Blackbox Construction of A More Than Non-Malleable CCA1 Encryption Scheme from Plaintext Awareness

Steven Myers
Indiana University
samyers@indiana.edu

Mona Sergi
University of Virginia
ms4bf@virginia.edu

abhi shelat
University of Virginia
abhi@virginia.edu
*

Abstract

We construct a Non-Malleable Chosen Ciphertext Attack (NM-CCA1) encryption scheme from any encryption scheme that is also plaintext aware and weakly simulatable. We believe this is the first construction of a NM-CCA1 scheme that follows strictly from encryption schemes with seemingly weaker or incomparable security definitions to NM-CCA1.

Previously, the statistical Plaintext Awareness #1 (PA1) notion was only known to imply CCA1. Our result is therefore novel because unlike the case of Chosen Plaintext Attack (CPA) and Chosen Chiphertext Attack (CCA2), it is unknown whether a CCA1 scheme can be transformed into an NM-CCA1 scheme. Additionally, we show both the Damgård Elgamal Scheme (DEG) [6] and the Cramer-Shoup Lite Scheme (CS-Lite) [5] are weakly simulatable under the DDH assumption. Since both are known to be statistical Plaintext Aware 1 (PA1) under the Diffie-Hellman Knowledge (DHK) assumption, they instantiate our scheme securely.

Furthermore, in response to a question posed by Matsuda and Matsuura [12], we define cNM-CCA1-security in which an NM-CCA1-adversary is permitted to ask a $c \geq 1$ number of parallel queries after

*This work is Sponsored by the NSF under grant 0939718, and under DARPA and AFRL.The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-11-2-0211.

receiving the challenge ciphertext. We extend our construction to yield a cNM-CCA1 scheme for any constant c. All of our constructions are black-box.

Keywords: Public-Key Encryption, Plaintext-Awareness, Non-Malleability.

1 Introduction

Public-key encryption is one of the most commonly used cryptographic primitives in practice and theory. However, our community's understanding of its different security definitions and their relationships is still poor. Goldwasser and Micali [11] formalized the notion of computational security against passive eavesdroppers through the concept of semantic security or chosen plaintext (CPA) security. However, security against passive eavesdroppers is too weak to be used in modern applications, and thus stronger notions of security have been proposed and studied.

Naor and Yung [15] introduced the first strengthening of security by considering adversaries who have the ability to decrypt messages of their choice. In their notion, called Chosen Ciphertext Attacks #1 (CCA1), the adversary is not allowed to decrypt ciphertexts related to the ciphertext of interest. Later, a more comprehensive notion of adversarial decryption introduced by Simon and Rackoff [17] and termed CCA2 security became the gold standard requirement for decryption on the Internet. While it is clear that stronger security notions imply weaker ones, and thus CCA2-secure schemes imply CCA1 secure ones which in turn imply CPA secure ones, the converse directions are not known to be true. While it is the case that a CPA (resp. CCA1) secure encryption scheme need not be CCA1 (resp. CCA2) secure, it is not known if the existence of a CPA (resp. CCA1) secure scheme *implies* the existence of a CCA1 (resp. CCA2) scheme. These are considered some of the major open questions in cryptography.

The CPA and CCA1 security notions for encryption suffer another weakness which must also be addressed for public key encryption to function in modern settings. In particular, the CPA and CCA1 security definitions do not prevent an adversary who observes an encryption of the message m from producing an encryption of the message $f(m)$ for some function f (even though the value m remains private). The seminal work of Dolev, Dwork, and Naor [10] addressed this security issue by introducing the notion of non-malleable cryptographic primitives such as encryption schemes, commitment schemes, and zero-knowledge. Later, Pass, shelat and Vaikuntanathan [16] strengthened the DDN definition

and presented a construction from CPA to non-malleable CPA encryption using non-blackbox use of the CPA encryption scheme. There have been many follow-up works that propose more efficient constructions of non-malleable primitives. A notable achievement in this line of research has been the construction of non-malleable CPA encryption from standard versions of encryption in a black-box manner [3].

In general, any progress on constructing public-key encryption schemes with stronger security properties from weaker ones is of great interest in furthering our understanding of public key encryption. Beyond theoretical importance, it is of practical value: when new cryptographic assumptions are shown to be sufficient for public-key encryption, it would be valuable to know that they are simultaneously sufficient for the strong forms we need for use in modern settings.

With this in mind, we consider the open question of whether an NM-CCA1 encryption scheme can be constructed from a CCA1 encryption scheme. We present a black-box construction of an NM-CCA1 encryption scheme from a subset of CCA1 encryption schemes which are also plaintext aware under multiple keys and weakly simulatable (we will formally define these concepts later). Intuitively, an encryption scheme is plaintext aware (called **sPA1** in [1]) if the only way that a ppt adversary can produce a valid ciphertext is to apply the (randomized) encryption algorithm to the public-key and a message. Notice that this definition does not imply non-malleability since there is no constraint on what an adversary can do *when given a valid ciphertext*. In fact, both plaintext-aware encryption schemes constructed in [1] are multiplicatively homomorphic, and thus clearly malleable. The weakly simulatable property in our construction is required for technical reasons and roughly corresponds to the ability to sample ciphertexts and pseudo-ciphertexts without knowing any underlying plaintext (if such a plaintext exists).

Note that there exist encryption schemes that satisfy security notions that "sit between" standard notions. One such example from Cramer et al. [4] consists of a black-box construction of a q-bounded CCA2 encryption scheme which is not NM-CPA-secure[1], but which satisfies a stronger security notion than CPA. In particular, as a generalization of NM-CPA, Matsuda and Matsuura [12] put forth the challenge of constructing encryption schemes that can handle more than one parallel query after revealing the challenge ciphertext. They write:

[1]The [4] construction supports only q queries, whereas an NM-CPA adversary can submit more than q ciphertexts in its final parallel query.

> "Since any (unbounded) CCA secure PKE construction from IND-CPA secure ones must first be secure against adversaries who make two or more parallel decryption queries, we believe that overcoming this barrier of *two parallel queries* is worth tackling."

In this spirit, we define an extension over NM-CCA1, cNM-CCA1, in which the adversary can make c adaptive parallel decryption queries after seeing the challenge ciphertext, where each parallel decryption query can request that a polynomial number of ciphertexts (excluding the challenge ciphertext) be decrypted. Thus that NM-CCA1 is cNM-CCA1 where the parameter c is set to be one. Next, we show how to construct a cNM-CCA1 secure encryption scheme for an arbitrary constant c. Unfortunately, the size of the ciphertext in our cNM-CCA1 encryption scheme is multiplicatively polynomially bigger than the size of the ciphertext in a $(c-1)$NM-CCA1 encryption scheme and thus c must be a constant to obtain an efficient construction.

While our initial goal was to construct an NM-CCA1 scheme from a subset of the CCA1-secure schemes, a result by Bellare and Palacio [1] shows that any plain-text aware scheme that is CPA secure is also CCA1-secure, and thus formally all of our results follow from any CPA-secure that also has the necessary plaintext aware and simulatability properties. However, we show that the weak simulatability requirement implies CPA security, and therefore all of our results follow from any scheme which is weakly simulatable and plaintext aware.

About Knowledge Extraction Assumptions Our constructions rely on encryption schemes that are plaintext aware (**sPA1**$_\ell$) in the multi-key setup and are weakly simulatable. In Theorem 5, we show that such encryption schemes exist under a suitable extension of the Diffie-Hellman Knowledge (DHK) assumption that was originally proposed by Damgård, and modified to permit interactive extractors by Bellare and Palacio [1]. Dent [9] has since shown that it is secure in the generic group model. Some critics of the DHK assumption have commented on its strength and observed that it is not efficiently falsifiable [14]. However, it is not our goal to argue whether or not it is an assumption which should be used in deployable systems. Instead we note it is seemingly a weaker assumption than the Random Oracle model, which is known to be incorrect in full generality, cf. [2] and is yet pervasively used in theory and practice. In contradistinction, we are not aware of any general security definitions that are non-

trivially weaker or incomparable to NM-CCA1 yet imply schemes which are NM-CCA1. Similarly, the gap between NM-CCA1 and CCA2 is poorly understood.

Techniques Both our NM-CCA1 and cNM-CCA1 constructions are based on the ideas of the nested encryption construction by Myers and shelat in [13]. We first encrypt the message under one key (we refer to this ciphertext as the *inner layer*), and encrypt the resulting inner layer ciphertext repetitively under an additional k keys, where k is the security parameter (we refer to these k keys as the "outer keys", and the ciphertexts they produce as the "outer layer"). During decryption, all the outer layer ciphertexts are decrypted, and it is verified that they all encode the same inner layer value. This idea is combined with the well-studied notion of non-duplicatable set selection (in this case of public-keys used to encrypt the outer-layer encryptions), such that anyone attempting to maul a ciphertext has to perform their own independent outer layer encryption. Intuitively, anyone that can encrypt to a consistent outer layer encryption under a new key must have knowledge of the underlying inner-layer.

On a more technical level, there are several challenges that need to be overcome. The technical difficulty in proving weaker public-key encryption security notions imply stronger security notions lies in the simulation of a decryption oracle. When beginning with a $\mathbf{sPA1}_\ell$-secure encryption primitive, we can easily simulate the first phase decryption oracle in the NM-CCA1 security definition by using the plaintext extractor guaranteed by the $\mathbf{sPA1}_\ell$ security definition. However, we cannot simply use the extractor to simulate the decryption oracle *after* the adversary receives the challenge ciphertext in the NM-CCA1 security experiment. This is because the plaintext-aware security definition does not guarantee that an extractor works if the underlying randomness used to create the ciphertext by the challenger is not known to the challenger. Generally, an adversary that mauls an input ciphertext may not have access to this underlying randomness. To overcome this problem, we make use of the notion of weak simulatability.

Contributions To summarize, our contribution is twofold. Our work shows the first black-box construction of a non-malleable CCA1 encryption scheme in the standard model from a weaker encryption primitive. Secondly, for the first time, we show how to construct an encryption scheme that is not CCA2 secure but is secure against an adversary that can ask a

bounded number of polynomial-parallel queries after receiving the challenge ciphertext, satisfying a natural extension to the notion of NM-CCA1 security. This might be of independent interest since the development of constructions that satisfy stronger notions than non-malleable CCA1 security but do not satisfy CCA2 security can provide insight into the technical difficulties with understanding the relationship between CCA1 and CCA2. For example, prior to this work the authors did not believe it was clear that such schemes existed. At least one of the authors felt it was plausible that being able to provide multiple parallel queries after access to a challenge ciphertext was equivalent to providing an arbitrary polynomial number of parallel queries.

Finally, we note that none of our constructions

2 Notations and Definitions

We use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We say a function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if for all polynomials p and all sufficiently large $n : \mu(n) \leq 1/p(n)$. Given two families of distributions $D_0 = \{D_{0,i}\}_{i\in\mathbb{N}}$ and $D_1 = \{D_{1,i}\}_{i\in\mathbb{N}}$, we denote that they are computationally indistinguishable by writing $D_0 \approx_c D_1$.

We use the standard definition for CPA/CCA1/CCA2 security, and a definition of non-malleability for CCA1 encryption schemes based on the non-malleability definition for CPA encryption schemes in [16]. In the NM-CCA1 game, the adversary is allowed to ask an unbounded number of decryption queries before seeing the challenge ciphertext, and one parallel query afterwards. A parallel decryption query is one that consists of unbounded number of ciphertexts, none of which will be decrypted until all the ciphertexts in the query are submitted.

To generalize NM-CCA1 security, we can define cNM-CCA1 security identically to NM-CCA1 except that the adversary can make $c \geq 1$ parallel queries after seeing the challenge ciphertext. For example, in the CCA2 security definition, the adversary may ask an unbounded number of queries before and after seeing the challenge ciphertext; thus, cNM-CCA1 is an intermediate notion that we study to understand public key encryption.

Definition 1 (cNM-CCA1)**.** *For an integer $c \geq 0$, we say that a scheme $\Pi = (\mathsf{nmg}, \mathsf{nme}, \mathsf{nmd})$ is* cNM-CCA1 *or* (c)NME *secure if for all ppt adversaries and distinguishers $\mathcal{A} = (\mathcal{A}_0, ..., \mathcal{A}_c)$ and $\mathcal{D}$ respectively and for all polynomials p, we*

have that

$$\left\{(\mathsf{c})\mathsf{NME}_0(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \approx_c \left\{(\mathsf{c})\mathsf{NME}_1(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k$$

where experiment (c)NME *is defined in Fig. 1.*

$(\mathsf{c})\mathsf{NME}_b(\Pi, \mathcal{A}, \mathcal{D}, k, p(k))$

1: $(\text{CNPK}, \text{CNSK}) \leftarrow \mathsf{nmg}(1^k)$
2: $(m_0, m_1, S_1) \leftarrow A_0^{\mathsf{nmd}(\text{CNSK},.)}(\text{CNPK})$ s.t. $|m_0| = |m_1|$
3: $y^* \leftarrow \mathsf{nme}(\text{CNPK}, m_b)$
4: $\vec{d}_1 \leftarrow \perp$
5: *for* $i = 1$ *to* c
6: $(\vec{y}, S_{i+1}) \leftarrow \mathcal{A}_i(y^*, S_i, \vec{d}_i)$ where $|\vec{y}| = p(k)$ **a: why do we need the $p(k)$ here?**
7: $\forall j \in [|\vec{y}|]$, $d_{i+1,j} \leftarrow \mathsf{nmd}(\text{CNSK}, y_j)$ if $y_j \neq y^*$ and $\perp$ otherwise
8: Output $\mathcal{D}(y, \vec{d}_{c+1}, S_{c+1})$

Fig. 1: The (c)NME Experiment For $c \geq 0$. An Adversary $\mathcal{A}$ gets c parallel queries to a decryption oracle.

2.1 Weakly Simulatable Encryption Scheme

Dent [8] introduced the notion of simulatability for an encryption scheme. Intuitively, an encryption scheme is simulatable if no attacker can distinguish valid ciphertexts from some family of pseudo-ciphertexts (which will include both valid encryptions and invalid encryptions). This family of pseudo-ciphertexts must be efficiently and publicly sampleable and somewhat invertible (given any pseudo-ciphertext, one can find a random looking string that generates it). In Dent's definition, a distinguisher is given a challenge "ciphertext" (i.e., either a legitimate ciphertext or a pseudo-ciphertext) and must classify it. The distinguisher has access to a decryption oracle to help it distinguish between pseudo-ciphertexts and legitimate ones, but it cannot query the oracle on the challenges that it is trying to distinguish. We introduce a weak notion of simulatability where the attacker is not given access to the decryption oracle.

Definition 2. *(**Weakly Simulatable Encryption Scheme**) An asymmetric encryption scheme* $\Pi = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ *is weakly simulatable if there exist two*

poly-time algorithms (f, f^{-1}), where f is deterministic and f^{-1} is probabilistic, such that for all $k \in \mathbb{N}$ there exists the polynomial function p where $l = p(k)$, and the following correctness properties hold for every pk in the range of gen*:*

1. *For each $r \in \{0,1\}^l$, assign $c \leftarrow f(pk, r)$ where $c \in \mathcal{C}$. The set $\mathcal{C}$ is the set of all "possible-ciphertext" strings that can be submitted to the decryption oracle (notice that members of $\mathcal{C}$ are both valid and invalid ciphertexts).*

2. *For each $c \in \mathcal{C}$, $f^{-1}(pk, c) \in \{0,1\}^l$.*

3. *For each $c \in \mathcal{C}$, $f(pk, f^{-1}(pk, c)) = c$.*

4. *For every efficient distinguisher $\mathcal{A}$ and all k, $|\Pr[\mathsf{DIST}_\Pi((f, f^{-1}), k, \mathcal{A}) = 1] - 1/2| \leq \mu(k)$, where μ is some negligible function and the* DIST *experiment is defined as follows:*

   $\mathsf{DIST}_\Pi(k, (f, f^{-1}), \mathcal{A})$
   1: $(pk, sk) \leftarrow \mathsf{gen}(1^k)$
   2: $(m, \sigma) \leftarrow \mathcal{A}(pk)$, *where σ is state.*
   3: $b \leftarrow \{0,1\}$, $r \leftarrow \{0,1\}^l$, $c \leftarrow \mathsf{enc}_{pk}(m)$
   4: *if $b = 0$, then* $p = (r, f(pk, r))$
   5: *else* $p = (f^{-1}(pk, c), c)$.
   6: $b' \leftarrow \mathcal{A}(\sigma, p)$.
   7: *Output 1 if $b = b'$.*

When valid ciphertexts cannot be distinguished from pseudo-ciphertexts that need not encode messages, CPA security is immediate. The converse need not hold because ciphertexts might be hard to generate and invalid ciphertexts might be easily distinguishable from illegitimate ones (for example, they might contain a zero-knowledge proof of validity). Notice that the weak simulatability notion is not equivalent to the Invertible Sampling notion introduced in [7] since in this definition the plaintext is not needed to compute the pseudo-random string that generates the ciphertext.

Theorem 1. *If **E** is a weakly simulatable encryption scheme, then **E** is* CPA *secure.*

Proof. Let **E** be weakly simulatable using the efficiently computable functions (f, f^{-1}). Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a CPA adversary, that breaks the CPA security of **E** with advantage ϵ. We will show that if **E** is a weakly simulatable encryption scheme, then ϵ should be negligible.

In Fig. 2 we present a distinguisher $\mathcal{B}_\mathcal{A} = (\mathcal{B}_{\mathcal{A},1}, \mathcal{B}_{\mathcal{A},2})$ that employs $\mathcal{A}$ internally and tries to break **E**'s weak simulatability property. We analyze

$\mathcal{B}_{\mathcal{A},1}(pk)$
1: $(m_0, m_1, \sigma) \leftarrow \mathcal{A}_1(pk)$
2: $d \leftarrow \{0,1\}$
3: $Output(m_d, \sigma' = (\sigma, d))$
$\mathcal{B}_{\mathcal{A},2}(\sigma' = (\sigma, d), (r, c))$
1: $d' \leftarrow \mathcal{A}_2(\sigma, c)$
2: If $d = d'$ output 0, otherwise output 1

Fig. 2: The distinguisher $\mathcal{B}_{\mathcal{A}}$ used in the DIST experiment to break the weak simulatable security of **E**.

the advantage of $\mathcal{B}_{\mathcal{A}}$ assuming $\mathcal{A}$ has advantage ϵ in breaking the CPA security of **E**.
Assuming that $\mathcal{A}$ has advance ϵ in breaking the CPA security of **E** implies that:

$$\Pr_{\mathsf{DIST}_{\mathbf{E}}(k,(f,f^{-1}),\mathcal{B}_{\mathcal{A}})}[d' = d \mid b = 0] > 1/2 + \epsilon(k) \tag{1}$$

Notice that $\Pr[d' = d|b = 0]$ is the probability that $\mathcal{A}$ guesses the encrypted message correctly when it is given a valid ciphertext. Also **E** is weakly simulatable if and only if for some negligible function ϵ'

$$|\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| < \epsilon'(k) \tag{2}$$

Note that by definition in $\mathsf{DIST}_{\mathbf{E}}(k, (f, f^{-1}), \mathcal{B}_{\mathcal{A}})$, $b' = 0$ if and only if $d' = d$. Hence $\Pr[b' = 0|b = 0] = \Pr[d = d'|b = 0]$ and $\Pr[b' = 0|b = 1] = \Pr[d = d'|b = 1]$. We have that

$$\Pr[d = d' \mid b = 1] = 1/2 \tag{3}$$

because whenever $b = 1$, the ciphertext c to the distinguisher $\mathcal{B}_{\mathcal{A},2}$ is independent of the bit d, and the distinguisher's probability guessing the random bit d is exactly $1/2$. We have that:

$$\begin{aligned} |\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| &= |\Pr[d = d'|b = 0] - \Pr[d = d'|b = 1] \\ &> 1/2 + \epsilon - 1/2 = \epsilon \end{aligned} \tag{4}$$

where the inequality follows from substituting the (in)equalities 1 & 3 respectively. We combine the inequalities 2 and 4:

$$\epsilon < |\Pr[b' = 0|b = 0] - \Pr[b' = 0|b = 1]| < \epsilon'(k)$$

$$\rightarrow \epsilon < \epsilon'(k)$$

Since ϵ is at most a negligible value in the security parameter, we conclude that $\mathcal{A}$ has at most negligible advantage in breaking the CPA security of **E**. Hence **E** is CPA secure.

□

Instantiating weak-simulatability Following the ideas of Dent [8], we show in Appendix A how the Damgård ElGamal (DEG) and CS-lite encryption schemes—summarized in Fig. 8 for convenience—can both be weakly simulatable when instantiated in the proper groups.

2.2 Plaintext Awareness For Multiple Key Setup

In Fig. 3 we present a slight generalization to the definition of **sPA1** by [1] in which multiple keys are given to the ciphertext creator and the extractor must be able to decrypt relative to any one of them.

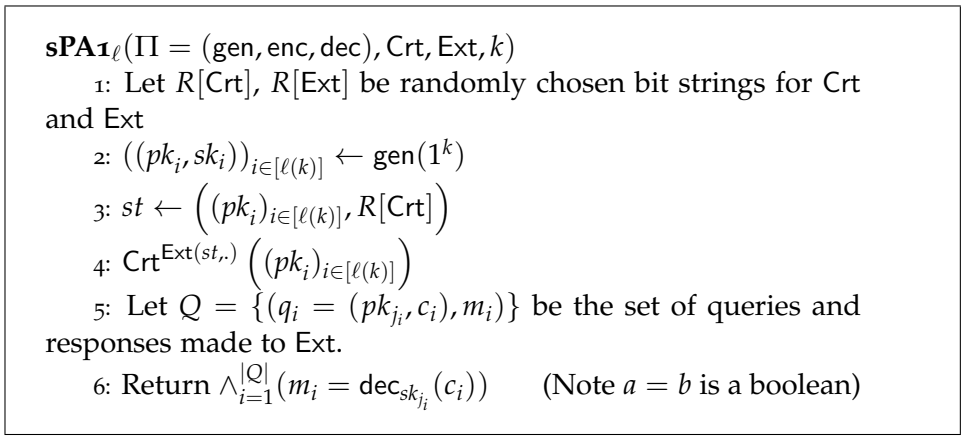

Fig. 3: The $\mathbf{sPA1}_\ell$ Definition

Definition 3 ($\mathbf{sPA1}_\ell$-Security). *A public-key encryption scheme* $\Pi = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ *is said to be* $\boldsymbol{sPA1}_\ell$ *secure, for a polynomial* ℓ*, if for each ppt ciphertext creator* Crt*, there exists a ppt extractor* Ext *and negligible function* μ *s.t. for all* $k \in \mathbb{N}$*:* $\Pr[\boldsymbol{sPA1}_\ell(\Pi, \mathsf{Crt}, \mathsf{Ext}, k) = 0] \leq \mu(k)$*, in which case the* Ext *is deemed* successful *for* Crt*. We define* $\boldsymbol{Adv}^{\boldsymbol{sPA1}_\ell}(\boldsymbol{E}, \mathsf{Crt}, \mathsf{Ext}, k)$ *to be* $\Pr[\boldsymbol{sPA1}_\ell(\boldsymbol{E}, \mathsf{Crt}, \mathsf{Ext}, k) = 0]$.

Notice that the **sPA1** definition is a special case of **sPA1**$_\ell$ where $\ell(k) = 1$. In Fig. 3, Crt is a ciphertext creator. Ext is a stateful ppt algorithm called the *extractor* that takes as input its state information st and a ciphertext given by the ciphertext creator Crt. Ext will return the decryption of that ciphertext and an updated state st. Ext's initial state is set to the public-keys pk_i and Crt's random coins $R[\mathsf{Crt}]$. The state gets updated by Ext as it answers each decryption query that Crt submits.

In Appendix B, we argue that Cramer-Shoup Lite (CS-Lite) and Damgard's ElGammal (DEG), described in Fig. 8, are **sPA1**$_\ell$ secure based on a suitable modification of the Diffie-Hellman Knowledge definition that was originally proposed by Damgård, and then later modified to permit interactive extractors by Bellare and Palacio [1].

2.3 Why sPA1$_\ell$ does not follow from sPA1 security

It may seem that the **sPA1**$_\ell$ definition should follow naturally from **sPA1** by composing extractors. The following toy example highlights the technical difficulty with natural composition. Let (g, e, d) be an **sPA1**-secure primitive, and define a new encryption scheme (g', e', d') which creates two pairs of keys from the original encryption scheme, and chooses one (at random) to use to encrypt during the encryption process. Formally, $g'(k) = (PK = (pk_0, pk_1), SK = (sk_0, sk_1))$, where (pk_b, sk_b) are an output of the bth invocation of $g(k)$. For encryption, $e'(PK, m)$ chooses a random coin $z \in \{0, 1\}$ and outputs $C = (z, e(pk_z, m))$; decryption is $d'(SK, C = (z, c))$ outputs $d(sk_z, c)$. One would expect that the resulting scheme is **sPA1** secure, but it is not clear that it is. In particular, one would think that for any ciphertext creator for the modified scheme, one could just use two extractors for the original scheme (one for each public-key) to simulate an extractor for the creator. However, this argument does not work, and we are not aware of any other methods for proving the equivalence. One issue is that if a creator switches between making encryptions under pk_b and pk_{1-b}, then at each switch we must incorporate the extractor in to the original ciphertext creator in order to construct a new extractor. The extractors must be continuously incorporated, because definitionally they have no ability to extract encryptions when the ciphertext creator has access to a decryption oracle other than the one simulated by the extractor.

More specifically, consider a ciphertext creator $\mathsf{Crt} = (\mathsf{Crt}_0, \mathsf{Crt}_1, .., \mathsf{Crt}_n)$ for the scheme (g', e', d') where Crt_i denotes the execution after the ith

query.[2] Let Crt switch between the public-key used to encrypt messages for each query, i.e. it encrypts its even queries with pk_0 and its odd queries with pk_1. To make an extractor for Crt (without including the oracles in the definition, as we have done), we would first create Crt'_0 using the standard **sPA1** definition and the extractor for pk_0 that is guaranteed to exist for Crt_0, call it Ext_0, by embedding the extractor as a subroutine into Crt_0. The running time of Crt'_0 is clearly the additive combination of the running time of Ext_0 and Crt_0. One would then compose Crt'_0 with Crt_1 and use the **sPA1** definition to construct an extractor for $\mathsf{Crt}_1 \circ \mathsf{Crt}'_0$, called Ext_1, which only queries decryptions for pk_1. We could continue this inductively, but after a super-constant number of iterations, the running time of the resulting extractor would be super-polynomial.

Finally, we note that common additional definitional traits, like the notion of a history of computation, do not port readily to these extractability definitions. In essence, one needs to consider the possibility that a history string encodes a Turing Machine, which is then run by an extractor acting as a Universal Turing machine. The semantic effect of such a notion in the definition is to swap the order of quantifiers relating to the extractor, further strengthening the definition.

2.4 A Note On PA1$^+$

Dent [8] also investigated an augmented notion of plaintext awareness called PA1$^+$ in which he provides the ciphertext creator access to an oracle that produces random bits. The extractor receives the answers to any queries generated by the creator, but only at the time these queries are issued. The point of this oracle in the context of a plaintext awareness definition is to model the fact that the extractor might not receive all of the random coins used by the creator *at the beginning* of the experiment. Much in the spirit of "adaptive soundness" and "adaptive zero-knowledge", this oracle requires the extractor to work even when it receives the random coins at the same time as the ciphertext creator. Therefore, the extractor potentially needs to be able to extract some ciphertexts independent of future randomness. This modification has implications when the notion of plaintext awareness is computational—as in the case of Dent's work. However, in the case of statistical plaintext awareness, we argue that $\mathbf{sPA1}_\ell$ security also implies $\mathbf{sPA1}^+_\ell$ security.

[2] We can assume the Crt_i outputs its state, which is then used as auxiliary information and passed as input to Crt_{i+1}.

Definition 4. *Define the* $\mathbf{sPA1}_\ell^+$ *experiment in a similar way to the* $\mathbf{sPA1}_\ell$ *experiment.* The only difference *between the two is that during the* $\mathbf{sPA1}_\ell^+$ *experiment, the ciphertext creator has access to a random oracle* $\mathcal{O}$ *that* takes no input, *but returns independent uniform random strings upon each access. Any time the creator access the oracle, the oracle's response is forwarded to both the creator and extractor.*

If an encryption scheme would be deemed $\mathbf{sPA1}_\ell$ *secure, when we replace the* $\mathbf{sPA1}_\ell$ *experiment in the definition with the modified* $\mathbf{sPA1}_\ell^+$ *experiment, then the encryption scheme is said to be* $\mathbf{sPA1}_\ell^+$ *secure.*

Lemma 1. *If an encryption scheme* Π *is* $\mathbf{sPA1}_\ell$ *secure, then it is* $\mathbf{sPA1}_\ell^+$ *secure.*

Proof. Notice that the only difference between **sPA1** and $\mathbf{sPA1}^+$ security definitions is that the latter makes use of an oracle $\mathcal{O}$ that returns random bits upon access that is not known in advance to either the adversary or the extractor. If the adversary Crt^+ does not access $\mathcal{O}$ during its execution then $\mathbf{sPA1}^+$ security holds since i) with no access to $\mathcal{O}$, $\mathbf{sPA1}^+$ and **sPA1** security are equivalent, ii) **sPA1** security holds (i.e. for any given adversary, there exists an extractor that can decrypt the queries correctly). Hence in what follows we only argue that $\mathbf{sPA1}^+$ security holds if the adversary accesses the oracle $\mathcal{O}$ at least once.

Let Crt^+ be an $\mathbf{sPA1}^+$ adversary. The intuition for this argument is that the answers that the Crt^+ receives from $\mathcal{O}$ can be interpreted as "the end of the random tape" for some **sPA1** adversary Crt. In other words, Crt runs Crt^+ internally and answers the queries to $\mathcal{O}$ by reading the end portion of its random tape. By properly formalizing this to handle polynomially many queries to $\mathcal{O}$, it is easy to see that Crt will make the same distribution of queries to its extractor that Crt^+ makes to its own extractor Ext^+. By **sPA1**-security, there exists an Ext that works for the queries that Crt produces.

This observation provides a plausible model for how Ext^+ could work: it begins by sampling a random tape $R^+ \leftarrow (R|r'_1|\cdots|r'_n)$ for Crt (i.e., with randomly sampled answers r_i to $\mathcal{O}$ queries at the end of the tape). When it is asked queries to decrypt, it simply runs Ext (which must work properly) using this random tape. At some point, the first oracle query to $\mathcal{O}$ will be made and thus Ext^+ will receive a random string r_1 as the first oracle $\mathcal{O}$ query answer. At this point, Ext^+ updates the tape $R^+ \leftarrow (R|r_1|r'_2|\cdots|r'_n)$ with the correct answer, restarts its execution of Ext on this new tape by feeding it all of the same decryption queries that have been received up until this point. This results in a new state for the extractor that will

be used to answer future decryption queries. The remaining decryption queries and queries to $\mathcal{O}$ will be handled similarly.

One can observe that if ℓ queries to $\mathcal{O}$ are made, then Ext^+ must restart the execution of Ext ℓ times, and thus its running time will be a summation of ℓ running times of Ext. This summation will still be polynomial in the security parameter. Moreover, if Ext^+ fails to answer a decryption query properly, then it serves as a polynomial-time procedure that—by running Ext at most ℓ times—is able to produce a set of queries that breaks **sPA1**-security. In what follows, we present a more formal argument that shows how an **sPA1**$^+$-adversary Crt^+ that succeeds in causing every Ext^+ to fail can be used, using the ideas above, to produce an **sPA1**-adversary Crt that violates **sPA1**-security.

Assume that the Crt^+ makes polynomially many queries using its random tape R, then accesses $\mathcal{O}$ once (and gets in return some random coins r_1) and asks one more query q that Ext^+ fails to decrypt correctly. Assume that $(R|r_1)$ is ℓ bits. We build an adversary Crt that simulates Crt^+ using the first ℓ bits of its random tape. Crt reads the first ℓ bits of its random tape, parses it as $(R|r_1)$ (call the rest of its random tape $r'_2|r'_3|\ldots|r'_n$) and simulates Crt^+ on random coins R. Crt submits Crt^+'s queries to its extractor and forwards back the answer to Crt^+. When Crt^+ calls $\mathcal{O}$, Crt returns the r_1 portion of its tape. Then it continues running Crt^+ until it gets its next decryption query q, submits this query to its extractor, and halts. Notice the distribution of queries that Crt asks to its extractor is the same as Crt^+. Also, Crt does a perfect simulation of the **sPA1**$^+$ game for Crt^+, so the query q is also distributed identically. Since Π is **sPA1** secure, there must be some extractor Ext such that $\Pr[\mathbf{sPA1}(\Pi, \mathsf{Crt}, \mathsf{Ext}, k) = 0]$ is negligible when Ext is run on input tape $(R|r_1|r'_2|r'_3|\ldots|r'_n)$ as the random coins for Crt. Thus, the probability that Ext answers the query q correctly must be $1 - \epsilon(k)$ for some negligible function ϵ. However, notice that Ext^+ also answers q by running Ext on $(R|r_1|r'_2|r'_3|\ldots|r'_n)$ and hence returns the same correct decryption of q to Crt^+. This contradicts our assumption that Ext^+ decrypt q incorrectly. Similar argument can be made about any other query that Crt^+ makes to show that Ext^+ returns the right decryption for that query. Hence we conclude that Ext^+ always returns the right answer to all of the Crt^+'s queries. □

3 More Than Non-Malleable CCA1 Encryption Scheme

We show how to construct an encryption scheme that is cNM-CCA1 secure where c is a constant. The high level idea for constructing a cNM-CCA1 scheme is to add $c-1$ layers of encryption atop the basic encryption of a message m, effectively redundantly re-encrypting the previous layer's ciphertext and forming a new layer of encryptions. Intuitively, each parallel query that the adversary asks can help it peel back the security of one of the layers of encryption in the challenge ciphertext, and therefore if a challenge ciphertext is composed of c layers, then the scheme can withstand c parallel queries.

3.1 The Construction

For the base case, we define $\text{NMGen}^{(0)} = \mathsf{gen}$; $\text{NMEnc}^{\dagger(0)}(pk, m, \mathsf{SigVK}) = \mathsf{enc}(pk, m)$; and $\text{NMDec}^{\dagger(0)}(sk, c, \mathsf{SigVK}) = \mathsf{dec}(sk, c)$ where the weakly simulatable and $\mathbf{sPA1}_\ell$ secure encryption primitive $\mathbf{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is the starting point for our work. Next, we recursively perform multiple redundant parallel encryptions of the last recursive steps output. In each of these steps, we use the standard practice of interpreting the bits of a freshly generated verification key for a one-time signature scheme to choose appropriate public keys with which to encrypt. The resulting set of ciphertexts is finally signed with the one-time signature's signing key to form the final encryption. Decryption proceeds as one might expect: first the signature is checked for validity, and next the encryption is recursively decrypted, where at each level it is ensured that the redundant parallel decryptions all encode the same underlying "message". The encryption scheme $\Pi^{(c)}$ parameterized by an integer $c > 0$ appears in Fig. 4.

3.2 Preliminary Notion

Before we present our security proof, we introduce an intermediate "tagged encryption" security game to simplify our proof. We call this notion $(\mathsf{c})\mathsf{NME}^*$ security and it allows each ciphertext to have an associated tag used during both encryption and decryption. The challenge ciphertext is tagged with the vector $\vec{0}$, and the adversary can submit any query with a non-zero tag.

Along with this new definition, we present a natural analog of our original encryption scheme $\Pi^{*(c)}$ in Fig. 6. The difference is that the signature

$\mathrm{NMGen}^{(c)}(1^k)$

1: $(\mathrm{NPK}^{(c-1)}, \mathrm{NSK}^{(c-1)}) \leftarrow \mathrm{NMGen}^{(c-1)}(1^k)$

2: $\forall i \in [k]$ and $b \in \{0,1\}$, $(pk_i^b, sk_i^b) \leftarrow \mathsf{gen}(1^k)$ s.t. pk_i^b encrypts the range of $\mathrm{NMEnc}^{\dagger(c-1)}$

3: Output $\mathrm{NPK}^{(c)} = \{\mathrm{NPK}^{(c-1)}, \{pk_i^b\}_{\substack{i\in[k] \\ b\in\{0,1\}}}\}$ and $\mathrm{NSK}^{(c)} = \{\mathrm{NSK}^{(c-1)}, \{sk_i^b\}_{\substack{i\in[k] \\ b\in\{0,1\}}}\}$

$\mathrm{NMEnc}^{(c)}(\mathrm{NPK}^{(c)}, m)$

1: $(\mathsf{SigSK}, \mathsf{SigVK}) \leftarrow \mathsf{GenKey}(1^k)$

2: $c \leftarrow \mathrm{NMEnc}^{\dagger(c)}(\mathrm{NPK}^{(c)}, m, \mathsf{SigVK})$

3: $\sigma \leftarrow \mathsf{Sign}_{\mathsf{SigSK}}(c)$

4: Output $C = (c, \mathsf{SigVK}, \sigma)$

$\mathrm{NMEnc}^{\dagger(c)}(\mathrm{NPK}^{(c)}, m, \mathsf{SigVK})$

1: Parse $\mathrm{NPK}^{(c)}$ into $(\mathrm{NPK}^{(c-1)}, \vec{pk} = \{pk_1^b, \ldots, pk_k^b\}_{b\in\{0,1\}})$

2: Let SigVK_i be the ith bit of SigVK.

3: $c_0'' \leftarrow \mathrm{NMEnc}^{\dagger(c-1)}(\mathrm{NPK}^{(c-1)}, m, \mathsf{SigVK})$

4: $c_i' \leftarrow \mathsf{enc}_{pk_i^{\mathsf{SigVK}_i}}(c_0''); \forall i \in [k]$

5: Output $c = \vec{c'}$

$\mathrm{NMDec}^{(c)}(\mathrm{NSK}^{(c)}, C)$

1: Parse C **as** $(c, \mathsf{SigVK}, \sigma)$ and let SigVK_i be the ith bit of SigVK.

2: **if** $\mathsf{Verify}_{\mathsf{SigVK}}(\sigma, \vec{c}) = 0$ **then** Output $\perp$

3: Output $\mathrm{NMDec}^{\dagger(c)}(\mathrm{NSK}^{(c)}, c, \mathsf{SigVK})$

$\mathrm{NMDec}^{\dagger(c)}(\mathrm{NSK}^{(c)}, c, \mathsf{SigVK})$

1: Parse $\mathrm{NSK}^{(c)}$ into $(\mathrm{NSK}^{(c-1)}, \vec{sk} = \{sk_1^b, \ldots, sk_k^b\}_{b\in\{0,1\}})$

2: $\forall i \in [k]$, compute $c_i' \leftarrow \mathsf{dec}_{sk_i^{\mathsf{SigVK}_i}}(c_i)$

3: **if** $\exists i \in [k]$ s.t. $c_1' \neq c_i'$ **then** Output $\perp$

4: Output $\mathrm{NMDec}^{\dagger(c-1)}(\mathrm{NSK}^{(c-1)}, c_1', \mathsf{SigVK})$

Fig. 4: The cNM-CCA1 Encryption Scheme $\Pi^{(c)}$

scheme used for unduplicatable set selection is replaced by the k-bit tag α.

$(\mathsf{c})\mathsf{NME}_b{}^*(\Pi^*, \mathcal{A}, \mathcal{D}, k, p(k))$

1: $(\textsc{npk}, \textsc{cnsk}) \leftarrow \mathrm{NMGen}^*(1^k)$

2: $(m_0, m_1, S_1) \leftarrow A_0^{\mathrm{NMDec}^*(\textsc{nsk},\cdot)}(\textsc{cnpk})$ s.t. $|m_0| = |m_1|$

The oracle $\mathrm{NMDec}^*(\textsc{nsk}, Y = (y, \alpha))$ returns $\perp$ if $\alpha = 0^k$

3: $Y^* \leftarrow \mathrm{NMEnc}^*(\textsc{npk}, m_b, \alpha = 0^k)$

4: $\vec{d}_1 \leftarrow (\perp)$

5: *for* $i = 1$ *to* c

6: $\quad (\vec{Y}, S_{i+1}) \leftarrow \mathcal{A}_i(Y^*, S_i, \vec{d}_i)$ where $|\vec{Y}| = p(k)$

7: $\quad$ If $\alpha \neq 0^k$, $d_{i+1,j} \leftarrow \mathrm{NMDec}^*(\textsc{nsk}, Y_j = (y_j, \alpha))$

8: $\quad$ Else $d_{i+1,j} \leftarrow \perp$; $\forall j \in [|\vec{Y}|]$

9: Output $\mathcal{D}(Y^*, \vec{d}_{c+1}, S_{c+1})$

Fig. 5: The $(\mathsf{c})\mathsf{NME}^*$ Experiment For $c \geq 0$

$\mathrm{NMGen}^{*(c)}(1^k)$

1: Defined as $\mathsf{NMGen}^{(c)}(1^k)$ in Fig. 4

$\mathrm{NMEnc}^{*(c)}(\textsc{npk}^{(c)}, m, \alpha \in \{0,1\}^k)$

2: Return $(\mathrm{NMEnc}^{\dagger(c)}(\textsc{npk}^{(c)}, m, \alpha), \alpha)$ where $\mathrm{NMEnc}^{\dagger(c)}$ is defined in Fig. 4,

$\mathrm{NMDec}^{*(c)}(\textsc{nsk}^{(c)}, Y = (y, \alpha \in \{0,1\}^k))$

3: Defined as $\mathrm{NMDec}^{\dagger(c)}(\textsc{nsk}^{(c)}, y, \alpha)$ in Fig. 4

Fig. 6: The Encryption Scheme $\Pi^{*(c)} = (\mathrm{NMGen}^{*(c)}, \mathrm{NMEnc}^{*(c)}, \mathrm{NMDec}^{*(c)})$

As the next lemma shows, there is no difference between these security games; for every adversary in the tagged security game, there exists an equivalently succesful adversary for the $(\mathsf{c})\mathsf{NME}$ game.

Lemma 2. *For any ppt adversary $\mathcal{A}$, integer $c > 0$, polynomial p and security parameter k, there exists an adversary $\mathcal{B}$ s.t.*

$$\left\{(\mathsf{c})\mathsf{NME}_b(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \equiv \left\{(\mathsf{c})\mathsf{NME}_b{}^*(\Pi^{*(c)}, \mathcal{B}, \mathcal{D}, k, p(k))\right\}_k$$

Proof. We build a $(\mathsf{c})\mathsf{NME}^*$ adversary $\mathcal{B}$ that interacts with the $(\mathsf{c})\mathsf{NME}^*$ experiment by simulating the $(\mathsf{c})\mathsf{NME}$ experiment for $\mathcal{A}$. $\mathcal{B}$ receives pk as in Line 2 of the experiment (Fig. 5) and proceeds to generate a pair

of signature keys $(\text{skSig}^*, \text{vkSig}^*) \leftarrow \mathsf{GenKey}(1^k)$. $\mathcal{B}$ then sets $pk' \leftarrow \text{ReArrange}(pk, \text{vkSig}^*)$ where the function ReArrange is presented in Fig. 7. Intuitively this function rearranges the keys in pk and pk' so that $\mathcal{B}$ can sign a challenge ciphertext that it will eventually receive using skSig^* and then produce an encryption according to the $\Pi^{(c)}$ scheme using only the keys in pk. $\mathcal{B}$ runs $\mathcal{A}_0$ on pk'.

ReArrange(pk, vkSig^*)

1: Parse pk as $\left(pk_0, \{pk_i^b\}_{i\in[c\cdot k],b\in\{0,1\}}\right)$
2: **for** $i \in [0..c-1]$
3: **for** $j \in [k]$
4: **if** $\text{vkSig}_j^* = 1$ **then** swap the values $pk_{i\cdot k+j}^0$ and $pk_{i\cdot k+j}^1$
5: **endFor**
6: **endFor**
7: Return $pk = \left(pk_0, \{pk_i^b\}_{i\in[0\ldots ck],b\in\{0,1\}}\right)$

Fig. 7: The Definition for the ReArrange Function

Whenever $\mathcal{A}_0$ asks a query $Y = (\vec{y}, \sigma, \text{vkSig})$, $\mathcal{B}$ does the followings: $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_0$ as the answer to the query if either $\text{vkSig} = \text{vkSig}^*$ or $\mathsf{Verify}_{\text{vkSig}}(\sigma, \vec{y}) = 0$. Otherwise $\mathcal{B}$ sets $\alpha \leftarrow \bigoplus(\text{vkSig}^*, \text{vkSig})$ where $\bigoplus$ is the bitwise XOR function on two vectors of the same length. Intuitively, this finds the right α that shows under which subset of pk keys the vector $\vec{y}$ is encrypted.

$\mathcal{B}$ then asks $(\vec{y}, \alpha)$ to its oracle and forwards the answer to its simulation of $\mathcal{A}_0$. Eventually $\mathcal{A}_0$ returns (m_0, m_1, S_1) and halts. $\mathcal{B}$ outputs (m_0, m_1) to the environment (i.e., its experiment) and receives a challenge ciphertext $(\vec{y^*}, 0^k)$. $\mathcal{B}$ computes $\sigma^* \leftarrow \mathsf{Sign}_{\text{skSig}}(\vec{y^*})$, sets $Y^* \leftarrow (\vec{y^*}, \sigma^*, \text{vkSig}^*)$, sets $\vec{d_1} \leftarrow \perp$ and does the following for all $q = 1$ to c:

> $\mathcal{B}$ simulates $\mathcal{A}_q$ on the input $(Y^*, S_q, \vec{d_q})$ and receives in return a vector of ciphertexts $\vec{Y}$ and the state information S_{q+1}. $\mathcal{B}$ computes the decryption to each of Y_i's in the same way that it computed the decryption to the CCA1 queries with the difference that it asks all of the queries in the same parallel query at once from the environment (instead of asking sequentially). Call the vector of decryption of the queries $\vec{d_{q+1}}$.

Eventually $\mathcal{B}$ outputs $\vec{d_{c+1}}$ and S_{c+1} and halts. □

3.3 Main Theorem

The heart of our main theorem relies on Lemma 3 (introduced shortly) which, informally, shows that for any ppt adversary $\mathcal{A}$, there exists a ppt adversary $\mathcal{B}$ such that

$$\left\{(\mathsf{c})\mathsf{NME}_b{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_{b,k} \approx_c \left\{(\mathsf{c}\text{-}1)\mathsf{NME}_b{}^*(\Pi^{*(c-1)}, \mathcal{B}, \mathcal{D}, k, p(k))\right\}_{b,k}.$$

Assuming Lemma 3, we state and give the proof our main theorem below. We then formally state and prove Lemma 3.

Theorem 2. *If the encryption scheme $\boldsymbol{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is weakly simulatable and $\textbf{sPA1}_\ell$ secure, then for any integer $c > 0$, construction $\Pi^{(c)} = (NMGen^{(c)}, NMEnc^{(c)}, NMDec^{(c)})$ in Fig. 4 is $(\mathsf{c})\mathsf{NME}$ secure.*

Proof. By applying Lemma 3 c times, we have that

$$\left\{(\mathsf{c})\mathsf{NME}_0{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \overset{\text{Lemma 3}}{\approx_c} \cdots \approx_c \left\{(\mathsf{o})\mathsf{NME}_0{}^*(\Pi^{*(0)}, \mathcal{B}, \mathcal{D}, k, p(k))\right\}_k$$

In Claim 1 (below), we show that construction $\Pi^{*(0)}$ is $(\mathsf{o})\mathsf{NME}^*$-secure, and thus it follows that

$$\left\{(\mathsf{c})\mathsf{NME}_0{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \approx_c \left\{(\mathsf{o})\mathsf{NME}_1{}^*(\Pi^{*(0)}, \mathcal{B}, \mathcal{D}, k, p(k))\right\}_k$$

Applying Lemma 3 again on the right hand side, , it follows that

$$\left\{(\mathsf{c})\mathsf{NME}_0{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \approx_c \left\{(\mathsf{c})\mathsf{NME}_1{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k$$

Finally applying Lemma 2 to show equivalence between $(\mathsf{c})\mathsf{NME}$ and $(\mathsf{c})\mathsf{NME}^*$ completes the theorem. □

Claim 1. *If the encryption scheme $\boldsymbol{E} = (\mathsf{gen}, \mathsf{enc}, \mathsf{dec})$ is weakly simulatable and $\textbf{sPA1}_\ell$ secure, then for all ppt adversaries and distinguishers $\mathcal{A}$ and $\mathcal{D}$ respectively and for all polynomials p:*

$$\left\{(0)\mathsf{NME}_0{}^*(\Pi^{*(0)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k \approx_c \left\{(0)\mathsf{NME}_1{}^*(\Pi^{*(0)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_k$$

where the experiment $(0)\mathsf{NME}^*$ is defined in Fig. 5 and the encryption scheme $\Pi^{*(0)} = (NMGen^{*(0)}, NMEnc^{*(0)}, NMDec^{*(0)})$ is defined in Fig. 6.

Proof. By definition, notice that $\text{NMEnc}^{*(0)} = \text{NMEnc}^{\dagger(0)}(pk, m, \alpha) = \mathsf{enc}(pk, m)$; in fact, $\Pi^{*(0)}$ is equivalent to $\mathbf{E}$. Second, the $(0)\mathsf{NME}^*$ experiment has no parallel queries after the challenge has been submitted, and is therefore (roughly) equivalent to the CCA1-security game. By assumption, $\mathbf{E}$ is $\mathbf{sPA1}_\ell$-secure and therefore CCA1-secure, which completes the claim. □

It remains to prove the key technical lemma; we first present a high-level overview of the proof. Our goal is to show how to simulate an Adversary $\mathcal{A}$ that makes c parallel decryption queries when given a c-layered challenge ciphertext, with an adversary $\mathcal{B}$ that only has access to $c-1$ parallel decryption queries, and a $c-1$ layered challenge ciphertext. It is easy to see how we can simulate the extra layer of the challenge ciphertext, $\mathcal{B}$ can simply generate its own keys and add an extra layer to its challenge ciphertext to simulate $\mathcal{A}$. The question remains how to simulate the extra parallel decryption that $\mathcal{A}$ has access to. It may seem, on first glance, to follow immediately for the $\mathbf{sPA1}_\ell$ security of the underlying encryption scheme, because the whole purpose of an extractor is to simulate a decryption oracle. However, $\mathcal{A}$ is fed a challenge ciphertext which it did not produce (and thus there is no extraction guarantee), and $\mathcal{A}$ might create its parallel decryption queries based on the challenge ciphertext, in which case there is no a priori reason to believe that $\mathsf{Ext}_\mathcal{A}$ will be able to "decrypt" properly when used to decrypt the "extra" c^{th} parallel decryption query.

To solve this issue, we use the non-duplicatable set selection to ensure that there is a new public-key with respect to which the adversary must have generated part of the ciphertext (and not just mauled part of the challenge ciphertext); we can then be assured that the extractor will work on this portion of the challenge ciphertext. However, this by itself does not allow us to simulate the consistency check in the decryption algorithm that ensures that all of the encryptions at a given level are of the same message. For the outer layer of ciphertexts that need to be decrypted, we have the corresponding secret-keys since $\mathcal{B}$ generated the corresponding public-keys. The inner-layers are another matter entirely. In order to argue this, we use the $\mathbf{sPA1}_\ell{}^+$ security of the underlying encryption scheme in conjunction with the fact that it is weakly simulatable. In essence this means that the extractor cannot tell the difference between when the outer layer of the challenge ciphertext is legitimate encryptions and when they were instead created on demand using the simulator with randomness provided via an oracle. However, in the latter case, by the definition of $\mathbf{sPA1}_\ell{}^+$ security, the extractor must function.

There is one last subtlety, which is that due to technical requirements of the proof, we actually do not necessarily have access to the secret-keys for the outer layer of the encryptions of $\mathcal{A}$ when we need it, and therefore cannot perform the outer layer consistency check via the extractor. It suffices for this check to be done in a hybrid experiment using the actual secret-key (independent of where it comes from). However, we need to ensure that the responses from these consistency checks do not affect the viability of finding a suitable extractor. Here, the fact that we have $p(k)$ parallel decryption queries to simulate, as opposed to $p(k)$ adaptive queries, is used. Essentially, we consider a system which decrypts all of the parallel queries via the extractor, ignoring the initial consistency checks.

Lemma 3. *For any integer $c > 0$, any ppt adversary $\mathcal{A}$, polynomial p and security parameter k, there exists a ppt adversary $\mathcal{B}$ such that*

$$\left\{(\mathsf{c})\mathsf{NME}_b{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))\right\}_{b,k} \approx_c \left\{(\mathsf{c\text{-}1})\mathsf{NME}_b{}^*(\Pi^{*(c-1)}, \mathcal{B}, \mathcal{D}, k, p(k))\right\}_{b,k}$$

Proof. Consider the following hybrid experiment:
Experiment $\mathsf{H}_b{}^*(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ proceeds similarly to $(\mathsf{c})\mathsf{NME}_b{}^*$ with the difference that the former experiment handles the decryption of all ciphertexts up to the second parallel query differently. After all of the public keys have been generated, initialize $st \leftarrow \left(\{pk_i\}_{i\in[2ck+1]}, R_{\mathcal{A}}\right)$ where $R_{\mathcal{A}}$ are the random coins that will be used to run $\mathcal{A}$. For all $\mathsf{CCA1}$ queries that $\mathcal{A}$ makes (i.e., queries that are made before the challenge ciphertext is produced), everytime that NMDec^* calls the decryption function dec on y_i, the experiment calls $\mathsf{Ext}_{\mathcal{A}}(st, y_i, \cdot)$ with the appropriate pk as the third argument. After $\mathcal{A}$ receives the challenge ciphertext, the *first* parallel query $\{d_i\}$ is decrypted using $\mathsf{NMDecAlt}$ defined below (without loss of generality, assume that $d_i = (\vec{C}, \sigma, \mathrm{vkSig})$). The remaining $(c-1)$ parallel queries are decrypted as per $(\mathsf{c})\mathsf{NME}^*$.

$\mathsf{NMDecAlt}(d_i = (\vec{C}, \alpha))$:

1: If $\alpha = 0^k$ output $\perp$, else let i' be the first index at which $\alpha_{i'} \neq 0$.
2: For $i \in [k]$, do $C'_i \leftarrow \mathsf{dec}_{sk_i^{\alpha_i}}(C_i)$
3: Call $Y^{(c-1)} \leftarrow \mathsf{Ext}_{\mathcal{A}}(st, C_{i'}, pk_{i'}^{\alpha_{i'}})$

(notice that $Y^{(c-1)} = (y_1^{(c-1)}, \ldots, y_k^{(c-1)})$ is a vector).

4: $m \leftarrow \mathrm{ExtractAll}(pk, Y^{(c-1)}, (c-1), \alpha)$
5: If $\exists j$ s.t. $C'_1 \neq C'_j$, return $\perp$. Else return m

$\mathrm{ExtractAll}(pk, Y = (y_1^c, \ldots, y_k^c), c, \alpha)$:

1: **for** $i = c-1$ *to* 0
2: **for** $j = 1$ *to* k
3: $y_j^i \leftarrow \mathsf{Ext}_{\mathcal{A}}(st, y_j^{(i+1)}, pk_{(i+1)\cdot k+j}^{\alpha_j})$
4: **if** $\exists d \in [k]$ **s.t.** $y_1^i \neq y_d^i$ **return** $\perp$
5: $m \leftarrow \mathsf{Ext}_{\mathcal{A}}(st, y_1^0, pk_0)$
6: **Return** m

Intuitively ExtractAll submits the inner layer of the query $Y^{(c-1)}$ to the extractor to be decrypted under the appropriate keys until it reaches the innermost layer containing message m.

To define the function extractor $\mathsf{Ext}_{\mathcal{A}}$ used in NMDecAlt above, we first define an $\mathbf{sPA1}^{+}_{2ck+1}$ ciphertext creator $\mathsf{Crt}_{\mathcal{A}}$ (which makes calls to an extractor oracle) that roughly mimics the queries made by adversary $\mathcal{A}$ in the H_b experiment. We define $\mathsf{Crt}_{\mathcal{A}}$ as follows:

1. $\mathsf{Crt}_{\mathcal{A}}$ receives $2ck+1$ public-keys $pk = \left(pk_0, \{pk_i^b\}_{i\in[1\ldots ck], b\in\{0,1\}}\right)$ from the $\mathbf{sPA1}^{+}_{2ck+1}$ experiment. $\mathsf{Crt}_{\mathcal{A}}$ reads its random tape as $R_{\mathcal{A}}$ and runs $\mathcal{A}_0(pk)$ using tape $R_{\mathcal{A}}$.

2. Whenever $\mathsf{Crt}_{\mathcal{A}}$ receives a query $\left(\{y_i\}_{i\in[k]}, \alpha\right)$ from $\mathcal{A}_0$, it returns $\perp$ if $\alpha = 0^k$. Otherwise, $\mathsf{Crt}_{\mathcal{A}}$ submits each $(y_i, pk_i^{\alpha_i})$ to its extractor. If all of the queries do not decrypt to the same value, $\mathsf{Crt}_{\mathcal{A}}$ returns $\perp$ to $\mathcal{A}_0$ as the answer to that query. Call the decrypted value $Y^{(c-1)}$ and notice that it should be a vector of ciphertexts encrypted under k public keys in pk. Next $\mathsf{Crt}_{\mathcal{A}}$ calls $m \leftarrow$ExtractAll$(pk, \alpha, Y^{(c-1)}, c-1)$. $\mathsf{Crt}_{\mathcal{A}}$ returns m to $\mathcal{A}_0$ as the answer to the query. Eventually $\mathcal{A}_0$ returns (m_0, m_1, St) and halts.

3. $\mathsf{Crt}_{\mathcal{A}}$ accesses its oracle $\mathcal{O}$ to generate k blocks of ℓ random bits $(x_1, \ldots, x_k)$ and then computes the vector $\vec{y} = \left(f(pk_{k(c-1)+1}^0, x_1), \ldots, f(pk_{k(c-1)+k}^0, x_k)\right)$. $\mathsf{Crt}_{\mathcal{A}}$ then runs $\mathcal{A}_1(y^*, St)$ where $y^* = (\vec{y}, 0^k)$ and St is the state information returned by $\mathcal{A}_0$.

4. $\mathcal{A}_1$ returns a vector of ciphertexts $\vec{Y}$ and the state information S and halts. For each query $Y_j = (\{y_i\}_{i\in[k]}, \alpha)$, $\mathsf{Crt}_{\mathcal{A}}$ executes steps 1,3, and 4 of procedure NMDecAlt to decrypt the message. After each ciphertext in the first parallel query has been decrypted in this way, $\mathsf{Crt}_{\mathcal{A}}$ halts.

The $\mathbf{sPA1}_\ell^+$ security of $\mathbf{E}$ implies there exists an extractor $\mathsf{Ext}_\mathcal{A}$ whose answers to the decryption queries submitted by $\mathsf{Crt}_\mathcal{A}$ are indistinguishable from their true decryptions. We have now defined $\mathsf{Ext}_\mathcal{A}$ used in NMDecAlt. Notice that $\mathsf{Crt}_\mathcal{A}$ does not exactly simulate $\mathcal{A}$'s view in H_b; we will argue below why $\mathsf{Ext}_\mathcal{A}$ continues to work properly when it is used in H_b.

Claim 2. *For* $b \in \{0,1\}$, $\{(\mathsf{c})\mathsf{NME}_b{}^*\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}} \approx_c \{\mathsf{H}_b\left(\Pi^{(c)}, \mathcal{A}, \mathcal{D}, k, p\left(k\right)\right)\}_{k\in\mathbb{N}}$

Proof. Experiments $(\mathsf{c})\mathsf{NME}_b{}^*$ and H_b differ only if $\mathsf{Ext}_\mathcal{A}$ answers with an incorrect decryption in the latter experiment. The assumption on the $\mathbf{sPA1}_\ell$-security of $\mathbf{E}$ implies:

$$\Pr[\mathbf{sPA1}_\ell^+(\Pi^{*(c)}, \mathsf{Crt}_\mathcal{A}, \mathsf{Ext}_\mathcal{A}, k) = 0] \leq \mu_1(k) \tag{5}$$

for some negligible μ_1 and therefore $\mathsf{Ext}_\mathcal{A}$ correctly answers all of the queries issued by $\mathsf{Crt}_\mathcal{A}$ with very high probability (recall that the $\mathbf{sPA1}$ random variable being o corresponds to an incorrect decryption event). As mentioned, $\mathsf{Crt}_\mathcal{A}$ does not exactly mimic $\mathcal{A}$'s view in H_b and so it is not obvious that $\mathsf{Ext}_\mathcal{A}$ answers correctly in H_b. The two notable differences are that (i) $\mathsf{Crt}_\mathcal{A}$ uses the weak simulatability of the base encryption scheme to create the challenge ciphertext instead of using enc to produce the challenge, and (ii) $\mathsf{Crt}_\mathcal{A}$ does not perform a consistency check that all $C'_1 = C'_j$ before decrypting the inner ciphertext but instead uses the extractor on the outer layer at a position in which α differs from 0^k and then uses the ExtractAll method on the resulting inner ciphertext.

In order to handle the first difference, we analyze $\Pr[\mathbf{sPA1}_\ell^{++}(\Pi^{*(c)}, \mathsf{Crt}_\mathcal{A}, \mathsf{Ext}_\mathcal{A}, k) = 0]$ where $\mathbf{sPA1}_\ell^{++}$ is an experiment identical to $\mathbf{sPA1}_\ell^+$ with two differences:

1. First, a random bit d is selected and fixed for the remainder of the game.
2. The oracle $\mathcal{O}$ returns random bits as follows: when $\mathsf{Crt}_\mathcal{A}$ accesses the oracle $\mathcal{O}$ for the i^{th} time, instead of $r \in \{0,1\}^l$, $\mathcal{O}$ returns $f^{-1}(pk^0_{k(c-1)+i}, \mathsf{enc}_{pk^0_{k(c-1)+i}}(m_d))$.

We argue that $\mathsf{Ext}_\mathcal{A}$ answers all queries correctly in these two games must be negligibly close by the weak-simulatability property:

Claim 3. $|\Pr[\mathbf{sPA1}_\ell^+(\Pi^{*(c)}, \mathsf{Crt}_\mathcal{A}, \mathsf{Ext}_\mathcal{A}, k) = 0] - \Pr[\mathbf{sPA1}_\ell^{++}(\Pi^{*(c)}, \mathsf{Crt}_\mathcal{A}, \mathsf{Ext}_\mathcal{A}, k) = 0]| < \mu(k)$.

Proof. Consider the weak-simulatability adversary $\mathcal{B}$ defined as follows:

(Recall that in the first step, the weak-simulatability challenger samples k pairs of keys $(pk_i, sk_i) \leftarrow \mathsf{gen}(1^k)$ for $i \in [k]$ and a random bit b.) The attacker $\mathcal{B}$ receives k public keys which we call $\{pk^0_{k(c-1)+i}\}_{i\in[k]}$ from the environment. $\mathcal{B}$ then samples another $k+2(c-1)k+1$ random keys using the gen algorithm and fresh random coins to build the public key $pk = \{pk_0, (pk^b_i, sk^b_i)_{i\in[ck],b\in\{0,1\}}\}$ (notice that $\{pk^0_{k(c-1)+i}\}_{i\in[k]}$ in pk are received from the environment and the rest are generated randomly). $\mathcal{B}$ samples random coins $R_{\mathcal{A}}$ for $\mathcal{A}$ and runs step (2) of the description of $\mathsf{Crt}_{\mathcal{A}}$ using $\mathsf{Ext}_{\mathcal{A}}$. Eventually $\mathcal{A}$ writes (m_0, m_1) to its write-only tape. $\mathcal{B}$ randomly chooses $d \in \{0,1\}$ and stores $c'_d = \mathrm{NMEnc}^{\dagger(c-1)}(pk', m_d, 0^k)$ where $pk' = \{pk_0, (pk^b_i, sk^b_i)_{i\in[(c-1)k],b\in\{0,1\}}\}$ (notice that pk' is the public key for the inner layer of ciphertexts encrypted under pk for the $\Pi^{*(c)}$ encryption scheme and c'_d is the inner layer for an encryption of m_d under pk). For ease of notation, we refer to $\{pk^0_{k(c-1)+i}\}_{i\in[k]}$ as $\vec{pk''}$ (these are the public keys for the outer layer of the challenge ciphertext). Next the challenger samples $r_i \in \{0,1\}^l$ for $1 \le i \le k$ and returns $\{y_i = (r_i, f(pk''_i, r_i))\}_{i\in[k]}$ if $b = 0$, and $\left\{y_i = \left(f(pk''_i, c_i = \mathsf{enc}_{pk''_i}(c'_d)), c_i\right)\right\}_{i\in[k]}$ if $b = 1$. $\mathcal{B}$ then simulates step (3) of $\mathsf{Crt}_{\mathcal{A}}$ by running $\mathcal{A}_1(y^*, St)$ where $y^* = (\vec{y}, 0^k)$ and St is the state information returned by $\mathcal{A}_0$. $\mathcal{A}_1$ returns a vector of ciphertexts $\vec{Y}$ and the state information S and halts. $\mathcal{B}$ runs step (4) of $\mathsf{Crt}_{\mathcal{A}}$ on $\vec{Y}$. Finally attacker $\mathcal{B}$ outputs $b' = 0$ if all the queries made to $\mathsf{Ext}_{\mathcal{A}}$ so far were answered correctly and $b' = 1$ otherwise. This check can be done by using the secret keys for the spots in pk that are generated by $\mathcal{B}$ (all of them except $\{pk^0_{k(c-1)+i}\}_{i\in[k]}$ which is received from the environment) because after $\mathcal{A}$ returns (m_0, m_1), the only queries that it asks to its extractor are with respect to ciphertexts encrypted under the mentioned keys in pk.

The case $b = 0$ corresponds to experiment $\mathbf{sPA1}^+_\ell(\Pi^{*(c)}, \mathsf{Crt}_{\mathcal{A}}, \mathsf{Ext}_{\mathcal{A}}, k)$ and the case $b = 1$ corresponds to $\mathbf{sPA1}^{++}_\ell(\Pi^{*(c)}, \mathsf{Crt}_{\mathcal{A}}, \mathsf{Ext}_{\mathcal{A}}, k)$. For convenience, in the following equations, we abbreviate the two experiments as $\mathbf{sPA1}^+_\ell$ and $\mathbf{sPA1}^{++}_\ell$ respectively. It follows that:

$$\begin{aligned}\Pr[\mathsf{DIST}_{\mathbf{E}'}((f,f^{-1}),k,\mathcal{B})=1] &= \Pr[b=0]\cdot\Pr[\mathbf{sPA1}^+_\ell=0] + \Pr[b=1]\cdot\Pr[\mathbf{sPA1}^{++}_\ell=1]\\ &= (1/2)\Pr[\mathbf{sPA1}^+_\ell=0] + (1/2)(1-\Pr[\mathbf{sPA1}^{++}_\ell=0])\\ &= 1/2 + 1/2(\Pr[\mathbf{sPA1}^+_\ell=0] - \Pr[\mathbf{sPA1}^{++}_\ell=0])\end{aligned}$$

Since the weak-simulatability property of $\mathbf{E}'$ implies that $|\Pr[\mathsf{DIST}_{\mathbf{E}'}((f,f^{-1}),k,\mathcal{B}) =$

$1] - 1/2| \leq \mu(k)$ for some negligible function μ, it must then follow that

$$|\Pr[\mathbf{sPA1}_\ell^+ = 0] - \Pr[\mathbf{sPA1}_\ell^{++} = 0]| \leq 2\mu(k)$$

which completes the proof of the claim.

□

Combining (5) with Claim 2 implies that $\Pr[\mathbf{sPA1}_\ell^{++} = 0] \leq \mu_2(k)$ for another negligible function μ_2. Moreover, by Bayes rule, we can conclude that there exists another negligible function μ_3 such that both $\Pr[\mathbf{sPA1}_\ell^{++} = 0 \mid d = 0] \leq \mu_3(k)$ and $\Pr[\mathbf{sPA1}_\ell^{++} = 0 \mid d = 1] \leq \mu_3(k)$; i.e., the possibility for incorrect extraction results remains negligible no matter which of m_0 or m_1 is used in the $\mathbf{sPA1}_\ell^{++}$ experiment.

In order to handle (ii), we observe that $\mathsf{Ext}_\mathcal{A}$ only receives queries generated by the first parallel query in both $\mathbf{sPA1}_\ell^{++}$ and experiment H_b. From the beginning of both experiments and up to the point of the challenge ciphertext generation, the initial state st and the distribution of queries fed to $\mathsf{Ext}_\mathcal{A}$ in H_b is identical to those in experiment $\mathbf{sPA1}_\ell^{++}$. (This explains why it is necessary for experiment H_b to make "dummy" calls to $\mathsf{Ext}_\mathcal{A}$ for every call to dec during the decryption of the CCA1 queries.)

When the ciphertext is generated, since $f(f^{-1}(c)) = c$, that challenge ciphertext in experiments H_b and $\mathbf{sPA1}_\ell^{++}$ *conditioned on* $d = b$ will also be identically distributed, and this implies that the parallel query that $\mathcal{A}_1$ issues will also be identically distributed. Once this parallel query has been fixed, the queries that are sent to $\mathsf{Ext}_\mathcal{A}$ are also fixed in both experiments. By inspection, again because $\mathsf{NMDec}^{*(c)^*}$ issues dummy queries to $\mathsf{Ext}_\mathcal{A}$ during the decryption of the outer layer, it follows that the query distribution will be identical, and the claim follows. Thus, the fact that $\mathsf{Crt}_\mathcal{A}$ does not perform the same consistency check is irrelevant since the same distribution of queries is fed to the extractor in both experiments.

It then follows that with high probability, all of the responses from $\mathsf{Ext}_\mathcal{A}$ in H_b coincide with the true decryption, and therefore $(\mathsf{c})\mathsf{NME}_b{}^*$ and H_b also output the same value which concludes the Claim. □

Claim 4. *For any ppt adversary $\mathcal{A}$, polynomial p and security parameter k, there exists an adversary $\mathcal{B}$ such that*

$$\mathsf{H}_b(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k)) \equiv (c-1)\mathsf{NME}_b{}^*(\Pi^{*(c-1)}, \mathcal{B}, \mathcal{D}, k, p(k))$$

Proof. We build the (c-1)NME^* adversary $\mathcal{B}$ as follows: $\mathcal{B}$ receives the public-key $\textsc{npk}^{*(c-1)} = (pk'_0, \vec{pk'} = \{pk_i'^b\}_{i\in[(c-1)k], b\in\{0,1\}})$ from the environment and generates another $2k$ keys as $(pk_i''^b, sk_i''^b) \leftarrow \mathsf{gen}(1^k)$ for $i \in [k]$

and $b \in \{0,1\}$. Let $pk = \left\{pk'_0, \vec{pk'}, \vec{pk''}\right\}$. $\mathcal{B}$ generates random coins $R_{\mathcal{A}}$ and initializes $st \leftarrow \left(\{pk_i\}_{i\in[2ck+1]}, R_{\mathcal{A}}\right)$. $\mathcal{B}$ runs $\mathcal{A}_0(pk; R_{\mathcal{A}})$.

For any CCA1 query $Y = (\vec{y}, \alpha)$ that $\mathcal{A}_0$ submits, $\mathcal{B}$ runs NMDecAlt using $\vec{sk''}$ to decrypt the outer layer. Eventually $\mathcal{A}_0$ returns (m_0, m_1) and the state information S_1 and halts. $\mathcal{B}$ then forwards (m_0, m_1) to the environment and receives a challenge ciphertext $Y' = (\vec{y'}, 0^k)$ from the environment. $\mathcal{B}$ computes $\{y_i^* \leftarrow \mathsf{enc}(pk''_i, \vec{y'})\}_{i\in[k]}$ and sets $Y^* = (\vec{y^*}, 0^k)$. $\mathcal{B}$ also computes $\{r_i \leftarrow f^{-1}(pk''_i, y_i^*)\}_{i\in[k]}$ and forwards $\{r_i\}_{i\in[k]}$ to $\mathsf{Ext}_{\mathcal{A}}$ and runs $\mathcal{A}_1$ on the challenge ciphertext Y^* and the state information S_1. Eventually $\mathcal{A}_1$ returns a vector of queries (the first parallel query) $\vec{Y}$ and the state information S_2 and halts. $\mathcal{B}$ submits each Y_i to the extractor and receives a value which we call m' from it. $\mathcal{B}$ then uses $\vec{sk''}$ to check that all the ciphertexts in the outer layer of Y_i decrypts to the same value. If so, it sets m' as the decryption of that query otherwise it sets $\perp$ as the decryption of that query. We refer to the resulting vector of decryptions to $\vec{Y}$ as $\vec{d_1}$.

For all $q = 2$ to c, $\mathcal{B}$ runs $\mathcal{A}_q$ on Y^*, S_2 and $\vec{d_q}$ and gets in return a vector of ciphertexts $\vec{Y}$ and the state information S_{q+1}. $\mathcal{B}$ decrypts $Y_i = (\vec{y}, \alpha)$ as follows: the decryption is $\perp$ if $\alpha = 0^k$. Otherwise $\mathcal{B}$ uses $\vec{sk''}$ to check all the ciphertexts in the outer layer of Y_i decrypts to the same value y_0. If not, the decryption to this query will be $\perp$. Otherwise $\mathcal{B}$ sets $Y'_i = (y_0, \alpha)$ and moves to the next query. After processing all queries, $\mathcal{B}$ submits $\vec{Y'}$ to the environment and gets in return a vector of decryptions to the $\vec{Y'}$. Using these answers and the results from the checks, $\mathcal{B}$ sets $\vec{d_i}$ (which is the vector of the decryption to $\vec{Y_i}$). Eventually $\mathcal{B}$ outputs $\vec{d_{c+1}}$ and the state information S_{c+1} and halts.

$\mathcal{B}$ performs a perfect simulation of the $\mathsf{H}_b(\Pi^{*(c)}, \mathcal{A}, \mathcal{D}, k, p(k))$ experiment, and thus the claim follows.

□

Combining Claim 2 and Claim 4 completes the proof of Lemma 3. □

3.4 Remarks about the proof

In our construction, we use the k-bit outer-most signature SigVK to pick the unduplicatable set for each of the c layers of encryption. Not only is this choice an efficiency improvement in that only one signature key is needed (instead of c), it is also a critical feature of our proof. This point

is used in Claim 4. Adversary $\mathcal{B}$ must not submit a 0-tag query to its (c-1)NME challenger; but if each layer could use a different α tag, then $\mathcal{A}$ might select 0 as the tag for the $(c-1)$ layer and therefore prevent $\mathcal{B}$ from submitting it to its oracle.

4 Conclusions

We have shown both the first construction of a non-malleable CCA1 encryption scheme from a seemingly weaker primitive, and that it is possible to realize cNM-CCA1 schemes without achieving full CCA2 security. All of our constructions are black-box, although based on hardness assumptions that are not efficiently falsifiable. Major open questions in the area are clearly if CPA security implies CCA1 security, CCA1 security implies CCA2, or the transitive closure. Progress on any of these questions, with either black-box or white-box constructions (or impossibility results), would be of foundational importance to the field.

4.1 Acknolwedgements

We thank the NSF (grant 0939718), DARPA and AFRL (joint contract FA8750-11-2-0211) for their gracious support.

References

[1] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In Pil Joong Lee, editor, *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer, 2004.

[2] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, July 2004.

[3] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2008.

[4] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan.

Bounded cca2-secure encryption. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2007.

[5] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.

[6] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

[7] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2000.

[8] Alexander W. Dent. The cramer-shoup encryption scheme is plaintext aware in the standard model. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 289–307. Springer, 2006.

[9] Alexander W. Dent. The hardness of the dhk problem in the generic group model. *IACR Cryptology ePrint Archive*, 2006:156, 2006.

[10] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[11] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[12] Takahiro Matsuda and Kanta Matsuura. Parallel decryption queries in bounded chosen ciphertext attacks. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 246–264. Springer, 2011.

[13] Steven Myers and Abhi Shelat. Bit encryption is complete. In *FOCS*, pages 607–616. IEEE Computer Society, 2009.

[14] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[15] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen cypher-text attack. In *STOC'90*, pages 427–437, 1990.

[16] Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 271–289. Springer, 2006.

[17] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO'91*, pages 433–444, 1991.

A Instantiating weak-simulatability

Algorithm 1: DEG

$\mathcal{G}(1^k)$
1: $(p, q, g) \leftarrow G(1^k)$
2: $x_1 \leftarrow \mathbb{Z}_q; X_1 \leftarrow g^{x_1} \bmod p$
3: $x_2 \leftarrow \mathbb{Z}_q; X_2 \leftarrow g^{x_2} \bmod p$
4: Return $(pk = (p, q, g, X_1, X_2), sk = (p, q, g, x_1, x_2))$

$\mathcal{E}(pk, M)$
1: $y \leftarrow \mathbb{Z}_q; Y \leftarrow g^y \bmod p$
2: $W \leftarrow X_1^y; V \leftarrow X_2^y \bmod p$
3: $U \leftarrow V \cdot M \bmod p$
4: Return $C = (Y, W, U)$

$\mathcal{D}(sk, C)$
1: **if** $W \neq Y^{x_1} \bmod p$ **then** Return $\perp$
2: Return $M \leftarrow U \cdot Y^{-x_2} \bmod p$

Algorithm 2: CS-Lite

$\mathcal{G}(1^k)$
1: $(p, q, g_1) \leftarrow G(1^k); g_2 \leftarrow G_q \backslash \{1\}$
2: $x_1 \leftarrow \mathbb{Z}_q; x_2 \leftarrow \mathbb{Z}_q; z \leftarrow \mathbb{Z}_q$
3: $X \leftarrow g_1^{x_1} . g_2^{x_2} \bmod p; Z \leftarrow g_1^z \bmod p$
4: Return $(pk = (p, q, g_1, g_2, X, Z), sk = (p, q, g_1, g_2, x_1, x_2, z))$

$\mathcal{E}(pk, M)$
1: $r \leftarrow \mathbb{Z}_q$
2: $R_1 \leftarrow g_1^r \bmod p; R_2 \leftarrow g_2^r \bmod p$
3: $E \leftarrow Z^r \cdot M \bmod p; V \leftarrow X^r \bmod p$
4: Return $C = (R_1, R_2, E, V)$

$\mathcal{D}(sk, C)$
1: **if** $V \neq R_1^{x_1} \cdot R_2^{x_2} \bmod p$ **then** Return $\perp$
2: Return $M \leftarrow E \cdot R_1^{-z} \bmod p$

Fig. 8: The Encryption Schemes DEG and CS-Lite

Definition 5. *(**Simulatable Group**) [8] A family of groups $\{G_k\}_{k \in \mathbb{N}}$ is simulatable if there exist two poly-time functions (h, h^{-1}) and a polynomial ℓ, such that the following correctness requirements are met.*

1. $\forall k, \forall r \in \{0,1\}^{\ell(k)}, h(r) \in G_k$.
2. *h^{-1} is probabilistic.* $\forall k, \forall \alpha \in G_k, h^{-1}(\alpha) \in \{0,1\}^{\ell(k)}$.

3. $\forall k, h(h^{-1}(\alpha)) = \alpha$ *for all* $\alpha \in G_k$.

Similarly, the following two security requirements are met, for all probabilistic distinguisher $\mathcal{A}$ *and all* $k \in \mathbb{N}$, *there exists a negligible function* ϵ *such that:* $\Pr[\mathsf{INV}_{\mathcal{A}}(k,(h,h^{-1})) = 1] \leq 1/2 + \epsilon(k)$ *and* $\Pr[\mathsf{IND}_{\mathcal{A}\{G\}}(k,h) = 1] \leq 1/2 + \epsilon(k)$, *where the experiments are defined in Fig. 9.*

$\mathsf{INV}_{\mathcal{A}}(k,(h,h^{-1}))$
1: $b \leftarrow \{0,1\}$
2: $b' \leftarrow \mathcal{A}^{\mathcal{O}_{h,h^{-1},b}}(1^k)$, where oracle $\mathcal{O}_{h,h^{-1},b}$ responds to a query by sampling $r \in \{0,1\}^{l(k)}$, and returning r if $b = 0$ or $h^{-1}(h(r))$ if $b = 1$
3: Output 1 iff $b = b'$

$\mathsf{IND}_{\mathcal{A}\{G\}}(k,h)$
1: $b \in \{0,1\}$
2: $b' \leftarrow \mathcal{A}^{\mathcal{O}_{b,h}}(1^k)$, where $\mathcal{O}_{b,h}$ responds to a query by sampling $r \in \{0,1\}^{l(k)}$ and sampling $h \in G_k$ and returning $h(r)$ if $b = 0$ or h if $b = 1$
3: Output 1 iff $b = b'$

Fig. 9: The Experiments $\mathsf{IND}_{\mathcal{A}}$ and $\mathsf{INV}_{\mathcal{A}\{G\}}$ used to define security for weakly simulatable security of a group family $\{G\}$.

Dent showed that groups in which the DDH assumptions are believed to hold are simulatable.

Lemma 4. *[8] For an infinite sequence of pairs of primes* q *and* p, *where* $p = 2q + 1$, *let* $G_{(p,q)}$ *be the subgroup of* $\mathbb{Z}_p^*$ *of order* q, *then* $\{G_{(p,q)}\}$ *is simulatable group family.*

Theorem 3. *The DEG encryption scheme is weakly simulatable if it is instantiated with a simulatable group family* $\{G_k\}$ *on which the DDH problem is hard.*

Proof. Let (h, h^{-1}) be the efficiently computable functions that exist by the fact that the $\{G_k\}$ is a simulatable group family. For ease of notation in this proof, we assume all functions get the required public parameters (e.g. the public-key) as part of their input. We need to give the two functions (f, f^{-1}) for DEG required by the definition of weakly simulatable. Define $f(x = (x_1, x_2, x_3)) = (h(x_1), h(x_2), h(x_3))$, and $f^{-1}(c = (c_1, c_2, c_3)) =$

$(h^{-1}(c_1), h^{-1}(c_2), h^{-1}(c_3))$. From the properties in the definition of a simulatable group (family), (f, f^{-1}) satisfies the corresponding requirements given in the definition of a weakly simulatable encryption scheme.

We now need to argue the final property of a weakly simulatable encryption scheme: no ppt adversary can distinguish between a valid ciphertext and one sampled via f (which may not be a valid ciphertext). Namely, we need to show $\Pr[\mathsf{DIST}_{\mathsf{DEG}}(k, (f, f^{-1}), \mathcal{A}) = 1] \approx 1/2$

To meet this goal, we design a series of games (or experiments) to show the advantage of an adversary being able to distinguish between legitimate ciphertexts, and sampled outputs of f is negligible. Let $W_{i,k}$ be the random variable output of the security experiment in Game i with security parameter k.

Let **Game 1** be exactly the experiment $\mathsf{DIST}_{\mathsf{DEG}}(k, (f, f^{-1}), \mathcal{A})$.

Let **Game 2** be a modification of Game 1, in which the ciphertext c produced on Line 3 of DIST returns an encryption of 1, independent of the value of m.

Claim 5. $\{W_{1,k}\}_k \approx_c \{W_{2,k}\}_k$.

Proof. Follows from CPA security of the DEG encryption scheme. □

Let **Game 3** be a modification of **Game 2** in which again in Line 3 of DIST, the value W computed in Line 2 of the DEG encryption algorithm is computed as follows: $W \leftarrow g^{r'}$, where $r' \in \mathbb{Z}_q$ (instead of $W \leftarrow X_1^y$).

Claim 6. $\{W_{2,k}\} \approx_c \{W_{3,k}\}$

Proof. (sketch) If there is any distinguisher D that can distinguish $\{W_{2,k}\}$ from $\{W_{3,k}\}$ with reasonable probability, then one can use D be used to build a DDH distinguisher D'. In particular, D' when given either a tuple $(g, g^{x_1}, g^y, g^{x_1 y})$ or $(g, g^{x_1}, g^y, g^{r'})$, can choose a random x_2, simulate a pk for the DEG scheme, and use x_2 and the provided information to compute an appropriate encryption for a perfect simulation of the either **Game 2** or **Game 3**. We can then simulate D, and use the result to break DDH. □

Let **Game 4** be a modification of **Game 3** where the value U in Line 2 of the DEG encryption algorithm is computed as $U = g^{r''}$ for randomly selected $r'' \in \mathbb{Z}_q$.

Claim 7. $\{W_{3,k}\} \approx_c \{W_{4,k}\}$

Proof. (sketch) The proof of this claim parallels that of the previous. If there is a distinguisher D of $\{W_{2,k}\}$ and $\{W_{3,k}\}$ then to can be used to build a DDH distinguisher D'. In particular, D' given a tuple $(g, g^{x_2}, g^y, g^{x_2 y})$ or $(g, g^{x_2}, g^y, g^{r'})$, it choose a random x_1, simulates a pk for the DEG scheme, and use x_1 and the provided information to provide an appropriate encryption, for a perfect simulation of the either **Game 3** or **Game 4**. □

In **Game 4** each of the components of the ciphertext in Line 3 of DIST is now random group element. In **Game 5** we replace these random group elements with random output from the group element sampling algorithm h, which due to the simulatable group properties are indistinguishable from random group elements. Specifically, in **Game 5** in Line 5 of DIST we return the "ciphertext" $(Y = h(r_1), W = h(r_2), U = h(r_3))$, for randomly chosen $r_1, r_2, r_3 \in \{0,1\}^{l(k)}$.

Claim 8. $\Pr\{W_{4,k}\} \approx_c \{W_{5,k}\}$

Proof. Follows immediately from simulatable group properties of G and h. □

We note that by the definition of f, the output of the encryption algorithm in **Game 5** Line 5 of DIST is an identically, but independently distributed random variable as the one output on Line 4 of DIST (i.e., $f(r_1, r_2, r_3)$ for randomly chosen r_1, r_2, r_3). It is clear that $\Pr[W_{k,5} = 1] = 1/2$.

Therefore, since we can combine all the claims to show that $\Pr[W_{k,1} = 1] \approx_c \Pr[W_{k,5} = 1] = 1/2$ we conclude that DEG is weakly simulatable. □

Theorem 4. *The Cramer-Shoup lite encryption scheme is weakly simulatable if it is instantiated on a simulatable group G on which the DDH problem is hard.*

Proof. Similar to the proof of Theorem 3. □

B Instantiating SPA secure schemes

Definition 6 (DHK_ℓ Assumption (modification of [1]))**.** *Let G be a prime-order-group generator. Let Crt_G be an algorithm that has access to an oracle, takes an ℓ sequence of two primes and two group elements, and returns nothing. Let Ext_G be an algorithm that takes a pair of group elements and some state information, and returns an exponent and a new state. We call Crt_G a DHK_ℓ-adversary and Ext_G a DHK_ℓ-extractor.*

$\mathrm{DHK}_\ell(G, \mathsf{Crt}_G, \mathsf{Ext}_G, k)$

1: $(p_i, q_i, g_i \leftarrow G(1^k);\ a_i \leftarrow \mathbb{Z}_{q_i};\ A_i \leftarrow g_i^{a_i} \mod p_i)_{i \in \ell(k)}$
2: Let $R[\mathsf{Crt}_G]$ and $R[\mathsf{Ext}_G]$ be randomly selected strings for Crt_G and Ext_G
3: $st \leftarrow ((p_i, q_i, g_i, A_i)_{i \in [\ell(k)]}, R[\mathsf{Crt}_G])$
4: **while** Simulate $\mathsf{Crt}_G((p_i, q_i, g_i, A_i)_{i \in [\ell(k)]}; R[\mathsf{Crt}_G])$ **do**
5: **if** Crt_G queries (i, B, W) **then**
6: $(b, st) \leftarrow \mathsf{Ext}_G((i, B, W), st; R[\mathsf{Ext}_G])$
7: **if** $W \equiv B^{a_i} \mod p_i$ and $B \not\equiv g_i^b \mod p_i$ **then** Return 1
8: **else** Return b to Crt_G
9: Return 0

Fig. 10: DHK_ℓ: An Extension to the DHK Definition

We say that G satisfies the DHK_ℓ assumption if for every polynomial-time Crt_G there exists a polynomial-time Ext_G and negligible function μ, s.t. for all $k \in \mathbb{N}$: $\Pr[DHK_\ell(G, \mathsf{Crt}_G, \mathsf{Ext}_G, k) = 1] \leq \mu(k)$.

Our modification to DHK_ℓ versus the definition in [1] requires that the ciphertext creator be able to generate ciphertexts relative to a polynomial number of randomly chosen public-keys. It seems reasonable to conjecture that any extractor that could extract exponents with respect to single value $A = g^a$, could do so efficiently for many A_i. We now argue that DEG is $\mathbf{sPA1}_\ell$ secure under the DHK_ℓ assumption.

Theorem 5. *For any polynomial ℓ, if the DHK_ℓ assumption holds and the DEG scheme is instantiated with a simulatable group family $\{G_k\}$, then the DEG scheme is $\mathbf{sPA1}_\ell$ secure.*

Proof. We need to show that for any adversary Crt there exists an extractor Ext that can decrypt its queries flawlessly. Ext receives $(pk_i = \langle p_i, q_i, g_i, A_i, \hat{A}_i \rangle)_{i \in \ell(k)}$ and $R[\mathsf{Crt}]$ as state information. Then Ext builds the DHK_ℓ adversary Crt_G that runs the $\mathbf{sPA1}_\ell$ adversary Crt internally and simulates the $\mathbf{sPA1}_\ell$ experiment for it. Crt_G receives $(p_i, q_i, g_i, A_i)_{i \in \ell(k)}$ and its random coins from Ext and parses its random coins as $(f_G^{-1}(\hat{A}_i))_{i \in [\ell(k)]} \mid R[\mathsf{Crt}]$ (prepared by Ext where $\hat{A}_i$ is a random group element in G). Notice that since G, the group from which $\hat{A}_i$ is sampled from, is simulatable, it follows that $f_G^{-1}(\hat{A}_i)$ is indistinguishable from random bits and should have indistinguishable effect on the output of the extraction. Because Crt_G is a DHK_ℓ adversary, therefore there exists an extractor for it Ext_G. For

each $i \in [\ell(k)]$, Crt_G sets $pk_i \leftarrow (p_i, q_i, g_i, A_i, \hat{A}_i)$. Crt_G then runs Crt on $(pk_i)_{i\in[\ell(k)]}$ and the random coins $R[\mathsf{Crt}]$ until Crt halts, answering Crt's queries as follows: upon receiving the query $C = (i, Y, W, U)$ from Crt, Crt_G submits (i, Y, W) to the DHK_ℓ extractor Ext_G. The DHK_ℓ extractor Ext_G returns the value b. If $Y \not\equiv g_i^b \mod p_i$ or $W \not\equiv A_i^b \mod p_i$ then Crt_G returns $\perp$ as the answer to this query, otherwise Crt_G computes $M \leftarrow U \cdot (\hat{A}_i{}^b)^{-1} \mod p_i$ and return the result to Crt. Notice that since Crt_G is a DHK_ℓ adversary, the extractor Ext_G should return the right answer to the queries Crt_G submits. Since Ext makes a mistake in answering Crt's queries only when there is a mistake in Ext_G's answers to Crt_G's queries, we conclude that Ext also returns the right decryption to the queries submitted by Crt and is an extractor for it.

□

Theorem 6. *For any polynomial ℓ, The CS-Lite scheme is $\mathbf{sPA1}_\ell$ secure if the followings hold: i) the DHK_ℓ assumption, and ii) CS-Lite is instantiated with a simulatable group family $\{G_k\}$.*

Proof. Similar to the proof of Theorem 5.

□