Black-Box Non-Interactive Non-Malleable Commitments

Rachit Garg*, Dakshita Khurana**, George Lu*, and Brent Waters***

Abstract. There has been recent exciting progress on building non-interactive non-malleable commitments from judicious assumptions. All proposed approaches proceed in two steps. First, obtain simple "base" commitment schemes for very small tag/identity spaces based on a various sub-exponential hardness assumptions. Next, assuming sub-exponential non-interactive witness indistinguishable proofs (NIWIs), and variants of keyless collision resistant hash functions, construct non-interactive compilers that convert tag-based non-malleable commitments for a small tag space into tag-based non-malleable commitments for a larger tag space.
We propose the first black-box construction of non-interactive non-malleable commitments. Our key technical contribution is a novel implementation of the non-interactive proof of consistency required for tag amplification. Prior to our work, the only known approach to tag amplification without setup and with black-box use of the base scheme (Goyal, Lee, Ostrovsky and Visconti, FOCS 2012) added multiple rounds of interaction.
Our construction satisfies the strongest known definition of non-malleability, i.e., CCA (chosen commitment attack) security. In addition to being black-box, our approach dispenses with the need for sub-exponential NIWIs, that was common to all prior work. Instead of NIWIs, we rely on sub-exponential hinting PRGs which can be obtained based on a broad set of assumptions such as sub-exponential CDH or LWE.

1 Introduction

Non-malleable commitments have been a well studied primitive in cryptography since their introduction by Dolev, Dwork and Naor [11]. They are an important component of nearly all multi-party protocols including multiparty computation, coin flipping and secure auctions. These commitments ensure security in the presence of "man in the middle" attacks. A man-in-the-middle adversary participates in two or more instantiations of

* University of Texas at Austin. Email: {rachg96, gclu}@cs.utexas.edu.
** University of Illinois Urbana-Champaign. Email: dakshita@illinois.edu. This material is based on work supported in part by DARPA under Contract No. HR001120C0024. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.
*** University of Texas at Austin and NTT Research. Email: bwaters@cs.utexas.edu.

a protocol, trying to use information obtained in one execution to breach security in the other protocol execution. A non-malleable protocol should ensure that an adversary gains no advantage from such behavior.

Non-Interactive Non-Malleable Commitments. For several years, provably secure constructions of non-malleable commitments required several rounds of interaction. On the other hand, practical constructions need to be highly efficient and often non-interactive. For these reasons, in practice, we often heuristically assume that a family of (keyless) SHA-like hash functions is non-malleable. Our technique gives the first provably secure black-box construction of non-interactive non-malleable commitments, taking us a step closer to efficient realizations.

We will focus on perfectly binding and computationally hiding non-interactive commitments. For these commitments, the perfect binding requirement asserts that a commitment cannot be opened to two different messages $m \neq m'$. Specifically, even for a maliciously generated commitment string c, there do not exist two openings to messages m and m' such that $m \neq m'$. The (computational) hiding property asserts that for any two messages, m and m' (of the same length), the distributions of commitments $\mathsf{com}(m)$ and $\mathsf{com}(m')$ are computationally indistinguishable.

Loosely speaking, a commitment scheme is said to be non-malleable if no adversary, given a commitment $\mathsf{com}(m)$, can efficiently generate a commitment $\mathsf{com}(m')$, such that the message m' is related to the original message m. This is equivalent (assuming the existence of one-way functions) to a tag-based notion where the commit algorithm obtains an additional input, a $\mathsf{tag} \in \{0,1\}^\kappa$, and where the adversary is restricted to using a tag, or identity, that is different from the tag used to generate its input commitment. We will rely on tag-based definitions throughout this paper. We will also model man-in-the-middle security as a CCA (chosen commitment attack) game between the adversary and a challenger.

Specifically, the hiding game is modified to give the adversary oracle access to an inefficient value function $\mathsf{CCA.Val}$ where on input a string c, $\mathsf{CCA.Val}(\mathsf{tag}, c)$ returns m if $\mathsf{CCA.Com}(\mathsf{tag}, m; r) \rightarrow c$ for some r. The adversary must first specify a challenge tag^* along with messages m_0^*, m_1^*. He is then allowed oracle access to $\mathsf{CCA.Val}(\mathsf{tag}, \cdot)$ for every $\mathsf{tag} \neq \mathsf{tag}^*$, and can make an arbitrary (polynomial) number of queries before and after obtaining the challenge commitment. [1] This CCA definition is the strongest

[1] The assumption that the commitment takes input a tag is w.l.o.g when the tag space is exponential. As is standard with non-malleable commitments, tags can be generically removed from this construction by setting the tag as the verification key of a signature scheme, and signing the commitment string using the signining key.

known definition of non-malleability. In the non-interactive setting, the often-used definition of (concurrent) non-malleability w.r.t. commitment is implied by this definition where the adversay is only allowed to make parallel oracle queries once it obtains the challenge commitment.

Our Results, in a Nutshell. In this work, we give the first black-box construction of CCA secure commitments, under weaker assumptions than prior work. In terms of assumptions, we substitute NIWIs with hinting PRGs [25] which can be instantiated under several standard assumptions like CDH and LWE. Additionally, while all prior work recursively applied NIWIs to prove cryptographic statements, making heavy non-black-box use of cryptography, our constructions are black-box. Combining this with base schemes due to [21], we obtain CCA secure commitments from black box use of the following assumptions: subexponential hinting PRGs, subexponential keyless collision-resistant hash functions, subexponential one-way functions against quantum adversaries, and subexponential one-way functions in BQP with hardness against classical adversaries. We note that subexponential hinting PRGs can be obtained based on black-box use of any group where CDH is subexponentially hard.

We believe this takes us one step closer to the goal of building provably secure and efficient non-interactive non-malleable commitments.

Prior Work on Non-Malleable Commitments. There has been a long line of work constructing non-malleable commitments in the plain model, without trusted setup. This research has been driven by two often competing goals: the first is to reduce the round complexity of commitment, which is important because it directly impacts the round complexity of applications like MPC. The second goal is to achieve non-malleable commitments under the weakest possible assumptions.

This research [11, 1, 31, 32, 29, 30, 26, 35, 34, 27, 14, 15, 18, 16, 9] culminated in three round stand-alone secure non-malleable commitments based on injective one-way functions [17] and concurrenct secure non-malleable commitments based on DDH/LWE [22], or subexponential injective one-way functions [8]. In the two round setting, we now have constructions based on sub-exponential time-lock puzzles [28] and sub-exponential DDH/LWE/QR/NR [23].

Very recently, research in non-malleable commitments moved to a final frontier of achieving non-interactive non-malleable commitments from well-studied assumptions without leveraging setup. In this non-interactive setting, Pandey, Pass and Vaikuntanathan [30] first gave constructions of non-malleable commitments based on a strong non-falsifiable assumption.

The primary research challenge has been to improve assumptions while realizing non-malleability without interaction and setup, which does not allow the use of tools like zero knowledge proof systems.

Nevertheless, the recent works of Bitansky and Lin [4] and Kalai and Khurana [21] made progress on improving these assumptions. All of these works [23, 28, 4, 21] proceed in two steps. First, they construct "base" commitment schemes that only support a constant-sized space of tags. Second, they give amplification techniques to convert commitments supporting a small space of tags into commitments that support a much larger tag space. Applying these amplification techniques to the base scheme helps generically increase the space of tags to $\{0,1\}^\kappa$. We summarize known results in the non-interactive setting by splitting up contributions into base constructions and tag amplification results.

Base Constructions. Three recent works [28, 4, 21] build non-interactive base schemes: non-malleable commitments for a tag space of size $c \log\log \kappa$ for a specific constant $c > 0$, based on various hardness assumptions. These are typically only secure in a setting where the adversary is restricted to using the same tag in all its queries to the $\mathsf{CCA.Val}$ oracle. This is primarily achieved by using families of assumptions, each of which is harder than the other along some axis of hardness. We list these assumptions below.

1. Lin, Pass and Soni [28] assume a sub-exponential variant of the hardness of time-lock puzzles. Specifically, they define a two-dimensional variant of the Rivest, Shamir and Wagner (RSW) repeated squaring assumption there is a security parameter n and another parameter t, and it is required that computing $h = g^{2^{2^t}}$ cannot be done by circuits of overall size 2^{n^ϵ} and depth 2^{t^δ}, for constants ϵ and δ.
2. Bitansky and Lin [4] rely on sub-exponentially hard one-way functions that admit a strong form of hardness amplification. Roughly speaking, they say that a one-way function f is amplifiable, if there is a way to combine (XOR), say ℓ hardcore bits corresponding to ℓ independent images $f(x_1), \ldots, f(x_\ell)$ that are each hard against T-time adversaries, so that the combined bit is 2^{ℓ^ϵ}-unpredicatable against T'-time adversaries; that is, the level of unpredictability increases at least subexponentially as more hardcore bits are combined (their assumption on unpredictability goes beyond the limit $\mathsf{poly}(\frac{T}{T'})$ that is commonly imposed by known provable results on hardness amplification).
3. Kalai and Khurana [21] assume classically sub-exponentially hard but quantum easy one-way functions (which can be based, e.g., on sub-

exponential hardness of DDH), and sub-exponentially quantum hard one-way functions (which can be based, e.g., on sub-exponential hardness of LWE).

Tag Amplification. Starting with non-malleable commitments for a tag space of size $c \log \log \kappa$ for a specific constant $c > 0$ (or sometimes even smaller), several works develop techniques to achieve non-malleable commitments for a tag space of $\{0,1\}^\kappa$. This is achieved by several applications of a tag-amplification compiler, that increases the tag space exponentially in each application. We also point out that these compilers often obtain as input base schemes that are secure against a restricted adversary; one that uses the same tag in all its queries to the $\mathsf{CCA.Val}$ oracle. The end goal, however, is to obtain security against a general adversary, that uses arbitrary tags in its oracle queries – as long as all tags in oracle queries are different from the challenge tag.

Such compilers were developed in [28, 4, 21] based various assumptions, and we summarize these results below.

- Lin, Pass and Soni [28] assume sub-exponential non-interactive witness indistinguishable (NIWI) proofs and keyless collision resistant hash functions against uniform adversaries. The resulting commitments for larger tags are secure only against uniform adversaries.
- Bitansky and Lin [4] assume sub-exponential non-interactive witness indistinguishable (NIWI) proofs and keyless collision resistant hash functions with limited security against non-uniform adversaries. Such a hash function $H : \{0,1\}^{3\kappa} \to \{0,1\}^\kappa$ guarantees that no superpolynomial adversary with non-uniform description of polynomial size S can find more than $K(S)$ collisions in the underlying function. Here, K is a fixed polynomial (e.g., quadratic). The resulting commitments for larger tags are secure against non-uniform adversaries.
- Kalai and Khurana [21] assume sub-exponential non-interactive witness indistinguishable (NIWI) proofs and obtain security against non-uniform adversaries. But their compiler, on input commitments that satisfy a weaker notion of non-malleability w.r.t. replacement generates commitments that are non-malleable w.r.t replacement for a larger tag space.

In [28, 4], NIWIs are combined with a hard-to-invert trapdoor statement to enable weak forms of NIZKs without setup. In contrast, [21] use NIWIs without associated trapdoors, but then only achieve weaker forms of non-malleability (that is, w.r.t. replacement).

But a common thread among the amplification techniques is that they all require the use of sub-exponential NIWI proofs. We remind that reader that NIWIs are one round proof systems with statistical soundness, for which no computationally bounded verifier can distinguish which witness in a relation was used to create the proof.

Reliance on NIWIs results in the following less than ideal consequences:

- Subexponential NIWIs are only known based on the hardness of the decisional linear problem over bilinear maps [19], or derandomization assumptions and subexponential trapdoor permutations [2].
- All these compilers use NIWIs to prove complex cryptographic statements, and therefore make non-black box use of the underlying non-malleable commitment for a smaller tag space. On the other hand, from the point of view of efficiency, it is desirable to have constructions that make black-box use of cryptography.

Our Results. In this work, we provide a new approach to non-interactive tag amplification for non-malleable commitments. This approach only makes black-box use of cryptography, and achieves provable security under a more diverse set of assumptions. Specifically, this compiler replaces the NIWI assumption with hinting PRGs, that were introduced by Koppula and Waters [25], and can be obtained based on CDH, LWE [25] and also ϕ-hiding and DBDHI assumptions [13]. (One can also alternatively execute the paradigm from any projective key-dependent secure symmetric key encryption scheme [24] which is realizable from the LPN assumption).

We summarize (a simplification of) our results via the following informal theorems. Recall that base schemes are typically only secure in a setting where the adversary is restricted to using the same tag in all its queries to the oracle. In what follows, we refer to such a commitment scheme that is only secure against this limited class of adversaries as a same-tag CCA secure commitment. We also refer to CCA commitments where the adversary is only allowed to make parallel oracle queries after obtaining the challenge commitment, as non-malleable commitments.

Theorem 1. *(Informal) (Removing the Same-Tag Restriction) Assuming the existence of sub-exponentially secure hinting PRGs and keyless hash functions that are collision-resistant against sub-exponential uniform adversaries, there exists a compiler that on input any same-tag CCA (respectively, non-malleable) non-interactive commitment for N tags secure against* non-uniform adversaries *where $N \leq \mathsf{poly}(\kappa)$, outputs a CCA (respectively, non-malleable) non-interactive commitment for N tags secure against* uniform adversaries.

Theorem 2. *(Informal) (Tag-Amplification for CCA commitments) Assuming the existence of sub-exponentially hinting PRGs and keyless hash functions that are collision-resistant against sub-exponential uniform adversaries, there exists a compiler that on input any CCA (respectively, non-malleable) non-interactive commitment for N tags secure against* non-uniform adversaries *where $N \leq \mathsf{poly}(\kappa)$, outputs a CCA (respectively, non-malleable) non-interactive commitment for $2^{N/2}$ tags secure against* uniform adversaries.

Unfortunately, using these informal theorems to ampify tag space from $c \log \log n$ for a small constant $c > 0$ immediately encounters the following issue: the input scheme to the compiler is required to be *non-uniform secure*, whereas the output scheme is only *uniform secure*.

To enable recursion, we strengthen our CCA abstraction. Specifically, we modify the CCA security game to allow an adversary to submit a Turing Machine P to the challenger, and obtain the evaluation of P on an input of the adversary's choice. We say that a scheme is e-"computation enabled" if it is secure against all adversaries that submit programs that run in time polynomial in 2^{κ^e} for constant e. As such, we will substitute the *non-uniform security* requirement for the base CCA scheme and instead require it to be e-"computation enabled" for an appropriate constant e. The output of the compiler will be an e'-"computation enabled" commitment for an appropriate constant e'. We describe this abstraction, and our techniques, in additional detail in Section 1.1.

1.1 Our Techniques

We now provide our technical overview. Recall that the core technical goal of our work is to provide a method for amplifying from a commitment scheme for $O(N)$ sized tag space to a 2^N sized space. If the computational overhead associated with the amplification step is polynomial in N and the security parameter κ, then the process can be applied iteratively $c + 1$ times to a base NM commitment scheme that handles tags of size $\lg \lg \cdots \lg(\kappa)$ for a c-times iterated log, for arbitrary constant c and results in a scheme that handles tags of size 2^κ. Here, we note that subexponential quantum hardness of LWE and subexponential hardness of DDH [21], or subexponential hardness amplifiable one-way functions [4], or subexponential variants of time-lock puzzles [28] imply base schemes for tags in $(c \lg \lg \kappa)$ for a small constant $c > 0$, which means they imply schemes for tags in $(\lg \lg \lg \kappa)$.

Now the traditional way to amplify such a tag space can be traced back to [11][2] They suggested a method of breaking a large tag T^j (say, in $[2^N]$) into N small tags $t_1^j, t_2^j, \ldots t_N^j$, each in $2N$, such that for two different large tags $T^1 \neq T^2$, there exists at least one index i such that $t_i^2 \notin \{t_1^1, t_2^1, \ldots t_N^1\}$. This is achieved by setting $t_i^j = i||T^j[i]$, where $T^j[i]$ denotes the i^{th} bit of T^j.

A scheme for tags in 2^N will have an algorithm $\mathsf{CCA.Com}$ that commits to a message m as $\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m; r) \rightarrow \mathsf{com}$. To commit to m under tag one first creates N tags $t_1, \ldots t_N$ by applying the DDN encoding to tag. Next, these (smaller) tags are used to generate commitments of m in the smaller tag scheme as $c_i = \mathsf{Small.Com}(1^\kappa, (t_i), \mathsf{msg} = m; r_i)$ for $i \in [N]$. Next, the committer attaches a zero knowledge (ZK) proof that all commitments are to the same message m using the random coins as a witness. Since we are interested in non-interactive amplification, the ZK proof will need to be non-interactive. Additionally, we will require it to be ZK against adversaries running in time T, where T is the time required to brute-force break the underlying CCA scheme for small tags.

CCA security of the scheme with larger tag space can be argued in two basic steps. Suppose the challenger commits to either m_0^* or m_1^* under tag T^* (we denote the DDN encoding of T^* by $t_1^*, \ldots t_N^*$). The adversary wins if it gets which out of m_0^* and m_1^* was committed. Recall that the adversary can request the CCA oracle to provide openings of commitment string with tags $\mathsf{tag} \neq \mathsf{tag}^* \in \{0,1\}^N$. This oracle generates a response as follows - (1) Verify the ZK proof in the commitment string. Return $\perp$ if verification does not accept. (2) Open the underlying commitment scheme with small tags at position 1 with tag t_1.

We will assume, for simplicity, that the adversary makes a single oracle query in the CCA game, with tag T, whose DDN encoding is denoted by $t_1, \ldots t_N$. We will focus on the index i in the adversary's oracle query, such that the tag $t_i \notin \{t_1^*, \ldots t_N^*\}$.

As a first step towards proving CCA security, one can modify the oracle to open the commitment string c with small tag t_i, in Step 2. Because of the soundness of the ZK proof system, this change cannot be detected by the adversary, except with negligible probability.

At this point, the challenge commitment is modified so that the ZK proof is simulated and does not need the random coins used in the small tag commitments anymore. To argue indistinguishability, we will need to answer the adversary's oracle queries. This will be done by extracting,

[2] This was recently further optimized by [23] but in this paper, we use the [11] technique for simplicity.

via brute-force, the value committed in the adversary's oracle query. As such, we will need to rely on ZK proofs where the ZK property holds even against machines that can (brute-force) break the small tag commitnents. Once this is done, we will change each of the small tag commitments in the challenge commitment from committing to the message m_b^* to committing to the all 0's string, one by one. At the same time, the oracle will continue to open the commitment string c with small tag t_i, in Step 2. Since $t_i \notin \{t_1^*, \ldots t_N^*\}$, we can rely on CCA security of the underlying small tag scheme and argue that the adversary will not be able to detect these changes. By the end, all information about the bit b will be erased.

Since non-interactive zero-knowledge proofs without setup are impossible, existing non-interactive tag amplification techniques [28, 23, 4] rely on weaker variants of zero-knowledge proofs, such as ZK with super-polynomial simulation and weak soundness, to perform tag amplification via the afore-mentioned outline. These required variants of non-interactive ZK proofs are obtained by including a trapdoor statement td. To prove that a statement x is in an NP language L, one typically provides a NIWI to establish that $(x \in L) \vee (\mathsf{td}$ is true$)$. The trapdoor statement helps perform simulation, whereas for soundness it is required that the adversary cannot prove the trapdoor statement. One exception is [21], which only relies on NIWIs and does not make use of on any trapdoor statements, but is limited to the weaker notion of replacement security. However, in addition to relying on NIWIs, the outline above makes non-black-box use of the underlying base commitment scheme.

Eliminating NIWIs. Our primary goal in this paper is to perform tag amplification without NIWIs, and while making black-box use of the underlying base commitments. Taking a step back, the reason ZK is required in the tag amplification argument discussed above, is that we can change the oracle to one that opens different underlying tags, without the adversary noticing. In other words, we would like to establish a system where the adversary cannot submit a commitment such that its opening will be different under the original and new oracle functions.

Here, inspired by recent work in chosen ciphertext secure public key encryption [25], our construction will allow the oracle to recover a PRG seed s that gives (a good part of) the randomness used to create the underlying commitments. Specifically, the oracle will use the commitment with a specific small tag to first recover a candidate PRG seed s' and then check for consistency by re-evaluating the underlying commitment pieces, and checking them against the original.

These checks will intuitively serve as a substitution for ZK proofs. Interestingly, our checking algorithm will allow some partially malformed commitments to go through – allowing this is essential to our security argument. This is in contrast to a ZK proof which enforces that all must be commitments to the same message. While creating such partially malformed commitments is actually easy for the adversary, the adversary will still not be able to differentiate between different forms of decryption. (We note that in non-malleable encryption some systems [33, 7] allow for somewhat malformed ciphertexts to be let through.) Importantly, unlike [25] that looked at two possible decryption strategies, we will need to ensure that up to polynomially many such strategies decrypt the same way. Furthermore, we will not be able rely on trusted setup to generate verification keys for a signature scheme. Instead, we will develop a new technique leveraging hinting PRGs, which we outline below.

We now describe our new tag amplification technique that converts CCA commitments with $4N$ tags to CCA commitments with 2^N tags. We point out that our technique also applies as is to converting parallel CCA commitments with $4N$ tags to parallel CCA commitments with 2^N tags. First, we summarize some of the tools we will use.

- **Hinting PRGs.** A hinting PRG, introduced in [25], satisfies the following property: for a uniformly random short seed s, the matrix M obtained by first expanding $PRG(s) = z_0 z_1 z_2 \dots z_n$, 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 uniform matrix. Hinting PRGs are known based on CDH, LWE [25] – more generally, any circular secure symmetric key encryption scheme [24].
- **Statistically Equivocal Commitments without Setup.** We will rely on statistically hiding bit commitments without setup, that satisfy binding against uniform adversaries. Additionally, these commitments will be statistically equivocal, that is, with overwhelming probability, a randomly chosen commitment string can be opened to both a 0 and a 1. These can be obtained from keyless collision resistant hash functions against uniform adversaries, based on the blueprint of [10] and [20], and more recently [3], in the keyless hash setting.

Outline of Our Tag Amplification Technique. Let ($\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover}$) be a non malleable commitment for $4N$ tags. We will assume tags take identities of the form $(i, \beta, \gamma) \in [N] \times \{0,1\} \times \{0,1\}$ and that the $\mathsf{Small.Com}$ algorithm requires randomness of length $\ell(\kappa)$.

Our transformation will produce three algorithms, ($\mathsf{CCA.Com}, \mathsf{CCA.Val}$, $\mathsf{CCA.Recover}$). The $\mathsf{CCA.Com}$ algorithm on input a tag tag from the large tag space, an input message, and uniform randomness, first samples a seed s of size n for a hinting PRG. It uses the first co-ordinate z_0 of the output of the hinting PRG on input s, as a one-time pad to mask the message m, resulting in string c. Next, it generates n equivocal commitments $\{\sigma_i\}_{i\in[n]}$, one to each bit of s. We will let y_i denote the opening of the i^{th} equivocal commitment (this includes the i^{th} bit s_i of s). Finally, it 'signals' each of the bits of s by generating commitments $\{c_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$ using the small tag scheme. For every $i \in [n]$, the commitments $\{c_{x,i,0}\}_{x\in[N]}$ and $\{c_{x,i,1}\}_{x\in[N]}$ are generated as follows:

1. If $s_i = 0$
   (a) $c_{x,i,0} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 0), \mathsf{msg} = y_i; r_{x,i})$
   (b) $c_{x,i,1} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 1), \mathsf{msg} = y_i; \tilde{r}_{x,i})$
2. If $s_i = 1$
   (a) $c_{x,i,0} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 0), \mathsf{msg} = y_i; \tilde{r}_{x,i})$
   (b) $c_{x,i,1} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 1), \mathsf{msg} = y_i; r_{x,i})$

where all the $\widetilde{r}_{x,i}$ values are uniformly random, whereas $r_{x,i}$ values correspond to the output of the hinting PRG on seed s. The output of $\mathsf{CCA.Com}$ is $\mathsf{tag}, c, \{\sigma_i\}_{i\in[n]}, \{c_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$.

On an oracle query of the form $\mathsf{CCA.Val}(\mathsf{tag}, \mathsf{com})$, we must return the message committed in the string com, if one exists. To do this, we parse $\mathsf{com} = \mathsf{tag}, c, \{\sigma_i\}_{i\in[n]}, \{c_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$, and then recover the values committed under small tags $(1, \mathsf{tag}_1, 0)$ and $(1, \mathsf{tag}_1, 1)$, which also helps recover the seed s of the hinting PRG. Next, we check that for every $i \in [n]$, the recovered values correspond to openings of the respective σ_i. We also compute hinting PRG(s), and use the resulting randomness to check that for all $x \in [N]$, the commitments that were supposed to use the outcome of the PRG were correctly constructed. If any of these checks fail, we know that the commitment string com cannot be a well-formed commitment to any message. Therefore, if any of the checks fail, the oracle outputs $\perp$. These checks are inspired by [25], and intuitively, ensure that it is computationally infeasible for an adversary to query the oracle on commitment strings that lead to different outcomes differently depending on which small tag was used. If all these checks pass, the $\mathsf{CCA.Val}$ algorithm uses c to recover and output m.

Proving Security. We will prove that the resulting scheme is CCA secure against uniform adversaries. To begin, we note that the set $\{(x, \mathsf{tag}_x)\}_{x\in[N]}$

is nothing but the DDN encoding of the tag tag. Recall that this encoding has the property that for every $\mathsf{tag}, \mathsf{tag}^* \in 2^N$, there exists an index $x \in [N]$ such that $(x, \mathsf{tag}_x) \notin \{(x^*, \mathsf{tag}^*_{x^*})\}_{x^* \in [N]}$. In the scheme described above, the tag used for each set $\{c_{x,i,b}\}_{i \in [n]}$ is (x, tag_x, b). This means that for our particular method of generating the commitments $c_{x,i,b}$ described above, for each of the adversary's oracle queries, there will be an index $x' \in [N]$ such that the tags $(x', \mathsf{tag}_{x'}, 0)$ and $(x', \mathsf{tag}_{x'}, 1)$ used to generate $\{c_{x',i,b}\}_{i \in [n], b \in \{0,1\}}$ in that query will differ from *all small tags used to generate the challenge commitment.*

Our first step towards proving security of the resulting commitment with large tags, will be to define an alternative $\mathsf{CCA.ValAlt}$ algorithm, that instead of recovering the values committed under tags $(1, \mathsf{tag}_1, 0)$ and $(1, \mathsf{tag}_1, 1)$, recovers values committed under $(x', \mathsf{tag}_{x'}, 0)$ and $(x', \mathsf{tag}_{x'}, 1)$. As already alluded to earlier, this scheme is designed so that it is computationally infeasible for a uniform adversary to query the oracle on commitment strings for which $\mathsf{CCA.Val}$ and $\mathsf{CCA.ValAlt}$ lead to different outcomes. Formally, we will first switch to a hybrid that uses the $\mathsf{CCA.ValAlt}$ algorithm instead of $\mathsf{CCA.Val}$ to answer the adversary's oracle queries.

When making this change, because of the checks performed by the valuation algorithms, we can formally argue that any adversary that distinguishes these hybrids must query the oracle with a commitment string that has following property: For some $i \in [n], x \in [N]$, $c_{x,i,0}$ and $c_{x,i,1}$ are small tag commitments to openings of the equivocal commitment to some bit b and $1 - b$ respectively. Assuming that the equivocal commitment satisfies binding against uniform adversaries that run in subexponential time, one can brute-force extract these openings from $c_{x,i,0}$ and $c_{x,i,1}$ to contradict the binding property.

The next hybrid is an exponential time hybrid that samples equivocal commitments $\{\sigma_i\}_{i \in [n]}$, for the challenge commitment, 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.

In the next hybrid, inspired by [25] we modify the components $\{c^*_{x,i,b}\}_{x \in [N], i \in [n], b \in \{0,1\}}$ in the challenge commitment to "drown" out information about s via noise. In particular, while in the real game, the values $c^*_{x,i,1}$ are always commitments to $y_{s_i,i}$, in the challenge commitment these values are modified to become commitments to $y^*_{i,1}$, irrespective of what s_i is. In the next step, the values $c^*_{x,i,0}$ are modified to become commitments to $y^*_{i,0}$, irrespective of what s_i is. We rely on CCA security of the underlying small tag scheme so that we can continue to run the $\mathsf{CCA.ValAlt}$ function to recover values committed under $(x', \mathsf{tag}_{x'}, 0)$ and $(x', \mathsf{tag}_{x'}, 1)$ while changing all the com-

ponents $\{c^*_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$ in the challenge commitment. This step crucially makes use of the fact that the tags $(x', \mathsf{tag}_{x'}, 0)$ and $(x', \mathsf{tag}_{x'}, 1)$ differ from *all small tags used to generate the challenge commitment.* Moreover, in spite of the fact that generating equivocal openings of $\{\sigma_i\}_{i\in[n]}$ 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 $\{\sigma_i\}_{i\in[n]}$ together with their openings can be fixed non-uniformly and independently of the strings $c^*_{x,i,b}$, and therefore these hybrids can be proven indistinguishable based on non-malleability of the small tag commitment against non-uniform adversaries. Since we must carefully manipulate the randomness used for $c^*_{x,i,b}$ in both games, this hybrid requires a delicate argument.

At this point, we have eliminated all information about the PRG seed s, except from the randomness $r_{x,i}$ and $\widetilde{r}_{x,i}$. In the final hybrid, we rely on the security of the hinting PRG to switch to using uniform randomness everywhere. Note that we still need to answer the adversary's oracle queries, but this can be done by ensuring that the time required to run the $\mathsf{CCA.ValAlt}$ algorithm is much smaller than that needed to break hinting PRG security. At this point, there is no information about s, and therefore about the message being committed to in the challenge commitment.

Issues with Recursion. At this point, it may seem like we are done, but the careful reader may have noticed a problem. To prove security, we assumed an input scheme that was secure against *non-uniform* adversaries, but due to the use of equivocal commitments against uniform adversaries, the transformation yields a scheme that is only secure against *uniform* adversaries. This would be no problem if we say were only amplifying once from κ to 2^κ tags. But unfortunately, the recursion will not work if our base scheme starts with $\lg\lg\lg(\kappa)$ size tags (which is the number of tags allowable by most existing base schemes), as we will need to recursively amplify multiple times.

It might seem that we are fundamentally stuck. The first hybrid in our argument requires the equivocal commitment scheme to be more secure than the underlying small tag commitment. Later hybrids require that the small tag commitment to satisfy CCA security even when equivocal commitments with openings to both ones and zeros are generated. If the small tag CCA scheme is only uniformly secure, it seems impossible to satisfy this requirement without violating the previous one.

However, if we peel the recursion back further, there appears to be a glimmer of hope. Suppose we are applying our transformation to an underlying CCA commitment, which is itself the result of applying the transformation one or more times. When our proof arrives at the security of the underlying scheme, the underlying scheme's security will rely both on an equivocal commitment itself, and at the deepest level the non-uniform security of the base scheme. If the equivocal commitments in the underlying scheme use a larger security parameter than the current one, then the lower level scheme may still be secure (and lower level equivocal commitments may still be binding) even when equivocal openings are found at the current level.

e-Computation Enabled Security. We capture this intuition by expanding our abstraction to include what we call e-computation enabled CCA commitments. Here, we modify the security game to allow an adversary to submit a Turing Machine P to the challenger. The adversary will receive the evaluation of P on an input of its choice. We say that a scheme is e-computation enabled if it is secure against all adversaries that submit programs that run in time polynomial in 2^{κ^e} for constant e. (The program output size itself is required to be polynomially bounded.)

With this abstraction in place, when proving security, our reduction can pass the task of generating equivocal openings as an appropriate program P to the enhanced CCA security game itself. Implicitly, this allows the equivocal opening requests to be satisfied in different ways depending on what stage the security proof of the lower scheme is at.

While this new property provides a useful tool for recursion, we also need to work a bit harder to prove e-computation enabled CCA security. Specifically, we prove in Section 3 that given a hinting PRG and an equivocal commitment scheme that are uniformly secure against 2^{κ^δ} time adversaries for $\delta \in (0, 1)$, we can transform an e-computation enabled CCA scheme for small tags into one that is e'-computation enabled CCA secure for large tags, where $e' = e \cdot \delta$.

In our proof, at the stages where we use a reduction to find equivocal openings, the reduction will run in time $2^{\kappa^{e'}}$ to satisfy the adversary's program request. When contradicting the hinting PRG, the reduction will run in time 2^{κ^e} to to find equivocal openings, and $2^{\kappa^{e'}}$ to satisfy the adversary's program request. To ensure that this gives us a contradiction, we will set the security parameter of the hinting PRG to be large enough. Finally, when the reduction is to the underlying small tag CCA commitment, the program request of the large tag adversary will be passed by

the reduction to the interface of the underlying small tag scheme, which is allowed since $e' < e$. In the base case, we note that we start with schemes secure against non-uniform adversaries (for $\lg \lg \lg \kappa$ tags). By definition, any scheme that is secure against non-uniform adversaries is trivially e-computation enabled secure for arbitrary e.

Issues due to Same-Tag Restrictions. The techniques described above capture our main ideas for tag amplification. Unfortunately, the base schemes that we start with may only be same-tag secure. On the other hand, we would like to end up with CCA schemes for 2^κ tags that do not have this restriction. This is because CCA commitments without such a restriction can be generically transformed, assuming signatures into schemes that do not use tags at all. We remedy the same-tag issue by applying a transformation that takes a scheme supporting a tag space of N tags with same-tag only queries to one that supports N tags without the same-tag restriction, for any $N \leq \mathsf{poly}(\kappa)$.

Removing the Same-Tag Requirement. We start with an underlying scheme that has the same-tag requirement, and modify it to remove this requirement as follows. To commit to a message with tag tag in the new scheme, commit to it with respect to all $N - 1$ tags *except* tag in the underlying same-tag scheme. Similar to the previous construction, we use hinting PRGs and attach a bunch of checks to ensure that recovering the committed value from the adversary's queries using any one tag is computationally indistinguishable from recovering it using a different tag.

The overall mechanics and guarantees are similar to our prior transformation. Suppose an adversary were given a challenge commitment tag^* in the transformed scheme, and got to make queries to several *different* tags $\mathsf{tag} \neq \mathsf{tag}^*$. By our construction, the adversary's challenge does not contain an underlying commitment with tag tag^* whereas all of the adversary's oracle queries will contain an underlying commitment with tag tag^*. We can therefore answer all of these queries by changing the oracle valuation function to one that uses only tag tag^* in underlying scheme.

We note that since the same-tag transformation incurs a blowup proportional to N, it is imperative to apply it early on in the sequence of transformations. If we first amplified the tag space to be of size 2^κ and then attempted to remove the same-tag restriction, the resulting scheme would have exponential sized commitments. Therefore, we start with a base scheme that is same-tag secure and supports tags of size iterated log, c times, as $\lg \lg \cdots \lg(\kappa)$ for some constant c, we will first apply the same-tag to many-tag transformation. Next, we apply the tag amplification

transformation $c+1$ times. We end up with a scheme that is polynomial sized and supports a tag space of size 2^κ with no same-tag restrictions.

Non-uniform Security. Our techniques give a CCA commitment scheme secure against uniform adversaries. One might ask whether we could use similar techniques, perhaps combined with new assumptions such as non-uniformly secure keyless hash functions [3, 4] to obtain security against non-uniform adversaries. We address this in two parts.

First, taking a step back, a primary motivation for obtaining non-uniform security is that it is useful for protocol composition. For example, if we were using a cryptographic primitive like public key encryption as an end application say for encrypting email, then obtaining uniform security would arguably be just fine. As the uniform model captures attackers in the real world. However, the extra power of non-uniform security might be helpful if our commitment scheme were a component used in building a larger cryptosystem. Here, we observe that our transformation actually outputs a CCA scheme with properties that are stronger than (plain) uniform security. Specifically, the output scheme satisfies e-computation enabled CCA security.

While the initial motivation for this abstraction was that it helps with recursion; we note that it can actually be a useful property for a CCA scheme to have. In particular, it can actually be viewed as a more fine-grained or nuanced view of non-uniform computation. This abstraction gives any adversary non-uniform advice so long as it can be computed in time 2^{κ^e}. If e is set appropriately, then we expect this would suffice in many circumstances, including for protocol composition. Indeed, this was true for the type of protocol composition that we needed to recursively amplify the tag space. Thus our amplification techniques and our abstraction can arguably deliver something that is the "best of both worlds": the outcome is as good as non-uniform security for many applications, but does not make any new non-uniform assumptions about the hash function.

Second, our techniques are also meaningful for constructing black-box two-message non-malleable commitments with (regular) non-uniform security. In our transformation, the primitive that requires uniform security is the keyless hash-based equivocal commitment scheme. In the two-message setting, it seems possible to slightly modify our scheme to have the receiver generate the key for a keyed (non-uniform secure) collision-resistant hash function. All of our other techniques appear to carry over to this setting, and it appears that one would be able to prove that the resulting scheme is a (regular) non-uniform secure non-malleable commitment that only makes black-box use of cryptography.

Organization We define "computation enabled" commitments in Section 2, present our tag amplification scheme in Section 3), and show how to compile these elements in Section 4. Details on preliminaries and proof analyses, as well as recovery-from-randomness and removing the same tag restriction can be found in our full version [12].

2 Computation Enabled CCA Commitments

We now define what we describe as "computation enabled" CCA secure commitments. Intuitively, these will be 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 will be statistically binding if 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 [6] 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 $\mathsf{CCA.Val}$ where $\mathsf{CCA.Val}(\mathsf{com})$ will return m if $\mathsf{CCA.Com}(\mathsf{tag}, m; r) \rightarrow \mathsf{com}$ for some r. The attacker is allowed oracle access to $\mathsf{CCA.Val}(\cdot)$ for any $\mathsf{tag} \neq \mathsf{tag}^*$. The traditional notion of non-malleability (as seen in [21], etc.) is simply a restriction of the CCA game where the adversary is only allowed to simultaneously submit a single set of decommitment queries. The proof of this is immediate and can be found in [5].

The primary difference in our definition is that we also allow the attacker to submit a randomized turing machine P at the beginning of the game. The challenger will run P and output its result to the attacker at the beginning of the game. This added property will allow us to successfully apply recursion for tag amplification later in our scheme. In addition, we require a recover from randomness property, which allows one to open the commitment given all the randomness used to generate said commitment.

2.1 Definition

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

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

$\mathsf{CCA.Recover}(\mathsf{com}, r) \rightarrow m$ 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.

Definition 1 (Correctness). *We say that our 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*

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

Definition 2 (Efficiency). *We say that our computation enabled CCA secure commitment scheme is efficient if* $\mathsf{CCA.Com}, \mathsf{CCA.Recover}$ *run in time* $\mathsf{poly}(|m|, \kappa)$*, while* $\mathsf{CCA.Val}$ *runs in time* $\mathsf{poly}(|m|, 2^\kappa)$.

Definition 3 (Security). *We say that our 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 1. We remark that this is implied by Definition 1, 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

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

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.[3]

[3] 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.

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 sends

$$\mathsf{CCA.Val}(\mathsf{com}).$$

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 sends $\perp$. Otherwise it responds as

$$\mathsf{CCA.Val}(\mathsf{com}).$$

7. The attacker finally outputs a guess b'.

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 4. *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 will always terminate in time $p(2^{\kappa^e})$ time and output at most $q(\kappa)$ bits for some polynomial functions p, q (For all possible random tapes given to the program P).*

Definition 5. *A computation enabled CCA secure commitment scheme scheme given by algorithms* $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ *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* $\mathsf{negl}(\kappa)$.

We also define another notion of security which we call "same tag" computation enabled secure for a weaker class of adversaries who only submit challenge queries that all have the same tag.

Definition 6. *A computation enabled CCA secure commitment scheme scheme given by algorithms* $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ *is said to be "same tag" e-computation enabled CCA secure if for any e-conforming adversary $\mathcal{A}$ which generates queries such that all commitment queries submitted by $\mathcal{A}$ are on the same tag, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the attacker's advantage in the game is* $\mathsf{negl}(\kappa)$.

Recovery From Randomness

Definition 7. *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 *we have that*

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

Claim. Let $(\mathsf{CCA.Com}, \mathsf{CCA.Val})$ be a set of algorithms which satisfy any of Definition 1, Definition 2, Definition 3, Definition 5. 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 7. We defer the construction and proof to our full version [12].

2.2 Connecting to Standard Security

We now connect our computation enabled definition 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 shows 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 8. *A commitment scheme* $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ *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* $\mathsf{negl}(\kappa)$.

Definition 9. *A commitment scheme* $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ *is said to be "same tag" CCA secure against uniform/non-uniform attackers if for any poly-time uniform/non-uniform adversary* $\mathcal{A}$ *such that all commitment queries submitted by* $\mathcal{A}$ *are on the same tag, 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. If $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ is an e-computation enabled CCA secure commitment scheme for some e as per Definition 5, then it is also a scheme that achieves standard CCA security against uniform poly-time attackers as per Definition 8.

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 $\mathcal{A}$. □

Claim. If $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ achieves standard CCA security against *non-uniform* poly-time attackers as per Definition 8, then it is an e-computation enabled CCA secure commitment scheme for any e as per Definition 5.

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. □

We remark that the above statements are also true for "same tag" conforming adversaries.

3 Tag Amplification

In this section we show a process from amplifying a computation enabled CCA commitment scheme for $N' = 4N$ tags to a scheme with 2^N tags. The amplification process imposes an overhead that is polynomial in N and the size/time of the original commitment scheme. Thus it is important that N be polynomially bounded in the security parameter.

Let $(\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover})$ be an e-computation enabled CCA commitment scheme for $N'(\kappa) = N' = 4N$ tags. We will assume tags take identities of the form $(i, \beta, \Gamma) \in [N] \times \{0,1\} \times \{0,1\}$ and that the $\mathsf{Small.Com}$ algorithm take in random coins of length $\ell(\kappa)$. In addition, for some constant $\delta \in (0,1)$[4] we assume a equivocal commitment without setup scheme $(\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ that is $T = 2^{\kappa^\delta}$ binding secure and statistically hiding.

[4] The constant δ must be less than 1 in order to meet the requirement that the $\mathsf{Equiv.Equivocate}$ algorithm runs in time polynomial in 2^κ.

We assume a hinting PRG scheme $(\mathsf{Setup}, \mathsf{Eval})$ that is $T = 2^{\kappa^\gamma}$ secure for some constant $\gamma \in (0,1)$ and has seed length $n(\kappa, |m|)$ (represented by n for ease) and block output length of $\max(|m|, \ell \times N)$. For ease of notation we assume that $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0) \in \{0,1\}^{|m|}$ and $\forall i \in [n]$, $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i) \in \{0,1\}^{\ell \cdot N}$.

Our transformation will produce three algorithms, $\mathsf{CCA.Com}$, $\mathsf{CCA.Val}$, and $\mathsf{CCA.Recover}$ which we prove e'-computation enabled where we require $e' = e \cdot \delta \geq 1$. We will also present a fourth algorithm $\mathsf{CCA.ValAlt}$, which is only used in the proof. The algorithms will make use of the auxiliary subroutines $\mathsf{CCA.Find}$ and $\mathsf{CCA.Check}$ described below. $\mathsf{CCA.ValAlt}(\mathsf{tag}^*, \mathsf{com}) \rightarrow m \cup \perp$ is a deterministic inefficient algorithm that takes in a tag tag^* and a commitment com and outputs either a message $m \in \{0,1\}^*$ or a reject symbol $\perp$. It will be used solely as an instrument in proving the scheme secure and not exported as part of the interface. We describe the transformation and due to space constraints analyze it's properties formally in [12] . We present the security games in the main body to give intuition on how our proof proceeds.

$\mathsf{CCA.Find}(x', \mathsf{com})$

Inputs: Index $x' \in [N]$

Commitment $\mathsf{com} = \left(\mathsf{tag}, \mathsf{HPRG.pp}, c, (\sigma_i, (c_{x,i,0}, c_{x,i,1})_{x \in [N]})_{i \in [n]}\right)$

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

- For each $i \in [n]$
  1. Let $\tilde{y}_i = \mathsf{Small.Val}(c_{x',i,0})$
  2. Set $\tilde{z}_i = \mathsf{Equiv.Decom}(\sigma_i, \tilde{y}_i)$. If $\tilde{z}_i = \perp$, set $\tilde{s}_i = 1$. Else, set $\tilde{s}_i = \tilde{z}_i$.
- Output $\tilde{s} = \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n$.

Fig. 1. Routine $\mathsf{CCA.Find}$

Transformation $\mathsf{Amplify}(\mathsf{Small} = (\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover}), \mathsf{HPRG}, \mathsf{Equiv}, e') \rightarrow \mathsf{NM} = (\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$:

$\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m \in \{0,1\}^*; r) \rightarrow \mathsf{com}$

1. Compute $\kappa' = \kappa^{\frac{e'}{\delta}} = \kappa^e$. Compute $\kappa'' = \kappa'^{\frac{1}{\gamma}}$.[5]
2. Sample $(\mathsf{HPRG.pp}, 1^n) \leftarrow \mathsf{HPRG.Setup}(\kappa'', 1^{\max(|m|, N \cdot \ell)})$.
3. Sample $s = s_1 \ldots s_n \xleftarrow{R} \{0,1\}^n$ as the seed of the hinting PRG.

[5] δ and γ are known from the security guarantees of $\mathsf{Equiv}, \mathsf{HPRG}$ respectively.

$\mathsf{CCA.Check}(\tilde{s}, \mathsf{com})$

Inputs: Seed candidate $\tilde{s} = \tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n$

Commitment $\mathsf{com} = \left(\mathsf{tag}, \mathsf{HPRG.pp}, c, (\sigma_i, (c_{x,i,0}, c_{x,i,1})_{x\in[N]})_{i\in[n]}\right)$

Output: $\{0,1\}$

- For $i \in [n]$
  1. Compute $(r_{1,i}, r_{2,i}, \ldots, r_{N,i}) = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, \tilde{s}, i)$
  2. For $x \in [N]$
     (a) Let $\tilde{y}_i = \mathsf{Small.Recover}(c_{x,i,\tilde{s}_i}, r_{x,i})$. If $\tilde{y}_i = \perp$, output 0
     (b) If $c_{x,i,\tilde{s}_i} \neq \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, \tilde{s}_i), \tilde{y}_i; r_{x,i})$, output 0.
     (c) If $\tilde{s}_i \neq \mathsf{Equiv.Decom}(\sigma_i, \tilde{y}_i)$, output 0.
- If all the above checks have passed, output 1.

Fig. 2. Routine CCA.Check

4. For all $i \in [n]$ run $\mathsf{Equiv.Com}(1^{\kappa'}, s_i) \to (\sigma_i, y_i)$.
5. Let $r_{x,i}, \tilde{r}_{x,i} \in \{0,1\}^\ell$ be defined as follows:
6. For $i \in [n]$
   (a) Compute $(r_{1,i}, r_{2,i}, \ldots, r_{N,i}) = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, i)$
   (b) Sample $(\tilde{r}_{1,i}, \tilde{r}_{2,i}, \ldots, \tilde{r}_{N,i}) \xleftarrow{R} \{0,1\}^{N\cdot\ell}$
7. Compute $c = m \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0)$
8. For $i \in [n]$, $x \in [N]$
   (a) If $s_i = 0$
      i. $c_{x,i,0} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 0), \mathsf{msg} = y_i; r_{x,i})$
      ii. $c_{x,i,1} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 1), \mathsf{msg} = y_i; \tilde{r}_{x,i})$
   (b) If $s_i = 1$
      i. $c_{x,i,0} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 0), \mathsf{msg} = y_i; \tilde{r}_{x,i})$
      ii. $c_{x,i,1} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 1), \mathsf{msg} = y_i; r_{x,i})$
9. Output $\mathsf{com} = \left(\mathsf{tag}, \mathsf{HPRG.pp}, c, (\sigma_i, (c_{x,i,0}, c_{x,i,1})_{x\in[N]})_{i\in[n]}\right)$ as the commitment. All of the randomness is used as the decommitment string.

$\mathsf{CCA.Val}(\mathsf{com}) \to m \cup \perp$

1. Set $\tilde{s} = \mathsf{CCA.Find}(1, \mathsf{com})$.
2. If $\mathsf{CCA.Check}(\tilde{s}, \mathsf{com}) = 0$ output $\perp$.
3. Output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, \tilde{s}, 0)$.

$\mathsf{CCA.ValAlt}(\mathsf{tag}^*, \mathsf{com}) \to m \cup \perp$

1. If $\mathsf{com.tag} = \mathsf{tag}^*$, output $\perp$.
2. Let x^* be the smallest index where the bits of $\mathsf{tag}^*, \mathsf{tag}$ differ.
3. Set $\tilde{s} = \mathsf{CCA.Find}(x^*, \mathsf{com})$.
4. If $\mathsf{CCA.Check}(\tilde{s}, \mathsf{com}) = 0$ output $\perp$.

5. Output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, \tilde{s}, 0)$.

$\mathsf{CCA.Recover}(\mathsf{com}, r) \rightarrow m \cup \perp$

1. From r, parse the seed s of the Hinting PRG.
2. From com, parse the commitment component c and the public parameter $\mathsf{HPRG.pp}$.
3. Output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0)$

3.1 Proof of Security

We now prove security by showing that our transformation leads to an $e' = e \cdot \delta$-computation enabled CCA commitment scheme. We do so in a sequence of security games.

In each proof step we will need to keep in mind that the attacker will be allowed to ask for a program P that runs in time polynomial in $2^{\kappa^{e'}}$ where $e' = e \cdot \delta$. This will be satisfied in one of two ways. In the proof steps that rely on the hinting PRG security or the equivocal commitment without setup scheme we leverage the that that these are subexponentially secure primitives. For relying on security of the equivocal commitment without setup we use security parameter $\kappa' = \kappa^e$, it is secure against attackers that run in time polynomial in $2^{(\kappa')^\delta} = 2^{\kappa^{e\delta}} = 2^{\kappa^{e'}}$ time. Thus our reduction algorithm in these steps can satisfy the requirement by running P itself and still be a legitimate $2^{(\kappa')^\delta}$ time attacker. For relying on security of the hinting PRG scheme, we use security parameter $\kappa'' = \kappa'^{\frac{1}{\gamma}}$, it is secure against attackers that run in time polynomial in $2^{\kappa'}$. Thus our reduction algorithm can run P and the equivocate algorithm.

The second situation is when we rely on the security of the smaller tag space e-computation enabled scheme. In this case the reduction will need to be polynomial time so there is no way for it to directly run a program P that takes $2^{\kappa^{e'}}$ time. However, in this case it can satisfy the requirement by creating a program $\tilde{P}$ and passing this onto the security game of the e-computation enabled challenger. The program $\tilde{P}$ will run P as well as n invocations of the $\mathsf{Equiv.Equivocate}$ algorithm. We present our sequence of games below, and proofs of indistinguishability between these games can be found in the Supplementary material.

Game 0. This is the original message hiding game between a challenger and an attacker for $e' = e \cdot \delta$ conforming attackers. 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.

2. The attacker sends a "challenge tag" $\mathsf{tag}^* \in \{0,1\}^N$.
3. **Pre Challenge Phase:** The attacker makes repeated queries commitments

$$\mathsf{com} = \left(\mathsf{tag}, \mathsf{HPRG.pp}, c, (\sigma_i, (c_{x,i,0}, c_{x,i,1})_{x\in[N]})_{i\in[n]}\right).$$

If $\mathsf{tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise responds as

$$\mathsf{CCA.Val}(\mathsf{com}).$$

4. **Challenge Phase**
   (a) The attacker sends two messages $m_0^*, m_1^* \in \{0,1\}^w$
   (b) **Part 1:**
   - Compute $\kappa' = \kappa^e$.
   - Compute $\kappa'' = \kappa'^{\frac{1}{\gamma}}$.
   - Sample $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(\kappa'', 1^{\max(w, N\cdot\ell)})$.
   - Sample $s^* = s_1^* \dots s_n^* \xleftarrow{R} \{0,1\}^n$ as the seed of the HPRG.
   - Let $r_{x,i}^*, \tilde{r}_{x,i}^* \in \{0,1\}^\ell$ be defined as follows:
   - For $i \in [n]$
     i. Compute $(r_{1,i}^*, r_{2,i}^*, \dots, r_{N,i}^*) = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}^*, s^*, i)$
     ii. Sample $(\tilde{r}_{1,i}^*, \tilde{r}_{2,i}^*, \dots, \tilde{r}_{N,i}^*) \xleftarrow{R} \{0,1\}^{N\cdot\ell}$
   - For all $i \in [n]$ run $\mathsf{Equiv.Com}(1^{\kappa'}, s_i^*) \to (\sigma_i^*, y_i^*)$.

   (c) **Part 2:**
   - It chooses $b \in \{0,1\}$ and sets $c^* = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}^*, s^*, 0) \oplus m_b^*$.
   - For $i \in [n]$, $x \in [N]$
     i. If $s_i^* = 0$
       A. $c_{x,i,0}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 0), y_i^*; r_{x,i}^*)$
       B. $c_{x,i,1}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 1), y_i^*; \tilde{r}_{x,i}^*)$
     ii. If $s_i^* = 1$
       A. $c_{x,i,0}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 0), y_i^*; \tilde{r}_{x,i}^*)$
       B. $c_{x,i,1}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 1), y_i^*; r_{x,i}^*)$
   - Finally, it sends $\mathsf{com}^* = \left(\mathsf{tag}^*, \mathsf{HPRG.pp}^*, c^*, (\sigma_i^*, \left(c_{x,i,0}^*, c_{x,i,1}^*\right)_{x\in[N]})_{i\in[n]})\right)$ as the commitment. All of the randomness is used as the decommitment string.

5. **Post Challenge Phase:** The attacker again makes commitment queries com. If $\mathsf{tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise it responds as

$$\mathsf{CCA.Val}(\mathsf{com}).$$

6. The attacker finally outputs a guess b'.

Game 1. This is same as Game 0, except that during the **Pre Challenge Phase** and **Post Challenge Phase**, challenger uses $\mathsf{CCA.ValAlt}(\mathsf{tag}^*, \mathsf{com})$ to answer queries.

Game 2. In this game in **Part 1** the (σ_i^*, y_i^*) are now generated from the $\mathsf{Equiv.Equivocate}$ algorithm instead of the $\mathsf{Equiv.Com}$ algorithm.

- Compute $\kappa' = \kappa^e$, $\kappa'' = \kappa'^{\frac{1}{\gamma}}$.
- Sample $(\mathsf{HPRG.pp}^*, 1^n) \leftarrow \mathsf{HPRG.Setup}(\kappa'', 1^{\max(w, N\cdot\ell)})$.
- Sample $s^* = s_1^* \dots s_n^* \xleftarrow{R} \{0,1\}^n$ as the seed of the hinting PRG.
- Let $r_{x,i}^*, \tilde{r}_{x,i}^* \in \{0,1\}^\ell$ be defined as follows, for $i \in [n]$
  1. Compute $(r_{1,i}^*, r_{2,i}^*, \dots, r_{N,i}^*) = \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}^*, s^*, i)$
  2. Sample $(\tilde{r}_{1,i}^*, \tilde{r}_{2,i}^*, \dots, \tilde{r}_{N,i}^*) \xleftarrow{R} \{0,1\}^{N\cdot\ell}$
- For all $i \in [n]$ run $\mathsf{Equiv.Equivocate}(1^{\kappa'}) \to (\sigma_i^*, y_{i,0}^*, y_{i,1}^*)$.
- For all $i \in [n]$, set $y_i^* = y_{i,s_i^*}^*$.

Game 3 In this game in **Part 2** we move to $c_{x,i,0}^*$ committing to $y_{i,0}^*$ and $c_{x,i,1}^*$ committing to $y_{i,1}^*$ for all $x \in [N], i \in [n]$ independently of s_i^*.

- For $i \in [n]$, $x \in [N]$
  1. If $s_i^* = 0$
     (a) $c_{x,i,0}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 0), y_{i,0}^*; r_{x,i}^*)$
     (b) $c_{x,i,1}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 1), y_{i,1}^*; \tilde{r}_{x,i}^*)$
  2. If $s_i^* = 1$
     (a) $c_{x,i,0}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 0), y_{i,0}^*; \tilde{r}_{x,i}^*)$
     (b) $c_{x,i,1}^* = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x^*, 1), y_{i,1}^*; r_{x,i}^*)$

Game 4. In all $r_{x,i}^*$ values are chosen uniformly at random (insted of choosing from $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}^*, s^*, i)$) and c^* is also chosen uniformly at random (instead of choosing $\mathsf{HPRG.Eval}(\mathsf{HPRG.pp}^*, s^*, 0) \oplus m_b^*$).

4 Compiling our Transformations

We conclude by showing how to compile our transformations. Suppose that we begin with a base scheme supporting $32 \cdot \mathsf{ilog}(c, \kappa)^6$ tags for some constant c that is secure against non-uniform attackers that make same tag queries. We will compile this into a scheme supporting $16 \cdot 2^\kappa$ space against uniform attackers with no same tag restriction.

[6] For brevity, $\mathsf{ilog}(c, \kappa)$ denotes $\underbrace{\lg \lg \cdots \lg}_{c \text{ times}}(\kappa)$.

We apply the transformation that removes the same tag restriction [12] to the base scheme which divides the tag space supported by 2 to get a scheme with $16 \cdot \mathsf{ilog}(c, \kappa)$ sized tag space, but removes the same-tag restriction. The we apply the Section 3 tag amplification process $c + 1$ times. Recall the transformation takes a $N' = 4N$ scheme to a scheme supporting 2^N tags. Since $16/4 = 4$ and $2^4 = 16$ the effect is of each application is to remove one of the lg iterations and keep the factor of 16. Since the transformation imposes a polynomial blowup on the underlying scheme and since it is applied a constant number of times, the size of the resulting scheme is also polynomial.

Below we give a formal construction utilizing the transformations $\mathsf{RecoverRandom}(\cdot)$ presented in [12] , $\mathsf{OneToMany}(\cdot)$ presented in [12] , and $\mathsf{Amplify}(\cdot)$ presented in Section 3. Since we are transforming a scheme that takes $32 \cdot \mathsf{ilog}(c, \kappa)$ tags to $16 \cdot 2^\kappa$ tags, we need to use the amplification transformation $c+1$ times. $\mathsf{OneToMany}(\cdot), \mathsf{Amplify}(\cdot)$ transformations take in a e-computation enabled scheme and output a $e' = e \cdot \delta$-computation enabled scheme where $e' \geq 1$ and $\delta \in (0, 1)$ and the equivocal commitment scheme is 2^{κ^δ} hiding secure. We set $\mathsf{OneToMany}(\cdot)$ to take a $e \cdot \delta^{-c-2}$-computation enabled and output a $e \cdot \delta^{-c-1}$-computation enabled scheme. $\mathsf{Amplify}(\cdot)$ takes a $e \cdot \delta^{-c-1}$-computation enabled scheme and outputs a e-computation enabled scheme after $c + 1$ tranformations.

$\mathsf{CompiledAmplify}(\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val}), \mathsf{HPRG}, \mathsf{Equiv}, e)$

1. $\mathsf{RandomBaseCCA} \leftarrow \mathsf{RecoverRandom}(\mathsf{BaseCCA})$
2. Let δ be the constant so that Equiv is 2^{κ^δ} binding secure and c be the constant such that the base scheme takes $32 \cdot \mathsf{ilog}(c, \kappa)$.
3. $\mathsf{AmplifiedCCA}^0 \leftarrow \mathsf{OneToMany}(\mathsf{RandomBaseCCA}, \mathsf{HPRG}, \mathsf{Equiv}, e \cdot \delta^{-c-1})$.
4. For $i \in [c + 1]$
   (a) $\mathsf{AmplifiedCCA}^i \leftarrow \mathsf{Amplify}(\mathsf{AmplifiedCCA}^{i-1}, \mathsf{HPRG}, \mathsf{Equiv}, e \cdot \delta^{i-c-1})$
5. Output $(\mathsf{AmplifiedCCA}^{c+1}.\mathsf{Com}, \mathsf{AmplifiedCCA}^{c+1}.\mathsf{Val})$

Below we analyze $\mathsf{CompiledAmplify}$ by stating theorems on correctness, efficiency and security. Due to space constraints, we defer the proofs of these theorems to the supplementary section.

Theorem 3. *For every $\kappa \in \mathbb{N}$, let $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val})$ be a perfectly correct CCA commitment scheme by Definition 1. Let $\mathsf{Equiv} = (\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ be a perfectly correct equivocal commitment scheme. Then, we have that the scheme $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{Equiv}, e)$ is a perfectly correct CCA commitment scheme.*

Theorem 4. *For every $\kappa \in \mathbb{N}$, let* $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val})$ *be an efficient CCA commitment scheme by Definition 2 with tag space* $32 \cdot \mathsf{ilog}(c, \kappa)$. *Let* $\mathsf{Equiv} = (\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ *be an efficient equivocal commitment scheme. Then,* $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{Equiv}, e)$ *is an efficient CCA commitment scheme.*

Theorem 5. *For every $\kappa \in \mathbb{N}$, let* $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val})$ *be a CCA commitment scheme that is hiding against non-uniform "same tag" adversaries according to Definition 9 for tag space* $32 \cdot \mathsf{ilog}(c, \kappa)$. $\mathsf{HPRG} = (\mathsf{HPRG.Setup}, \mathsf{HPRG.Eval})$ *be a hinting PRG scheme that is $T = 2^{\kappa^\gamma}$ secure for $\gamma \in (0, 1)$.* Equiv *be an equivocal commitment without setup scheme that is $T = 2^{\kappa^\delta}$ binding and statistically hiding for some constant $\delta \in (0, 1)$. Then,* $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{Equiv}, e)$ *is a e-computation enabled CCA commitment scheme that is hiding against uniform adversaries according to Definition 8 for tag space* $16 \cdot 2^\kappa$.

We import the following theorems about instantiating base schemes, from prior work.

Theorem 6. *[21] For every constant $c > 0$, there exist CCA secure commitments satisfying Definition 9 against non-uniform adversaries, with tag space $(c \lg \lg \lg \kappa)$, that make black-box use of subexponential quantum hard one-way functions and subexponential classically hard one-way functions in* BQP.

We point out that while [21] prove that their construction satisfies non-malleability with respect to commitment, their proof technique also directly exhibits same-tag CCA security against non-uniform adversaries.

Combining this theorem with Theorem 5 yields the following corollary.

Corollary 1. *There exists a constant $e > 0$ for which there exists a perfectly correct and polynomially efficient e-computation enabled CCA secure commitment satisfying Definition 5 against uniform adversaries, with tag space 2^κ for security parameter κ, that makes black-box use of subexponential quantum hard one-way functions, subexponential classically hard one-way functions in* BQP*, subexponential hinting PRGs and subexponential keyless collision-resistant hash functions.*

Alternatively, [28] showed that for every constant $c > 0$, assuming a family of $(c \lg \lg \lg \kappa)$ time-lock puzzles that are simultaneously increasingly depth-robust and decreasingly time-robust, there exist CCA secure commitments satisfying Definition 9 against non-uniform adversaries, with

tag space ($c \lg \lg \lg \kappa$). Our compiler applies to their base scheme as well, yielding e-computation enabled CCA secure commitment satisfying Definition 5 against uniform adversaries, with tag space 2^κ, that make black-box use of the LPS base scheme.

Finally, we point out that while all our formal theorems discuss CCA security, our transformations also apply as is to the case of amplifying parallel CCA security (equivalently, concurrent non-malleability w.r.t. commitment). That is, given a base scheme that is only same-tag parallel CCA secure (or non-malleable w.r.t. commitment) for small tags, our transformations yield a scheme for all tags that is parallel CCA secure (or concurrent non-malleable w.r.t. commitment) for tags in 2^κ, without the same tag restriction.

References

1. Barak, B.: Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In: FOCS (2002)
2. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. (2007)
3. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: STOC (2018)
4. Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: Theory of Cryptography (2018)
5. Broadnax, B., Fetzer, V., Müller-Quade, J., Rupp, A.: Non-malleability vs. cca-security: the case of commitments. In: IACR International Workshop on Public Key Cryptography (2018)
6. Canetti, R., Lin, H., Pass, R.: Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions. In: FOCS (2010)
7. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Simple, Black-Box Constructions of Adaptively Secure Protocols. In: TCC (2009)
8. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Concurrent non-malleable commitments (and more) in 3 rounds. In: CRYPTO (2016)
9. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Four-round concurrent non-malleable commitments from one-way functions. In: CRYPTO (2017)
10. Damgård, I.B., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In: CRYPTO (1993)
11. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography (Extended Abstract). In: STOC (1991)
12. Garg, R., Khurana, D., Lu, G., Waters, B.: Black-box non-interactive non-malleable commitments (2020), `https://eprint.iacr.org/2020/1197`
13. Goyal, R., Vusirikala, S., Waters, B.: New constructions of hinting prgs, owfs with encryption, and more. IACR Cryptology ePrint Archive (2019)
14. Goyal, V.: Constant Round Non-malleable Protocols Using One-way Functions. In: STOC (2011)
15. Goyal, V., Lee, C.K., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: FOCS (2012)

16. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: STOC (2016)
17. Goyal, V., Richelson, S.: Non-malleable commitments using goldreich-levin list decoding. In: FOCS (2019)
18. Goyal, V., Richelson, S., Rosen, A., Vald, M.: An algebraic approach to non-malleability. In: FOCS (2014)
19. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM (2012)
20. Halevi, S., Micali, S.: Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In: CRYPTO (1996)
21. Kalai, Y.T., Khurana, D.: Non-interactive non-malleability from quantum supremacy. In: CRYPTO (2019)
22. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: TCC (2017)
23. Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: FOCS (2017)
24. Kitagawa, F., Matsuda, T., Tanaka, K.: CCA security and trapdoor functions via key-dependent-message security. In: CRYPTO (2019)
25. Koppula, V., Waters, B.: Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In: CRYPTO (2019)
26. Lin, H., Pass, R.: Non-malleability Amplification. In: STOC (2009)
27. Lin, H., Pass, R.: Constant-round Non-malleable Commitments from Any One-way Function. In: STOC (2011)
28. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: FOCS (2017)
29. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent Non-malleable Commitments from Any One-Way Function. In: TCC (2008)
30. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive One-Way Functions and Applications. In: CRYPTO (2008)
31. Pass, R., Rosen, A.: Concurrent Non-Malleable Commitments. In: FOCS (2005)
32. Pass, R., Rosen, A.: New and Improved Constructions of Nonmalleable Cryptographic Protocols. SIAM J. Comput. (2008)
33. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a non-malleable encryption scheme from any semantically secure one. In: CRYPTO (2006)
34. Pass, R., Wee, H.: Constant-round non-malleable commitments from sub-exponential one-way functions. In: EUROCRYPT (2010)
35. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS (2010)