On Non-uniform Security for Black-box Non-Interactive CCA Commitments

Rachit Garg[1], Dakshita Khurana[2], George Lu[1], and Brent Waters[1,3]

[1] University of Texas at Austin, Austin, TX, USA
[2] University of Illinois Urbana-Champaign, Champaign, Illinois, USA
[3] NTT Research, Sunnyvale, CA, USA

Abstract. We obtain a black-box construction of non-interactive CCA commitments against non-uniform adversaries. This makes black-box use of an appropriate base commitment scheme for small tag spaces, variants of sub-exponential hinting PRG (Koppula and Waters, Crypto 2019) and variants of keyless sub-exponentially collision-resistant hash function with security against non-uniform adversaries (Bitansky, Kalai and Paneth, STOC 2018 and Bitansky and Lin, TCC 2018).

All prior works on non-interactive non-malleable or CCA commitments without setup first construct a "base" scheme for a relatively small identity/tag space, and then build a tag amplification compiler to obtain commitments for an exponential-sized space of identities. Prior black-box constructions either add multiple rounds of interaction (Goyal, Lee, Ostrovsky and Visconti, FOCS 2012) or only achieve security against uniform adversaries (Garg, Khurana, Lu and Waters, Eurocrypt 2021).

Our key technical contribution is a novel tag amplification compiler for CCA commitments that replaces the non-interactive proof of consistency required in prior work. Our construction satisfies the strongest known definition of non-malleability, i.e., CCA2 (chosen commitment attack) security. In addition to only making black-box use of the base scheme, our construction replaces sub-exponential NIWIs with sub-exponential hinting PRGs, which can be obtained based on assumptions such as (sub-exponential) CDH or LWE.

1 Introduction

Non-malleable commitments [18] and their stronger counterparts CCA commitments [12] are core cryptographic primitives that provide security in the presence of "man in the middle" attacks. They ensure that a man-in-the-middle adversary, that simultaneously participates in two or more protocol sessions, cannot use information obtained in one session to breach security in another. They also enable secure multi-party computation, coin flipping and auctions.

This work builds non-interactive CCA commitments, which involve just a single commit message from the committer. We focus on the (standard) notion of

security against non-uniform adversaries, which necessitates that these commitments be perfectly binding and computationally hiding. For these commitments, the perfect binding requirement is that for any commitment string c generated maliciously with potentially an arbitrary amount of preprocessing, there do not exist two openings to messages m and m' such that $m \neq m'$. The (computational) hiding property requires that for every pair of equal-length messages m and m', the distributions of commitments $\mathsf{com}(m)$ and $\mathsf{com}(m')$ are computationally indistinguishable.

The notion of CCA security for commitments is defined analogously to encryption schemes, except that the adversary is given access to a decommitment oracle. However, unlike the case of encryption, non-interactive commitments without setup do not allow for efficient decommitment given a trapdoor/secret key. In more detail, the hiding game is strengthened significantly to give the adversary oracle access to an *inefficient* decommitment/value function $\mathsf{CCA.Val}$ where on input a string c, $\mathsf{CCA.Val}(\mathsf{tag}, c)$ will return m if $\mathsf{CCA.Com}(\mathsf{tag}, m; r) \rightarrow c$ for some r. The adversary must first specify a challenge tag tag^*, along with messages m_0^*, m_1^*. It 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. [4]

This CCA-based definition is the strongest known definition of non-malleability. In the non-interactive setting, the often-used definition of (concurrent) non-malleability with respect to commitment is a special case of this definition where the adversay is only allowed to make parallel oracle queries once it obtains the challenge commitment.

Prior Work on Non-Malleable Commitments. There have been several results [18, 4, 40, 41, 37, 38, 34, 44, 42, 35, 21, 22, 25, 23, 14, 15, 30, 36, 32, 24] that gradually reduced the round complexity and the cryptographic assumptions required to achieve non-malleable commitments. In the non-interactive setting, Pandey, Pass and Vaikuntanathan [38] first obtained non-malleable commitments from a strong non-falsifiable assumption. A lower bound due to Pass [39] demonstrated the difficulty of obtaining a non-interactive construction from standard assumptions.

Nevertheless, recent works of Lin, Pass and Soni [36], Bitansky and Lin [8], Kalai and Khurana [29], Garg et al. [19] and Khurana [31] made progress towards improving these assumptions. These works proceed in two steps: the first step builds a "base" scheme supporting a small (typically, constant-sized) tag space and the second step converts commitments supporting a small tag space to commitments that support a much larger tag space.

Base Constructions. Three recent works [36, 8, 29] build non-interactive base schemes: non-malleable commitments for a tag space of size $c \log\log \kappa$ for a spe-

[4] The assumption that the commitment takes input a tag is without loss of generality when the tag space is exponential. As is standard with non-malleable commitments, tags can be generically removed by setting the tag as the verification key of a signature scheme, and signing the commitment string using the signing key.

cific constant $c > 0$, based on various hardness assumptions. Specifically, Lin, Pass and Soni [36] assume a sub-exponential variant of the hardness of time-lock puzzles, and Bitansky and Lin [8] rely on sub-exponentially hard injective one-way functions that admit hardness amplification beyond negligible. Finally, Kalai and Khurana [29] assume classically sub-exponentially hard but quantum easy non-interactive commitments (which can be based, e.g., on sub-exponential hardness of DDH), and sub-exponentially quantum hard non-interactive commitments (which can be based, e.g., on sub-exponential hardness of LWE).

Tag Amplification. The second step, as discussed above, builds a tag amplficiation compiler that increases the tag space exponentially. 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), multiple applications of this compiler yield commitments for a tag space of size 2^κ.

This step, which is also the focus of the current work, typically involves encoding a single tag from a larger space into many tags from a smaller space, and then committing to a given message several times, once w.r.t. each small tag. In addition, an implicit/explict *proof of consistency* of these commitments is provided, and this proof is required to hide the committed message. Such a proof becomes challenging to implement in the non-interactive setting without setup.

Nevertheless, tag amplification was obtained in [36] against *uniform* man-in-the-middle adversaries based on sub-exponential non-interactive witness indistinguishable (NIWI) proofs and keyless collision resistant hash functions against uniform adversaries. It was also obtained in [8] against *non-uniform* man-in-the-middle adversaries based on sub-exponential non-interactive witness indistinguishable (NIWI) proofs and keyless collision resistant hash functions with a form of collision resistance even against *non-uniform* adversaries. Somewhat orthogonally, [31] obtained tag amplification from sub-exponential indistinguishability obfuscation and sub-exponential one-way functions, while avoiding the need for keyless collision resistant hashing.

Black-box Tag Amplification. Recently, [19] developed the first tag amplification technique that only made *black-box use of the base commitment*. That work additionally assumed (black-box access to) hinting PRGs and keyless collision resistant hash functions against uniform adversaries. Hinting PRGs themselves admit constructions from the CDH and LWE assumptions. Besides being black-box , this was the first solution that *did not* rely on non-interactive witness indistinguishable (NIWI) proofs, which so far are only known based on the hardness of the decisional linear problem over bilinear maps [26], or derandomization assumptions and trapdoor permutations [5], or indistinguishability obfuscation and one-way functions [9]. However, GKLW only obtain security against uniform adversaries.

But non-uniform security is often necessary when using non-malleable commitments within a bigger protocol. For instance, round efficient secure multi-party computation protocols in the plain model [10, 1, 27, 6, 2, 13] against

malicious adversaries usually include a step where participants commit to their inputs via a non-malleable/CCA commitment, in addition to providing a proof that the CCA commitment is consistent with other messages sent in the protocol. In low-interaction settings such as those of super-polynomial secure MPC in two or three [3] messages, these proofs of consistency are often simulated non-uniformly, which ends up necessitating the use of non-malleable commitments with security against non-uniform adversaries.

Our work addresses the following natural gap in our understanding of non-interactive non-malleable/CCA commitments.

Is it possible to obtain black-box non-interactive CCA commitments against non-uniform adversaries?

Our Results. This work provides a *black-box approach* to achieving non-interactive CCA commitments with security against *non-uniform adversaries*, by relying on keyless hash functions that satisfy collision-resistance against non-uniform adversaries, and by overcoming seemingly fundamental limitations from the prior work of [19]. In addition, our tag amplification technique achieves provable security without the need for NIWIs as in prior work [8], and by instead relying on a sub-exponentially secure variant of hinting PRGs, which can themselves be obtained from (sub-exponential) CDH or LWE just like their counterparts in [33].

2 Overview of Techniques

We now give an overview of our amplification technique, where the goal is to amplify a scheme for $O(N)$ tags to a scheme for 2^N tags, with computational cost that grows polynomially with N and the security parameter κ. This process can be applied iteratively $c+1$ times to a base NM commitment scheme that handles tags of size $\underbrace{\lg \lg \cdots \lg}_{c \text{ times}}(\kappa)$ for some constant c and results in a scheme that handles tags of size 2^κ.

Templates for Tag Amplification. To perform tag amplification, we will build on a tag encoding scheme that was first suggested by [18]. They suggest 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.

Given this tag amplification technique, we start by describing a template for non-interactive tag amplification suggested in [32, 36]. A CCA commitment scheme for tags in 2^N will generate a commitment to a message m as $\mathsf{CCA.Com}(1^\kappa, \mathsf{tag}, m; r) \rightarrow \mathsf{com}$. The string com is generated by first applying the DDN encoding to tag to obtain N tags $t_1, \ldots t_N$. Next, these (smaller) tags are used to generate commitments to m in the smaller tag scheme as $c_i = \mathsf{Small.Com}(1^\kappa, (t_i), \mathsf{msg} = m; r_i)$ for $i \in [N]$. The intuition for security

is as follows: recall that the DDN encoding ensures 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 (roughly) implies that the commitment generated by an adversary w.r.t. tag t_i^2 is *independent* of the challenge commitment string, as we desire. However, the commitments w.r.t. other tags t_j^2 could potentially depend on the challenge commitment, which is undesirable. To get around this issue, the templates in [32, 36][5] suggest that the committer attach a type of zero knowledge (ZK) proof that all commitments are to the same message m using the random coins as a witness. In the setting of non-interactive amplification, the ZK proof will need to be non-interactive. For technical reasons, it is in fact required 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.

Since non-interactive ZK proofs do not exist without trusted setup, the techniques in [36, 32, 8, 29] rely on weaker variants of ZK such as NIWIs, and [36, 32, 8] combine NIWIs with a trapdoor statement that an (inefficient) ZK simulator uses to simulate the ZK proof. At the same time, for soundness, we require that an adversary cannot use the trapdoor statement to cheat. This is challenging when the trapdoor statement is fixed independently of the statement being proven, because a *non-uniform* adversary can always hardwire the trapdoor and use this to provide convincing proofs of false statements.

Given this barrier, [36] restricted themselves to achieving tag amplification against *uniform adversaries*, based on (sub-exponential) NIWIs and keyless collision-resistant hash functions against uniform adversaries. Subsequently [8] developed a technique to obtain tag amplification against *non-uniform adversaries*, based on NIWIs and assuming the existence of keyless collision-resistant hash functions that satisfy some form of security against non-uniform adversaries. Very roughly, they assume that no adversary with non-uniform advice of size S can find more than $\mathsf{poly}(S)$ collisions[6].

More recently, [19] developed a method for performing non-interactive tag amplification without NIWIs, and while only making *black-box use* of the underlying base commitment. However, the resulting scheme is secure only against *uniform adversaries*. On the other hand, the goal of this work is to achieve a *black-box construction* that avoids NIWIs and achieves security against *non-uniform adversaries*, under a similar keyless assumption as [8]. To highlight the bottlenecks in the non-uniform setting, we give a brief overview of the technique of [19].

Black-box Tag Amplification. To begin, we note that the tag amplification technique sketched above is not black-box in the base commitment due to the use of variants of ZK. Recall that ZK is used to ensure consistency of adversarial

[5] These are the non-interactive versions of templates previously suggested in [18, 34, 44].

[6] Technically, they rely on a more general notion of incompressible problems, which is a collection of efficiently recognizable and sufficiently dense sets, one for each security parameter, for which no adversary with non-uniform description of polynomial size in S can find more than $K(S)$ elements in the set.

commitments generated w.r.t. different small tags. In the CCA setting, this allows using a CCA decommitment oracle that opens a commitment under any one of the adversary's small tags, without the adversary noticing which one was opened. In other words, ZK is used to establish a system where the adversary cannot submit a commitment such that its opening will be different under oracle functions that open different commitments, which turns out to be crucial to achieving CCA security.

In [19], this system is established by means of a *hinting PRG* [33]. At a high level, the construction in [19] sets things up so that the CCA oracle that opens a commitment under one of the adversary's small tags will recover a candidate PRG seed s. This seed deterministically generates (a significant part of) the randomness used to create commitments with respect to *all* the adversary's small tags. The oracle uses this property to check for consistency by re-evaluating the underlying small-tag commitments, and checking them against the original. These checks intuitively serve as a substitute for ZK proofs, however they differ from ZK in that the checking algorithm sometimes allows partially malformed commitments to be opened to valid values. While creating such partially malformed commitments is actually easy for the adversary, the adversary is still unable to distinguish between oracles that open different small tag commitments.

The work [19] converts CCA commitments with $4N$ tags to CCA commitments with 2^N tags, assuming hinting PRGs and statistically equivocal commitments without setup, that satisfy binding against uniform adversaries. A hinting PRG satisfies the following property: for a uniformly random short seed s, expand $PRG(s) = z_0 z_1 z_2 \dots z_n$. Then compute matrix x by sampling uniformly random $v_1 v_2 \dots v_n$, and setting for all $i \in [n]$, $M_{s_i,i} = z_i$ and $M_{1-s_i,i} = v_i$. The requirement is that z_0, M generated using a uniformly random seed must be indistinguishable from a uniform random string.

Here, we actually note that prior works [33, 19] can be made to work based on a hinting PRG that actually satisfies a weaker property: namely, that z_0, M obtained as described above should be indistinguishable from u, M where u is generated uniformly at random and M is generated as described above. Looking ahead, we will define a variant of a hinting PRG and will rely on the fact that this weaker property can be used instead.

Hinting PRGs were built based on CDH, LWE [33], as well as more efficient versions based on the ϕ-hiding and DBDHI assumptions [20]. The required equivocal commitments can be obtained from keyless collision resistant hash functions against uniform adversaries, based on the blueprint of [17] and [28], and more recently [7], in the keyless hash setting.

The [19] technique. We now provide a brief overview of the [19] technique, since their construction will serve as a starting point for our work.

Let $(\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover})$ be a CCA commitment for $4N$ tags. Then [19] 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 $\ell(\kappa)$. Their transformation produces 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 in-

put 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 $\tilde{r}_{x,i}$ 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 [33], and intuitively, ensure that it is computationally infeasible for an adversary to query the oracle on commitment strings that lead to different outcomes depending on which small tag was used. If all these checks pass, the $\mathsf{CCA.Val}$ algorithm uses c to recover and output m.

To prove that the resulting scheme is CCA secure against uniform adversaries, note that the set $\{(x, \mathsf{tag}_x)\}_{x\in[N]}$ is nothing but the DDN encoding of the tag tag. 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.*

The first step towards proving security of the resulting commitment 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 $(x', \mathsf{tag}_{x'}, 1)$. The goal is to ensure that it is computationally infeasible for an adversary to query the oracle on commitment strings for which

$\mathsf{CCA.Val}$ and $\mathsf{CCA.ValAlt}$ lead to different outcomes. In more detail, because of the checks performed by the valuation algorithms, it is possible to argue that any adversary that distinguishes $\mathsf{CCA.Val}$ from $\mathsf{CCA.ValAlt}$ 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. One can then brute-force extract these openings from $c_{x,i,0}$ and $c_{x,i,1}$ to contradict the binding property of the commitment against *uniform* sub-exponential adversaries.

This first step already becomes a bottleneck in the non-uniform setting: in general, an adversary with bounded polynomial advice can always sample an equivocal (non-interactive) commitment string together with an opening to 0 and another opening to 1.

The problem in the non-uniform case. As discussed above, the proof/construction in [19] falls apart in the very first step when considering a non-uniform adversary. In fact, such an adversary can attack the [19] scheme by non-uniformly sampling equivocal commitments $\{\widetilde{\sigma}_i\}_{i\in[n]}$ together with randomness $\{\widetilde{y}_{0,i}\}_{i\in[n]}$ and $\{\widetilde{y}_{1,i}\}_{i\in[n]}$ that can be used to open these commitments to both 0 and 1 respectively. Next, it can set the components $\{\widetilde{c}_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$ as small-tag commitments to both types of openings. This allows the attacker to explicitly break CCA2 security, as we describe next.

Let $x' \in [N]$ be an index 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 differ from *all small tags used to generate the challenge commitment.* On one hand, CCA2 security of the small-tag scheme will ensure that seed recovered from small-tag commitments $(x', \widetilde{\mathsf{tag}}_{x'}, 0)$ and $(x', \widetilde{\mathsf{tag}}_{x'}, 1)$ are independent of the seed in the challenge commitment. On the other hand, the actual committed value, which is defined via the seed recovered from $(1, \widetilde{\mathsf{tag}}_1, 0), (1, \widetilde{\mathsf{tag}}_1, 1)$ will exactly match the value in the challenge commitment, allowing this adversary to break CCA2 security. The equivocation described above would allow the adversary to ensure that all the hinting PRG checks pass, despite the use of different types of seeds in small tags $(1, \widetilde{\mathsf{tag}}_1, 0), (1, \widetilde{\mathsf{tag}}_1, 1)$ versus $(x', \widetilde{\mathsf{tag}}_{x'}, 0), (x', \widetilde{\mathsf{tag}}_{x'}, 1)$.

Towards a Solution. Now, one could hope to rely on some form of *non-uniform* security of keyless hash functions [7, 8]. Prior works [7, 8] have formulated and used the assumption that there exist keyless hash functions where any adversary with non-uniform advice of size S can only find $\mathsf{poly}(S)$ collisions. Inspired by a technique in [8], we could hope to define a "bad" CCA2 query as one that contains openings to both a zero and a one for the equivocal commitment. Next, we could hope to limit the number of "bad" CCA2 queries that a non-uniform adversary will make to its decommitment oracle. As long as this set of "bad" queries is bounded and is just a function of the adversary's non-uniform advice, our challenger could also hope to non-uniformly obtain answers to such queries and use these instead of running the $\mathsf{CCA.Val}$ or $\mathsf{CCA.ValAlt}$ function.

Unfortunately, in the [19] protocol, even given just bounded (polynomial) non-uniform advice, an adversary will be able to equivocate *all of its* commit-

ments and generate an *unbounded* number of bad queries. Moreover, because the hinting PRG is not injective, each bad query could have multiple possible openings to different seeds. This indicates that the [19] protocol needs to be fundamentally modified to enable security against non-uniform attacks.

Our Approach. We begin by understanding how the [19] protocol can possibly be modified to disallow the attack described above.

- As described above, we want to force the adversary to "use up" bits of non-uniform advice for each new bad query that it makes. This will hopefully help limit the number of unique bad queries, and our reduction could then non-uniformly obtain answers to each of these queries.
- To allow the reduction to non-uniformly answer bad queries, we will aim to pair every possible bad query with a *unique* seed value that can be used to answer this bad query in place of running the $\mathsf{CCA.Val}$ or $\mathsf{CCA.ValAlt}$ function.

Limiting bad seeds instead of bad queries. The first bullet aims to *limit the number of bad queries*. While we will not be able to achieve this, we will achieve a slightly weaker property that will nevertheless suffice for our proof idea to go through. In more detail, we will tie every CCA2 query, and in particular the *equivocal commitment part of every CCA2 query* to an auxiliary input parameter. That is, in addition to message and randomness, each equivocal commitment will obtain as input an auxiliary parameter. There will be no hiding requirement on the auxiliary parameter; it will only serve to strengthen the binding property of the equivocal commitment. We will require that there exists a fixed polynomial $K(\cdot)$ such that any adversary with non-uniform advice of size S is unable to output $K(S)$ different pairs of auxiliary parameters and commitment strings, with valid openings for each pair to both a zero and a one. We will rely on keyless collision-resistant hash functions against non-uniform adversaries to build modified equivocal commitments with this guarantee. While this does not limit the number of bad queries that an adversary can make, it does limit the number of unique auxiliary input parameters that an adversary can use to generate CCA2 queries where it is able to open the equivocal commitments to both a zero and a one.

The goal of the second bullet is to allow a reduction to answer all bad queries by pairing every such query with a *unique seed* that can be used to non-uniformly answer this query in place of running the $\mathsf{CCA.Val}$ or $\mathsf{CCA.ValAlt}$ function. To get this idea to work, we must assign a "right" candidate seed to each bad query. As discussed above, in the [19] protocol, any adversary that can find two openings for the equivocal commitments could submit a bad query where multiple possible seed values match the output of the HPRG. To prevent this, we will explicitly force the HPRG to be injective. In more detail, we add what we call an"injective extension" to the HPRG. This is an additional algorithm $\mathsf{ExtEval}(s) \to \mathsf{r}_{\mathsf{ext}}$ that is an injective function on the HPRG seed s. The HPRG security requirement is also slightly modified to ensure that an adversary will not be able to distinguish

the PRG output z from uniform given the hint matrix M (described above) and *additionally given* $\mathsf{r}_{\mathsf{ext}}$.

Now the CCA2 commitment will additionally consist of the value $\mathsf{r}_{\mathsf{ext}} = \mathsf{ExtEval}(s)$, and $\mathsf{CCA.Val}/\mathsf{CCA.ValAlt}$ will reject if for a recovered candidate seed s', $\mathsf{ExtEval}(s') \neq \mathsf{r}_{\mathsf{ext}}$. As a result, there will be at most a single seed s that will be "compatible" with any commitment string.

Going back to the construction of our CCA2 commitment, we will compute the modified equivocal commitments with auxiliary parameter set to $\mathsf{r}_{\mathsf{ext}}$, where recall that $\mathsf{r}_{\mathsf{ext}} = \mathsf{ExtEval}(s)$. At this point, we will be able to assign (at most) one unique 's' to each auxiliary parameter. Moreover, by the (strengthened) binding property of equivocal commitments, any non-uniform attacker will be able to equivocate on at most a small number of auxiliary parameter values.

Analyzing Security. To prove CCA2 security of the resulting construction, we will proceed as follows. In the first hybrid (Game 1), we will switch to a challenger that depending on the adversary's non-uniform advice, stores a "cheat-sheet" consisting of all 'bad' $\mathsf{r}_{\mathsf{ext}}$ that the adversary can query on (with more than a certain inverse-polynomial probability), together with their inverses s under the injective algorithm $\mathsf{ExtEval}(\cdot)$. Our challenger will (1) rely on the cheat-sheet to answer any adversarial queries for which $\mathsf{r}_{\mathsf{ext}}$ lies on the cheat-sheet, and (2) use $\mathsf{CCA.Val}$ to decrypt only those queries for which $\mathsf{r}_{\mathsf{ext}}$ lies outside the cheat-sheet.

In the second hybrid (Game 2), the challenger will behave similarly as the previous hybrid, except using $\mathsf{CCA.ValAlt}$ to decrypt queries for which $\mathsf{r}_{\mathsf{ext}}$ lies outside the cheat-sheet. By the strong binding property of the equivocal commitment, the adversary is guaranteed to not equivocate on these queries (except with low probability). Therefore by the argument outlined in the proof of the [19] technique, the outputs of $\mathsf{CCA.Val}$ and $\mathsf{CCA.ValAlt}$ will be indistinguishable on these queries. The rest of the proof will follow similarly to [19]. There is one major hurdle in realizing this outline, as we discuss next.

Modifying the $\mathsf{CCA.Val}$ *algorithm.* The first hybrid (Game 1) described above will actually *not* be indistinguishable from the output of the actual CCA2 game. This is because a non-uniform adversary may generate equivocation queries for which $\mathsf{r}_{\mathsf{ext}}$ lies on the cheat-sheet and has an inverse (a hinting PRG seed), but the $\mathsf{CCA.Val}$ algorithm run by the CCA2 challenger may *not* be able to find this seed. To deal with this issue, we will change the $\mathsf{CCA.Val}$ algorithm so that it performs a brute-force search through all possible seeds to find the one (if any) that matches $\mathsf{r}_{\mathsf{ext}}$.

At first it appears that the rest of the proof should be easy once this is done. It should be possible to rely on security of the (1) auxiliary-input equivocal commitments and (2) hinting PRGs with injective extension, to show that the (updated) CCA2 game is indistinguishable from the first hybrid. However, while this is true, proving it turns out to be fairly tricky. To prove indistinguishability, we must design an efficient reduction B that has oracle access to an adversary A which distinguishes between the CCA2 game and the first hybrid. This reduction B should be able to use such an adversary to break security of

equivocal commitments, by generating many more equivocal openings than its (non-uniform) advice would allow it to. The adversary A is a CCA2 adversary, which means it makes multiple (a-priori unbounded) calls to a $\mathsf{CCA.Val}$ oracle, and B must find a way to answer these queries. But recall that the oracle needs to perform a brute-force search through all possible seeds to find the one (if any) that matches $\mathsf{r}_{\mathsf{ext}}$ – simulating this process will make B inefficient. As such, B will need to maintain its own cheat-sheet to answer $\mathsf{CCA.Val}$ queries. Even with such a cheat-sheet, the proof is not straightforward: the set of most common equivocal queries in the CCA2 game may in general be different from the set of most common queries when B answers from its cheat-sheet.

Intermediate Cheat-Sheets. To make the proof go through, we will rely on a sequence of carefully defined intermediate cheat-sheets (that we will call lists from this point on). These will be defined inductively, and in the base case $\mathcal{L}^{(0)}$ will be empty. Let $Q = Q(\kappa)$ denote the total number of oracle calls that the attacker makes. For $j \in [1, Q]$, the j^{th} intermediate list, denoted by $\mathcal{L}^{(j)}$ will contain the $\mathsf{r}_{\mathsf{ext}}$ values and corresponding seeds for A's most common equivocal queries in its first j oracle calls. Note that this does not suffice to fully define $\mathcal{L}^{(j)}$, since we also need to determine how the first $j-1$ oracle calls of A will be answered: in the definition of $\mathcal{L}^{(j)}$, the first j oracle calls will be answered using the $\mathsf{CCA.ValAlt}$ algorithm with access to the list $\mathcal{L}^{(j-1)}$. The final list $\mathcal{L}$ used by $\mathsf{CCA.ValAlt}$ in Game 1 will correspond exactly to $\mathcal{L} = \mathcal{L}^{(Q)}$. We show the following inductively for every j: when the first $j-1$ $\mathsf{CCA.Val}$ queries are answered using list $\mathcal{L}^{(j-1)}$, then it is possible to add new common equivocal queries and update the list to $\mathcal{L}^{(j)}$. This will eventually allow us to switch to the first hybrid described above, which uses $\mathsf{CCA.ValAlt}$ (plus the final list $\mathcal{L}^{(Q)}$).

We point the reader to our full version for a more detailed overview of this part of the proof. There we also discuss why for technical reasons, we require as building blocks for our equivocal commitment, keyless hash functions with specific parameters. In more detail, we require that an adversary with $S(\kappa)$ bits of advice cannot produce more than $S(\kappa) \cdot p(\kappa)$ pairs of "distinct collisions" for some a-priori fixed polynomial $p(\cdot)$, where "distinct collisions" means that no entry in any pair of collisions matches an entry in another pair. The assumption is described formally and analyzed in Section 4.1.

Completing the Analysis. After switching to $\mathsf{CCA.ValAlt}$ (plus the cheat-sheet), the next hybrid will sample 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. Next, inspired by [33] the components $\{c^*_{x,i,b}\}_{x\in[N],i\in[n],b\in\{0,1\}}$ are modified in the challenge commitment to "drown" out information about s via noise, while relying on CCA2 security of the underlying small tag scheme to run the $\mathsf{CCA.ValAlt}$ function and recover values committed under $(x', \mathsf{tag}_{x'}, 0)$ and $(x', \mathsf{tag}_{x'}, 1)$. 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.* Finally, we

rely on the security of the hinting PRG to switch to using uniform randomness everywhere.

Hinting PRGs with Injective Extension. We now describe how to achieve hinting PRGs with injective extension by modifying the constructions in [33]. Recall that we require hinting PRGs with injective extensions that satisfy a different security property than prior work: namely, for a uniformly random short seed s, expand $PRG(s) = z_0 z_1 z_2 \dots z_n$ and compute the injective output $\mathsf{r}_{\mathsf{ext}}$. Then compute matrix M by sampling uniformly random $v_1 v_2 \dots v_n$, and setting for all $i \in [n]$, $M_{s_i,i} = z_i$ and $M_{1-s_i,i} = v_i$. The requirement is that z_0 generated using a uniformly random seed must be indistinguishable from uniform, *even given M and given the output $\mathsf{r}_{\mathsf{ext}}$ of the injective extension.*

We build hinting PRGs with an injective extension by modularly combining the constructions in [33] with any leakage-resilient injective one-way function (LRIOWF). To enable this, we note that hinting PRG constructions in [33] from CDH and LWE have a "lossy" property, where PRG parameters can be generated in lossy mode in such a way that the output of the hinting PRG is simulatable given just a small amount of advice. We call the resulting abstraction a lossy hinting function. To achieve injectivity, we rely on a leakage resilient injective one-way function (LRIOWF) applied to the seed s of the lossy hinting function[7]. Finally, we generate the 'mask' z_0 of the hinting PRG as the Goldreich-Levin hardcore bits of the LRIOWF. To prove that z_0 is pseudorandom even in the presence of $\mathsf{r}_{\mathsf{ext}}$ and M, we will switch the lossy hinting function to lossy mode. In this mode the hinting function will only leak a few bits about the inverse s of the LRIOWF. We will then invoke the Goldreich-Levin theorem to argue that distinguishing the mask from uniform will require inverting the LRIOWF given just a few bits of leakage on s, which is impossible by assumption on the LRIOWF. This completes an overview of our techniques.

Comparison with Prior Work. We conclude with a comparison of our techniques against prior work that relies on keyless collision-resistant hash functions against non-uniform adversaries. While [7] relies on this assumption to obtain 3-message zero-knowledge via substantially different techniques, [8] applies this to a setting that is much closer to our work, that is, to achieving non-interactive non-malleable commitments. In more detail, [8] use keyless hash functions against non-uniform adversaries to build a special type of 1-message zero-knowledge for NP with a *weak* soundness guarantee against non-uniform provers. They achieve this by building on the usual template for 1-message ZK, where a prover proves (via a NIWI) that either $x \in L$ or that the prover knows a trapdoor. The trapdoor, roughly, corresponds to a collision in a keyless hash function; and is derived as a function of the statement x. This ensures that a prover that can (non-uniformly) find a fixed set of non-uniform collisions will only be able to provide convincing proofs for a fixed set of statements. In their construction of

[7] For example, any sub-exponentially secure injective one-way function will suffice for our purposes.

non-malleable commitments, the use of NIWIs to prove a statement of the form "$x \in L$ or the prover knows a trapdoor" results in non-black-box use of the underlying base scheme.

Unlike [8], we do not construct any variant of non-interactive ZK (or rely on assumptions like NIWI that imply non-interactive ZK). We develop a new template to directly achieve tag amplification for non-malleable commitments against non-uniform adversaries, without reliance on NIWIs. Our methodology to "tie" together the set of collisions an adversary can find with the number of commitments that an adversary can cheat on is entirely different from that of [8].

3 Background

3.1 Non-uniform Security

We say that a cryptographic game is $\mathsf{T}(\cdot)$-non-uniform secure if for any Turing Machine in $\mathsf{poly}(\mathsf{T}(\kappa))$ time with $\mathsf{poly}(\kappa)$ non-uniform advice only has only negligible advantage in said game. We will refer to $\mathsf{poly}(\cdot)$-non-uniform secure schemes as achieving 'plain' non-uniform security.

In addition, we will say a cryptographic scheme is subexponentially secure against non-uniform adversaries if there exists some constant $c > 0$ such that the scheme is 2^{n^c}-non-uniform secure. When the constant c is explicitly required, we will say c-subexponentially secure.

3.2 CCA Commitments

We present our definition of CCA secure commitments [12], which is derived from [19] with modifications made for defining security against non-uniform attackers. 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 will be statistically binding, i.e., for all $\mathsf{tag}_0, \mathsf{tag}_1$, r_0, r_1 and $m_0 \neq m_1$ we have that $\mathsf{CCA.Com}(\mathsf{tag}_0, m_0; r_0) \neq \mathsf{CCA.Com}(\mathsf{tag}_1, m_1; r_1)$.

The hiding property is a strengthened CCA2-style definition where an attacker outputs a challenge tag tag^* along with messages m_0, m_1 and receives a challenge commitment com^* to either m_0 or m_1. The attacker's job is to guess the message that was committed to with 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 $\mathsf{tag} \neq \mathsf{tag}^*$. In the non-interactive setting, the traditional notion of non-malleability (as seen in [8, 29], 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 [11].

We mention two distinct features of our definition.First, we explicitly denote the running time of the $\mathsf{CCA.Val}$ algorithm despite the fact that it is not polynomial time. Explicitly specifying the runtime of the $\mathsf{CCA.Val}$ oracle will help us in complexity leveraging when performing tag amplification. We will call the

commitment scheme to be 2^{κ^v}-efficient, i.e. can run in time (polynomially in) 2^{κ^v} where $v \geq 1$ and the security of the scheme is considered for subexponential adversaries. This additional specification was not required in [19].

Second, (as in [19]) we require a recover from randomness property, which allows one to open the commitment given all the randomness used to generate said commitment. This can be achieved generically with no additional assumptions.

Remark 3.1. We note that by considering non-uniform attackers our definition actually becomes simpler than that of [19] where they considered security against a stronger than uniform adversary, which they labeled as e-computationally enabled security. Such an adversary can run any Turing Program that runs in time $\mathsf{poly}(2^{\kappa^e})$ and obtain it's output as a non-uniform advice. This notion helped them perform complexity leveraging and obtain a uniformly secure non-malleable commitment scheme. Since we consider security against non-uniform adversaries, which are allowed to obtain non-uniform advice that may take an arbitrary amount of time to compute, our presentation is simpler.

Definition A CCA secure commitment is parameterized by a tag space of size $N = N(\kappa)$ where tags are in $[1, N]$ for message space $\mathcal{M} = \{0,1\}^{w(\kappa)}$ where $w(\cdot)$ is a polynomial function (for simplicity in notation we often skip the dependence on κ). 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 $m \in \{0,1\}^w$ 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 $m \in \{0,1\}^w$ 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 perfect binding and message hiding.

Correctness

Definition 3.2. *We say that our CCA secure commitment scheme is perfectly correct if the following holds.* $\forall m \in \{0,1\}^w$, $\mathsf{tag} \in [N]$ *and* r *we have that*

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

Efficiency

Definition 3.3. *We say that our CCA secure commitment scheme is* $\mathsf{T}(\cdot)$*-efficient, if* $\mathsf{CCA.Com}, \mathsf{CCA.Recover}$ *run in time* $\mathsf{poly}(|m|, \kappa)$*, while* $\mathsf{CCA.Val}$ *runs in time* $\mathsf{poly}(|m|, \mathsf{T}(\kappa)))$.[8]

Security

Binding.

Definition 3.4. *We say that our CCA secure commitment is perfectly binding if* $\forall c, \forall m_0, m_1 \in \{0,1\}^w$ *s.t.* $m_0 \neq m_1$ *and* $\mathsf{CCA.Val}(c) \in \{m_1, \perp\}$*, there does not exist* r *such that*

$$\mathsf{CCA.Recover}(c, r) = m_0$$

Moreover, for any c *such that* $\mathsf{CCA.Val}(c) = m_1 \neq \perp$*, then there exists* r *such that* $\mathsf{CCA.Recover}(c, r) = m_1$.

Weak Binding.

Definition 3.5. *We say that our CCA secure commitment is perfectly binding if* $\forall c, \forall m_0, m_1 \in \{0,1\}^w$ *s.t.* $m_0 \neq m_1$ *and* $\mathsf{CCA.Val}(c) \in \{m_1, \perp\}$*, there does not exist* r *such that*

$$\mathsf{CCA.Recover}(c, r) = m_0$$

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

1. The attacker sends a "challenge tag" $\mathsf{tag}^* \in [N]$.
2. The attacker makes a polynomial number of repeated commitment queries com. If $\mathsf{com.tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise it responds as
$$\mathsf{CCA.Val}(\mathsf{com}).$$
3. The attacker sends two messages $m_0, m_1 \in \{0,1\}^w$.
4. 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.
5. The attacker again makes a polynomial number of repeated queries of commitment com. If $\mathsf{com.tag} = \mathsf{tag}^*$ the challenger responds with $\perp$. Otherwise it responds as
$$\mathsf{CCA.Val}(\mathsf{com}).$$
6. The attacker finally outputs a guess b'.

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.

[8] In order for the scheme to be secure, the runtime of the $\mathsf{CCA.Val}$ oracle should be bigger than the runtime of the subexponential adversary. We will imagine runtime of the $\mathsf{CCA.Val}$ oracle to be 2^{κ^v} where $v > 1$.

Definition 3.6. *A CCA secure commitment scheme scheme given by algorithms* (CCA.Com, CCA.Val, CCA.Recover) *is said to be* $\mathsf{T}(\cdot)$*-CCA secure if for any* $\mathsf{T}(\cdot)$*-non-uniform 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 3.7. *A CCA secure commitment scheme scheme given by algorithms* (CCA.Com, CCA.Val, CCA.Recover) *is said to be "same tag"* $\mathsf{T}(\cdot)$*-CCA secure if for any* $\mathsf{T}(\cdot)$*-non-uniform 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 3.8. *We say that our CCA secure commitment scheme can be recovered from randomness if the following holds. For all* $m \in \{0,1\}^w$, $\mathsf{tag} \in [N]$, *and* r *we have that*

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

4 Setupless Equivocal Commitments against Non-Uniform Adversaries

Equivocal commitments are commitments introduced by DiCrescenzo et al [16] that have two computationally indistinguishable modes of setup. In the normal mode the setup outputs public parameters such that the commitment is statistically binding. In the alternate mode, the setup outputs public parameters and a trapdoor which can output commitments that open to both 0 and 1.

A setupless equivocal commitment sceme doesn't have a trusted setup algorithm. Instead we have an inefficient equivocation algorithm that can output commitments to both 0 and 1. The security of the scheme is guaranteed for adversaries that run in less than the equivocation time. A setupless equivocal commitment scheme, secure against uniform adversaries can be constructed from any setupless statistical hiding, computationally binding commitment scheme [19]. These can be built using a strong extractor and a keyless collision resistant hash function ([17, 28, 7]). But for non-uniform adversaries, it is easy to hardwire collisions for the setupless collision resistant hash function and hence break binding security of the scheme.

In order to achieve non-uniform security, Bitansky et al [7], suggested a multi-collision resistance assumption that essentially claims that hardwiring collisions is the best that an adversary can do. Informally, the K strong multi-collision resistant property states that any non-uniform adversary with advice advice can not output more than $\mathsf{K}(|\mathsf{advice}|)$ many collisions (assume that K blows up the

length). This assumption was used by Bitansky et al [7] to create statistically hiding commitments with a special binding against non-uniform adversaries.

We introduce a modified notion called "Setupless Equivocal Commitment with Auxiliary Input" that builds on these prior work, assumptions and takes in an auxiliary input $\mathsf{aux} \in \{0,1\}^*$ additionally and commits to a bit b and aux. The inefficient equivocation algorithm can take in any aux and output a commitment that can be open to both 0 and 1. We hide b (aux can not be hidden) while guaranteeing computational binding against non-uniform adversaries. We show that a similar construction showed by [7] using multi-collision resistant hash functions and a strong extractor also gives this notion.

4.1 Distinct Strong Keyless Multi-Collision Resistance

The definition from [7, 8] states that a non-uniform attacker with advice string advice cannot output more than $\mathsf{K}(\kappa, |\mathsf{advice}|)$ collisions (one can think of K as a polynomial that grows the advice length, [8] say this could, for instance, be a quadratic polynomial). We further weaken the definition so that the adversary is required to output all distinct elements in its pairs of collisions, i.e. letting $\mathbf{X} = \left(X_1^{(0)}, X_1^{(1)}, \ldots, X_K^{(0)}, X_K^{(1)}\right)$, we require that there do not exist any $i, j \in [K]^2$, $b, c \in \{0,1\}^2$ such that $X_i^{(b)} = X_j^{(c)}$. We call this modified notion distinct strong multi-collision resistance. Formally,

Definition 4.1 ((T,K)-Distinct Strong Multi-Collision Resistance). *Let $\mathsf{T} = \mathsf{T}(\cdot)$ and $\mathsf{K} = \mathsf{K}(\cdot,\cdot)$ be functions of the security parameter κ. A keyless hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ is (T,K) distinct strong multi-collision resistant if there is a negligible function negl such that for every polynomial size non-uniform adversary $\mathcal{A}$ that runs in time $\mathsf{poly}(\mathsf{T})$ and is given advice advice of length $\mathsf{poly}(\kappa)$, for every security parameter κ, for $\mathsf{T} = \mathsf{T}(\kappa)$ and $K = \mathsf{K}(\kappa, |\mathsf{advice}|)$,*

$$\Pr\left[\begin{array}{c}\left(X_1^{(0)}, X_1^{(1)}, \ldots, X_K^{(0)}, X_K^{(1)}\right) \\ \leftarrow \mathcal{A}(1^\kappa)\end{array} : \begin{array}{c}\forall (i,b) \neq (j,c) \in [K] \times \{0,1\}, \\ X_i^{(b)} \neq X_j^{(c)}, \\ \forall i \in [K], \mathsf{H.Hash}(1^\kappa, X_0^{(i)}) = \\ \mathsf{H.Hash}(1^\kappa, X_1^{(i)})\end{array}\right] \leq \mathsf{negl}(\kappa).$$

While this is not part of our definition, for applications we will require that the number of collisions remain linear in the size of advice, i.e., there is a fixed polynomial $p(\cdot)$ such that $\mathsf{K}(\kappa, |\mathsf{advice}|) \leq p(\kappa) \cdot |\mathsf{advice}|$. In our full version, we show that our assumption, namely (T,K)-distinct strong multi-collision resistance holds in the auxiliary-input random oracle model [43] with $p(\kappa)$ as small as 1, i.e. $\mathsf{K}(\kappa, |\mathsf{advice}|) \leq |\mathsf{advice}|$.

4.2 Setupless Equivocal Commitment with Auxillary Input

An auxiliary input equivocal commitment scheme $\mathsf{AuxEquiv}$ without setup consists of the algorithms:

$\mathsf{AuxEquiv.Com}(1^\kappa, \mathsf{aux}, b) \to (c, d)$ is a randomized PPT algorithm that takes in a bit $b \in \{0,1\}$, some auxiliary information $\mathsf{aux} \in \{0,1\}^*$ and security parameter $\kappa \in \mathbb{N}$ and outputs a commitment c, decommitment string d.

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

$\mathsf{AuxEquiv.Equivocate}(1^\kappa, \mathsf{aux}) \to (c, d_0, d_1)$ is an (inefficient) randomized algorithm that takes in the security parameter and some auxiliary information aux and outputs a commitment string c and decommitment strings to both 0 and 1.

Definition 4.2. *Correctness - We say an equivocal commitment scheme is perfectly correct if for all* $b \in \{0,1\}, \mathsf{aux} \in \{0,1\}^*$,

$$\Pr\left[\begin{matrix}(c,d) \leftarrow \mathsf{AuxEquiv.Com}(1^\kappa, \mathsf{aux}, b) \\ b' \leftarrow \mathsf{AuxEquiv.Decom}(\mathsf{aux}, c, d) \\ b' = b\end{matrix}\right] = 1$$

Definition 4.3. *Efficiency - We say an equivocal commitment scheme is efficient if* $\mathsf{AuxEquiv.Com}$ *and* $\mathsf{AuxEquiv.Decom}$ *run in* $\mathsf{poly}(\kappa, |\mathsf{aux}|)$ *time, and* $\mathsf{AuxEquiv.Equivocate}$ *runs in time* $\mathsf{poly}(2^\kappa, |\mathsf{aux}|)$.

We now define the binding and equivocal properties.

Definition 4.4. *An equivocal commitment without setup scheme is said to be* $(\mathsf{T}(\cdot), \mathsf{K}(\cdot))$ *binding secure if for any* non-uniform *adversary* $\mathcal{A}$ *running in time* $\mathsf{poly}(\mathsf{T}(\kappa))$ *for some polynomial and given an advice* $\mathsf{advice}(\kappa)$ *(for simplicity, denoted as* advice*) of length* $\mathsf{poly}(\kappa)$ *and a setting of* $K = \mathsf{K}(|\mathsf{advice}|, \kappa)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\Pr\left[\begin{matrix}\Big((\mathsf{aux}^{(1)}, c^{(1)}, d_0^{(1)}, d_1^{(1)}), \ldots, \\ (\mathsf{aux}^{(K)}, c^{(K)}, d_0^{(K)}, d_1^{(K)})\Big) \leftarrow \mathcal{A}(1^\kappa)\end{matrix} : \begin{matrix}\forall i \in [K], \\ \mathsf{Decom}(\mathsf{aux}^{(i)}, c^{(i)}, d_0^{(i)}) = 0, \\ \mathsf{Decom}(\mathsf{aux}^{(i)}, c^{(i)}, d_1^{(i)}) = 1 \\ \forall i \neq j \in [K], \mathsf{aux}^{(i)} \neq \mathsf{aux}^{(j)}\end{matrix}\right] \leq \mathsf{negl}(\kappa).$$

Definition 4.5. *We say that a scheme is equivocal if for all* $b \in \{0,1\}, \mathsf{aux} \in \{0,1\}^*$ *the statistical difference between the following two distributions is negligible in* κ.

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

4.3 Construction

We construct auxiliary-input equivocal commitments assuming a keyless hash function that is distinct strong multi-collision resistant and a strong extractor.

This is based on constructions introduced and presented in [17, 28, 7]. Let the keyless hash function be $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$. A $(\kappa, \mathsf{negl}(\kappa))$ strong extractor SExt (see full version for detailed preliminaries) that takes a seed of κ bits and an input of 3κ bits and outputs a single bit, $\mathsf{SExt} : \{0,1\}^\kappa \times \{0,1\}^{3\kappa} \to \{0,1\}$.

$\mathsf{AuxEquiv.Com}(1^\kappa, \mathsf{aux}, b) \to (c, d)$.
Sample a seed $g \leftarrow \{0,1\}^\kappa$. Choose $v \leftarrow \{0,1\}^{3\kappa}$. Compute $w = b \oplus \mathsf{SExt}(g, v)$. Compute $h = \mathsf{H.Hash}(1^\kappa, (\mathsf{aux}, v))$. Compute $c = (g, w, h)$ and $d = v$.

$\mathsf{AuxEquiv.Decom}(\mathsf{aux}, c, d) \to \{0, 1, \perp\}$
Parse c as (g, w, h). Check if $h = \mathsf{H.Hash}(1^\kappa, (\mathsf{aux}, d))$, output $\perp$ if fails. Output $w \oplus \mathsf{SExt}(g, d)$.

$\mathsf{AuxEquiv.Equivocate}(1^\kappa, \mathsf{aux}) \to (c, d_0, d_1)$
Sample a seed $g \leftarrow \{0,1\}^\kappa$ for a SExt. Sample $w \leftarrow \{0,1\}$. Sample $t \xleftarrow{R} \{0,1\}^{3\kappa}$.
Define $\mathcal{V}_t = \{v : \mathsf{H.Hash}(1^\kappa, (\mathsf{aux}, v)) = \mathsf{H.Hash}(1^\kappa, (\mathsf{aux}, t))\}$. Partition $\mathcal{V}_t = \mathcal{V}_t^0 \cup \mathcal{V}_t^1$ where $\mathcal{V}_t^i = \{v : v \in \mathcal{V}_t \wedge \mathsf{SExt}(g, v) = i\}$, output $\perp$ if either $\mathcal{V}_t^0$ or $\mathcal{V}_t^1$ are $\emptyset$.
Sample $v_0 \xleftarrow{R} \mathcal{V}_t^w, v_1 \xleftarrow{R} \mathcal{V}_t^{w \oplus 1}$. Output $\perp$ if no such v_0 or v_1 exist. $h \leftarrow \mathsf{H.Hash}(1^\kappa, (\mathsf{aux}, t))$. Output $((g, w, h), v_0, v_1)$.

We defer the analysis of this construction and a proof of the following lemma to the full version.

Lemma 4.6. *If $\mathsf{H}(\cdot)$ is a $(\mathsf{T}(\cdot), \mathsf{K}(\cdot,\cdot))$ distinct strong multi-collision resistant keyless hash function against non-uniform adversaries and SExt is a $(k, \epsilon) = (\kappa, \mathsf{negl}(\kappa))$ Strong Seeded extractor, then the construction above is a correct and efficient equivocal commitment scheme (Definition 4.3), and is $(\mathsf{T}(\cdot), \mathsf{K}(\cdot,\cdot))$-binding secure (Definition 4.4).*

4.4 Amplification

Lemma 4.7. *If there exists a $(\mathsf{T}(\cdot), \mathsf{K}(\cdot,\cdot))$-binding equivocal commitment scheme, then for any polynomial $p(\cdot)$, there exists a $(\mathsf{T}(\cdot), \mathsf{K}(\cdot,\cdot))/p(\kappa))$-binding equivocal commitment scheme.*

Proof. Let $\mathsf{Small.AuxEquiv.Com}, \mathsf{Small.AuxEquiv.Decom}, \mathsf{Small.AuxEquiv.Equivocate}$ be a $(\mathsf{T}(\cdot), \mathsf{K}(\cdot))$-binding equivocal commitment scheme. Consider a $p(\cdot)-$parallel repetition of $\mathsf{Small.AuxEquiv}$

$\mathsf{AuxEquiv.Com}(1^\kappa, \mathsf{aux}, b) \to (c, d)$.
For $i \in [p(\kappa)]$, run $(c_i, d_i) \leftarrow \mathsf{Small.AuxEquiv.Com}(1^\kappa, (\mathsf{aux}, i), b)$. Output $(c = \{c_i\}, d = \{d_i\})$

$\mathsf{AuxEquiv.Decom}(\mathsf{aux}, c, d) \to \{0, 1, \perp\}$
If $\exists b \in \{0,1\} : \forall i \in [p(\kappa)]$, $\mathsf{AuxEquiv.Decom}((\mathsf{aux}, i), c_i, d_i) = b$, output b. Otherwise output $\perp$.

$\mathsf{AuxEquiv.Equivocate}(1^\kappa, \mathsf{aux}) \to (c, d_0, d_1)$
For $i \in [p(\kappa)]$, run $(c_i, d_{0,i}, d_{1,i}) \leftarrow \mathsf{Small.AuxEquiv.Equivocate}(1^\kappa, (\mathsf{aux}, i))$.
Output $(c = \{c_i\}, d_0 = \{d_{0,i}\}, d_1 = \{d_{1,i}\})$

We defer the analysis of this construction to the full version.

□

Corollary 4.8. *Suppose there exists a* $(\mathsf{T}(\cdot), \mathsf{K}(\cdot,\cdot))$ *distinct strong collision resistant hash function satisfying Definition 4.1, for some* $\mathsf{K}(\kappa, |\mathsf{advice}|) = |\mathsf{advice}| \cdot p(\kappa)$ *for some* $p \in \mathsf{poly}(\kappa)$*. Then for every polynomial* $\mathsf{poly}(\cdot)$*, there exists a* $(\mathsf{T}(\cdot), \frac{|\mathsf{advice}|}{\mathsf{poly}(\kappa)})$*-binding equivocal commitment scheme.*

Proof. Fix the polynomial $p(\cdot)$ and the distinct strong collision resistant hash function that is guaranteed by the assumption. By lemma 4.6, there exists a correct and efficienct equivocal commitment that is $(\mathsf{T}(\cdot), p(\kappa) \cdot |\mathsf{advice}|)$-binding. Fix any polynomial $\mathsf{poly}(\kappa)$. Then by invoking lemma 4.7 on the polynomial $\mathsf{poly}(\kappa) \cdot p(\kappa)$, we have that there exists a $(\mathsf{T}(\cdot), \frac{|\mathsf{advice}|}{\mathsf{poly}(\kappa)})$-binding equivocal commitment scheme. □

5 Hinting PRGs with injective extension

A hinting pseudorandom generator as introduced by Koppula and Waters[33] is a pseudorandom generator with an enhanced security property. In this security game blocks that are output from the PRG are interspersed with random blocks where the placement is according to the seed of the PRG.

In this section we introduce a variant of Hinting PRGS that we call Hinting PRGs with injective extension. Our variant follows along the lines of the original, but with two critical modifications. The first is that we slightly relax the security game. On a seed s of length n bits, the hinting PRG outputs length $n+1$ blocks each consisting of ℓ bits. Informally, our security guarantee is that the adversary cannot distinguish between the following two distributions, each consisting of $(2n + 1)$ blocks. In both distributions, all blocks but the first are generated identically: these output as a $2 \times n$ matrix where for all $i \in [n]$ the $(s_i, i)^{th}$ entry is set according to the $(i+1)^{th}$ block of the PRG evaluation, while the $(1-s_i, i)^{th}$ entry is a uniformly random string. In the first distribution, the first ℓ-bit block is set as the first block of the PRG evaluation, and in the second distribution, the first ℓ-bit block is set uniformly at random.

This relaxed security definition differs from the original security definition in which the second distribution consists of all random blocks. It is fairly easy to observe that our relaxed notion also suffices for performing the CCA transformation of [33] and will also suffice for our purposes. The primary reason for relaxing the security definition, is that it makes it easier to realize our second modification.

We additionally define an injective extension for the hinting PRG, where we require that the Hinting PRG evaluation algorithm additionally outputs a

separate block that is injective with respect to the seed. To ensure injectivity we will define an algorithm that checks the Hinting PRG public parameters and outputs 0 if the public parameters were sampled so that the extended block might not be an injective function of the seed. That is there could be two seeds that output the same extended block. If the check function outputs 1, the extended block will be an injective function of the seed. The hinting PRG scheme consists of the following algorithms,

- $\mathsf{Setup}(1^\kappa, 1^\ell)$: 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.
- $\mathsf{ExtEval}\,(\mathsf{pp}, s \in \{0,1\}^n)$: The extended evaluation algorithm takes as input the public parameters pp, an n bit string s and outputs a string of length $m = m(\kappa, \ell)$.
- $\mathsf{CheckParams}\,(\mathsf{pp}, n)$: The algorithm takes as input the public parameters pp, the seed input length n and checks them to see if the function sampled is injective or not. It outputs $\{0,1\}$ accordingly.

Definition 5.1. *A hinting PRG scheme is said to be non-uniform $T(\cdot)$-secure if for any polynomial $\ell(\cdot)$ and any adversary $\mathcal{A}$ running in time $\mathsf{poly}(T(\kappa))$ and $\mathsf{poly}(\kappa)$ advice, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds:*

$$\left|\Pr\left[\beta \leftarrow \mathcal{A}\Big(\mathsf{pp}, \big(r_0^\beta, r_{\mathsf{ext}}, \{r_{i,b}\}_{i\in[n],b\in\{0,1\}}\big)\Big) : \begin{array}{c}(\mathsf{pp}, n) \leftarrow \mathsf{Setup}(1^\kappa, 1^{\ell(\kappa)}), s \leftarrow \{0,1\}^n, \\ r_0^0 = \mathsf{Eval}(\mathsf{pp}, s, 0), r_0^1 \leftarrow \{0,1\}^\ell, \\ r_{\mathsf{ext}} = \mathsf{ExtEval}(\mathsf{pp}, s), \beta \leftarrow \{0,1\}, \\ r_{i,s_i} = \mathsf{Eval}(\mathsf{pp}, s, i), r_{i,\overline{s_i}} \leftarrow \{0,1\}^\ell \ \forall\ i \in [n]\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\kappa).$$

Definition 5.2. *A hinting PRG scheme is said to be extended injectively if for any security parameter $\kappa \in \mathbb{N}$, any polynomial $\ell(\cdot)$ and any $\mathsf{pp} \in \{0,1\}^*$ the following holds,*

$$\Pr\left[\begin{array}{c}\exists s_1 \neq s_2 \in \{0,1\}^n, \\ \mathsf{ExtEval}(\mathsf{pp}, s_1) = \mathsf{ExtEval}(\mathsf{pp}, s_2)\end{array} : \begin{array}{c} n \in \mathbb{N} \\ \mathsf{CheckParams}(\mathsf{pp}, n) = 1\end{array}\right] = 0.$$

Definition 5.3. *A hinting PRG scheme is setup such that it outputs injective parameters if for any security parameter $\kappa \in \mathbb{N}$, any polynomial $\ell(\cdot)$ the following holds,*

$$\Pr\left[\mathsf{CheckParams}(\mathsf{pp}, n) = 0 : (\mathsf{pp}, n) \leftarrow \mathsf{Setup}(1^\kappa, 1^{\ell(\kappa)})\right] = 0.$$

Definition 5.4. *A hinting PRG scheme is succinct if the length of the seed n, public parameters and injective extension are independent of the block length parameter ℓ.*

Theorem 5.5. *If there exists an injective sub-exponentially secure one way function, either of the three assumptions - DDH, CDH or LWE - are sub-exponentially secure, and there exists a sub-exponentially secure pseudorandom generator, then there exists a hinting PRG scheme that can be extended injectively, outputs injective parameters, is succinct and for some constant $\delta \in (0,1)$, satisfies non-uniform 2^{κ^δ}-security.*

We defer the construction and its analysis to the full version.

6 Tag Amplification

We discuss how to amplify a non-uniform subexponentially secure CCA scheme for $N' = 4N$ tags to a scheme with 2^N tags. We will perform the amplification using non uniform subexponentially secure primitives $\mathsf{AuxEquiv}$ (Section 4), extended hinting PRG (Section 5). The amplification algorithm runs in time polynomial in N and the runtime of the primitives involved, thus N should always stay polynomial in the security parameter for the amplification to be an efficient algorithm.

Let the hinting PRG scheme $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{ExtEval}, \mathsf{CheckParams})$ be a succinct $T = 2^{\kappa^\gamma}$ secure for some constant $\gamma \in (0,1)$. Let $\mathsf{AuxEquiv}$ be $T = 2^{\kappa^\delta}$-binding secure and statistically hiding where $\delta \in (0,1)$. Let $(\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover})$ be a 2^{κ^c}-subexponentially secure, weak binding, 2^{κ^v}-efficient CCA commitment scheme for $N'(\kappa) = N' = 4N$ tags where $c < 1$ and $v \geq 1$ for message length $u(\kappa)$[9]. 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 $\ell(\kappa)$.

Let m be the message input to the commitment algorithm and length be denoted by $|m|$. Let $n' = n'(\kappa)$ be the length of the seed plus public parameters plus injective extension of the hinting PRG scheme when invoked on security parameter $\kappa'' = \kappa^{\frac{v}{\delta\cdot\gamma}}$. Since the scheme is succinct, n' is a function of only κ'' (and hence κ) and not the block length, which we will specify later. By Lemma 4.7, we will use a $(2^{\kappa^\delta}, \frac{|\mathsf{advice}|}{2\cdot n'})$-binding secure commitment scheme $\mathsf{AuxEquiv}$, and let $|y|$ refer to the length of the decommitment strings of said scheme. Finally, we run $\mathsf{Small.Com}$ on messages of size $|y|$, and let ℓ be the size of randomness used by $\mathsf{Small.Com}$ on said input size. We set the block size of our hinting PRG scheme to be the maximum of $|m|, N\cdot\ell$. For ease of notation we assume that $\mathsf{HPRG.Eval}(\mathsf{pp}, s, 0) \in \{0,1\}^{|m|}$ and $\forall i \in [n]$, $\mathsf{HPRG.Eval}(\mathsf{pp}, s, i) \in \{0,1\}^{\ell\cdot N}$, i.e. we ignore any extra bits output by the $\mathsf{HPRG.Eval}$ algorithm. Let $\Theta(\kappa^{\tilde{v}})$ denote the length of the seed n in relation to the security parameter.

Our transformation will produce three algorithms, $(\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$ which we prove non-uniform 2^{κ^c}-subexponentially secure and $2^{\kappa^{v'}}$-efficient where $v' = \frac{v\cdot\tilde{v}}{\delta\cdot\gamma}$. The construction will call $\mathsf{AuxEquiv}$ on security parameter $\kappa' = \kappa^{\frac{v}{\delta}}$, HPRG on security parameter $\kappa'' = \kappa^{\frac{v}{\delta\cdot\gamma}}$ and Small on security parameter κ.

[9] Recall from Definition 3.3 that a 2^{κ^v}-efficient scheme with $v \geq 1$ implies that the runtime of $\mathsf{Small.Val}$ is polynomial in 2^{κ^v}.

The different parameters will help us perform complexity leveraging. For simplicity, we assume that the message space of Small, $u(\kappa)$ is equal to the length of the decommitment string of the equivocal commitment called on κ'. We will ensure this property is satisfied in Section 7 when we recursively amplify the tags. The $\mathsf{CCA.Val}$ procedure in our transformation will be an inefficient algorithm that brute forces through each hinting PRG seed and run in time 2^n where $n = \Theta(\kappa''^{\tilde{v}})$. Thus our transformation will increase the runtime of $\mathsf{CCA.Val}$ from $\mathsf{Small.Val}$ that runs in time 2^{κ^v} to $2^{\kappa^{v'}}$.

Additionally, we will also present a fourth non-uniform algorithm $\mathsf{CCA.ValAlt}$, which is only used in the proof and depends on the non-uniform advice it gets. In our proof we will first change how we answer an adversary's decommitment queries by using $\mathsf{CCA.ValAlt}$ to answer instead of $\mathsf{CCA.Val}$. Since the queries made to the $\mathsf{CCA.Val}$ oracles differ in at least one position from tag^*, $\mathsf{CCA.ValAlt}$ will crucially rely on the security of $\mathsf{Small.Com}$ at this position by making calls to $\mathsf{Small.Val}$ to help in decommitment.

$\mathsf{CCA.ValAlt}(\mathsf{tag}^*, \mathsf{com}, \mathcal{L}) \to m \cup \perp$ is a deterministic inefficient algorithm that takes in tag^*, a commitment com and a non-uniform advice list $\mathcal{L}$ and outputs either a message $m \in \{0,1\}^w$ 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.

$\mathsf{CCA.FindSeed}(\mathsf{aux})$

Inputs: String $\mathsf{aux} = (\mathsf{HPRG.pp}, \mathsf{aux}')$ **Output:** $\tilde{s} \in \{0,1\}^n \cup \perp$

- Parse aux as $(\mathsf{HPRG.pp}, \mathsf{aux}')$
- Iterate through all $\tilde{s} \in \{0,1\}^n$
  - If $\mathsf{aux}' = \mathsf{HPRG.ExtEval}(\mathsf{HPRG.pp}, \tilde{s})$, return s.
- Return $\perp$

Fig. 1: Routine $\mathsf{CCA.FindSeed}$

We now describe our transformation.

Transformation $\mathsf{Amplify}(\mathsf{Small} = (\mathsf{Small.Com}, \mathsf{Small.Val}, \mathsf{Small.Recover}),$
$\mathsf{HPRG}, \mathsf{AuxEquiv}, w(\kappa), v') \to \mathsf{NM} = (\mathsf{CCA.Com}, \mathsf{CCA.Val}, \mathsf{CCA.Recover})$:

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

1. Compute $\kappa' = \kappa^{\frac{v}{\delta}}$. Compute $\kappa'' = \kappa'^{\frac{1}{\gamma}}$.[10]

[10] The variables δ and γ are known from the security guarantees of $\mathsf{AuxEquiv}, \mathsf{HPRG}$ respectively.

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

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

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

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 $\tilde{s}_i \neq \mathsf{AuxEquiv.Decom}(\mathsf{aux}, \sigma_i, \tilde{y}_i)$, output 0.
- Parse aux as $(\mathsf{HPRG.pp}, \mathsf{aux}')$.
- If $\mathsf{HPRG.CheckParams}(\mathsf{HPRG.pp}, n) = 0$, output 0.
- If $\mathsf{aux}' \neq \mathsf{HPRG.ExtEval}(\mathsf{HPRG.pp}, s)$ output 0.
- If all the above checks have passed, output 1.

Fig. 2: Routine CCA.Check

$\mathsf{CCA.FindAlt}(x', \mathsf{com}, \mathcal{L})$

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

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

Polynomial Size Non-Uniform Advice List $\mathcal{L}$

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

- If for some $\tilde{s} \in \{0,1\}^n$, $(\mathsf{com.aux}, \tilde{s}) \in \mathcal{L}$, where $\tilde{s}$ is the seed recorded from the advice. Output $\tilde{s}$.
- Else if $\mathsf{com.aux}$ is not recorded in $\mathcal{L}$,
  - For each $i \in [n]$
    1. Let $\tilde{y}_i = \mathsf{Small.Val}(c_{x',i,0})$
    2. Set $\tilde{z}_i = \mathsf{AuxEquiv.Decom}(\mathsf{aux}, \sigma_i, \tilde{y}_i)$. 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. 3: Routine CCA.FindAlt

2. Sample $(\mathsf{HPRG.pp}, n) \leftarrow \mathsf{HPRG.Setup}(1^{\kappa''}, 1^{\max(|m|, N\cdot\ell)})$.
3. Sample $s = s_1 \ldots s_n \xleftarrow{R} \{0,1\}^n$ as the seed of the extended hinting PRG.
4. Set $\mathsf{aux} = (\mathsf{HPRG.pp}, \mathsf{HPRG.ExtEval}(\mathsf{HPRG.pp}, s))$.
5. For all $i \in [n]$ run $\mathsf{AuxEquiv.Com}(1^{\kappa'}, \mathsf{aux}, s_i) \rightarrow (\sigma_i, y_i)$.
6. Let for $x \in [N], i \in [n]$, $r_{x,i}, \tilde{r}_{x,i} \in \{0,1\}^\ell$ be defined as follows:
7. 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 $(\tilde{r}_{1,i}, \tilde{r}_{2,i}, \ldots, \tilde{r}_{N,i}) \xleftarrow{R} \{0,1\}^{N \cdot \ell}$
8. Compute $c = m \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, s, 0)$
9. 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. $c_{x,i,1} = \mathsf{Small.Com}(1^\kappa, (x, \mathsf{tag}_x, 1), \mathsf{msg} = y_i; r_{x,i})$
10. Output $\mathsf{com} = \left(\mathsf{tag}, \mathsf{aux}, c, (\sigma_i, (c_{x,i,0}, c_{x,i,1})_{x \in [N]})_{i \in [n]})\right)$ as the commitment. All of the randomness is used as the decommitment string.

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

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

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

1. If $\mathsf{com.tag} = \mathsf{tag}^*$, output $\perp$.
2. Let x^* be the smallest index where the bits of $\mathsf{tag}^*, \mathsf{com.tag}$ differ.
3. Set $\tilde{s} = \mathsf{CCA.FindAlt}(x^*, \mathsf{com}, \mathcal{L})$.
4. If $\mathsf{CCA.Check}(\tilde{s}, \mathsf{com}) = 0$ output $\perp$.
5. Output $c \oplus \mathsf{HPRG.Eval}(\mathsf{HPRG.pp}, \tilde{s}, 0)$.

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

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

7 Compilation of Transformations

We show how to combine our transformations Amplify and OneToMany to prove that if we start with a base scheme that is secure against non-uniform "same tag" adversaries (see Definition 3.7) for $32 \cdot \mathsf{ilog}(q, \kappa)$ tags where the notation $\mathsf{ilog}(q, \kappa)$ denotes $\underbrace{\lg \lg \cdots \lg}_{q \text{ times}}(\kappa)$ [11] and q is some constant, then using our described transformations, we can construct a scheme that is secure against non-uniform adversaries (see Definition 3.6) for $16 \cdot 2^\kappa$ tags.

Our sequence of transformations is very similar to [19], where we start with a base scheme BaseCCA that satisfies property Definition 3.8. We then remove the same tag restriction on the adversary by using the transformation OneToMany (described in the full version) and then amplify the tag space by using the transformation Amplify in Section 6 $q + 1$ times. The two main deviations from

[11] The notation $\mathsf{ilog}(0, \kappa)$ is defined as κ.

the formal treatment of [19] is due to our proof technique, i.e. we need to keep track of the message and efficiency of the val oracle when we perform the sequence of transformations.

We remind the reader that the order of the sequence of transformations is important as to perform Amplify and OneToMany we need the commitment scheme to be recoverable from randomness. Additionally, OneToMany does computation that is polynomial in the number of tags for the input scheme. Thus, we must remove the "same tag" restriction from our adversary before amplifying our tags with Amplify. Based on the sequence of transformations we have discussed, our tag space will amplify as follows. At the end of OneToMany, we will end up with $16 \cdot \mathsf{ilog}(q, \kappa)$ sized tag space. And after $q + 1$ applications of Amplify, we will end up with $16 \cdot 2^{\kappa}$ sized tag space. One application of Amplify converts a $4N$ tag space scheme to a 2^N tag space scheme. Thus on input a $4 \cdot 4 \cdot \mathsf{ilog}(q, \kappa)$ tag space, one gets a $2^{4 \cdot \mathsf{ilog}(q,\kappa)} = 16 \cdot \mathsf{ilog}(q-1, \kappa)$ tag space.

Additionally, when using the schemes in a sequence of transformations we need to keep track of the message spaces we chose in our output scheme. For instance, to perform the transformation Amplify and OneToMany, the constructions output committment σ to each seed bit of the hinting PRG. The base scheme here takes in the decommitment string of σ as input. Thus the length of the base scheme being transformed should be able to support messages of this length for the transformation to be correct. Let the length of the decommitment string be denoted by a polynomial function $\mathsf{DecomLen}(\cdot)$ that takes as input the security parameter κ [12]. Thus for the transformations Amplify and OneToMany, u (input message length of the base scheme) should be equal to $\mathsf{DecomLen}(\kappa')$ where κ' is the security parameter input to the equivocal commitment. In our transformations κ' is set as $\kappa^{\frac{v}{\delta}}$ where there exists a constant δ such that the setupless equivocal commitment scheme is $2^{\kappa^{\delta}}$-hiding secure and the base scheme is $2^{\kappa^{v}}$-efficient[13].

Our formal transformation is below. We start with a base commitment scheme BaseCCA and output the scheme $(\mathsf{AmplifiedCCA}^{q+1}.\mathsf{Com}, \mathsf{AmplifiedCCA}^{q+1}.\mathsf{Val})$. We list a few assumptions on our transformation -

- Let there exist variables $\delta, \gamma, \tilde{v}$ such that $\delta \in (0,1)$ and the setupless equivocal commitment scheme is $2^{\kappa^{\delta}}$-hiding secure, $\gamma \in (0,1)$ and the hinting PRG with injective extension is $2^{\kappa^{\gamma}}$-secure and the dependence of seed on the security parameter be such that seed length $n = \Theta(\kappa^{\tilde{v}})$.
- We start with a base scheme that is 2^{κ}-efficient and secure against non-uniform "same tag" $2^{\kappa^{c}}$-subexponentially secure adversaries for tag space $32\mathsf{ilog}(q, \kappa)$ tags for any constant q.

[12] The length of the decommitment string can depend on aux, but since aux is also called with a polynomial function in κ based on the hinting PRG construction, we simplify the notation. In our specific construction for AuxEquiv in Section 4, the decommitment string length doesn't depend on aux.

[13] Recall from Definition 3.3 that a $2^{\kappa^{v}}$-efficient scheme with $v \geq 1$ implies that the runtime of Small.Val is polynomial in $2^{\kappa^{v}}$.

If the base scheme runs in time some constant $\mathsf{poly}(2^{\kappa^a})$ where $a \in (0,1)$ then the scheme is 2^κ-efficient. Otherwise, on input security parameter κ, we can run the scheme with parameters $\kappa^{\frac{1}{a}}$ to get a 2^κ-effcent scheme that is still 2^{κ^c} sub-exponentially secure with $c \in (0,1)$ for some constant c. Thus we can wlog claim that we start with a 2^κ-efficient scheme. This will help simplify notation.

– Let the base scheme support messages of length $u = \mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{1}{\delta}})$ and the final scheme support messages of length w.

Recall that the transformations OneToMany (see full version) and Amplify (Section 6) take in the following parameters - a scheme to be transformed, hinting PRG with injective extension HPRG, setupless equivocal commitment scheme AuxEquiv, the length of the messages supported by the output scheme and an efficiency parameter v such that the output scheme is 2^{κ^v}-efficient.

$\mathsf{CompiledAmplify}(\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val}, u), \mathsf{HPRG}, \mathsf{AuxEquiv}, w)$

1. $\mathsf{AmplifiedCCA}^0 \leftarrow \mathsf{OneToMany}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{AuxEquiv}, \mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{v_0}{\delta}}), v_0)$ where $v_0 = \frac{\tilde{v}}{\delta \cdot \gamma}$.
2. For $i \in [q]$,
   (a) $\mathsf{AmplifiedCCA}^i \leftarrow \mathsf{Amplify}(\mathsf{AmplifiedCCA}^{i-1}, \mathsf{HPRG}, \mathsf{AuxEquiv}, \mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{v_i}{\delta}}), v_i)$ where $v_i = \left(\frac{\tilde{v}}{\delta \cdot \gamma}\right)^{i+1}$.
3. $\mathsf{AmplifiedCCA}^{q+1} \leftarrow \mathsf{Amplify}(\mathsf{AmplifiedCCA}^q, \mathsf{HPRG}, \mathsf{AuxEquiv}, w, v_{q+1})$ where $v_{q+1} = \left(\frac{\tilde{v}}{\delta \cdot \gamma}\right)^{q+2}$.
4. Output $(\mathsf{AmplifiedCCA}^{q+1}.\mathsf{Com}, \mathsf{AmplifiedCCA}^{q+1}.\mathsf{Val})$

Below we analyze CompiledAmplify by stating theorems on correctness, efficiency and security.

Theorem 7.1. *For every $\kappa \in \mathbb{N}$, any constant q, any polynomial w, let* $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val}, u)$ *be a perfectly correct CCA commitment scheme for message space $\{0,1\}^u$ by Definition 3.2 with tag space $32 \cdot \mathsf{ilog}(q, \kappa)$. Let* $\mathsf{AuxEquiv} = (\mathsf{AuxEquiv.Com}, \mathsf{AuxEquiv.Decom}, \mathsf{AuxEquiv.Equivocate})$ *be a perfectly correct equivocal commitment scheme by Definition 4.2. Let there exist a constant δ such that $u = \mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{1}{\delta}})$.*

Then, we have that the scheme $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{AuxEquiv}, w)$ *is a perfectly correct CCA commitment scheme for $16 \cdot 2^\kappa$ tags.*

Theorem 7.2. *For every $\kappa \in \mathbb{N}$, any constant q, any polynomial w, let* $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val}, u)$ *be an 2^κ-efficient CCA commitment scheme by Definition 3.3 with tag space $32 \cdot \mathsf{ilog}(q, \kappa)$. Let* $\mathsf{AuxEquiv} = (\mathsf{Equiv.Com}, \mathsf{Equiv.Decom}, \mathsf{Equiv.Equivocate})$ *be an efficient equivocal commitment scheme by Definition 4.3. Let there exist constants $\delta, \gamma, \tilde{v}$ such that setupless equivocal commitment scheme is 2^{κ^δ}-hiding secure and $u = \mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{1}{\delta}})$; $\gamma \in (0,1)$ and the hinting PRG with injective extension is 2^{κ^γ}-secure; the dependence of seed on the security parameter be such that $n = \Theta(\kappa^{\tilde{v}})$.*

Then, $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{AuxEquiv}, w)$ *is an* $2^{\kappa^v_{q+1}}$*-efficient CCA commitment scheme for* $16 \cdot 2^\kappa$ *tags where* $v_{q+1} = \left(\frac{\tilde{v}}{\delta \cdot \gamma}\right)^{q+2}$.

Theorem 7.3. *For every* $\kappa \in \mathbb{N}$*, any constant* q*, any polynomial* w*, let* $\mathsf{BaseCCA} = (\mathsf{BaseCCA.Com}, \mathsf{BaseCCA.Val}, u)$ *be a CCA commitment scheme that is hiding against non-uniform "same tag"* 2^{κ^c}*-subexponential adversaries according to Definition 3.7 for tag space* $32 \cdot \mathsf{ilog}(q, \kappa)$*.* $\mathsf{HPRG} = (\mathsf{HPRG.Setup}, \mathsf{HPRG.Eval})$ *be a hinting PRG scheme with injective extension that is* $T = 2^{\kappa^\gamma}$ *secure by Definition 5.1 for* $\gamma \in (0, 1)$*.* $\mathsf{AuxEquiv} = (\mathsf{AuxEquiv.Com}, \mathsf{AuxEquiv.Decom}, \mathsf{AuxEquiv.Equivocate})$ *be an equivocal commitment without setup scheme that is* $T = 2^{\kappa^\delta}$ *binding secure Definition 4.4 and statistically hiding for some constant* $\delta \in (0, 1)$*. Let* u *be equal to* $\mathsf{AuxEquiv.DecomLen}(\kappa^{\frac{1}{\delta}})$*.*

Then, $\mathsf{CompiledAmplify}(\mathsf{BaseCCA}, \mathsf{HPRG}, \mathsf{AuxEquiv}, w)$ *is a CCA commitment scheme that is hiding against non-uniform* 2^{κ^c}*-subexponential adversaries according to Definition 3.6 for tag space* $16 \cdot 2^\kappa$*.*

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

Theorem 7.4. *[29] For every constant* $c > 0$*, there exist correct, polynomially efficient, binding (3.4), same-tag CCA secure commitments with randomness recovery satisfying Definition 3.7 against non-uniform adversaries, with tag space* $(c \lg \lg \lg \kappa)$*, message space* $u = \mathsf{poly}(\kappa)$ *that make black-box use of subexponential quantum hard non-interactive commitments and subexponential classically hard non-interactive commitments in* BQP*, both with randomness recovery.*

Theorem 7.5. *[36] For every constant* $c > 0$*, there exist correct, polynomially efficient, weak binding (3.5), same-tag CCA secure commitments with randomness recovery satisfying same-tag CCA security according to Definition 3.7 against non-uniform adversaries, with tag space* $(c \lg \lg \lg \kappa)$*, that make black-box use of subexponential time-lock puzzles [36].*

We remark that while [36, 29] prove that their constructions satisfy non-malleability with respect to commitment, their proof techniques also extend to exhibit same-tag CCA security against non-uniform adversaries. In a nutshell, both these works rely on two simultaneous axes of hardness to build their base schemes. As a consequence of this in the same-tag setting, for any pair of tags $(\mathsf{tag}, \widetilde{\mathsf{tag}})$ corresponding to the challenge query and CCA oracle queries of the adversary respectively, there is an oracle that inverts all commitments generated under $\widetilde{\mathsf{tag}}$ but where commitments under tag remain secure in the presence of this oracle. In both these works [36, 29], we note that while the specific oracle is only used to invert parallel queries of the adversary (thereby obtaining many-many non-malleability), the oracle is actually capable of inverting (unbounded) polynomially many *adaptive* queries, thereby also achieving same-tag CCA security. In [36], this oracle over-extracts, therefore achieving the weaker property of same-tag CCA security with weak binding. The [29] scheme does not suffer from over-extraction and achieves the stronger notion of (standard) binding.

The [29] scheme can be observed to satisfy randomness recovery by relying on the recovery algorithm of the underlying commitments. The [36] scheme outputs a commitment to a bit b as

$$f(s;r), r', \langle s, r' \rangle \oplus b$$

which satisfies randomness recovery given all the randomness used to commit.

Combining this theorem with Theorem 7.3, we obtain the following corollaries.

Corollary 7.6. *There exists a perfectly correct, polynomially efficient, binding (Definition 3.4) and CCA secure commitment satisfying Definition 3.6 against non-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.*

Corollary 7.7. *There exists a perfectly correct, polynomially efficient, binding (Definition 3.4) and CCA secure commitment satisfying Definition 3.6 against non-uniform adversaries, with tag space 2^κ for security parameter κ, that makes black-box use of subexponential time-lock puzzles as used in [36], subexponential hinting PRGs and subexponential keyless collision-resistant hash functions.*

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.

8 Acknowledgments

We thank Daniel Wichs for a useful discussion about the construction of our new Hinting PRGs, anonymous reviewers for helpful feedback on a preliminary version of this work, and Nir Bitansky and Rachel Lin for answering our questions about keyless collision-resistant hash functions.

D. Khurana was supported in part by NSF CNS - 2238718, DARPA SIEVE and a gift from Visa Research. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024. Brent Waters was supported by NSF CNS-1908611, Simons Investigator award and Packard Foundation Fellowship.

References

1. Ananth, P., Choudhuri, A.R., Jain, A.: A new approach to round-optimal secure multiparty computation. In: CRYPTO (2017)

2. Badrinarayanan, S., Goyal, V., Jain, A., Kalai, Y.T., Khurana, D., Sahai, A.: Promise zero knowledge and its applications to round optimal MPC. In: CRYPTO (2018)
3. Badrinarayanan, S., Goyal, V., Jain, A., Khurana, D., Sahai, A.: Round optimal concurrent MPC via strong simulation. In: TCC (2017)
4. Barak, B.: Constant-Round Coin-Tossing with a Man in the Middle or Realizing the Shared Random String Model. In: FOCS (2002)
5. Barak, B., Ong, S.J., Vadhan, S.P.: Derandomization in cryptography. SIAM J. Comput. (2007)
6. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: EUROCRYPT (2018)
7. Bitansky, N., Kalai, Y.T., Paneth, O.: Multi-collision resistance: a paradigm for keyless hash functions. In: STOC (2018)
8. Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: Theory of Cryptography (2018)
9. Bitansky, N., Paneth, O.: Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In: TCC (2015)
10. Brakerski, Z., Halevi, S., Polychroniadou, A.: Four round secure computation without setup. In: TCC (2017)
11. 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)
12. Canetti, R., Lin, H., Pass, R.: Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions. In: FOCS (2010)
13. Choudhuri, A.R., Ciampi, M., Goyal, V., Jain, A., Ostrovsky, R.: Round optimal secure multiparty computation from minimal assumptions. Cryptology ePrint Archive, Report 2019/216 (2019)
14. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Concurrent non-malleable commitments (and more) in 3 rounds. In: CRYPTO (2016)
15. Ciampi, M., Ostrovsky, R., Siniscalchi, L., Visconti, I.: Four-round concurrent non-malleable commitments from one-way functions. In: CRYPTO (2017)
16. Crescenzo, G.D., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: Vitter, J.S. (ed.) STOC (1998)
17. Damgård, I.B., Pedersen, T.P., Pfitzmann, B.: On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In: CRYPTO (1993)
18. Dolev, D., Dwork, C., Naor, M.: Non-Malleable Cryptography (Extended Abstract). In: STOC (1991)
19. Garg, R., Khurana, D., Lu, G., Waters, B.: Black-box non-interactive non-malleable commitments. In: EUROCRYPT (2021)
20. Goyal, R., Vusirikala, S., Waters, B.: New constructions of hinting prgs, owfs with encryption, and more. IACR Cryptology ePrint Archive (2019)
21. Goyal, V.: Constant Round Non-malleable Protocols Using One-way Functions. In: STOC (2011)
22. Goyal, V., Lee, C.K., Ostrovsky, R., Visconti, I.: Constructing non-malleable commitments: A black-box approach. In: FOCS (2012)
23. Goyal, V., Pandey, O., Richelson, S.: Textbook non-malleable commitments. In: STOC (2016)
24. Goyal, V., Richelson, S.: Non-malleable commitments using goldreich-levin list decoding. In: FOCS (2019)
25. Goyal, V., Richelson, S., Rosen, A., Vald, M.: An algebraic approach to non-malleability. In: FOCS (2014)

26. Groth, J., Ostrovsky, R., Sahai, A.: New techniques for noninteractive zero-knowledge. J. ACM (2012)
27. Halevi, S., Hazay, C., Polychroniadou, A., Venkitasubramaniam, M.: Round-optimal secure multi-party computation. In: CRYPTO (2018)
28. Halevi, S., Micali, S.: Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In: CRYPTO (1996)
29. Kalai, Y.T., Khurana, D.: Non-interactive non-malleability from quantum supremacy. In: CRYPTO (2019)
30. Khurana, D.: Round optimal concurrent non-malleability from polynomial hardness. In: TCC (2017)
31. Khurana, D.: Non-interactive distributional indistinguishability and non-malleable commitments. In: EUROCRYPT (2021)
32. Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: FOCS (2017)
33. Koppula, V., Waters, B.: Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In: CRYPTO (2019)
34. Lin, H., Pass, R.: Non-malleability Amplification. In: STOC (2009)
35. Lin, H., Pass, R.: Constant-round Non-malleable Commitments from Any One-way Function. In: STOC (2011)
36. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: FOCS (2017)
37. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent Non-malleable Commitments from Any One-Way Function. In: TCC (2008)
38. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive One-Way Functions and Applications. In: CRYPTO (2008)
39. Pass, R.: Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In: TCC (2013)
40. Pass, R., Rosen, A.: Concurrent Non-Malleable Commitments. In: FOCS (2005)
41. Pass, R., Rosen, A.: New and Improved Constructions of Nonmalleable Cryptographic Protocols. SIAM J. Comput. (2008)
42. Pass, R., Wee, H.: Constant-round non-malleable commitments from sub-exponential one-way functions. In: EUROCRYPT (2010)
43. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO (2007)
44. Wee, H.: Black-box, round-efficient secure computation via non-malleability amplification. In: FOCS (2010)