On the CCA Compatibility of Public-Key Infrastructure

Dakshita Khurana[*] Brent Waters[†]

July 9, 2021

Abstract

In this work, we study the *compatibility* of any key generation or setup algorithm. We focus on the specific case of encryption, and say that a key generation algorithm KeyGen is X-compatible (for $\mathsf{X} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$) if there exist encryption and decryption algorithms that together with KeyGen, result in an X-secure public-key encryption scheme.

We study the following question: Is every CPA-compatible key generation algorithm also CCA-compatible? We obtain the following answers:

- Every sub-exponentially CPA-compatible KeyGen algorithm is $\mathsf{CCA1}$-compatible, assuming the existence of hinting PRGs and sub-exponentially secure keyless collision resistant hash functions.
- Every sub-exponentially CPA-compatible KeyGen algorithm is also $\mathsf{CCA2}$-compatible, assuming the existence of non-interactive CCA2 secure commitments, in addition to sub-exponential security of the assumptions listed in the previous bullet.

Here, sub-exponentially CPA-compatible KeyGen refers to any key generation algorithm for which there exist encryption and decryption algorithms that result in a CPA-secure public-key encryption scheme *against sub-exponential adversaries.*

This gives a way to perform CCA secure encryption given any public key infrastructure that has been established with only (sub-exponential) CPA security in mind. The resulting CCA encryption makes black-box use of the CPA scheme and all other underlying primitives.

[*]University of Illinois Urbana-Champaign. Email: `dakshita@illinois.edu`
[†]University of Texas at Austin and NTT Research. Email: `bwaters@cs.utexas.edu`

Contents

1 Introduction

Any public-key encryption scheme enables a receiver to recover the encrypted message only if they know a secret key corresponding to their public key. But what if the receiver only ever published a verification key for a digital signature scheme for which they possessed a signing key? Or published a hard puzzle for which they possessed a solution?

This question was one of the original motivations for the study of witness encryption. Garg et al. [GGSW13] showed that it is possible to encrypt a message so that it can only be opened by a recipient who knows an NP witness, assuming the existence of an appropriate witness encryption scheme. Put differently, assuming an appropriate witness encryption, almost any KeyGen algorithm that outputs a hard-to-invert string and a corresponding secret (such as a verification and signing key pair for a signature scheme) can be used to derive CPA-secure public key encryption.

In this work, we generalize this study. We study the notion of *compatibility* of any key generation or setup algorithm, while focusing on the specific case of encryption schemes. Here, recall that semantic security of (public key) encryption in [GM84] was only the first step towards formalizing security of encryption schemes. Semantic security, or equivalently indistinguishability-based security against chosen plaintext attacks (CPA) requires that encryptions of every pair of plaintexts appear indistinguishable to a computationally bounded attacker. Unfortunately, starting with the attacks of Bleichenbacher [Ble98] against PKCS#1, it was quickly realized that systems that *only* satisfy CPA security often fail in practice. As a result, security against adaptive chosen ciphertext attacks (or, CCA security) has been accepted as the standard requirement from encryption schemes that need to withstand active attacks [CS98, NY90, DDN00, RS92]. This guarantees security even against attackers that make oracle decryption queries to keys they do not have. If the adversary only has access to a decryption oracle *before* obtaining the challenge ciphertext, the resulting scheme is said to be CCA1 secure. On the other hand, if the adversary has access to the decryption oracle both before and after obtaining the challenge ciphertext, the resulting scheme is CCA2 secure.

We investigate whether arbitrary setup of KeyGen algorithms can be used to derive *CCA-secure* schemes. We will say that a key generation algorithm KeyGen is X-compatible (for $\mathsf{X} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$) if there exist encryption and decryption algorithms that together with KeyGen, result in an X-secure public-key encryption scheme. As already discussed, the existence of (extractable) witness encryption suffices to prove CPA-compatibility for many non-trivial KeyGen algorithms. The focus of our work is to take a closer look at CCA-compatibility. We note that a similar notion of compatibility of key generation schemes was studied in [BKSW18] in the context of functional encryption. Informally, [BKSW18, GGH$^+$16] showed that (assuming indistinguishability obfuscation) given any public key encryption scheme, one can retain only the setup algorithm and design new encryption, decryption and function secret key generation algorithms to derive a functional encryption scheme scheme.

In this work, we analyze what it takes for a KeyGen algorithm to be CCA-compatible. Our primary result stated informally, is the following:

It is always possible to get CCA secure encryption from any KeyGen *procedure that gives rise to (sub-exponentially secure) CPA encryption.*

Combined with the CPA-compatibility of non-trivial KeyGen, this also implies CCA-compatibility of many non-trivial KeyGen algorithms.

This also means that we can always achieve CCA security using keys for cryptosystems that were originally deployed for CPA mode, without having to modify the public key. This would allow parties with access to public key infrastructures that have been established with only CPA security in mind, to use these infrastructures to perform CCA secure encryption instead. For instance, over the years, multiple encryption schemes have been developed that satisfy IND-CPA security alone. A recent example that gained some popularity is the messaging service telegram, that supplies end-to-end encryption using a new protocol employing AES, RSA, and Diffie-Hellman key exchange. Recently, [JO16] showed that this protocol is not IND-CCA secure. Our result ensures that (under reasonable cryptographic assumptions) careful participants can use the same underlying infrastructure to engage in encrypted communication without worrying about CCA2 attacks. Alternatively, suppose a user or an organization sets up what is supposed to be a CCA secure system, but the underlying computational assumption turned out to be false. For example, perhaps an attack on DDH was found in a specific group [CSV20], and the scheme is somehow adjusted to recover CPA security. Then, the scheme can also be adjusted to recover CCA security (under our assumptions), with the same infrastructure as that used for CPA security. In these settings, while one could potentially ask users to switch to using a new key from the same system, changing an entire public key infrastructure would be far more cumbersome. We note that simple key encapsulation strategies would be insufficient: for example, sampling the key for a CCA secure encryption scheme and encapsulating it using a key for the original CCA-insecure infrastructure would not lead to the resulting ciphertext being CCA secure.

Preliminary Solutions in Idealized Models. In idealized models, there are known methods that implicitly allow one to obtain CCA security from any CPA-compatible KeyGen algorithm. For instance, in the Random Oracle model, the famous Fujisaki-Okamoto transform [FO13] converts any CPA secure encryption scheme to a CCA secure one, with the same KeyGen algorithm. We are interested in whether a similar effect can be achieved in the plain model.

A natural approach without a random oracle would be to leverage a common reference string (CRS) and implement the Naor-Yung methodology [NY90, DDN00] using simulation-sound NIZKs. We recall that the Naor-Yung (CCA secure) encryption of a message typically consists of two encryptions, under independent public keys, of the same message; and a simulation-sound NIZK proof that both ciphertexts encrypt the same message. If implemented naively, it appears that the KeyGen algorithm for the resulting CCA mode would have to output *two* independent public keys corresponding to the underlying CPA secure scheme. Even if we found a method to get rid of the second key, this still requires participants to place their trust in a central setup assumption to enable the (simulation-sound) NIZK. Given this state of affairs, we ask if it is possible to obtain CCA secure encryption by relying on the KeyGen algorithm of any CPA secure encryption scheme:

- in the plain model without assuming setup, CRS, or a random oracle,
- with black-box use of the CPA scheme (and additional primitives),
- and while making the weakest possible cryptographic assumptions.

Our Results. We take a novel approach to obtain a plain model solution that makes black-box use of the CPA scheme, and does not resort to NIZK (or NIWI) style assumptions. Specifically, we demonstrate CCA1-compatibility of any sub-exponentially CPA-compatible KeyGen algorithm while making black-box use of hinting PRGs and sub-exponential keyless collision resistant hash functions. We also demonstrate CCA2-compatibility of any sub-exponentially CPA-compatible

KeyGen algorithm while additionally making black-box use of non-interactive CCA secure commitments. Such commitments were recently obtained [GKLW20] based on black-box use of subexponential one-way functions in BQP, and sub-exponential quantum-hard one-way functions, in addition to the assumptions listed above. Alternatively, these can be based on sub-exponential time-lock puzzles [LPS17, GKLW20] in addition to the assumptions listed above. We informally summarize our results below.

Informal Theorem 1. Every sub-exponentially CPA-compatible KeyGen algorithm against non-uniform adversaries is also CCA1-compatible against uniform adversaries, assuming the existence of hinting PRGs and sub-exponential keyless collision-resistant hash functions against uniform adversaries.

Informal Theorem 2. Every sub-exponentially CPA-compatible KeyGen algorithm against non-uniform adversaries is also CCA2-compatible against uniform adversaries, assuming the existence of sub-exponential hinting PRGs, sub-exponential keyless collision-resistant hash functions against uniform adversaries and sub-exponential non-interactive CCA secure commitments.

2 Our Technique

2.1 Background: A variant of Koppula-Waters [KW19]

Our starting point is a variant of the recent Koppula-Waters [KW19] approach to achieving CCA1 secure encryption based on CPA secure encryption and a new primitive they introduced, called a *hinting PRG.* A hinting PRG satisfies the following property: for a uniformly random short seed s, the matrix M obtained by first expanding $\mathrm{PRG}(s) = z_0 z_1 z_2 \dots z_n$, sampling uniformly random $v_1 v_2 \dots v_n$, and setting for all $i \in [n]$, $M_{s_i,i} = z_i$ and $M_{1-s_i,i} = v_i$, should be indistinguishable from a uniformly random matrix. Hinting PRGs are known based on CDH, LWE [KW19]. (One can also pursue a similar path using any circular secure symmetric key encryption [KMT19] in lieu of the Hinting PRG.) Koppula and Waters [KW19] also require the CPA scheme to have two properties. First, the scheme should have perfect decryption correctness for most public/secret keys and second, any ciphertext should be decryptable given the encryption randomness.

Now, the KeyGen algorithm of the CCA1 scheme constructed by [KW19, KMT19] executes the CPA KeyGen setup *twice* to obtain two independent public/secret key pairs, denoted by (pk_0, sk_0) and (pk_1, sk_1). Additionally, the CCA1 KeyGen algorithm samples and outputs the public parameters pp of an equivocal commitment scheme. To encrypt a message m, the encryption algorithm chooses a seed $s \leftarrow \{0,1\}^n$ and computes $H(s) = z_0 z_1 \dots z_n$, where H is a hinting PRG. It uses z_0 to mask the message m; that is, it computes $c = m \oplus z_0$. The remaining ciphertext will contain n signals that help the decryption algorithm to recover s bit by bit, which in turn will allow it to compute z_0 and hence unmask c.

The i^{th} signal (for each $i \in [n]$) has three components $c_{0,i}, c_{1,i}, c_{2,i}$. If the i^{th} bit of s is 0, then $c_{0,i}$ is an encryption of a random string y_i using the public key pk_0 and randomness z_i, and $c_{1,i}$ is an encryption of y_i using pk_1 (encrypted using true randomness). If the i^{th} bit of s is 1, then $c_{0,i}$ is an encryption of y_i using public key pk_0 (encrypted using true randomness), $c_{1,i}$ is an encryption of y_i using public key pk_1 and randomness z_i. In both cases, $c_{2,i}$ is an equivocal commitment to s_i using randomness y_i. As a result, half the ciphertexts are encryptions with fresh randomness, while the remaining are encryptions with blocks of the hinting PRG output being used as randomness, and the positioning of the random/pseudorandom encryptions reveals the seed s.

The decryption algorithm first decrypts each $c_{0,i}$ (using secret key is sk_0) to obtain $y_1 y_2 \dots y_n$. It then checks if $c_{2,i}$ is an equivocal commitment to 0 with randomness y_i. If so, it guesses that $s_i = 0$, else it guesses that $s_i = 1$. With this estimate for s, the decryption algorithm can compute $H(s) = z_0 z_1 \dots z_n$ and then compute $c \oplus z_0$ to recover the message m. The decryption algorithm needs to enforce additional checks to prevent malicious decryption queries (made during the CCA1 experiment). In particular, the decryption algorithm needs to check that the guess for s is indeed correct. It conducts the following checks and outputs $z_0 \oplus c$ if they all pass.

- If the i^{th} bit of s is guessed to be 0, then the decryption algorithm checks that $c_{0,i}$ is a valid ciphertext - it simply re-encrypts y_i using randomness z_i and checks if this equals $c_{0,i}$. Recall that the decryption procedure, before guessing the i^{th} bit of s to be 0, also checks that $c_{2,i}$ is a commitment to 0 with randomness y_i.

- If the i^{th} bit of s is guessed to be 1, then the decryption algorithm first recovers the message underlying ciphertext $c_{1,i}$. Note that $c_{1,i}$ should be encrypted using randomness z_i, hence using z_i, one can recover message $\widetilde{y}_i$ from $c_{1,i}$ (using the randomness recovery property of the PKE scheme). It then re-encrypts $\widetilde{y}_i$ and checks if it is equal to $c_{1,i}$, and also checks that $c_{2,i}$ is a commitment to 1 with randomness $\widetilde{y}_i$.

Inadequacies of this transformation. At this point, we are far from having established CCA1 compatibility of arbitrary CPA infrastructure due the following reasons:

1. The transformation described so far crucially uses equivocal commitments, which require trusted setup or a common reference/random string, and this is something that we would like to avoid.

2. This transformation makes use of two public keys, and we are only guaranteed to get *a single key* from existing setup.

In fact, achieving CCA2 security is even more complex: in particular, the CCA2 setup in the transformation of [KW19] must also contain pairwise independent hash functions $h_1, h_2, \dots, h_n$. These are used to prevent the adversary from mauling the challenge ciphertext into related ciphertexts and querying the decryption oracle on these ciphertexts.

In the coming section, we discuss how to achieve CCA1 compatibility by eliminating the two problems listed above. In the section after that, we discuss the more complex case of CCA2 compatibility.

2.2 Techniques for CCA1 Compatibility

To address the first item listed above, we rely on equivocal commitments without setup, that satisfy binding against uniform adversaries. The resulting CCA1 compatibility is also established against uniform PPT adversaries. We briefly recall how such equivocal commitments can be obtained based on keyless collision resistant hash functions against uniform adversaries [DPP93, HM96, BKP18, GKLW20]: the commit algorithm samples a uniformly random seed for a strong extractor, $g \leftarrow \{0,1\}^\kappa$ and a value v in the domain of a sufficiently compressing keyless collision resistant hash function. A commitment to a bit b is given by the string $H(v), (\mathsf{Ext}(g, v) \oplus b)$.

We use the resulting commitment scheme to generate $c_{2,i}$ values in the outline described above. Note that this commitment scheme cannot be efficiently equivocated by uniform adversaries, since that would lead to an efficient uniform algorithm that finds collisions in the hash function H. On the other hand, our proof of security will rely on the fact that most strings in the support of the commitment can be non-uniformly equivocated. Next, we discuss how to argue security when using these equivocal commitments in the transformation described above.

Arguing Security. To argue security, first observe that the equivocal commitment satisfies computational binding against PPT adversaries, which makes it infeasible for a CCA1 adversary to query the challenger on *ambiguous* ciphertexts that pass the decryption checks but potentially decrypt to different values under sk_0 and sk_1. This is because for such ciphertexts, for some $i \in [n]$, the component $c_{2,i}$ is both a commitment to 0 with randomness y_i recovered from $c_{0,i}$ and a commitment to 1 with randomness $\widetilde{y}_i$ recovered from $c_{1,i}$: clearly violating the binding property of the commitment scheme.

At the same time, the equivocality of the commitment enables the challenger to set for every $i \in [n]$, the values $c_{2,i}$ that are both commitments to 0 with randomness y_i and 1 with randomness $\widetilde{y}_i$. Next, via a careful hybrid argument that relies on perfect correctness of the encryption scheme, the binding property of the equivocal commitment and CPA security of the public key encryption scheme, the challenge ciphertext can be modified and made ambiguous: this means that in the challenge ciphertext for every $i \in [n]$, $c_{0,i}$ is an encryption of y_i, $c_{1,i}$ is a non-interactive commitment to $\widetilde{y}_i$, $c_{2,i}$ an equivocal commitment: i.e., a commitment to 0 with randomness y_i and to 1 with randomness $\widetilde{y}_i$.

At a very high level, this involves changing values encrypted under $c_{0,i}$ and $c_{1,i}$ by relying on CPA security of the encryption scheme. Values encrypted under $c_{1,i}$ can be modified relatively easily because only the secret key sk_0 is used to perform decryption queries. Arguing security when changing values encrypted under $c_{0,i}$ requires more care: in particular, such an argument is possible only if sk_0 is no longer used to answer the adversary's decryption queries. Therefore we first switch to using an alternative decryption strategy that relies on sk_1 instead of sk_0 to decrypt the adversary's ciphertexts. Unambiguity of the adversary's ciphertexts helps ensure that alternative decryption yields the same outputs as the original decryption strategy. When using alternate decryption, it becomes possible to change values encrypted under $c_{0,i}$ since sk_0 is no longer being used.

At the end of this argument, information about the hinting PRG seed s has *almost* been removed from the ciphertext, except that for all i where $s_i = 0$, $c_{s_i,i}$ is encrypted using randomness r_i which came from running the hinting PRG on s; whereas $c_{1-s_i,i}$ is encrypted using uniform randomness. These can all be replaced with uniformly random values by the property of the hinting PRG, thereby eliminating all information about s, and therefore m, from the ciphertext.

So far, the construction and security argument also relied on the use of *two* public/private key pairs. But as already pointed out, a CPA-compatible KeyGen algorithm will output a single public and private key. Next, we discuss how to eliminate the need for the second key.

Removing the second key via Non-interactive Commitments. Here, we begin by observing that the actual decryption algorithm only makes use of the secret key sk_0, and does not need the second secret key sk_1. It recovers messages underlying $c_{1,i}$ for all $i \in [n]$ where $s_i = 1$, using the *randomness* z_i that was supposedly used to create $c_{1,i}$.

As a result, the actual decryption algorithm does not need to *efficiently decrypt* $c_{1,i}$ and has no use for the secret key sk_1. Therefore, we eliminate the need for the second public key by setting the strings $c_{1,i}$ to be *non-interactive perfectly binding commitments* that do not require any public keys or public parameters, and where the committed message can be efficiently recovered given the randomness used to commit. Lombardi and Schaeffer [LS19] showed that such commitments can be obtained from any perfectly correct public-key encryption scheme. Specifically, we modify the encryption algorithm so that if $s_i = 0$, $c_{1,i}$ is a non-interactive commitment to 0^n using true randomness, and if $s_i = 1$, then $c_{1,i}$ is a non-interactive commitment to a random string x_i using randomness z_i. The remaining parts $\{c_{0,i}, c_{2,i}\}_{i \in [n]}$ will remain unmodified.

Now recall that the security argument outlined above points to an alternative decryption strategy that does actually use sk_1, instead of sk_0, to efficiently decrypt the adversary's ciphertexts. However, this alternative decryption algorithm is only used in a few hybrids in the proof of security, and when using non-interactive commitments, we allow these hybrids to *inefficiently* recover the values committed under $c_{1,i}$ by running an exponential time brute-force algorithm that checks all possible randomnees values that could potentially be used to build $c_{1,i}$.

In order to make the hybrid strategy still go through, we rely on complexity leveraging: we set security parameters so that all other primitives are secure against adversaries that can run in time large enough to execute the brute-force algorithm that recovers values committed under $c_{1,i}$ for $i \in [n]$. Specifically, we assume that the CPA encryption scheme to be upgraded has security parameter k and is 2^{k^e} secure for some constant $0 < e < 1$. We also assume that the keyless collision-resistant hash function responsible for the binding property of the equivocal commitments is 2^{k^ϵ}-secure for some constant $0 < \epsilon < 1$, and we set the security parameter for the non-interactive commitment to be $k^{\mathsf{min}(e,\epsilon)}$.

Additional Details of the Proof. We now provide additional details on the proof of CCA1 security of the resulting scheme. Recall that in the CCA1 security game, the adversary is allowed access to a decryption oracle before the challenge phase, where the adversary outputs m_0, m_1 and then obtains an encryption of m_b for b sampled uniformly at random.

We develop a sequence of hybrid experiments where the decryption oracle as well as the challenge ciphertext are modified in small increments, and where the first hybrid corresponds to providing the adversary access to the actual decryption oracle together with an encryption of m_b and the last one corresponds to providing the adversary access to the actual decryption oracle together with an encryption of uniform randomness.

The very next hybrid experiment is an exponential time hybrid that samples equivocal commitments $\{c_{2,i}\}_{i \in [n]}$ for the challenge ciphertext, together with randomness $\{y_{0,i}\}_{i \in [n]}$ and $\{y_{1,i}\}_{i \in [n]}$ that can be used to equivocally open these commitments to 0 and 1 respectively.

The third hybrid additionally modifies the components $c_{1,i}$ to "drown" out information about s via noise. In particular, while in the real game, the values $c_{1,i}$ are always commitments to $y_{s_i,i}$, in the challenge ciphertext these values are modified to become commitments to $y_{1,i}$, irrespective of what s_i is. On the other hand, the values $c_{0,i}$ remain encryptions of $y_{s_i,i}$, exactly as in the real experiment. In spite of the fact that equivocation takes exponential time, the proof of indistinguishability between this hybrid and the previous one does not need to rely on an exponential time reduction. Instead, we observe that the equivocal commitment strings $\{c_{2,i}\}_{i \in [n]}$ together with their openings can be fixed non-uniformly and independently of the strings $c_{1,i}$, and therefore these hybrids can be proven indistinguishable based on non-uniform hiding of the non-interactive commitment scheme. Since we must carefully manipulate the randomness used for $\{c_{1,i}\}_{i \in [n]}$ in both games, this hybrid requires a delicate argument.

The fourth hybrid modifies the decryption oracle so that instead of decrypting using the secret key of the public key encryption scheme, decryption is performed by running in time exponential in the security parameter of the commitment scheme (specifically, in time $2^{k^{\mathsf{min}(e,\epsilon)}}$) and performing a brute-force search for the randomness used to create the commitments $\{c_{1,i}\}_{i\in[n]}$. This hybrid is only indistinguishable from the previous one if an adversary cannot find ciphertexts that decrypt differently when using the secret key of the encryption scheme versus the brute-force algorithm discussed above. This hybrid requires a subtle argument that relies on the fact that no adversary can query the decryption oracle with "ambiguous" ciphertexts, in spite of being provided such ciphertexts in the challenge phase. Specifically, we crucially use the fact that the adversary does not observe any equivocations before obtaining the challenge ciphertext, and therefore cannot query the decryption oracle with any "ambiguous" ciphertexts (as this would lead to the adversary breaking binding of the equivocal commitment). This is the primary reason that we only obtain CCA1 security.

In the fifth hybrid, some of the challenge ciphertext values, that are independent of the message being encrypted, are chosen ahead of time. This maneuver helps us with the sixth hybrid, where in the challenge ciphertext, information about the PRG seed s is removed from the ciphertext components $\{c_{1,i}\}_{i\in[n]}$, making them all encryptions of $y_{0,i}$ instead of being encryptions of $y_{s_i,i}$. Again, since we must carefully manipulate the randomness used for $\{c_{0,i}\}_{i\in[n]}$ in both games, this hybrid requires a delicate argument.

In the seventh hybrid, we modify the decryption oracle again to go back to using the secret key of the public key encryption scheme to decrypt. Note that the only remaining information about s is in the *randomness* used to obtain $\{c_{i,0}, c_{i,1}\}_{i\in[n]}$ in the challenge ciphertext. In the seventh and eighth hybrids, we carefully re-order the randomness and rely on the security of the hinting PRG to switch to using uniform randomness everywhere. This eliminates all information about s and therefore about the message being encrypted in the challenge ciphertext.

2.3 Techniques for CCA2 Compatibility

We observe that the key barrier to proving CCA2 security in the hybrid arguments outlined above is the specific hybrid that *modifies the challenge ciphertext so it contains a commitment to both a* 0 *and a* 1. Given such a ciphertext, in a CCA2 game, an adversary could generate new strings that are a commitment to both a 0 and a 1, and use them to create ambiguous ciphertexts. Arguing that this cannot happen requires us to develop a much deeper technical toolkit.

Our first insight is that the requirement that an adversary, given an ambiguous ciphertext, be unable to generate *additional* ambiguous ciphertexts is reminiscent of *non-malleability*. As such, we will rely on non-interactive non-malleable (more precisely, CCA secure) commitments without setup. Up until recently, there were perceived strong barriers to obtaining non-malleable commitmens with less than 3 rounds of interaction [Pas16]. But a sequence of recent works obtained two round [KS17, LPS17] and even non-interactive [LPS17, BL18, KK19, GKLW20] based on well-studied sub-exponential hardness assumptions. In particular, a recent work [GKLW20] obains black-box non-interactive non-malleable (and in fact CCA2 secure) commitments assuming kelyess collision resistant hash functions, against uniform adversaries.

Relying on CCA2 Secure Commitments. We now discuss modifications to the CCA1 transformation discussed in the previous section. Specifically, we will replace the non-interactive commitment (used to generate ciphertext components $\{c_{1,i}\}_{i\in[n]}$) in the construction outlined above, with a CCA2 secure commitment. Intuitively, using CCA2 secure commitments ensures that no matter how we change the $\{c_{1,i}\}_{i\in[n]}$ components in the challenge ciphertext, the corresponding $\{c_{1,i}\}_{i\in[n]}$ components in the adversary's decryption queries do not change (except in a computationally indistinguishable way). Proving that the resulting protocol is actually a CCA2 secure encryption scheme, is much trickier. We encounter several technical barriers in this process, which we discuss below.

Arguing Security. Recall that in the CCA2 security game, the adversary is allowed access to a decryption oracle both before and after the challenge phase, where the adversary outputs m_0, m_1 and then obtains an encryption of m_b for b sampled uniformly at random.

We will consider a sequence of hybrid experiments similar to the CCA1 setting, where the decryption oracle as well as the challenge ciphertext are modified in small increments. The first hybrid corresponds to providing the adversary access to the actual decryption oracle together with an encryption of m_b and the last one corresponds to providing the adversary access to the actual decryption oracle together with an encryption of uniform randomness.

The very next hybrid experiment, just like the CCA1 setting, is an exponential time hybrid that samples equivocal commitments $\{c_{2,i}\}_{i\in[n]}$ for the challenge ciphertext, together with randomness $\{y_{0,i}\}_{i\in[n]}$ and $\{y_{1,i}\}_{i\in[n]}$ that can be used to equivocally open these commitments to 0 and 1 respectively.

The third hybrid additionally modifies the components $c_{1,i}$ to "drown" out information about s via noise. In particular, while in the real game, the values $c_{1,i}$ are always commitments to $y_{s_i,i}$, in the challenge ciphertext these values are modified to become commitments to $y_{1,i}$, irrespective of what s_i is. On the other hand, the values $c_{0,i}$ remain encryptions of $y_{s_i,i}$, exactly as in the real experiment. At this point, the proof of indistinguishability of hybrids already significantly diverges from the CCA1 setting. Specifically, the proof of indistinguishability between this hybrid and the previous one, in the CCA1 setting, relied on non-uniform security of the non-interactive commitment – in order to perform the exponential time computation needed to equivocate the hash function. Here, we would ideally like to rely on CCA secure commitments which are potentially only secure against uniform adversaries (eg, the black-box construction in [GKLW20] which is only secure against uniform adversaries). One option could be to assume that the CCA2 commitment is "hard" against adversaries running in time that is sufficient to compute openings of the equivocal commitment.

In the fourth hybrid, we would like to modify the decryption oracle so that instead of decrypting using the secret key of the public key encryption scheme, decryption is performed by running in time exponential in the security parameter of the commitment scheme (specifically, in time $2^{k^{\mathsf{min}(e,\epsilon)}}$) and performing a brute-force search for the randomness used to create the commitments $\{c_{1,i}\}_{i\in[n]}$. This hybrid is indistinguishable from the previous one only if an adversary cannot find ciphertexts that decrypt differently when using the secret key of the encryption scheme versus the brute-force algorithm discussed above: in other words if the adversary cannot query the oracle with "ambiguous" ciphertexts.

This is where the CCA2 setting diverges most significantly from the CCA1 setting. In the CCA1 setting, we could prove that the adversary does not make ambiguous decryption queries by relying on uniform binding of the equivocal commitment, but this is no longer true in the CCA2 setting. Specifically, we need to rule out an adversary that given ambiguous ciphertexts, creates new ones.

Therefore, in the proof, we will now have to rely on CCA2 commitments to maintain an invariant across all the hybrids discussed above. The invariant is as follows: except with negligible probability, the adversary does not make any oracle query for which there exists some $i \in [n]$ such that the components $(c_{0,i}, c_{1,i})$ encrypt/commit to two different openings of the string $c_{2,i}$.

To ensure that this invariant holds in the initial hybrid that corresponds to the real CCA2 experiment, we will use any adversary that breaks the invariant to contradict the binding property of the equivocal commitment. The corresponding reduction would have to *extract* two openings for the same equivocal commitment string, from a decryption query provided by the adversary. In particular, these openings will actually be the plaintexts underlying the ciphertext $c_{0,i}$ and the commitment string $c_{1,i}$. Extracting these two openings involves decrypting $c_{0,i}$ under sk_0, and brute-force breaking the CCA2 commitment string $c_{1,i}$. This use of brute force necessitates that the binding property of the equivocal commitment be hard to break even in time that is sufficient to break the CCA2 commitment.

But recall that arguing indistinguishability for the third hybrid actually required the exact opposite property: that the CCA2 commitment be hard to break even by adversaries running in time that is sufficient to compute openings of the equivocal commitment. It appears that we are at an impasse here, since we need the equivocal commitment and the CCA2 commitment to each take longer time to break than the other. One way to resolve this is to rely on a non-uniform reduction to argue indistinguishability between the second and third hybrids. But recall that the underlying black-box CCA commitments of [GKLW20] achieve only uniform security, at least when relying on on keyless collision resistant hash functions against uniform adversaries.

Fortunately for us, it turns out that [GKLW20] prove a much stronger property than uniform CCA security – they actually establish computation enabled CCA security. The computation enabled property allows the attacker to submit a randomized turing machine P at the beginning of the game. The challenger can run the program P and output the result for the attacker at the beginning of the game: crucially, the running time of P can be much larger than the uniform running time allowed to the adversary. This added property actually achieves a flavor of non-uniformity that helps our argument go through, by allowing us to perform special heavy computation at the beginning of the reduction between hybrids 2 and 3, while at the same time, allowing the binding property of the equivocal commitment to be hard to break even in time that is sufficient to break the CCA2 commitment.

Once we have these ingredients in place, we still need to ensure that the invariant continues to hold in all the other hybrids described above. This is tricky because checking the invariant involves decrypting $\{c_{0,i}\}_{i\in[n]}$ under sk_0, and also finding the messages committed via the CCA2 commitment strings $\{c_{1,i}\}_{i\in[n]}$, which may not necessarily be an efficient process. Recall that in the very next hybrid, we simply sample the commitment strings in an equivocal way – this hybrid is statistically indistinguishable from the previous one, and therefore the invariant also holds in this hybrid. In the hybrid after that, the commitment strings $c_{1,i}$ are modified in the challenge ciphertext to drown out information about s. Here, in order to prove that the invariant holds, we rely on CCA2 security of the commitment to find the messages committed via the CCA2 commitment strings $\{c_{1,i}\}_{i\in[n]}$ in all of the adversary's queries.

In the fourth hybrid, we change the way the adversary's queries are decrypted: here, we can prove (this time, by relying on the invariant) that the adversary does not make decryption queries that decrypt differently, except with negligible probability. In the next hybrid, we modify the decryption oracle again to go back to using the secret key sk_0 of the public key encryption scheme to decrypt. At this point, we are no longer able to argue that the invariant holds, but note that we only needed the invariant to argue that the way the adversary's queries are decrypted can be changed without affecting the adversary's advantage. Therefore, this point on, we will not make any changes to how the adversary's queries are decrypted, and so all we will need to do is argue indistinguishability of the subsequent hybrids. At this point, the only remaining information about s is in the *randomness* used to obtain $\{c_{i,0}, c_{i,1}\}_{i\in[n]}$ in the challenge ciphertext. In the next couple of hybrids, we carefully re-order the randomness and rely on the security of the hinting PRG to switch to using uniform randomness everywhere. All this while, we decrypt the adversary's oracle queries by breaking the CCA commitments (via brute-force). As a result, our reductions run in superpolynomial time, and we rely on sub-exponential hardness of the hinting PRG. This is different from the CCA1 setting where we could first go back to decrypting the adversary's oracle queries in polynomial time and then rely on polynomial hardness of the hinting PRG.

We provide some additional technical details about how we implement the invariant discussed in this overview. Specifically, we insert a hybrid after the first hybrid, where the experiment aborts (and the adversary wins) if he makes an oracle query that breaks the invariant: that is, if the adversary makes an oracle query for which there exists $i \in [n]$ such that the components $(c_{0,i}, c_{1,i})$ encrypt/commit to two different openings of the string $c_{2,i}$. This (inefficient) check is performed in all subsequent hybrids up until the fourth one, where we change the way the adversary's queries are decrypted. We perform careful reductions to argue indistinguishability of these hybrids while performing this inefficient check (as described above). After the fourth hybrid, we no longer need the invariant and we therefore remove this check before proceeding with subsequent hybrids. This concludes an overview of our construction and proof of security.

2.4 On Security against Non-Uniform Adversaries

Very recently, the security of keyless hash functions against adversaries with *non-uniform advice* has also been explored; in particular, [BDRV18, BKP18, KNY18] defined and constructed keyless collision-resistant hash functions that satisfy the following property: there exists a polynomial $p(\cdot)$ such that for any polynomial $s(\cdot)$, any PPT adversary with $s(\kappa)$ bits of non-uniform advice cannot find more than $p(s(\kappa))$ pairs of collisions. Subsequently, [BL18] used these hash functions and (sub-exponential) NIWIs to obtain one-message zero-knowledge without trusted setup and a weak form of soundness against provers with non-uniform advice.

We observe that relying on non-uniform secure primitives; more specifically substituting keyless collision-resistant hash functions against uniform adversaries with keyless collision-resistant hash functions against adversaries with non-uniform advice as described above, helps make our CCA constructions secure against non-uniform adversaries. In other words, we can make a stronger assumption on the underlying keyless hash function, to obtain stronger (non-uniform) security. The only difference would be the observation that an adversary with polynomial advice can only find polynomially many collisions, and therefore query the decryption oracle with only polynomially many ambiguous ciphertexts – the answers to which can be non-uniformly fixed and hardwired into the oracle.

2.5 On Setting Parameters for CCA Compatibility

For both the CCA1 and CCA2 transformations, our non-interactive commitment scheme used to create $\{c_{1,i}\}_{i\in[n]}$ needs to be easier to break along some axis of hardness than the PKE scheme so that there is a way to open it, while the PKE scheme is still hard. Our axis of choice in this paper, is basic computation time. As a result, our theorem statement requires the KeyGen algorithm to be sub-exponentially CPA compatible, i.e. to give rise to a sub-exponentially secure CPA encryption scheme. This could also potentially lead to issues if the original PKE scheme had parameters on edge of being broken: since we would need commitment scheme to be even easier to break in terms of computation time.

We point out that in these cases, there could be other different axes of hardness (eg,time-lock puzzles [LPS17, BL18]) that could be exploited to achieve the same effect. As another example, following [KK19], one could show that any KeyGen that gives rise to *polynomially hard* PKE scheme secure against quantum adversaries can be combined with a commitment scheme that is quantum *in*-secure, to achieve CCA compatibility. As a result, there is still a way to open the commitments in BQP, while the CPA-secure PKE scheme is still hard. These approaches could improve the concrete parameters that one would need to use to instantiate these transformations, and the exact axis of hardness can be chosen depending upon the specifics of the application.

In the coming sections, we first discuss some key building blocks used by our transformations in Section 3, and define the notion of compatibility in Section 4. Next, we describe our CCA2 compatibility construction in Section 6, with analysis and proof of security deferred to the full version. We can use simpler assumptions and a simpler construction to achieve the weaker goal of CCA1 compatibility, as discussed above. This construction and analysis are deferred to the full version due to lack of space.

3 Preliminaries

In this section we will provide notions and security definitions for public key encryption, keyless collision resistant hash functions and non-interactive perfectly binding commitments. For public key encryption we will formulate a definition that can capture IND-CPA, IND-CCA1 and IND-CCA2 security. For all of our definitions we will be explicit to whether we are describing security against uniform or non-uniform adversaries as our results will be sensitive to this nuance.

We will use κ to denote the security parameter. We will denote by $\mathsf{negl}(\kappa)$ a function that is asymptotically smaller than the inverse of every polynomial in κ.

Public Key Encryption

A public key encryption scheme is specified by a triple of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, with the following syntax:

- $\mathsf{KeyGen}(1^\kappa; r_{\mathsf{KeyGen}}) \to (\mathsf{sk}, \mathsf{pk})$
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg}; r_{\mathsf{Enc}}) \to \mathsf{ct}$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \mathsf{msg}$

(Perfect) Correctness. A public key encryption scheme with message space $\mathcal{M}$ is correct if $\forall \kappa, r_{\mathsf{KeyGen}}, \mathsf{msg} \in \mathcal{M}$ we have $\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}; r_{\mathsf{Enc}})) = \mathsf{msg}$ where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa; r_{\mathsf{KeyGen}})$.

Security Parameter Retrievability. We introduce a new property and say that a public key encryption scheme is *security parameter retrievable* if there exists a polynomial time algorithm $\mathsf{RetrieveParam}$ that can extract the security parameter used to generate a public key. More formally $\forall \kappa, r_{\mathsf{KeyGen}}$ it must be that $\mathsf{RetrieveParam}(\mathsf{pk}) = \kappa$ where $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\kappa; r_{\mathsf{KeyGen}})$.

Message Recovery from Randomness. We will additionally assume a property of message recovery from randomness given in [KW19]. Suppose that ct is an encryption of message msg under a (validly generated) public key $\mathsf{cpa.pk}$ and randomness r. Then the exists an algorithm $\mathsf{CPA.Recover}$ where $\mathsf{CPA.Recover}(\mathsf{cpa.pk}, \mathsf{ct}, r) = \mathsf{msg}$.

While not all public key encryption schemes have this property, the encryption algorithm of any IND-CPA secure PKE scheme can be modified to included this property. Assume that messages are n bits long. Then one can simply use n additional random coins r' during encryption and append $\mathsf{msg} \oplus r'$ to the end of the ciphertext. The message can then be recovered from the random coins by a simple XOR operation with r'. Moreover, since the r' portion of the coins are not used elsewhere in encryption, IND-CPA security is preserved. Also noteworthy is that this simple transformation only modifies the encryption algorithm and not the public key structure. Thus, from a compatibility perspective the setup algorithm remains the same. In the presentation of our construction we will assume that the public key encryption scheme has this property.

Security. We now describe security for public key encryption schemes. Our security definition presentation will present a single game of (full) chosen ciphertext security and then derive IND-CCA-1 and IND-CPA security

We define the following security game between a challenger and a *stateful* attacker.

1. The challenger first runs $\mathsf{KeyGen}(1^\kappa; r_{\mathsf{KeyGen}}) \rightarrow (\mathsf{sk}, \mathsf{pk})$ and gives pk to the attacker.
2. The attacker then is allowed to make oracle queries to the function $\mathsf{Dec}(\mathsf{sk}, \cdot)$
3. The attacker submits two messages $\mathsf{msg}_0, \mathsf{msg}_1 \in \mathcal{M} \times \mathcal{M}$ to the challenger.
4. The challenger choose a coin $b \in \{0, 1\}$ responds with $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{msg}_b; r_{\mathsf{Enc}})$ for random r_{Enc}.
5. The attacker then is allowed to make oracle queries to the function $\mathsf{Dec}(\mathsf{sk}, \cdot)$ with the restriction that ct^* is not given as input to the oracle.
6. The attacker outputs a bit b'.

We refer to the above security game as the IND-CCA2 security game. We define IND-CCA1 security as the above game, with the exception that the attacker is not allowed any decryption oracle queries in Step 5. We define the IND-CPA security game as above with the exception that the attacker is not allowed any decryption oracle queries in Step 2 and none in Step 5.

Definition 3.1 (Secure Public Key Encryption)**.** We will say that a public key encryption scheme is (IND-CCA2, IND-CCA1, IND-CPA) secure if for all *non-uniform* poly-time attackers $\mathcal{A}$ most poly-sized bits of advice we have that there exists a negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$ in the (IND-CCA2, IND-CCA1, IND-CPA) security game.

Definition 3.2 (Uniform Secure Public Key Encryption)**.** We will say that a public key encryption scheme is (IND-CCA2, IND-CCA1, IND-CPA) secure if for all poly-time uniform attackers $\mathcal{A}$ we have that there exists a negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$ in the (IND-CCA2, IND-CCA1, IND-CPA) security game.

In our construction we will also need to consider more fined-grained notions of security where we will specific a time function T that the attacker is allowed to run in. Typically, this will be used to specify security against an attacker that runs in time subexponential in the security parameter.

Definition 3.3 (Non-uniform T-secure Public Key Encryption)**.** We will say that a public key encryption scheme is T-(IND-CCA2, IND-CCA1, IND-CPA) secure if for every polynomial $p(\cdot)$ all *non-uniform* attackers $\mathcal{A}$ running in time at most $p(T(\kappa))$ and with at most $p(T(\kappa))$ bits of advice we have that there exists a negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$ in the (IND-CCA2, IND-CCA1, IND-CPA) security game.

Definition 3.4 (Uniform T-secure Public Key Encryption)**.** We will say that a public key encryption scheme is T-(IND-CCA2, IND-CCA1, IND-CPA) secure if for every polynomial $p(\cdot)$ and all uniform attackers $\mathcal{A}$ running in time at most $p(T(\kappa))$ we have that there exists a negligible function negl such that $\Pr[b' = b] \leq \frac{1}{2} + \mathsf{negl}(\kappa)$ in the (IND-CCA2, IND-CCA1, IND-CPA) security game.

Non-Interactive Perfect Binding Commitments

Definition 3.5 (Non-Interactive Perfectly Binding Commitments with Non-Uniform Security)**.** A non-interactive perfectly binding commitment is specified by a poly-time computable randomized algorithm Com that on input $(1^\kappa, \mathsf{msg}; r)$ outputs a commitment string c of length $\ell(\kappa)$, where $\ell(\cdot)$ is a polynomially bounded function. This algorithm satisfies the following properties:

- **Perfect Binding:** For all $c \in \{0,1\}^*$, κ,

$$\nexists(\mathsf{msg}_0, \mathsf{msg}_1, r_0, r_1) \text{ such that } \mathsf{msg}_0 \neq \mathsf{msg}_1, \text{ and}$$

$$c = \mathsf{Com}(1^\kappa, \mathsf{msg}_0; r_0) \text{ and } c = \mathsf{Com}(1^\kappa, \mathsf{msg}_1; r_1).$$

- **Computational Hiding:** There exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall \mathsf{msg}_0, \mathsf{msg}_1 \in \{0,1\}^*$ s.t. $|\mathsf{msg}_0| = |\mathsf{msg}_1|, \forall$ non-uniform PPT $\mathcal{A}$,

$$\Big| \Pr[\mathcal{A}\left(\mathsf{Com}(1^\kappa, \mathsf{msg}_0, r)\right) = 1] - \Pr[\mathcal{A}\left(\mathsf{Com}(1^\kappa, \mathsf{msg}_1, r)\right) = 1] \Big| \leq \mathsf{negl}(\kappa)$$

  where $\mathcal{M}$ denotes message space and the probability is over the randomness of sampling r.

Message Recovery from Randomness. We will also assume a property of message recovery from randomness for our commitment scheme. Suppose that c is a commitment of message msg under randomness r. Then the exists an algorithm $\mathsf{Com.Recover}$ where $\mathsf{Com.Recover}(c, r) = \mathsf{msg}$. A similar argument to the one given above for public key encryption shows how one can derive a commitment scheme with the message recover from randomness property from any ordinary one.

Decryption in exponential time. We will implicitly assume that any message m committed to using security parameter 1^κ can be retrieved with probability 1 by an algorithm running in time $2^\kappa q(\kappa)$ for some polynomial function q. We denote $\mathsf{Com.Dec}$ as the algorithm for doing this.

Equivocal Commitments without Setup

Equivocal commitments were proposed by DiCrescenzo, Ishai and Ostrovsky [CIO98] as a bit commitment scheme with a trusted setup algorithm. During normal setup, the bit commitment scheme is statistically binding. However, there exists an alternative setup which produces public parameters along with a trapdoor, that produces commitments which can be opened to either 0 or 1. Moreover, the public parameters of the normal and alternative setup are computationally indistinguishable.

Here we will define a similar primitive, but without utilizing a trusted setup algorithm. In order for such a notion to be meaningful, we will require the commitment scheme to be computationally binding for any *uniform* T-time attacker, but there will exist an algorithm running in time $\mathsf{poly}(2^\kappa)$ that can be opened to 0 or 1. Moreover, such a commitment with one of the openings should be statistically indistinguishable from a commitment created in the standard manner.
An equivocal commitment scheme without setup consists of three algorithms:

$\mathsf{Equiv.Com}(1^\kappa, b) \to (c, d)$ is a randomized PPT algorithm that takes in a bit and security parameter and outputs a commitment c, decommitment string d.

$\mathsf{Equiv.Decom}(c, d) \to \{0, 1, \perp\}$ is a deterministic polytime algorithm that takes in part of the commitment and it's opening and reveals the bit that it was committed to or $\perp$ to indicate failure.

$\mathsf{Equiv.Equivocate}(1^\kappa) \to (c, d_0, d_1)$ is an (inefficient) randomized algorithm that takes in the security parameter, and outputs decommitment strings to both 0 and 1.

Definition 3.6. We say an equivocal commitment scheme is perfectly correct if for all $b \in \{0, 1\}$

$$\Pr\begin{bmatrix}(c, d) \leftarrow \mathsf{Equiv.Com}(1^\kappa, b) \\ b' \leftarrow \mathsf{Equiv.Decom}(c, d) \\ b' = b\end{bmatrix} = 1$$

Definition 3.7. We say an equivocal commitment scheme is efficient if $\mathsf{Equiv.Com}$ and $\mathsf{Equiv.Decom}$ run in $\mathsf{poly}(\kappa)$ time, and $\mathsf{Equiv.Equivocate}$ runs in time 2^κ.

We now define the binding and equivocal properties.

Definition 3.8. An equivocal commitment without setup scheme ($\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}$, $\mathsf{Equiv.Equivocate}$) is said to be $T(\cdot)$ binding secure if for any *uniform* adversary $\mathcal{A}$ running in time $p(T(\kappa))$ for some polynomial p, there exists a negligible function $\mathsf{negl}(\cdot)$,

$$\Pr\left[(c, d_0, d_1) \leftarrow \mathcal{A}(1^\kappa) : \mathsf{Equiv.Decom}(c, d_0) = 0 \wedge \mathsf{Equiv.Decom}(c, d_1) = 1\right] \leq \mathsf{negl}(\kappa).$$

Definition 3.9. We sat that a scheme is equivocal if for all $b \in \{0, 1\}$ the statistical difference between the following two distributions is negligible in κ.

- $\mathcal{D}_0 = (c, d)$ where $\mathsf{Equiv.Com}(1^\kappa, b) \to (c, d)$.
- $\mathcal{D}_1 = (c, d_b)$ where $\mathsf{Equiv.Equivocate}(1^\kappa) \to (c, d_0, d_1)$.

We observe that our security definitions do not include an explicit hiding property of a committed bit. This property is actually implied by our equivocal property, and hiding will not be explicitly needed by our proof.

Hinting PRGs

We now provide the definition of hinting PRGs taken from [KW19]. Let $n(\cdot,\cdot)$ be a polynomial. A n-hinting PRG scheme consists of two PPT algorithms $\mathsf{Setup}, \mathsf{Eval}$ with the following syntax.

$\mathsf{Setup}(1^\kappa, 1^\ell)$: The setup algorithm takes as input the security parameter κ, and length parameter ℓ, and outputs public parameters pp and input length $n = n(\kappa, \ell)$.

$\mathsf{Eval}\,(\mathsf{pp}, s \in \{0,1\}^n, i \in [n] \cup \{0\})$: The evaluation algorithm takes as input the public parameters pp, an n bit string s, an index $i \in [n] \cup \{0\}$ and outputs an ℓ bit string y.

Definition 3.10. A hinting PRG scheme $(\mathsf{Setup}, \mathsf{Eval})$ is said to be secure if for any PPT adversary $\mathcal{A}$, polynomial $\ell(\cdot)$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds:

$$\left| \Pr\left[\beta \leftarrow A\left(\mathsf{pp}, \left(y_0^\beta, \left\{y_{i,b}^\beta\right\}_{i\in[n], b\in\{0,1\}}\right)\right) : \begin{array}{c} (\mathsf{pp}, n) \leftarrow \mathsf{Setup}(1^\kappa, 1^{\ell(\lambda)}), s \leftarrow \{0,1\}^n, \\ \beta \leftarrow \{0,1\}, y_0^0 \leftarrow \{0,1\}^\ell, y_0^1 = \mathsf{Eval}(\mathsf{pp}, s, 0), \\ y_{i,b}^0 \leftarrow \{0,1\}^\ell \ \forall\ i \in [n], b \in \{0,1\}, \\ y_{i,s_i}^1 = \mathsf{Eval}(\mathsf{pp}, s, i), y_{i,\overline{s_i}}^1 \leftarrow \{0,1\}^\ell \ \forall\ i \in [n] \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

Computation Enabled CCA Commitments

We now define "computation enabled" CCA secure commitments [GKLW20]. Intuitively, these are tagged commitments where a commitment to message m under tag tag and randomness r is created as $\mathsf{CCA.Com}(\mathsf{tag}, m; r) \rightarrow \mathsf{Com}$. The scheme is statistically binding in that for all $\mathsf{tag}_0, \mathsf{tag}_1$, r_0, r_1 and $m_0 \neq m_1$ we have that $\mathsf{CCA.Com}(\mathsf{tag}_0, m_0; r_0) \neq \mathsf{CCA.Com}(\mathsf{tag}_1, m_1; r_1)$.

Our hiding property follows along the lines of chosen commitment security definitions [CLP10] where an attacker gives a challenge tag tag^* along with messages m_0, m_1 and receives a challenge commitment Com^* to either m_0 or m_1 from the experiment. The attacker's job is to guess the message that was committed to with the aid of oracle access to an (inefficient) value function $\mathsf{CCACom.Val}$ where $\mathsf{CCACom.Val}(\mathsf{Com})$ will return m if $\mathsf{CCA.Com}(\mathsf{tag}, m; r) \rightarrow \mathsf{Com}$ for some r. The attacker is allowed oracle access to $\mathsf{CCACom.Val}(\cdot)$ for any $\mathsf{tag} \neq \mathsf{tag}^*$.

The computation enabled property allows the attacker to submit a randomized turing machine P at the beginning of the game. The challenger will run the program P and output the result for the attacker at the beginning of the game. This added property will be useful in our proof of security. In addition, we require a message recovery from randomness property, which allows one to open the commitment given all the randomness used to generate said commitment.

A computation enabled CCA secure commitment is parameterized by a tag space of size $N = N(\kappa)$ and tags in $[1, N]$. It consists of 3 algorithms:

$\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m; r) \rightarrow \mathsf{Com}$ is a randomized PPT algorithm that takes as input the security parameter κ, a tag $\mathsf{tag} \in [N]$, a message $m \in \{0,1\}^*$ and outputs a commitment Com, including the tag $\mathsf{Com.tag}$. We denote the random coins explicitly as r.

$\mathsf{CCACom.Val}(\mathsf{Com}) \rightarrow m \cup \perp$ is a deterministic inefficient algorithm that takes in a commitment Com and outputs either a message $m \in \{0,1\}^*$ or a reject symbol $\perp$.

$\mathsf{CCACom.Recover}(\mathsf{Com}, r) \rightarrow m$ is a deterministic algorithm which takes a commitment Com and the randomness r used to generate Com and outputs the underlying message m.

We now define the correctness, efficiency properties, as well as the security properties of perfectly binding and message hiding.

A computation enabled CCA secure commitment scheme is perfectly correct if the following holds. $\forall m \in \{0,1\}^*$, $\mathsf{tag} \in [N]$ and r we have that

$$\mathsf{CCACom.Val}(\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m; r)) = m.$$

A computation enabled CCA secure commitment scheme is efficient if $\mathsf{CCA.Com}$, $\mathsf{CCACom.Recover}$ run in time $\mathsf{poly}(|m|, \kappa)$, while $\mathsf{CCACom.Val}$ runs in time $\mathsf{poly}(|m|, 2^\kappa)$.

A computation enabled CCA secure commitment is perfectly binding if $\forall m_0, m_1 \in \{0,1\}^*$ s.t. $m_0 \neq m_1$ there does not exist $\mathsf{tag}_0, \mathsf{tag}_1, r_0, r_1$ such that $\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_0, m_0; r_0) = \mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_1, m_1; r_1)$.

Remark 3.1. We remark that this is implied by correctness, as we know that if $\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_0, m_0; r_0) = \mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_1, m_1; r_1)$, then

$$\begin{aligned} m_0 &= \mathsf{CCACom.Val}(\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_0, m_0; r_0)) \\ &= \mathsf{CCACom.Val}(\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}_1, m_1; r_1)) = m_1, \end{aligned}$$

but $m_0 \neq m_1$, a contradiction.

We define our message hiding game between a challenger and an attacker. The game is parameterized by a security parameter κ.

1. The attacker sends a randomized and inputless Turing Machine algorithm P. The challenger runs the program on random coins and sends the output to the attacker. If the program takes more than 2^{2^κ} time to halt, the outputs halts the evaluation and outputs the empty string.[1]

2. The attacker sends a "challenge tag" $\mathsf{tag}^* \in [N]$.

3. The attacker makes repeated commitment queries Com. If $\mathsf{Com.tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise it responds as

$$\mathsf{CCACom.Val}(\mathsf{Com}).$$

4. For some w, the attacker sends two messages $m_0, m_1 \in \{0,1\}^w$.

5. The challenger flips a coin $b \in \{0,1\}$ and sends $\mathsf{Com}^* = \mathsf{CCA.Com}(\mathsf{tag}^*, m_b; r)$ for randomly chosen r.

6. The attacker again makes repeated queries of commitment Com. If $\mathsf{Com.tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise it sends

$$\mathsf{CCACom.Val}(\mathsf{Com}).$$

7. The attacker finally outputs a guess b'.

[1] The choice of 2^{2^κ} is somewhat arbitrary as the condition is in place so that the game is well defined on all P.

We define the attacker's advantage in the game to be $\Pr[b' = b] - \frac{1}{2}$ where the probability is over all the attacker and challenger's coins.

Definition 3.11. An attack algorithm $\mathcal{A}$ is said to be e-conforming for some real value $e > 0$ if:

1. $\mathcal{A}$ is a (randomized) uniform algorithm.
2. $\mathcal{A}$ runs in polynomial time.
3. The program P output by $\mathcal{A}$ in Step 1 of the game terminates in time $p(2^{\kappa^e})$ and outputs at most $q(\kappa)$ bits for some polynomial functions p, q (For all possible random tapes given to the program P).

Definition 3.12. A computation enabled CCA secure commitment scheme scheme given by algorithms $(\mathsf{CCA.Com}, \mathsf{CCACom.Val}, \mathsf{CCACom.Recover})$ is said to be e-computation enabled CCA secure if for any e-conforming adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that the attacker's advantage in the game is $\mathsf{negl}(\kappa)$.

Definition 3.13. We say that our CCA secure commitment scheme can be recovered from randomness if the following holds. For all $m \in \{0,1\}^*$, $\mathsf{tag} \in [N]$, and r, $\mathsf{CCACom.Recover}(\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m; r), r) = m$.

Claim 3.1. Let $(\mathsf{CCA.Com}, \mathsf{CCACom.Val})$ be a set of algorithms which satisfy the correctness, efficiency, binding and Definition 3.12. Then there exists a set of algorithms $(\mathsf{CCA'.Com}, \mathsf{CCA'.Val}, \mathsf{CCA'.Recover})$ which satisfy the same properties as well as Definition 3.13.

Proof. Consider the following transformation:

$$\mathsf{RecoverRandom}(\mathsf{NM} = (\mathsf{CCA.Com}, \mathsf{CCACom.Val})) \to \mathsf{NM}' =$$

$$(\mathsf{CCA'.Com}, \mathsf{CCA'.Val}, \mathsf{CCA'.Recover}):$$

$\mathsf{CCA'.Com}(\mathsf{tag}, m; r = (r_0, r_1))$: Let $\mathsf{Com} = \mathsf{CCA.Com}(\mathsf{tag}, r_0)$, and $c = r_1 \oplus m$. Output (Com, c).

$\mathsf{CCA'.Val}(\mathsf{Com}' = (\mathsf{Com}, c))$: Output $\mathsf{CCACom.Val}(\mathsf{Com})$.

$\mathsf{CCA'.Recover}(\mathsf{Com}' = (\mathsf{Com}, c), r = (r_0, r_1))$: Output $c \oplus r_1$.

We can see that correctness, efficiency and binding all hold if they do in the underlying scheme as they call the underlying $\mathsf{CCA.Com}, \mathsf{CCACom.Val}$ once. To see that Definition 3.12 still holds, we can consider an attacker $\mathcal{A}$ against NM'. We can construct an attacker for NM by taking the challenge commitment Com, appending w uniformly random bits c' to it, and running $\mathcal{A}$ on (Com, c'). Let m be the underlying message in Com. Since c' is independent and uniformly random, so is $c' \oplus m$, meaning that (Com, c') produces a distribution of Com' identical to $\mathsf{CCA'.Com}$. Finally, we can see that our transformation satisfies Definition 3.13 as $c \oplus r_1 = m \oplus r_1 \oplus r_1 = m$.

$\square$

Connecting to Standard Security

We now connect our computation enabled definition of security to the standard notion of chosen commitment security. In particular, the standard notion of chosen commitment security is simply the computation enabled above, but removing the first step of submitting a program P. We prove two straightforward lemmas. The first showing that any computation enabled CCA secure commitment scheme is a standard secure one against uniform attackers. The second is that any non-uniformly secure standard scheme satisfies e-computation enabled security for any constant $e \geq 0$.

Definition 3.14. A commitment $(\mathsf{CCA.Com}, \mathsf{CCACom.Val}, \mathsf{CCACom.Recover})$ is said to be CCA secure against uniform/non-uniform attackers if for any poly-time uniform/non-uniform adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\mathcal{A}$'s advantage in the above game *with Step 1 removed* is $\mathsf{negl}(\kappa)$.

Claim 3.2. If $(\mathsf{CCA.Com}, \mathsf{CCACom.Val}, \mathsf{CCACom.Recover})$ is an e-computation enabled CCA secure commitment scheme for some e as per Definition 3.12, then it is also a scheme that achieves standard CCA security against uniform poly-time attackers as per Definition 3.14.

Proof. This follows from the fact that any uniform attacker $\mathcal{A}$ in the standard security game with advantage $\epsilon(\kappa) = \epsilon$ immediately implies an e-conforming attacker $\mathcal{A}'$ with the same advantage where $\mathcal{A}'$ outputs a program P that immediately halts and then runs $\mathcal{A}$. $\square$

Claim 3.3. If $(\mathsf{CCA.Com}, \mathsf{CCACom.Val}, \mathsf{CCACom.Recover})$ achieves standard CCA security against *non-uniform* poly-time attackers as per Definition 3.14, then it is an e-computation enabled CCA secure commitment scheme for any e as per Definition 3.12.

Proof. Suppose $\mathcal{A}$ is an e-conforming attacker for some e with some advantage $\epsilon = \epsilon(\kappa)$. Then our non-uniform attacker $\mathcal{A}'$ can fix the random coins of $\mathcal{A}$ and to maximize its probability of success. Since now $\mathcal{A}$ is deterministic save for randomness produced by the challenger in step 5, this deterministically fixes the P $\mathcal{A}$ sends, so $\mathcal{A}'$ can fix the coins of P to maximize success. Thus, $\mathcal{A}'$ can simulate $\mathcal{A}$ given the above aforementioned random coins of $\mathcal{A}$ and the output of P, both of which are poly-bounded by the fact that $\mathcal{A}$ is e-conforming. Since all non-challenger randomness was non-uniformly fixed to maximize success, $\mathcal{A}'$ has at least advantage ϵ as well. By our definition of standard security hiding, the advantage of $\mathcal{A}'$ must be negligible, so $\mathcal{A}$'s advantage must be as well. $\square$

Decryption in exponential time. We will implicitly assume that any message m committed to using security parameter 1^κ can be retrieved with probability 1 by an algorithm running in time $2^\kappa q(\kappa)$ for some polynomial function q. We denote $\mathsf{CCACom.Dec}$ as the algorithm for doing this.

4 Defining CCA Compatibility

In this section we provide formal definitions of what it means for a scheme to be CPA/CCA compatible. This will be a property of any KeyGen algorithm, and our main technical result will establish that CPA compatibility implies CCA compatibility (under appropriate hardness assumptions).

Definition 4.1 (CPA Compatibility)**.** An algorithm KeyGen is said to be non-uniform (resp., uniform) T-CPA-compatible for $T = T(\kappa)$ and message space $\mathcal{M}(\kappa)$, if there exist poly-time algorithms $\mathsf{Enc}, \mathsf{Dec}$ such that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ comprise a public key encryption scheme for message space $\mathcal{M}(\kappa)$, that satisfies $p(T)$-IND-CPA according to Definition 3.3 (resp., Definition 3.4), for every polynomial function $p(\cdot)$.

Definition 4.2 (CCA1 Compatibility)**.** An algorithm KeyGen is said to be non-uniform (resp., uniform) T-CCA1-compatible for message space $\mathcal{M}(\kappa)$, if there exist poly-time algorithms $\mathsf{Enc}, \mathsf{Dec}$ such that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ comprise a public key encryption scheme for message space $\mathcal{M}(\kappa)$, that satisfies $p(T)$-IND-CCA1 according to Definition 3.3 (resp., Definition 3.4), for every polynomial function $p(\cdot)$.

Definition 4.3 (CCA2 Compatibility)**.** An algorithm KeyGen is said to be non-uniform (resp., uniform) T-CCA2-compatible for message space $\mathcal{M}(\kappa)$, if there exist poly-time algorithms $\mathsf{Enc}, \mathsf{Dec}$ such that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ comprise a public key encryption scheme for message space $\mathcal{M}(\kappa)$, that satisfies $p(T)$-IND-CCA2 according to Definition 3.3 (resp., Definition 3.4), for every polynomial function $p(\cdot)$.

Our main result is that any KeyGen that is non-uniform $T(\lambda)$-CPA-compatible where $T = 2^{\lambda^c}$ for any constant $c > 0$, is uniform λ-CCA1-compatible and uniform λ-CCA2-compatible, under appropriate computational hardness assumptions.

5 On CCA1 Compatibility

5.1 Our Construction

Let $e > 0$ be a constant. We now provide our construction of an IND-CCA1 secure encryption system that uses any 2^{κ^e}-CPA compatible KeyGen algorithm. Our construction also relies on a hinting PRG, non-interactive binding commitments and subexponentially secure equivocal commitments.

Let $(\mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ be the encryption and decryption algorithms of the non-uniform secure $T = 2^{\kappa^e}$-$\mathsf{IND\text{-}CPA}$ secure public key encryption scheme with randomness-recoverable ciphertexts and perfect decryption correctness, that is guaranteed to exist by Definition 4.1. In addition, we will assume an equivocal commitment $(\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ that is also $T = 2^{\kappa^e}$ secure (for the same e), a perfectly binding commitment scheme Com and a hinting PRG scheme $\mathsf{HPRG} = (\mathsf{HPRG.Setup}, \mathsf{HPRG.Eval})$.

We will set the security parameter of the commitment scheme to be κ^e, so that the scheme can be broken in brute force in time 2^{κ^e}. We will also assume that our underlying encryption scheme has message space $\{0,1\}^{\lambda+1}$ and uses $\ell(\cdot)$ bits of randomness for encryption.

We will now describe our $\mathsf{CCA1}$ secure public key encryption scheme $\mathsf{PKE_{CCA}} = (\mathsf{KeyGen}$, $\mathsf{CCA.Enc}$, $\mathsf{CCA.Dec})$ with message space $\ell(\kappa)$. For simplicity of notation, we will skip the dependence of ℓ on κ.

$\mathsf{KeyGen}(1^\kappa)$: The KeyGen algorithm outputs a public key $\mathsf{cca.pk}$.

$\mathsf{CCA.Enc}(\mathsf{cca.pk}, m \in \{0,1\}^\ell)$: The encryption algorithm does the following:

1. It runs $\mathsf{RetrieveParam}(\mathsf{cca.pk}) \to \kappa$ and then calculates $\kappa' = \kappa^e$.

2. It samples $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
3. It then chooses $s \leftarrow \{0,1\}^n$.
4. For each $i \in [n]$, it chooses random $r_i \leftarrow \{0,1\}^\ell$ and sets $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$.
5. For each $i \in [n]$, it chooses $v_i \leftarrow \{0,1\}^\kappa$.
6. For each $i \in [n]$, let $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s_i; v_i)$, and $y_i = s_i|v_i$.
7. It sets $c = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$ and for each $i \in [n]$
   - If $s_i = 0$, it sets $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; r_i)$.
   - If $s_i = 1$, it sets $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; \widetilde{r_i})$.[2]
8. Finally, it outputs $\Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$ as the ciphertext.

$\mathsf{CCA.Dec}(\mathsf{cca.sk}, \mathsf{cca.pk}, \mathsf{cca.ct})$: Let the ciphertext $\mathsf{cca.ct}$ be parsed as $\Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$ and $\mathsf{cca.sk} = \mathsf{cpa.sk}$. The decryption algorithm computes $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$ (where $\mathsf{PKE.Find}$ is defined in Figure 4), and outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ (where $\mathsf{PKE.Check}$ is defined in Figure 5).

$\mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$

Inputs: Public Key $\mathsf{cca.pk} = \mathsf{cpa.pk}$
Secret Key $\mathsf{cca.sk} = \mathsf{cpa.sk}$
Ciphertext $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$
Output: $d \in \{0,1\}^n$

- Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$.
- For each $i \in [n]$, do the following:
  1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i})$.
  2. If $m_i = 0|v_i$ and $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i)$, set $d_i = 0$. Else set $d_i = 1$.
- Output $d = d_1 d_2 \ldots d_n$.

Figure 1: Routine $\mathsf{PKE.Find}$

5.2 Correctness

For any $(\mathsf{cca.pk}, \mathsf{cca.sk}) \in \mathsf{CCA.Setup}(1^\kappa)$ and any $m \in \{0,1\}^\ell$, denote the output of $\mathsf{CCA.Enc}(\mathsf{cca.pk}, m)$ by ciphertext $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$. By definition of the encryption algorithm, there exists
$\mathsf{HPRG.pp} \in \mathsf{HPRG.Setup}(1^\kappa, 1^\ell)$, s, $\{r_i\}_{i\in[n]}$, $\{y_i\}_{i\in[n]}$ of appropriate length such that:

[2]For ease of exposition we assume that ℓ coins are both used for encryption with security parameter κ as well as a commitment with security parameter *secparalt*. In practice if one is less than then other the extraneous bits can be truncated.

```
PKE.Check(cca.pk, cca.ct, d)

    Inputs: Public Key $\mathsf{cca.pk} = \mathsf{cpa.pk}$
            Ciphertext $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$
            $d \in \{0,1\}^n$
    Output: $\mathsf{msg} \in \{0,1\}^\ell \cup \perp$

• Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$. Compute $\kappa' = \kappa^e$.
• Let $\mathsf{flag} = \mathsf{true}$. For $i = 1$ to $n$, do the following:
    1. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$.
    2. If $d_i = 0$, let $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{0,i}, \widetilde{r_i})$. Parse $m = (s'|v')$ and perform the
       following checks. If any of the checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
         – $s' = 0$, $\mathsf{CPA.Enc}(\mathsf{cpa.pk}, m; \widetilde{r_i}) = c_{0,i}$.
         – $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
    3. If $d_i = 1$, let $m \leftarrow \mathsf{Com.Recover}(1^{\kappa'}, c_{1,i}, \widetilde{r_i})$. Parse $m = (s'|v')$ and perform the following
       checks. If any of the checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
         – $s' = 1$, $\mathsf{CPA.Enc}(\mathsf{cpa.pk}, m; \widetilde{r_i}) = c_{1,i}$.
         – $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
• If $\mathsf{flag} = \mathsf{true}$, output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)$. Else output $\perp$.
```

Figure 2: Routine PKE.Check

- $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$,
- $y_i = s_i|v_i$,
- $c = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$, and
- For each $i \in [n]$,
  - If $s_i = 0$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; r_i)$.
  - If $s_i = 1$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; \widetilde{r_i})$.

The algorithm $\mathsf{CCA.Dec}(\mathsf{cca.sk}, \mathsf{cca.pk}, \mathsf{cca.ct})$ computes the value $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$. For each $i \in [n]$, denote by $y_i' = \mathsf{CPA.Dec}(c_{0,i}, \mathsf{cpa.sk})$. Because of the way PKE.Find is defined, for every $i \in [n]$, $d_i = 0 \iff \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s_i; v_i)$ where y_i' is parsed as $s_i|v_i$.

By correctness of the CPA encryption scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $y_i' = y_i$ except with probability $\mathsf{negl}(\kappa)$ over the randomness of the encryption algorithm. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ such that with probability at least $1 - \mathsf{negl}(\kappa)$ over the randomness of encryption, for every $i \in [n]$, $d_i = 0 \iff \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i)$ for $y_i = 0|v_i$. This is true if and only if $s_i = 0$. Otherwise, $d_i = s_i = 1$. Therefore, with probability at least $1 - \mathsf{negl}(\kappa)$ over the randomness of encryption, $d = s$.

The routine $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, s)$ first computes $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$, and as long as cca.ct is such that for each $i \in [n]$,

- If $s_i = 0$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; r_i)$.
- If $s_i = 1$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{Com}(1^{\kappa'}, y_i; \widetilde{r_i})$.

outputs $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0)$. By construction, $\mathsf{cca.ct}$ satisfies the above constraint, and therefore $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, s)$ outputs $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) = m$. The proof of correctness follows by observing that the decryption algorithm outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ where $d = s$ except with probability $\mathsf{negl}(\kappa)$.

5.3 Security Proof

We will prove security by a sequence of hybrid games. We will write our first hybrid game H_0 in full which corresponds to the original CCA1 security game on our construction. We will then describe each subsequent game relative to the prior one.

5.4 Hybrids

Hybrid H_0 : This corresponds to the original CCA1 security game.

- **Setup Phase**
  1. The challenger samples a public/private key pair as $(\mathsf{cpa.sk}, \mathsf{cpa.pk}) \leftarrow \mathsf{CPA.Setup}(1^\kappa)$.
  2. It sends $\mathsf{cca.pk} = \mathsf{cpa.pk}$ to $\mathcal{A}$, and sets the secret key $\mathsf{cca.sk} = \mathsf{cpa.sk}$.
  3. Note that we will define $\kappa' = \kappa^e$.

- **Pre-challenge Query Phase**
  - *Decryption Queries*
    1. Let $\left(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\right)$ denote the input ciphertext.
    2. The challenger first computes $d = \mathsf{PKE.Find}\,(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$.
    3. It outputs $\mathsf{PKE.Check}\,(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

- **Challenge Phase**
  1. The adversary sends two challenge messages (m_0^*, m_1^*).
  2. **Part 1:**
     - It samples $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
     - The challenger chooses $s^* \leftarrow \{0,1\}^n$.
     - It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and and $r_i^* \leftarrow \{0,1\}^\ell$.
     - For each $i \in [n]$, it samples $v_i^* \leftarrow \{0,1\}^\kappa$, and sets $\sigma_i^* = \mathsf{Equiv.Com}(1^\kappa, s_i^*; v_i^*), y_i^* = s_i^*|v_i^*$.
  3. **Part 2:**
     - It chooses $\beta \in \{0,1\}$ and sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, 0) \oplus m_\beta^*$.
     - It sets $(c_{0,i}^*, c_{1,i}^*)$ as follows.
       * If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; \widetilde{r_i}^*)$, $c_{1,i}^* = \mathsf{Com}(1^{\kappa'}, y_i^*; r_i^*)$
       * If $s_i^* = 1$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; r_i^*)$, $c_{1,i}^* = \mathsf{Com}(1^{\kappa'}, y_i^*; \widetilde{r_i}^*)$.

4. Finally, it sends $\left(\mathsf{HPRG.pp}^*, 1^n, c^*, \left(c^*_{0,i}, c^*_{1,i}, \sigma^*_i\right)_{i\in[n]}\right)$ to $\mathcal{A}$.

- Finally, the adversary sends its guess b. The challenger outputs 1 if $b = \beta$ and 0 otherwise.

Hybrid H_1 : This will be the same as above except for how the v^*_i, w^*_i, y^*_i values are sampled. In this hybrid, the following procedure is performed in **Part 1**.

1. The challenger samples $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
2. It chooses $s^* \leftarrow \{0,1\}^n$.
3. It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and and $r^*_i \leftarrow \{0,1\}^\ell$.
4. Next, it does the following:
   - For each $i \in [n]$, it samples random $(\sigma^*_i, v^*_{0,i}, v^*_{1,i}) \leftarrow \mathsf{Equiv.Equivocate}(1^\kappa)$.
   - It sets $y^*_{0,i} = 0|v^*_{0,i}$, and $y^*_{1,i} = 1|v^*_{1,i}$.

   We will denote by a^* the tuple of values $(\{y^*_{0,i}, y^*_{1,i}, \sigma^*_i\}_{i\in[\kappa]})$.
5. For each $i \in [n]$, set $y^*_i = y^*_{i,s_i}$.

We remark that the challenger in this game now takes exponential time to sample the $y^*_{0,i}, y^*_{1,i}$ values.

Hybrid H_2 : The same as H_1 with the exception that in the challenge phase, the challenger sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_i; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_i; r^*_i)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; \widetilde{r_i}^*)$.

The effect of this change is that $c^*_{1,i}$ will be a commitment of $y^*_{1,i}$ no matter what s^*_i is. However, $c^*_{0,i}$ will remain dependent on s^*_i. It is an encryption of $y^*_{0,i}$ if $s^*_i = 0$ and an encryption of $y^*_{1,i}$ if $s^*_i = 1$.

Hybrid H_3 : The same as H_2 with the exception that the decryption algorithm uses an alternative "finding" algorithm $\mathsf{PKE.FindAlternative}$ in Figure 6 in place of $\mathsf{PKE.Find}$. The main difference is that the algorithm $\mathsf{PKE.FindAlternative}$ runs in time polynomial in $2^{\kappa'}$ to open the commitment scheme using the algorithm $\mathsf{Com.Dec}$ in lieu of decrypting using a secret key. The new steps are:

1. Let $\left(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\right)$ denote the input ciphertext
2. The challenger first computes $d = \mathsf{PKE.FindAlternative}\,(\mathsf{cca.pk}, \mathsf{cca.ct})$.
3. It outputs $\mathsf{PKE.Check}\,(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

Hybrid H_4 : The hybrid is the same as the previous one except that all the values in **Part 1**, that is, $\mathsf{HPRG.pp}^*, s^*, \widetilde{r_i}^*$, r^*_i, $v^*_i, w^*_i, y^*_i, \sigma^*_i$ (for all $i \in [n]$), are sampled/computed as before, but immediately before the Setup phase instead of in the Challenge phase.

PKE.FindAlternative(cca.pk, cca.ct)

Inputs: Ciphertext $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$

Output: $d \in \{0,1\}^n$

- Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$ and $\kappa' = \kappa^e$.
- For each $i \in [n]$, do the following:
  1. Let $m_i = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i})$.
  2. If $m_i = 1|v_i$, $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, 1; v_i)$, set $d_i = 1$. Else set $d_i = 0$.
- Output $d = d_1 d_2 \dots d_n$.

Figure 3: Routine PKE.FindAlternative

Hybrid H_5 : The same as H_4 with the exception that in the challenge phase: It sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_i)$, $c^*_{0,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; \widetilde{r_i}^*)$.

The effect of this change is that $c^*_{0,i}$ will become an encryption of $y^*_{0,i}$ no matter what s^*_i is.

Hybrid H_6 : The same as H_5 with the exception that the decryption algorithm goes back to using the original "finding" algorithm PKE.Find in Figure 4. Now the secret key is used for decryption again for a polynomial time process. The new steps are back to:

1. Let $\Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$ denote the input ciphertext
2. The challenger first computes $d = \mathsf{PKE.Find}\,(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct})$.
3. It outputs $\mathsf{PKE.Check}\,(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

Hybrid H_7 : The same as H_6 with the following syntax changes. For each $i \in [n]$ set $r^*_{i,s_i} = \widetilde{r_i}^*$ and set $r^*_{i,1-s_i} = r^*_i$. It sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_{0,i})$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_{1,i})$
- If $s^*_i = 1$, it sets $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_{0,i})$, $c^*_{0,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_{1,i})$.

Hybrid H_8 : The same as H_7 except for each $i \in [n]$ $r^*_{0,i}$ and $r^*_{1,i}$ are both chosen uniformly at random. In addition c^* is chosen uniformly at random.

5.5 Analysis

We will let $\mathsf{adv}^x_{\mathcal{A}}$ denote the probability that the challenger outputs 1 in Hybrid H_x. We note here that $\mathsf{adv}^x_{\mathcal{A}}$ takes values in $[0,1]$.

Lemma 5.1. For any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. The difference between H_0 and H_1 is that for each $i \in [n(\kappa)]$, H_0 samples v_i^* uniformly at random, and sets $\sigma_i^* = \mathsf{Equiv.Com}(1^\kappa, s_i^*; v_i^*)$. whereas H_1 samples $(\sigma_i^*, v_{0,i}^*, v_{1,i}^*) = \mathsf{Equiv.Equivocate}(1^\kappa)$, and sets $v_i^* = v_{i,s_i}^*$. The two distributions are statistically indistinguishable by equivocality of the commitment scheme.

$\square$

Lemma 5.2. Assuming Com is a secure non-interactive commitment scheme against *non-uniform* attackers per Definition 3.5, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{1}$$

For every κ, we consider a sequence of $n = n(\kappa)$ hybrids. We denote these by $H_{1,i}$ for $i \in [0, n]$, where $H_{1,0} \equiv H_1$ and $H_{1,n} \equiv H_2$. Moreover, for $j \in [1, n]$, $H_{1,j}$ is the same as H_1 except that for $i \in [1, j]$, the challenger sets sets $(c_{0,i}^*, c_{1,i}^*)$ as follows.

- If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; \widetilde{r_i}^*)$, $c_{1,i}^* = \mathsf{Com}(1^{\kappa'}, y_{1,i}^*; r_i^*)$
- If $s_i^* = 1$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; r_i^*)$, $c_{1,i}^* = \mathsf{Com}(1^{\kappa'}, y_{1,i}^*; \widetilde{r_i}^*)$.

By equation (1), for infinitely many $\kappa \in \mathbb{N}$, there exists $i = i(\kappa) \in [1, n(\kappa)]$ such that:

$$|\mathsf{adv}^{1,i-1}_{\mathcal{A}} - \mathsf{adv}^{1,i}_{\mathcal{A}}| \geq \frac{1}{n(\kappa) \cdot p(\kappa)} \tag{2}$$

We will construct an adversary $\mathcal{B}$ that obtains $i(\kappa)$ as non-uniform advice and with oracle access to $\mathcal{A}$, breaks computational hiding of the commitment Com against *non-uniform attackers* according to Definition 3.5.

First, note that $H_{1,i-1}$ and $H_{1,i}$ are identical until just before Part 2 of the challenge phase. For any security parameter κ, let $z = z(\kappa)$ denote the "best-possible" fixing of $a^* = (\{y_{0,i}^*, y_{1,i}^*, \sigma_i^*\}_{i \in [\kappa]})$. Note that these are the results of the exponential time sampling procedure in Step 4 of H_1 and H_2, such that conditioned on these fixed values, $|\mathsf{adv}^{1,i-1}_{\mathcal{A}} - \mathsf{adv}^{1,i}_{\mathcal{A}}|$ takes the highest value (over all possible choices of z). Thus z is such that,

$$\left|(\mathsf{adv}^{1,i-1}_{\mathcal{A}} | z) - (\mathsf{adv}^{1,i}_{\mathcal{A}} | z)\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}.$$

$\mathcal{B}$ obtains z as non-uniform advice. It executes the Setup and pre-challenge query phase exactly according to H_1. It also samples $\mathsf{HPRG.pp}^*$, s^* and computes $\widetilde{r_i}^*$ as well as $\{\widetilde{r_j^*}, r_j^*\}_{j \in [n] \setminus \{i\}}$ according to H_1. It then outputs messages (x_0, x_1) for the commitment challenger as follows:

$$x_0 = y_{0,i}^*, x_1 = y_{1,i}^*$$

It obtains from the challenger a commitment string γ (that is a commitment to $x_{b'}$ for some $b' \in \{0,1\}$), and proceeds to create the challenge ciphertext by executing Part 2 of the Challenge Phase exactly according to the strategy in $H_{1,i-1}$, sampling uniform $\beta \in \{0,1\}$. The only exception is that if $s_i^* = 0$, it sets $c_{1,i}^* = \gamma$.

$\mathcal{B}$ runs $\mathcal{A}$ on the challenge ciphertext, and outputs 1 iff β (chosen by $\mathcal{B}$) equals b (which is the output of $\mathcal{A}$). By assumption,

$$\Big|(\mathsf{adv}_{\mathcal{A}}^{1,i-1}|z) - (\mathsf{adv}_{\mathcal{A}}^{1,i}|z)\Big| \geq \frac{1}{n(\kappa)\cdot p(\kappa)}.$$

We observe that if γ is a commitment to x_0, then the experiment corresponds to $H_{1,i-1}$ and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{1,i-1}|z)$. On the other hand, if γ is a commitment to x_1, then the experiment corresponds to $H_{1,i}$ and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{1,i}|z)$.

This implies that

$$\Big|\Pr[\mathcal{B}(z) = 1|b' = 0] - \Pr[\mathcal{B}(z) = 1|b' = 1]\Big| \geq \frac{1}{n(\kappa)\cdot p(\kappa)}$$

which is a contradiction to the non-uniform hiding of Com. This completes the proof. $\square$

Lemma 5.3. Assuming $(\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ is 2^{κ^e} binding secure against *uniform* adversaries, for any uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}^2 - \mathsf{adv}_{\mathcal{A}}^3| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}_{\mathcal{A}}^2 - \mathsf{adv}_{\mathcal{A}}^3| \geq \frac{1}{p(\kappa)}.$$

The only difference between H_2 and H_3 is that in H_3, the adversary's pre-challenge queries ct are decrypted using the $\mathsf{PKE.FindAlternative}$ algorithm to obtain d and then $\mathsf{PKE.Check}$ is run on input d. On the other hand in H_2, the adversary's pre-challenge queries ct are decrypted using $\mathsf{PKE.Find}$ to obtain d followed by running $\mathsf{PKE.Check}$ on d.

Consider any pre-challenge ciphertext $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i, g)_{i\in[n]}\Big)$. We will define a set $\mathsf{BAD}_{\mathsf{CT}}$ of "bad" pre-challenge ciphertexts as follows:

$$\mathsf{cca.ct} \in \mathsf{BAD}_{\mathsf{CT}} \iff \exists i \in [n] \text{ s.t. for } m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i}), m_i' = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i}),$$
$$\text{Equations (3) and (4) are both true.}$$

$$m_i = (0|v_i), \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i) \tag{3}$$
$$m_i = (1|v_i'), \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 1; v_i') \tag{4}$$

Claim 5.1. Let $\mathbb{S}$ denote the set of all pre-challenge queries of $\mathcal{A}$. Then,

$$\Pr[(\mathbb{S} \cap \mathsf{BAD}_{\mathsf{CT}}) \neq \phi] \geq \frac{1}{p(\kappa)}.$$

Proof. Because the two hybrids only differ in the way pre-challenge queries are decrypted, with probability $\frac{1}{p(\kappa)}$, $\mathcal{A}$ makes a pre-challenge query $\mathsf{cca.ct}$ in $\mathbb{S}$ such that:

$$\begin{aligned}&\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct}))\\ \neq\ &\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, \mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \mathsf{cca.ct})).\end{aligned} \tag{5}$$

Recall that by description of $\mathsf{PKE.Check}$, for every pre-challenge query $\mathsf{cca.ct}$ of $\mathcal{A}$ and every possible $d \in \{0,1\}^n$, the output of $\mathsf{PKE.Check}$ on input $(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ is $\big(c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)\big)$, if and only if:

1. If $d_i = 0$, the following holds for $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{0,i}, \widetilde{r_i})$.
   - $\mathsf{CPA.Enc}(\mathsf{cpa.pk}, m; \widetilde{r_i}) = c_{0,i}$.
   - Parse $m = s'|v'$ for $s' \in \{0,1\}$ and $v' \in \{0,1\}^\kappa$. Then $s' = 0, \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
2. If $d_i = 1$, the following holds for $m \leftarrow \mathsf{Com.Recover}(1^{\kappa'}, c_{1,i}, \widetilde{r_i})$.
   - $\mathsf{Com}(1^{\kappa'}, m; \widetilde{r_i}) = c_{1,i}$.
   - Parse $m = s'|v'$ for $s' \in \{0,1\}$ and $v' \in \{0,1\}^\kappa$. Then $s' = 1, \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.

Otherwise, the output of $\mathsf{PKE.Check}$ on input $\mathsf{cca.pk}, \mathsf{cca.ct}$ and d is $\perp$.

Next, fix any $\mathsf{cca.pk}, \mathsf{cca.ct}$ that satisfies equation (14). For all $i \in [n]$,

- Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i}), m'_i = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i})$.
- Set $d_i = 0 \iff m_i$ satisfies equation (3),
- Set $d'_i = 1 \iff m'_i$ satisfies equation (4).

Note that equation (14) implies that:

$$\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d) \neq \mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d') \tag{6}$$

which implies that $d \neq d'$, that is, $\exists j \in [n]$ such that $d_j \neq d'_j$. Then by definition of d and d', one of the following is true:

- $d_j = 0, d'_j = 1 \implies m_j$ satisfies equation (3) and m'_j satisfies equation (4).
- $d_j = 1, d'_j = 0 \implies m_j$ does not satisfy equation (3) and m'_j does not satisfy equation (4).
  $\implies \mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d) = \perp$ and $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d') = \perp$.
  (which contradicts Equation (15)).

We note that the implications above follow by definition of d and d', by perfect correctness of the CPA encryption scheme which implies that $\mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{0,i}, \widetilde{r_i}) = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i})$, and by perfect binding of the commitment scheme which implies $\mathsf{Com.Recover}(1^{\kappa'}, c_{1,i}, \widetilde{r_i}) = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i})$.

Therefore,

$$\begin{aligned}&\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \mathsf{cca.ct}))\\ \neq\ &\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, \mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \mathsf{cca.ct}))\end{aligned}$$

if and only if $\exists j \in [n]$ such that for $m_j = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i})$ and $m'_j = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i})$, equations (3) and (4) are both true. This completes the proof of the claim. □

Claim 5.2. Let $\mathbb{S}$ denote the set of all pre-challenge queries of $\mathcal{A}$. There exists a negligible function $\delta(\cdot)$ such that:

$$\Pr[(\mathbb{S} \cap \mathsf{BAD}_{\mathsf{CT}}) \neq \phi] = \delta(\kappa).$$

Proof. Suppose the claim is not true. Then with probability at least $\frac{1}{p(\kappa)}$, $\mathcal{A}$ outputs a pre-challenge ciphertext

$$\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i, g)_{i\in[n]}\Big)$$

such that for $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i}) = (t_i|v_i), m'_i = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i}) = (1 - t_i|v'_i)$, the commitment string $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, t_i; v_i) = \mathsf{Equiv.Com}(1^\kappa, 1 - t_i; v'_i)$.

We will build a uniform reduction $\mathcal{B}$ that with oracle access to $\mathcal{A}$, contradicts $T = 2^{\kappa^e}$ binding security of the equivocal commitment. $\mathcal{B}$ runs the Setup phase, and outputs parameters to $\mathcal{A}$. Next, for each pre-challenge query $\mathsf{cca.ct} = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$ of $\mathcal{A}$, $\mathcal{B}$ computes $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i})$ and $m'_i = \mathsf{Com.Dec}(1^{\kappa'}, c_{1,i})$ where $m_i \neq m'_i$. Whenever $m_i \neq m'_i$, $\mathcal{B}$ outputs m_i, m'_i. Note that $\mathcal{B}$ runs in time $\mathsf{poly}(2^{\kappa^\epsilon})$. We have already established that this event occurs with probability at least $\frac{1}{p(\kappa)}$, therefore contradicting $T = 2^{\kappa^e}$ binding security of the equivocal commitment. □

The proof of the lemma follows immediately from Claim 6.1 and 6.2. □

Lemma 5.4. For any adversary $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^3_{\mathcal{A}} - \mathsf{adv}^4_{\mathcal{A}}| = 0$.

Proof. The only difference between these games is that in H_4 the values $\mathsf{HPRG.pp}^*, s^*, \widetilde{r_i}^*, r^*_i, v^*_i, w^*_i, y^*_i, \sigma^*_i$ (for all $i \in [n]$) are chosen earlier before key setup instead of during the challenge phase. However, since these values are independent of the message being encrypted, the games are actually identical from an external perspective. □

Lemma 5.5. Assuming $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is PKE scheme which is $T = 2^{\kappa^e}$-IND-CPA secure against *non-uniform* adversaries (where e is the constant used in the construction) per Definition 3.3, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{7}$$

For every κ, we consider a sequence of $n = n(\kappa)$ hybrids. We denote by $H_{4,i}$ for $i \in [0, n]$, where $H_{4,0} \equiv H_4$ and $H_{4,n} \equiv H_5$. Moreover, for $j \in [1, n]$, $H_{4,j}$ is the same as H_4 except that for $i \in [1, j]$, the challenger sets sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_i)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; \widetilde{r_i}^*)$.

By equation (7), for infinitely many $\kappa \in \mathbb{N}$, there exists $i = i(\kappa) \in [1, n(\kappa)]$ such that:

$$|\mathsf{adv}_{\mathcal{A}}^{4,i-1} - \mathsf{adv}_{\mathcal{A}}^{4,i}| \geq \frac{1}{n(\kappa) \cdot p(\kappa)} \tag{8}$$

We will construct an adversary $\mathcal{B}$ that obtains $i(\kappa)$ as non-uniform advice and with oracle access to $\mathcal{A}$, breaks $T = 2^{\kappa^e}$-IND-CPA security of the PKE scheme against non-uniform attackers according to Definition 3.3.

First, note that $H_{4,i-1}$ and $H_{4,i}$ are identical until just before Part 2 of the challenge phase. Let z denote the "best-possible" fixing of $a^* = (\{y^*_{0,i}, y^*_{1,i}, \sigma^*_i\}_{i \in [n]})$ such that conditioned on these fixed values, $|\mathsf{adv}_{\mathcal{A}}^{4,i-1} - \mathsf{adv}_{\mathcal{A}}^{4,i}|$ takes the highest value (over all possible choices of z). Thus z is such that,

$$\left|(\mathsf{adv}_{\mathcal{A}}^{4,i-1}|z) - (\mathsf{adv}_{\mathcal{A}}^{4,i}|z)\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}.$$

$\mathcal{B}$ obtains z as non-uniform advice, and computes $\mathsf{HPRG.pp}^*$ for the Setup phase according to H_4. It then outputs messages (x_0, x_1) for the CPA encryption challenger as follows:

$$x_0 = y^*_{1,i}, x_1 = y^*_{0,i}$$

It obtains from the challenger a public key $\mathsf{cpa.pk}$, and a ciphertext γ (that is an encryption of $x_{b'}$ for some $b' \in \{0,1\}$). It outputs $\mathsf{cca.pk} = \mathsf{cpa.pk}$ to $\mathcal{A}$. In the pre-challenge query phase, it decrypts $\mathcal{A}$'s queries by using subroutines $\mathsf{PKE.FindAlternative}$ (in time polynomial in $2^{\kappa'}$) and $\mathsf{PKE.Check}$. Next, in the challenge phase, on input (m^*_0, m^*_1) from $\mathcal{A}$, it proceeds to create the challenge ciphertext exactly according to the strategy in $H_{4,i-1}$ by sampling uniform $\beta \in \{0,1\}$, except that if $s^*_i = 1$, it sets $c^*_{1,i} = \gamma$.

$\mathcal{B}$ then runs $\mathcal{A}$ on the challenge ciphertext, and outputs 1 iff β (chosen by $\mathcal{B}$) equals b (which is the output of $\mathcal{A}$). By assumption,

$$\left|(\mathsf{adv}_{\mathcal{A}}^{4,i-1}|z) - (\mathsf{adv}_{\mathcal{A}}^{4,i}|z)\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}.$$

We observe that if γ is an encryption of x_0, then the experiment corresponds to $H_{4,i-1}$ and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{4,i-1}|z)$. On the other hand, if γ is an encryption of x_1, then the experiment corresponds to $H_{4,i}$ and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{4,i}|z)$.

This implies that

$$\left|\Pr[\mathcal{B}(z) = 1|b' = 0] - \Pr[\mathcal{B}(z) = 1|b' = 1]\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}$$

which is a contradiction to the $T = 2^{\kappa^e}$-IND-CPA security of the PKE against *non-uniform* adversaries. This completes the proof. □

Lemma 5.6. Assuming $(\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ is 2^{κ^e} binding secure against *uniform* adversaries, for any uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}^5 - \mathsf{adv}_{\mathcal{A}}^6| \leq \mathsf{negl}(\kappa)$.

Proof. This lemma follows by first reordering Part 2 to happen *after* the pre-challenge query phase, and then applying an identical argument to Lemma 6.5. □

Lemma 5.7. For any adversary $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^6_{\mathcal{A}} - \mathsf{adv}^7_{\mathcal{A}}| = 0$.

Proof. These hybrids are identical to each other. The only change is that we rename $\widetilde{r_i}^*$ to r^*_{i,s_i} and r^*_i to $r^*_{1-s_i,i}$. □

Lemma 5.8. Assuming $(\mathsf{Setup}, \mathsf{Eval})$ is a secure hinting PRG scheme against *non-uniform* adversaries per Definition 3.10, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{9}$$

We will construct an adversary $\mathcal{B}$ that with oracle access to $\mathcal{A}$, breaks security of the hindting PRG against *non-uniform attackers* according to Definition 3.10.

The only difference between H_7 and H_8 is that in H_7, for all $i \in [n]$, r^*_{i,s_i} is chosen as the output of $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$, $r^*_{1-s_i,i}$ is chosen uniformly at random and c^* is chosen as $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, 0) \oplus m^*_\beta$ whereas in H_8, r^*_{i,s_i}, $r^*_{1-s_i,i}$ and c^* are chosen uniformly at random. Also note that H_7 and H_8 are identical until the pre-challenge query phase.

Let z denote the "best-possible" fixing of $a^* = (\{y^*_{0,i}, y^*_{1,i}, \sigma^*_i\}_{i\in[\kappa]})$ such that conditioned on these fixed values, $|\mathsf{adv}^7_{\mathcal{A}} - \mathsf{adv}^8_{\mathcal{A}}|$ takes the highest value (over all possible choices of z). Thus z is such that,

$$\left|(\mathsf{adv}^7_{\mathcal{A}}|z) - (\mathsf{adv}^8_{\mathcal{A}}|z)\right| \geq \frac{1}{p(\kappa)}.$$

$\mathcal{B}$ obtains z as non-uniform advice. It then begins its game with the hinting PRG challenger, and externally obtains $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i\in[n],b\in\{0,1\}}\right)$. It then sets $r^*_{i,b} = \gamma_{i,b}$ for $i \in [n], b \in \{0,1\}$ and sets $c^* = \gamma_0 \oplus m^*_\beta$. It samples the CPA public key, answers queries in the pre-challenge phase and creates the rest of the challenge ciphertext exactly according to the strategy in H_8.

Finally, $\mathcal{B}$ runs $\mathcal{A}$ on the challenge ciphertext, and outputs 1 iff β (chosen by $\mathcal{B}$) equals b (which is the output of $\mathcal{A}$). By assumption,

$$\left|(\mathsf{adv}^7_{\mathcal{A}}|z) - (\mathsf{adv}^8_{\mathcal{A}}|z)\right| \geq \frac{1}{p(\kappa)}.$$

We observe that if $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i\in[n],b\in\{0,1\}}\right)$ is the output of a hinting PRG on a uniform seed, then the experiment corresponds to H_7 and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^7_{\mathcal{A}}|z)$. On the other hand, if $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i\in[n],b\in\{0,1\}}\right)$ is uniformly random, then the experiment corresponds to H_8 and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^8_{\mathcal{A}}|z)$.

This implies that

$$\left|\Pr[\mathcal{B}(z) = 1|b' = 0] - \Pr[\mathcal{B}(z) = 1|b' = 1]\right| \geq \frac{1}{p(\kappa)}$$

which is a contradiction to the non-uniform security of the hinting PRG. This completes the proof. □

Lemma 5.9. For any adversary $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^8_{\mathcal{A}}| \leq \frac{1}{2}$.

Proof. The challenge ciphertext component c^* now contains no information about β and β is not reflected anywhere else in the experiment. Thus the probability that $b = \beta$ is $\frac{1}{2}$ where the game does not abort. When it does abort the challenger outputs 0. Therefore the advantage of any attacker, which is the probability that the challenger outputs 1, is $\leq \frac{1}{2}$. □

Theorem 5.1. Assume the following: (1) Com is a secure non-interactive commitment scheme per Definition 3.5, (2) (Equiv.Com, Equiv.Decom, Equiv.Equivocate) is 2^{κ^e} binding secure against *uniform* adversaries and equivocal (3) (KeyGen, Enc, Dec) is PKE scheme which is $T = 2^{\kappa^e}$-IND-CPA secure against non-uniform adversaries per Definition 3.3 and (4) (Setup, Eval) is a secure hinting PRG scheme against non-uniform adversaries per Definition 3.10.

Then for any uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}|^0 \leq \mathsf{negl}(\kappa)$ and the construction of Section 5.1 is CCA1 secure against uniform attackers.

Proof. The proof of this Theorem follows immediately from Lemmas 5.1 to 5.9. □

6 On CCA2 Compatibility

Our Construction

Let κ denote the security parameter, $0 < \delta < 1$ be a constant and $\kappa' = \kappa^{\delta}$. We now provide our construction of an IND-CCA2 secure encryption system that uses any $2^{\kappa'}$-CPA compatible KeyGen algorithm, according to Definition 4.1. Our construction relies on a hinting PRG, non-interactive computation enabled CCA commitments and subexponentially secure equivocal commitments.

Let (CPA.Enc, CPA.Dec) be the encryption and decryption algorithms of the non-uniform $2^{\kappa'}$-IND-CPA secure public key encryption scheme with randomness-recoverable ciphertexts and perfect decryption correctness, that is guaranteed to exist by Definition 4.1. We will also assume that the following exist:

- An equivocal commitment (Equiv.Com, Equiv.Decom, Equiv.Equivocate) that is $T = 2^{\kappa'}$ binding secure.
- A $2^{\kappa'}$-secure hinting PRG scheme HPRG = (HPRG.Setup, HPRG.Eval) against non-uniform adversaries.
- A non-interactive e-computation enabled CCA commitment scheme represented by algorithms (CCA.Com,CCACom.Val, CCACom.Recover), with security parameter κ' and with $e = 1/\delta$ (for the same δ), such that the commitment scheme can be broken in brute force in time $2^{\kappa'}$.
- An existentially unforgeable under chosen message attack (EUF-CMA) signature scheme with (Signature.Setup, Sign, Verify) with security parameter κ'.

We will now describe our CCA secure public key encryption scheme $\mathsf{PKE}_{\mathsf{CCA}}$ = (KeyGen, CCA.Enc, CCA.Dec) with message space $\{0,1\}^{\ell(\kappa)}$. For simplicity of notation, we will skip the dependence of ℓ on κ. We will also assume that the CPA scheme has message space $\{0,1\}^{\kappa+1}$ and uses $\ell(\kappa)$ bits of randomness for encryption.

$\mathsf{KeyGen}(1^\kappa)$: The KeyGen algorithm outputs a public key $\mathsf{cca.pk}$.

$\mathsf{CCA.Enc}(\mathsf{cca.pk}, m \in \{0,1\}^\ell)$: The encryption algorithm is as follows:

1. It runs $\mathsf{RetrieveParam}(\mathsf{cca.pk}) \to \kappa$ and then calculates $\kappa' = \kappa^\delta$.
2. It samples $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
3. It then chooses $s \leftarrow \{0,1\}^n$.
4. For each $i \in [n]$, it chooses random $r_i \leftarrow \{0,1\}^\ell$ and sets $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$.
5. For each $i \in [n]$, it chooses $v_i \leftarrow \{0,1\}^\kappa$. It sets $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s_i; v_i)$, and $y_i = s_i | v_i$.
6. It sets $c = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$ and for each $i \in [n]$
   - If $s_i = 0$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; r_i)$.
   - If $s_i = 1$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; \widetilde{r_i})$.[3]
7. It sets $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i \in [n]}\Big)$.
8. It samples $(vk, sk) \leftarrow \mathsf{Signature.Setup}(1^{\kappa'})$.
9. Finally, it computes $\tau = \mathsf{Sign}(sk, \alpha)$, and outputs (vk, α, τ) as the ciphertext.

```
                         PKE.Find(cca.pk, cca.sk, α)

    Inputs:  Public Key $\mathsf{cca.pk} = \mathsf{cpa.pk}$
             Secret Key $\mathsf{cca.sk} = \mathsf{cpa.sk}$
             Ciphertext $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i \in [n]}\Big)$
    Output:  $d \in \{0,1\}^n$

• Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$.
• For each $i \in [n]$, do the following:
    1. Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{0,i})$.
    2. If $m_i = 0|v_i$ and $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i)$, set $d_i = 0$. Else set $d_i = 1$.
• Output $d = d_1 d_2 \ldots d_n$.
```

Figure 4: Routine $\mathsf{PKE.Find}$

$\mathsf{CCA.Dec}(\mathsf{cca.sk}, \mathsf{cca.pk}, \mathsf{cca.ct})$: Parse ciphertext $\mathsf{cca.ct}$ as (vk, α, τ) where $\mathsf{cca.sk} = \mathsf{cpa.sk}$ and $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i \in [n]}\Big)$. Output $\perp$ if $\mathsf{Verify}(vk, \alpha, \tau) = 0$. Otherwise, set $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha)$ (where $\mathsf{PKE.Find}$ is defined in Figure 4), and output $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ (where $\mathsf{PKE.Check}$ is defined in Figure 5).

We now prove correctness and security of our CCA2 scheme.

[3]For ease of exposition we assume that ℓ coins are both used for encryption with security parameter κ as well as a commitment with security parameter κ'. In practice if one is less than then other the extraneous bits can be truncated.

$\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$

Inputs: $\mathsf{cca.pk} = \mathsf{cpa.pk}, \mathsf{cca.ct} = (vk, \alpha, \tau)$ where

$\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i \in [n]}\Big), d \in \{0,1\}^n$

Output: $\mathsf{msg} \in \{0,1\}^\ell \cup \bot$

- Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$. Compute $\kappa' = \kappa^e$.
- Let $\mathsf{flag} = \mathsf{true}$. For $i = 1$ to n, do the following:
  1. Let $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, i)$.
  2. If $d_i = 0$, let $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{0,i}, \widetilde{r_i})$. Parse $m = (s'|v')$ and perform the following checks. If any of the checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
     - $s' = 0$, $\mathsf{CPA.Enc}(\mathsf{cpa.pk}, m; \widetilde{r_i}) = c_{0,i}$.
     - $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
  3. If $d_i = 1$, let $m \leftarrow \mathsf{CCACom.Recover}(1^{\kappa'}, c_{1,i}, \widetilde{r_i})$. Parse $m = (s'|v')$ and perform the following checks. If any of the checks fail, set $\mathsf{flag} = \mathsf{false}$ and exit loop.
     - $s' = 1$, $\mathsf{CCA.Com}(1^{\kappa'}, vk, m; \widetilde{r_i}) = c_{1,i}$.
     - $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
- If $\mathsf{flag} = \mathsf{true}$, output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)$. Else $\bot$.

Figure 5: Routine $\mathsf{PKE.Check}$

6.1 Correctness

For any $(\mathsf{cca.pk}, \mathsf{cca.sk}) \in \mathsf{CCA.Setup}(1^\kappa)$ and any $m \in \{0,1\}^\ell$, denote the output of $\mathsf{CCA.Enc}(\mathsf{cca.pk}, m)$ by ciphertext $\mathsf{cca.ct} = (vk, \alpha, \tau)$ where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i \in [n]}\Big)$. By definition of the encryption algorithm and correctness of the signature scheme, τ is indeed a valid signature w.r.t. vk on α.

Furthermore, there exists $\mathsf{HPRG.pp} \in \mathsf{HPRG.Setup}(1^\kappa, 1^\ell), s, \{r_i\}_{i \in [n]}, \{y_i\}_{i \in [n]}$ of appropriate length such that:

- $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$,
- $y_i = s_i | v_i$,
- $c = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \oplus m$, and
- For each $i \in [n]$,
  - If $s_i = 0$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; r_i)$.
  - If $s_i = 1$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; \widetilde{r_i})$.

The algorithm $\mathsf{CCA.Dec}(\mathsf{cca.sk}, \mathsf{cca.pk}, \mathsf{cca.ct})$ computes the value $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha)$. For each $i \in [n]$, let $(s'_i | v'_i) = \mathsf{CPA.Dec}(c_{0,i}, \mathsf{cpa.sk})$. Because of the way $\mathsf{PKE.Find}$ is defined, for every $i \in [n]$, $d_i = 0 \iff \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s_i; v_i)$.

By correctness of the CPA encryption scheme, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $(s'_i|v'_i) = y_i$ except with probability $\mathsf{negl}(\kappa)$ over the randomness of the encryption algorithm. Therefore, there exists a negligible function $\mathsf{negl}(\cdot)$ such that with probability at least $1 - \mathsf{negl}(\kappa)$ over the randomness of encryption, for every $i \in [n]$, $d_i = 0 \iff \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i)$ for $y_i = 0|v_i$. This is true if and only if $s_i = 0$. Otherwise, $d_i = s_i = 1$. Therefore, with probability at least $1 - \mathsf{negl}(\kappa)$ over the randomness of encryption, $d = s$.

The routine $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, s)$ first computes the value $\widetilde{r_i} = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$, and as long as α is such that for each $i \in [n]$,

- If $s_i = 0$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; \widetilde{r_i})$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; r_i)$.
- If $s_i = 1$, $c_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i; r_i)$, $c_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk, y_i; \widetilde{r_i})$.

outputs $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0)$. By construction, $\mathsf{cca.ct}$ satisfies the above constraint, and therefore $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, s)$ outputs $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) = m$. The proof of correctness follows by observing that the decryption algorithm outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$ where $d = s$ except with probability $\mathsf{negl}(\kappa)$.

6.2 Security Proof

We will prove security by a sequence of hybrid games. We will write our first hybrid game H_0 in full which corresponds to the original CCA2 security game on our construction. We will then describe each subsequent game relative to the prior one.

Hybrid H_0 : This corresponds to the original CCA2 security game.

- **Setup Phase**
  1. The challenger samples a public/private key pair as $(\mathsf{cpa.sk}, \mathsf{cpa.pk}) \leftarrow \mathsf{CPA.Setup}(1^\kappa)$.
  2. It sends $\mathsf{cca.pk} = \mathsf{cpa.pk}$ to $\mathcal{A}$, and sets the secret key $\mathsf{cca.sk} = \mathsf{cpa.sk}$.
  3. Note that we will define $\kappa' = \kappa^\delta$.
- **Pre-challenge Query Phase**
  - *Decryption Queries*
    1. Let (vk, α, τ) where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, be the input ciphertext.
    2. If $\mathsf{Verify}(vk, \alpha, \tau) = 0$, the challenger outputs $\perp$.
    3. Otherwise, the challenger computes $d = \mathsf{PKE.Find}\,(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha)$ and outputs $\mathsf{PKE.Check}\,(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.
- **Challenge Phase**
  1. The adversary sends two challenge messages (m_0^*, m_1^*).
  2. **Part 1:**
     - The challenger samples $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
     - It chooses $s^* \leftarrow \{0,1\}^n$ and samples $(vk^*, sk^*) \leftarrow \mathsf{Signature.Setup}(1^{\kappa'})$.
     - It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and and $r_i^* \leftarrow \{0,1\}^\ell$.

    – For $i \in [n]$, it samples $v_i^* \leftarrow \{0,1\}^\kappa$, and sets $\sigma_i^* = \mathsf{Equiv.Com}(1^\kappa, s_i^*; v_i^*), y_i^* = s_i^*|v_i^*$.

3. **Part 2:**
    – It chooses a bit $\beta \in \{0,1\}$ and sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, 0) \oplus m_\beta^*$.
    – It sets $(c_{0,i}^*, c_{1,i}^*)$ as follows.
        * If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; \widetilde{r_i}^*)$, $c_{1,i}^* = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y_i^*; r_i^*)$
        * If $s_i^* = 1$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; r_i^*)$, $c_{1,i}^*\mathsf{CCA.Com}(1^{\kappa'}, vk^*, y_i^*; \widetilde{r_i}^*)$.
4. Finally, it sets $\alpha^* = \left(\mathsf{HPRG.pp}^*, 1^n, c^*, \left(c_{0,i}^*, c_{1,i}^*, \sigma_i^*\right)_{i\in[n]}\right)$.
5. It computes $\tau^* = \mathsf{Sign}(sk^*, \alpha^*)$ and sends (vk^*, α^*, τ^*) to $\mathcal{A}$.

- **Post-challenge Query Phase**
    – *Decryption Queries*
        1. Let (vk, α, τ) where $\alpha = \left(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\right)$, be the input ciphertext.
        2. If $\mathsf{Verify}(vk, \alpha, \tau) = 0$, the challenger outputs $\perp$.
        3. Otherwise, the challenger computes $d = \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha)$ and outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

- Finally, the adversary sends its guess b. The challenger outputs 1 if $b = \beta$ and 0 otherwise.

Hybrid H_1 : The same as H_0 except that the challenger samples $(vk^*, sk^*) \leftarrow \mathsf{Signature.Setup}(1^\kappa)$ in the Setup phase (but does not send them to the adversary until the challenge phase. If any query of the adversary, denoted by (vk, α, τ) is such that $vk = vk^*$, then instead of decrypting this query, the challenger outputs $\perp$. It continues the rest of the hybrid the same way as H_0.

Hybrid H_2 : This hybrid is the same as H_1 except that on any decryption query, denoted by (vk, α, τ) where α is parsed as $\left(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\right)$, the challenger returns $\mathsf{Special\text{-}Abort}$ to $\mathcal{A}$ if the decryption query satisfies the following condition:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

Hybrid H_3 : This will be the same as H_2 except for how the v_i^*, w_i^*, y_i^* values are sampled. In this hybrid, **Part 1** of the challenge phase occurs immediately after the setup phase, and is modified as follows.

1. The challenger samples $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(1^\lambda, 1^\ell)$.
2. It chooses $s^* \leftarrow \{0,1\}^n$ and samples $(vk^*, sk^*) \leftarrow \mathsf{Signature.Setup}(1^{\kappa'})$.
3. It sets $\widetilde{r_i}^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$ and and $r_i^* \leftarrow \{0,1\}^\ell$.
4. Next, it does the following:
    - For each $i \in [n]$, it samples random $(\sigma_i^*, v_{0,i}^*, v_{1,i}^*) \leftarrow \mathsf{Equiv.Equivocate}(1^\kappa)$.
    - It sets $y_{0,i}^* = 0|v_{0,i}^*$, and $y_{1,i}^* = 1|v_{1,i}^*$.

   We will denote by a^* the tuple of values $(\{y_{0,i}^*, y_{1,i}^*, \sigma_i^*\}_{i\in[\kappa]})$.

5. For each $i \in [n]$, set $y_i^* = y_{s_i,i}^*$.

We remark that the challenger in this game now takes exponential time to sample the $y_{0,i}^*, y_{1,i}^*$ values.

Hybrid H_4 : The same as H_3 with the exception that in the challenge phase, the challenger sets $(c_{0,i}^*, c_{1,i}^*)$ as follows.

- If $s_i^* = 0$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; \widetilde{r_i}^*)$, $c_{1,i}^* = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y_{1,i}^*; r_i^*)$
- If $s_i^* = 1$, it sets $c_{0,i}^* = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y_i^*; r_i^*)$, $c_{1,i}^* = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y_{1,i}^*; \widetilde{r_i}^*)$.

The effect of this change is that $c_{1,i}^*$ will be a commitment to $y_{1,i}^*$ no matter what s_i^* is. However, $c_{0,i}^*$ will remain dependent on s_i^*. It is an encryption of $y_{0,i}^*$ if $s_i^* = 0$ and an encryption of $y_{1,i}^*$ if $s_i^* = 1$.

Hybrid H_5 : The same as H_4 with the exception that the decryption algorithm uses an alternative "finding" algorithm PKE.FindAlternative in Figure 6 in place of PKE.Find. The main difference is that the algorithm PKE.FindAlternative runs in time polynomial in $2^{\kappa'}$ to open the commitment scheme via brute force in lieu of decrypting using a secret key. The new steps are:

1. Let $\mathsf{cca.ct} = (vk, \alpha, \tau)$ where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$ be the input ciphertext.
2. The challenger first computes $d = \mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \alpha)$.
3. It outputs $\mathsf{PKE.Check}(\mathsf{cca.pk}, \mathsf{cca.ct}, d)$.

$\mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \alpha)$

Inputs: Ciphertext $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$

Output: $d \in \{0,1\}^n$

- Let $\kappa = \mathsf{RetrieveParam}(\mathsf{cpa.pk})$ and $\kappa' = \kappa^e$.
- For each $i \in [n]$, do the following:
  1. Let $m_i = \mathsf{CCACom.Val}(1^{\kappa'}, c_{1,i})$.
  2. If $m_i = 1|v_i$, $\sigma_i = \mathsf{Equiv.Com}(1^\kappa, 1; v_i)$, set $d_i = 1$. Else set $d_i = 0$.
- Output $d = d_1 d_2 \dots d_n$.

Figure 6: Routine PKE.FindAlternative

Hybrid H_6 : The same as H_5 except that the abort condition introduced in H_1 is removed. That is, the challenger no longer sends Special-Abort to $\mathcal{A}$ if any of the adversary's decryption queries, denoted by (vk, α, τ) where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, satisfy the following condition:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c},\ \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

Instead, the experiment uses the alternative "finding" algorithm from H_5 to decrypt all queries, irrespective of whether or not they satisfy the above condition.

Hybrid H_7 : The same as H_6 with the exception that in the challenge phase: It sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_i)$, $c^*_{0,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; \widetilde{r_i}^*)$.

The effect of this change is that $c^*_{0,i}$ will become an encryption of $y^*_{0,i}$ no matter what s^*_i is.

Hybrid H_8 : The same as H_7 with the following syntax changes. For each $i \in [n]$ set $r^*_{s_i,i} = \widetilde{r_i}^*$ and set $r^*_{1-s_i,i} = r^*_i$. It sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_{0,i})$, $c^*_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; r^*_{1,i})$
- If $s^*_i = 1$, it sets $c^*_{1,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_{0,i})$, $c^*_{0,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; r^*_{1,i})$.

Hybrid H_9 : The same as H_8 except for each $i \in [n]$ $r^*_{0,i}$ and $r^*_{1,i}$ are both chosen uniformly at random. In addition, c^* is chosen uniformly at random. Note that this hybrid contains no information about the challenge message m^*_β.

6.3 Analysis

We will let $\mathsf{adv}^x_{\mathcal{A}}$ denote the probability that the challenger outputs 1 in Hybrid H_x. We note here that $\mathsf{adv}^x_{\mathcal{A}}$ takes values in $[0, 1]$.

Lemma 6.1. Assuming $(\mathsf{Sign}, \mathsf{Verify})$ is an existentially unforgeable under chosen message attack (EUF-CMA) secure signature scheme, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^0_{\mathcal{A}} - \mathsf{adv}^1_{\mathcal{A}}| \le \mathsf{negl}(\kappa)$.

Proof. The only difference between H_0 and H_1 is that the challenger in H_1 returns $\perp$ to $\mathcal{A}$ when asked to decrypt any query with $vk = vk^*$, where the challenge ciphertext is denoted by (vk^*, α^*, τ^*). By existential unforgeability of the signature scheme, $\mathcal{A}$ cannot output a valid signature for vk^* on any message (apart from α^*), except with probability $\mathsf{negl}(\kappa)$. Therefore, in any such query of $\mathcal{A}$, denoted by (vk^*, α, τ), that differs from the challenge ciphertext, the probability that $\mathsf{Verify}(vk^*, \alpha, \tau) = 1$ and $(\alpha, \tau) \neq (\alpha^*, \tau^*)$ is at most $\mathsf{negl}(\kappa)$. This completes the proof of the lemma. □

Lemma 6.2. Assuming that $\mathsf{Equiv.Com}$ is 2^{κ^δ} secure, for any adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^2_{\mathcal{A}}| \le \mathsf{negl}(\kappa)$.

Proof. The difference between H_1 and H_2 is that the challenger in H_2 returns $\mathsf{Special\text{-}Abort}$ to $\mathcal{A}$ if $\mathcal{A}$ makes a query denoted by (vk, α, τ) where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, that satisfies the following condition:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

But an adversary $\mathcal{A}$ that makes such a query, can be used to contradict 2^{κ^δ} binding security of $\mathsf{Equiv.Com}$, as follows. The reduction R interacts with the adversary $\mathcal{A}$, generating messages for the setup phase according to H_0. Next, on input a decryption query from $\mathcal{A}$ in the pre-challenge query phase, R uses $\mathsf{cca.sk}$ to check if the following holds:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

If this holds, R outputs $(\widetilde{b}, \widetilde{u})$ and $(\widetilde{c}, \widetilde{v})$ as two openings of the equivocal commitment. Note that CPA decryption is a polynomial time algorithm and $\mathsf{CCACom.Val}$ can be computed in time 2^{κ^δ}, therefore R runs in time 2^{κ^δ}. This contradicts binding of the equivocal commitment against uniform adversaries. □

Lemma 6.3. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for every unbounded adversary $\mathcal{A}$ and all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^2_{\mathcal{A}} - \mathsf{adv}^3_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. This claim follows by the definition of equivocality of $\mathsf{Equiv.Com}$. □

Lemma 6.4. Assuming Com is an e-computation enabled CCA secure commitment scheme per Definition 3.12, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^3_{\mathcal{A}} - \mathsf{adv}^4_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. For every κ, we consider a sequence of $n = n(\kappa)$ hybrids. We denote these by $H_{3,i}$ for $i \in [0, n]$, where $H_{3,0} \equiv H_3$ and $H_{3,n} \equiv H_4$. Moreover, for $j \in [1, n]$, $H_{3,j}$ is the same as H_3 except that for $i \in [1, j]$, the challenger sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_i; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_i; r^*_i)$, $c^*_{1,i} = \mathsf{CCA.Com}(1^{\kappa'}, vk^*, y^*_{1,i}; \widetilde{r_i}^*)$.

We will prove that for large enough $\kappa \in \mathbb{N}$, for every $i = i(\kappa) \in [1, n(\kappa)]$:

$$|\mathsf{adv}^{3,i-1}_{\mathcal{A}} - \mathsf{adv}^{3,i}_{\mathcal{A}}| = \mathsf{negl}(\kappa) \tag{10}$$

Towards a contradiction, assume that this equation is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$, there exists $i^* = i^*(\kappa) \in [n(\kappa)]$ such that:

$$|\mathsf{adv}^{3,i^*-1}_{\mathcal{A}} - \mathsf{adv}^{3,i^*}_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{11}$$

We will construct an adversary $\mathcal{B}$ that with oracle access to $\mathcal{A}$, contradicts e-computation enabled CCA security of the commitment $\mathsf{CCA.Com}$ according to Definition 3.12. $\mathcal{B}$ does the following:

- Execute the Setup phase exactly according to H_2, to obtain $(\mathsf{cca.pk}, \mathsf{cca.sk})$.
- Let a^* denote the results of the inefficient sampling procedure in Step 4 of H_2. Obtain (a^*, i^*) by submitting a Turing Machine to the e computation-enabled CCA oracle as follows.
  - The Turing Machine first runs the $\mathsf{Equiv.Equivocate}$ algorithm to compute a^*, which can be computed in time 2^κ. Note that the e computation-enabled CCA oracle allows for $2^{\kappa'^e} = 2^{\kappa'^{\frac{1}{\delta}}} = 2^\kappa$-time computations.

- Next the (randomized) Turing Machine, given the code of $\mathcal{A}$, estimates i^* by iterating over all possible $i \in [n(\kappa)]$ and estimating the advantage of $\mathcal{A}$ between successive experiments. The error in estimation can be reduced to $\mathsf{negl}(\kappa') = \mathsf{negl}(\kappa)$ as follows. For every $i \in [0, n]$, initialize $\widetilde{\mathsf{adv}}^i_{\mathcal{A}}$ to 0 and iterate the following $T' = \mathsf{poly}(\kappa'^{\log \kappa'})$ times.
  - $*$ Run an execution with $\mathcal{A}$ by generating challenger messages exactly according to $H_{1,i}$. If the challenger output equals 1, set $\widetilde{\mathsf{adv}}^i_{\mathcal{A}} = \widetilde{\mathsf{adv}}^i_{\mathcal{A}} + \frac{1}{T'}$.

  Finally, output $i^* \in [n]$ that maximizes $|\widetilde{\mathsf{adv}}^{i^*}_{\mathcal{A}} - \widetilde{\mathsf{adv}}^{i^*-1}_{\mathcal{A}}|$. Note that this process takes time $\mathsf{poly}(T') < 2^{\kappa'^e}$, and the error in estimation will be $\mathsf{negl}(\kappa)$, by a Chernoff bound.

- Next, start a fresh execution with $\mathcal{A}$ on public key $\mathsf{cca.pk}$. For every query of $\mathcal{A}$ in the pre-challenge query phase, denoted by (vk, α, τ) where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, forward $\{c_{1,i}\}_{i\in[n]}$ to the CCA commitment valuation oracle. Use the values returned by the CCA commitment oracle, and $\mathsf{cca.sk}$, to efficiently check and return $\mathsf{Special\text{-}Abort}$ to $\mathcal{A}$ on input any query of $\mathcal{A}$ that satisfies the following condition:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

- For the challenge query, sample $\mathsf{HPRG.pp}^*, s^*$ and compute $\widetilde{r}^*_{i^*}$ as well as $\{\widetilde{r}^*_j, r^*_j\}_{j\in[n]\setminus\{i^*\}}$ according to H_2. Sample $\{y^*_{0,i}, y^*_{1,i}\}_{i\in[n]}$ according to H_2. Output messages (x_0, x_1) for the commitment challenger as follows:

$$x_0 = y^*_{0,i^*}, x_1 = y^*_{1,i^*}$$

  Obtain from the challenger a commitment string γ (that is a commitment to $x_{b'}$ for some $b' \in \{0,1\}$), and proceed to create the challenge ciphertext by executing Part 2 of the Challenge Phase exactly according to the strategy in H_{3,i^*-1}, sampling uniform $\beta \in \{0,1\}$. The only exception is that if $s^*_{i^*} = 0$, set $c^*_{1,i^*} = \gamma$.

- Run $\mathcal{A}$ on the challenge ciphertext. For every query of $\mathcal{A}$ in the post-challenge query phase, denoted by (vk, α, τ) where α equals
$\Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, forward $\{c_{1,i}\}_{i\in[n]}$ to the CCA commitment valuation oracle. Use the values returned by the CCA commitment oracle, and use $\mathsf{cca.sk}$, to efficiently check and output $\perp$ if any query of $\mathcal{A}$ satisfies the following condition:
$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

- Finally, obtain output b from $\mathcal{A}$ and output 1 iff β (chosen by $\mathcal{B}$) equals b. By assumption,

$$\left|\mathsf{adv}^{3,i^*-1}_{\mathcal{A}} - \mathsf{adv}^{3,i^*}_{\mathcal{A}}\right| \geq \frac{1}{p(\kappa)}.$$

We observe that if γ is a commitment to x_0, then the experiment corresponds to H_{3,i^*-1} and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^{3,i^*-1}_{\mathcal{A}})$. On the other hand, if γ is a commitment to x_1, then the experiment corresponds to H_{3,i^*} and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^{3,i^*}_{\mathcal{A}})$.

This implies that

$$\Big|\Pr[\mathcal{B}=1|b'=0]-\Pr[\mathcal{B}=1|b'=1]\Big| \geq \frac{1}{p(\kappa)}$$

which is a contradiction to the CCA security of the commitment. This completes the proof. ☐

Lemma 6.5. For every unbounded adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^4_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \geq \frac{1}{p(\kappa)}.$$

The only difference between H_4 and H_5 is that in H_5, the adversary's queries ct are decrypted using the $\mathsf{PKE.FindAlternative}$ algorithm to obtain d and then $\mathsf{PKE.Check}$ is run on input d. On the other hand in H_4, the adversary's queries ct are decrypted using $\mathsf{PKE.Find}$ to obtain d followed by running $\mathsf{PKE.Check}$ on d.

Consider any ciphertext $\mathsf{cca.ct} = (vk, \alpha, \tau)$ where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$. We will define a set $\mathsf{BAD}_{\mathsf{CT}}$ of "bad" pre-challenge and post-challenge ciphertexts as follows:

$\mathsf{cca.ct} \in \mathsf{BAD}_{\mathsf{CT}} \iff$ for $\mathsf{cca.ct} = (vk, \alpha, \tau)$ where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$,
$\exists i \in [n]$ s.t. for $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{i,0}), m'_i = \mathsf{CCACom.Dec}(1^{\kappa'}, c_{i,1})$, Eqns (12) and (13) are true.

$$\exists v_i \text{ s.t. } m_i = (0|v_i), \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i) \tag{12}$$

$$\exists v'_i \text{ s.t. } m'_i = (1|v'_i), \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 1; v'_i) \tag{13}$$

Claim 6.1. Let $\mathbb{S}$ denote the set of all queries of $\mathcal{A}$, such that each query in $\mathbb{S}$, denoted by (vk, α, τ) with $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, satisfies the following condition:

$$\forall i \in [n], \text{ for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\text{either } \widetilde{b} = \widetilde{c}, \text{ or } \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) \neq \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

Then,

$$\Pr[(\mathbb{S} \cap \mathsf{BAD}_{\mathsf{CT}}) \neq \phi] \geq \frac{1}{p(\kappa)}.$$

Proof. By construction of both hybrids, for all queries outside the set $\mathbb{S}$, the challenger returns $\perp$.

Now because the two hybrids only differ in the way the adversary's queries are decrypted, with probability $\frac{1}{p(\kappa)}$, $\mathcal{A}$ necessarily makes a pre-challenge query $\mathsf{cca.ct} = (vk, \alpha, \tau)$ in $\mathbb{S}$ such that:

$$\begin{aligned}&\mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha))\\ &\neq \mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, \mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \alpha)).\end{aligned} \tag{14}$$

Recall that by description of $\mathsf{PKE.Check}$, for every pre-challenge query $\mathsf{cca.ct} = (vk, \alpha, \tau)$ of $\mathcal{A}$ and every possible $d \in \{0,1\}^n$, the output of $\mathsf{PKE.Check}$ on input $(\mathsf{cca.pk}, \alpha, d)$ is $\big(c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, d, 0)\big)$, if and only if:

1. If $d_i = 0$, the following holds for $m \leftarrow \mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{i,0}, \widetilde{r_i})$.
   - $\mathsf{CPA.Enc}(\mathsf{cpa.pk}, m; \widetilde{r_i}) = c_{i,0}$.
   - Parse $m = s'|v'$ for $s' \in \{0,1\}$ and $v' \in \{0,1\}^\kappa$. Then $s' = 0, \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.
2. If $d_i = 1$, the following holds for $m \leftarrow \mathsf{Com.Recover}(1^{\kappa'}, c_{i,1}, \widetilde{r_i})$.
   - $\mathsf{Com}(1^{\kappa'}, m; \widetilde{r_i}) = c_{i,1}$.
   - Parse $m = s'|v'$ for $s' \in \{0,1\}$ and $v' \in \{0,1\}^\kappa$. Then $s' = 1, \sigma_i = \mathsf{Equiv.Com}(1^\kappa, s'; v')$.

Otherwise, the output of $\mathsf{PKE.Check}$ on input $\mathsf{cca.pk}, \alpha$ and d is $\perp$.

Next, fix any $\mathsf{cca.pk}, \alpha$ that satisfies equation (14). For all $i \in [n]$,

- Let $m_i = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{i,0}), m'_i = \mathsf{Com.Dec}(1^{\kappa'}, c_{i,1})$.
- Set $d_i = 0 \iff m_i$ satisfies equation (12),
- Set $d'_i = 1 \iff m'_i$ satisfies equation (13).

Note that equation (14) implies that:

$$\mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, d) \neq \mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, d') \tag{15}$$

which implies that $d \neq d'$, that is, $\exists j \in [n]$ such that $d_j \neq d'_j$ (where d_j and d'_j denote the j^{th} bit of d and d' respectively). Then by definition of d and d', one of the following is true:

- $d_j = 0, d'_j = 1 \implies m_j$ satisfies equation (12) and m'_j satisfies equation (13).
- $d_j = 1, d'_j = 0 \implies m_j$ doesn't satisfy equation (12) and m'_j does not satisfy equation (13).
  $\implies \mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, d) = \perp$ and $\mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, d') = \perp$.
  (which contradicts Equation (15)).

We note that the implications above follow by definition of d and d', by perfect correctness of the CPA encryption scheme which implies that $\mathsf{CPA.Recover}(\mathsf{cpa.pk}, c_{i,0}, \widetilde{r_i}) = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{i,0})$, and by perfect binding of the commitment scheme which implies $\mathsf{CCACom.Recover}(1^{\kappa'}, c_{i,1}, \widetilde{r_i}) = \mathsf{CCACom.Val}(1^{\kappa'}, c_{i,1})$.

Therefore,

$$\begin{aligned}&\mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, \mathsf{PKE.Find}(\mathsf{cca.pk}, \mathsf{cca.sk}, \alpha))\\ \neq\ &\mathsf{PKE.Check}(\mathsf{cca.pk}, \alpha, \mathsf{PKE.FindAlternative}(\mathsf{cca.pk}, \alpha))\end{aligned}$$

if and only if $\exists j \in [n]$ such that for $m_j = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{i,0})$ and $m'_j = \mathsf{Com.Dec}(1^{\kappa'}, c_{i,1})$, equations (12) and (13) are both true. This completes the proof of the claim. □

Claim 6.2. Let $\mathbb{S}$ denote the set of all pre-challenge queries of $\mathcal{A}$. There exists a negligible function $\delta(\cdot)$ such that:

$$\mathbb{S} \cap \mathsf{BAD}_{\mathsf{CT}} = \phi$$

Proof. By definition, every query in $\mathbb{S}$, denoted by (vk, α, τ) with $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, satisfies the following condition:

$$\forall i \in [n], \text{ for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\text{either } \widetilde{b} = \widetilde{c}, \text{ or } \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) \neq \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

By definition of $\mathsf{BAD_{CT}}$,

$$\begin{aligned}\mathsf{cca.ct} \in \mathsf{BAD_{CT}} \iff & \text{ for } \mathsf{cca.ct} = (vk, \alpha, \tau) \text{ where } \alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big),\\ & \exists i \in [n] \text{ s.t. } (0|v_i) = \mathsf{CPA.Dec}(\mathsf{cpa.sk}, c_{i,0}), (1, v_i') = \mathsf{CCACom.Dec}(1^{\kappa'}, c_{i,1}),\\ & \sigma_i = \mathsf{Equiv.Com}(1^\kappa, 0; v_i) = \mathsf{Equiv.Com}(1^\kappa, 1; v_i')\end{aligned}$$

Therefore, by definition, $\mathbb{S} \cap \mathsf{BAD_{CT}} = \phi$. □

Claims 5.1 and 5.2 lead to a contradiction, therefore proving the lemma. □

Lemma 6.6. For every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^5_{\mathcal{A}} - \mathsf{adv}^6_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. The only difference between the two hybrids is that the extra $\mathsf{Special\text{-}Abort}$ condition is removed. That is, the challenger no longer sends $\mathsf{Special\text{-}Abort}$ to $\mathcal{A}$ if any of the adversary's decryption queries, denoted by (vk, α, τ) where $\alpha = \Big(\mathsf{HPRG.pp}, 1^n, c, (c_{0,i}, c_{1,i}, \sigma_i)_{i\in[n]}\Big)$, satisfy the following condition:

$$\exists i \in [n] \text{ s.t. for } \widetilde{b}|\widetilde{u} = \mathsf{CPA.Dec}(\mathsf{cca.sk}, c_{0,i}) \text{ and } \widetilde{c}|\widetilde{v} = \mathsf{CCACom.Val}(c_{1,i}),$$
$$\widetilde{b} \neq \widetilde{c}, \text{ and } \sigma_i = \mathsf{Equiv.Com}(\widetilde{b}; \widetilde{u}) = \mathsf{Equiv.Com}(\widetilde{c}; \widetilde{v}).$$

Therefore to argue that the hybrids are indistinguishable, it suffices to prove that the special-abort condition occurs in hybrid 5 with negligible probability. By Lemmas 6.2 to 6.5, we have that for every PPT adversary $\mathcal{A}$, there exists a negligible function δ such that $|\mathsf{adv}^1_{\mathcal{A}} - \mathsf{adv}^5_{\mathcal{A}}| \leq \delta(\kappa)$. This implies that

$$\Pr[\mathsf{Special\text{-}Abort}|H_5] \leq \Pr[\mathsf{Special\text{-}Abort}|H_1] + \mathsf{negl}(\kappa) \leq \mathsf{negl}(\kappa).$$

This completes the proof of the claim. □

Lemma 6.7. Assuming $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is PKE scheme which is $T = 2^{\kappa^\delta}$-IND-CPA secure against *non-uniform* adversaries (where δ is the constant used in the construction) per Definition 3.3, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^6_{\mathcal{A}} - \mathsf{adv}^7_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial function $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^6_{\mathcal{A}} - \mathsf{adv}^7_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{16}$$

For every κ, we consider a sequence of $n = n(\kappa)$ hybrids. We denote these by $H_{6,i}$ for $i \in [0, n]$, where $H_{6,0} \equiv H_6$ and $H_{6,n} \equiv H_7$. Moreover, for $j \in [1, n]$, $H_{6,j}$ is the same as H_6 except that for $i \in [1, j]$, the challenger sets sets $(c^*_{0,i}, c^*_{1,i})$ as follows.

- If $s^*_i = 0$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; \widetilde{r_i}^*)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; r^*_i)$
- If $s^*_i = 1$, it sets $c^*_{0,i} = \mathsf{CPA.Enc}(\mathsf{cpa.pk}, y^*_{0,i}; r^*_i)$, $c^*_{1,i} = \mathsf{Com}(1^{\kappa'}, y^*_{1,i}; \widetilde{r_i}^*)$.

By equation (16), for infinitely many $\kappa \in \mathbb{N}$, there exists $i^* = i^*(\kappa) \in [1, n(\kappa)]$ such that:

$$|\mathsf{adv}_{\mathcal{A}}^{6,i^*-1} - \mathsf{adv}_{\mathcal{A}}^{6,i^*}| \geq \frac{1}{n(\kappa) \cdot p(\kappa)} \tag{17}$$

We will construct an adversary $\mathcal{B}$ that obtains $i^*(\kappa)$ as non-uniform advice and with oracle access to $\mathcal{A}$, breaks $T = 2^{\kappa^\delta}$-IND-CPA security of the PKE scheme against non-uniform attackers according to Definition 3.3.

First, note that H_{6,i^*-1} and H_{6,i^*} are identical until just before Part 2 of the challenge phase. Let z denote the "best-possible" fixing of $a^* = (\{y^*_{0,i}, y^*_{1,i}, \sigma^*_i\}_{i\in[n]})$ such that conditioned on these fixed values, $|\mathsf{adv}_{\mathcal{A}}^{6,i^*-1} - \mathsf{adv}_{\mathcal{A}}^{6,i^*}|$ takes the highest value (over all possible choices of z). Thus z is such that,

$$\left|(\mathsf{adv}_{\mathcal{A}}^{6,i^*-1}|z) - (\mathsf{adv}_{\mathcal{A}}^{6,i^*}|z)\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}.$$

$\mathcal{B}$ obtains z as non-uniform advice, and computes $\mathsf{HPRG.pp}^*$ for the Setup phase according to H_6. It then outputs messages (x_0, x_1) for the CPA encryption challenger as follows:

$$x_0 = y^*_{1,i^*}, x_1 = y^*_{0,i^*}$$

It obtains from the challenger a public key $\mathsf{cpa.pk}$, and a ciphertext γ (that is an encryption of $x_{b'}$ for some $b' \in \{0,1\}$). It outputs $\mathsf{cca.pk} = \mathsf{cpa.pk}$ to $\mathcal{A}$. In the pre-challenge query phase, it decrypts $\mathcal{A}$'s queries by using subroutines $\mathsf{PKE.FindAlternative}$ (in time polynomial in $2^{\kappa'}$) and $\mathsf{PKE.Check}$. Next, in the challenge phase, on input (m^*_0, m^*_1) from $\mathcal{A}$, it proceeds to create the challenge ciphertext exactly according to the strategy in H_{6,i^*-1} by sampling uniform $\beta \in \{0,1\}$, except that if $s^*_{i^*} = 1$, it sets $c^*_{1,i^*} = \gamma$.

$\mathcal{B}$ then runs $\mathcal{A}$ on the challenge ciphertext, and in the post-challenge query phase, it decrypts $\mathcal{A}$'s queries by using subroutines $\mathsf{PKE.FindAlternative}$ (in time polynomial in $2^{\kappa'}$) and $\mathsf{PKE.Check}$. Finally $\mathcal{B}$ outputs 1 iff β (chosen by $\mathcal{B}$) equals b (which is the output of $\mathcal{A}$). By assumption,

$$\left|(\mathsf{adv}_{\mathcal{A}}^{6,i^*-1}|z) - (\mathsf{adv}_{\mathcal{A}}^{6,i^*}|z)\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}.$$

We observe that if γ is an encryption of x_0, then the experiment corresponds to H_{6,i^*-1} and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{6,i^*-1}|z)$. On the other hand, if γ is an encryption of x_1, then the experiment corresponds to H_{6,i^*} and $\Pr[\beta = b]$ is simply $(\mathsf{adv}_{\mathcal{A}}^{6,i^*}|z)$.

This implies that

$$\left|\Pr[\mathcal{B}(z) = 1|b' = 0] - \Pr[\mathcal{B}(z) = 1|b' = 1]\right| \geq \frac{1}{n(\kappa) \cdot p(\kappa)}$$

which is a contradiction to the $T = 2^{\kappa^\delta}$-IND-CPA security of the PKE against *non-uniform* adversaries. This completes the proof. □

Lemma 6.8. For any adversary $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}_{\mathcal{A}}^{7} - \mathsf{adv}_{\mathcal{A}}^{8}| = 0$.

Proof. These hybrids are identical to each other. The only change is that we rename $\widetilde{r_i}^*$ to r^*_{i,s_i} and r^*_i to $r^*_{i,1-s_i}$. □

Lemma 6.9. Assuming $(\mathsf{Setup}, \mathsf{Eval})$ is a 2^{κ^δ} secure hinting PRG scheme against *non-uniform* adversaries per Definition 3.10, for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^8_{\mathcal{A}} - \mathsf{adv}^9_{\mathcal{A}}| \leq \mathsf{negl}(\kappa)$.

Proof. Towards a contradiction, assume that the lemma is not true. That is, there exists a PPT adversary $\mathcal{A}$ and a polynomial $p(\cdot)$ such that for infinitely many $\kappa \in \mathbb{N}$,

$$|\mathsf{adv}^8_{\mathcal{A}} - \mathsf{adv}^9_{\mathcal{A}}| \geq \frac{1}{p(\kappa)} \tag{18}$$

We will construct an adversary $\mathcal{B}$ that with oracle access to $\mathcal{A}$, breaks sub-exponential security of the hinting PRG against non-uniform attackers according to Definition 3.10.

The only difference between H_8 and H_9 is that in H_8, for all $i \in [n]$, r^*_{i,s_i} is chosen as the output of $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, i)$, $r^*_{i,1-s_i}$ is chosen uniformly at random and c^* is chosen as $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s^*, 0) \oplus m^*_\beta$ whereas in H_9, r^*_{i,s_i}, $r^*_{i,1-s_i}$ and c^* are chosen uniformly at random. Also note that H_8 and H_9 are identical until the pre-challenge query phase.

Let z denote the "best-possible" fixing of $a^* = (\{y^*_{0,i}, y^*_{1,i}, \sigma^*_i\}_{i \in [\kappa]})$ such that conditioned on these fixed values, $|\mathsf{adv}^8_{\mathcal{A}} - \mathsf{adv}^9_{\mathcal{A}}|$ takes the highest value (over all possible choices of z). Thus z is such that,

$$\left|(\mathsf{adv}^8_{\mathcal{A}}|z) - (\mathsf{adv}^9_{\mathcal{A}}|z)\right| \geq \frac{1}{p(\kappa)}.$$

$\mathcal{B}$ obtains z as non-uniform advice. It then begins its game with the hinting PRG challenger, and externally obtains $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i \in [n], b \in \{0,1\}}\right)$. It then sets $r^*_{i,b} = \gamma_{i,b}$ for $i \in [n], b \in \{0,1\}$ and sets $c^* = \gamma_0 \oplus m^*_\beta$. It samples the CPA public key, answers queries in the pre-challenge and post-challenge query phase, and creates the rest of the challenge ciphertext exactly according to the strategy in H_8.

Finally, $\mathcal{B}$ runs $\mathcal{A}$ on the challenge ciphertext, and outputs 1 iff β (chosen by $\mathcal{B}$) equals b (which is the output of $\mathcal{A}$). Note that $\mathcal{B}$ uses the subroutine $\mathsf{PKE.FindAlternative}$, and therefore runs in time polynomial in $2^{\kappa'} = 2^{\kappa^\delta}$. By assumption,

$$\left|(\mathsf{adv}^8_{\mathcal{A}}|z) - (\mathsf{adv}^9_{\mathcal{A}}|z)\right| \geq \frac{1}{p(\kappa)}.$$

We observe that if $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i \in [n], b \in \{0,1\}}\right)$ is the output of a hinting PRG on a uniform seed, then the experiment corresponds to H_8 and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^8_{\mathcal{A}}|z)$. On the other hand, if $\left(\gamma_0, \left\{\gamma_{i,b}\right\}_{i \in [n], b \in \{0,1\}}\right)$ is uniformly random, then the experiment corresponds to H_9 and $\Pr[\beta = b]$ is simply $(\mathsf{adv}^9_{\mathcal{A}}|z)$.

This implies that

$$\left|\Pr[\mathcal{B}(z) = 1 | b' = 0] - \Pr[\mathcal{B}(z) = 1 | b' = 1]\right| \geq \frac{1}{p(\kappa)}$$

which is a contradiction to the non-uniform security of the hinting PRG. This completes the proof. □

Lemma 6.10. For any adversary $\mathcal{A}$, for all $\kappa \in \mathbb{N}$, $|\mathsf{adv}^9_{\mathcal{A}}| \leq \frac{1}{2}$.

Proof. The challenge ciphertext component c^* now contains no information about β and β is not reflected anywhere else in the experiment. Thus the probability that $b = \beta$ is $\frac{1}{2}$ where the game does not abort. When it does abort the challenger outputs 0. Therefore the advantage of any attacker, which is the probability that the challenger outputs 1, is $\leq \frac{1}{2}$. $\square$

Theorem 6.1. Assume the following: (1) $\mathsf{CCA.Com}$ is a non-interactive e-computation enabled CCA secure commitment scheme per Definition 3.12, (2) $(\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ is 2^{κ^δ} binding secure against *uniform* adversaries and equivocal (3) $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is PKE scheme which is $T = 2^{\kappa^\delta}$-IND-CPA secure against non-uniform adversaries per Definition 3.3 and (4) $(\mathsf{Setup}, \mathsf{Eval})$ is a 2^{κ^δ}-secure hinting PRG scheme against non-uniform adversaries per Definition 3.10.

Then for any uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\kappa \in \mathbb{N}$, $\mathsf{adv}^0_{\mathcal{A}} \leq \mathsf{negl}(\kappa)$ and the construction of Section 6 is CCA2 secure against uniform attackers.

Proof. The proof of this Theorem follows immediately from Lemmas 6.2 to 6.10. $\square$

References

[BD18] Amos Beimel and Stefan Dziembowski, editors. *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, volume 11239 of *Lecture Notes in Computer Science*. Springer, 2018.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. Multi-collision resistant hash functions and their applications. In Nielsen and Rijmen [NR18], pages 133–161.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 671–684. ACM, 2018.

[BKSW18] Saikrishna Badrinarayanan, Dakshita Khurana, Amit Sahai, and Brent Waters. Upgrading to functional encryption. In Beimel and Dziembowski [BD18], pages 629–658.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In Beimel and Dziembowski [BD18], pages 209–234.

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Krawczyk [Kra98], pages 1–12.

[BM19] Alexandra Boldyreva and Daniele Micciancio, editors. *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*. Springer, 2019.

[CIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the Thirtieth Annual ACM STOC, Dallas, Texas, USA, May 23-26, 1998*, pages 141–150, 1998.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '10, pages 541–550, 2010.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [Kra98], pages 13–25.

[CSV20] Wouter Castryck, Jana Sotkov, and Frederik Vercauteren. Breaking the decisional diffie-hellman problem for class group actions using genus theory. Cryptology ePrint Archive, Report 2020/151, 2020. `https://eprint.iacr.org/2020/151`.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

[DPP93] Ivan B Damgård, Torben P Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *Annual International Cryptology Conference*, pages 250–265. Springer, 1993.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptology*, 26(1):80–101, 2013.

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013.

[GKLW20] Rachit Garg, Dakshita Khurana, George Lu, and Brent Waters. Black-box non-interactive non-malleable commitments. Manuscript, 2020.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[HM96] Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '96, pages 201–215, 1996.

[JO16] Jakob Jakobsen and Claudio Orlandi. On the CCA (in)security of mtproto. In *Proc. of the 6th Workshop on Security and Privacy in Smartphones Mobile Devices, SPSM@CCS 2016*, pages 113–116, 2016.

[KK19] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In Boldyreva and Micciancio [BM19], pages 552–582.

[KMT19] Fuyuki Kitagawa, Takahiro Matsuda, and Keisuke Tanaka. CCA security and trapdoor functions via key-dependent-message security. In Boldyreva and Micciancio [BM19], pages 33–64.

[KNY18] Ilan Komargodski, Moni Naor, and Eylon Yogev. Collision resistant hashing for paranoids: Dealing with multiple collisions. In Nielsen and Rijmen [NR18], pages 162–194.

[Kra98] Hugo Krawczyk, editor. *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[KS17] Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In Umans [Uma17], pages 564–575.

[KW19] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 671–700. Springer, 2019.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Umans [Uma17], pages 576–587.

[LS19] Alex Lombardi and Luke Schaeffer. A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279, 2019. `https://eprint.iacr.org/2019/279`.

[NR18] Jesper Buus Nielsen and Vincent Rijmen, editors. *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM STOC, 1990.*, pages 427–437. ACM, 1990.

[Pas16] Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. *Computational Complexity*, 25(3):607–666, 2016.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 433–444, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[Uma17] Chris Umans, editor. *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017.* IEEE Computer Society, 2017.