.6, so

$$\Pr[C = C'] \leq \sqrt{\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}\,.$$

Since $\mathcal{A}$ makes at most q_d decryption queries and at most q_r randomness queries, we see that

$$\Pr[F] \leq q_d q_r \sqrt{\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}\,,$$

So

$$|\Pr[S_2] - \Pr[S_1]| \leq q_d q_r \sqrt{\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}\,.$$

Game 3: We modify the randomness oracle so that it does not limit the

number of attempts to find C such that $H_k(C) = \rho$.

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
$R[\mathcal{A}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R[\mathcal{B}] \leftarrow k||R[\mathcal{B}]$
Run $x_3 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $m \leftarrow \mathcal{P}_\beta(pk, s)$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\rho \xleftarrow{R} \{0,1\}$
    repeat
      $C \leftarrow \mathsf{Encrypt}(pk, 0)$
    until $H_k(C) = \rho$
    Append C to Clist
    return ρ
$b'_3 \leftarrow D(x_3)$
return b'_3

Note that this game no longer runs in polynomial time (although the expected running time is still polynomial). This is not a problem, as there is no need for the challenger to be efficient. We will restore the polynomial running time in the next game hop.

Let F_i be the event that Game 3 aborts because $j = \lambda$ during the i^{th} randomness query, and let $F = \bigcup_{i=1}^{q_r} F_i$. We now compute an upper bound on $\Pr[F]$. As noted earlier, $\kappa(C) \leq \mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)$, where $\mathcal{C}$ is as in Lemma 2.7.6. By Lemma 2.7.5, it follows that the statistical distance $\Delta((k, H_k(C)), (k, \rho))$ where $\rho \xleftarrow{R} \{0,1\}$ is negligible as a function of λ. Thus for sufficiently large values of λ, $\Delta((k, H_k(C)), (k, b)) \leq 1/6$. This implies that $\frac{1}{3} \leq \Pr[H_k(C) = 0] = (1 - \Pr[H_k(C) = 1]) \leq \frac{2}{3}$, where the probability statements are taken over

the random choice of k and the coins used to compute $C \leftarrow \mathsf{Encrypt}(pk, 0)$. Thus $\Pr[F_i] \leq (\frac{2}{3})^\lambda$ for all i and so $\Pr[F] \leq q_r(\frac{2}{3})^\lambda$ which is negligible. If the event F does not occur then the two games proceed identically, so

$$|\Pr[S_3] - \Pr[S_2]| \leq q_r \left(\frac{2}{3}\right)^\lambda .$$

Game 4: We modify the randomness oracle so that instead of choosing ρ at random, it computes a ciphertext $C' \leftarrow \mathsf{Encrypt}(pk, 0)$ and sets $\rho \leftarrow H_k(C)$.

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
$R[\mathcal{A}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R[\mathcal{B}] \leftarrow k||R[\mathcal{B}]$
Run $x_4 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $m \leftarrow \mathcal{P}_\beta(pk, s)$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $\boxed{C' \leftarrow \mathsf{Encrypt}(pk, 0)}$
    $\boxed{\rho \leftarrow H_k(C')}$
    repeat
      $C \leftarrow \mathsf{Encrypt}(pk, 0)$
    until $H_k(C) = \rho$
    Append C to Clist
    return ρ
$b'_4 \leftarrow D(x_4)$
return b'_4

Consider an algorithm G which takes a security parameter λ and a sequence of bits $(\rho_1, \ldots, \rho_{q_r})$ as input and then runs according to the definition of Game 3 except that instead of choosing random bits to respond to randomness

queries, it uses the sequence provided as input. Let $(\rho_1, \ldots, \rho_{q_r}) \xleftarrow{R} \{0,1\}^{q_r}$, let $C_i \leftarrow \mathsf{Encrypt}(pk, 0)$ and let $\rho'_i \leftarrow H_k(C')$ for $i = 1$ to q_r. Then $\Pr[S_3] = \Pr[G(\lambda, \rho_1, \ldots, \rho_{q_r}) = 1]$ and $\Pr[S_4] = \Pr[G(\lambda, \rho'_1, \ldots, \rho'_{q_r}) = 1]$ by definition, since the code is identical.

By Lemma 2.7.6, there is an IND-CPA adversary $\mathcal{C}$ such that $\kappa(C) = \mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)$. Thus by Theorem 2.7.5, $\Delta[(k, \rho'_1, \ldots, \rho'_{q_r}), (k, \rho_1, \ldots, \rho_{q_r})] \leq q_r\sqrt{2\kappa(C)}/2$, where q_r is an upper bound on the number of randomness queries made by $\mathcal{A}$.

Thus by Lemma 1.2.1,

$$\begin{aligned}\Delta[x_4, x_3] &= \Delta[G(\rho'_1, \ldots, \rho'_{q_r}), G(\rho_1, \ldots, \rho_{q_r})] \\ &\leq \Delta[(\rho'_1, \ldots, \rho'_{q_r}), (\rho_1, \ldots, \rho_{q_r})] \\ &\leq q_r\sqrt{2\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}/2\,.\end{aligned}$$

So

$$|\Pr[S_4] - \Pr[S_3]| \leq q_r\sqrt{2\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}/2\,.$$

Game 5: We modify the $\mathsf{Randomness}$ oracle so that it simply computes $C \leftarrow \mathsf{Encrypt}(pk, 0)$ and lets $\rho \leftarrow H_k(C)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{R} \{0,1\}^{2\ell(\lambda)+2}$
$R[\mathcal{A}] \xleftarrow{R} \{0,1\}^{T_{\mathcal{A}}(\lambda)}$
$R[\mathcal{B}] \leftarrow k||R[\mathcal{B}]$
Run $x_5 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $(m, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(pk, C, R[\mathcal{B}], \mathsf{Clist}, \mathsf{state}_{\mathcal{B}^*})$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $m \leftarrow \mathcal{P}_\beta(pk, s)$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist

return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\boxed{C \leftarrow \mathsf{Encrypt}(pk, 0)}$
$\boxed{\rho \leftarrow H_k(C)}$
Append C to Clist
return ρ
$b'_5 \leftarrow D(x_5)$
return b'_5

This is a bridging step; in game 4 C and C' are independent and identically distributed (i.e. they are drawn from the distribution of $C'' \leftarrow \mathsf{Encrypt}(pk, 0)$ restricted to those ciphertexts C'' such that $H_k(C'') = \rho$). As C' is not used elsewhere, we may replace C' with C.

Game 6: Note that Game 5 exactly simulates $\mathbf{Expt}^{\mathrm{PA2-Fake}}_{\Pi,\mathcal{B},\mathcal{B}^*,\mathcal{P}_\beta,D}(\lambda)$. Let Game 6 be $\mathbf{Expt}^{\mathrm{PA2-Real}}_{\Pi,\mathcal{B},\mathcal{P}_\beta,D}(\lambda)$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{\mathrm{R}} \{0,1\}^{2\ell(\lambda)+2}$
Run $x_6 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$\boxed{m \leftarrow \mathsf{Decrypt}(sk, C)}$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$m \leftarrow \mathcal{P}_\beta(pk, s)$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$C \leftarrow \mathsf{Encrypt}(pk, 0)$
$\rho \leftarrow H_k(C)$
Append C to Clist
return ρ
$b'_6 \leftarrow D(x_6)$
return b'_6

By the 2PA2 property of Π,

$$|\Pr[S_6] - \Pr[S_5]| \leq \mathbf{Adv}^{\mathrm{2PA2}}_{\Pi,\mathcal{B},\mathcal{P}_\beta,D}(\lambda)\,,$$

which is negligible in λ.

Game 7: We modify the randomness oracle so that it no longer adds C to Clist. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{\mathrm{R}} \{0,1\}^{2\ell(\lambda)+2}$
Run $x_7 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $m \leftarrow \mathsf{Decrypt}(sk, C)$
    return m
  if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
    $m \leftarrow \mathcal{P}_\beta(pk, s)$
    $C \leftarrow \mathsf{Encrypt}(m)$
    Append C to Clist
    return C
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $C \leftarrow \mathsf{Encrypt}(pk, 0)$
    $\rho \leftarrow H_k(C)$
    return ρ
$b'_7 \leftarrow D(x_7)$
return b_7

By the same reasoning as in Game 2,

$$|\Pr[S_7] - \Pr[S_6]| \leq q_d q_r \sqrt{\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}\,.$$

Game 8: We modify the randomness oracle again, so that it simply chooses $\rho \xleftarrow{\mathrm{R}} \{0,1\}$ again. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$k \xleftarrow{\mathrm{R}} \{0,1\}^{2\ell(\lambda)+2}$
Run $x_8 \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(pk)$

if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$m \leftarrow \mathsf{Decrypt}(sk, C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(s)$ **then**
$m \leftarrow \mathcal{P}_\beta(pk, s)$
$C \leftarrow \mathsf{Encrypt}(m)$
Append C to Clist
return C
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$\boxed{\rho \xleftarrow{R} \{0,1\}}$
return ρ
$b'_8 \leftarrow D(x_8)$
return b'_8

By the same reasoning as in Game 4,

$$|\Pr[S_8] - \Pr[S_7]| \leq q_r \sqrt{2\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda)}/2\,.$$

Finally, we see that Game 8 is exactly $\mathbf{Expt}^{\mathrm{PA2+Real}}_{\Pi,\mathcal{A},\mathcal{P}_\beta,D}(\lambda)$. Putting it all together we see that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{PA2+}}_{\Pi,\mathcal{A},\mathcal{A}^*,\mathcal{P},D}(\lambda) &= |\Pr[S_7] - \Pr[S_0]| \\
&\leq |\Pr[S_7] - \Pr[S_6]| + \ldots + |\Pr[S_1] - \Pr[S_0] \\
&= (2q_r/\sqrt{2} + 2q_d q_r)\mathbf{Adv}^{\mathrm{IND-CPA}}_{\Pi,\mathcal{C}}(\lambda) + \mathbf{Adv}^{\mathrm{2PA2}}_{\Pi,\mathcal{B},\mathcal{P}_\beta,D}(\lambda) \\
&\quad + q_r \left(\frac{2}{3}\right)^{\lambda}
\end{aligned}$$

which is negligible in λ as required.

□

Corollary 2.7.8. *Suppose a public key encryption scheme* Π *is PA2 plaintext aware, and IND-CPA secure. Then* Π *is PA2+ plaintext aware.*

Proof. Since Π is PA2 plaintext aware and IND-CPA secure, we have that it

is LH-IND-CCA2 secure by Theorem 2.6.2. Since Π is 2PA2 plaintext aware and IND-CPA, we have that it is 2PA2+ plaintext aware by Theorem 2.7.7. Since Π is LH-IND-CCA2 secure, and is 2PA2+ plaintext aware, we have that it is PA2+ plaintext aware by Theorem 2.5.2. □

2.8 Conclusion

Theorem 2.6.2 shows that the PA2 definition rules out encryption schemes which allow arbitrarily long messages, and I think this is a valuable property for an encryption scheme to have. On the other hand, Theorem 2.4.6 shows that for schemes where the comparison makes sense, 2PA2 is as good as PA2 plaintext awareness. For these reasons, it is my belief that although 2PA2 is a weaker definition than PA2, it is more useful because it can cover a wider range of encryption schemes.

Chapter 3

Encryption Schemes based on Hash Proof Systems

Hash proof systems based on subset membership problems were introduced by Cramer and Shoup [11] in 2002 to generalise their earlier, DDH-based, encryption scheme [10] to a wider class of computational problems. Kurosawa and Desmedt [22] proposed a more efficient scheme in 2004 based on the DDH problem and which makes use of similar techniques to the original Cramer-Shoup scheme. In the same paper, Kurosawa and Desmedt presented a version of their scheme based on hash proof systems.

These constructions based on hash proof systems are one of only two generic constructions of IND-CCA2 secure public-key encryption schemes in the standard model, the other being the Canetti-Halevi-Katz construction, which uses an IND-CPA secure identity-based encryption scheme. The schemes based on hash-proof systems are a particularly important group, because there

are instantiations based on a diverse selection of computational problems, including the decisional Diffie-Hellman (DDH) problem, the quadratic residuosity problem (QR), the decision composite residuosity (DCR) problem and others. This wide range of assumptions gives us some flexibility: if, for example, the DDH problem was found to be easy on elliptic curves, we could still rely on the DCR based instantiation of the scheme. We will discuss the various constructions of hash proof systems further in Section 3.3.

3.1 Definitions

In this chapter, we present definitions for hash proof systems, and concepts introduced by Dent [14] to show that the Cramer-Shoup encryption scheme is PA2 plaintext aware. In the next two chapters we will apply these techniques to the generalised Kurosawa-Desmedt and Cramer-Shoup schemes based on hash proof systems are PA2 plaintext aware.

3.1.1 Simulatability

Dent [14] showed that a variant of the standard discrete-log-based Cramer-Shoup encryption scheme is PA2 plaintext aware by first showing it was PA1+ plaintext aware, and that is satisfies a notion he called "simulatability". This property captures the idea that there is a pair of functions, which we call a simulator, mapping random ciphertexts to random-looking bit strings and vice-versa. Essentially this allows you to "simulate" the output of an encryption oracle in the PA2 game using the randomness oracle of the PA1+ game. The definitions presented here are based on the work of Dent [14] but are presented with significantly increased formalism.

Simulatable Sets

A simulator for a family of collections of finite sets $\mathbf{S} = ((S_i)_{i \in I_\lambda})_{\lambda \in \mathbb{N}}$ indexed by a set I_λ is a tuple $\mathbf{f_S} = (f_\mathbf{S}, f_\mathbf{S}^{-1}, n_\mathbf{S})$ where $n_\mathbf{S} : \mathbb{N} \to \mathbb{N}$ is a polynomial (called the length function) and $(f_\mathbf{S}, f_\mathbf{S}^{-1})$ is a pair of polynomial-time algorithms with the following properties:

- $f_\mathbf{S}$ is a deterministic algorithm which takes as input a security parameter 1^λ, an index $i \in I_\lambda$, and $r \in \{0,1\}^{n_\mathbf{S}(\lambda)}$, and returns an element $s \in S_i$.
- $f_\mathbf{S}^{-1}$ is a probabilistic algorithm which takes as input a security parameter 1^λ, an index $i \in I_\lambda$, and $s \in S_i$, and returns a string $r \in \{0,1\}^{n_\mathbf{S}(\lambda)}$.
- For all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, $s \in S_i$, $f_\mathbf{S}(1^\lambda, i, f_\mathbf{S}^{-1}(1^\lambda, i, s)) = s$.

We require a simulator for $\mathbf{S}$ to satisfy two security properties, related to indistinguishability. An adversary $\mathcal{A}$ against the Set-Sim property of $\mathbf{f_S}$ is a probabilistic, polynomial-time algorithm which takes a security parameter 1^λ, an index $i \in I_\lambda$, and an element $s \in S_i$ as input, and returns a bit b. We require that no adversary can distinguish a randomly chosen element of the set from an element chosen using the simulator. We define experiments Set-Sim-0 and Set-Sim-1 which capture this notion as follows:

$\mathbf{Expt}^{\text{Set-Sim-0}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda)$
$\quad i \xleftarrow{R} I_\lambda$
$\quad s \xleftarrow{R} S_i$
$\quad b' \leftarrow \mathcal{A}(1^\lambda, i, s)$
$\quad$ **return** b

$\mathbf{Expt}^{\text{Set-Sim-1}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda)$
$\quad i \xleftarrow{R} I_\lambda$
$\quad r \xleftarrow{R} \{0,1\}^{n_\mathbf{S}(\lambda)}$
$\quad s \xleftarrow{R} f_\mathbf{S}(1^\lambda, i, r)$
$\quad b' \leftarrow \mathcal{B}(1^\lambda, i, s)$
$\quad$ **return** b

For all $\lambda \in \mathbb{N}$, we define the Set-Sim advantage of $\mathcal{A}$ as:

$$\mathbf{Adv}^{\text{Set-Sim}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda) = \left| \Pr[\mathbf{Expt}^{\text{Set-Sim-0}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\text{Set-Sim-1}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda) = 1] \right|.$$

A Rand-Sim adversary $\mathcal{B}$ against the randomness of $\mathbf{f_S}$ is a probabilistic, polynomial-time algorithm which takes a security parameter 1^λ, an index $i \in I_\lambda$ and an string $r \in \{0,1\}^{n_\mathbf{S}(\lambda)}$ as input, and returns a bit b'. We require that no adversary should be able to distinguish a randomly chosen string $r \in \{0,1\}^{n_\mathbf{S}(\lambda)}$ from $f_\mathbf{S}^{-1}(1^\lambda, i, f_\mathbf{S}(1^\lambda, i, r))$. Instead of using the simulator and the inverse, we could have selected an element of S_i and used only the inverse f^{-1}, but our method has the advantage that we can use the same Rand-Sim definition when we come to consider simulatable encryption schemes. We define the two experiments as follows:

$\mathbf{Expt}^{\text{Rand-Sim-0}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda)$
$i \xleftarrow{R} I_\lambda$
$r \xleftarrow{R} \{0,1\}^{n_\mathbf{S}(\lambda)}$
$b' \leftarrow \mathcal{B}(1^\lambda, i, r)$
return b

$\mathbf{Expt}^{\text{Rand-Sim-1}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda)$
$i \xleftarrow{R} I_\lambda$
$r \xleftarrow{R} \{0,1\}^{n_\mathbf{S}(\lambda)}$
$r' \leftarrow f_\mathbf{S}^{-1}(1^\lambda, i, f_\mathbf{S}(1^\lambda, i, r))$
$b' \leftarrow \mathcal{B}(1^\lambda, i, r')$
return b

For all $\lambda \in \mathbb{N}$, we define the Rand-Sim advantage of $\mathcal{A}$ as:

$$\mathbf{Adv}^{\text{Rand-Sim}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda) = |\Pr[\mathbf{Expt}^{\text{Rand-Sim-0}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\text{Rand-Sim-1}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda) = 1]|\,.$$

Definition 3.1.1 (Computationally Simulatable Sets)**.** *A family of sets* $\mathbf{S} = ((S_i)_{i \in I_\lambda})_{\lambda \in \mathbb{N}}$ *is computationally simulatable if there exists a simulator* $\mathbf{f_S} = (f_\mathbf{S}, f_\mathbf{S}^{-1}, n_\mathbf{S})$ *such that:*

- *For all probabilistic polynomial-time Set-Sim adversaries* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\text{Set-Sim}}_{\mathbf{S},\mathcal{A},\mathbf{f_S}}(\lambda)$ *is negligible in* λ*, and*
- *For all probabilistic polynomial-time Rand-Sim adversaries* $\mathcal{B}$*, the advantage* $\mathbf{Adv}^{\text{Rand-Sim}}_{\mathbf{S},\mathcal{B},\mathbf{f_S}}(\lambda)$ *is negligible in* λ*.*

Definition 3.1.2 (Statistically Simulatable Sets)**.** *A family* $\mathbf{S} = ((S_i)_{i \in I_\lambda})_{\lambda \in \mathbb{N}}$ *of sets is statistically simulatable if there exists a simulator* $\mathbf{f_S} = (f_\mathbf{S}, f_\mathbf{S}^{-1}, n_\mathbf{S})$ *such that*

- *For all* $\lambda \in \mathbb{N}$*,* $i \in I_\lambda$*, for* $r \xleftarrow{R} \{0,1\}^{n_\mathbf{S}(\lambda)}$*, for* $s \xleftarrow{R} S_i$*, and for* $s' \leftarrow f_\mathbf{S}(1^\lambda, i, r)$*, the statistical distance* $\Delta[(1^\lambda, i, s), (1^\lambda, i, s')]$ *is negligible as a function of* λ*.*

- *For all* $\lambda \in \mathbb{N}$*,* $i \in I_\lambda$*, for* $r \xleftarrow{R} \{0,1\}^{n_\mathbf{S}(\lambda)}$ *and for* $r' \leftarrow f_\mathbf{S}^{-1}(1^\lambda, i, f_\mathbf{S}(1^\lambda, i, r))$*, the statistical distance* $\Delta[(1^\lambda, i, r), (1^\lambda, i, r')]$ *is negligible as a function of* λ*.*

Note that these are simply statistical analogues of the computational Set-Sim and Rand-Sim properties respectively.

Simulatable Public-Key Encryption Schemes

We say a public-key encryption scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is simulatable if there exists a polynomial-time simulator $\mathbf{f}_\Pi = (f_\Pi, f_\Pi^{-1}, n_\Pi)$ for the family $(\mathbf{CiphSp}(pk)_{pk \in PK_\lambda})_{\lambda \in \mathbb{N}}$, where PK_λ is the set of public keys, i.e $PK_\lambda = \{pk \in \{0,1\}^* | \Pr[\mathsf{KeyGen}(1^\lambda) = (pk, sk) \text{ for some } sk] > 0\}$.

A PKE-Sim adversary against the simulatability of a public-key encryption scheme is a pair of probabilistic polynomial-time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$, takes a security parameter 1^λ and a public key pk as input, and returns a message $m \in \mathbf{MsgSp}(pk)$ and some state information state, and $\mathcal{A}_2$ takes a security parameter 1^λ, a ciphertext $C^* \in \mathbf{CiphSp}(pk)$ and the value state that was returned by $\mathcal{A}_1$ as input, and returns a single bit b'. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to an oracle $\mathsf{Decrypt}$ which takes a ciphertext C as input and returns $\mathsf{Decrypt}(sk, C)$, but $\mathcal{A}_2$ may not query $\mathsf{Decrypt}$ on the challenge

ciphertext C^*. We require that it is hard to distinguish a valid encryption of the message m returned by $\mathcal{A}_1$ from a simulated encryption computed using f_Π. We now describe the PKE-Simulatability experiment:

$\mathbf{Expt}^{\text{PKE-Sim-0}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, pk)$
$C^* \leftarrow \mathsf{Encrypt}(pk, m)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, C^*, \mathsf{state})$
return b'

$\mathbf{Expt}^{\text{PKE-Sim-1}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, pk)$
$r \xleftarrow{\text{R}} \{0,1\}^{n_\Pi(\lambda)}$
$C^* \leftarrow f_\Pi(pk, r)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, C^*, \mathsf{state})$
return b'

For all $\lambda \in \mathbb{N}$, we define the PKE-Sim advantage of $\mathcal{A}$ as:

$$\mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) = \left|\Pr[\mathbf{Expt}^{\text{PKE-Sim-1}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\text{PKE-Sim-0}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) = 1]\right|.$$

Definition 3.1.3. *A public-key encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is simulatable if there exists a simulator* $\mathbf{f}_\Pi = (f_\Pi, f_\Pi^{-1}, n_\Pi)$ *such that:*

- *For all probabilistic polynomial-time PKE-Sim adversaries* $\mathcal{A}$*, the advantage* $\mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$ *is negligible in* λ*.*
- *For all probabilistic polynomial-time Rand-Sim adversaries* $\mathcal{B}$*, the advantage* $\mathbf{Adv}^{\text{Rand-Sim}}_{\Pi,\mathcal{B},\mathbf{f}_\Pi}(\lambda)$ *is negligible in* λ*.*

We only define computational simulatability for encryption schemes for two reasons: Firstly, we will not need statistical notions of simulatability for encryption schemes. Secondly, if we did define a statistical analogue of this notion, it is clear that no public-key encryption scheme could meet it, since that would contradict the soundness requirement of the scheme.

Simulatable Symmetric Encryption Schemes

Let $\mathbf{f}_\Sigma = (f_\Sigma, f_\Sigma^{-1}, n_\Sigma)$ be a polynomial-time simulator for $(\mathbf{CiphSp}(\lambda))_{\lambda\in\mathbb{N}}$. An adversary against the simulatability of a DEM $\Sigma = (\mathsf{enc}, \mathsf{dec})$ is a pair of probabilistic polynomial-time algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that $\mathcal{A}_1$ takes a security parameter 1^λ as input and returns a message $m \in \mathbf{MsgSp}(\lambda)$ and some state information state, and $\mathcal{A}_2$ takes a ciphertext $\chi^* \in \mathbf{CiphSp}(\lambda)$ and the value state that was returned by $\mathcal{A}_1$ as input and returns a single bit b'. $\mathcal{A}_1$ and $\mathcal{A}_2$ have access to an oracle dec which takes a ciphertext χ as input and returns a message m. $\mathcal{A}_2$ may not query dec on the challenge ciphertext χ^*. We now describe the DEM-Simulatability experiment:

$\mathbf{Expt}^{\mathrm{DEM-Sim-0}}_{\Sigma,\mathcal{A},\mathbf{f}_\Sigma}(\lambda)$
$\quad \kappa \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
$\quad (m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{dec}(\kappa,\cdot)}(1^\lambda)$
$\quad \chi^* \leftarrow \mathsf{enc}(\kappa, m)$
$\quad b' \leftarrow \mathcal{A}_2^{\mathsf{dec}(\kappa,\cdot)}(\chi^*, \mathsf{state})$
$\quad$ **return** b'

$\mathbf{Expt}^{\mathrm{DEM-Sim-1}}_{\Sigma,\mathcal{A},\mathbf{f}_\Sigma}(\lambda)$
$\quad \kappa \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
$\quad (m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{dec}(\kappa,\cdot)}(1^\lambda)$
$\quad r \xleftarrow{\mathrm{R}} \{0,1\}^{n(\lambda)}$
$\quad \chi^* \leftarrow f_\Sigma(1^\lambda, r)$
$\quad b' \leftarrow \mathcal{A}_2^{\mathsf{dec}(\kappa,\cdot)}(\chi^*, \mathsf{state})$
$\quad$ **return** b'

For all $\lambda \in \mathbb{N}$, we define the DEM-Sim advantage of $\mathcal{A}$ as:

$$\mathbf{Adv}^{\mathrm{DEM-Sim}}_{\Sigma,\mathcal{A},\mathbf{f}_\Sigma}(\lambda) = \left|\Pr[\mathbf{Expt}^{\mathrm{DEM-Sim-1}}_{\Pi,\mathcal{A},\mathbf{f}_\Sigma}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathrm{DEM-Sim-0}}_{\Sigma,\mathcal{A},\mathbf{f}_\Sigma}(\lambda) = 1]\right|.$$

Definition 3.1.4. *A DEM $\Sigma = (\mathsf{enc}, \mathsf{dec})$ is simulatable if there exists a simulator $\mathbf{f}_\Sigma = (f_\Sigma, f_\Sigma^{-1}, n_\Sigma)$ such that*

- *For all probabilistic polynomial-time DEM-Sim adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\mathrm{DEM-Sim}}_{\Sigma,\mathcal{A},f_\Sigma}(\lambda)$ is negligible in λ, and*

- *For all probabilistic polynomial-time Rand-Sim adversaries, the advantage* $\mathbf{Adv}^{\mathrm{Rand-Sim}}_{\Sigma,\mathcal{A},f_\Sigma}(\lambda)$ *is negligible in* λ.

As with the public-key case, we will not need statistical notions of simulatability for DEMs.

3.1.2 Differences from Dent's formulation

The notions of simulatability presented above are based on those introduced by Dent [14]. We have modified Dent's definitions in several ways; we describe the changes below:

- Dent defined simulatable groups rather than simulatable sets; this difference is irrelevant, since the definition does not rely on the group operation in any way, only the underlying set.

- Dent's definitions did not take into account the fact that the size of the set or group in question must depend on the security parameter. We address this by defining simulatability for an infinite family of sets rather than a single set.

- Dent only defined computational notions of simulatability, while the statistical notions appear to be necessary for some parts of the proof.

- In Dent's definitions, the adversary is given access to an oracle which will supply elements of the set (or ciphertext space of an encryption scheme), rather than providing a single element as we have done here. We feel our formalism is clearer, and will now show that the two definitions are equivalent.

3.1.3 Single vs. Multiple element Simulatability

The biggest change we have made to the definition is that in Dent's model, instead of providing a single element, the adversary may query an oracle as many times as it likes to get new element sampled according to the appropriate distribution.

Dent's variant of the PKE-Sim property is defined as follows:

$\mathbf{Expt}^{\text{Dent-PKE-Sim-0}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  Run $b \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(1^\lambda, pk)$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      $m \leftarrow \mathsf{Decrypt}(sk, C)$
      return m
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(m)$ **then**
      $C \leftarrow \mathsf{Encrypt}(pk, m)$
      return C
  return b

$\mathbf{Expt}^{\text{Dent-PKE-Sim-1}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$
  Run $b \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Encrypt}}(1^\lambda, pk)$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      $m \leftarrow \mathsf{Decrypt}(sk, C)$
      return m
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(m)$ **then**
      $r \xleftarrow{\text{R}} \{0,1\}^{n_\Pi(\lambda)}$
      $C^* \leftarrow f_\Pi(pk, r)$
  return b

In the above, $\mathcal{A}$ may not query the $\mathsf{Decrypt}$ oracle on any ciphertext C returned by the $\mathsf{Encrypt}$ oracle. For all $\lambda \in \mathbb{N}$, we define the Dent-PKE-Sim advantage of $\mathcal{A}$ as:

$$\begin{aligned}\mathbf{Adv}^{\text{Dent-PKE-Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) \;=\; &|\Pr[\mathbf{Expt}^{\text{Dent-PKE-Sim-1}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) = 1] \\ &- \Pr[\mathbf{Expt}^{\text{Dent-PKE-Sim-0}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) = 1]|\end{aligned}$$

We will prove that our definition is equivalent up to a factor of the number of queries, which is bounded by a polynomial in the security parameter.

Lemma 3.1.5. *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be a public-key encryption scheme. Then for every Dent-PKE-Sim adversary* $\mathcal{A}$ *there is a sequence of*

adversaries $\mathcal{B}_1, \mathcal{B}_2, \ldots$, *such that for all* $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\text{Dent}-\text{PKE}-\text{Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) \leq \sum_{i=1}^{q_e(\lambda)} \mathbf{Adv}^{\text{PKE}-\text{Sim}}_{\Pi,\mathcal{B}_i,\mathbf{f}_\Pi}(\lambda)$$

where $q_e(\lambda)$ *is an upper bound on the number of encryption queries made by* $\mathcal{A}$ *when run in* $\mathbf{Expt}^{\text{PKE}-\text{Sim}-0}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$ *or* $\mathbf{Expt}^{\text{PKE}-\text{Sim}-1}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$. *Since* $\mathcal{A}$ *runs in polynomial time,* $q_e(\lambda)$ *is polynomially-bounded. In particular, this implies that if* Π *is simulatable then* $\mathbf{Adv}^{\text{Dent}-\text{PKE}-\text{Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$ *is negligible in* λ. *Conversely, for every PKE-Sim adversary* $\mathcal{C}$ *there is a Dent-PKE-Sim adversary* $\mathcal{D}$ *such that* $\mathbf{Adv}^{\text{PKE}-\text{Sim}}_{\Pi,\mathcal{C},\mathbf{f}_\Pi}(\lambda) = \mathbf{Adv}^{\text{Dent}-\text{PKE}-\text{Sim}}_{\Pi,\mathcal{D},\mathbf{f}_\Pi}(\lambda)$.

Proof. Let $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public-key encryption scheme and let $\mathbf{f}_\Pi = (f_\Pi, f_\Pi^{-1}, n_\Pi)$ be a simulator for Π. Let $\mathcal{A}$ be an adversary against the Dent-PKE-Sim property. We construct PKE-Sim adversaries $\mathcal{B}_1, \mathcal{B}_2, \ldots$ as follows:

function $\mathcal{B}_{j,1}(1^\lambda, pk)$
  Run $(m, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{Encrypt},\mathsf{Decrypt}}(1^\lambda, pk)$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
    if $\mathcal{A}$ queries $\mathsf{Encrypt}(m)$ **then**
      $n_e \leftarrow n_e + 1$
      if $n_e < j$ **then**
        $r \xleftarrow{\text{R}} \{0,1\}^{n_\Pi(\lambda)}$
        $C \leftarrow f_\Pi(pk, r)$
        return C
    else if $n_e = j$ **then**
      $\mathsf{state}_\mathcal{A} \leftarrow \mathbf{Suspend}(\mathcal{A})$
      $\mathsf{state}_\mathcal{B} \leftarrow (pk, \mathsf{state}_\mathcal{A})$
  return $(m, \mathsf{state}_\mathcal{B})$

function $\mathcal{B}_{j,2}(1^\lambda, C^*, \mathsf{state}_\mathcal{B})$
  Parse $\mathsf{state}_\mathcal{B}$ as $(pk, \mathsf{state}_\mathcal{A})$
  Run $b' \leftarrow \mathbf{Resume}(\mathcal{A}(\mathsf{state}_\mathcal{A}))$

return C^* in response to $\mathcal{A}$'s $\mathsf{Encrypt}$ query
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
Query $m \leftarrow \mathsf{Decrypt}(C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(m)$ **then**
$C \leftarrow \mathsf{Encrypt}(pk, m)$
return C
return b'

Fix $\lambda \in \mathbb{N}$. We now show that the advantage $\mathbf{Adv}^{\text{Dent-PKE-Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) \leq \sum_{i=1}^{q_e(\lambda)} \mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{B}_i,\mathbf{f}_\Pi}(\lambda)$ via a sequence of games: Game 0 to Game $q_e(\lambda)$

Game j: For $0 \leq j \leq q_e$, let Game j be as follows:

Run $b' \leftarrow \mathcal{A}(1^\lambda, pk)$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
Query $m \leftarrow \mathsf{Decrypt}(C)$
return m
if $\mathcal{A}$ queries $\mathsf{Encrypt}(m)$ **then**
$n_e \leftarrow n_e + 1$
if $n_e \leq j$ **then**
$r \xleftarrow{\text{R}} \{0,1\}^{n_\Pi(\lambda)}$
$C \leftarrow f_\Pi(pk, r)$
else if $n_e > j$ **then**
$C \leftarrow \mathsf{Encrypt}(pk, m)$
return C
return b

Let $S_j = \Pr[b = 1]$ in Game j. Since every encryption query is answered using the real $\mathsf{Encrypt}$ algorithm, Game 0 is equivalent to $\mathbf{Expt}^{\text{Dent-PKE-Sim-0}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda)$. Likewise, Game q_e is identical to $\mathbf{Expt}^{\text{Dent-PKE-Sim-1}}_{\Pi,\mathcal{B}_j,\mathbf{f}_\Pi}(\lambda)$. To complete the proof, we note that $\mathbf{Expt}^{\text{PKE-Sim-0}}_{\Pi,\mathcal{B}_j,\mathbf{f}_\Pi}(\lambda)$ exactly simulates the environment of Game $j-1$, and $\mathbf{Expt}^{\text{PKE-Sim-1}}_{\Pi,\mathcal{B}_i,\mathbf{f}_\Pi}(\lambda)$ exactly simulates the environment of Game j. Thus

$$\mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{B}_j,\mathbf{f}_\Pi}(\lambda) \leq |\Pr[S_{j-1}] - \Pr[S_j]|.$$

Putting it all together, we have

$$\begin{aligned}
\mathbf{Adv}^{\text{Dent-PKE-Sim}}_{\Pi,\mathcal{A},\mathbf{f}_\Pi}(\lambda) &= |\Pr[S_{q_e(\lambda)}] - \Pr[S_0]| \\
&\leq |\Pr[S_{q_e(\lambda)}] - \Pr[S_{q_e(\lambda)-1}]| + \ldots + |\Pr[S_1] - \Pr[S_0]| \\
&\leq \mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{B}_{q_e(\lambda)},\mathbf{f}_\Pi}(\lambda) + \ldots + \mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{B}_1,\mathbf{f}_\Pi}(\lambda) \\
&\leq q_e(\lambda) \max_{j=1}^{q_e(\lambda)} \mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{B}_j,\mathbf{f}_\Pi}(\lambda)
\end{aligned}$$

which is negligible in λ as required. For the converse, let $\mathcal{C}$ be an adversary against the PKE-Sim property of Π. We construct an adversary $\mathcal{D}$ against the Dent-PKE-Sim property as follows:

function $\mathcal{D}(1^\lambda, pk)$
  Run $(m, \mathsf{state}) \leftarrow \mathcal{C}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
    if $\mathcal{C}_1$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
  Query $C^* \leftarrow \mathsf{Encrypt}(m)$
  Run $b' \leftarrow \mathcal{C}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
    if $\mathcal{C}_2$ queries $\mathsf{Decrypt}(C)$ **then**
      Query $m \leftarrow \mathsf{Decrypt}(C)$
      return m
  return b'

$\mathcal{D}$ satisfies the requirement that it may not query $\mathsf{Decrypt}$ on any ciphertext returned by the encryption oracle by the corresponding property of $\mathcal{C}$. $\mathcal{D}$ exactly simulates the environment of $\mathcal{C}$, and $\mathcal{D}$ wins if and only if $\mathcal{C}$ does. Thus

$$\mathbf{Adv}^{\text{PKE-Sim}}_{\Pi,\mathcal{C},\mathbf{f}_\Pi}(\lambda) = \mathbf{Adv}^{\text{Dent-PKE-Sim}}_{\Pi,\mathcal{D},\mathbf{f}_\Pi}(\lambda)$$

as claimed. □

We may define similar (multi-query) security notions for DEM-Sim, Set-Sim and Rand-Sim. These multi-query security notions can be shown to be

equivalent to the single-query notions defined above. In particular, this means we may employ the following theorem:

Theorem 3.1.6 (Dent)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be an encryption scheme which is both simulatable and PA1+ plaintext aware. Then* Π *is PA2 plaintext aware.*

The proof of this is given as Theorem 3 in Dent's paper [14].

3.1.4 Subset Membership Problems

The following definitions are based on those originally presented by Cramer and Shoup [11]. A subset membership problem consists of a pair of probabilistic polynomial-time algorithms $\mathbf{M} = (\mathsf{ISA}, \mathsf{SSA})$ with the following properties:

- ISA, the *instance sampling algorithm*, takes a security parameter 1^λ as input and returns an instance $\Lambda = (X, L, W, R)$, where X, L and W are non-empty sets, and $R \subset X \times W$ is an NP relation for L, i.e a polynomial-time checkable relation with the property $x \in L \iff (x, w) \in R$ for some $w \in W$. For all $\lambda \in \mathbb{N}$, we let I_λ be the distribution defined by $\mathsf{ISA}(1^\lambda)$. We will write X_Λ for the set X associated with the instance Λ if necessary for clarity, or simply X if it is unambiguous.

- SSA, the *subset sampling algorithm* takes a security parameter 1^λ and an instance $\Lambda \in I_\lambda$ as input, and returns a pair (x, w), where x is uniformly distributed on L, and $(x, w) \in R$.

We will assume that X is polynomial-time decidable, i.e, there is a deterministic polynomial-time algorithm which takes a security parameter 1^λ, an instance $\Lambda \in I_\lambda$, and a bitstring $\zeta \in \{0,1\}^*$, and returns 1 if $\zeta \in X_\Lambda$ and 0 otherwise.

An adversary $\mathcal{A}$ against the hardness of a subset membership problem $\mathbf{M}$ is a probabilistic polynomial-time algorithm which takes a security parameter 1^λ, an instance $\Lambda \in I_\lambda$ and an element $x \in X$ as input, and returns a bit b. We define two experiments:

$\mathbf{Expt}^{\mathrm{SMP-L}}_{\mathbf{M},\mathcal{A}}(\lambda)$
$\quad \Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\quad (x, w) \leftarrow \mathsf{SSA}(\Lambda)$
$\quad b \leftarrow \mathcal{A}(1^\lambda, \Lambda, x)$
$\quad$ **return** b

$\mathbf{Expt}^{\mathrm{SMP-X\backslash L}}_{\mathbf{M},\mathcal{A}}(\lambda)$
$\quad \Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\quad x \xleftarrow{\mathrm{R}} X \setminus L$
$\quad b \leftarrow \mathcal{A}(1^\lambda, \Lambda, x)$
$\quad$ **return** b

Note that we do not assume it is possible to sample the distribution $x \xleftarrow{\mathrm{R}} X \setminus L$ in polynomial time.

Definition 3.1.7 (Subset Membership Problem)**.** *A subset membership problem is* hard *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, the advantage*

$$\mathbf{Adv}^{\mathrm{SMP}}_{\mathbf{M},\mathcal{A}}(\lambda) = |\Pr[\mathbf{Expt}^{\mathrm{SMP-X\backslash L}}_{\mathbf{M},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Expt}^{\mathrm{SMP-L}}_{\mathbf{M},\mathcal{A}}(\lambda) = 1]|$$

is negligible in λ.

We now show that if a subset membership problem is hard, no polynomial-time algorithm can distinguish X from $X \setminus L$.

Lemma 3.1.8. *Let $\mathbf{M}$ be a hard subset membership problem, and let $\Lambda = (X, L, W, R) \leftarrow \mathsf{ISA}_{\mathbf{M}}(1^\lambda)$. Let $x_1 \xleftarrow{R} X \setminus L$ and let $x_2 \xleftarrow{R} X$. Then for any polynomial-time algorithm $\mathcal{C}$, then $\epsilon = |\Pr[\mathcal{C}(x_1) = 1] - \Pr[\mathcal{C}(x_2) = 1]|$ is negligible.*

Proof. Let $x_3 \xleftarrow{R} L$. Then

$$\begin{aligned}
\Pr[\mathcal{C}(x_2) = 1] &= \Pr[\mathcal{C}(x_2) = 1 \mid x_2 \in X \setminus L] \Pr[x_2 \in X \setminus L] \\
&\quad + \Pr[\mathcal{C}(x_2) = 1 \mid x_2 \in L] \Pr[x_2 \in L] \\
&= \Pr[\mathcal{C}(x_2) = 1 \mid x_2 \in X \setminus L] \cdot \left(1 - \frac{|L|}{|X|}\right) \\
&\quad + \Pr[\mathcal{C}(x_2) = 1 \mid x_2 \in L] \cdot \frac{|L|}{|X|} \\
&= \Pr[\mathcal{C}(x_1) = 1] \cdot \left(1 - \frac{|L|}{|X|}\right) + \Pr[\mathcal{C}(x_3) = 1] \cdot \frac{|L|}{|X|},
\end{aligned}$$

since the distribution of x_2 conditioned on $x_2 \in X \setminus L$ is uniform on $X \setminus L$, and the distribution of x_2 conditioned on $x_2 \in L$ is uniform on L. So

$$\begin{aligned}
\epsilon &= |\Pr[\mathcal{C}(x_1) = 1] - \Pr[\mathcal{C}(x_2) = 1]| \\
&= \left|\Pr[\mathcal{C}(x_1) = 1] - \left(\Pr[\mathcal{C}(x_1) = 1] \cdot \left(1 - \frac{|L|}{|X|}\right) + \Pr[\mathcal{C}(x_3) = 1] \cdot \frac{|L|}{|X|}\right)\right| \\
&= \left|\Pr[\mathcal{C}(x_1) = 1] \cdot \frac{|L|}{|X|} - \Pr[\mathcal{C}(x_3) = 1] \cdot \frac{|L|}{|X|}\right| \\
&= \frac{|L|}{|X|} \cdot |\Pr[\mathcal{C}(x_1) = 1] - \Pr[\mathcal{C}(x_3) = 1]|.
\end{aligned}$$

Since $|\Pr[\mathcal{C}(x_1) = 1] - \Pr[\mathcal{C}(x_3) = 1]| = \mathbf{Adv}^{\mathrm{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda)$, it is negligible in λ by assumption, so $|\Pr[\mathcal{C}(x_1) = 1] - \Pr[\mathcal{C}(x_2) = 1]|$ is negligible as required. □

3.1.5 Projective Hash Families

We present definitions for universal projective hash families, originally defined by Cramer and Shoup [11]. Let X, S and Π be finite non-empty sets, and let $L \subset X$ be a non-empty subset of X. Let $H = (H_k)_{k \in K}$ be a collection of functions indexed by a set K such that $H_k : X \to \Pi$ and let $\alpha : K \to S$ be a polynomial-time computable function. The tuple $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ is

a projective hash family if the value $s = \alpha(k)$ uniquely determines the action of H_k on L, i.e. for all $k, k' \in K$ and for all $x \in L$, $\alpha(k) = \alpha(k') \implies H_k(x) = H_{k'}(x)$.

Definition 3.1.9 (Universal$_2$ Projective Hashing)**.** *A projective hash family* $\mathbf{H}$ *is* ϵ*-universal*$_2$ *if for all* $x \in X \setminus L$*, for all* $x^* \in X \setminus (L \cup \{x\})$*, for all* $\pi, \pi^* \in \Pi$*, for all* $s \in S$ *and for* $k \xleftarrow{R} K$*:*

$$\Pr[H_k(x) = \pi \wedge H_k(x^*) = \pi^* \wedge \alpha(k) = s] \leq \epsilon \Pr[H_k(x^*) = \pi^* \wedge \alpha(k) = s] .$$

A projective hash family $\mathbf{H}$ *is* ϵ*-universal if for all* $x \in X \setminus L$*, for all* $\pi \in \Pi$*, for all* $s \in S$ *and for* $k \xleftarrow{R} K$*:*

$$\Pr[H_k(x) = \pi \wedge \alpha(k) = s] \leq \epsilon \Pr[\alpha(k) = s] .$$

Lemma 3.1.10. *Let* $\mathbf{H}$ *be an* ϵ*-universal*$_2$ *projective hash family. Then* $\mathbf{H}$ *is* ϵ*-universal.*

Proof. For any x in $X \setminus L$, $\pi, \pi^* \in \Pi$, $s \in S$ and for $k \xleftarrow{\mathrm{R}} K$, fix an element $x^* \in X \setminus (L \cup \{x\})$. Then

$$\begin{aligned}
&\Pr[H_k(x) = \pi \wedge \alpha(k) = s] \\
&= \sum_{\pi^* \in \Pi} \Pr[H_k(x) = \pi \wedge H_k(x^*) = \pi^* \wedge \alpha(k) = s] \\
&\leq \sum_{\pi^* \in \Pi} \epsilon \Pr[H_k(x^*) = \pi^* \wedge \alpha(k) = s] \\
&= \epsilon \sum_{\pi^* \in \Pi} \Pr[H_k(x^*) = \pi^* \wedge \alpha(k) = s] \\
&= \epsilon \Pr[H_k(x^*) \in \Pi \wedge \alpha(k) = s] \\
&= \epsilon \Pr[\alpha(k) = s] .
\end{aligned}$$

□

3.1.6 Strong Universality

Kurosawa and Desmedt [22] define a projective hash family $\mathbf{H}$ to be *strongly*-universal if for all $x \in X \setminus L$, for all $\pi \in \Pi$, for all $s \in S$ and for $k \xleftarrow{R} K$:

$$\Pr[H_k(x) = \pi \wedge \alpha(k) = s] = \frac{1}{|\Pi|} \Pr[\alpha(k) = s]\,,$$

and projective hash family $\mathbf{H}$ is is strongly universal$_2$ if for all $x \in X \setminus L$, for all $x^* \in X \setminus (L \cup \{x\})$, for all $\pi, \pi^* \in \Pi$, for all $s \in S$ and for $k \xleftarrow{R} K$:

$$\Pr[H_k(x) = \pi \wedge H_k(x^*) = \pi^* \wedge \alpha(k) = s] = \frac{1}{|\Pi|} \Pr[H_k(x^*) = \pi^* \wedge \alpha(k) = s]\,.$$

Now, suppose that $\mathbf{H}$ is $\frac{1}{|\Pi|}$-universal, i.e for all $x \in X \setminus L$, for all $\pi \in \Pi$, for all $s \in S$ and for $k \xleftarrow{R} K$:

$$\Pr[H_k(x) = \pi \wedge \alpha(k) = s] \leq \frac{1}{|\Pi|} \Pr[\alpha(k) = s]\,,$$

and suppose also that there exist $\hat{x} \in X$, $\hat{\pi} \in \Pi$ and $\hat{s} \in S$ such that for $k \xleftarrow{R} K$

$$\Pr[H_k(\hat{x}) = \hat{\pi} \wedge \alpha(k) = \hat{s}] < \frac{1}{|\Pi|} \Pr[\alpha(k) = \hat{s}]\,.$$

Then for $k \xleftarrow{R} K$,

$$
\begin{aligned}
\Pr[\alpha(k) = \hat{s}] &= \sum_{\pi \in \Pi} \Pr[H_k(\hat{x}) = \pi \wedge \alpha(k) = \hat{s}] \\
&= \sum_{\pi \in \Pi \setminus \{\hat{\pi}\}} \Pr[H_k(\hat{x}) = \pi \wedge \alpha(k) = \hat{s}] + \Pr[H_k(\hat{x}) = \hat{\pi} \wedge \alpha(k) = \hat{s}] \\
&\leq \frac{|\Pi| - 1}{|\Pi|} \Pr[\alpha(k) = \hat{s}] + \Pr[H_k(\hat{x}) = \hat{\pi} \wedge \alpha(k) = \hat{s}] \\
&< \frac{|\Pi| - 1}{|\Pi|} \Pr[\alpha(k) = \hat{s}] + \frac{1}{|\Pi|} \Pr[\alpha(k) = \hat{s}] \\
&= \Pr[\alpha(k) = \hat{s}]
\end{aligned}
$$

which is a contradiction. This implies that if a projective hash family $\mathbf{H}$ is $1/|\Pi|$-universal, then it is strongly-universal. Conversely, a strongly uniform projective hash family is $1/|\Pi|$-universal, thus strongly-universal is equivalent to $1/|\Pi|$-universal. We will use the notation $1/|\Pi|$-universal throughout this work.

Definition 3.1.11 (Smooth Projective Hashing)**.** $\mathbf{H}$ *is* ϵ*-smooth if for* $k \xleftarrow{R} K$*,* $s \leftarrow \alpha(k)$*,* $x \xleftarrow{R} X \setminus L$*,* $\pi \leftarrow H_k(x)$ *and* $\pi' \xleftarrow{R} \Pi$*,*

$$\Delta[(x, s, \pi'), (x, s, \pi)] \leq \epsilon \,.$$

3.1.7 Hash Proof Systems

A hash proof system (HPS) $\mathbf{P}$ for a subset membership problem $\mathbf{M}$ associates a projective hash family $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ to each instance $\Lambda = (X, L, W, R)$ of $\mathbf{M}$. The HPS must also provide the following polynomial-time algorithms:

- $\mathsf{Private}_{\mathbf{H}}$ takes input (k, x) where $k \in K$ and $x \in X$ and returns $H_k(x)$.

- $\mathsf{Public}_{\mathbf{H}}$ takes input (s, x, w) where $s = \alpha(k)$ and $(x, w) \in R$, and returns $H_k(x)$.

Note that the output of Public is well defined, since $\alpha(k)$ determines the value of H_k on L, even though it may not determine the value of H_k on $X \setminus L$.

An extended hash proof system $\hat{\mathbf{P}}$ for a subset membership problem $\mathbf{M}$ is similar to an ordinary hash proof system, except that $\hat{\mathbf{P}}$ associates a set E as well as projective hash family $\hat{\mathbf{H}} = (\hat{H}, \hat{K}, X \times E, L \times E, \hat{\Pi}, \hat{S}, \hat{\alpha})$ to each instance $\Lambda = (X, L, W, R)$ of $\mathbf{M}$. The Public and $\mathsf{Private}$ evaluation algorithms take $e \in E$ as an additional input.

If the hash family $\mathbf{H}$ is clear from context, such as when we are working with only one hash family, we will drop the subscript from $\mathsf{Public}_{\mathbf{H}}$ and $\mathsf{Private}_{\mathbf{H}}$.

3.2 On approximating a projective hash family

Cramer and Shoup rightly state that the requirements on a projective hash family are so strong that many families cannot be efficiently implemented, but can be efficiently approximated. For this reason they include extra stages in their proofs, that go along the lines of:

"We replace the projective hash family $\mathbf{H}$ that $\mathbf{P}$ associates with Λ with its idealisation, which is an $\epsilon(\lambda)$-smooth projective hash family that is $\delta(\lambda)$-close to $\mathbf{H}$."

In this work, I have followed the example of Kurosawa and Desmedt [22], and chosen not to do this. Such approximations are statistically close, and seem only to cloud the exposition of the proofs. Suffice it to say that it would be straight forward to modify any proof in this work to include such steps

as needed, but I do not believe it would add add any value to the work, and merely obscure the real content of the proofs.

3.2.1 The Subset Witness Knowledge Assumption

Dent's proof [14] that the Cramer-Shoup scheme based on the Diffie-Hellman problem [10] is PA2 plaintext aware relies on an extractor assumption, known as the Diffie-Hellman Knowledge assumption. The DHK assumption was introduced in 1991 by Damgård [13]. To apply Dent's methodology in the setting of general subset membership problems, we present a generalised DHK assumption, which we call the Subset Witness Knowledge (SWK) assumption.

The subset witness knowledge experiment for an adversary $\mathcal{A}$ and a stateful extractor $\mathcal{A}^*$ works as follows:

$\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{A},\mathcal{A}^*}(1^\lambda)$
  $\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
  Run $\mathcal{A}^{\mathsf{SWK}}(\Lambda)$
    if $\mathcal{A}$ queries $\mathsf{SWK}(x)$ **then**
      $i \leftarrow i + 1$
      $x_i \leftarrow x$
      $(w_i, \mathsf{state}_{\mathcal{A}^*}) \leftarrow \mathcal{A}^*(\Lambda, x_i, R[\mathcal{A}], \mathsf{state}_{\mathcal{A}^*})$
      return w
  if $x_i \in L$ and $(x_i, w_i) \notin R$ for some i **then**
    return 1
  else
    return 0

$\mathcal{A}$ has access to an oracle SWK which takes an element $x \in X$ as input and returns $\mathcal{A}^*(\Lambda, x, R[\mathcal{A}])$. Note that the test $x \in L$ cannot be performed in polynomial time; this test is implicit in Damgård's statement of the definition of the DHK problem, but not written explicitly. Instead, he states that $\mathcal{A}^*$ returns a correct witness whenever $x \in L$, except perhaps with negligible probability. We prefer to include this test in the definition of the problem

itself, to allow our notation to be more consistent with other similar security experiments. We define

$$\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{A},\mathcal{A}^*}(\lambda) = \Pr[\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{A},\mathcal{A}^*}(\lambda) = 1]\,.$$

A subset membership problem $\mathbf{M}$ satisfies the SWK assumption if for every probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a stateful probabilistic polynomial-time extractor $\mathcal{A}^*$ such that $\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{A},\mathcal{A}^*}(\lambda)$ is negligible in λ. We note that if $x \notin L$ then the response of $\mathcal{A}^*$ may be arbitrary. This definition is unlike most definitions made in security proofs; the "win" condition is not simply guessing a value correctly. The adversary wins by submitting a value $x \in L$ to the SWK oracle for which the extractor fails to return a witness w such that $(x, w) \in R$.

Naor [24] classifies cryptographic assumptions according to how hard they would be to disprove. For example, he shows that the DDH assumption is "Efficiently Falsifiable", but the DHK assumption does not even satisfy the looser requirement of "Somewhat Falsifiable". The key difference is that to show that the DDH assumption is false, one would simply have to present a polynomial-time DDH adversary with non-negligible advantage, but to disprove the SWK assumption, one would have to show that there is an SWK adversary for which *no* extractor exists. This means that proving the SWK assumption is false is more like proving a conventional computational assumption is true, i.e. proving that there exists no efficient algorithm which solves a computational problem. However, the SWK assumption is a fairly natural generalisation of the DHK assumption, and we believe it is a reasonable trade off to show the plaintext awareness of a large class of schemes under

this assumption. On the other hand, there are subset membership problems which are believed to be true, but the corresponding subset witness knowledge assumption appears to be false.

3.3 On the Validity of the SWK Assumption

We have introduced a new family of assumptions in order to show the PA2 plaintext awareness of the Cramer-Shoup and Kurosawa-Desmedt families of encryption schemes. Cramer and Shoup [11] gave hash proof systems based on the DDH assumption, the Quadratic Residuosity assumption and the Decision Composite Residuosity assumption. We will discuss the corresponding SWK assumptions for these and other subset membership problems that have been used to instantiate hash proof systems.

As we have already mentioned, extractor assumptions of this type are even more difficult to justify than standard computational hardness assumptions, but nevertheless we will now examine the validity of SWK assumption for a variety of subset membership problems, and give our best guess as to whether they hold.

Parts of this section are derived from earlier work done jointly with Dent [5].

3.3.1 The DDH assumption

Let $\mathbb{G}$ be a cyclic group of prime order p, with generators g_0 and $g_1 = g_0^\alpha$ for some $\alpha \in \mathbb{Z}_p^*$. We define $X = \mathbb{G} \times \mathbb{G}$, $W = \mathbb{Z}_p$, $L = \{(g_0^w, g_1^w) : w \in \mathbb{Z}_p\}$, and let R be the relation $\{((g_0^w, g_1^w), w) : w \in \mathbb{Z}_p\}$. Then $\Lambda = (X, L, W, R)$ is an instance of the decision Diffie-Hellman problem.

The associated SWK assumption is precisely the Diffie-Hellman knowledge

assumption, also known as the knowledge of exponent assumption [13]. This is the same assumption that Dent used to prove the PA2 plaintext awareness of the Diffie-Hellman-based version of the Cramer-Shoup encryption scheme [14].

3.3.2 The DBDH assumption

Galindo *et al.* [17] give a construction of a hash proof system based on the decision bilinear Diffie-Hellman problem. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p with randomly chosen generators $g, g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be an efficiently computable, non-degenerate bilinear map. Let $X = \mathbb{G}_1 \times \mathbb{G}_T$, $L = \{(g^w, e(g_1, g_2)^w)\}$, $W = \mathbb{Z}_p$ and let R be the relation $\{((g^w, e(g_1, g_2)^w), w : w \in \mathbb{Z}_p)\}$. Then $\Lambda = (X, L, W, R)$ is an instance of the decision bilinear Diffie-Hellman assumption.

Distinguishing elements $(h, Z) \in L$ from $(h, Z) \in X \setminus L$ involves determining whether (g_1, g_2, h, Z) form a BDH-tuple. The associated SWK assumption, which we term the bilinear Diffie-Hellman knowledge (BDHK) assumption captures the idea that it is hard to compute tuples $(g_1, g_2, h, Z) \in L$ except by choosing w and computing $h = g^w$ and $e(g_1, g_2)^w$ directly. This assumption can be proven in the generic bilinear group model in much the same way that one can prove the DHK assumption in the generic group model.

3.3.3 The Quadratic Residuosity assumption

Let p and q be distinct primes, and let $N = pq$. Let $X = \mathbb{Z}_N^*$ and $L = \{w^2 : w \in \mathbb{Z}_N^*\}$. Let $W = \mathbb{Z}_N^*$ and let $R = \{(w^2, w) : w \in \mathbb{Z}_n^*\}$. Then $\Lambda = (X, L, W, R)$ is an instance of the quadratic residuosity problem. This problem is believed to be hard, but if factoring is hard, then the corresponding SWK assumption does not hold.

This is because a randomly selected element of $\mathbb{Z}_N^*$ is a square root with probability 1/4, or to put it another way $|L|/|X| = 1/4$. Let $\mathcal{A}$ be an SWK adversary which chooses a random element $x \in \mathbb{Z}_N^*$ and queries the SWK oracle on x. If the SWK assumption holds, then there is an extractor $\mathcal{A}^*$ which takes the random coins of $\mathcal{A}$ as input and returns a witness w such that $\Pr[x \in L \wedge w^2 \neq x]$ is negligible. In other words, $\mathcal{A}^*$ can compute square roots modulo N, which is known to be equivalent to factoring [28].

3.3.4 The DCR assumption

Let $N = pq$ where $p = 2p' + 1$ $q = 2q' + 1$ and p, q, p', q' are all primes. Let $X = \mathbb{Z}_{N^2}^*$ and $L = \{w^N : w \in \mathbb{Z}_{N^2}^*\}$. Let $W = \mathbb{Z}_{N^2}^*$, and let R be the relation $\{(w^N, w) : w \in \mathbb{Z}_{N^2}^*\}$. Then $\Lambda = (X, L, W, R)$ is an instance of the decision composite residuosity (DCR) assumption. The decisional composite residuosity assumption was introduced by Paillier [27]. The corresponding SWK assumption, which we term the decisional composite residuosity knowledge (DCRK) assumption, is that it is hard to compute a value $x \in L$ except by choosing w and computing $x = w^N$. Unlike the QR assumption, there is no difficulty in building an extractor for an adversary which simply outputs an element $x \xleftarrow{R} X$ because the probability that $x \in L$ is $1/N$ which is negligible. For this reason, we do not dismiss the possibility that the DCRK assumption holds.

3.3.5 The GBD assumption

González Nieto *et al.* [26] introduced a subgroup membership problem which we will call the GBD assumption. A modified form of this assumption was later used by Brown *et al.* [7] to construct a hash proof membership problem

and instantiate the Cramer-Shoup and Kurosawa-Desmedt schemes. It is this modified form that we present here.

Let $p = 2N + 1$, where $N = q_0 q_1$ and p, q_0 and q_1 are all primes. Let X be the subgroup of $\mathbb{Z}_p^*$ order N and let L be the subgroup of X of order q_0. Let g_0 be a generator of L, let $W = \{0, \ldots, N-1\}$ and let R be the relation $\{(g_0^w, w) : w \in W\}$. Then $\Lambda = (X, L, W, R)$ is an instance of the GBD problem.

The associated SWK assumption, which we will call the GBDK assumption, essentially states that the only way to choose an element $x \in L$ is to compute some power of g_0. If an adversary knew the factorisation of N then it could simply take a random element $y \in \mathbb{Z}_p^*$ and compute $x = y^{2q_1}$, but q_1 is not part of the instance description. If the discrete logarithm problem in L is easy, then then there is an extractor for any GBDK adversary. On the other hand, if the discrete logarithm problem in L is hard but factoring is easy, then it seems unlikely that the GBDK assumption would hold: one could construct an adversary which factors N and computes an element of L as described above. Finally, if both factoring and the discrete logarithm problem in L are hard, as we believe they are, then it seems reasonable that the GBDK assumption holds.

3.4 The Integers Modulo N are Simulatable

All of the above subset membership problems involve either abstract groups or the integers mod N for some N. We will show that the integers mod N are simulatable.

Dent claimed [14] that for $p = 2q + 1$ where p and q are prime then the

subgroup of $\mathbb{Z}_p^*$ of order q is simulatable, and presents a simulator, but he does not give a proof that the simulator's output is indistinguishable from a random element of the group. Although we believe it would be possible to fill in the details of Dent's proof, our proof uses an alternative methodology.

Theorem 3.4.1. *Let* $\mathbf{S} = ((\mathbb{Z}_N^*)_{N \in I_\lambda})_{\lambda \in \mathbb{N}}$*, and suppose that there is a constant* c *such that* $I_\lambda \subseteq \{1, \ldots, 2^{\lambda+c}\}$ *for all* $\lambda \in \mathbb{N}$*. Then* $\mathbf{S}$ *is statistically simulatable.*

Proof. Let $n_{\mathbf{S}}(\lambda) = 2\lambda + c$ for all $\lambda \in \mathbb{N}$. This means that $n_{\mathbf{S}}(\lambda)$ is greater than the bit length of the largest integer in I_λ by at least λ. Let $B = 2^{2\lambda+c}$, $q_{max} = \lfloor \frac{B}{N} \rfloor$ and let $R = B - Nq_{max}$

In the following algorithms, we interpret the bitstring $r \in \{0,1\}^{n_{\mathbf{S}}(\lambda)}$ as an integer.

function $f_{\mathbf{S}}(1^\lambda, N, r)$
  $x \leftarrow r \mod N$
  return x

function $f_{\mathbf{S}}(1^\lambda, N, x)$
  $q \xleftarrow{R} \{0, \ldots, q_{max} - 1\}$
  $r \leftarrow Nq + x$
  return r

Firstly, we must ensure that $\mathbf{f} = (f_{\mathbf{S}}, f_{\mathbf{S}}^{-1}, n_{\mathbf{S}})$ is a simulator for $\mathbf{S}$. Specifically,

- $f_{\mathbf{S}}$ is a deterministic algorithm which takes as input a security parameter 1^λ, an index $N \in I_\lambda$, and $r \in \{0,1\}^{n_{\mathbf{S}}(\lambda)}$, and returns an element $x \in \mathbb{Z}_N^*$.
- $f_{\mathbf{S}}^{-1}$ is a probabilistic algorithm which takes as input a security parameter 1^λ, an index $N \in I_\lambda$, and $x \in \mathbb{Z}_N^*$, and returns a string $r \in \{0,1\}^{n_{\mathbf{S}}(\lambda)}$.
- For all $\lambda \in \mathbb{N}$, $N \in I_\lambda$, $x \in \mathbb{Z}_N^*$, $f_{\mathbf{S}}(1^\lambda, N, f_{\mathbf{S}}^{-1}(1^\lambda, N, x)) = x$.

These properties follow by definition of $\mathbf{f}$.

We now show that the statistical Set-Sim property of $\mathbf{f}$ holds. Let $x \xleftarrow{R} \mathbb{Z}_N^*$, $r \xleftarrow{R} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$ and let $y \leftarrow f_{\mathbf{S}}(1^\lambda, N, r) = r \mod N$. Then for all $0 \leq s < R$, $\Pr[y = s] = \frac{q_{max}+1}{B}$ and for all $R \leq s < N$, $\Pr[y = s] = \frac{q_{max}}{B}$.

$$\begin{aligned} \left| \frac{q_{max}+1}{B} - \frac{1}{N} \right| &\leq \frac{N + Nq_{max} - B}{BN} = \frac{N-R}{BN} \\ &\leq \frac{N}{BN} = \frac{1}{B} \end{aligned}$$

Similarly

$$\begin{aligned} \left| \frac{q_{max}}{B} - \frac{1}{N} \right| &\leq \left| \frac{Nq_{max} - B}{BN} \right| = \frac{R}{BN} \\ &\leq \frac{N}{BN} = \frac{1}{B} \end{aligned}$$

Therefore

$$\Delta[(1^\lambda, N, x), (1^\lambda, N, y)] \leq \frac{1}{2} \sum_{s \in \mathbb{Z}_N^*} |\Pr[x = s] - \Pr[y = s]| < \frac{N}{B} \leq \frac{2^{c+\lambda}}{2^{c+2\lambda}} = \frac{1}{2^\lambda}$$

which is negligible in λ as desired.

Now let us consider the statistical Rand-Sim property. Let $r \xleftarrow{R} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$, and let $r' \leftarrow f_{\mathbf{S}}^{-1}(1^\lambda, i, x)$. Fix $s \in \{0,1\}^{n_{\mathbf{S}}(\lambda)}$ and let $x = s \mod N$. We consider three cases:

- Case 1: $0 \leq s < Nq_{max} - 1$ and $0 \leq x < R$

In this case, $\Pr[r = s \mod N] = \frac{q_{max}+1}{B}$. $\Pr[r' = s | r = s \mod N] = \frac{1}{q_{max}}$. Therefore $\Pr[r' = s] = \frac{q_{max}+1}{Bq_{max}}$, and

$$|\Pr[r' = s] - \Pr[r = s]| = \left| \frac{q_{max}+1}{Bq_{max}} - \frac{1}{B} \right| = \frac{1}{Bq_{max}}.$$

- Case 2: $0 \leq s \leq Nq_{max} - 1$ and $R \leq x \leq N - 1$

In this case, $\Pr[r = s \mod N] = \frac{q_{max}}{B}$. $\Pr[r' = s | r = s \mod N] = \frac{1}{q_{max}}$. Therefore $\Pr[r' = s] = \frac{1}{B}$ and

$$|\Pr[r' = s] - \Pr[r = s]| = 0\,.$$

- Case 3: $s \geq Nq_{max}$

In this case $\Pr[r' = s] = 0$ since $f_{\mathbf{S}}^{-1}$ never returns a value greater than $Nq_{max} - 1$. Thus

$$|\Pr[r' = s] - \Pr[r = s]| = \frac{1}{B}\,.$$

Putting this together, we see that the statistical distance

$$\begin{aligned}
\Delta[(1^\lambda, i, r), (1^\lambda, i, r')] &\leq \frac{1}{2}\sum_{s \in S} |\Pr[x = s] - \Pr[y = s]| \\
&\leq \frac{1}{2}\left(Nq_{max}\frac{1}{Bq_{max}} + R\frac{1}{B}\right) \\
&= \frac{1}{2}\frac{N + R}{B} \leq \frac{2N}{B} \\
&\leq \frac{1}{2}\frac{2 \cdot 2^{c+\lambda}}{2^{c+2\lambda}} = \frac{1}{2^\lambda}
\end{aligned}$$

which is negligible in λ as desired. Therefore the family of sets $\mathbf{S}$ is simulatable.

□

Chapter 4

The Generalised Cramer–Shoup Encryption Scheme is PA2 Plaintext Aware

In this chapter, we will use the methodology of Dent [14] which we introduced in the previous chapter to prove that a version of the Cramer-Shoup encryption scheme is PA2 plaintext aware under the Subset Witness Knowledge assumption which we introduced in Section 3.2.1.

4.1 The Generalised Cramer-Shoup Encryption Scheme

Let $\mathbf{M}$ be a subset membership problem and let $\Lambda = (X, L, W, R) \leftarrow \mathsf{ISA}(1^\lambda)$ be a randomly chosen instance of $\mathbf{M}$. Let $\mathbf{P}$ be a hash proof system for $\mathbf{M}$, and let $\hat{\mathbf{P}}$ be an extended hash proof system for $\mathbf{M}$. Let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ and $\hat{\mathbf{H}} = (H, \hat{K}, X \times \Pi, L \times \Pi, \hat{\Pi}, \hat{S}, \hat{\alpha})$ for $(X \times \Pi, L \times \Pi)$ be the projective

hash family that $\mathbf{P}$ and $\hat{\mathbf{P}}$ associate with Λ respectively. We also require that Π is an abelian group with respect to some efficiently computable group operation which we will write additively. The Cramer-Shoup universal hash proof encryption scheme is then defined as follows:

function $\mathsf{KeyGen}(\Lambda)$
  $k \xleftarrow{R} K$
  $\hat{k} \xleftarrow{R} \hat{K}$
  $s \leftarrow \alpha(k)$
  $\hat{s} \leftarrow \hat{\alpha}(\hat{k})$
  $pk \leftarrow (s, \hat{s})$
  $sk \leftarrow (k, \hat{k})$
  return (pk, sk)

function $\mathsf{Encrypt}(pk, m)$
  Parse pk as $(s, \hat{s})$
  $(x, w) \leftarrow \mathsf{SSA}(\Lambda)$
  $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x, w)$
  $e \leftarrow m + \pi$
  $\hat{\pi} \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x, e, w)$
  $C \leftarrow (x, e, \hat{\pi})$
  return C

function $\mathsf{Decrypt}(sk, C)$
  Parse sk as $(k, \hat{k})$
  Parse C as $(x, e, \hat{\pi})$
  $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
  if $\hat{\pi}' = \hat{\pi}$ **then**
    $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
    $m \leftarrow e - \pi$
    return m
  else
    return $\perp$

4.2 Cramer-Shoup is PA1+

Theorem 4.2.1. *Suppose that $\hat{\mathbf{H}}$ is $\epsilon_{\hat{\mathbf{H}}}$-universal, that the public keys returned by $\mathsf{KeyGen}(\Lambda)$ are uniformly distributed on $S \times \hat{S}$, and that the family of sets $\mathbf{S} = ((S_\Lambda \times \hat{S}_\Lambda)_{\Lambda \in I_\lambda})_{\lambda \in \mathbb{N}}$ of public keys is statistically simulatable, with simulator $\mathbf{f_S} = (f_{\mathbf{S}}, f_{\mathbf{S}}^{-1}, n_{\mathbf{S}})$. Assume also that the SWK assumption holds for the underlying subset membership problem $\mathbf{M}$. Then the Cramer-Shoup encryption scheme based on the hash proof systems $\mathbf{H}$ and $\hat{\mathbf{H}}$ is PA1+.*

Proof. Let $\mathcal{A}$ be a PA1+ ciphertext creator which makes at most q_r randomness queries. We will construct a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ by constructing an SWK adversary $\mathcal{B}$ and using the existence of an SWK extractor $\mathcal{B}^*$ to construct a plaintext extractor $\mathcal{A}^*$. In the following, we abuse notation slightly by treating Rlist as either a list or a string. However, the appropriate meaning should be clear from the context.

$\mathcal{B}$ takes input $\Lambda = (X, L, W, R)$ and works as follows:

function $\mathcal{B}(\Lambda)$
  $r_s \xleftarrow{R} \{0,1\}^{n_S(\lambda)}$
  $r_{\hat{s}} \xleftarrow{R} \{0,1\}^{n_{\hat{S}}(\lambda)}$
  $\mathsf{Rlist} \xleftarrow{R} \{0,1\}^{q_r}$
  $s \leftarrow f_{\mathbf{S}}(r_s)$
  $\hat{s} \leftarrow f_{\hat{\mathbf{S}}}(r_{\hat{s}})$
  $pk \leftarrow (\Lambda, s, \hat{s})$
  Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(1^\lambda, pk)$
    if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
      $n_r \leftarrow n_r + 1$
      return $\mathsf{Rlist}[n_r]$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      Parse C as $(x, e, \hat{\pi})$
      $w \leftarrow$ Query $\mathsf{SWK}(x)$
      if $(x, w) \notin R$ **then**
        return $\perp$
      else
        $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x, e, w)$
        if $\hat{\pi}' = \hat{\pi}$ **then**
          $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(w, x, w)$
          $m \leftarrow e - \pi$
          return m
        else
          return $\perp$

By the SWK assumption, there is a witness extractor $\mathcal{B}^*$ such that the advantage $\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}$ is negligible. We use $\mathcal{B}^*$ to construct a PA1+ plaintext extractor $\mathcal{A}^*$ in much the same way as we did for Kurosawa-Desmedt:

function $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{state}_{\mathcal{A}^*})$
  if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
    Parse pk as (Λ, s)
    $r_s \leftarrow f_{\mathbf{S}}^{-1}(s)$
    $r_{\hat{s}} \leftarrow f_{\hat{\mathbf{S}}}^{-1}(\hat{s})$
  else
    Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(r_s, r_{\hat{s}}, x_1, \ldots, x_{n_d-1})$
  Parse C as $(x_{n_d}, e, \hat{\pi})$
  $n_r \leftarrow |\mathsf{Rlist}|$
  $\mathsf{Rlist}' \xleftarrow{\mathrm{R}} \{0,1\}^{q_r - n_r}$
  $R_{\mathcal{B}} \leftarrow r_s || r_{\hat{s}} || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]$
  $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
  for $t = 1$ to n_d **do**
    $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
  if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
    $m \leftarrow \perp$
  else
    $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w_{n_d,n_d})$
    if $\hat{\pi}' \neq \hat{\pi}$ **then**
      $m \leftarrow \perp$
    else
      $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w)$
      $m \leftarrow e - \pi$
  $\mathsf{state}_{\mathcal{A}^*} \leftarrow (r_s, r_{\hat{s}}, x_1, \ldots, x_{n_d})$
  return $(m, \mathsf{state}_{\mathcal{A}^*})$

To construct the SWK adversary $\mathcal{B}$, we had to supply the complete Rlist to $\mathcal{B}$ in advance. Unfortunately, $\mathcal{A}^*$ does not know the random bits Rlist' when it ran on the j^{th} decryption query, so instead it generates new random bits for itself to simulate the environment of $\mathcal{B}^*$ in the SWK-Fake experiment. If $\mathcal{A}^*$ preserved the state variable of $\mathcal{B}^*$ between decryption queries, $\mathcal{B}^*$ could detect that Rlist' had changed next time it is called. To prevent this, $\mathcal{A}^*$ re-initialises $\mathcal{B}^*$ with an empty state variable, and runs $\mathcal{B}^*$ again on each of the previous inputs. If there is more than one witness for a particular $x_t \in L$, then $\mathcal{B}^*$ may generate a different witness each time, but as long as $(x_t, w_{j,t}) \in R$, then $\mathsf{Public}(s, x_t, w_{j,t}) = H_k(x_j)$ for all $1 \leq j \leq t$, so the view of $\mathcal{A}$ is independent

of the witness returned by $\mathcal{B}^*$.

We now proceed by a sequence of games to show that $\mathbf{Adv}^{\mathrm{PA1+}}_{\Pi,\mathcal{A},\mathcal{A}^*,D}$ is negligible. Let T_i be the event that $D(aux)$ returns 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\mathrm{PA1+Fake}}_{\mathbf{CS},\mathcal{A},\mathcal{A}^*,D}(\lambda)$. Written out in full, it is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\mathrm{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
      Parse pk as $(s, \hat{s})$
      $r_s \leftarrow f^{-1}_{\mathbf{S}}(s)$
      $r_{\hat{s}} \leftarrow f^{-1}_{\hat{\mathbf{S}}}(\hat{s})$
    else
      Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(r_s, r_{\hat{s}}, x_1, \ldots, x_{n_d-1})$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    $\mathsf{Rlist}' \xleftarrow{\mathrm{R}} \{0,1\}^{q_r - n_r}$
    $R_{\mathcal{B}} \leftarrow r_s || r_{\hat{s}} || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    for $t = 1$ to n_d **do**
      $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
    if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
      $m_{n_d} \leftarrow \perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
      if $\hat{\pi}' \neq \hat{\pi}$ **then**
        $m_{n_d} \leftarrow \perp$
      else
        $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
        $m_{n_d} \leftarrow e - \pi$
    $\mathsf{state}_{\mathcal{A}^*} \leftarrow (r_s, r_{\hat{s}}, x_1, \ldots, x_{n_d})$
    return m_{n_d}

$b \leftarrow D(aux)$
return b

Game 1: This is a bridging step. Game 1 is the same as Game 0 except that we generate the public key at random, without the corresponding secret key. We also remove the $\mathsf{state}^*_{\mathcal{A}}$ variable and move the generation of r_s and $r_{\hat{s}}$ to the beginning of the game.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{s \xleftarrow{R} S}$
$\boxed{\hat{s} \xleftarrow{R} \hat{S}}$
$\boxed{pk \leftarrow (s, \hat{s})}$
$\boxed{r_s \leftarrow f_{\mathbf{S}}^{-1}(s)}$
$\boxed{r_{\hat{s}} \leftarrow f_{\hat{\mathbf{S}}}^{-1}(\hat{s})}$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    $\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$
    $R_{\mathcal{B}} \leftarrow r_s || r_{\hat{s}} || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    for $t = 1$ to n_d **do**
      $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
    if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
      $m_{n_d} \leftarrow \perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
      if $\hat{\pi}' \neq \hat{\pi}$ **then**
        $m_{n_d} \leftarrow \perp$
      else
        $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
        $m_{n_d} \leftarrow e - \pi$
    return m_{n_d}

$b \leftarrow D(aux)$
return b

By the assumption that the public key $pk = (s, \hat{s})$ is uniformly distributed on $S \times \hat{S}$, we see that the inputs to all algorithms are distributed exactly as in Game 0, so:

$$\Pr[T_1] = \Pr[T_0]\,.$$

Game 2: We simulate pk using the simulators $f_{\mathbf{S}}(r'_s)$ and $f_{\hat{\mathbf{S}}}(r'_{\hat{s}})$ for random strings r'_s, $r'_{\hat{s}}$. Note we continue to generate the randomness r_s and $r_{\hat{s}}$ which get passed to $\mathcal{B}^*$ using $f_{\mathbf{S}}^{-1}(s)$ and $f_{\hat{\mathbf{S}}}^{-1}(\hat{s})$ just as we did before.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{r'_s \xleftarrow{\mathrm{R}} \{0,1\}^{n_{\mathbf{S}}(\lambda)}}$
$\boxed{r'_{\hat{s}} \xleftarrow{\mathrm{R}} \{0,1\}^{n_{\hat{\mathbf{S}}}(\lambda)}}$
$\boxed{s \leftarrow f_{\hat{\mathbf{S}}}(r'_s)}$
$\boxed{\hat{s} \leftarrow f_{\hat{\mathbf{S}}}(r'_{\hat{s}})}$
$pk \leftarrow (s, \hat{s})$
$r_s \leftarrow f_{\mathbf{S}}^{-1}(s)$
$r_{\hat{s}} \leftarrow f_{\hat{\mathbf{S}}}^{-1}(\hat{s})$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\mathrm{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    $\mathsf{Rlist}' \xleftarrow{\mathrm{R}} \{0,1\}^{q_r - n_r}$
    $R_{\mathcal{B}} \leftarrow r_s || r_{\hat{s}} || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    for $t = 1$ to n_d **do**
      $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
    if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**

$m_{n_d} \leftarrow \perp$
else
$\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
if $\hat{\pi}' \neq \hat{\pi}$ **then**
$m_{n_d} \leftarrow \perp$
else
$\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
$m_{n_d} \leftarrow e - \pi$
return m_{n_d}
$b \leftarrow D(aux)$
return b

By the statistical Set-Sim property of $\mathbf{S}$, $\epsilon_s = \Delta[(1^\lambda, \Lambda, s), (1^\lambda, \Lambda, s')]$ is negligible. Similarly, $\epsilon_{\hat{s}} = \Delta[(1^\lambda, \Lambda, \hat{s}), (1^\Lambda, \Lambda, \hat{s}')]$ is negligible. Thus

$$|\Pr[T_2] - \Pr[T_1]| \leq \epsilon_{\mathbf{S}} + \epsilon_{\hat{\mathbf{S}}}$$

by Lemma 1.2.1 and a hybrid argument.

Game 3: We provide the randomness r'_s and $r'_{\hat{s}}$ that we used to simulate pk to $\mathcal{B}^*$, instead of using $f_{\mathbf{S}}^{-1}$ and to generate a second random string. Written out in full, the game is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r'_s \xleftarrow{R} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$
$r'_{\hat{s}} \xleftarrow{R} \{0,1\}^{n_{\hat{\mathbf{S}}}(\lambda)}$
$s \leftarrow f_{\hat{\mathbf{S}}}(r'_s)$
$\hat{s} \leftarrow f_{\hat{\mathbf{S}}}(r'_{\hat{s}})$
$pk \leftarrow (s, \hat{s})$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$n_r \leftarrow n_r + 1$
$b \xleftarrow{R} \{0,1\}$
Append b to Rlist
return $\mathsf{Rlist}[n_r]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Parse C as $(x_{n_d}, e, \hat{\pi})$
$\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$

$\boxed{R_{\mathcal{B}} \leftarrow r'_s||r'_{\hat{s}}||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]}$
$\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
for $t = 1$ to n_d **do**
$(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
$m_{n_d} \leftarrow \perp$
else
$\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
if $\hat{\pi}' \neq \hat{\pi}$ **then**
$m_{n_d} \leftarrow \perp$
else
$\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
$m_{n_d} \leftarrow e - \pi$
return m_{n_d}
$b \leftarrow D(aux)$
return b

By the statistical Rand-Sim property of $\mathbf{S}$, $\epsilon_{r_s} = \Delta[(1^\lambda, \Lambda, r_s), (1^\lambda, \Lambda, r'_s)]$ is negligible as a function of λ, and $\epsilon_{r_{\hat{s}}} = \Delta[(1^\lambda, \Lambda, r_{\hat{s}}), (1^\lambda, \Lambda, r'_{\hat{s}})]$ is negligible by the statistical Rand-Sim property of $\hat{\mathbf{S}}$. We may regard $\mathcal{A}$ together with the $\mathsf{Randomness}$ and $\mathsf{Decrypt}$ oracles as a single algorithm which takes either $(r_s, r_{\hat{s}})$ or $(r'_s, r'_{\hat{s}})$ as one of its inputs, so by Lemma 1.2.1 and using a simple hybrid argument, this implies that

$$|\Pr[T_3] - \Pr[T_2]| \leq \epsilon_r + \epsilon_{r_{\hat{s}}}\,.$$

Game 4: We replace $\mathcal{B}^*$ with an idealised algorithm that always returns a valid witness w if one exists. Since w is only used to calculate $\pi \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$, and π does not depend on the choice of witness by the definition of a hash proof system, if there are multiple witnesses it does not matter which one is selected. Written in full the game is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r'_s \xleftarrow{\mathrm{R}} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$

$r'_{\hat{s}} \xleftarrow{R} \{0,1\}^{n_{\hat{\mathbf{S}}}(\lambda)}$
$s \leftarrow f_{\hat{\mathbf{S}}}(r'_s)$
$\hat{s} \leftarrow f_{\hat{\mathbf{S}}}(r'_{\hat{s}})$
$pk \leftarrow (s, \hat{s})$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    if $\boxed{\text{there exists } w \text{ such that } (x_{n_d}, w) \in R}$ **then**
      $\boxed{w_{n_d} \leftarrow w}$
    else
      $\boxed{\textbf{return } \perp}$
    $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
    if $\hat{\pi}' \neq \hat{\pi}$ **then**
      $m_{n_d} \leftarrow \perp$
    else
      $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
      $m_{n_d} \leftarrow e - \pi$
    return m_{n_d}
$b \leftarrow D(aux)$
return b

We do not require this computation to be performed in polynomial time. Let F_3 be the event that $\mathcal{B}^*$ returns an invalid witness during one of the decryption queries in Game 3. If F_3 does not occur then each ciphertext is decrypted correctly and Game 3 proceeds identically to Game 4. We will show that $\Pr[F_3]$ is negligible via a sequence of games, Game $(3,0), \ldots,$ Game $(3, q_d)$, where Game $(3,0)$ = Game 3 and Game $(3, q_d)$ = Game 4.

Game $(3, i)$:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r'_s \xleftarrow{R} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$

$r'_{\hat{s}} \xleftarrow{R} \{0,1\}^{n_{\hat{\mathbf{S}}}(\lambda)}$
$s \leftarrow f_{\hat{\mathbf{S}}}(r'_s)$
$\hat{s} \leftarrow f_{\hat{\mathbf{S}}}(r'_{\hat{s}})$
$pk \leftarrow (s, \hat{s})$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    if $n_d \leq i$ **then**
      if there exists w such that $(x_{n_d}, w) \in R$ **then**
        $w_{n_d} \leftarrow w$
      else
        return $\perp$
    else
      $\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$
      $\boxed{R_{\mathcal{B}} \leftarrow r'_s || r'_{\hat{s}} || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]}$
      $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
      for $t = 1$ to n_d **do**
        $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
      if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
        $m_{n_d} \leftarrow \perp$
      else
        $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
        if $\hat{\pi}' \neq \hat{\pi}$ **then**
          $m_{n_d} \leftarrow \perp$
        else
          $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
          $m_{n_d} \leftarrow e - \pi$
      return $(m_{n_d}, \mathsf{state}_{\mathcal{A}^*})$
$b \leftarrow D(aux)$
return b

Let $F_{3,i}$ be the event that $\mathcal{B}^*$ returns an incorrect witness when called in response to the i^{th} decryption query, i.e. $x_i \in L$ but $(x_i, w_{i,i}) \notin R$. If $F_{3,i}$ does not occur then Game $(3, i)$ proceeds identically to Game $(3, i-1)$.

We would like to show that each time $\mathcal{B}^*$ is called in Game $(3, i-1)$, the inputs to $\mathcal{B}^*$, namely $\Lambda, x_t, R_{\mathcal{B}}$ and $\mathsf{state}_{\mathcal{B}^*}$, are distributed as in $\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)$. The instance Λ is correctly sampled using ISA and each of the bits of $R_{\mathcal{B}} = r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$ are distributed uniformly at random (although the random tape used in subsequent queries depends on this random tape, an issue we will return to below).

Consider running $\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)$, using the instance Λ and with $\mathcal{B}$'s random tape set to $r||\mathsf{Rlist}||\hat{\mathsf{Rlist}}'||R[\mathcal{A}]$, where $\Lambda, r, \mathsf{Rlist}$ and $R[\mathcal{A}]$ are as above, and $\hat{\mathsf{Rlist}}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$ is independent of Rlist'. To avoid ambiguity, we will use $\hat{x}_i$ to distinguish variables in the SWK experiment from their counterparts in Game $(3, i-1)$. In the SWK experiment, the output of $\mathcal{B}^*$ at any stage may affect the next query made by $\mathcal{A}$. In contrast, $\mathcal{A}$'s first $i-1$ decryption queries in Game $(3, i-1)$ are answered without using $\mathcal{B}^*$. $\mathcal{B}^*$ is invoked for the first time when answering the i^{th} query, and list of elements $x_1, \ldots, x_{i-1}$ obtained previous decryption queries are successively used as inputs to $\mathcal{B}^*$. Despite this, we will now show by induction that $x_1 = \hat{x}_t$ for $1 \leq t \leq i-1$.

By definition of $\mathcal{B}$ we see that $\mathcal{B}$ computes a public key $pk \leftarrow f_{\mathbf{S}}(r)$, and runs $\mathcal{A}(pk; R[\mathcal{A}])$. Let $1 \leq t \leq i$, and suppose by induction that for all $1 \leq t' \leq t$, $\hat{x}_{t'} = x_{t'}$, and $(\hat{x}_{t'}, \hat{w}_{t'}) \in R)$ if $\hat{x}_{t'} \in L$. $\hat{w}_{t'}$ may be different to $w_{i,t'}$, but these values are never passed on to $\mathcal{A}$. Since $\mathsf{Public}(s, \hat{x}_{t'}, \hat{w}_{t'}) = H_k(\hat{x}_{t'})$ for all valid witnesses, it follows that the decrypted messages returned to $\mathcal{A}$ are identical in the two games. The values of Rlist are identical in the two experiments, and $\mathcal{B}$ does not pass any of $\hat{\mathsf{Rlist}}'$ to $\mathcal{A}$ until after the i^{th} decryption query is made the responses to $\mathcal{A}$'s randomness queries are also identical, because $\mathcal{B}$ does not use any part of $\hat{\mathsf{Rlist}}'$ to $\mathcal{A}$ until after the i^{th} decryption query is made.

This shows that the inputs to $\mathcal{A}$ in the SWK experiment, including the randomness, are identical to their values in Game $(3, i-1)$. It follows that $\hat{x}_t = x_t$, since ciphertexts that $\mathcal{A}$ passes to its $\mathsf{Decrypt}$ oracle are functions of $\mathcal{A}$'s randomness, it's inputs and the responses to previous queries, which are all held fixed. By induction, this implies that all inputs to $\mathcal{B}^*$ have the same distribution in the i^{th} decryption query of Game $(3, i-1)$ as they do in the SWK experiment, so

$$\Pr[F_{3,i}] = \Pr[x_t \in L \wedge (x_t, w_{i,t}) \notin R] \leq \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*} .$$

We must now address the dependence of the events F_j. Consider the event $G_j = \bigcup_{i=1}^{j} F_{3,j}$. Then $G_1 = F_1$ and for all $1 \leq j \leq q_d$,

$$\Pr[G_{j+1}] = \Pr[G_j \cup F_{3,j+1}] = \Pr[G_j] + \Pr[F_{3,j+1}] - \Pr[G_j \cap F_{3,j+1}] \leq \Pr[G_j] + \Pr[F_{3,j+1}] .$$

Since the events $F_{3,j+1}$ and G_j are not independent, we do not know $\Pr[G_j \cap F_{j+1}]$, but we do not need to know it, since we only need an upper bound on the probability of G_{j+1}. By induction, this shows that

$$\begin{aligned}
\Pr[F_3] &= \Pr[G_{q_d}] \\
&\leq \sum_{j=1}^{q_d} \Pr[F_{3,j+1}] \\
&\leq \sum_{j=1}^{q_d} \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*} \\
&\leq q_d \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*} .
\end{aligned}$$

If F_3 does not occur, then $\mathcal{B}^*$ correctly computes a witness every time it is called in Game 3, and so Game 4 is identical to Game 3. Thus

$$|\Pr[T_4] - \Pr[T_3]| \leq \Pr[F_3] \leq q_d \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}\,.$$

Game 5: We choose the public key at random.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{s \xleftarrow{\mathrm{R}} S}$
$\boxed{\hat{s} \xleftarrow{\mathrm{R}} \hat{S}}$
$pk \leftarrow (s, \hat{s})$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\mathrm{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    if there exists w such that $(x_{n_d}, w) \in R$ **then**
      $w_{n_d} \leftarrow w$
    else
      return $\perp$
    $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
    if $\hat{\pi}' \neq \hat{\pi}$ **then**
      $m_{n_d} \leftarrow \perp$
    else
      $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
      $m_{n_d} \leftarrow e - \pi$
    return $(m_{n_d}, \mathsf{state}_{\mathcal{A}^*})$
$b \leftarrow D(aux)$
return b

By the statistical Set-Sim property of $\mathbf{S}$, $\epsilon_s = \Delta[(1^\lambda, \Lambda, s), (1^\lambda, \Lambda, s')]$ is negligible. Similarly, $\epsilon_{\hat{s}} = \Delta[(1^\lambda, \Lambda, \hat{s}), (1^\Lambda, \Lambda, \hat{s}')]$ is negligible. Thus

$$|\Pr[T_5] - \Pr[T_4]| \leq \epsilon_{\mathbf{S}} + \epsilon_{\hat{\mathbf{S}}}$$

by Lemma 1.2.1 and a hybrid argument. Note that because we are relying on a

perfect witness extractor, we cannot assume the algorithm is polynomial-time, so computational Set-Sim would not suffice for this transition.

Game 6: We compute the public key using KeyGen once more:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)}$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as $(x_{n_d}, e, \hat{\pi})$
    if there exists w such that $(x_{n_d}, w) \in R$ **then**
      $w_{n_d} \leftarrow w$
    else
      return $\perp$
    $\hat{\pi}' \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_{n_d}, e, w)$
    if $\hat{\pi}' \neq \hat{\pi}$ **then**
      $m_{n_d} \leftarrow \perp$
    else
      $\pi \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x_{n_d}, w_{n_d,n_d})$
      $m_{n_d} \leftarrow e - \pi$
    return m_{n_d}
$b \leftarrow D(aux)$
return b

By the assumption that the public key $pk = (s, \hat{s})$ is uniformly distributed on $S \times \hat{S}$, we see that the inputs to all algorithms are distributed exactly as in Game 5, so $\Pr[T_6] = \Pr[T_5]$

Game 7: We use $\mathsf{Private}$ instead of Public to answer decryption queries:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**

$n_r \leftarrow n_r + 1$
$b \xleftarrow{R} \{0,1\}$
Append b to Rlist
return $\mathsf{Rlist}[n_r]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Parse C as $(x_{n_d}, e, \hat{\pi})$
if there exists w such that $(x_{n_d}, w) \in R$ **then**
$w_{n_d} \leftarrow w$
else
return $\perp$
Parse sk as $(k, \hat{k})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x_{n_d}, e)$
if $\hat{\pi}' \neq \hat{\pi}$ **then**
$m_{n_d} \leftarrow \perp$
else
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x_{n_d})$
$m_{n_d} \leftarrow e - \pi$
return m_{n_d}
$b \leftarrow D(aux)$
return b

If $x_j \in L$, then there exists $w_j \in W$ such that $(x_j, w_j) \in R$ by definition, and this w_j is used in Game 7 to answer decryption queries. In particular this means that $\mathsf{Public}_{\mathbf{P}}(s, x_j, w_j) = H_k(x_j) = \mathsf{Private}_{\mathbf{P}}(k, x_j)$, and $\mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x_j, e, w_j) = \hat{H}_{\hat{k}}(x_j, e) = \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x_j, e)$, so the value returned by the decryption oracle is unchanged in Game 7.

If $x_j \notin L$, then the decrypt oracle will return $\perp$ in Game 6. However, in Game 7, $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x_{n_d}, e)$. By the $\epsilon_{\hat{\mathbf{H}}}$-universal property of $\hat{\mathbf{H}}$, it follows that $\Pr[\hat{\pi}' = \hat{\pi}] \leq \epsilon$. Since the decrypt oracle rejects whenever $\hat{\pi}' \neq \hat{\pi}$, it follows that it rejects with probability at least $1 - \epsilon$. Since there are at most q_d decryption queries, it follows that

$$|\Pr[T_7] - \Pr[T_6]| \leq q_d \epsilon_{\hat{\mathbf{H}}} .$$

Finally, we note that Game 7 is exactly $\mathbf{Expt}^{\mathrm{PA1+Real}}_{\mathbf{CS},\mathcal{A},D}(\lambda)$. Thus:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{PA1+}}_{\mathbf{CS},\mathcal{A},\mathcal{A}^*}(\lambda) &= |\Pr[T_7] - \Pr[T_0]| \\ &\leq 2\epsilon_{\mathbf{S}} + 2\epsilon_{\hat{\mathbf{S}}} + \epsilon_r + \epsilon_{r_{\hat{s}}} + q_d\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*} + q_d\epsilon_{\hat{\mathbf{H}}}\end{aligned}$$

which is negligible in λ as required. □

4.3 Cramer-Shoup is Simulatable

Theorem 4.3.1. *Suppose that $((X_\Lambda)_{\Lambda\in I_\lambda})_{\lambda\in\mathbb{N}}$ is a computationally simulatable family of sets, with simulator $\mathbf{f}_X = (f_X, f_X^{-1}, n_X)$, that $((\Pi_\Lambda)_{\Lambda\in I_\lambda})_{\lambda\in\mathbb{N}}$ is a computationally simulatable family of sets, with simulator $\mathbf{f}_\Pi = (f_\Pi, f_\Pi^{-1}, n_\Pi)$ and that $((\hat{\Pi}_\Lambda)_{\Lambda\in I_\lambda})_{\lambda\in\mathbb{N}}$ is also a computationally simulatable family of sets, with simulator $\mathbf{f}_{\hat{\Pi}} = (f_{\hat{\Pi}}, f_{\hat{\Pi}}^{-1}, n_{\hat{\Pi}})$. Suppose also that the subset membership problem $\mathbf{M}$ is hard, $\mathbf{H}$ is δ-smooth and that $\hat{\mathbf{H}}$ is both ϵ-universal$_2$ and γ-smooth. Then the Cramer-Shoup encryption scheme based on the hash proof systems $\mathbf{H}$ and $\hat{\mathbf{H}}$ is simulatable.*

Proof. We construct a simulator $\mathbf{f_{CS}} = (f_{\mathbf{CS}}, f_{\mathbf{CS}}^{-1}, n_{\mathbf{CS}})$ for the Cramer-Shoup scheme, where $n_{\mathbf{CS}}(\lambda) = n_X(\lambda) + n_\Pi(\lambda) + n_{\hat{\Pi}}(\lambda))$ for all $\lambda \in \mathbb{N}$, and $f_{\mathbf{CS}}$, $f_{\mathbf{CS}}^{-1}$ are as follows:

function $f_{\mathbf{CS}}(1^\lambda, pk, r)$
  Parse r as $r_1||r_2||r_3$ where $|r_1| = n_X(\lambda)$, $|r_2| = n_\Pi(\lambda)$ and $|r_3| = n_{\hat{\Pi}}(\lambda)$
  $x \leftarrow f_X(1^\lambda, \Lambda, r_1)$
  $e \leftarrow f_\Pi(1^\lambda, r_2)$
  $\hat{\pi} \leftarrow f_{\hat{\Pi}}(1^\lambda, r_3)$
  return $(x, e, \hat{\pi})$

function $f_{\mathbf{CS}}^{-1}(1^\lambda, pk, C)$
  Parse C as $(x, e, \hat{\pi})$
  $r_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, x)$

$r_2 \leftarrow f_{\Pi}^{-1}(1^\lambda, e)$
$r_3 \leftarrow f_{\hat{\Pi}}^{-1}(1^\lambda, \hat{\pi})$
return $r_1||r_2||r_3$

Since the ciphertext space of **CS**, $\mathbf{CiphSp}_{\mathbf{CS}}(pk) = X_\Lambda \times \Pi_\Lambda \times \hat{\Pi}_\Lambda$, the following required properties hold as a direct consequence of the corresponding properties of f_X and f_Π and $f_{\hat{\Pi}}$:

- $f_{\mathbf{CS}}$ is a deterministic algorithm which takes as input $(1^\lambda, pk, r)$ where pk is a public key and $r \in \{0,1\}^{n_{\mathbf{CS}}(\lambda)}$, and outputs an element $C \in \mathbf{CiphSp}(pk)$, the ciphertext space.

- $f_{\mathbf{CS}}^{-1}$ is a probabilistic algorithm which takes as input $(1^\lambda, pk, C)$ where pk is a public key and $C \in \mathbf{CiphSp}(pk)$, and outputs a string $r \in \{0,1\}^{n_{\mathbf{CS}}(\lambda)}$.

- For all public keys pk, and for all $C \in \mathbf{CiphSp}(pk)$, $f_{\mathbf{CS}}(pk, f_{\mathbf{CS}}^{-1}(pk, C)) = C$.

We must now show that for all probabilistic, polynomial-time algorithms $\mathcal{A}$, the Rand-Sim advantage of $\mathcal{A}$ against $f_{\mathbf{CS}}$ is negligible. Let $\mathcal{A}$ be an arbitrary probabilistic, polynomial-time Rand-Sim adversary.

We will now show that

$$\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{CS},\mathcal{A},\mathbf{f_{CS}}}(\lambda) \leq \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{X,\mathcal{B},f_X}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Pi,\mathcal{C},\mathbf{f}_\Pi}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\hat{\Pi},\mathcal{D},\mathbf{f}_{\hat{\Pi}}}(\lambda)$$

for some polynomial-time adversaries $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$. The proof is structured as a sequence of games. We let T_i be the event that $b' = 1$ in game i.

Game 0: Let Game 0 be the experiment $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{\mathbf{CS},\mathcal{A},\mathbf{f_{CS}}}(\lambda)$. Written out in full it is as follows:

$\Lambda \xleftarrow{R} I_\lambda$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
$b' \leftarrow \mathcal{A}(1^\lambda, \Lambda, r_1||r_2||r_3)$
return b

Note that we have written r_1, r_2 and r_3 separately, but this is simply a change of notation, since $n_{\mathbf{CS}}(\lambda) = n_X(\lambda) + n_\Pi(\lambda) + n_{\hat{\Pi}}(\lambda)$.

Game 1: Let Game 1 be as follows:

$\Lambda \xleftarrow{R} I_\lambda$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
$r_1' \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$b' \leftarrow \mathcal{A}(1^\lambda, \Lambda, r_1'||r_2||r_3)$
return b

We construct an adversary $\mathcal{B}$ against the Rand-Sim property of (X_Λ) as follows:

function $\mathcal{B}(1^\lambda, \Lambda, r_1)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
  $r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
  $b \leftarrow \mathcal{A}(1^\lambda, pk, r_1||r_2||r_3)$
  return b

We now see that Game 0 is simply $\mathbf{Expt}^{\mathrm{Rand-Sim-0}}_{X,\mathcal{B},f_X}(\lambda)$ while Game 1 is $\mathbf{Expt}^{\mathrm{Rand-Sim-1}}_{X,\mathcal{B},f_X}(\lambda)$. Thus

$$|\Pr[T_1] - \Pr[T_0]| = \mathbf{Adv}^{\mathrm{Rand-Sim}}_{X,\mathcal{B},f_X}(\lambda)\,.$$

Game 2: Let Game 2 be as follows:

$\Lambda \xleftarrow{R} I_\lambda$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$

$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
$r'_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$r'_2 \leftarrow f_\Pi^{-1}(1^\lambda, \Lambda, f_\Pi(1^\lambda, \Lambda, r_1))$
$b \leftarrow \mathcal{A}(1^\lambda, \Lambda, r'_1||r'_2||r_3)$
return b

We construct an adversary $\mathcal{C}$ against the Rand-Sim property of (Π_Λ) as follows:

function $\mathcal{C}(1^\lambda, \Lambda, r_2)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
$r'_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$b \leftarrow \mathcal{A}(1^\lambda, pk, r'_1||r_2||r_3)$
return b

We now see that Game 2 is simply $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{\Pi,\mathcal{C},f_\Pi}(\lambda)$ while Game 2 is $\mathbf{Expt}^{\text{Rand}-\text{Sim}-1}_{\Pi,\mathcal{C},f_\Pi}(\lambda)$. Thus

$$|\Pr[T_2] - \Pr[T_1]| = \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Pi,\mathcal{B},f_\Pi}(\lambda)\,.$$

Game 3: Let Game 3 be $\mathbf{Expt}^{\text{Rand}-\text{Sim}}_{\mathbf{CS},\mathcal{A},f_{\mathbf{CS}}}(\lambda)$. Written in full it is as follows:

$\Lambda \xleftarrow{R} I_\lambda$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$r_3 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$
$r'_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$r'_2 \leftarrow f_\Pi^{-1}(1^\lambda, f_\Pi(1^\lambda, r_2))$
$r'_3 \leftarrow f_{\hat{\Pi}}^{-1}(1^\lambda, f_{\hat{\Pi}}(1^\lambda, r_3))$
$b \leftarrow \mathcal{A}(1^\lambda, \Lambda, r'_1||r'_2||r'_3)$
return b

We construct an adversary $\mathcal{D}$ against the Rand-Sim property of $\hat{\Pi}$ as follows:

function $\mathcal{D}(1^\lambda, r_3)$
$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$r_1' \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$r_2' \leftarrow f_\Pi^{-1}(1^\lambda, f_\Pi(1^\lambda, r_2))$
$b \leftarrow \mathcal{A}(1^\lambda, pk, r_1'||r_2'||r_3)$
return b

We now see that Game 1 is simply $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{\hat{\Pi},\mathcal{D},f_{\hat{\Pi}}}(\lambda)$, while Game 2 is $\mathbf{Expt}^{\text{Rand}-\text{Sim}-1}_{\hat{\Pi},\mathcal{D},f_{\hat{\Pi}}}(\lambda)$. Thus

$$|\Pr[T_3] - \Pr[T_2]| = \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\hat{\Pi},\mathcal{D},f_{\hat{\Pi}}}(\lambda)\,.$$

Putting it all together we see that

$$\begin{aligned}
\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{CS},\mathcal{A},f_{\mathbf{CS}}}(\lambda) &= |\Pr[T_3] - \Pr[T_0]| \\
&\leq |\Pr[T_1] - \Pr[T_0]| + |\Pr[T_2] - \Pr[T_1]| + |\Pr[T_3] - \Pr[T_2] \\
&= \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{X,\mathcal{B},f_X}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Pi,\mathcal{C},f_\Pi}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\hat{\Pi},\mathcal{D},f_{\hat{\Pi}}}(\lambda)
\end{aligned}$$

Thus $\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{CS},\mathcal{A},f_{\mathbf{CS}}}(\lambda)$ is negligible in λ as required.

Finally, we must show that for any PKE-Sim adversary $\mathcal{A}$, the advantage $\mathbf{Adv}^{\text{PKE}-\text{Sim}}_{\mathbf{CS},\mathcal{A},f_{\mathbf{CS}}}(\lambda)$ is negligible in λ. Let $\mathcal{A}$ be an arbitrary adversary against the PKE-Sim property of **CS**. Let q_d be an upper bound for the number of decryption queries made by $\mathcal{A}$. We prove that the $\mathbf{Adv}^{\text{PKE}-\text{Sim}}_{\mathbf{CS},\mathcal{A},f_{\mathbf{CS}}}(\lambda)$ is negligible in λ using a sequence of games. Let T_i be the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\text{PKE}-\text{Sim}-0}_{\mathbf{CS},\mathcal{A},\mathbf{f_{CS}}}(\lambda)$. Written in full, it is as follows:

$\mathbf{Expt}^{\text{PKE-Sim-0}}_{\mathbf{CS},\mathcal{A},\mathbf{f_{CS}}}(\lambda)$
$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, pk)$
Parse pk as $(s, \hat{s})$
$(x^*, w^*) \leftarrow \mathsf{SSA}(\Lambda)$
$\pi^* \leftarrow \mathsf{Public}_{\mathbf{P}}(s, x^*, w^*)$
$e^* \leftarrow m + \pi^*$
$\hat{\pi}^* \leftarrow \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x^*, e^*, w^*)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, C^*, \mathsf{state})$
return b'

$\mathcal{A}$ has access to a decryption oracle $\mathsf{Decrypt}$ which takes a ciphertext C as input and returns $\mathsf{Decrypt}(sk, C)$. This oracle is restricted so that $\mathcal{A}$ may not request the decryption of the challenge ciphertext C^*.

Game 1: We modify the challenge ciphertext so that it is generated using the $\mathsf{Private}_{\mathbf{P}}$ and $\mathsf{Private}_{\hat{\mathbf{P}}}$ evaluation functions instead of the Public functions. Written in full, it is as follows:

$\mathbf{Expt}^{\text{PKE-Sim-0}}_{\mathbf{CS},\mathcal{A},\mathbf{f_{CS}}}(\lambda)$
$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, pk)$
$\boxed{\text{Parse } sk \text{ as } (k, \hat{k})}$
$(x^*, w^*) \leftarrow \mathsf{SSA}(\Lambda)$
$\boxed{\pi^* \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x^*)}$
$e^* \leftarrow m + \pi^*$
$\boxed{\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)}$
$C \leftarrow (x^*, e^*, \hat{\pi}^*)$
return C
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, C^*, \mathsf{state})$
return b'

This is a bridging step; $\mathsf{Private}_{\mathbf{P}}(k, x^*) = H_k(x^*) = \mathsf{Public}_{\mathbf{P}}(s, x^*, w^*)$ since $x^* \in L$, and similarly, $\mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*) = \hat{H}_{\hat{k}}(x^*, e^*) = \mathsf{Public}_{\hat{\mathbf{P}}}(\hat{s}, x^*, e^*, w^*)$ so the challenge ciphertext in Game 1 is equal to the challenge ciphertext in

game 0. Thus $\Pr[T_1] = \Pr[T_0]$.

Game 2: This is a transition based on indistinguishability. We choose x^* at random from $X \setminus L$ instead of using the subset sampling algorithm to generate $x^* \in L$. Written out in full, it is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, pk)$
Parse sk as $(k, \hat{k})$
$\boxed{x^* \xleftarrow{\mathrm{R}} X \setminus L}$
$\pi^* \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x^*)$
$e^* \leftarrow m + \pi^*$
$\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(1^\lambda, C^*, \mathsf{state})$
return b'

In this game, x is chosen at random from $X \setminus L$. We do not assume that sampling $x \xleftarrow{\mathrm{R}} X \setminus L$ can be done in polynomial time, since the challenger need not be efficient.

Let $\mathcal{B}$ be the adversary described below. $\mathcal{B}$ takes an element $x^* \in X$ and runs as follows:

function $\mathcal{B}(1^\lambda, \Lambda, x^*)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
  $\pi^* \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x^*)$
  $e^* \leftarrow m + \pi^*$
  $\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)$
  $C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
  return b'

To respond to a query $\mathsf{Decrypt}(C)$, $\mathcal{B}$ computes $m \leftarrow \mathsf{Decrypt}(sk, C)$ and returns m. If x^* is chosen at random from L, then $\mathcal{B}$ exactly simulates Game

1, but if $x^* \xleftarrow{R} X \setminus L$, $\mathcal{B}$ exactly simulates Game 2. Thus

$$\Pr[\mathcal{B}(1^\lambda, \Lambda, x^*) = 1 : x^* \xleftarrow{R} L] = \Pr[T_1]$$

and

$$\Pr[\mathcal{B}(1^\lambda, \Lambda, x^*) = 1 : x^* \xleftarrow{R} X \setminus L] = \Pr[T_2]\,,$$

so

$$|\Pr[T_2] - \Pr[T_1]| \leq \mathbf{Adv}^{\mathrm{SMP}}_{\mathbf{M},\mathcal{B}}(\lambda)\,.$$

Game 3: We modify the $\mathsf{Decrypt}$ oracle so that if $x \notin L$, the oracle returns $\perp$

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    if $\boxed{x \notin L}$ **then**
      $\boxed{\textbf{return } \perp}$
    else
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
$x^* \xleftarrow{R} X \setminus L$
$\pi^* \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x^*)$
$e^* \leftarrow m + \pi^*$
$\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$

```
        if [x ∉ L] then
            [return ⊥]
        else
            π̂′ ← Private_P̂(k̂, x, e)
            if π̂′ = π̂ then
                π ← Private_P(k, x)
                m ← e − π
                return m
            else
                return ⊥
    return b′
```

In Game 2, any ciphertext $C = (x, e, \hat{\pi})$ is rejected if $\hat{H}_{\hat{k}}(x, e) \neq \hat{\pi}$. In Game 3, ciphertexts are additionally rejected if $x \notin L$. Let F_3 be the event that $\mathcal{A}$ makes a query $\mathsf{Decrypt}(x, e, \hat{\pi})$ such that $x \notin L$ but $\hat{H}_{\hat{k}}(x, e) = \hat{\pi}$. If F_3 does not occur then Game 2 and Game 3 proceed identically, since all all decryption queries for which $x \notin L$ are rejected.

Consider the conditional probability space defined by fixing the random coins of the adversary and the public key $pk = (s, \hat{s})$. This determines the message m output by $\mathcal{A}_1$. In addition, fix the values x^* and $\hat{\pi}^*$ used in computing the challenge ciphertext, and the value k. This determines the challenge ciphertext $(x^*, e^*, \hat{\pi}^*)$, because e^* is a function of x^*, m and k. In this conditional probability space, $\hat{k}$ is uniformly distributed on the set $\{\hat{k} \in \hat{K} | \hat{\alpha}(\hat{k}) = \hat{s}\}$. For any decryption query made by $\mathcal{A}$ on a ciphertext $C = (x, e, \hat{\pi})$ where $x \notin L$, there are two possibilities:

1. If $(x, e) = (x^*, e^*)$ then $\hat{\pi} \neq \hat{\pi}^*$ since $\mathcal{A}$ may not decrypt the challenge ciphertext. But $\hat{H}_{\hat{k}}(x^*, e^*) = \hat{\pi}^*$ by definition, so $\hat{H}_{\hat{k}}(x, e) = \hat{\pi}^* \neq \hat{\pi}$

2. If $(x, e) \neq (x^*, e^*)$ then in the conditional probability space defined above, $\Pr[\hat{H}_{\hat{k}}(x, e) = \hat{\pi}] \leq \epsilon(\lambda)$ by the ϵ-universal$_2$ property of $\hat{\mathbf{P}}$.

Thus $\Pr[F_3] \leq q_d \epsilon(\lambda)$ and so

$$|\Pr[T_3] - \Pr[T_2]| \leq q_d \epsilon(\lambda)\,.$$

Game 4: We choose π^* at random.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    if $x \notin L$ **then**
      return $\perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
$x^* \xleftarrow{\text{R}} X \setminus L$
$\boxed{\pi^* \xleftarrow{\text{R}} \Pi}$
$e^* \leftarrow m + \pi^*$
$\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    if $x \notin L$ **then**
      return $\perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
return b'

Let x^*, e^*, s and π_3^*, be as in Game 3, let π_4^* be as in Game 4. By the δ-smoothness of $\mathbf{H}$, we see that $\Delta[(x^*, e^*, \hat{s}, \pi_3^*), (x^*, e^*, \hat{s}, \pi_4^*)] \leq \delta(\lambda)$. By Lemma 1.2.1 it follows that

$$|\Pr[T_4] - \Pr[T_3]| \leq \delta(\lambda)\,.$$

Game 5: We choose e^* at random.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    if $x \notin L$ **then**
      return $\perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
$x^* \xleftarrow{\mathrm{R}} X \setminus L$
$\boxed{e^* \xleftarrow{\mathrm{R}} \Pi}$
$\hat{\pi}^* \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x^*, e^*)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    if $x \notin L$ **then**
      return $\perp$
    else
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m

else
return $\perp$
return b'

This game is a bridging step: In Game 4, $e^* = \pi^* + m$ and π^* is selected uniformly at random, so e^* is distributed at random. Since the distribution of e^* is unchanged, we see that

$$\Pr[T_5] = \Pr[T_4]$$

Game 6: We choose $\hat{\pi}^*$ at random

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
if $x \notin L$ **then**
return $\perp$
else
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
$x^* \xleftarrow{\text{R}} X \setminus L$
$e^* \xleftarrow{\text{R}} \Pi$
$\boxed{\hat{\pi}^* \xleftarrow{\text{R}} \hat{\Pi}}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
if $x \notin L$ **then**
return $\perp$
else
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$

if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
return b'

Let x^*, e^*, s and $\hat{\pi}_5^*$, be as in Game 5, let $\hat{\pi}_6^*$ be as in Game 6. By the γ-smoothness of $\hat{\mathbf{H}}$, we see that we see that $\Delta[(x^*, e^*, \hat{s}, \hat{\pi}_5^*), (x^*, e^*, \hat{s}, \hat{\pi}_6^*)] \leq \gamma(\lambda)$. By Lemma 1.2.1 it follows that

$$|\Pr[T_6] - \Pr[T_5]| \leq \gamma(\lambda)\,.$$

Game 7: We modify the decryption oracle once more so that it no longer checks if $x \in L$, i.e. it runs as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
$x^* \xleftarrow{\text{R}} X \setminus L$
$e^* \xleftarrow{\text{R}} \Pi$
$\hat{\pi}^* \xleftarrow{\text{R}} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**

$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
return b'

By the same reasoning as in Game 3

$$|\Pr[T_7] - \Pr[T_6]| \leq q_d \epsilon(\lambda)\,.$$

We note that in Dent's proof [14] of plaintext awareness for the DDH based Cramer-Shoup scheme, the decryption oracle is not returned to its original behaviour and the proof is therefore incomplete.

Game 8: We select x^* from X instead of $X \setminus L$:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
$\boxed{x^* \xleftarrow{\mathrm{R}} X}$
$e^* \xleftarrow{\mathrm{R}} \Pi$
$\hat{\pi}^* \xleftarrow{\mathrm{R}} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$

$m \leftarrow e - \pi$
return m
else
return $\perp$
return b'

Let $\mathcal{C}$ be the following adversary:

function $\mathcal{C}(1^\lambda, \Lambda, x^*)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
$e^* \xleftarrow{\text{R}} \Pi$
$\hat{\pi}^* \xleftarrow{\text{R}} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
return b'

If $\mathcal{C}$ is given $x^* \xleftarrow{\text{R}} X \setminus L$ as input, then it exactly simulates Game 7, while if it is given $x^* \xleftarrow{\text{R}} X$, then it exactly simulates Game 8.

By Lemma 3.1.8 and Lemma 1.2.1 this implies that

$$|\Pr[T_8] - \Pr[T_7]| \leq \frac{|L|}{|X|}\mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda)\,.$$

Game 9: We choose x^* using the simulation function f_X:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
$\boxed{r_1 \xleftarrow{\mathrm{R}} \{0,1\}^{n_X(\lambda)}}$
$\boxed{x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)}$
$e^* \xleftarrow{\mathrm{R}} \Pi$
$\hat{\pi}^* \xleftarrow{\mathrm{R}} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
return b'

Note that the adversary $\mathcal{C}$ defined in Game 8 serves just as well as a Set-Sim adversary as it does as an adversary against the subset membership problem.

If $\mathcal{C}$ is given $x^* \xleftarrow{\mathrm{R}} X$ as input, then it exactly simulates Game 8, but if $\mathcal{C}$ is given $x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$, where $r_1 \xleftarrow{\mathrm{R}} \{0,1\}^{n_X(\lambda)}$ then $\mathcal{C}$ exactly simulates

Game 9. By the Set-Sim property of X,

$$|\Pr[T_9] - \Pr[T_8]| = \mathbf{Adv}^{\text{Set-Sim}}_{X,\mathcal{C},f_X}(\lambda)\,.$$

Game 10: We choose e^* using the simulation function f_Π:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$\boxed{r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}}$
$x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
$\boxed{e^* \leftarrow f_\Pi(1^\lambda, \Lambda, r_2)}$
$\hat{\pi}^* \xleftarrow{R} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
return b'

We construct an adversary $\mathcal{D}$ against the Set-Sim property of Π in the natural way:

function $\mathcal{D}(1^\lambda, \Lambda, e^*)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  Parse sk as $(k, \hat{k})$
  Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
    if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
      Parse C as $(x, e, \hat{\pi})$
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
  $r_1 \xleftarrow{\mathrm{R}} \{0,1\}^{n_X(\lambda)}$
  $x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
  $\hat{\pi}^* \xleftarrow{\mathrm{R}} \hat{\Pi}$
  $C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
  Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
    if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
      Parse C as $(x, e, \hat{\pi})$
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
        $m \leftarrow e - \pi$
        return m
      else
        return $\perp$
  return b'

If $\mathcal{D}$ is given $e^* \xleftarrow{\mathrm{R}} \Pi$ as input, then it exactly simulates Game 9, but if $\mathcal{D}$ is given $e^* \leftarrow f_\Pi(1^\lambda, \Lambda, r_2)$, where $r_1 \xleftarrow{\mathrm{R}} \{0,1\}^{n_\Pi(\lambda)}$ then it exactly simulates Game 10. By the Set-Sim property of Π,

$$|\Pr[T_{10}] - \Pr[T_9]| = \mathbf{Adv}^{\mathrm{Set-Sim}}_{\Pi,\mathcal{D},f_\Pi}(\lambda)\,.$$

Game 11: We choose $\hat{\pi}^*$ using the simulation function $f_{\hat{\Pi}}$:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$

Parse sk as $(k, \hat{k})$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
$r_1 \xleftarrow{\text{R}} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{\text{R}} \{0,1\}^{n_\Pi(\lambda)}$
$\boxed{r_3 \xleftarrow{\text{R}} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}}$
$x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
$e^* \leftarrow f_\Pi(1^\lambda, \Lambda, r_2)$
$\boxed{\hat{\pi}^* \leftarrow f_{\hat{\Pi}}(1^\lambda, \Lambda, r_3)}$
$\hat{\pi}^* \xleftarrow{\text{R}} \hat{\Pi}$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as $(x, e, \hat{\pi})$
    $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
    if $\hat{\pi}' = \hat{\pi}$ **then**
      $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
      $m \leftarrow e - \pi$
      return m
    else
      return $\perp$
return b'

We construct another Set-Sim adversary $\mathcal{E}$ as follows:

function $\mathcal{E}(1^\lambda, \Lambda, \hat{\pi}^*)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  Parse sk as $(k, \hat{k})$
  Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}}(1^\lambda, pk)$
    if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
      Parse C as $(x, e, \hat{\pi})$
      $\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
      if $\hat{\pi}' = \hat{\pi}$ **then**
        $\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$

$m \leftarrow e - \pi$
return m
else
return $\perp$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{R} \{0,1\}^{n_\Pi(\lambda)}$
$x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
$e^* \leftarrow f_\Pi(1^\lambda, \Lambda, r_2)$
$C^* \leftarrow (x^*, e^*, \hat{\pi}^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}}(1^\lambda, C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as $(x, e, \hat{\pi})$
$\hat{\pi}' \leftarrow \mathsf{Private}_{\hat{\mathbf{P}}}(\hat{k}, x, e)$
if $\hat{\pi}' = \hat{\pi}$ **then**
$\pi \leftarrow \mathsf{Private}_{\mathbf{P}}(k, x)$
$m \leftarrow e - \pi$
return m
else
return $\perp$
return b'

If $\mathcal{E}$ is given $\hat{\pi}^* \xleftarrow{R} \hat{\Pi}$ as input, then it exactly simulates Game 10, but if $\mathcal{E}$ is given $\hat{\pi}^* \leftarrow f_{\hat{\Pi}}(1^\lambda, \Lambda, r_3)$, where $r_1 \xleftarrow{R} \{0,1\}^{n_{\hat{\Pi}}(\lambda)}$ then it exactly simulates Game 11. By the Set-Sim property of $\hat{\Pi}$,

$$|\Pr[T_{11}] - \Pr[T_{10}]| = \mathbf{Adv}^{\text{Set-Sim}}_{\hat{\Pi},\mathcal{E},f_{\hat{\Pi}}}(\lambda)\,.$$

We now see that Game 11 is exactly $\mathbf{Expt}^{\text{PKE-Sim}-1}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$. Putting it all together, we see that

$$\begin{aligned}
|\Pr[T_{11}] - \Pr[T_0]| &\leq |\Pr[T_{11}] - \Pr[T_{10}]| + \ldots + |\Pr[T_1] - \Pr[T_0]| \\
&\leq \mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{B}}(\lambda) + 2q_d\epsilon(\lambda) + \delta(\lambda) + \gamma(\lambda) + \frac{|L|}{|X|}\mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda) \\
&\quad + \mathbf{Adv}^{\text{Set-Sim}}_{X,\mathcal{C},f_X}(\lambda) + \mathbf{Adv}^{\text{Set-Sim}}_{\Pi,\mathcal{D},f_\Pi}(\lambda) + \mathbf{Adv}^{\text{Set-Sim}}_{\hat{\Pi},\mathcal{E},f_{\hat{\Pi}}}(\lambda)
\end{aligned}$$

which is negligible in λ as required. □

Finally, we see that the Cramer-Shoup scheme is PA1+ plaintext aware and simulatable, and thus it is PA2 plaintext aware by Theorem 3.1.6.

Chapter 5

The Generalised Kurosawa–Desmedt Encryption Scheme is PA2 Plaintext-Aware

In this chapter, we follow up the previous work on the Cramer-Shoup encryption scheme with a proof that the variants of the Kurosawa-Desmedt encryption scheme based on hash proof systems are also PA2 plaintext aware.

5.1 The Kurosawa-Desmedt Encryption Scheme

We now present the variant of the Kurosawa-Desmedt encryption scheme based on a hash proof system [22]. Let $\mathbf{M}$ be a subset membership problem and let $\mathbf{P}$ be a hash proof system for $\mathbf{M}$. Let $\Lambda = (X, L, W, R) \leftarrow \mathsf{ISA}(1^\lambda)$ be a randomly chosen instance of $\mathbf{M}$, and let $\mathbf{H} = (H, K, X, L, \Pi, S, \alpha)$ be the projective hash family that $\mathbf{P}$ associates with Λ. Let $\Sigma = (\mathsf{enc}, \mathsf{dec})$ with key-length function $\ell(\lambda)$ and let $\mathsf{Hash} : \Pi \to \{0,1\}^{\ell(\lambda)}$ be a hash function. The Kurosawa-Desmedt

universal hash proof encryption scheme is then defined as follows:

function $\mathsf{KeyGen}(\Lambda)$
$sk \xleftarrow{R} K$
$pk \leftarrow \alpha(sk)$
return (pk, sk)

function $\mathsf{Decrypt}(sk, C)$
$\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m

function $\mathsf{Encrypt}(pk, m)$
$(x, w) \leftarrow \mathsf{SSA}(\Lambda)$
$\pi \leftarrow \mathsf{Public}_{\mathbf{H}}(pk, x, w)$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$\chi \leftarrow \mathsf{enc}(\kappa, m)$
$C \leftarrow (x, \chi)$
return C

Note that we have abused the notation slightly, by defining KeyGen to take an instance Λ as input instead of a security parameter 1^λ; this is done to simplify the writing of the proofs. One could define KeyGen so that it first generates an instance $\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$ and then runs as above without affecting the results.

Theorem 5.1.1 (Kurosawa-Desmedt)**.** *Let* $\mathbf{M}$ *be a subset membership problem,* $\mathbf{H}$ *be a projective hash proof system, let* Σ *be a DEM and let* Hash *be a hash function. Then if* $\mathbf{M}$ *is hard,* $\mathbf{H}$ *is* $1/|\Pi|$*-universal*$_2$*,* Σ *is both one-time IND-CCA2 secure and* ϵ*-rejection secure, and* Hash *is* δ*-smooth, where* ϵ *and* δ *are negligible, then the resulting scheme is IND-CCA2 secure.*

5.2 Kurosawa-Desmedt is PA1+

Theorem 5.2.1. *Suppose that* $\mathbf{H}$ *is* $1/|\Pi|$*-universal*$_2$*, that* Σ *is* ϵ*-rejection secure for some negligible* ϵ*, the hash function* Hash *is* δ*-smooth for some negligible* δ*, the public keys returned by* $\mathsf{KeyGen}(\Lambda)$ *are uniformly distributed on the set* S *and that the family of sets* $\mathbf{S} = ((S_\Lambda)_{\Lambda \in I_\lambda})_{\lambda \in \mathbb{N}}$ *of public keys*

is statistically simulatable, with simulator $\mathbf{f_S} = (f_{\mathbf{S}}, f_{\mathbf{S}}^{-1}, n_{\mathbf{S}})$. *Assume also that the SWK assumption holds for the underlying subset membership problem* $\mathbf{M}$. *Then the Kurosawa-Desmedt encryption scheme based on the hash proof system* $\mathbf{H}$ *is PA1+.*

The proof of this is almost identical to the proof of Theorem 4.2.1, the analogous theorem for the Cramer-Shoup encryption scheme. The only differences are technical, the result of the differences between the two encryptions schemes, but conceptually the proof follows the same steps. Nevertheless, we include the proof in its entirety here.

Proof. Let $\mathcal{A}$ be a PA1+ ciphertext creator which makes at most q_r randomness queries. We will construct a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ by constructing an SWK adversary $\mathcal{B}$ and using the existence of an SWK extractor $\mathcal{B}^*$ to construct a plaintext extractor $\mathcal{A}^*$. In the following, we abuse notation slightly by treating Rlist as either a list or a string. However, the appropriate meaning should be clear from the context.

$\mathcal{B}$ takes input $\Lambda = (X, L, W, R)$ and works as follows:

function $\mathcal{B}(\Lambda)$
  $r \xleftarrow{R} \{0,1\}^{n_{\mathbf{S}}(\lambda)}$
  $\mathsf{Rlist} \xleftarrow{R} \{0,1\}^{q_r}$
  $pk \leftarrow f_{\mathbf{S}}(r)$
  Run $aux \leftarrow \mathcal{A}(1^\lambda, pk)$
    if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
      $n_r \leftarrow n_r + 1$
      return $\mathsf{Rlist}[n_r]$
    if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
      Parse C as (x, χ)
      $w \leftarrow$ Query $\mathsf{SWK}(x)$
      if $(x, w) \notin R$ **then**
        return $\perp$
      else
        $\pi \leftarrow \mathsf{Public}_{\mathbf{H}}(s, x, w)$

$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m

By the SWK assumption, there is a witness extractor $\mathcal{B}^*$ such that the advantage $\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)$ is negligible in λ. We use $\mathcal{B}^*$ to construct a PA1+ plaintext extractor $\mathcal{A}^*$ as follows:

function $\mathcal{A}^*(pk, C, R[\mathcal{A}], \mathsf{Rlist}, \mathsf{state}_{\mathcal{A}^*})$
  $s \leftarrow pk$
  if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
    $r \leftarrow f_{\mathbf{S}}^{-1}(s)$
  else
    Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(r, x_1, \ldots, x_{n_d-1})$
  Parse C as (x_{n_d}, χ)
  $n_r \leftarrow |\mathsf{Rlist}|$
  $\mathsf{Rlist}' \xleftarrow{\mathrm{R}} \{0,1\}^{q_r - n_r}$
  $R_{\mathcal{B}} \leftarrow r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$
  $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
  for $t = 1$ to n_d **do**
    $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
  if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
    $m \leftarrow \perp$
  else
    $\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d,n_d})$
    $\kappa \leftarrow \mathsf{Hash}(\pi)$
    $m \leftarrow \mathsf{dec}(\kappa, \chi)$
  $\mathsf{state}_{\mathcal{A}^*} \leftarrow (r, x_1, \ldots, x_{n_d})$
  return $(m, \mathsf{state}_{\mathcal{A}^*})$

We now proceed by a sequence of games to show that $\mathbf{Adv}^{\mathrm{PA1+}}_{\mathbf{KD},\mathcal{A},\mathcal{A}^*,D}(\lambda)$ is negligible in λ. Let T_i be the event that $D(aux)$ returns 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\mathrm{PA1+Fake}}_{\mathbf{KD},\mathcal{A},\mathcal{A}^*,D}(\lambda)$. Written out in full, it is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$s \leftarrow pk$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow \varepsilon$

Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\text{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as (x_{n_d}, χ)
    if $\mathsf{state}_{\mathcal{A}^*} = \varepsilon$ **then**
      $r \leftarrow f_{\mathbf{S}}^{-1}(s)$
    else
      Parse $\mathsf{state}_{\mathcal{A}^*}$ as $(r, x_1, \ldots, x_{n_d-1})$
    $\mathsf{Rlist}' \xleftarrow{\text{R}} \{0,1\}^{q_r - n_r}$
    $R_{\mathcal{B}} \leftarrow r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    for $t = 1$ to n_d **do**
      $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
    if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
      return $\perp$
    $\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d,n_d})$
    $\kappa \leftarrow \mathsf{Hash}(\pi)$
    $m \leftarrow \mathsf{dec}(\kappa, \chi)$
    $\mathsf{state}_{\mathcal{A}^*} \leftarrow (r, x_1, \ldots, x_{n_d})$
    return m
$b \leftarrow D(aux)$

Game 1: Let Game 1 be as Game 0, except that we choose a public key at random. Since we have written $\mathcal{A}$, $\mathcal{A}^*$ and the challenger as one algorithm, we no longer track the $\mathsf{state}_{\mathcal{A}^*}$ variable, as it just consists of constants we are recording already. Instead of generating $r \leftarrow f_{\mathbf{S}}^{-1}(s)$ the first time the decryption oracle is used, we move this to the beginning of the game.

Written out in full it is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{s \xleftarrow{\text{R}} S}$
$pk \leftarrow s$
$\boxed{r \leftarrow f_{\mathbf{S}}^{-1}(s)}$

Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    Parse C as (x_{n_d}, χ)
    $\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$
    $R_{\mathcal{B}} \leftarrow r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$
    $\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
    for $t = 1$ to n_d **do**
      $(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
    if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
      return $\perp$
    $\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d,n_d})$
    $\kappa \leftarrow \mathsf{Hash}(\pi)$
    $m \leftarrow \mathsf{dec}(\kappa, \chi)$
    return m
$b \leftarrow D(aux)$

By assumption that s is uniformly distributed on S_Λ, the distribution of pk is unchanged, so $\Pr[T_1] = \Pr[T_0]$

Game 2: We simulate s using the simulator $f_{\mathbf{S}}(r')$ where r' is a random string. Note that we continue to generate the string r which gets passed to $\mathcal{B}^*$ using $f_{\mathbf{S}}^{-1}(s)$, which is not in general equal to r'.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{r' \xleftarrow{R} \{0,1\}^{n(\lambda)}}$
$\boxed{s \leftarrow f_{\mathbf{S}}(r')}$
$r \leftarrow f_{\mathbf{S}}^{-1}(s)$
$pk \leftarrow s$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{R} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$

if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Parse C as (x_{n_d}, χ)
$\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$
$R_{\mathcal{B}} \leftarrow r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$
$\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
for $t = 1$ to n_d **do**
$(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d,n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$b \leftarrow D(aux)$

By the statistical Set-Sim property of $\mathbf{S}$, $\epsilon_{\mathbf{S}} = \Delta[(1^\lambda, \Lambda, s), (1^\lambda, \Lambda, s')]$ is negligible in λ. Thus

$$|\Pr[T_2] - \Pr[T_1]| \leq \epsilon_{\mathbf{S}}(\lambda)$$

by Lemma 1.2.1.

Game 3: We provide the randomness r' that we used to simulate pk to $\mathcal{B}^*$, instead of using $f_{\mathbf{S}}^{-1}$ to generate a second random string. Written out in full, the game is as follows:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r' \xleftarrow{R} \{0,1\}^{n(\lambda)}$
$s \leftarrow f_{\mathbf{S}}(r')$
$r \leftarrow f_{\mathbf{S}}^{-1}(s)$
$pk \leftarrow s'$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$b \xleftarrow{R} \{0,1\}$
Append b to Rlist
return b
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Parse C as (x_{n_d}, χ)
$\mathsf{Rlist}' \xleftarrow{R} \{0,1\}^{q_r - n_r}$

$\boxed{R_{\mathcal{B}} \leftarrow r' || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]}$
$\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
for $t = 1$ to n_d **do**
$(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d,n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$b \leftarrow D(aux)$

By the statistical Rand-Sim property of $\mathbf{S}$, $\epsilon_r = \Delta[(1^\lambda, \Lambda, r), (1^\lambda, \Lambda, r')]$ is negligible in λ. We may regard $\mathcal{A}$ together with the Randomness and Decrypt oracles as a single algorithm which takes either r or r' as one of its inputs, so by Lemma 1.2.1, this implies

$$|\Pr[T_3] - \Pr[T_2]| \leq \epsilon_r(\lambda)\,.$$

Game 4: We replace $\mathcal{B}^*$ with an idealised algorithm that always returns a valid witness if one exists. Note that the value r computed in Game 3 is no longer used, so we omit it here.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r' \xleftarrow{R} \{0,1\}^{n(\lambda)}$
$s' \leftarrow f_{\mathbf{S}}(r')$
$pk \leftarrow s'$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
if $\mathcal{A}$ queries Randomness **then**
$b \xleftarrow{R} \{0,1\}$
Append b to Rlist
return b
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
Parse C as (x_{n_d}, χ)
if $\boxed{\text{there exists } w \text{ such that } (x_{n_d}, w) \in R}$ **then**
$\boxed{w_{n_d} \leftarrow w}$

else
return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$b \leftarrow D(aux)$

We do not require this computation to be performed in polynomial time. Let F_3 be the event that $x_t \in L$ but $(x_t, w_{n_d,t}) \notin R$ for some n_d, t in Game 3. If F_3 does not occur then each ciphertext is decrypted correctly and Game 3 proceeds identically to Game 4. We will show that $\Pr[F_3]$ is negligible via a sequence of games, Game $(3,0), \ldots,$ Game $(3, q_d)$, where Game $(3,0)$ = Game 3 and Game $(3, q_d)$ = Game 4.

Game $(3, i)$:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$r' \xleftarrow{\mathrm{R}} \{0,1\}^{n(\lambda)}$
$s' \leftarrow f_{\mathbf{s}}(r)$
$pk \leftarrow (\Lambda, s')$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$n_r \leftarrow n_r + 1$
$b \xleftarrow{\mathrm{R}} \{0,1\}$
Append b to Rlist
return $\mathsf{Rlist}[n_r]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
$C_{n_d} \leftarrow C$
Parse C as (x_{n_d}, χ)
if $n_d \leq i$ **then**
if there exists w such that $(x_{n_d}, w) \in R$ **then**
$w_{n_d} \leftarrow w$
else
return $\perp$
else
$\mathsf{Rlist}' \xleftarrow{\mathrm{R}} \{0,1\}^{q_r - n_r}$
$\boxed{R_{\mathcal{B}} \leftarrow r' || \mathsf{Rlist} || \mathsf{Rlist}' || R[\mathcal{A}]}$

$\mathsf{state}_{\mathcal{B}^*} \leftarrow \varepsilon$
for $t = 1$ to n_d **do**
$(w_{n_d,t}, \mathsf{state}_{\mathcal{B}^*}) \leftarrow \mathcal{B}^*(\Lambda, x_t, R_{\mathcal{B}}, \mathsf{state}_{\mathcal{B}^*})$
if $(x_{n_d}, w_{n_d,n_d}) \notin R$ **then**
return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow (r, x_1, \ldots, x_{n_d})$
return m
$b \leftarrow D(aux)$

Let $F_{3,i}$ be the event that $x_i \in L$ but $(x_i, w_{i,i}) \notin R$. If $F_{3,i}$ does not occur then Game $(3, i)$ proceeds identically to Game $(3, i-1)$.

We would like to show that each time $\mathcal{B}^*$ is called during the i^{th} decryption query in Game $(3, i-1)$, the inputs to $\mathcal{B}^*$, namely $\Lambda, x_t, R_{\mathcal{B}}$ and $\mathsf{state}_{\mathcal{B}^*}$, are distributed as in $\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)$. Λ is correctly sampled using ISA and each of the bits of $R_{\mathcal{B}} = r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$ are distributed uniformly at random (though the random tape used in subsequent queries depends on this random tape, an issue we will return to below). The big difference is that in Game $(3, i)$, the list of elements $(x_1, \ldots, x_{i-1})$ obtained from previous decryption queries are successively used as inputs to $\mathcal{B}^*$, and this list is fixed prior to the execution of $\mathcal{B}^*$. In contrast, in the SWK experiment, the output of $\mathcal{B}^*$ at any stage is returned to $\mathcal{B}$, and so may affect the next query made by $\mathcal{A}$. However, whenever $\mathcal{B}^*$ returns a valid witness, $\mathcal{B}$ correctly decrypts the ciphertext, so the message returned to $\mathcal{A}$ is independent of the choice of witness.

More formally, consider running $\mathbf{Expt}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)$, using the instance Λ and with $\mathcal{B}$'s random tape set to $r||\mathsf{Rlist}||\mathsf{Rlist}'||R[\mathcal{A}]$, where $\Lambda, r, \mathsf{Rlist}, \mathsf{Rlist}'$ and $R[\mathcal{A}]$ are as above. To avoid ambiguity, we will use $\hat{x}_i$ to distinguish variables in the SWK experiment from their counterparts in Game $(3, i-1)$. We will now show by induction that $x_t = \hat{x}_t$ for $1 \leq t \leq i-1$.

By definition of $\mathcal{B}$ we see that $\mathcal{B}$ computes a public key $pk \leftarrow f_{\mathbf{S}}(r)$, and runs $\mathcal{A}(pk; R[\mathcal{A}])$. Let $1 \leq t \leq i-1$, and suppose by induction that for all $1 \leq t' \leq t$, $\hat{x}_{t'} = x_{t'}$, and $(\hat{x}_{t'}, \hat{w}_{t'}) \in R$ if $\hat{x}_{t'} \in L$. $\hat{w}_{t'}$ may be different to $w_{i,t'}$, but these values are never passed on to $\mathcal{A}$. Since $\mathsf{Public}(s, \hat{x}_{t'}, \hat{w}_{t'}) = H_k(\hat{x}_{t'})$ for all valid witnesses, it follows that the decrypted messages returned to $\mathcal{A}$ are identical in the two games. The values of Rlist are identical in the two experiments, and $\mathcal{B}$ does not pass any of $\hat{\mathsf{Rlist}}'$ to $\mathcal{A}$ until after the i^{th} decryption query is made the responses to $\mathcal{A}$'s randomness queries are also identical, because $\mathcal{B}$ does not use any part of $\hat{\mathsf{Rlist}}'$ to $\mathcal{A}$ until after the i^{th} decryption query is made.

This shows that the inputs to $\mathcal{A}$ in the SWK experiment, including the randomness, are identical to their values in Game $(3, i-1)$. It follows that $\hat{x}_t = x_t$, since ciphertexts that $\mathcal{A}$ passes to its $\mathsf{Decrypt}$ oracle are functions of $\mathcal{A}$'s randomness, it's inputs and the responses to previous queries, which are all held fixed. By induction, this implies that all inputs to $\mathcal{B}^*$ have the same distribution in the i^{th} decryption query of Game $(3, i-1)$ as they do in the SWK experiment, so

$$\Pr[F_{3,i}] = \Pr[x_t \in L \wedge (x_t, w_{i,t}) \notin R] \leq \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)\,.$$

We must now address the dependence of the events F_j Consider the event $G_j = \bigcup_{i=1}^{j} F_{3,j}$. Then $G_1 = F_1$ and for all $1 \leq j \leq q_d$,

$$\begin{aligned}
\Pr[G_{j+1}] &= \Pr[G_j \cup F_{3,j+1}] \\
&= \Pr[G_j] + \Pr[F_{3,j+1}] - \Pr[G_j \cap F_{3,j+1}] \\
&\leq \Pr[G_j] + \Pr[F_{3,j+1}]\,.
\end{aligned}$$

Since the events $F_{3,j+1}$ and G_j are not independent, we do not know $\Pr[G_j \cap F_{j+1}]$, but we do not need to know it, since we only need an upper bound on the probability of G_{j+1}. By induction, this shows that

$$\begin{aligned}\Pr[F_3] &= \Pr[G_{q_d}] \\ &\leq \sum_{j=1}^{q_d} \Pr[F_{3,j+1}] \\ &\leq \sum_{j=1}^{q_d} \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda) \\ &\leq q_d \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)\,.\end{aligned}$$

If F_3 does not occur, then $\mathcal{B}^*$ correctly computes a witness every time it is called in Game 3, and so Game 4 is identical to Game 3. Thus

$$|\Pr[T_4] - \Pr[T_3]| \leq \Pr[F_3] \leq q_d \mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*}(\lambda)\,.$$

Game 5: We choose the public key at random.

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{s \xleftarrow{\mathrm{R}} S}$
$pk \leftarrow s$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow \varepsilon$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\mathrm{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    $C_{n_d} \leftarrow C$
    Parse C as (x_{n_d}, χ)
    if there exists w such that $(x_{n_d}, w) \in R$ **then**
      $w_{n_d} \leftarrow w$
    else

return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$b \leftarrow D(aux)$

By the statistical Set-Sim property of S, $\epsilon_1 = \Delta[(1^\lambda, \Lambda, s), (1^\lambda, \Lambda, s')]$ is negligible. Thus $|\Pr[T_5] - \Pr[T_4]| \leq \epsilon_1(\lambda)$ by Lemma 1.2.1. Note that because we are relying on a perfect witness extractor, we cannot assume the algorithm is polynomial time, so computational Set-Sim would not suffice for this transition.

Game 6: We compute the public key using KeyGen once more:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$\boxed{(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)}$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
$n_r \leftarrow n_r + 1$
$b \xleftarrow{R} \{0, 1\}$
Append b to Rlist
return $\mathsf{Rlist}[n_r]$
if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
$n_d \leftarrow n_d + 1$
$C_{n_d} \leftarrow C$
Parse C as (x_{n_d}, χ)
if there exists w such that $(x_{n_d}, w) \in R$ **then**
$w_{n_d} \leftarrow w$
if $(x_{n_d}, w_{n_d}) \notin R$ **then**
return $\perp$
$\pi \leftarrow \mathsf{Public}(s, x_{n_d}, w_{n_d})$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$b \leftarrow D(aux)$

By assumption that s is uniformly distributed on S_Λ, the distribution of pk is unchanged, so $\Pr[T_6] = \Pr[T_5]$

Game 7: We use $\mathsf{Private}$ instead of Public to answer decryption queries:

$\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$\mathsf{state}_{\mathcal{A}^*} \leftarrow \varepsilon$
Run $aux \leftarrow \mathcal{A}^{\mathsf{Decrypt},\mathsf{Randomness}}(\Lambda, pk)$
  if $\mathcal{A}$ queries $\mathsf{Randomness}$ **then**
    $n_r \leftarrow n_r + 1$
    $b \xleftarrow{\mathrm{R}} \{0,1\}$
    Append b to Rlist
    return $\mathsf{Rlist}[n_r]$
  if $\mathcal{A}$ queries $\mathsf{Decrypt}(C)$ **then**
    $n_d \leftarrow n_d + 1$
    $C_{n_d} \leftarrow C$
    Parse pk as (Λ, s)
    Parse C as (x_{n_d}, χ_{n_d})
    $\boxed{\pi \leftarrow \mathsf{Private}(sk, x_{n_d})}$
    $\kappa \leftarrow \mathsf{Hash}(\pi)$
    $m \leftarrow \mathsf{dec}(\kappa, \chi)$
    return m
$b \leftarrow D(aux)$

If $x_j \in L$, then there exists $w_j \in W$ such that $(x_j, w_j) \in R$ by definition, and this w_j is used in Game 7 to answer decryption queries. In particular this means that $\mathsf{Public}(s, x_j, w_j) = H_{sk}(x_j) = \mathsf{Private}(sk, x_j)$, so the value returned by the decryption oracle is unchanged in Game 7.

If $x_j \notin L$, then the decrypt oracle will return $\perp$ in Game 6. However, in Game 7, $\pi_j = \mathsf{Private}(sk, x_j)$ is uniformly distributed on Π by the $1/|\Pi|$-universal property. By the δ-smoothness of Hash, the ϵ-rejection security of Σ and Lemma 1.5.2, $\Pr[\mathsf{dec}(\kappa_j, \chi) \neq \perp] \leq \epsilon(\lambda) + \delta(\lambda)$. Since there are at most q_d decryption queries, it follows that

$$|\Pr[T_7] - \Pr[T_6]| \leq q_d(\epsilon(\lambda) + \delta(\lambda)) .$$

Finally, we note that Game 7 is exactly $\mathbf{Expt}^{\mathrm{PA1+Real}}_{\mathbf{KD},\mathcal{A},D}(\lambda)$. Thus

$$\begin{aligned}\mathbf{Adv}^{\mathrm{PA1+}}_{\mathbf{KD},\mathcal{A},\mathcal{A}^*}(\lambda) &= |\Pr[T_7] - \Pr[T_0]| \\ &\leq 2\epsilon_{\mathbf{S}}(\lambda) + \epsilon_r(\lambda) + q_d\mathbf{Adv}^{\mathrm{SWK}}_{\mathbf{M},\mathcal{B},\mathcal{B}^*} + q_d(\epsilon(\lambda) + \delta(\lambda))\end{aligned}$$

is negligible in λ as required. □

5.3 Kurosawa-Desmedt is Simulatable

Theorem 5.3.1. *Assume that $((X_\Lambda)_{\Lambda\in I_\lambda})_{\lambda\in\mathbb{N}}$ is a computationally simulatable family of sets, with simulator $\mathbf{f}_X = (f_X, f_X^{-1}, n_X)$, that the symmetric encryption scheme Σ is ϵ-rejection secure for some negligible ϵ, and simulatable with simulator $\mathbf{f}_\Sigma = (f_\Sigma, f_\Sigma^{-1}, n_\Sigma)$, that the subset membership problem $\mathbf{M}$ is hard, that* Hash *is δ-smooth for some negligible δ and that $\mathbf{H}$ is $(1/|\Pi|)$-universal$_2$. Then the Kurosawa-Desmedt encryption scheme based on the universal hash proof system $\mathbf{H}$ is simulatable.*

Like the proof of PA1 plaintext awareness, the proof that Kurosawa-Desmedt is simulatable is similar to the proof of Theorem 4.3.1, the analogous theorem for the Cramer-Shoup encryption scheme. The differences slightly more pronounced in this case though, in particular we must take care to answer decryption queries for ciphertexts of the form (x^*, χ), using the challenge symmetric encryption key κ^*, where (x^*, χ^*) is the challenge ciphertext. This proof also inherits the requirement that $\mathbf{H}$ is $(1/|\Pi|)$-universal$_2$ from the original proof that the Kurosawa-Desmedt encryption scheme is IND-CCA2 secure [22]. This requirement is needed to ensure the key κ for the DEM component is distributed uniformly at random.

Proof. We construct a simulator $\mathbf{f_{KD}} = (f_{\mathbf{KD}}, f_{\mathbf{KD}}^{-1}, n_{\mathbf{KD}})$ for the Kurosawa-Desmedt scheme, where $n_{\mathbf{KD}}(\lambda) = n_X(\lambda) + n_\Sigma(\lambda)$ for all $\lambda \in \mathbb{N}$, and $f_{\mathbf{KD}}$, $f_{\mathbf{KD}}^{-1}$ are as follows:

function $f_{\mathbf{KD}}(1^\lambda, pk, r)$
  Parse r as $r_1||r_2$ where $|r_1| = n_X(\lambda)$ and $|r_2| = n_\Sigma(\lambda)$
  $x \leftarrow f_X(1^\lambda, \Lambda, r_1)$
  $\chi \leftarrow f_\Sigma(1^\lambda, r_2)$
  return (x, χ)

function $f_{\mathbf{KD}}^{-1}(1^\lambda, pk, C)$
  Parse C as (x, χ)
  $r_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, x)$
  $r_2 \leftarrow f_\Sigma^{-1}(1^\lambda, \chi)$
  return $r_1||r_2$

Since the ciphertext space of **KD**, $\mathbf{CiphSp_{KD}}(pk) = X_\Lambda \times \mathbf{CiphSp}_\Sigma(\lambda)$, the following required properties hold as a direct consequence of the corresponding properties of f_X and f_Σ:

- $f_{\mathbf{KD}}$ is a deterministic algorithm which takes as input $(1^\lambda, pk, r)$ where pk is a public key and $r \in \{0,1\}^{n_{\mathbf{KD}}(\lambda)}$, and outputs an element $C \in \mathbf{CiphSp}(pk)$, the ciphertext space.

- $f_{\mathbf{KD}}^{-1}$ is a probabilistic algorithm which takes as input $(1^\lambda, pk, C)$ where pk is a public key and $C \in \mathbf{CiphSp}(pk)$, and outputs a string $r \in \{0,1\}^{n_{\mathbf{KD}}(\lambda)}$.

- For all public keys pk, and for all $C \in \mathbf{CiphSp}(pk)$, $f_{\mathbf{KD}}(pk, f_{\mathbf{KD}}^{-1}(pk, C)) = C$.

We must now show that for all probabilistic, polynomial-time algorithms $\mathcal{A}$, the Rand-Sim advantage of $\mathcal{A}$ against $f_{\mathbf{KD}}$ is negligible. Let $\mathcal{A}$ be an arbitrary probabilistic, polynomial-time algorithm.

We will now show that $\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{KD},\mathcal{A},\mathbf{f_{KD}}}(\lambda) \leq \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{X,\mathcal{B},f_X}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Sigma,\mathcal{C},\mathbf{f}_\Sigma}(\lambda)$ for some polynomial time adversaries $\mathcal{B}$ and $\mathcal{C}$. The proof is structured as a sequence of games. We let T_i be the event that $b' = 1$ in game i.

Game 0: Let Game 0 be the experiment $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{\mathbf{KD},\mathcal{A},\mathbf{f_{KD}}}(\lambda)$. Written out in full it is as follows:

$\Lambda \xleftarrow{\text{R}} I_\lambda$
$r_1 \xleftarrow{\text{R}} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{\text{R}} \{0,1\}^{n_\Sigma(\lambda)}$
$b' \leftarrow \mathcal{A}(1^\lambda, \Lambda, r_1 || r_2)$
return b

Note that we have written r_1 and r_2 separately, but this is simply a change of notation, since $n_{\mathbf{KD}}(\lambda) = n_X(\lambda) + n_\Sigma(\lambda)$.

Game 1: Let Game 1 be as follows:

$\Lambda \xleftarrow{\text{R}} I_\lambda$
$r_1 \xleftarrow{\text{R}} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{\text{R}} \{0,1\}^{n_\Sigma(\lambda)}$
$r_1' \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$b' \leftarrow \mathcal{A}(1^\lambda, \Lambda, r_1' || r_2)$
return b

We construct an adversary $\mathcal{B}$ against the Rand-Sim property of $((X_\Lambda)_{\Lambda \in I_\lambda})_{\lambda \in \mathbb{N}}$ as follows:

function $\mathcal{B}(1^\lambda, \Lambda, r_1)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $r_2 \xleftarrow{\text{R}} \{0,1\}^{n_\Sigma(\lambda)}$
  $b \leftarrow \mathcal{A}(1^\lambda, pk, r_1 || r_2)$
  return b

We now see that Game 0 is simply $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{X,\mathcal{B},f_X}(\lambda)$ while Game 1 is $\mathbf{Expt}^{\text{Rand}-\text{Sim}-1}_{X,\mathcal{B},f_X}(\lambda)$. Thus

$$|\Pr[T_1] - \Pr[T_0]| = \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{X,\mathcal{B},f_X}(\lambda)\,.$$

Game 2: Let Game 2 be $\mathbf{Expt}^{\text{Rand}-\text{Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$. Written in full it is as follows:

$\Lambda \xleftarrow{\text{R}} I_\lambda$
$r_1 \xleftarrow{\text{R}} \{0,1\}^{n_X(\lambda)}$
$r_2 \xleftarrow{\text{R}} \{0,1\}^{n_\Sigma(\lambda)}$
$r'_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
$r'_2 \leftarrow f_\Sigma^{-1}(1^\lambda, f_\Sigma(1^\lambda, r_2))$
$b' \leftarrow \mathcal{A}(1^\lambda, \Lambda, r'_1||r'_2)$
return b

We construct an adversary $\mathcal{C}$ against the Rand-Sim property of Σ as follows:

function $\mathcal{C}(1^\lambda, r_2)$
  $\Lambda \leftarrow \mathsf{ISA}(1^\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $r_1 \xleftarrow{\text{R}} \{0,1\}^{n_\Sigma(\lambda)}$
  $r'_1 \leftarrow f_X^{-1}(1^\lambda, \Lambda, f_X(1^\lambda, \Lambda, r_1))$
  $b \leftarrow \mathcal{A}(1^\lambda, pk, r'_1||r_2)$
  return b

We now see that Game 1 is simply $\mathbf{Expt}^{\text{Rand}-\text{Sim}-0}_{\Sigma,\mathcal{C},f_\Sigma}(\lambda)$, while Game 2 is $\mathbf{Expt}^{\text{Rand}-\text{Sim}-1}_{\Sigma,\mathcal{C},f_\Sigma}(\lambda)$. Thus

$$|\Pr[T_2] - \Pr[T_1]| = \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Sigma,\mathcal{C},f_\Sigma}(\lambda)\,.$$

Putting it all together we see that

$$\begin{aligned}
\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda) &= |\Pr[T_0] - \Pr[T_2]| \\
&\leq |\Pr[T_0] - \Pr[T_1]| + |\Pr[T_1] - \Pr[T_2]| \\
&= \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{X,\mathcal{B},f_X}(\lambda) + \mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\Sigma,\mathcal{C},f_\Sigma}(\lambda)
\end{aligned}$$

Thus $\mathbf{Adv}^{\text{Rand}-\text{Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$ is negligible in λ as required.

Finally, we must show that for any PKE-Sim adversary $\mathcal{A}$, the advantage

$\mathbf{Adv}^{\text{PKE-Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$ is negligible in λ. This proof is based on the proof that Kurosawa-Desmedt is IND-CCA secure [22]. Let $\mathcal{A}$ be an arbitrary adversary against the PKE-Sim property of **KD**. Let q_d be an upper bound for the number of decryption queries made by $\mathcal{A}$. We prove that the $\mathbf{Adv}^{\text{PKE-Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$ is negligible in λ using a sequence of games. Let T_i be the event that $\mathcal{A}$ outputs 1 in Game i.

Game 0: Let Game 0 be $\mathbf{Expt}^{\text{PKE-Sim-0}}_{\mathbf{KD},\mathcal{A},\mathbf{f_{KD}}}(\lambda)$. Written in full, it is as follows:

$\mathbf{Expt}^{\text{PKE-Sim-0}}_{\mathbf{KD},\mathcal{A},\mathbf{f_{KD}}}(\lambda)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$(x^*, w^*) \leftarrow \mathsf{SSA}(\Lambda)$
$\pi^* \leftarrow \mathsf{Public}_{\mathbf{H}}(pk, x^*, w^*)$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

$\mathcal{A}$ has access to a decryption oracle $\mathsf{Decrypt}$ which takes a ciphertext C as input and returns $\mathsf{Decrypt}(sk, C)$. This oracle is restricted so that $\mathcal{A}$ may not request the decryption of the challenge ciphertext C^*.

Game 1: We modify the challenge ciphertext so that it is generated using the $\mathsf{Private}$ evaluation function for **H** instead of the Public function. Written in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$(x^*, w^*) \leftarrow \mathsf{SSA}(\Lambda)$
$\boxed{\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)}$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

This is a bridging step; $\mathsf{Private}(sk, x^*) = H_{sk}(x^*) = \mathsf{Public}(pk, x^*, w^*)$ since $x^* \in L$, so the challenge ciphertext in Game 1 is equal to the challenge ciphertext in game 0. Thus $\Pr[T_1] = \Pr[T_0]$.

Game 2: This is a transition based on indistinguishability. We choose x^* at random from $X \setminus L$ instead of using the subset sampling algorithm to generate $x^* \in L$. Written out in full, it is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk, \cdot)}(pk)$
$\boxed{x^* \xleftarrow{\mathrm{R}} X \setminus L}$
$\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk, \cdot)}(C^*, \mathsf{state})$
return b'

In this game, x is chosen at random from $X \setminus L$. We do not assume that sampling $x \xleftarrow{\mathrm{R}} X \setminus L$ can be done in polynomial time, since the challenger need not be efficient.

Let $\mathcal{B}$ be the adversary described below:

function $\mathcal{B}(1^\lambda, \Lambda, x^*)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk, \cdot)}(pk)$
  $\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)$
  $\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
  $\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
  $C^* \leftarrow (x^*, \chi^*)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk, \cdot)}(C^*, \mathsf{state})$
  return b'

To respond to a query $\mathsf{Decrypt}(C)$, $\mathcal{B}$ computes $m \leftarrow \mathsf{Decrypt}(sk, C)$ and returns m. If x^* is chosen at random from L, then $\mathcal{B}$ exactly simulates Game

1, but if $x^* \xleftarrow{R} X \setminus L$, $\mathcal{B}$ exactly simulates Game 2. Thus

$$\Pr[\mathcal{B}(1^\lambda, \Lambda, x^*) = 1 : x^* \xleftarrow{R} L] = \Pr[T_1]$$

and

$$\Pr[\mathcal{B}(1^\lambda, \Lambda, x^*) = 1 : x^* \xleftarrow{R} X \setminus L] = \Pr[T_2]\,,$$

so

$$|\Pr[T_2] - \Pr[T_1]| \leq \mathbf{Adv}^{\mathrm{SMP}}_{\mathbf{M},\mathcal{B}}(\lambda)\,.$$

Game 3: This is a bridging step. We move the generation of κ^* to the beginning of the game.

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$\boxed{x^* \xleftarrow{R} X \setminus L}$
$\boxed{\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)}$
$\boxed{\kappa^* \leftarrow \mathsf{Hash}(\pi^*)}$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\,\cdot\,)}(pk)$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\,\cdot\,)}(C^*, \mathsf{state})$
return b'

Since x^*, π^* and κ^* are independent of the view of $\mathcal{A}$, $\Pr[T_3] = \Pr[T_2]$. This step ensures that κ^* is defined when $\mathcal{A}_1$ is run in the next game.

Game 4: We now modify the decryption oracle as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$x^* \xleftarrow{R} X \setminus L$
$\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\,\cdot\,)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    if $\boxed{x = x^* \text{ and } \chi \neq \chi^*}$ **then**
      return $\mathsf{dec}(\kappa^*, \chi)$

else
$\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk, \cdot)}(C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Respond as in $\mathcal{A}_1$
return b'

This is a bridging step, since if $x = x^*$, then $\kappa^* = \mathsf{Hash}(\mathsf{Private}_{\mathbf{H}}(sk, x^*))$ by definition, so the result is the same as in Game 2. Thus $\Pr[T_4] = \Pr[T_3]$.

Game 5: We modify the decryption oracle again so that if $\mathcal{A}$ submits a ciphertext $C = (x, \chi)$ such that $x \neq x^*$ and $x \in X \setminus L$, the decryption oracle returns $\perp$, i.e.

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$x^* \xleftarrow{\mathrm{R}} X \setminus L$
$\pi^* \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x^*)$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk, \cdot)}(pk)$
if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
Parse C as (x, χ)
if $x = x^*$ and $\chi \neq \chi^*$ **then**
return $\mathsf{dec}(\kappa^*, \chi)$
else if $\boxed{x \notin L}$ **then**
$\boxed{\textbf{return } \perp}$
else
$\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk, \cdot)}(C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Respond as in $\mathcal{A}_1$

return b'

Note that the check if $x \notin L$ cannot be implemented in polynomial time, but this is not necessary as the simulator need not be efficient.

Let F_5 be the event that a ciphertext is rejected by the decryption oracle in Game 5 that would not be rejected in Game 4. We will now show that $\Pr[F_5]$ is negligible.

Consider the conditional probability space defined by fixing the random coins of the adversary and the public key pk. This determines the message m output by $\mathcal{A}_1$. In addition, fix the values x^* and π^* used in computing the challenge ciphertext. Together, this determines the values of κ^* and χ^*. In this conditional probability space, all values known to the adversary are fixed, but k is uniformly distributed on the set $\{sk \in K | \alpha(sk) = pk\}$.

Now, suppose $\mathcal{A}$ makes a query $\mathsf{Decrypt}(x, \chi)$. Then by the $1/|\Pi|$-universal$_2$ property of $\mathbf{P}$, for all $\pi \in \Pi$,

$$\Pr[H_{sk}(x) = \pi] \leq 1/|\Pi|]$$

where this probability is taken over the choice of sk subject to the constraints above. In other words, π is uniformly distributed on Π. Since Hash is δ smooth, and Σ is ϵ-rejection secure, it follows by Lemma 1.5.2 that $\Pr[\mathsf{dec}(\kappa, \chi) \neq \perp] \leq \epsilon(\lambda) + \delta(\lambda)$.

Thus

$$|\Pr[T_5] - \Pr[T_4]| \leq q_d(\epsilon(\lambda) + \delta(\lambda)) .$$

Game 6: We modify the challenge ciphertext so that it uses π^* chosen at random. Written in full, Game 6 is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$

$x^* \xleftarrow{R} X \setminus L$
$\boxed{\pi^* \xleftarrow{R} \Pi}$
$\kappa^* \leftarrow \mathsf{Hash}(\pi^*)$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as (x, χ)
    if $x = x^*$ and $\chi \neq \chi^*$ **then**
      return $\mathsf{dec}(\kappa^*, \chi)$
    else if $x \notin L$ **then**
      return $\perp$
    else
      $\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
      $\kappa \leftarrow \mathsf{Hash}(\pi)$
      $m \leftarrow \mathsf{dec}(\kappa, \chi)$
      return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Respond as in $\mathcal{A}_1$
return b'

In Game 6, the joint distribution of (sk, x^*, π^*) is changed, but since $\mathcal{A}$ is not given sk, and the inputs to $\mathcal{A}$ and the responses to its queries are functions of the variables pk, x^* and π^*, it suffices to show that the joint distribution of (pk, x^*, π^*) is unchanged, by Lemma 1.1.3. Let $(\hat{pk}, \hat{x^*}, \hat{\pi^*})$ be an arbitrary element of $S \times X \setminus L \times \Pi$. Then in Game 5,

$$\begin{aligned} &\Pr[H_{sk}(x^*) = \hat{\pi^*} \wedge \alpha(sk) = \hat{pk} \wedge x^* = \hat{x^*}] \\ &= \Pr[H_{sk}(x^*) = \hat{\pi^*} \wedge \alpha(sk) = \hat{pk}] \Pr[x^* = \hat{x^*}] \end{aligned}$$

since x^* is chosen independently. But by the $1/|\Pi|$-universal$_2$ property we see that this equals

$$1/|\Pi| \Pr[\alpha(sk) = \hat{pk}] \Pr[x^* = \hat{x^*}] .$$

In particular, this shows that π^* is independent of pk and x^*. In Game 6, x^*, π^* and pk are explicitly chosen independently from one another. Thus the view of $\mathcal{A}$ is unchanged in Game 6. So

$$\Pr[T_6] = \Pr[T_5]\,.$$

Game 7: We modify the challenge ciphertext so that κ^* is chosen at random. Since this leaves π^* unused, we remove it from the game. Written in full, Game 7 as is follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$x^* \xleftarrow{\mathrm{R}} X \setminus L$
$\boxed{\kappa^* \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}}$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as (x, χ)
    if $x = x^*$ and $\chi \neq \chi^*$ **then**
      return $\mathsf{dec}(\kappa^*, \chi)$
    else if $x \notin L$ **then**
      return $\perp$
    else
      $\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
      $\kappa \leftarrow \mathsf{Hash}(\pi)$
      $m \leftarrow \mathsf{dec}(\kappa, \chi)$
      return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Respond as in $\mathcal{A}_1$
return b'

Let $\kappa_7 \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$, $\pi \xleftarrow{\mathrm{R}} \Pi$ and $\kappa_6 \leftarrow \mathsf{Hash}(\pi)$. By the δ-smoothness of Hash, we see that $\Delta(\kappa_1, \kappa_2) \leq \delta(\lambda)$. Note that $\mathcal{A}$ does not receive κ^* at any point, but it does receive χ^* which depends on κ^*. We may view the entire

game as an algorithm which takes κ_7 as one of its inputs in Game 7, but receives instead in κ_6 in Game 6. By Lemma 1.2.1, it follows that

$$|\Pr[T_7] - \Pr[T_6]| \leq \delta(\lambda)\,.$$

Game 8: We modify the decryption oracle once more so that it no longer checks if $x \in L$, i.e. it runs as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$x^* \xleftarrow{\mathrm{R}} X \setminus L$
$\kappa^* \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\,\cdot\,)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as (x, χ)
    if $x = x^*$ and $\chi \neq \chi^*$ **then**
      return $\mathsf{dec}(\kappa^*, \chi)$
    else
      $\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
      $\kappa \leftarrow \mathsf{Hash}(\pi)$
      $m \leftarrow \mathsf{dec}(\kappa, \chi)$
      return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\,\cdot\,)}(C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Respond as in $\mathcal{A}_1$
return b'

This means it is computable in polynomial time once more. By the same logic as in Game 4,

$$|\Pr[T_8] - \Pr[T_7]| \leq q_d(\epsilon(\lambda) + \delta(\lambda))\,.$$

Game 9: We modify the challenge ciphertext so that it chooses x^* at random from the whole of X instead of $X \setminus L$, i.e:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$\boxed{x^* \xleftarrow{\mathrm{R}} X}$
$\kappa^* \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\,\cdot\,)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as (x, χ)
    if $x = x^*$ and $\chi \neq \chi^*$ **then**
      return $\mathsf{dec}(\kappa^*, \chi)$
    else
      $\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$
      $\kappa \leftarrow \mathsf{Hash}(\pi)$
      $m \leftarrow \mathsf{dec}(\kappa, \chi)$
      return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\,\cdot\,)}(C^*, \mathsf{state})$
  if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
    Respond as in $\mathcal{A}_1$
return b'

Let $\mathcal{C}$ be the algorithm which is identical to Game 9, except that it takes an element x^* as an input instead of choosing x^* for itself, in the same manner as the algorithm $\mathcal{B}$ described in Game 2.

By Lemma 3.1.8 and Lemma 1.2.1 this implies that

$$|\Pr[T_9] - \Pr[T_8]| \leq \frac{|L|}{|X|}\mathbf{Adv}^{\mathrm{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda)\,.$$

Game 10: We must return the decryption oracle to its original state. Written out in full, the game is as follows:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$x^* \xleftarrow{\mathrm{R}} X$
$\kappa^* \xleftarrow{\mathrm{R}} \{0,1\}^{\ell(\lambda)}$
Run $(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\,\cdot\,)}(pk)$
  if $\mathcal{A}_1$ queries $\mathsf{Decrypt}(C)$ **then**
    Parse C as (x, χ)
    $\pi \leftarrow \mathsf{Private}_{\mathbf{H}}(sk, x)$

$\kappa \leftarrow \mathsf{Hash}(\pi)$
$m \leftarrow \mathsf{dec}(\kappa, \chi)$
return m
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
Run $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
if $\mathcal{A}_2$ queries $\mathsf{Decrypt}(C)$ **then**
Respond as in $\mathcal{A}_1$
return b'

Since κ^* is randomly chosen, any ciphertext of the form (x^*, χ) will decrypt to $\perp$ with probability $1 - \epsilon(\lambda)$ by the ϵ-rejection property of the DEM.

Hence,

$$|\Pr[T_{10}] - \Pr[T_9]| \leq q_d \epsilon(\lambda)\,.$$

Game 11: We choose x^* using the simulation function f_X:

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$\boxed{r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}}$
$\boxed{x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)}$
$\kappa^* \xleftarrow{R} \{0,1\}^{\ell(\lambda)}$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

We construct an adversary $\mathcal{D}$ against the Set-Sim property of (X_Λ) as follows:

function $\mathcal{D}(1^\lambda, \Lambda, x)$
$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$x^* \leftarrow x$
$\chi^* \leftarrow \mathsf{enc}(\kappa^*, m)$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

If $\mathcal{D}$ is given $x \xleftarrow{R} X_\Lambda$ as input, then it exactly simulates Game 10, while if it is given $x \leftarrow f_X(1^\lambda, \Lambda, r_1)$ as input, it exactly simulates Game 11. Thus

$$|\Pr[T_{11}] - \Pr[T_{10}]| = \mathbf{Adv}^{\mathrm{Set-Sim}}_{X,\mathcal{D},f_X}(\lambda)\,.$$

Game 12: We choose χ^* using the simulation function f_Σ

$(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
$(m, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
$r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
$\boxed{r_2 \xleftarrow{R} \{0,1\}^{n_\Sigma(\lambda)}}$
$x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
$\boxed{\chi^* \leftarrow f_X(1^\lambda, r_2)}$
$C^* \leftarrow (x^*, \chi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
return b'

We construct an adversary $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$ against the DEM-Sim property of Σ as follows:

function $\mathcal{E}_1(1^\lambda)$
  $(pk, sk) \leftarrow \mathsf{KeyGen}(\Lambda)$
  $(m, \mathsf{state}_\mathcal{A}) \leftarrow \mathcal{A}_1^{\mathsf{Decrypt}(sk,\cdot)}(pk)$
  $\mathsf{state}_\mathcal{E} \leftarrow (pk, sk, \mathsf{state}_\mathcal{A})$
  return $(m, \mathsf{state}_\mathcal{E})$

function $\mathcal{E}_2(1^\lambda, \chi, \mathsf{state}_\mathcal{E})$
  Parse $\mathsf{state}_\mathcal{E}$ as $(pk, sk, \mathsf{state}_\mathcal{A})$
  $r_1 \xleftarrow{R} \{0,1\}^{n_X(\lambda)}$
  $x^* \leftarrow f_X(1^\lambda, \Lambda, r_1)$
  $\chi^* \leftarrow \chi$
  $C^* \leftarrow (x^*, \chi^*)$
  $b' \leftarrow \mathcal{A}_2^{\mathsf{Decrypt}(sk,\cdot)}(C^*, \mathsf{state})$
  return b'

If $\mathcal{E}_2$ is given $\chi \leftarrow \mathsf{enc}(\kappa^*, m)$ as input, then it exactly simulates Game 10, while if it is given $x \leftarrow f_\Sigma(1^\lambda, r_2)$ as input, it exactly simulates Game 11. Thus

$$|\Pr[T_{12}] - \Pr[T_{11}]| = \mathbf{Adv}^{\mathrm{Set-Sim}}_{X,\mathcal{D},f_X}(\lambda)\,.$$

We now see that Game 12 is exactly $\mathbf{Expt}^{\text{PKE}-\text{SIM}-1}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda)$. So

$$\mathbf{Adv}^{\text{PKE}-\text{Sim}}_{\mathbf{KD},\mathcal{A},f_{\mathbf{KD}}}(\lambda) = |\Pr[T_{11}] - \Pr[T_0]|\,.$$

Putting it all together, we get

$$\begin{aligned}
|\Pr[T_{11}] - \Pr[T_0]| &\leq |\Pr[T_{11}] - \Pr[T_{10}]| + \ldots + |\Pr[T_1] - \Pr[T_0]| \\
&\leq \mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{B}}(\lambda) + q_d(\epsilon(\lambda) + \delta(\lambda)) + \frac{q_d}{|X \setminus L|} + \delta(\lambda) + \\
&\quad q_d(\epsilon(\lambda) + \delta(\lambda)) + \frac{|L|}{|X|}\mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda) + q_d\epsilon(\lambda) + \\
&\quad \mathbf{Adv}^{\text{Set}-\text{Sim}}_{X,\mathcal{D},f_X}(\lambda) + \mathbf{Adv}^{\text{DEM}-\text{Sim}}_{X,\mathcal{E},f_\Sigma}(\lambda) \\
&= q_d(3\epsilon(\lambda) + 2\delta(\lambda) + \frac{1}{|X \setminus L|}) + \delta(\lambda) + \frac{|L|}{|X|}\mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{B}}(\lambda) \\
&\quad + \mathbf{Adv}^{\text{SMP}}_{\mathbf{M},\mathcal{C}}(\lambda) + \mathbf{Adv}^{\text{Set}-\text{Sim}}_{X,\mathcal{D},f_X}(\lambda) + \mathbf{Adv}^{\text{DEM}-\text{Sim}}_{X,\mathcal{E},f_\Sigma}(\lambda)
\end{aligned}$$

which is negligible in λ as required. □

Finally, we see that the Kurosawa-Desmedt scheme is PA1+ plaintext aware and simulatable, and thus it is PA2 plaintext aware by Theorem 3.1.6.

Chapter 6

Conclusion

In chapter 2 we introduced a new definition of plaintext awareness, which we called 2PA2, which is simpler than the existing PA2 definition. We showed that any encryption scheme which is PA2 plaintext aware must hide the length of a message as well as its contents, but that our definition does not have this limitation. We also showed that for schemes which do hide the length of a message, our definition is equivalent to the standard PA2 definition. We also showed that our definition is in some sense close to minimal, because a simpler definition which omits the plaintext creator entirely, which we called PA2E, is inadequate because it does not suffice to prove the fundamental theorem of plaintext awareness.

In chapters 3 we introduced a new computational assumption related to a subset membership problems, which we called the subset witness knowledge assumption. This is a generalisation of the DHK assumption. We showed that for some subset membership problems our new assumption seems comparable to the DHK assumption, while at least one case, namely the one based on the quadratic residuosity assumption, it is almost certainly false. In chapters 4

and 5, we use Dent's techniques to show that the generalised Cramer-Shoup and Kurosawa-Desmedt encryption schemes are PA2 plaintext aware, in the cases where our new assumptions hold.

6.1 Future Work and Open Problems

As noted in Section 2.2, it may be possible to show that a scheme which is PA2E plaintext aware and LH-IND-CPA secure is LH-IND-RCCA2 secure. This would give us a nicer notion of plaintext awareness at the expense of a weaker and less well-known notion of indistinguishability.

Our techniques cannot be used to prove the plaintext awareness of the quadratic residuosity variants of the Cramer-Shoup and Kurosawa-Desmedt encryption schemes, because the SWK assumption does not appear to hold in this case. It would be interesting to prove that this encryption scheme is actually not plaintext aware, since this will demonstrate the intuitively obvious claim that plaintext awareness is not necessary to achieve IND-CCA2 security.

Aside from the Cramer-Shoup and Kurosawa-Desmedt encryption schemes, the only known efficient encryption schemes which are IND-CCA2 secure in the standard model are based on the CHK construction [8]. It is not yet known whether these schemes are plaintext aware.

Finally, the long standing open question in plaintext awareness is whether full PA2 plaintext awareness can be achieved in the standard model without using extractor assumptions, like the SWK assumption we introduced here. This question remains open.

Bibliography

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[2] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, 1998.

[3] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2004.

[4] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[5] James Birkett and Alexander W. Dent. Constructing plaintext-aware encryption schemes. 2007. Unpublished Manuscript.

[6] Dan Boneh. Simplified oaep for the rsa and rabin functions. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2001.

[7] Jaimee Brown, Juan Manuel González Nieto, and Colin Boyd. Concrete chosen-ciphertext secure encryption from subgroup membership problems. In *CANS*, volume 4301 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2006.

[8] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222, 2004.

[9] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[10] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, 1998.

[11] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[12] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[13] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

[14] Alexander W. Dent. The Cramer-Shoup encryption scheme is plaintext aware in the standard model. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 289–307. Springer, 2006.

[15] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[16] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer, 2001.

[17] D. Galindo and J. L. Villar. An instantiation of the Cramer-Shoup encryption paradigm using bilinear map groups. In *Proc. Workshop on Mathematical Problems and Techniques in Cryptology*, Bellaterra, Spain, 2005.

[18] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[19] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[20] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *STOC*, pages 12–24. ACM, 1989.

[21] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[22] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2004.

[23] James Manger. A chosen ciphertext attack on rsa optimal asymmetric encryption padding (oaep) as standardized in pkcs #1 v2.0. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238. Springer, 2001.

[24] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[25] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[26] Juan Manuel González Nieto, Colin Boyd, and Ed Dawson. A public key cryptosystem based on the subgroup membership problem. In *ICICS*, volume 2229 of *Lecture Notes in Computer Science*, pages 352–363. Springer, 2001.

[27] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.

[28] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[29] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[30] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In *ACM Conference on Computer and Communications Security*, pages 400–409. ACM, 2006.

[31] Victor Shoup. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/112`.

[32] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/2004/332`.

[33] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2005.

[34] Isamu Teranishi and Wakaha Ogata. Relationship between standard model plaintext awareness and message hiding. In *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2006.

[35] Mark N. Wegman and Larry Carter. New classes and applications of hash functions. In *FOCS*, pages 175–182. IEEE, 1979.