Non-Malleable Vector Commitments via Local Equivocability

Lior Rotem* Gil Segev†

Abstract

Vector commitments (VCs), enabling to commit to a vector and locally reveal any of its entries, play a key role in a variety of both classic and recently-evolving applications. However, security notions for VCs have so far focused on passive attacks, and non-malleability notions considering active attacks have not been explored. Moreover, existing frameworks that may enable to capture the non-malleability of VCs seem either too weak (non-malleable non-interactive commitments that do not account for the security implications of local openings) or too strong (non-malleable zero-knowledge sets that support both membership and non-membership proofs).

We put forward a rigorous framework capturing the non-malleability of VCs, striking a careful balance between the existing weaker and stronger frameworks: We strengthen the framework of non-malleable non-interactive commitments by considering attackers that may be exposed to local openings, and we relax the framework of non-malleable zero-knowledge sets by focusing on membership proofs. In addition, we strengthen both frameworks by supporting (inherently-private) updates to entries of committed vectors, and discuss the benefits of non-malleable VCs in the context of both UTXO-based and account-based stateless blockchains, and in the context of simultaneous multi-round auctions (that have been adopted by the US Federal Communications Commission as the standard auction format for selling spectrum ranges).

Within our framework we present a direct approach for constructing non-malleable VCs whose efficiency essentially matches that of the existing standard VCs. Specifically, we show that any VC can be transformed into a non-malleable one, relying on a new primitive that we put forth. Our new primitive, *locally-equivocable commitments with all-but-one binding*, is evidently both conceptually and technically simpler compared to multi-trapdoor mercurial trapdoor commitments (the main building block underlying existing non-malleable zero-knowledge sets), and admits more efficient instantiations based on the same number-theoretic assumptions.

*Computer Science Department, Stanford University, 353 Jane Stanford Way, Stanford, CA 94305, USA. Email: lrotem@cs.stanford.edu. Supported by a research grant from Protocol Labs.

†School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: segev@cs.huji.ac.il. Supported by the Israel Science Foundation (Grant No. 1336/22) and by the European Union (ERC, FTRC, 101043243). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

Contents

1 Introduction

Vector commitments (VCs) [LY10, CF13] enable to non-interactively commit to a vector $(x_1, \ldots, x_q)$ while offering the useful property of *local opening*: The committer can reveal any individual entry x_i without the overhead of revealing the entire vector. At the same time, VCs are also required to be *position binding*: The committer should not be able to reveal any entry of an even maliciously-committed vector to more than a single value.

The main measure of efficiency for VCs, which makes them extremely useful for a variety of applications but highly non-trivial to construct, is their succinctness: Both the size of the commitment and the size of the local openings should be sublinear in the number q of elements in the committed vector. Whereas the classic notion of a Merkle tree [Mer87] can be seen as a VC in which the size of the commitment is independent of q and the size of local openings scales logarithmically with q, Libert and Yung [LY10] and Catalano and Fiore [CF13] presented constructions in which both sizes are independent of q.

Starting already with Merkle's early work, VCs consistently play a key role in a wide range of applications as a communication-efficient method for authenticating rather large amounts of data by allowing users to retrieve small parts of the data alongside short proofs of authenticity. Such applications include, for example, verifiable databases and authenticated data structures (e.g., [NN98, MND$^+$04, BGV11, SvDJ$^+$12, KSS$^+$16, CFG$^+$20]), zero-knowledge sets (e.g., [MRK03, LY10, CRF$^+$11, CHL$^+$13]), cryptographic accumulators [BdM93] (which have many applications on their own right – see for example [BP97, GR97, CL02, DKN$^+$04, Ngu05, ABC$^+$12, MGG$^+$13, FVY14] and the references therein), stateless blockchains (e.g., [STS99, Tod16, But17, BBF19, TAB$^+$20]), and succinct arguments (e.g., [Kil92, Mic94, BBF19, LM19, OWW$^+$20]).

Non-malleable commitments. Another long line of research regarding commitment schemes, initiated by the seminal work of Dolev, Dwork and Naor [DDN00], deals with the construction of *non-malleable* commitments. Roughly speaking, a commitment scheme is non-malleable if an adversary which receives a commitment to some value x, cannot produce a commitment to some "non-trivially related" value x'. Non-malleable commitments have established themselves as instrumental in a host of cryptographic tasks, especially those requiring to protect against man-in-the-middle attacks. Numerous constructions of non-malleable commitments have been suggested over the years, satisfying various flavors of security notions and achieving different efficiency tradeoffs, based on wide range of cryptographic assumptions (e.g., [CIO98, DDN00, FF00, CKO$^+$01, Bar01, CF01, PR05, PR08, PPV08, LPV08, LP09, PW10, Wee10, LP11, GLO$^+$12, GPR16, COS$^+$17, Khu17] and the many references therein).

This work: Non-malleable vector commitments. The fundamental importance of VCs and of non-malleable commitments motivates the study of non-malleable VCs with the premise of significantly strengthening the security and improving the efficiency of the wide range of applications in which they play a key role. For example, non-malleable VCs would directly give rise to verifiable databases, authenticated data structures and cryptographic accumulators offering non-malleability guarantees. As additional, less direct examples, in Section 1.2 we discuss the benefits of using non-malleable VCs as building blocks in the contexts of stateless blockchains and simultaneous multi-round auctions.

However, the notion of non-malleable VCs has not yet been explored, and the existing framework and constructions of standard non-malleable commitments do not take into account the significant security implications of local openings. A closely-related notion, which has been thoroughly explored, is that of non-malleable zero-knowledge sets (ZKS), introduced in the beautiful work of Gennaro and

Micali [GM06] (extending the notion of standard ZKS [MRK03]). Non-malleable ZKS can be seen as a substantial strengthening of non-malleable VCs, supporting *non-membership* proofs in addition to membership proofs. The work of Gennaro and Micali initiated an exciting line of research leading to constructions of non-malleable ZKS based on gradually weaker assumptions and with increasingly better parameters (see [LY10, CF13] and the references therein). However, these constructions rely on the useful yet intricate notion of multi-trapdoor mercurial trapdoor commitments [GM06], specifically tailored to support non-membership proofs (see also [CHL$^+$13, CDV06] for basic background on mercurial commitments). As prominent applications of VCs generally do not require non-membership proofs (as we exemplify in Section 1.2), this raises the following question:

Can non-malleable VCs be constructed within a simplified framework
both **conceptually** *(e.g., simpler and more intuitive notions)*
and **technically** *(e.g., direct and more efficient constructions)?*

1.1 Our Contributions

Notion of non-malleability for VCs. We put forward a strong notion of non-malleability for vector commitment schemes. Our framework strikes a careful balance between the weaker notion of non-malleable non-interactive commitments [CIO98, CKO$^+$01] and the considerably stronger notion of non-malleable zero-knowledge sets [GM06]. Concretely, we generalize the notion of non-malleable non-interactive commitments by incorporating the adversarial adaptivity and additional information resulting from local openings. That is, the key difference from the notion of non-malleable non-interactive commitments is that we aim at achieving non-malleability against adversaries which may have already been exposed to several local openings. Looking ahead, this key difference is the reason that simple attempts of combining VCs and non-malleable commitments do not seem to suffice for realizing our notion (as we demonstrate in Section 3.2).

Warm-up: Merkle trees are non-malleable in the random-oracle model. As a first step within our framework, we examine the non-malleability of existing vector commitments schemes and observe that they are easily malleable (some of them by design in order to support public updates). Then, as a warm-up towards our main result, we show that a Merkle tree does satisfy our requirements when its underlying hash function is modeled as a random oracle [BR93] (and we show that this does not generally hold in the standard model):

Theorem 1.1 (informal)**.** *Let* H *be a hash function and let* $\mathsf{tree}\mathcal{VC}$ *be the Merkle tree vector commitment scheme that is obtained via* H*. Then,* $\mathsf{tree}\mathcal{VC}$ *is a non-malleable vector commitment scheme when* H *is modeled as a random oracle.*

Theorem 1.1 demonstrates the feasibility of realizing our notion of non-malleable vector commitments via a direct construction whose proof is not explicitly based on multi-trapdoor mercurial trapdoor commitments. However, the non-malleability of this construction heavily relies on the random-oracle model and, more importantly, the construction has local openings whose size scales logarithmically with the number q of elements in the committed vector.

Main result: Efficient non-malleable VCs via local equivocability. We present a direct approach for constructing non-malleable VCs whose efficiency essentially matches that of the existing standard VCs. Inspired by constructions of non-malleable zero-knowledge sets [GM06, LY10, CF13] (and, more generally, of non-malleable cryptographic primitives [DDN00]), we show that any vector commitment scheme can be transformed into a non-malleable one, relying on a new primitive that we

put forth. Our new primitive, *locally-equivocable commitments with all-but-one binding*, is evidently both conceptually and technically simpler when compared to multi-trapdoor mercurial trapdoor commitments, as we discuss below. We prove the following theorem:

Theorem 1.2 (informal)**.** *Any vector commitment scheme can be transformed into a non-malleable one using: (1) a locally-equivocable commitment scheme with all-but-one binding, (2) a one-time strongly-unforgeable signature scheme, and (3) a universal one-way hash family.*

We note that our notions of non-malleability and our construction extend to accumulators [BdM93]. Specifically, in our construction, the underlying VC can be replaced with an accumulator, and the underlying locally-equivocable commitment scheme can be replaced with one that supports an a-priori unbounded number of commitments (this is already the case with our number-theoretic constructions).

Intuitive, simple & efficient: Locally-equivocable commitments with all-but-one binding. Our new notion of commitments is obtained by augmenting the standard notion of tag-based commitments with the following two requirements:

- **Local equivocability:** A committer can generate several equivocal commitments with respect to a single common-reference string.
- **All-but-one binding:** Equivocal commitments generated with respect to a predetermined tag τ should be binding with respect to any other tag even when given the trapdoor associated with τ.

This new notion is evidently both conceptually and technically simpler than the notion of multi-trapdoor mercurial trapdoor commitments. From the conceptual perspective, it has a short and intuitive description. This is evident not only from the above informal description, but also from the fact that in addition to the standard setup, commitment and decommitment procedures, our notion consists of only 3 additional procedures, whereas the notion of a multi-trapdoor mercurial trapdoor commitment consists of 7 additional procedures (already in its non-vector variant) together with a non-trivial number of correctness and security requirements.

From the technical perspective, on the one hand we observe that our new notion strengthens Fischlin's notion of identity-based trapdoor commitments [Fis01, Ch. 2.6]; whereas on the other hand we nevertheless show that Fischlin's highly-efficient number-theoretic constructions satisfy our strengthened notion [Fis01, Ch. 3.3]. Specifically, this yields constructions based on the discrete logarithm assumption and on the RSA assumption, in which commitments consist of a *single* group element. This should be contrasted with the known constructions of multi-trapdoor mercurial trapdoor commitments based on the same assumptions in which commitments consist of *two* group elements. The difference between producing one or two group elements might not be significant on its own, but both in our construction and in those based on multi-trapdoor mercurial trapdoor commitments the underlying commitment scheme is used for producing q commitments (where q is the number of elements in the committed vector), and this translates into a more significant difference between producing q and $2q$ group elements.

In addition to these highly-efficient number-theoretic constructions, we also present a construction based on the existence of any standard commitment scheme (and thus based on the existence of any one-way function [Nao91, HIL$^+$99]). However, this construction is mainly of theoretical significance as it supports only an a-priori bounded of number q of equivocal commitments, and the length of its common-reference string is linear in this bound. Such guarantees still suffice for our non-malleable vector commitment, but lead to somewhat impractical efficiency guarantees.

Extension: Non-malleable dynamic VCs. Catalano and Fiore [CF13] constructed VCs in which individual entries of the committed vector can be updated *publicly* (i.e., without knowledge of the committer's private state). Such public updates, however, are inherently incompatible with the motivation underlying the notion of non-malleability, and indeed with our definition of non-malleable VCs. In light of this inherent limitation, we show that our framework and construction can nevertheless support updates in a *private* manner, requiring knowledge of the private state generated by the committer in order to update entries of the underlying vector.

We extend our definition of non-malleable VCs to support dynamic VCs as well, essentially requiring that non-malleability is maintained even when the adversary receives a vector commitment which has undergone adversarially-chosen updates. We then revisit our construction from Theorem 1.2 and show that if the underlying VC supports private updates,[1] then so does our resulting non-malleable VC (which is indeed non-malleable with respect to our extended definition).

Theorem 1.3 (informal)**.** *Any privately-updatable vector commitment scheme can be transformed into a non-malleable privately-updatable one using: (1) a locally-equivocable commitment scheme with all-but-one binding, (2) a strongly-unforgeable signature scheme, and (3) a universal one-way hash family.*

1.2 Applications

The notion of non-malleable commitments is over three decades old [DDN00], and has found a variety of applications. Since our notion of non-malleable VCs strengthens this notion in the non-interactive setting, it can be applied in any case in which non-interactive non-malleable commitments can be used, while offering significant efficiency improvement via local openings. Specifically, VCs play a key role in a wide range of applications both as an intermediate building block and as a direct communication-efficient method for authenticating large amounts of data (allowing users to retrieve small parts of the data alongside short proofs of their authenticity). Here, we focus our attention on discussing the benefits of non-malleable VCs in the contexts of stateless blockchains and simultaneous multi-round auctions.

Stateless blockchains. VCs are used as a direct communication-efficient method for authenticating large amounts of data in stateless blockchains both in the UTXO model (e.g., Bitcoin [Nak08]) and in the account model (e.g., Ethereum [But14]).[2] In both models, transactions and smart-contracts consist of local opening of VCs, where the VCs represent a compressed version of a current state, and are stored by validating parties. Their local openings are verified either as unspent transactions in the UTXO model, or as account balances and various other user-specific properties in the account model (see for example, [BBF19, GRW$^+$20, BBB$^+$18, TAB$^+$20], for extensive discussions and additional related work – which is far beyond the context of our work).

In such scenarios, the basic security properties of VCs are generally insufficient in order to guarantee cross-transaction independence (also known as transaction non-malleability [BCG$^+$14]). Specifically, in such highly interactive scenarios, attackers may indeed observe both VCs and local openings, then manipulate the VCs to represent a malleated state (e.g., either in an implicitly-malicious manner by issuing honest yet tailored transactions that lead to specific state updates, or in an explicitly-malicious manner by potentially controlling to some extent some of the verifying parties), and then produce local openings with respect to the malleated VCs – as captured by our notion

[1] Note that a VC which supports public updates trivially supports private updates.

[2] In fact, in some cases, accumulators are used instead of vector commitments. As noted about, our notions of non-malleability and our construction apply also to accumulators.

for non-malleable VCs. Thus, relying on non-malleable VCs in the context of stateless blockchains can significantly reduce both storage and communication while guaranteeing cross-transaction independence.

Simultaneous multi-round auctions. One of the most classic and direct applications of (non-malleable) commitments is that of sealed-bid auctions [DDN00], and in this context our notion of non-malleable VCs seems particularly suitable for Simultaneous Multi-Round Auctions (SMRA) [Bic17, Ch. 6]. Such auctions provide a widespread multi-round format for selling multiple items. SMRAs were designed for the US Federal Communications Commission in the early 1990s, and since then they have become the standard auction format for selling spectrum worldwide.

SMRAs proceed in rounds, where in each round some or all bidders bid for multiple items, and each item may either be sold or not sold in each round depending of the specific rules of the auction and the submitted bids. After each round is closed the auctioneer discloses which items were won, who wins each of these items, and at what price. Depending on the specific rules of the auction, there are differences in the level of information revealed about other bidders' bids. In some cases all bids are publicly revealed after each round, whereas in other cases only prices of the currently winning bids are publicly revealed.

From the perspective of using vector commitments, submitting each bidder's bids for all available items in each round using a VC, and then publicly revealing local openings for the required (e.g., winning) bids according to the rules of the auction, can lead to significant communication savings (at least in the case of spectrum ranges, the number of ranges may be rather large – although not as large as in the context of using VCs for stateless blockchains). However, this enables a malicious bidder to malleate vector commitments (i.e., bids) provided in earlier rounds or even in the same round after having seen some of their local openings, and to generate a vector commitment (i.e., a bid) to related values together with corresponding local openings at a later stage – as captured by our notion of non-malleable VCs. Thus, relying on non-malleable VCs in the context of SMRAs can significantly reduce communication while guaranteeing cross-round and cross-bid independence.

1.3 Overview of Our Approach

In this section we provide a high-level overview of our notion of non-malleability and of our main construction of a non-malleable vector commitment scheme (Theorem 1.2). For brevity, the main ideas underlying our additional results are described within the corresponding sections.

The starting point of our work is the notion of a vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ [CF13] with the following syntax: The algorithms $\mathsf{VC.Setup}$ and $\mathsf{VC.Commit}$ are invoked in order to produce a common-reference string crs, and in order to produce a commitment vcom for a vector $(x_1, \ldots, x_q)$, respectively. In turn, the algorithms $\mathsf{VC.Open}$ and $\mathsf{VC.Verify}$ are then invoked in order to produce a local opening π_i for each entry $i \in [q]$ of the committed vector, and in order to verify it, respectively. In terms of security, a vector commitment scheme should provide *position binding*, essentially asking that no efficient algorithm can generate a commitment vcom together with two valid openings for the same entry $i \in [q]$ corresponding to different values x_i and x'_i. The main measure of efficiency for vector commitments, which makes them non-trivial to construct, is their succinctness. This is captured by asking for upper bounds on the sizes of the resulting commitments and local openings (e.g., asking that both sizes are nearly independent of the length q of the committed vector). We refer the reader to Section 2.2 for the formal description of the position binding and succinctness requirements.

Our notion of non-malleability. Based on the standard notion of non-malleability for non-interactive commitment schemes [CIO98, CKO+01], any non-malleable vector commitment scheme should at least satisfy the following informal property: An efficient adversary which receives a commitment vcom to a vector $\vec{x} = (x_1, \ldots, x_q)$, should not be able to produce (and then open) a vector commitment $\widehat{\mathsf{vcom}}$ to some vector $\vec{\widehat{x}} = (\widehat{x_1}, \ldots, \widehat{x_q})$ which is "non-trivially related" to $\vec{x}$. However, this property does not capture the adversarial adaptivity and additional information resulting from local openings. Therefore, our notion of non-malleability for vector commitments asks that the above property holds even when the adversary can request local openings for some of the entries of $\vec{x}$ before deciding on $\widehat{\mathsf{vcom}}$, and then open only some of the entries $\vec{\widehat{x}}$ after obtaining local opening for all other entries of $\vec{x}$.

This is formalized by considering a "real" security experiment involving an adversary and an "ideal" security experiment involving a simulator. At a high level, in the real experiment, the adversary is provided with a commitment vcom to a vector $\vec{x} = (x_1, \ldots, x_q)$, and is allowed to request local openings $(\pi_i)_{i \in \mathcal{I}}$ for any subset $\mathcal{I} \subseteq [q]$ of the entries of $\vec{x}$ for producing a commitment $\widehat{\mathsf{vcom}}$. Then, the adversary is provided with local openings for all other entries of $\vec{x}$, and outputs local openings $(\widehat{\pi}_j)_{j \in \mathcal{J}}$ for a subset $\mathcal{J} \subseteq [q]$ of the entries of a malleated vector $(\widehat{x_1}, \ldots, \widehat{x_q})$ (although, note that $\widehat{\mathsf{vcom}}$ is not required to actually correspond to any such malleated vector). In the ideal experiment, the simulator is provided only with a description of the distribution $\mathcal{D}$ from which $\vec{x}$ is sampled (i.e., without the commitment vcom) and the values $(x_i)_{i \in \mathcal{I}}$ (i.e., without the local openings $(\pi_i)_{i \in \mathcal{I}}$), and outputs malleated values $(\widehat{x}_j)_{j \in \mathcal{J}}$.

The outputs of both experiments consist of the values $(x_i)_{i \in [q]}$ and $(\widehat{x}_j)_{j \in [q]}$, where in the real experiment we replace with $\perp$ each value $\widehat{x}_j$ for which either $j \notin \mathcal{J}$ or $\widehat{\pi}_j$ does not properly verify, and in the ideal experiment we replace with $\perp$ each value $\widehat{x}_j$ for which $j \notin \mathcal{J}$. Our notion of non-malleability then asks that for any efficient adversary there exists an efficient simulator such that the outputs of the two experiments are computationally indistinguishable. We refer the reader to Section 3 for our formal definition, and for an in-depth discussion of its various technical aspects (including, the underlying distribution $\mathcal{D}$, the relation between the sets $\mathcal{I}$ and $\mathcal{J}$, and more).

Our main construction. Given any vector commitment scheme $\mathcal{VC}$ we transform it into a non-malleable one as follows. In order to commit to a vector $(x_1, \ldots, x_q)$ we first sample a signing key sk and a corresponding verification key vk for a one-time strongly-unforgeable signature scheme. Then, for each $i \in [q]$ we generate a commitment c_i to the value x_i using a locally-equivocable commitment scheme $\mathcal{LE}$ with all-but-one binding (our newly-introduced primitive augmenting the standard notion of tag-based commitments with two additional requirements). Each of these q commitments is generated with respect to the tag $\tau = h(vk)$ for a universal one-way hash function h. Then, we commit to the vector $(c_1, \ldots, c_q)$ using the underlying vector commitment scheme $\mathcal{VC}$, and output the resulting vector commitment vcom, the verification key vk and a signature σ on vcom using the signing key sk.

In turn, for every $i \in [q]$, a local opening of the value x_i consists of the commitment c_i and its corresponding decommitment d_i, and of a local opening π_i of the commitment c_i with respect to the vector commitment vcom. The verification algorithm first verifies the one-time signature σ, and then verifies the decommitment d_i and the local opening π_i. We refer the reader to Section 5 for a formal description of our construction.

Note that from a foundational perspective, the required building blocks can all be based on the existence of any vector commitment scheme. Specifically, any vector commitment scheme implies the existence of a one-way function, which in turns implies the existence of a locally-equivocable commitment scheme with all-but-one binding (see Section B.1), a one-time strongly unforgeable

signature scheme and a universal one-way hash family. In addition, from a more practical perspective, the above building blocks can all be realized based on a variety of number-theoretic assumptions leading to practical implementations (see, in particular, Sections B.2 and B.3 for practical number-theoretic construction of locally-equivocable commitments with all-but-one Binding).

Focusing on the main measures of efficiency for vector commitments, namely the lengths of resulting commitments and local openings, and the verification time of the local openings, we observe the following:

- A commitment produced by our scheme consists of a commitment produced by the underlying vector commitment scheme, and of a verification key and a signature which can be instantiated with any practical strongly-unforgeable signature scheme[3]. Thus, the length of commitments produced by our scheme is essentially dominated by that of the underlying vector commitment scheme, which can be as short as a single group element.
- A local opening produced by our scheme consists of a local opening produced by the underlying vector commitment scheme together with a commitment and a decommitment produced by the underlying locally-equivocable commitment scheme with all-but-one binding. Relying on existing constructions of vector commitment schemes and on our number-theoretic constructions of locally-equivocable commitment schemes with all-but-one binding (see Section B.2 and B.3), leads to local openings that are essentially as short as three group elements.
- The verification of a local opening produced by our scheme consists of a verification of a local opening produced by the underlying vector commitment scheme, a decommitment of the underlying locally-equivocable commitment scheme with all-but-one binding, and a signature verification. Once again, relying on our number-theoretic constructions of locally-equivocable commitment schemes with all-but-one binding and on practical signature schemes, this is dominated by the verification time of the underlying vector commitment scheme.

Proving the security of our main construction. Recall that for proving the security of our construction, we have to show that for any efficient adversary there exists an efficient simulator for which the outputs of the above-mentioned real and ideal experiments are computationally indistinguishable. Given the informal flavor of the current exposition, we refer the reader to Section 5.2 for an overview of the simulator's description and of the indistinguishably of the two experiments (in addition, of course, to the formal proof of security). For avoiding additional notation and various additional technical details, here we focus only on the adversary's behavior in the real experiment.

Consider an adversary $\mathcal{A}$ that is provided in the real experiment with a commitment vcom to a vector $\vec{x} = (x_1, \ldots, x_q)$. Recall that, in our construction, the commitment vcom is of the form $\mathsf{vcom} = \mathsf{vcom}_0 \| vk \| \sigma$, where vcom_0 is a commitment produced using the underlying vector commitment scheme $\mathcal{VC}$ to the vector of commitments $(c_1, \ldots, c_q)$ produced using the locally-equivocable scheme $\mathcal{LE}$ to $(x_1, \ldots, x_q)$ with respect to the tag $h(vk)$ (for a universal one-way hash function h included in the common-reference string), vk is a verification key for a one-time strongly-unforgeable signature scheme, and σ is a signature on vcom_0 produced using the corresponding signing key. The adversary $\mathcal{A}$ requests local openings $(\pi_i)_{i \in \mathcal{I}}$ for some subset $\mathcal{I} \subseteq [q]$ of the entries of $\vec{x}$, and produces a commitment $\widehat{\mathsf{vcom}} = \widehat{\mathsf{vcom}_0} \| \widehat{vk} \| \widehat{\sigma}$. Then, the adversary is provided with local openings for all other entries of $\vec{x}$, and outputs local openings $(\widehat{\pi}_j)_{j \in \mathcal{J}}$ for a subset $\mathcal{J} \subseteq [q]$. Our proof considers the following three cases (the first and second cases are straightforward, and the third case is the main technical argument):

[3] See, for example, [BSW06, BS07] and the many references therein for a variety of practical strongly-unforgeable signature schemes both in the random-oracle model and in the standard model.

- **Case 1: $\widehat{vk} = vk$.** This case reduces to the one-time strong unforgeability of the signature scheme, unless $\widehat{\mathsf{vcom}_0} = \mathsf{vcom}_0$ or the signature σ does not verify properly (and in these cases our simulator guarantees that the outputs of the real and ideal experiments are identical).

- **Case 2: $\widehat{vk} \neq vk$ but $h(\widehat{vk}) = h(vk)$.** This case reduces to the universal one-wayness of h.

- **Case 3: $h(\widehat{vk}) \neq h(vk)$.** In this case we rely on the position binding of the underlying vector commitment scheme $\mathcal{VC}$, and on the equivocability[4] and all-but-one binding of the locally-equivocable scheme $\mathcal{LE}$. Our main observation is that essentially any advantage that may be obtained in the real experiment must follow from the adversary's ability to choose the values $(\widehat{x_j})_{j\in\mathcal{J}}$ to which it opens the commitment $\widehat{\mathsf{vcom}}$ *after* issuing $\widehat{\mathsf{vcom}}$. That is, any such advantage must follow from the adversary's ability to produce a commitment $\widehat{\mathsf{vcom}}$ and then to provide local openings to more than a single tuple of values $(\widehat{x_j})_{j\in\mathcal{J}}$. These local openings are obtained by relying on the fact that generating $c_1, \ldots, c_q$ using the equivocation algorithms of $\mathcal{LE}$ is indistinguishable from the real experiment and does not bind them to a single tuple of values with respect to the tag $\tau = h(vk)$. Thus, we can rewind the adversary to obtain corresponding local openings with respect to the tag $\widehat{\tau} = h(\widehat{vk})$. But, if $\mathcal{A}$ can open, say, the j-th location of $\widehat{\mathsf{vcom}}$ in two different ways, we show that this contradicts either the position binding of $\mathcal{VC}$ or the all-but-one binding of $\mathcal{LE}$.

1.4 Open Problems

Our framework and constructions lead to various open problems, and here we discuss two such problems focusing on further extending our approach both in the context of vector commitments and in the more general context of non-interactive non-malleable commitments.

Non-malleable subvector commitments. The recent works of Lai and Malavolta [LM19] and of Boneh, Bünz and Fisch [BBF19] introduced the notion of VCs with *subvector openings*. These are VCs which allow the committer to open k entries of the committed vector simultaneously, with a proof whose length is sublinear in k. Our construction, being quite modular, does not seem to support such concise openings, and an interesting open problem is to construct non-malleable VCs that do support subvector openings. A possible starting point may be the recent work Gorbunov et al. [GRW+20], presenting the notion of commitments with aggregatable proofs. Constructing commitments which satisfy both this notion and our notion of local equivocability with all-but-one binding would seem to enable the construction of non-malleable subvector commitments, using our underlying approach for constructing non-malleable VCs.

Implications to non-malleable commitments. Finally, note that any non-malleable vector commitment scheme is also a non-interactive non-malleable commitment scheme (when the vector is of length 1). In that respect, our work presents a general and unified framework for constructing non-interactive non-malleable commitments, capturing both the generic construction of Di Crescenzo, Ishai and Ostrovsky from any one-way function [CIO98] and the efficient number-theoretic constructions of Di Cresccenzo, Katz, Ostrovsky and Smith [CKO+01]. As such, it may enable to construct efficient non-interactive non-malleable commitments based on new assumptions (e.g., isogenies or lattice-based assumptions) by constructing equivocable tag-based commitments with all-but-one binding based on such assumptions.

[4] In our formal proof, we actually rely on the equivocability guarantee earlier in order to enable the simulator to invoke the adversary in the ideal experiment.

1.5 Paper Organization

The remainder of this paper is organized as follows. First, in Section 2 we present the basic notation and standard cryptographic primitives that are used throughout the paper. In Section 3 we present our framework for non-malleable VCs, show that existing VCs do not satisfy our requirements, and demonstrate that simple attempts of combining VCs and non-malleable commitments do not suffice for realizing our notion. In Section 4 we introduce our notion of a locally-equivocable commitment scheme with all-but-one binding, and in Section 5 we present our construction of a non-malleable VC and prove its security. In Section 6 we show that our framework and construction extend to the dynamic setting. In Appendix A we show that a Merkle tree is a non-malleable VC in the random-oracle model, and in Appendix B we present our constructions of locally-equivocable commitment schemes with all-but-one binding.

2 Preliminaries

In this section we present the basic notions and standard cryptographic tools that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for any polynomial $p(\cdot)$ there exists an integer N such that for all $n > N$ it holds that $\nu(n) \leq 1/p(n)$.

2.1 Equivocable Commitment Schemes

We rely on the standard notion of a (non-interactive) equivocable commitment scheme which can be realized based on the existence of any one-way function [Nao91, CIO98]. An equivocable commitment scheme over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ is a 5-tuple $\mathcal{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Commit}, \mathsf{EQ.Decommit}, \mathsf{EQ.Equiv}_1, \mathsf{EQ.Equiv}_2)$ of polynomial-time algorithms defined as follows:

- The algorithm $\mathsf{EQ.Setup}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and outputs a common-reference string crs.
- The algorithm $\mathsf{EQ.Commit}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, an element $x \in \mathcal{X}_\lambda$, and outputs a commitment c and a decommitment d.
- The algorithm $\mathsf{EQ.Decommit}$ is a deterministic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, a commitment c and a decommitment d, and outputs an element $x \in \mathcal{X}_\lambda$ or the rejection symbol $\perp$.
- The algorithm $\mathsf{EQ.Equiv}_1$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, and outputs a common-reference string $\widehat{\mathsf{crs}}$, a commitment $\widehat{c}$ and a state st.
- The algorithm $\mathsf{EQ.Equiv}_2$ is a deterministic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a state st and an element $x \in \mathcal{X}_\lambda$, and outputs a decommitment $\widehat{d}$.

Correctness. We rely on the standard notion of correctness for commitment schemes. That is, for any security parameter $\lambda \in \mathbb{N}$ and for any $x \in \mathcal{X}_\lambda$ it should hold that

$$\Pr\left[\mathsf{EQ.Decommit}(1^\lambda, \mathsf{crs}, c, d) = x\right] = 1,$$

where $\mathsf{crs} \leftarrow \mathsf{EQ.Setup}(1^\lambda)$ and $(c, d) \leftarrow \mathsf{EQ.Commit}(1^\lambda, \mathsf{crs}, x)$, and the probability is taken over the internal randomness of all algorithms.

Equivocability. We rely on the following notion of equivocability [CIO98, CKO+01]:

Definition 2.1. A commitment scheme $\mathcal{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Commit}, \mathsf{EQ.Decommit}, \mathsf{EQ.Equiv}_1, \mathsf{EQ.Equiv}_2)$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is *equivocable* if the following requirements hold:

- **Equivocation correctness:** For any $\lambda \in \mathbb{N}$ and $x \in \mathcal{X}_\lambda$ it holds that

$$\Pr\left[\mathsf{EQ.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, \widehat{c}, \widehat{d}) = x\right] = 1,$$

  where $(\widehat{\mathsf{crs}}, \widehat{c}, \mathsf{st}) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$ and $\widehat{d} := \mathsf{EQ.Equiv}_2(1^\lambda, \mathsf{st}, x)$, and the probability is taken over the internal randomness of all algorithms.
- **Equivocation indistinguishability:** For any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{Equiv}}_{\mathcal{EQ},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Equiv}^{(0)}_{\mathcal{EQ},\mathcal{A}}(\lambda)\right] - \Pr\left[\mathsf{Equiv}^{(1)}_{\mathcal{EQ},\mathcal{A}}(\lambda)\right]\right| \leq \nu(\lambda)$$

  for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ the experiment $\mathsf{Equiv}^{(b)}_{\mathcal{EQ},\mathcal{A}}(\lambda)$ is defined as follows:
  1. $(x, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda)$.
  2. $\mathsf{crs}_0 \leftarrow \mathsf{EQ.Setup}(1^\lambda)$.
  3. $(c_0, d_0) \leftarrow \mathsf{EQ.Commit}(1^\lambda, \mathsf{crs}, x)$.
  4. $(\mathsf{crs}_1, c_1, \mathsf{st}_1) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$.
  5. $d_1 = \mathsf{EQ.Equiv}_2(1^\lambda, \mathsf{st}_1, x)$.
  6. $b' \leftarrow \mathcal{A}(\mathsf{st}_\mathcal{A}, \mathsf{crs}_b, c_b, d_b)$.
  7. Output b'.

Binding. We rely on the standard notion of computational binding for commitment schemes.

Definition 2.2. A commitment scheme $\mathcal{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Commit}, \mathsf{EQ.EQ.Decommit}, \mathsf{EQ.Equiv}_1, \mathsf{EQ.Equiv}_2)$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is *binding* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{Bind}}_{\mathcal{EQA}} \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Bind}_{\mathcal{EQ},\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{PosBind}_{\mathcal{EQ},\mathcal{A}}(\lambda)$ is defined as follows:

1. $\mathsf{crs} \leftarrow \mathsf{EQ.Setup}(1^\lambda)$
2. $(c, (d, x), (d', x')) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs})$.
3. Output 1 if the following conditions hold:
   - $x \neq x'$ and $x, x' \in \mathcal{X}_\lambda$.
   - $\mathsf{EQ.Decommit}(1^\lambda, \mathsf{crs}, c, d) = x$.
   - $\mathsf{EQ.Decommit}(1^\lambda, \mathsf{crs}, c, d') = x'$.

   Otherwise, output 0.

2.2 Vector Commitment Schemes

We follow the notion of a vector commitment scheme as formalized by Libert and Yung [LY10] and Catalano and Fiore [CF13]. As discussed in Section 1.1, we first consider the static setting (i.e., vector commitment schemes without updates), and then extend our approach to the dynamic setting in Section 6.

Definition 2.3. A vector commitment scheme over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is a quadruple $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ of algorithms defined as follows:

- The algorithm $\mathsf{VC.Setup}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and a polynomial $q = q(\lambda)$ and outputs common-reference string crs.
- The algorithm $\mathsf{VC.Commit}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs and a vector $(x_1, \ldots, x_q) \in (\mathcal{X}_\lambda)^q$, and outputs a commitment vcom and a state st.
- The algorithm $\mathsf{VC.Open}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, a commitment vcom, a state st and an index $i \in [q]$, and outputs a proof π.
- The algorithm $\mathsf{VC.Verify}$ is a deterministic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, a commitment vcom, an index $i \in [q]$, an element $x \in \mathcal{X}_\lambda$ and a proof π, and outputs a bit $b \in \{0, 1\}$.

Correctness. A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is correct if for any $\lambda \in \mathbb{N}$, for any polynomial $q = q(\lambda)$, for any vector $(x_1, \ldots, x_q) \in (\mathcal{X}_\lambda)^q$, and for any index $i \in [q]$, it holds that

$$\Pr\left[\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}, i, x_i, \pi\right) = 1\right] = 1,$$

where $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda)$, $(\mathsf{vcom}, \mathsf{st}) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}, (x_1, \ldots, x_q))$ and $\pi \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i)$; and the probability is taken over the randomness of all algorithms.

Security. Catalano and Fiore introduced the following notion of position binding for capturing the security of vector commitment schemes.

Definition 2.4. A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is *position binding* if for any polynomial $q = q(\lambda)$ and for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{A}} \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{PosBind}_{\mathcal{VC},q,\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{PosBind}_{\mathcal{VC},q,\mathcal{A}}(\lambda)$ is defined as follows:

1. $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, q)$
2. $(\mathsf{vcom}, i, x_i, x'_i, \pi, \pi') \leftarrow \mathcal{A}(1^\lambda, q, \mathsf{crs})$.
3. Output 1 if the following conditions hold:
   - $x_i \neq x'_i$.
   - $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}, i, x_i, \pi\right) = 1$.
   - $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}, i, x'_i, \pi'\right) = 1$.

   Otherwise, output 0.

Succinctness. The main measure of efficiency for vector commitments, which makes them non-trivial to construct, is their succinctness. This may be captured by asking for upper bounds $\ell_{\mathsf{Commit}}(\lambda, q)$ and $\ell_{\mathsf{Open}}(\lambda, q)$ on the size of the commitment and the size of the local openings, respectively, as follows.

Definition 2.5. A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is $(\ell_{\mathsf{Commit}}, \ell_{\mathsf{Open}})$-succinct if for any $\lambda \in \mathbb{N}$, for any polynomial $q = q(\lambda)$, for any common-reference string crs produced by $\mathsf{VC.Setup}(1^\lambda, q)$, for any vector $(x_1, \ldots, x_q) \in (\mathcal{X}_\lambda)^q$, and for any commitment and state $(\mathsf{vcom}, \mathsf{st})$ produced by $\mathsf{VC.Commit}(1^\lambda, \mathsf{crs}, (x_1, \ldots, x_q))$ the following two requirements are satisfied:

- The bit-length of vcom is at most $\ell_{\mathsf{Commit}}(\lambda, q)$.
- For any index $i \in [q]$ and for any proof π produced by $\mathsf{VC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i)$, the bit-length of π is at most $\ell_{\mathsf{Open}}(\lambda, q)$.

2.3 One-Time Strongly-Unforgeable Signature Schemes

We rely on the standard notion of a one-time strongly-unforgeable signature scheme, which is known to exist based on the existence of any one-way function [Lam79, NY89, Rom90] (and thus, in particular, based on any of the number-theoretic assumptions that we consider in this paper). A signature scheme is a tuple $\mathcal{SIG} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ of algorithms defined as follows:

- The algorithm $\mathsf{Sig.Gen}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and outputs a pair (sk, vk) of a signing key and a verification key.
- The algorithm $\mathsf{Sig.Sign}$ is a (possibly) probabilistic algorithm that receives as input a signing key sk and a message m and outputs a signature σ.
- The algorithm $\mathsf{Sig.Verify}$ is a deterministic algorithm that receives as input a verification key vk, a message m and a signature σ, and outputs a bit $b \in \{0, 1\}$.

In terms of correctness, the standard requirement for signature schemes asks that

$$\Pr\left[\mathsf{Sig.Verify}_{vk}(m, \mathsf{Sig.Sign}_{sk}(m)) = 1\right] = 1$$

for every $\lambda \in \mathbb{N}$ and for every message m, where $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$, where the probability is taken over the internal randomness of all algorithms. In terms of security, we rely on the following standard notion of one-time strong unforgeability.

Definition 2.6. A signature scheme $\mathcal{SIG} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ is *one-time strongly unforgeable* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{Forge}}_{\mathcal{SIG},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{Forge}_{\mathcal{SIG},\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{Forge}_{\mathcal{SIG},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$.
2. $(m, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, vk)$.
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{A}}, \sigma)$, where $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(m)$.
4. If $\mathsf{Sig.Verify}_{vk}(m^*, \sigma^*)$ and $(m^*, \sigma^*) \neq (m, \sigma)$ then output 1 and otherwise output 0.

2.4 Universal One-Way Hash Functions

We rely on the standard notion of universal one-way hash functions, which is known to exist based on the existence of any one-way function [NY89, Rom90] (and thus, in particular, based on any of the number-theoretic assumptions that we consider in this paper). A hash family from domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ to range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ is a collection $\mathcal{H} = \{\mathcal{H}\}_{\lambda\in\mathbb{N}}$ where each $\mathcal{H}_\lambda$ consists of functions $h : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. For simplifying our notation we let $h \leftarrow \mathcal{H}_\lambda$ denote the process of sampling a function h from $\mathcal{H}_\lambda$ without explicitly describing a sampling algorithm, where h denotes both the description of the sampled function and its evaluation algorithm.

Definition 2.7. A hash family $\mathcal{H}$ from domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ to range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ is a *universal one-way hash family* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{UOWHF}_{\mathcal{H},\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{UOWHF}_{\mathcal{H},\mathcal{A}}(\lambda)$ is defined as follows:

1. $(x, \mathsf{st}) \leftarrow \mathcal{A}(1^\lambda)$.
2. $h \leftarrow \mathcal{H}_\lambda$.
3. $x' \leftarrow \mathcal{A}(\mathsf{st}, h)$.
4. If $x \neq x'$ and $h(x) = h(x')$ then output 1, and otherwise output 0.

3 Non-Malleable Vector Commitments

In this section we begin by presenting our notion of non-malleability for vector commitment schemes. Then, in Section 3.1 we show that existing vector commitment schemes do not satisfy it (some of them by design in order to support public updates). As mentioned in Section 1.1, the key difference from the standard notion of non-malleable non-interactive commitments is that we aim at achieving non-malleability even with respect to adversaries which have already been exposed to several local openings. This key difference is the reason that simple attempts of combining VCs and non-malleable commitments, that we discuss in Section 3.2, do not suffice for realizing our new notion.

Loosely speaking, a vector commitment scheme is non-malleable if an efficient adversary which receives a vector commitment vcom to a vector $\vec{x} = (x_1, \ldots, x_q)$, cannot produce (and open) a vector commitment $\widehat{\mathsf{vcom}}$ to some vector $\vec{\widehat{x}} = (\widehat{x_1}, \ldots, \widehat{x_q})$ which is "non-trivially related" to $\vec{x}$. This property should hold even when the adversary can request local openings for some of the entries of $\vec{x}$ before deciding on $\widehat{\mathsf{vcom}}$; and open only some of the entries $\vec{\widehat{x}}$. Definition 3.1 below uses the term "valid distribution" which is formally clarified following the definition. As discussed in Section 1.1, we start by considering the static setting of vector commitments without updates, and then, in Section 6, we extend our approach to the dynamic setting.

Definition 3.1. A vector commitment $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is *non-malleable* if for any polynomially-bounded integer $q = q(\lambda)$ and for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exist a probabilistic polynomial-time algorithm $\mathcal{S}$ such that the following holds:
For any probabilistic polynomial-time algorithm $\mathcal{R}$ and for any valid distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ over $\{(\mathcal{X}_\lambda)^q\}_{\lambda\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{NM}}_{\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)\right) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiments $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and $\mathsf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)$ are defined as follows:

The Experiment $\mathbf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$:

1. $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, q)$.
2. $(x_1, \ldots, x_q) \leftarrow \mathcal{D}_\lambda$.
3. $(\mathsf{vcom}, \mathsf{st}) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}, (x_1, \ldots, x_q))$.
4. $(\mathcal{I}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \mathsf{vcom})$ where $\mathcal{I} \subseteq [q]$.
5. $\pi_i \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i)$ for each $i \in [q]$.
6. $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_\mathcal{A}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_\mathcal{A}, (x_i)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$, where $\mathcal{J} \subseteq [q]$.
7. $((\widehat{x_j})_{j\in\mathcal{J}}, (\widehat{\pi_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{A}\left(\mathsf{st}_\mathcal{A}, (x_i)_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$, where $\overline{\mathcal{I}} = [q] \setminus \mathcal{I}$.
8. If $\widehat{\mathsf{vcom}} = \mathsf{vcom}$ or if $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}\right) = 0$ for some $j \in \mathcal{J}$, then output $((x_1, \ldots, x_q), (\bot)^q, \mathcal{I})$.
   Otherwise, output $((x_1, \ldots, x_q), (\widehat{x_1}, \ldots, \widehat{x_q}), \mathcal{I})$, where $\widehat{x_j} = \bot$ for each $j \in [q] \setminus \mathcal{J}$.

The Experiment $\mathbf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)$:

1. $(x_1, \ldots, x_q) \leftarrow \mathcal{D}_\lambda$.
2. $(\mathcal{I}, \mathsf{st}_\mathcal{S}) \leftarrow \mathcal{S}(1^\lambda, \mathcal{D})$.
3. $(\mathcal{J}, (\widehat{x_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{S}(\mathsf{st}_\mathcal{S}, (x_i)_{i\in\mathcal{I}})$.
4. Output $((x_1, \ldots, x_q), (\widehat{x_1}, \ldots, \widehat{x_q}), \mathcal{I})$ where $\widehat{x_i} = \bot$ for every $i \in [q] \setminus \mathcal{J}$.

Succinctness. Recall that the main measure of efficiency for vector commitments, which makes them non-trivial to construct, is their succinctness: Both the size of the commitment and the size of the local openings should be sublinear in the number q of elements in the committed vector. That is, the standard notion of vector commitments does not require any hiding guarantees [CF13], and thus can be trivially satisfied if succinctness is not required (in this case a vector commitment scheme can simply output the vector itself). When additionally requiring a vector commitment scheme to hide all entries of the committed vector for which local openings were not provided, the task becomes non-trivial even when succinctness is not required (since this introduces a selective decommitment problem whenever an attacker can request local openings after having seen the commitment).

Our notion of non-malleability implies, in particular, such a hiding guarantee, and is therefore non-trivial to realize even when succinctness is not required. Nevertheless, as discussed in Section 1.1, the non-malleable vector commitments resulting from our transformation are essentially as succinct as the existing standard vector commitments that do not require any hiding guarantees.

Valid distributions. Definition 3.1 considers *valid* distributions, and here we formally define this notion. On the face of it, one can hope to consider all distributions that are samplable in polynomial time. However, exactly as in case of non-malleable zero-knowledge sets [GM06], our notion of non-malleable vector commitments faces a "selective decommitment" problem (since it considers attackers which may be exposed to several adaptively-chosen local openings). One approach to overcome this difficulty, which is the approach that we follow in this work, is to restrict our attention to considering the natural subclass of all efficiently samplable distributions that was considered by Gennaro and Micali [GM06]). This subclass consists of all distributions that are not only efficiently samplable, but also all of their marginal distributions are efficiently samplable.

That is, we say that a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ over $\{(\mathcal{X}_\lambda)^q\}_{\lambda\in\mathbb{N}}$ is valid if the following holds: For every $\lambda \in \mathbb{N}$, for every $\vec{x} = (x_1, \ldots, x_{q(\lambda)})$ in the support of $\mathcal{D}_\lambda$, and for every subset $\mathcal{I} = (i_1, \ldots, i_{|\mathcal{I}|}) \subseteq [q(\lambda)]$, it is possible to efficiently sample a vector $\vec{y}$ from the conditional distribution $\mathcal{D}_\lambda|(\forall i \in \mathcal{I} : y_i = x_i)$. We denote the process of sampling the entries of $\vec{y}$ in $\overline{\mathcal{I}} = [q] \setminus \mathcal{I}$ by

$(y_j)_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}|(\mathcal{I},(x_i)_{i\in\mathcal{I}})$. Note that this requirement is fairly reasonable, and in particular, it is satisfied by any product distribution $\mathcal{D}$ over $(\mathcal{X}_\lambda)^q$.

An alternative approach, as pointed out by Gennaro and Micali, is to rely on an underlying commitment scheme that provides a certain form of security against selective decommitment attacks. In their context, it seems that the underlying commitment scheme would have to be at least both mercurial and provide security against selective decommitment attacks (realizing this alternative approach for non-malleable zero-knowledge sets still remains an interesting open problem). Similarly, in our context it would have to be at least locally equivocable with all-but-one binding (as we define in Section 4) and provide security against selective decommitment attacks. We leave the exploration of this alternative approach as an avenue for further research.

$\mathcal{J}$ cannot be chosen later. Note that we allow the adversary $\mathcal{A}$ in the experiment $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ to choose the subset $\mathcal{J}$ at the latest stage possible. This is true because had we let A choose $\mathcal{J}$ in Step 7 of the experiment, then $\mathcal{A}$ could have encoded information about $(x_j)_{j\in\overline{\mathcal{I}}}$ within their choice of $\mathcal{J}$. For example, assume that we let the adversary choose $\mathcal{J}$ in Step 7 of $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ (after observing $(x_j)_{j\in\overline{\mathcal{I}}}$), and consider an adversary which chooses $\mathcal{J}$ to be of size 1 if the parity of the bit-description of $x_{j_1}\|\cdots\|x_{j_{|\overline{\mathcal{I}}|}}$ is 1, and chooses $\mathcal{J}$ to be of size 0 if this parity is 0, where $\overline{\mathcal{I}} = \{j_1,\ldots,j_{|\overline{\mathcal{I}}|}\}$. Of course, this cannot be simulated, since the simulator never gets access to $x_{j_1},\ldots,x_{j_{|\overline{\mathcal{I}}|}}$.

Invalid openings. Whenever the adversary $\mathcal{A}$ provides an invalid opening for *any* index in $\mathcal{J}$, then the output of the real experiment is set to be of the form $((x_1,\ldots,x_q),(\perp)^q,\mathcal{I})$. We argue that this choice is indeed a necessary one. To see why that is the case, consider the following alternative (and faulty) approach: For all $j\in\mathcal{J}$ for which $\mathcal{A}$ provides invalid openings set $\widehat{x_j}=\perp$, but for all indices for which $\mathcal{A}$ provides valid openings, keep the $\widehat{x_j}$'s in the output of the experiment as is (that is, as outputted by $\mathcal{A}$ in Step 7). The problem with this approach is that it effectively gives $\mathcal{A}$ the power to choose $\mathcal{J}$ in Step 7 of the experiment, for example by outputting $\mathcal{J}=[q]$ in Step 6 and then providing valid openings for a different set $\mathcal{J}'\subsetneq[q]$ in Step 7. As explained above, such a definition cannot be satisfied, as it allows $\mathcal{A}$ to encode information about $(x_j)_{j\in\overline{\mathcal{I}}}$ via the set of validly-opened positions.

Letting $\mathcal{J}$ intersect $\mathcal{I}$. At first glance, it might seem uncanny that we let the adversary choose the set $\mathcal{J}$ such that it includes locations for which the adversary has seen openings before producing $\widehat{\mathsf{vcom}}$ (i.e., it intersects $\mathcal{I}$). On the face of it, this allows for trivial attacks, since the adversary can trivially commit, via $\widehat{\mathsf{vcom}}$, to values that are related to $(x_i)_{i\in\mathcal{I}}$. However, Definition 3.1 "discounts" such trivial attacks from the adversary's advantage, by allowing the simulator to access values $(x_i)_{i\in\mathcal{I}}$ as well.

Choosing $\mathcal{I}$ adaptively. We note that Definition 3.1 can be strengthened, by allowing the adversary in $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ to choose the set $\mathcal{I}$ in an adaptive manner. That is, to choose the indices included in $\mathcal{I}$ one by one, each index being chosen after $\mathcal{A}$ has observed the values x_i (and the associated proof π_i) for each previous chosen index i. Our construction in Section 5 remains secure under this strengthened definition, and its proof of security readily extends to it.

Reusability. One might consider a strengthening of Definition 3.1, by providing the adversary with many vector commitments $\mathsf{vcom}_1,\ldots,\mathsf{vcom}_k$ (and to local openings of their choice) to vectors $\vec{v_1},\ldots,\vec{v_k}$, and requiring that they cannot produce (and later open) a vector commitment $\widehat{\mathsf{vcom}}$ to a

vector $\vec{v}$ which is non-trivially related to $\vec{v_1}, \ldots, \vec{v_k}$. Such a strengthening is in line with the notion of a *reusable* non-malleable non-interactive commitment scheme [DG03] and more generally, with the notion of concurrent non-malleable commitments [DDN00]. We believe that our framework and constructions can be generalized to support such a definition, and we leave this task to future work.

3.1 Existing Schemes Do Not Satisfy Our Notion

Merkle trees in the standard model. Consider the Merkle tree construction of vector commitments with respect to a hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. That is, a commitment vcom to a vector $\vec{x} \in \{0,1\}^{\lambda \times q}$ is the root of the binary hash tree whose left leaves (i.e., leaves which are left children) are the values of $\vec{x}$; the right leaves are assigned some predetermined arbitrary values; and the value of each node is obtained by applying h to the concatenation of its children.[5] In Appendix A we present a formal description of this construction, and show that if h is modeled as a random oracle, then this construction is indeed non-malleable per Definition 3.1. Alas, if h is instantiated via a standard-model collision resistant hash function, this is not necessarily the case. Loosely speaking, this is because the function h itself may be malleable.

As a concrete and simple example, consider the case in which $h(z) = z_1 \| h'(z_2 \| \cdots \| z_{2\lambda})$, where $z = z_1 \| \cdots \| z_{2\lambda} \in \{0,1\}^{2\lambda}$ and $h' : \{0,1\}^{2\lambda-1} \to \{0,1\}^{\lambda-1}$ is a collision-resistant hash function. It is not hard to verify that h is also collision resistant; but still, the vector commitment it induces is malleable. In fact, this vector commitment is not even completely hiding: Consider the following attacker which first request to see an opening of the first entry x_1 of $\vec{x}$ (by outputting $\mathcal{I} = \{1\}$ in Step 4 of the real experiment of Definition 3.1). This opening includes the value assigned to the sibling of the parent of x_1 (which is the parent of x_2); denote this value by $y = y_1 \| \cdots \| y_\lambda \in \{0,1\}^{\lambda}$. Then y_1 is equal to the first bit of x_2. This means that the adversary can commit from scratch to some vector $(\widehat{x_1}, \ldots, \widehat{x_q})$ such that the first bit of $\widehat{x_2}$ is also y_1 (and the other entries are chosen arbitrarily), satisfying a non-trivial relation with $\vec{x}$. This is just one simple example, and many more examples exist for the malleability of standard-model instantiation of Merkle trees.

Algebraic constructions. More recent algebraic constructions of vector commitments turn out to be malleable as well. To start, consider the constructions of Catalano and Fiore [CF13], based on either the discrete logarithm assumption or the RSA assumption. In both of these construction, a user commits to a vector $\vec{x}$ of integers, by computing $\mathsf{vcom} = \prod_{i \in [q]} g_i^{x_i}$, where $g_1, \ldots, g_q$ are publicly-known group elements. It is not hard to see, that an attacker receiving vcom can produce a commitment $\widehat{\mathsf{vcom}}$ to any affinely-related vector $a \cdot \vec{x} + \vec{z}$, by computing $\mathsf{vcom}^a \cdot \prod_{i \in [q]} g_i^{z_i}$.

Lai and Malavolta [LM19] recently generalized the constructions of Catalano and Fiore to Euclidean rings (they also presented an additional construction in bilinear groups, which falls into the same template as the constructions of Catalano and Fiore, and hence the same attack applies to it). Concretely, they consider a module over a ring R, consisting of an Abelian group $(\mathbb{G}, \times)$ and a binary operation $\circ : R \times \mathbb{G} \to \mathbb{G}$. A vector commitment to a vector $\vec{x} \in \mathcal{X}^q$ is then computed by the inner product $\left\langle \vec{x}, \vec{S} \right\rangle = (x_1 \circ S_1) \times \cdots \times (x_q \circ S_q)$, where $\mathcal{X} \subseteq R$ is a subset satisfying some natural property and $\vec{S}$ is a vector of publicly-known group elements. Unsurprisingly, the afore-described attack easily generalizes to this construction as well. For any $a \in R$ and $z \in R^q$, an attacker which receies a commitment vcom to a vector $\vec{x} \in \mathcal{X}^q$ can compute a commitment to any affinely-related vector $a \cdot \vec{x} + \vec{z}$, where $(+, \cdot)$ are the two ring operations, by computing $(a \circ \mathsf{vcom}) \times \left\langle \vec{z}, \vec{S} \right\rangle$. Note that this attack works as long as $a \cdot \vec{x} + \vec{z}$ lies in $\mathcal{X}$.

[5] We embed the entries of $\vec{x}$ only as left leaves as to avoid trivial attacks. Doing so, the opening of say, the i-th entry does not trivially reveal any other entries.

3.2 Simple Attempts That Fail

For obtaining an initial understanding of the challenges in constructing non-malleable vector commitments, consider the following two constructions which are based on rather simple and direct combinations of vector commitments and non-malleable commitments, and fail to satisfy Definition 3.1. In what follows, $\mathsf{nm}\mathcal{COM}$ is a standard non-malleable commitments scheme and $\mathcal{VC}$ is a (potentially malleable) vector commitment scheme.

Applying $\mathsf{nm}\mathcal{COM}$ and then $\mathcal{VC}$. As a first attempt, consider what happens when in order to commit to some vector $\vec{x}$, one first applies $\mathsf{nm}\mathcal{COM}$ locally to each entry of $\vec{x}$ to obtain q commitments $c_1, \ldots, c_q$; and then uses $\mathcal{VC}$ to commit to these commitments. The problem with this approach is that $\mathcal{VC}$ might be malleable. For example, if $\mathcal{VC}$ appends a random bit to the end of each commitment, then an adversary which receives a commitment vcom to a vector $\vec{x}$ produced using the approach described above, can easily produce a different commitment $\widehat{\mathsf{vcom}}$ to the same $\vec{x}$ by flipping the last bit of vcom. It might be also the case that $\mathcal{VC}$ is malleable in the following sense: Given a commitment vcom to a vector $\vec{x}$ produced using $\mathcal{VC}$, it is easy to "replace" some of the entries of the vector underlying vcom, resulting in a commitment to a related vector $\vec{x'}$ which identifies with $\vec{x}$ on some of its locations. If this is the case, then such an attack is also possible for the combined vector commitment scheme which first applies $\mathsf{nm}\mathcal{COM}$ locally.[6]

Applying $\mathcal{VC}$ and then $\mathsf{nm}\mathcal{COM}$. Consider a construction which, in order to commit to a vector $\vec{x}$, first applies $\mathcal{VC}$ to produce a commitment vcom_0 and commits to vcom_0 using $\mathsf{nm}\mathcal{COM}$ to produce a commitment vcom. Alas, this approach also does not meet Definition 3.1. The main issue is unique to the setting of non-malleable vector commitments: Per Definition 3.1, an adversary can request to see openings of individual entries of $\vec{x}$ before outputting their own commitment $\widehat{\mathsf{vcom}}$. These openings must include in particular the intermediate commitment vcom_0. Hence, if $\mathcal{VC}$ is malleable, then the adversary, having observed vcom_0 can come up with a different commitment $\widehat{\mathsf{vcom}_0}$ with respect to $\mathcal{VC}$ for some related vector $\vec{x'}$. Then, the adversary can simply commit to $\widehat{\mathsf{vcom}_0}$ using $\mathsf{nm}\mathcal{COM}$ to produce the desired commitment $\widehat{\mathsf{vcom}}$.

4 Locally-Equivocable Commitments with All-But-One Binding

In this section we introduce our notion of a locally-equivocable commitment scheme with all-but-one binding, which serves as one of the main building-blocks underlying our construction of a non-malleable vector commitment scheme. Our notion is obtained by augmenting the standard notion of a non-interactive tag-based commitment scheme with two additional requirements, namely local equivocability and all-but-one binding.

In addition, we present both a somewhat theoretical realization of the our new notion based on the existence of any one-way function, and two efficient number-theoretic realizations: A construction based on the discrete logarithm assumption, and a construction based on the RSA assumption. In both cases, the common-reference string consists of 2-3 group elements (in addition to the description of the group), and a commitment consists of a single group element. As discussed in Section 1.1, these number-theoretic constructions were described by Fischlin in his Ph.D. thesis [Fis01] (and also

[6] Another issue which may arise, is that $\mathsf{nm}\mathcal{COM}$ might not be *concurrent* non-malleable (see, for example, [DDN00, PR05, PR08, LPV08] and the references therein). In this case, an adversary which observes some of the local commitments and openings produced via $\mathsf{nm}\mathcal{COM}$ may be able to come up with $\mathsf{nm}\mathcal{COM}$ commitments to related values. This issue, however, can be relatively easily resolved by using a commitment scheme which offers non-malleability even against adversaries which observe at most q commitments and openings.

used by Crescenzo, Katz, Ostrovsky and Smith [CKO+01] in their number-theoretic constructions of non-malleable non-interactive commitment schemes[7]). Although our notion of a locally-equivocable commitment scheme with all-but-one binding strengthens Fischlin's notion of identity-based trapdoor commitments (as we discuss below), we nevertheless show that these constructions satisfy our notion. Our constructions are provided in Appendix B.

Formally, a locally-equivocable commitment scheme with all-but-one binding over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$ is a 6-tuple $\mathcal{LE} = (\mathsf{LE.Setup}, \mathsf{LE.Commit}, \mathsf{LE.Decommit}, \mathsf{LE.AltSetup}, \mathsf{LE.Equiv}_1, \mathsf{LE.Equiv}_2)$ of polynomial-time algorithms defined as follows:

- The algorithm $\mathsf{LE.Setup}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and a polynomially-bounded integer $q = q(\lambda)$, and outputs a common-reference string crs.
- The algorithm $\mathsf{LE.Commit}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, an element $x \in \mathcal{X}_\lambda$, an index $i \in [q]$ and a tag $\tau \in \mathcal{T}_\lambda$, and outputs a commitment c and a decommitment d.[8]
- The algorithm $\mathsf{LE.Decommit}$ is a deterministic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, a commitment c, a decommitment d, an index $i \in [q]$ and a tag $\tau \in \mathcal{T}_\lambda$, and outputs an element $x \in \mathcal{X}_\lambda$ or the rejection symbol $\perp$.
- The algorithm $\mathsf{LE.AltSetup}$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and a polynomially-bounded integer $q = q(\lambda)$, and outputs a state st_0.
- The algorithm $\mathsf{LE.Equiv}_1$ is a probabilistic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ a state st_0, a polynomially-bounded integer $q = q(\lambda)$ and a tag $\tau \in \mathcal{T}_\lambda$, and outputs a common-reference string $\widehat{\mathsf{crs}}$, commitments $\widehat{c_1}, \ldots, \widehat{c_q}$ and a state st_1.
- The algorithm $\mathsf{LE.Equiv}_2$ is a deterministic algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, an element $x \in \mathcal{X}_\lambda$, an index $i \in [q]$, a state st_1 and a tag $\tau \in \mathcal{T}_\lambda$, and outputs a decommitment $\widehat{d}$.

A commitment scheme as described above should satisfy the standard correctness requirement of commitment schemes. That is, for any security parameter $\lambda \in \mathbb{N}$, for any tag $\tau \in \mathcal{T}_\lambda$, for any polynomially-bounded $q = q(\lambda)$, for any $i \in [q]$ and for any $x \in \mathcal{X}_\lambda$ it holds that

$$\Pr\left[\mathsf{LE.Decommit}(1^\lambda, \mathsf{crs}, c, d, i, \tau) = x\right] = 1,$$

where $\mathsf{crs} \leftarrow \mathsf{LE.Setup}(1^\lambda, q)$ and $(c, d) \leftarrow \mathsf{LE.Commit}(1^\lambda, \mathsf{crs}, x, i, \tau)$, and the probability is taken over the internal randomness of all algorithms.

The following two definitions formally capture our local equivocability and all-but-one binding requirements.

Definition 4.1 (Local equivocability)**.** A commitment scheme $\mathcal{LE} = (\mathsf{LE.Setup}, \mathsf{LE.Commit}, \mathsf{LE.Decommit}, \mathsf{LE.AltSetup}, \mathsf{LE.Equiv}_1, \mathsf{LE.Equiv}_2)$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$ is *locally equivocable* if the following requirements hold:

[7] Although Crescenzo et al. did not explicitly frame their construction as relying on an underlying equivocable commitment scheme, we follow a somewhat more fine-grained abstraction via our local equivocability and all-but-one binding properties.

[8] We note that the commitment and decommitment algorithms $\mathsf{LE.Commit}$ and $\mathsf{LE.Decommit}$ receive the index $i \in [q]$ as input for technical reasons that come up in our generic construction based on one-way functions (Appendix B).

- **Equivocation correctness:** For any $\lambda \in \mathbb{N}$, for any $\tau \in \mathcal{T}_\lambda$, for any polynomially-bounded $q = q(\lambda)$, for any $i \in [q]$ and for any $x \in \mathcal{X}_\lambda$ it holds that

$$\Pr\left[\mathsf{LE.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, \widehat{c_i}, \widehat{d}, i, \tau) = x\right] = 1,$$

  where $(\widehat{\mathsf{crs}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{LE.AltSetup}(1^\lambda), q, \tau)$ and $\widehat{d} = \mathsf{LE.Equiv}_2(1^\lambda, x, i, \mathsf{st}_1)$, and the probability is taken over the internal randomness of all algorithms.
- **Equivocation indistinguishability:** For any probabilistic polynomial-time algorithm $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$ such that for any polynomially bounded $q = q(\lambda)$ it holds that

$$\mathbf{Adv}^{\mathsf{LocalEquiv}}_{\mathcal{LE},q,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} |\Pr\left[\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{A},0}(\lambda)\right] - \Pr\left[\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{A},1}(\lambda)\right]| \leq \nu(\lambda)$$

  for all sufficiently large $\lambda \in \mathbb{N}$, where for any bit $b \in \{0,1\}$ the experiment $\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{A},b}(\lambda)$ is defined as follows:
  1. $(\tau, x_1, \ldots, x_q, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda)$.
  2. $\mathsf{crs}_0 \leftarrow \mathsf{LE.Setup}(1^\lambda, q)$.
  3. $(c_{0,i}, d_{0,i}) \leftarrow \mathsf{LE.Commit}(1^\lambda, \mathsf{crs}, x_i, i, \tau)$ for each $i \in [q]$.
  4. $\mathsf{st}_0 \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$.
  5. $(\mathsf{crs}_1, c_{1,1}, \ldots, c_{1,q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_0, q, \tau)$.
  6. $d_{1,i} = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1)$ for each $i \in [q]$.
  7. $b' \leftarrow \mathcal{A}(\mathsf{st}_\mathcal{A}, \mathsf{crs}_b, (c_{b,i})_{i\in[q]}, (d_{b,i})_{i\in[q]})$.
  8. Output b'.

Intuitively, the all-but-one binding property requires that an adversary which generates equivocable public parameters (via the $\mathsf{LE.Equiv}_1$ algorithm) using a tag τ of their choice, cannot break the binding property with respect to these parameters and a different tag $\tau' \neq \tau$.

Definition 4.2 (All-but-one binding)**.** A commitment scheme $\mathcal{LE} = (\mathsf{LE.Setup}, \mathsf{LE.Commit}, \mathsf{LE.Decommit}, \mathsf{LE.AltSetup}, \mathsf{LE.Equiv}_1, \mathsf{LE.Equiv}_2)$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$ is *all-but-one binding* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that for polynomially-bounded $q = q(\lambda)$ it holds that

$$\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \Pr\left[\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{A}}(\lambda) = 1\right] \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiment $\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{A}}(\lambda)$ is defined as follows:

1. $(\tau, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda)$, where $\tau \in \mathcal{T}_\lambda$.
2. $\mathsf{st}_0 \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$.
3. $\rho \leftarrow \{0,1\}^r$, where $r = r(\lambda)$ is the number of random coins used by $\mathsf{LE.Equiv}_1$ on security parameter $\lambda \in \mathbb{N}$.
4. $(\widehat{\mathsf{crs}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) = \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_0, q, \tau; \rho)$.
5. $(c, d, d', i, \tau') \leftarrow \mathcal{A}(\mathsf{st}_\mathcal{A}, \mathsf{st}_0, \rho)$.
6. $x = \mathsf{LE.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, c, d, i, \tau')$ and $x' = \mathsf{LE.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, c, d', i, \tau')$.
7. Output 1 if $\tau' \neq \tau$, $x \neq \bot$, $x' \neq \bot$ and $x \neq x'$. Otherwise, output 0.

Comparing our notion to identity-based and simulation-sound trapdoor commitments. Having formally defined our notion of a locally-equivocable commitment scheme with all-but-one binding, we can now compare it to Fischlin's notion of an identity-based trapdoor commitment scheme [Fis01, Ch. 2.6]. Both notions are obtained by augmenting the standard notion of a non-interactive tag-based commitment scheme with equivocability and all-but-one binding requirements. Our requirements, however, are more strict compared to those of Fischlin, both in terms of equivocability and in terms of all-but-one binding.

First, in terms of equivocability, Fischlin asks for an equivocation algorithm that produces an equivocable common-reference string and a *single* equivocable commitment which should be indistinguishable from an honestly-generated common-reference string and an honestly-generated commitment. However, for our construction of a non-malleable vector commitment scheme, producing a single equivocable commitment seems insufficient. Thus, we ask for an equivocation algorithm that produces an equivocable common-reference string and q equivocable commitments (where $q = q(\lambda)$ is any predetermined polynomial) which should be indistinguishable from an honestly-generated common-reference string and an honestly-generated vector of q independent commitments. We note that such a requirement does not necessarily follow from the case $q = 1$ due to potential dependencies between the equivocable common-reference string and the single equivocable commitment that may be efficiently identifiable when producing more than a single equivocable commitment (this is evident in our generic construction based any non-interactive equivocable commitment scheme, where the common-reference string grows with q).

Second, in terms of all-but-one binding, Fischlin asks that when generating an equivocable common-reference string with respect to a predetermined tag τ, commitments with respect to all other tags should still be binding even when given the trapdoor associated with τ. For our construction we strengthen this requirements, and ask that commitments with respect to all other tags should still be binding even when given the trapdoor associated with τ and the internal randomness of the equivocation algorithm.

An additional related notion is that of a simulation-sound trapdoor commitment scheme, put forth by Garay, MacKenzie, and Yang [GMY03], which can be seen as augmenting standard trapdoor commitments [Rey01, Ch. A.5] with tags. Garay et al. also considered an enhanced binding property, requiring that binding with respect to a tag τ should be preserved, even if the attacker can obtain a single "fake" opening (using the trapdoor) for any commitment with respect to τ, as well as an unbounded number of openings for any commitment with respect to any other tag $\tau' \neq \tau$. This notion seems to be incomparable to our notion of locally-equivocable commitments with all-but-one binding. First, the trapdoor in simulation-sound trapdoor commitments is a global trapdoor generated by the honest parameters generation algorithm. There are no alternative procedures to generate equivocable parameters and commitments, and the trapdoor is not tied to any particular tag. This means that knowledge of the trapdoor allows one to open any (honestly generated) commitment to any value they desires. Second, whereas in our enhanced binding property the attacker receives the trapdoor associated with a tag τ of their choice, the attacker in the notion of Garay et al. does not receive the trapdoor, but only openings computed using it (this is unavoidable, since knowledge of the trapdoor in their notion allows the attacker to break binding with respect to all tags).

5 Our Construction of a Non-Malleable Vector Commitment Scheme

In this section we present our main construction (Section 5.1) and then prove its security (Section 5.2).

5.1 The Construction

Our construction relies on the following building blocks:

- A vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ (see Section 2.2).[9]
- A locally-equivocable commitment scheme with all-but-one binding $\mathcal{LE} = (\mathsf{LE.Setup}, \mathsf{LE.Commit}, \mathsf{LE.Decommit}, \mathsf{LE.AltSetup}, \mathsf{LE.Equiv}_1, \mathsf{LE.Equiv}_2)$ over the domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$ (see Section 4) with tags of length $t = t(\lambda)$ bits.
- A one-time strongly-unforgeable signature scheme $\mathcal{SIG} = (\mathsf{Sig.Gen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ (see Section 2.3). Let $v = v(\lambda)$ denote the bit-length of the verification keys that are produced by $\mathsf{Sig.Gen}(1^\lambda)$.
- A universal one-way hash family $\mathcal{H} = \{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ (see Section 2.4), where each $\mathcal{H}_\lambda$ consists of functions mapping $v(\lambda)$-bit strings to $t(\lambda)$-bit strings for every security parameter $\lambda \in \mathbb{N}$.

As discussed in Section 1.3, from a foundational perspective, the above building blocks can all be based on the existence of any vector commitment scheme. Additional, from a more practical perspective, the above building blocks can all be realized based on a variety of number-theoretic assumptions leading to practical implementations.

Given the above building blocks, our construction of a non-malleable vector commitment scheme, denoted $\mathsf{nm}\mathcal{VC} = (\mathsf{nmVC.Setup}, \mathsf{nmVC.Commit}, \mathsf{nmVC.Open}, \mathsf{nmVC.Verify})$, is defined as follows:

A non-malleable vector commitment scheme $\mathsf{nm}\mathcal{VC}$

$\mathsf{nmVC.Setup}(1^\lambda, q)$:

1. Sample $\mathsf{crs}_{\mathsf{LE}} \leftarrow \mathsf{LE.Setup}(1^\lambda, q)$, $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda, q)$ and $h \leftarrow \mathcal{H}_\lambda$.
2. Output $\mathsf{crs} = \mathsf{crs}_{\mathsf{LE}} \| \mathsf{crs}_{\mathsf{VC}} \| h$.

$\mathsf{nmVC.Commit}(1^\lambda, \mathsf{crs}, (x_1, \ldots, x_q))$:

1. Parse crs as $\mathsf{crs}_{\mathsf{LE}} \| \mathsf{crs}_{\mathsf{VC}} \| h$.
2. Sample $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$ and compute $\tau = h(vk)$.
3. For each $i \in [q]$ compute $(c_i, d_i) \leftarrow \mathsf{LE.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{LE}}, x_i, i, \tau)$.
4. Compute $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, (c_1, \ldots, c_q))$ and $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
5. Output $(\mathsf{vcom}, \mathsf{st})$, where $\mathsf{vcom} = \mathsf{vcom}_0 \| vk \| \sigma$ and $\mathsf{st} = \mathsf{st}_0 \| c_1 \| \cdots \| c_q \| d_1 \| \cdots \| d_q$.

$\mathsf{nmVC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i)$:

1. Parse crs as $\mathsf{crs}_{\mathsf{LE}} \| \mathsf{crs}_{\mathsf{VC}} \| h$, vcom as $\mathsf{vcom}_0 \| vk \| \sigma$ and st as $\mathsf{st}_0 \| c_1 \| \cdots \| c_q \| d_1 \| \cdots \| d_q$.
2. Compute $\pi_0 \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
3. Output $\pi = c_i \| d_i \| \pi_0$.

$\mathsf{nmVC.Verify}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, i, x, \pi)$:

1. Parse crs as $\mathsf{crs}_{\mathsf{LE}} \| \mathsf{crs}_{\mathsf{VC}} \| h$, vcom as $\mathsf{vcom}_0 \| vk \| \sigma$ and st as π as $c_i \| d_i \| \pi_0$.
2. Compute $\tau := h(vk)$.
3. Output 1 if all of the following conditions hold:
   - $\mathsf{Sig.Verify}_{vk}(\mathsf{vcom}_0, \sigma) = 1$.

[9] We emphasize that the security of our construction does not rely on $\mathcal{VC}$ providing any flavor of hiding or succinctness, and this is discussed below in the overview of our proof.

- $\mathsf{VC.Verify}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, i, c_i, \pi_0) = 1$.
- $\mathsf{LE.Decommit}(1^\lambda, \mathsf{crs}_{\mathsf{LE}}, c_i, d_i, i, \tau) = x$.

Otherwise, output 0.

Finally, we note that for simplifying our construction and its proof, the length of the secret state $\mathsf{st} = \mathsf{st}_0 \| c_1 \| \cdots \| c_q \| d_1 \| \cdots \| d_q$ produced by the commitment algorithm $\mathsf{nmVC.Commit}$ in the above description depends linearly on q but this can be easily avoided whenever the committed vector $(x_1, \ldots, x_q)$ is additionally provided. Specifically, given $x_1, \ldots, x_q$, the entire sequence of values $c_1, \ldots, c_q, d_1, \ldots, d_q$ can be replaced with a single key K for a pseudorandom function PRF that will allow the algorithm $\mathsf{nmVC.Open}$ to recompute any of these values when needed. Specifically, instead of computing $(c_i, d_i) \leftarrow \mathsf{LE.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{LE}}, x_i, i, \tau)$ by feeding the algorithm $\mathsf{LE.Commit}$ with a fresh random string $r_i \leftarrow \{0,1\}^*$, we can instead feed it with a pseudorandom string $r_i = \mathsf{PRF}_K(\mathsf{crs}_{\mathsf{LE}}, x_i, i, \tau)$ which is reproducible via knowledge of K and x_i.

5.2 Proof of Security

The following theorem captures the security of our construction, showing that it satisfies our notion of non-malleability for vector commitment schemes (recall Definition 3.1) based on the security of its underlying building blocks: (1) a vector commitment scheme $\mathcal{VC}$, (2) a locally-equivocable commitment scheme with all-but-one binding $\mathcal{LE}$, (3) a one-time strongly-unforgeable signature scheme $\mathcal{SIG}$, and (4) a universal one-way hash family $\mathcal{H}$.

Theorem 5.1. *For every probabilistic polynomial-time algorithm $\mathcal{A}$ and polynomial $q = q(\lambda)$, there exists a probabilistic polynomial-time algorithm $\mathcal{S}_{\mathcal{A}}$ such that the following holds: For any probabilistic polynomial-time algorithm $\mathcal{R}$, there are probabilistic polynomial-time algorithms $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ and $\mathcal{B}_5$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{S}_{\mathcal{A}},\mathcal{R},\mathcal{D}}(\lambda) \leq\ & \mathbf{Adv}^{\mathsf{LocalEq}}_{\mathcal{LE},q,\mathcal{B}_1}(\lambda) + \mathbf{Adv}^{\mathsf{Forge}}_{\mathcal{SIG},\mathcal{B}_2}(\lambda) + \mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda) \\ & + 2 \cdot \left(\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda) + \mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{B}_5}(\lambda) \right)\end{aligned}$$

for every $\lambda \in \mathbb{N}$.

Proof overview. Our simulator $\mathcal{S}_{\mathcal{A}}$ first generates a vector commitment vcom to q equivocal commitments $c_1, \ldots, c_q$ using the equivocation algorithms of $\mathcal{LE}$ (and using all other building blocks honestly). It then invokes $\mathcal{A}$ on vcom to obtain the subset $\mathcal{I}$ from $\mathcal{A}$, and outputs $\mathcal{I}$ as the simulator's output in Step 2 of the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$. The simulator receives the values $(x_i)_{i\in\mathcal{I}}$, and provides $\mathcal{A}$ with the values $(x_i)_{i\in\mathcal{I}}$ along with proofs asserting their authenticity with respect to vcom, where these proofs are produced using the algorithm $\mathsf{LE.Equiv}_2$ that enables to open each c_i to each x_i. When $\mathcal{A}$ outputs a vector commitment $\widehat{\mathsf{vcom}}$ along with a subset $\mathcal{J}$, the simulator needs to provide it with local openings of vcom to the locations in $\overline{\mathcal{I}}$. To do so, it samples values for these locations from the conditional distribution $(\widetilde{x}_j)_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$ (recall the discussion on valid distributions from Section 3). The simulator then provides $\mathcal{A}$ with $(\widetilde{x}_j)_{j\in\overline{\mathcal{I}}}$ along with proofs asserting their authenticity with respect to vcom (which, again, are produced via $\mathsf{LE.Equiv}_2$). Finally, when $\mathcal{A}$ outputs values $(\widehat{x}_j)_{j\in\mathcal{J}}$ along with proofs for their authenticity with respect to $\widehat{\mathsf{vcom}}$, the simulator $\mathcal{S}_{\mathcal{A}}$ outputs the same values $(\widehat{x}_j)_{j\in\mathcal{J}}$.

In order to prove that the outputs of the experiments $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$ are computationally indistinguishable, we introduce a hybrid experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$. This

experiment is obtained from the real experiment $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ with one modification: Similarly to what the simulator $\mathcal{S}_{\mathcal{A}}$ does, the challenger now uses the equivocation algorithms of $\mathcal{LE}$ in order to produce the common reference string crs and the vector commitment vcom given to $\mathcal{A}$. Whenever $\mathcal{A}$ is due to receive a value x_i and a proof asserting its authenticity with respect to vcom (Steps 6 and 7 of the experiment $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$), the challenger uses $\mathsf{LE.Equiv}_2$ to produce this proof.

The proof of Theorem 5.1 then proceeds in two steps. In the first step, we prove that

$$\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda) \approx_c \mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda), \tag{5.1}$$

while in the second step, we prove that

$$\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda) \approx_c \mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda). \tag{5.2}$$

The first step is fairly straightforward. The only difference between the two experiments considered in Eq. (5.1) is in the way in which the public parameters, commitments and decommitments of $\mathcal{LE}$ are generated (i.e., generating them honestly vs. using the equivocation algorithms of $\mathcal{LE}$). But according to Definition 4.1, no polynomial-time algorithm should be able to tell these two scenarios apart. Hence, the success probability of $\mathcal{A}$ in the two experiments should be essentially the same.

The second step is more involved. Let h be the hash function included in the common-reference string; denote $\mathsf{vcom} = \mathsf{vcom}_0 \| vk \| \sigma$ where vcom is the vector commitment which $\mathcal{A}$ receives; and denote $\widehat{\mathsf{vcom}} = \widehat{\mathsf{vcom}_0} \| \widehat{vk} \| \widehat{\sigma}$ where $\widehat{\mathsf{vcom}}$ is the vector commitment which $\mathcal{A}$ outputs. In order to prove Eq. (5.2) we consider three separate cases:

- **Case 1: $\widehat{vk} = vk$.** This case reduces to the one-time strong unforgeability of $\mathcal{SIG}$. Conditioned on $\widehat{\mathsf{vcom}} = \mathsf{vcom}$ or $\mathsf{Sig.Verify}_{\widehat{vk}}(\widehat{\mathsf{vcom}_0}, \widehat{\sigma}) = 0$, the outputs of the two experiments considered in Eq. (5.2) are identically-distributed. Hence, any advantage that a distinguisher might have must originate from the case $\widehat{\mathsf{vcom}} \neq \mathsf{vcom}$ and $\mathsf{Sig.Verify}_{\widehat{vk}}(\widehat{\mathsf{vcom}_0}, \widehat{\sigma}) = 1$. But since $\widehat{vk} = vk$, it means that $(\widehat{\mathsf{vcom}_0}, \widehat{\sigma}) \neq (\mathsf{vcom}_0, \sigma)$, in contradiction to the assumption that $\mathcal{SIG}$ is one-time strongly unforgeable.

- **Case 2: $\widehat{vk} \neq vk$ but $h(\widehat{vk}) = h(vk)$.** This case reduces to the universal one-wayness of $\mathcal{H}$. The reduction is immediate from the observation that vk is chosen independently of h.

- **Case 3: $h(\widehat{vk}) \neq h(vk)$.** In this case we use the position binding of $\mathcal{VC}$ and the all-but-one binding of $\mathcal{LE}$ in order to bound the distinguishing advantage between the experiments considered in Eq. (5.2) conditioned on the event $h(\widehat{vk}) \neq h(vk)$. Technical details omitted, the main observation is that the view of the adversary $\mathcal{A}$ in $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ is distributed identically to the view of $\mathcal{A}$ in the simulation carried out by $\mathcal{S}_{\mathcal{A}}$ in $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$. Hence, the commitment $\widehat{\mathsf{vcom}}$ and the subset $\mathcal{J}$ that $\mathcal{A}$ outputs are also identically distributed in both experiments. As a result, roughly speaking, any change in the output distribution of $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ vis-à-vis the output distribution of $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$ must follow from $\mathcal{A}$'s ability to choose the values $(\widehat{x}_j)_{j\in\mathcal{J}}$ to which it opens the commitment $\widehat{\mathsf{vcom}}$ *after* issuing $\widehat{\mathsf{vcom}}$. That is, any advantage a distinguisher between these two outputs has must follow from $\mathcal{A}$'s ability to produce a commitment $\widehat{\mathsf{vcom}}$ and then to provide local openings to more than a single tuple of values $(\widehat{x}_j)_{j\in\mathcal{J}}$. These local openings are obtained by relying on the fact that generating $c_1, \ldots, c_q$ using the equivocation algorithms of $\mathcal{LE}$ does not bind them to a single tuple of values with respect to the tag τ, and as a result we can rewind $\mathcal{A}$ to obtain corresponding local openings with respect to the tag $\widehat{\tau}$. But, if $\mathcal{A}$ can open, say, the j-th location of $\widehat{\mathsf{vcom}}$ in two different ways, then there are two possibilities:

– Either it can open the j-th location of $\widehat{\mathsf{vcom}_0}$ in two different ways, in contradiction to the position binding of $\mathcal{VC}$;
– or it can produce a commitment $\widehat{c_j}$ of the scheme $\mathcal{LE}$ with respect to the tag $\widehat{\tau} = h(\widehat{vk})$ and open it in two different ways. But since we assumed here that $h(\widehat{vk}) \neq h(vk)$, it holds that $\widehat{\tau} \neq \tau$, where τ is the tag with respect to which the equivocal common-reference string was generated, in contradiction to the all-but-one binding of $\mathcal{LE}$ (recall Definition 4.2).

Finally, it should be noted that the security of our construction does not require any flavor of hiding or succinctness from the underlying vector commitment $\mathcal{VC}$. That is, our construction is non-malleable even if the locally-equivocable commitments $c_1, \ldots, c_q$ are completely revealed to the adversary. It is useful to consider, for example, the extreme case where the vector commitment vcom_0 for the vector $(c_1, \ldots, c_q)$ is simply the vector itself, and then to observe that the above proof overview still applies.

Proof of Theorem 5.1. Let $q = q(\lambda)$ be a polynomial, let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ be a valid distribution over $\{(\mathcal{X}_\lambda)^q\}_{\lambda\in\mathbb{N}}$, and let $\mathcal{A}$ be a probabilistic polynomial-time algorithm taking part in the experiment $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$. Consider the following simulator $\mathcal{S}_\mathcal{A}$:

The Simulator $\mathcal{S}_\mathcal{A}$

Input: The security parameter $\lambda \in \mathbb{N}$ and a description of the distribution $\mathcal{D}$.

1. Sample $\mathsf{st}_{\mathsf{equiv}} \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$.
2. Sample $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$.
3. Sample $h \leftarrow \mathcal{H}_\lambda$ and compute $\tau = h(vk)$.
4. Compute $(\widehat{\mathsf{crs}_{\mathsf{LE}}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_{\mathsf{equiv}}, q, \tau)$.
5. Sample $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda)$.
6. Compute $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, (\widehat{c_1}, \ldots, \widehat{c_q}))$.
7. Compute $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
8. Compute $(\mathcal{I}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \widehat{\mathsf{crs}_{\mathsf{LE}}} \| \mathsf{crs}_{\mathsf{VC}} \| h, \mathsf{vcom}_0 \| vk \| \sigma)$.
9. Output $\mathcal{I}$ as the output of the simulator $\mathcal{S}_\mathcal{A}$ in Step 2 of the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$, and receive values $(x_i)_{i\in\mathcal{I}}$ as additional input.
10. For each $i \in \mathcal{I}$:
    (a) Compute $\widehat{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1, \tau)$.
    (b) Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
    (c) Let $\pi_i = \widehat{c_i} \| \widehat{d_i} \| \pi_{0,i}$.
11. Compute $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_\mathcal{A}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_\mathcal{A}, (x_i)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$, where $\mathcal{J} \subseteq [q]$.
12. Sample $(\widetilde{x_j})_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$, where $\overline{\mathcal{I}} = [q] \setminus \mathcal{I}$.
13. For each $i \in \overline{\mathcal{I}}$:
    (a) Compute $\widehat{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, \widetilde{x_i}, i, \mathsf{st}_1, \tau)$.
    (b) Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
    (c) Let $\pi_i = \widehat{c_i} \| \widehat{d_i} \| \pi_{0,i}$.

14. Compute $((\widehat{x_j})_{j\in\mathcal{J}}, (\widehat{\pi_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (\widetilde{x_i})_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$.
15. If $\widehat{\mathsf{vcom}} = \mathsf{vcom}_0 \| vk \| \sigma$ or if there exists an index $j \in \mathcal{J}$ for which $\mathsf{nmVC.Verify}(1^\lambda, \widehat{\mathsf{crs}_{\mathsf{LE}}} \| \mathsf{crs}_{\mathsf{VC}} \| h, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}) = 0$, then output $(\mathcal{J}, (\perp)^{|\mathcal{J}|})$ as the output of the simulator $\mathcal{S}_{\mathcal{A}}$ in step 3 of the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$. Otherwise, output $(\mathcal{J}, (\widehat{x_j})_{j\in\mathcal{J}})$.

Let $\mathcal{R}$ be a probabilistic polynomial-time distinguisher. We wish to bound the advantage

$$\begin{aligned}&\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{S}_{\mathcal{A}},\mathcal{R},\mathcal{D}}(\lambda)\\ &\qquad = \left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1\right]\right|.\end{aligned}$$

To that end, we introduce a hybrid experiment. The experiment, denoted $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$, is obtained from the real experiment $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$, with one difference: All parameters associated with the locally equivocable commitment scheme with all-but-one binding $\mathcal{LE}$ given to $\mathcal{A}$ are generated using the equivocation algorithms $\mathsf{LE.Equiv}_1$ and $\mathsf{LE.Equiv}_2$ (instead of $\mathsf{LE.Setup}, \mathsf{LE.Commit}$ and $\mathsf{LE.Decommit}$). In detail, the experiment is defined as follows:

The Experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$

1. Sample $\mathsf{st}_{\mathsf{equiv}} \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$.
2. Sample $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$.
3. Sample $h \leftarrow \mathcal{H}_\lambda$ and compute $\tau = h(vk)$.
4. Compute $(\widehat{\mathsf{crs}_{\mathsf{LE}}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_{\mathsf{equiv}}, q, \tau)$.
5. Sample $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda)$.
6. Let $\mathsf{crs} = \widehat{\mathsf{crs}_{\mathsf{LE}}} \| \mathsf{crs}_{\mathsf{VC}} \| h$
7. Sample $(x_1, \ldots, x_q) \leftarrow \mathcal{D}$.
8. Compute $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, (\widehat{c_1}, \ldots, \widehat{c_q}))$.
9. Compute $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
10. Compute $(\mathcal{I}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_0 \| vk \| \sigma)$.
11. For each $i \in [q]$:
    (a) Compute $\widehat{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1, \tau)$.
    (b) Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
    (c) Let $\pi_i = \widehat{c_i} \| \widehat{d_i} \| \pi_{0,i}$.
12. Compute $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_{\mathcal{A}}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$.
13. Compute $((\widehat{x_j})_{j\in\mathcal{J}}, (\widehat{\pi_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i)_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$, where $\mathcal{J} \subseteq [q]$.
14. If $\widehat{\mathsf{vcom}} = \mathsf{vcom}$ or if there exists an index $j \in \mathcal{J}$ for which $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}\right) = 0$, then output $((x_1, \ldots, x_q), (\perp)^q, \mathcal{I})$.
    Otherwise, output $((x_1, \ldots, x_q), (\widehat{x_1}, \ldots, \widehat{x_q}), \mathcal{I})$, where for each $j \in [q] \setminus \mathcal{J}$, $\widehat{x_j} = \perp$.

By the triangle inequality, it holds that

$$\begin{aligned}&\left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1\right]\right|\\ &\quad \le \left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right]\right|\\ &\qquad + \left|\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1\right]\right|.\end{aligned}$$

The proof proceeds by bounding each of the above differences separately. This is captured by the following two lemmata.

Lemma 5.2. *There exists a probabilistic polynomial-time algorithm $\mathcal{B}_1$ such that*

$$\left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]-\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]\right|\leq \mathbf{Adv}^{\mathsf{LocalEq}}_{\mathcal{LE},q,\mathcal{B}_1}(\lambda)$$

for every $\lambda\in\mathbb{N}$.

Lemma 5.3. *There exist probabilistic polynomial-time algorithms $\mathcal{B}_2,\mathcal{B}_3,\mathcal{B}_4$ and $\mathcal{B}_5$ such that*

$$\begin{aligned}&\left|\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]-\Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\right]\right|\\&\quad\leq \mathbf{Adv}^{\mathsf{Forge}}_{\mathcal{SIG},\mathcal{B}_2}(\lambda)+\mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda)+2\cdot\left(\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda)+\mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{B}_5}(\lambda)\right)\end{aligned}$$

for every $\lambda\in\mathbb{N}$.

Theorem 5.1 follows immediately from Lemma 5.2 and Lemma 5.3. ■

Proof of Lemma 5.2. We construct an adversary $\mathcal{B}_1$ which, given a tuple $(\mathsf{crs}_{\mathsf{LE}},c_1,\ldots,c_q,d_1,\ldots,d_q)$, distinguishes between the case in which it was generated using the algorithms $(\mathsf{LE.Setup},\mathsf{LE.Commit},\mathsf{LE.Decommit})$ and the case in which it was generated using the algorithms $(\mathsf{LE.AltSetup},\mathsf{LE.Equiv}_1,\mathsf{LE.Equiv}_2)$, with advantage at least

$$\left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]-\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]\right|.$$

Concretely, for a security parameter $\lambda\in\mathbb{N}$ and a bit $b\in\{0,1\}$, the algorithm $\mathcal{B}_1$ takes part in the experiment $\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{B}_1,b}(\lambda)$ and is defined as follows:

1. Sample $(sk,vk)\leftarrow\mathsf{Sig.Gen}(1^\lambda)$, $h\leftarrow\mathcal{H}_\lambda$ and $(x_1,\ldots,x_q)\leftarrow\mathcal{D}$.
2. Compute $\tau=h(vk)$ and output τ and $x_1,\ldots,x_q$ as the adversary's output in Step 1 of $\mathsf{IndParam}^{\mathcal{LE}}_{q,\mathcal{B}_1,b}(\lambda)$. In response, $\mathcal{B}_1$ receives from the challenger a common-reference string $\mathsf{crs}_{\mathsf{LE}}$, commitments $c_1,\ldots,c_q$ and decommitments $d_1,\ldots,d_q$.
3. Invoke $\mathcal{A}$ and simulate to it either $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ or $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ as follows:
   (a) Sample $\mathsf{crs}_{\mathsf{VC}}\leftarrow\mathsf{VC.Setup}(1^\lambda)$ and $(\mathsf{vcom}_0,\mathsf{st}_0)\leftarrow\mathsf{VC.Commit}(1^\lambda,\mathsf{crs}_{\mathsf{vcom}},(c_1,\ldots,c_q))$. Compute $\sigma\leftarrow\mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$ and invoke $(\mathcal{I},\mathsf{st}_{\mathcal{A}})\leftarrow\mathcal{A}(1^\lambda,\mathsf{crs}_{\mathsf{LE}}\|\mathsf{crs}_{\mathsf{vcom}}\|h,\mathsf{vcom}_0\|vk\|\sigma)$.
   (b) For each $i\in[q]$ compute $\pi_{0,i}\leftarrow\mathsf{VC.Open}(1^\lambda,\mathsf{crs}_{\mathsf{vcom}},\mathsf{vcom}_0,\mathsf{st}_0,i)$, and set $\pi_i=c_i\|d_i\|\pi_{0,i}$.
   (c) Invoke $\left(\widehat{\mathsf{vcom}},\mathcal{J},\mathsf{st}_{\mathcal{A}}\right)\leftarrow\mathcal{A}\left(\mathsf{st}_{\mathcal{A}},(x_i)_{i\in\mathcal{I}},(\pi_i)_{i\in\mathcal{I}}\right)$.
   (d) Invoke $((\widehat{x}_j)_{j\in\mathcal{J}},(\widehat{\pi}_j)_{j\in\mathcal{J}})\leftarrow\mathcal{A}\left(\mathsf{st}_{\mathcal{A}},(x_i)_{i\in\overline{\mathcal{I}}},(\pi_i)_{i\in\overline{\mathcal{I}}}\right)$.
4. If $\widehat{\mathsf{vcom}}=\mathsf{vcom}$ or if there exists an index $j\in\mathcal{J}$ for which $\mathsf{VC.Verify}\left(1^\lambda,\mathsf{crs},\widehat{\mathsf{vcom}},j,\widehat{x}_j,\widehat{\pi}_j\right)=0$, then set $\mathsf{RInput}=((x_1,\ldots,x_q),(\bot)^q,\mathcal{I})$.
   Otherwise, set $\mathsf{RInput}=((x_1,\ldots,x_q),(\widehat{x_1},\ldots,\widehat{x_q}),\mathcal{I})$, where for each $j\in[q]\setminus\mathcal{J}$, $\widehat{x}_j=\bot$.
5. Output $b'\leftarrow\mathcal{R}(\mathsf{RInput})$.

Observe that if $b=0$ (i.e., if $\mathcal{B}_1$ takes part in $\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{B}_1,0}(\lambda)$) then $\mathcal{B}_1$ perfectly simulates $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ to $\mathcal{A}$ and hence

$$\Pr\left[\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{B}_1,0}(\lambda)=1\right]=\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]$$

for each $\lambda\in\mathbb{N}$. Similarly, if $b=1$ (i.e., if $\mathcal{B}_1$ takes part in $\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{B}_1,1}(\lambda)$) then $\mathcal{B}_1$ perfectly simulates $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ to $\mathcal{A}$ and hence

$$\Pr\left[\mathsf{IndParam}_{\mathcal{LE},q,\mathcal{B}_1,1}(\lambda)=1\right]=\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]$$

for each $\lambda \in \mathbb{N}$. This implies that

$$\left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]-\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]\right| \leq \mathbf{Adv}^{\mathsf{LocalEq}}_{\mathcal{LE},q,\mathcal{B}_1}(\lambda)$$

for every $\lambda \in \mathbb{N}$, concluding the proof of the lemma. ■

Proof of Lemma 5.3. Let VKCopy be the event in which the verification key outputted by $\mathcal{A}$ as part of the vector commitment $\widehat{\mathsf{vcom}}$ is the same as the one included in the vector commitment given to $\mathcal{A}$ as part of vcom. Let $\mathsf{HashColl}$ denote the event in which the verification key outputted by $\mathcal{A}$ as part of $\widehat{\mathsf{vcom}}$ is different than the one given to $\mathcal{A}$ in vcom, but the hash values of these two verification keys under the hash function h chosen from $\mathcal{H}_\lambda$ are equal. Let $\mathsf{DiffTags}$ denote the event in which the hash values of these two verification keys are distinct, and note that $\mathsf{DiffTags} = \overline{\mathsf{VKCopy}} \vee \mathsf{HashColl}$.

Note that the view of the adversary $\mathcal{A}$ in the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ is distributed identically to the view of the adversary $\mathcal{A}$ in the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$. In particular, this means that $\Pr[\mathsf{DiffTags}]$ is the same over both experiments, and thus

$$\begin{aligned}
&\left|\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]-\Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\right]\right| \\
&\quad\leq \left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{VKCopy}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{VKCopy}\end{array}\right]\right| && (5.3)\\
&\qquad+\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{HashColl}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{HashColl}\end{array}\right]\right| && (5.4)\\
&\qquad+\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]\right| && (5.5)
\end{aligned}$$

We bound each of the expressions in Eq. (5.3), (5.4) and (5.5) separately, using respective reductions to the unforgeability of the signature scheme $\mathcal{SIG}$, to the security of universal one-way hash family $\mathcal{H}$, and to either the position binding of $\mathcal{VC}$ or to the special binding property of the commitment scheme $\mathcal{LE}$. This is captured by the following three claims which settle the proof of Lemma 5.3.

Claim 5.4. *There exists a probabilistic polynomial-time algorithm $\mathcal{B}_2$ such that*

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{VKCopy}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{VKCopy}\end{array}\right]\right| \leq \mathbf{Adv}^{\mathsf{Forge}}_{\mathcal{SIG},\mathcal{B}_2}(\lambda)$$

for every $\lambda \in \mathbb{N}$.

Claim 5.5. *There exists a probabilistic polynomial-time algorithm $\mathcal{B}_3$ such that*

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{HashColl}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{HashColl}\end{array}\right]\right| \leq \mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda)$$

for every $\lambda \in \mathbb{N}$.

Claim 5.6. *There exist probabilistic polynomial-time algorithms $\mathcal{B}_4$ and $\mathcal{B}_5$ such that*

$$\begin{aligned}
&\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]\right| \\
&\qquad\leq 2\cdot\left(\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda)+\mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{B}_5}(\lambda)\right)
\end{aligned}$$

for every $\lambda \in \mathbb{N}$.

■

We now turn to prove Claims 5.4, 5.5 and 5.6.

Proof of Claim 5.4. We construct an adversary $\mathcal{B}_2$ against the one-time strong unforgeability of $\mathcal{SIG}$. On input vk, where $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$, the algorithm $\mathcal{B}_2$ is defined as follows:

1. Invoke $\mathcal{A}(1^\lambda)$ and simulate to it a partial execution of the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ (equivalently, $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$) using the verification key vk as follows:
   (a) Sample $\mathsf{st}_{\mathsf{equiv}} \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$, $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda)$ and $h \leftarrow \mathcal{H}_\lambda$, and set $\tau = h(vk)$.
   (b) Compute $(\widehat{\mathsf{crs}_{\mathsf{LE}}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_{\mathsf{equiv}}, q, \tau)$, $(x_1, \ldots, x_q) \leftarrow \mathcal{D}$, $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, (\widehat{c_1}, \ldots, \widehat{c_q}))$.
   (c) Output vcom_0 as the output message of the adversary in Step 2 of $\mathsf{Forge}_{\mathcal{SIG},\mathcal{A}}(\lambda)$. Receive in response a signature $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
   (d) Invoke $(\mathcal{I}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_0 \| vk \| \sigma)$, where $\mathsf{crs} = \widehat{\mathsf{crs}_{\mathsf{LE}}} \| \mathsf{crs}_{\mathsf{VC}} \| h$.
   (e) For each $i \in [q]$:
      i. Compute $\widehat{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1, \tau)$.
      ii. Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
      iii. Set $\pi_i = \widehat{c_i} \| \widehat{d_i} \| \pi_{0,i}$.
   (f) Invoke $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_\mathcal{A}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_\mathcal{A}, (x_i)_{i \in \mathcal{I}}, (\pi_i)_{i \in \mathcal{I}}\right)$.
2. Parse $\widehat{\mathsf{vcom}}$ as $(\widehat{\mathsf{vcom}_0}, \widehat{vk}, \widehat{\sigma})$. If $\widehat{vk} = vk$ and $(\widehat{\mathsf{vcom}_0}, \widehat{\sigma}) \neq (\mathsf{vcom}_0, \sigma)$, then output $(\widehat{\mathsf{vcom}_0}, \widehat{\sigma})$. Otherwise, output $\perp$.

Let $\mathsf{VKForge}$ denote the event in which $(\widehat{\mathsf{vcom}_0}, \widehat{\sigma}) \neq (\mathsf{vcom}_0, \sigma)$. Observe that $\mathcal{B}_2$ perfectly simulates $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ (equivalently, $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$) until the stage in which the simulation terminates. Moreover, conditioned on $\mathsf{VKCopy} \wedge \overline{\mathsf{VKForge}}$, the outputs of $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ and of $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$ are identically distributed. Therefore,

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{Forge}}_{\mathcal{SIG},\mathcal{B}_2}(\lambda) &= \Pr\left[\mathsf{Forge}_{\mathcal{SIG},\mathcal{A}}(\lambda) = 1\right] \\
&\geq \Pr\left[\mathsf{VKCopy} \wedge \mathsf{VKForge}\right] \\
&\geq \left| \Pr\left[\begin{array}{c} \mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{VKCopy} \wedge \mathsf{VKForge} \end{array}\right] - \Pr\left[\begin{array}{c} \mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{VKCopy} \wedge \mathsf{VKForge} \end{array}\right] \right| \\
&= \left| \Pr\left[\begin{array}{c} \mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{VKCopy} \end{array}\right] - \Pr\left[\begin{array}{c} \mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{VKCopy} \end{array}\right] \right|,
\end{aligned}$$

concluding the proof of the claim. ■

Proof of Claim 5.5. We construct an adversary $\mathcal{B}_3$ against the universal one-way hash function family $\mathcal{H}$. On input 1^λ, the algorithm $\mathcal{B}_3$ is defined as follows:

1. Sample $(sk, vk) \leftarrow \mathsf{Sig.Gen}(1^\lambda)$.
2. Set $\mathsf{st}_{\mathcal{B}_3} = (sk, vk)$ and output (vk, st) as the adversary's output in Step 1 of the experiment $\mathsf{UOWHF}_{\mathcal{H},\mathcal{B}_3}(\lambda)$. In response, receive from the challenger a hash function $h \leftarrow \mathcal{H}_\lambda$.
3. Invoke $\mathcal{A}$ and simulate to it a partial execution of the hybrid experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ using the pair (sk, vk) as follows:

(a) Set $\tau = h(vk)$.
(b) Sample $\mathsf{st}_{\mathsf{equiv}} \leftarrow \mathsf{LE.AltSetup}(1^\lambda, q)$, $(\widehat{\mathsf{crs}_{\mathsf{LE}}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st}_1) \leftarrow \mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_{\mathsf{equiv}}, q, \tau)$, and $\mathsf{crs}_{\mathsf{VC}} \leftarrow \mathsf{VC.Setup}(1^\lambda)$ and set $\mathsf{crs} = \widehat{\mathsf{crs}_{\mathsf{LE}}} \| \mathsf{crs}_{\mathsf{VC}} \| h$.
(c) Sample $(x_1, \ldots, x_q) \leftarrow \mathcal{D}$, and compute $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, (\widehat{c_1}, \ldots, \widehat{c_q}))$ and $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
(d) Invoke $(\mathcal{I}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_0 \| vk \| \sigma)$.
(e) For each $i \in [q]$:
  i. Compute $\widehat{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1, \tau)$.
  ii. Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
  iii. Set $\pi_i = \widehat{c_i} \| \widehat{d_i} \| \pi_{0,i}$.
(f) Invoke $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_{\mathcal{A}}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i)_{i \in \mathcal{I}}, (\pi_i)_{i \in \mathcal{I}}\right)$.

4. Parse $\widehat{\mathsf{vcom}}$ as $(\widehat{\mathsf{vcom}_0}, \widehat{vk}, \widehat{\sigma})$. If $\widehat{vk} \neq vk$ and $h(\widehat{vk}) = \tau$, then output $\widehat{vk}$ as the output of the adversary in Step 3 of the experiment $\mathsf{UOWHF}_{\mathcal{H},\mathcal{B}_3}(\lambda)$. Otherwise, output $\perp$.

Observe that $\mathcal{B}_3$ perfectly simulates $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ until the stage in which the simulation terminates, and hence by the definition of the event $\mathsf{HashColl}$, the probability that $\mathcal{B}_3$ outputs $\widehat{vk}$ such that $\widehat{vk} \neq vk$ and $h(\widehat{vk}) = \tau$ is equal to $\Pr[\mathsf{HashColl}]$, where the latter probability is taken over the randomness of the challenger and $\mathcal{A}$ in a random execution of $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$. Hence,

$$\begin{aligned}\mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda) &= \Pr\left[\mathsf{UOWHF}_{\mathcal{H},\mathcal{A}}(\lambda) = 1\right] \\ &= \Pr\left[\mathsf{HashColl}\right] \\ &\geq \Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge \mathsf{HashColl}\right]. \end{aligned} \tag{5.6}$$

Since h, vk and $\widehat{vk}$ are identically distributed in $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and in $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$, the same analysis implies that:

$$\mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda) \geq \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1 \wedge \mathsf{HashColl}\right]. \tag{5.7}$$

Overall, Eq. (5.6) and (5.7) imply that

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{HashColl}\end{array}\right] - \Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{HashColll}\end{array}\right]\right| \leq \mathbf{Adv}^{\mathsf{UOWHF}}_{\mathcal{H},\mathcal{B}_3}(\lambda),$$

concluding the proof of the claim. ∎

Proof of Claim 5.6. Let $\epsilon = \epsilon(\lambda)$ denote the difference

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{DiffTags}\end{array}\right] - \Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1 \\ \wedge \mathsf{DiffTags}\end{array}\right]\right|.$$

Recall that our goal is to construct probabilistic polynomial-time algorithms $\mathcal{B}_4$ and $\mathcal{B}_5$ such that

$$\epsilon \leq 2 \cdot \left(\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda) + \mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{B}_5}(\lambda)\right)$$

for every $\lambda \in \mathbb{N}$.

Let $\mathsf{ValidOpen}$ be the event in which $\mathsf{nmVC.Verify}(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}) = 1$ for all $j \in \mathcal{J}$. Note that over the simulation $\mathcal{S}_{\mathcal{A}}$, it holds that $\mathsf{ValidOpen}$ is contained in the event $\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right) = 1$. Similarly, over the hybrid experiment, $\mathsf{ValidOpen}$ is contained in $\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1$. Hence,

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]\right|$$
$$=\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\end{array}\right]\right|.$$

Assume without loss of generality that after Step 5 of the security experiment $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ the adversary $\mathcal{A}$ is deterministic (any random coins used by $\mathcal{A}$ after Step 5 can be included in the state $\mathsf{st}_{\mathcal{A}}$ maintained by $\mathcal{A}$). Let ω denote the random coins used by $\mathcal{A}$, and consider the tuple $(\mathsf{crs}, vk, \omega, (x_i)_{i\in\mathcal{I}})$ that consists of the common-reference string crs generated by the simulator $\mathcal{S}_{\mathcal{A}}$, the verification key vk used by the simulator to generate vcom, the random coins ω on which $\mathcal{A}$ is invoked, and the committed values $(x_i)_{i\in\mathcal{I}}$ corresponding to the locations in the subset $\mathcal{I}$ requested by $\mathcal{A}$. We call such a tuple $(\mathsf{crs}, vk, \omega, (x_i)_{i\in\mathcal{I}})$ *good* if

$$\Pr\left[\forall j\in\mathcal{J}:\mathsf{nmVC.Verify}(1^{\lambda},\mathsf{crs},\widehat{\mathsf{vcom}},j,\widehat{x_j},\widehat{\pi_j})=1\middle|(\mathsf{crs},vk,\omega,(x_i)_{i\in\mathcal{I}})\right]\geq\frac{\epsilon}{2},$$

where $\widehat{\mathsf{vcom}}$ and $\mathcal{J}$ are the vector commitment and the subset of indices outputted by $\mathcal{A}$ in Step 11 of the simulation (or Step 12 of the hybrid experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$), and $(\widehat{x_j},\widehat{\pi_j})_{j\in\mathcal{J}}$ are the local openings outputted by $\mathcal{A}$ in Step 14 of the simulation (or Step 13 of the hybrid experiment). Note that this probability is solely over the choice of $(x_i)_{i\in\overline{\mathcal{I}}}$, and it is identical in both $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$ and in $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$, as the view of $\mathcal{A}$ is identically distributed in both experiments. Let Good be the event in which the tuple $(\mathsf{crs}, vk, \omega, (x_i)_{i\in\mathcal{I}})$ is good. Then by the triangle inequality:

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\end{array}\right]\right|$$
$$\leq\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right]\right|$$
$$+\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\overline{\mathsf{Good}}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\overline{\mathsf{Good}}\end{array}\right]\right|.$$

Note that conditioned on $\overline{\mathsf{Good}}$, the probability that the output of the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ is not of the form $((x_1,\ldots,x_q),(\perp)^q,\mathcal{I})$ is at most $\epsilon/4$. Observe that the same applies to the output of $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$ as well. Hence, it holds that

$$\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\end{array}\right]\right| \tag{5.8}$$
$$\leq\left|\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right]-\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right]\right|+\frac{\epsilon}{2}.$$

Assume without loss of generality that

$$\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right]>\Pr\left[\begin{array}{c}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{array}\right].$$

If this is not the case, then the rest of the proof is symmetric. As is the case for the event Good, it is also the case that $\mathsf{DiffTags}\wedge\mathsf{ValidOpen}$ occurs with equal probabilities in both $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$

and $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$. Hence, we can rewrite

$$\begin{aligned}
&\Pr\left[\begin{matrix}\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{matrix}\right]-\Pr\left[\begin{matrix}\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\\ \wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\end{matrix}\right] \qquad (5.9)\\
&=\left(\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\middle|\begin{matrix}\mathsf{DiffTags}\\ \mathsf{ValidOpen}\\ \wedge\mathsf{Good}\end{matrix}\right]-\Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1\middle|\begin{matrix}\mathsf{DiffTags}\\ \mathsf{ValidOpen}\\ \wedge\mathsf{Good}\end{matrix}\right]\right)\\
&\quad\cdot\Pr\left[\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\right]\\
&\leq\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\overline{\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=1}\middle|\begin{matrix}\mathsf{DiffTags}\\ \mathsf{ValidOpen}\\ \wedge\mathsf{Good}\end{matrix}\right] \qquad (5.10)\\
&\quad\cdot\Pr\left[\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\right]\\
&=\Pr\left[\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)\right)=0\wedge\mathsf{DiffTags}\wedge\mathsf{ValidOpen}\wedge\mathsf{Good}\right],
\end{aligned}$$

where Eq. (5.10) follows from a union bound, and the probability is taken over the choice of the tuple $(\mathsf{crs},vk,\omega,(x_i)_{i\in\mathcal{I}})$, the choice of $(x_i)_{i\in\overline{\mathcal{I}}}$ and over the choice of $(\widetilde{x_i})_{i\in\overline{\mathcal{I}}}$ (recall that these are the surrogate values chosen by the simulator $\mathcal{S}_\mathcal{A}$ in Step 12 of the simulation).

For brevity, we denote the conjunction

$$\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},R,\mathcal{D}}(\lambda)\right)=0\wedge\mathsf{DiffTags}\wedge\mathsf{Good}\wedge\mathsf{ValidOpen}$$

by the event E. Then, From Eq. (5.9) and (5.8), we get that

$$\Pr\left[\mathsf{E}\right]\geq\frac{\epsilon}{2} \qquad (5.11)$$

for infinitely many values of $\lambda\in\mathbb{N}$.

We now present two adversaries $\mathcal{B}_4$ and $\mathcal{B}_5$ attempting to break the all-but-one binding of $\mathcal{LE}$ and the position binding of $\mathcal{VC}$, respectively. We will use Eq. (5.11) in order to prove that the sum of their advantages is at least $\epsilon/2$. The two algorithm are somewhat similar, so we start by defining $\mathcal{B}_4$ in detail, and then describe how $\mathcal{B}_5$ is defined vis-à-vis $\mathcal{B}_4$. As $\mathcal{B}_4$ invokes the adversary $\mathcal{A}$, we remind the reader that we assume without loss of generality that $\mathcal{A}$ is deterministic after Step 5 of $\mathsf{Real}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$.

The Algorithm $\mathcal{B}_4$

Initial input: The security parameter $\lambda\in\mathbb{N}$.

1. Sample $(sk,vk)\leftarrow\mathsf{Sig.Gen}(1^\lambda)$ and $h\leftarrow\mathcal{H}_\lambda$, and compute $\tau=h(vk)$.
2. Output τ as the tag outputted by the adversary in Step 1 of $\mathsf{ABOBind}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda)$. At this point, the challenger samples $\mathsf{st}_{\mathsf{equiv}}\leftarrow\mathsf{LE.AltSetup}(1^\lambda,q)$ and $\rho\leftarrow\{0,1\}^r$, where $r=r(\lambda)$ is the number of random coins used by $\mathsf{LE.Equiv}_1$ on security parameter $\lambda\in\mathbb{N}$. The challenger then passes $\mathsf{st}_{\mathsf{equiv}}$ and ρ to $\mathcal{B}_4$.
3. Compute $(\mathsf{crs}_{\mathsf{com}},c_1,\ldots,c_q,\mathsf{st}_1)=\mathsf{LE.Equiv}_1(1^\lambda,\mathsf{st}_{\mathsf{equiv}},q,\tau;\rho)$, sample $\mathsf{crs}_{\mathsf{VC}}\leftarrow\mathsf{VC.Setup}(1^\lambda)$, and set $\mathsf{crs}=\mathsf{crs}_{\mathsf{LE}}\|\mathsf{crs}_{\mathsf{VC}}\|h$.
4. Sample $(x_1,\ldots,x_q)\leftarrow\mathcal{D}$.
5. Compute $(\mathsf{vcom}_0,\mathsf{st}_0)\leftarrow\mathsf{VC.Commit}(1^\lambda,\mathsf{crs}_{\mathsf{VC}},(c_1,\ldots,c_q))$ and $\sigma\leftarrow\mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
6. Sample $\omega\leftarrow\{0,1\}^{r'}$, where $r'=r'(\lambda)$ is the number of random coins used by $\mathcal{A}$.
7. Compute $(\mathcal{I},\mathsf{st}_\mathcal{A})=\mathcal{A}(1^\lambda,\mathsf{crs},\mathsf{vcom}_0\|vk\|\sigma;\omega)$.

8. For each $i \in [q]$:
   (a) Compute $d_i = \mathsf{LE.Equiv}_2(1^\lambda, x_i, i, \mathsf{st}_1, \tau)$.
   (b) Compute $\pi_{0,i} \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i)$.
   (c) Set $\pi_i = c_i \| d_i \| \pi_{0,i}$.
9. Compute $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}'_{\mathcal{A}}\right) = \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$, where $\mathcal{J} \subseteq [q]$. Parse $\widehat{\mathsf{vcom}}$ as $\widehat{\mathsf{vcom}_0} \| \widehat{vk} \| \widehat{\sigma}$, and let $\widehat{\tau} = h(\widehat{vk})$.
10. Compute $((\widehat{x_j})_{j\in\mathcal{J}}, (\widehat{\pi_j})_{j\in\mathcal{J}}) := \mathcal{A}\left(\mathsf{st}'_{\mathcal{A}}, (x_i)_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$. Parse each $\widehat{\pi_j}$ as $\widehat{c_j} \| \widehat{d_j} \| \widehat{\pi_{0,j}}$.
11. Sample $(\widetilde{x_j})_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$.
12. For each $i \in \overline{\mathcal{I}}$:
   (a) Compute $\widetilde{d_i} = \mathsf{LE.Equiv}_2(1^\lambda, \widetilde{x_i}, i, \mathsf{st}_1, \tau)$.
   (b) Compute $\widetilde{\pi_i} = \widehat{c_i} \| \widetilde{d_i} \| \pi_{0,i}$.
13. Compute $\left((\widehat{x'_j})_{j\in\mathcal{J}}, (\widehat{\pi'_j})_{j\in\mathcal{J}}\right) = \mathcal{A}\left(\mathsf{st}'_{\mathcal{A}}, (\widetilde{x_i})_{i\in\overline{\mathcal{I}}}, (\widetilde{\pi_i})_{i\in\overline{\mathcal{I}}}\right)$.
14. Parse each $\widehat{\pi'_j}$ as $\widehat{c'_j} \| \widehat{d'_j} \| \widehat{\pi'_{0,j}}$, and verify that for each $j \in \mathcal{J}$ it holds that $\widehat{c'_j} = \widehat{c_j}$. If this is not the case, then output $\perp$ and terminate.
15. Verify that $\widehat{\tau} \neq \tau$, and that for each $j \in \mathcal{J}$ it holds that $\mathsf{nmVC.Verify}(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}) = 1$ and $\mathsf{nmVC.Verify}(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x'_j}, \widehat{\pi'_j}) = 1$. If any of these conditions does not hold, output $\perp$ and terminate.
16. If there exists an index $j^* \in \mathcal{J}$ such that $\widehat{x_{j^*}} \neq \widehat{x'_{j^*}}$, then output the quadruple $(\widehat{c_{j^*}}, \widehat{d_{j^*}}, \widehat{c'_{j^*}}, \widehat{\tau})$ as the output of the adversary in Step 5 of the experiment $\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{B}_4}(\lambda)$. If no such index exists, output $\perp$ and terminate.

The adversary $\mathcal{B}_5$ attempting to break the position binding of $\mathcal{VC}$ is defined similarly to $\mathcal{B}_4$, but with the following modifications:

- Instead of receiving $\mathsf{st}_{\mathsf{equiv}}$ and the randomness ρ used by $\mathsf{LE.Equiv}_1$ from the challenger (Step 4 of $\mathcal{B}_4$), $\mathcal{B}_5$ sample them on their own, where $\mathsf{st}_{\mathsf{equiv}}$ is sampled using $\mathsf{LE.AltSetup}$.
- Instead of sampling $\mathsf{crs}_{\mathsf{VC}}$ (Step 3 of $\mathcal{B}_4$), $\mathcal{B}_5$ receives it from the challenger.
- $\mathcal{B}_5$ does not terminate in Step 14 if for some $j \in \mathcal{J}$ it holds that $\widehat{c'_j} \neq \widehat{c_j}$.
- In Step 16 (if reached), $\mathcal{B}_5$ decides on their output as follows: If there exists an index $j^* \in \mathcal{J}$ such that $\widehat{c'_{j^*}} \neq \widehat{c_{j^*}}$, then output $(\widehat{\mathsf{vcom}_0}, j^*, \widehat{c'_{j^*}}, \widehat{c_{j^*}}, \widehat{\pi'_{0,j^*}}, \widehat{\pi_{0,j^*}})$ as the output of the adversary in Step 3 of the position binding security experiment $\mathsf{PosBind}^{\mathcal{VC}}_{q,\mathcal{B}_5}(\lambda)$, and otherwise output $\perp$ (if more then one such j^* exist, choose one arbitrarily).

Let $\mathsf{DiffCom}$ denote the event in which there exists an index $j^* \in \mathcal{J}$ such that $\widehat{c'_{j^*}} \neq \widehat{c_{j^*}}$, defined over a random execution of $\mathcal{B}_4$ or of $\mathcal{B}_5$ (observe that the commitments $(\widehat{c_j})_{j\in\mathcal{J}}$ and $(\widehat{c'_j})_{j\in\mathcal{J}}$ are identically distributed in both executions). On the one hand, for every $\lambda \in \mathbb{N}$ it holds that

$$\mathbf{Adv}^{\mathsf{PosBind}}_{\mathcal{VC},q,\mathcal{B}_5}(\lambda) = \Pr\left[\mathsf{PosBind}^{\mathcal{VC}}_{q,\mathcal{B}_5}(\lambda) = 1\right] \geq \Pr\left[\mathsf{E} \wedge \mathsf{DiffCom}\right]. \tag{5.12}$$

This holds since $\widehat{\mathsf{vcom}_0}$ is distributed as in the experiments $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$, the pairs $(\widehat{c_j}, \widehat{\pi_{0,j}})_{j\in\mathcal{J}}$ are distributed as in the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and the pairs $(\widehat{c'_j}, \widehat{\pi'_{0,j}})_{j\in\mathcal{J}}$ are distributed as in $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_{\mathcal{A}},\mathcal{D}}(\lambda)$. Moreover, by definition, whenever $\mathsf{DiffCom}$ occurs, there exists an index $j^* \in \mathcal{J}$ such that $\widehat{c'_{j^*}} \neq \widehat{c_{j^*}}$.

Assume without loss of generality that $\mathcal{R}$ always outputs 0 on input of the form $((x_1,\ldots,x_q),(\perp)^q,\mathcal{I})$ (note that any distinguisher can be transformed into such a distinguisher without affecting its advantage). In this case, whenever the event $\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\mathsf{ValidOpen}$ (which contains the event E) occurs, it means in particular that

$$\mathsf{VC.Verify}(1^\lambda,\mathsf{crs}_{\mathsf{vcom}},\widehat{\mathsf{vcom}_0},j,\widehat{c_j},\widehat{\pi_{0,j}})=1$$

and

$$\mathsf{VC.Verify}(1^\lambda,\mathsf{crs}_{\mathsf{vcom}},\widehat{\mathsf{vcom}_0},j,\widehat{c'_j},\widehat{\pi'_{0,j}})=1$$

for every $j\in\mathcal{J}$. In particular, this holds for j^*, implying that indeed $\mathsf{PosBind}^{\mathcal{VC}}_{q,\mathcal{B}_5}(\lambda)=1$.

On the other hand, for every $\lambda\in\mathbb{N}$ it holds that

$$\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{B}_4}(\lambda)=\Pr\left[\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{B}_4}(\lambda)=1\right]\geq\Pr\left[\mathsf{E}\wedge\overline{\mathsf{DiffCom}}\right]. \tag{5.13}$$

As before, $\widehat{\mathsf{vcom}_0}$ is distributed exactly as in the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and in the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$, the pairs $(\widehat{c_j},\widehat{\pi_{0,j}})_{j\in\mathcal{J}}$ are distributed as in the experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and the pairs $(\widehat{c'_j},\widehat{\pi'_{0,j}})_{j\in\mathcal{J}}$ are distributed as in the experiment $\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},\mathcal{D}}(\lambda)$. In addition, the event

$$\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{nm}\mathcal{VC},q,\mathcal{S}_\mathcal{A},R,\mathcal{D}}(\lambda)\right)=0\wedge\mathsf{ValidOpen}\wedge\mathsf{DiffTags}$$

implies that there exists at least one index $j^*\in\mathcal{J}$ for which $\widehat{x_{j^*}}\neq\widehat{x'_{j^*}}$ and $\widehat{x_{j^*}},\widehat{x'_{j^*}}\neq\perp$. The event $\overline{\mathsf{DiffCom}}$ means that for every $j\in\mathcal{J}$ it holds that $\widehat{c_j}=\widehat{c'_j}$. The event $\mathcal{R}\left(\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge\mathsf{ValidOpen}$ further means that for every $j\in\mathcal{J}$, it holds that $\mathsf{LE.Decommit}(1^\lambda,\mathsf{crs}_{\mathsf{LE}},\widehat{c_j},\widehat{d_j},\widehat{\tau})=\widehat{x_j}$ and that $\mathsf{LE.Decommit}(1^\lambda,\mathsf{crs}_{\mathsf{LE}},\widehat{c'_j},\widehat{d'_j},\widehat{\tau})=\mathsf{LE.Decommit}(1^\lambda,\mathsf{crs}_{\mathsf{LE}},\widehat{c_j},\widehat{d'_j},\widehat{\tau})=\widehat{x'_j}$. This holds in particular for j^*. Finally, $\mathsf{DiffTags}$ implies that $\widehat{\tau}\neq\tau$. Hence, in this case, $\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{B}_4}(\lambda)=1$. Taking Eq. (5.12) and (5.13) together with Eq. (5.11), immediately implies Claim 5.6. ■

6 Non-Malleable Dynamic Vector Commitments

In this section we show that our framework for non-malleable vector commitment schemes, as well as our construction, extend to a useful notion of non-malleable *dynamic* vector commitments (i.e., vector commitments that support updates).

The framework of Catalano and Fiore [CF13] considers vector commitments that can be updated *publicly* (i.e., without knowledge of the committer's private state): Any user with knowledge of the value x_i at the i-th location of the committed vector can update the commitment with respect to this location. Such public updates, however, are inherently incompatible with the motivation underlying the notion of non-malleability (and are, in particular, ruled out by Definition 3.1). Specifically, vector commitments supporting public updates are malleable, as they enable an adversary receiving a vector commitment vcom to produce a vector commitment $\widehat{\mathsf{vcom}}$ to a related vector by updating one or more of its coordinates.

In light of this inherent limitation, we show that our framework and construction can nevertheless support updates in a *private* manner. That is, updating a vector commitment requires knowledge of the private state generated by the committer alongside the initial generation of the vector commitment. This inherently-limited, yet still useful, form of updates disables any attack via the update procedure as described above, since an adversary does not have access to the private state of the vector commitment.

It should be noted that knowledge of the private state may enable to trivially update a vector commitment by recomputing it from scratch. The efficiency of such a trivial update procedure, however, depends (at least) linearly of the length of the entire committed vector. Thus, our main challenge in this setting is to support private updates in a completely local manner.

In what follows, in Section 6.1 we introduce the syntax and correctness requirement of dynamic vector commitments when adapted to support private updates, and introduce an additional property of vector commitments on which we rely for extending our construction (and which is satisfied by existing constructions). Then, in Section 6.2 we extend our notion of non-malleability to consider private updates, and in Section 6.3 we show that our construction and its proof of security can be extended as well.

6.1 Syntax, Correctness and Invisibility of Updates

A dynamic vector commitment scheme $\mathcal{VC}$ with private updates includes, in addition to the four algorithms specified in Section 2, an update algorithm $\mathsf{VC.Update}$. This is a probabilistic polynomial-time algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$, a common-reference string crs, a commitment vcom, a state st, an index $i \in [q]$ and a value $x \in \mathcal{X}_\lambda$, and outputs an update commitment vcom' and an updated state st'.

Correctness. A dynamic vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify}, \mathsf{VC.Update})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is correct if for any $\lambda \in \mathbb{N}$, if for any polynomials $q = q(\lambda)$ and $u = u(\lambda)$, for any vector $\left(x_1^{(0)}, \ldots, x_q^{(0)}\right) \in (\mathcal{X}_\lambda)^q$, for any sequence of pairs $\left(x_{i_j}^{(j)}, i_j\right)_{j\in[u]} \in (\mathcal{X}_\lambda \times [q])^u$, and for any index $i \in [q]$, it holds that

$$\Pr\left[\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}_u, i, x_i^*, \pi\right) = 1\right] = 1,$$

where:

1. $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda)$.
2. $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}\left(1^\lambda, \mathsf{crs}, \left(x_1^{(0)}, \ldots, x_q^{(0)}\right)\right)$.
3. $(\mathsf{vcom}_j, \mathsf{st}_j) \leftarrow \mathsf{VC.Update}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_{j-1}, \mathsf{st}_{j-1}, i_j, x_{i_j}^{(j)})$ for $j = 1, \ldots, u$.
4. $x_i^* \stackrel{\mathsf{def}}{=} x_{i_j}^{(j)}$ for the maximal $j \in [u]$ for which $i_j = i$, and $x_i^* \stackrel{\mathsf{def}}{=} x_i^{(0)}$ if no such j exists.
5. $\pi \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_u, \mathsf{st}_u, i)$.

Invisibility of updates. For extending our construction to the adaptive setting, we rely on the following natural property of vector commitments, to which we refer as *invisible updates*, and which is satisfied by the constructions of Catalano and Fiore [CF13]. Roughly speaking, this property requires that a vector commitment and its local openings that are obtained via a sequence of updates is computationally indistinguishable from a newly generated vector commitment and its local openings. We note that the constructions of Catalano and Fiore in fact satisfy this property in a perfect information-theoretic sense.

Definition 6.1. A dynamic vector commitment scheme $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify}, \mathsf{VC.Update})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ has *invisible updates* if for any polynomial $q = q(\lambda)$ and for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{InvisUpdts}}_{\mathsf{VC},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{InvisUpdts}^{(0)}_{\mathsf{VC},\mathcal{A}}(\lambda)\right] - \Pr\left[\mathsf{InvisUpdts}^{(1)}_{\mathsf{VC},\mathcal{A}}(\lambda)\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where for each $b \in \{0,1\}$ the experiment $\mathsf{InvisUpdts}^{(b)}_{\mathsf{VC},\mathcal{A}}(\lambda)$ is defined as follows:

1. $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda)$.
2. $\left(\left(x_i^{(0)}\right)_{i\in[q]}, \left(x_{i_j}^{(j)}, i_j\right)_{j\in[u]}, \mathsf{st}_{\mathcal{A}}\right) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs})$.
3. $\mathsf{vcom}_0^{(0)} \leftarrow \mathsf{VC.Commit}\left(1^\lambda, \mathsf{crs}, \left(x_1^{(0)}, \ldots, x_q^{(0)}\right)\right)$.
4. $\left(\mathsf{vcom}_j^{(0)}, \mathsf{st}_j^{(0)}\right) \leftarrow \mathsf{VC.Update}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}_{j-1}, \mathsf{st}_{j-1}, i_j, x_{i_j}^{(j)}\right)$ for each $j = 1, \ldots, u$.
5. $\mathsf{vcom}^{(0)} := \mathsf{vcom}_u^{(0)}$ and $\mathsf{st}^{(0)} := \mathsf{st}_u^{(0)}$.
6. $\left(\mathsf{vcom}^{(1)}, \mathsf{st}^{(1)}\right) \leftarrow \mathsf{VC.Commit}(1^\lambda, \mathsf{crs}, (x_1^*, \ldots, x_q^*))$, where $x_i^* \stackrel{\mathsf{def}}{=} x_{i_j}^{(j)}$ for the maximal $j \in [u]$ for which $i_j = i$, and $x_i^* \stackrel{\mathsf{def}}{=} x_i^{(0)}$ if no such j exists.
7. $\pi_i^{(0)} \leftarrow \mathsf{VC.Open}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}^{(0)}, \mathsf{st}^{(0)}\right)$ and $\pi_i^{(1)} \leftarrow \mathsf{VC.Open}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}^{(1)}, \mathsf{st}^{(1)}\right)$ for $i \in [q]$.
8. $b' \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, \mathsf{vcom}^{(b)}, \left(\pi_i^{(b)}\right)_{i\in[q]}\right)$.
9. Output b'.

6.2 Dynamic Non-Malleability

The following definition naturally extends Definition 3.1 to consider adversaries that may receive a vector commitment which is produced by a sequence of updates. For simplicity, the following definition considers non-adaptive updates. However, we note that it naturally extends to the case of adaptive updates (and so does our construction's proof of security).

Definition 6.2. A dynamic vector commitment $\mathcal{VC} = (\mathsf{VC.Setup}, \mathsf{VC.Commit}, \mathsf{VC.Open}, \mathsf{VC.Verify}, \mathsf{VC.Update})$ over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ is *dynamically non-malleable* if for any polynomials $q = q(\lambda)$ and $u = u(\lambda)$ and for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exist a probabilistic polynomial-time algorithm $\mathcal{S}$ such that the following holds: For any probabilistic polynomial-time algorithm $\mathcal{R}$ and for any valid distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ over $\{(\mathcal{X}_\lambda)^q \times (\mathcal{X}_\lambda \times [q])^u\}_{\lambda\in\mathbb{N}}$, there exists a negligible function $\nu(\cdot)$ such that

$$\mathbf{Adv}^{\mathsf{NM}}_{\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)\right) = 1\right]\right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the experiments $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ and $\mathsf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)$ are defined as follows:

The experiment $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$:

1. $\mathsf{crs} \leftarrow \mathsf{VC.Setup}(1^\lambda, q)$.
2. $\left(\left(x_i^{(0)}\right)_{i\in[q]}, \left(x_{i_j}^{(j)}, i_j\right)_{j\in[u]}\right) \leftarrow \mathcal{D}_\lambda$.
3. $(\mathsf{vcom}_0, \mathsf{st}_0) \leftarrow \mathsf{VC.Commit}\left(1^\lambda, \mathsf{crs}, \left(x_1^{(0)}, \ldots, x_q^{(0)}\right)\right)$.
4. $(\mathsf{vcom}_j, \mathsf{st}_j) \leftarrow \mathsf{VC.Update}\left(1^\lambda, \mathsf{crs}, \mathsf{vcom}_{j-1}, \mathsf{st}_{j-1}, i_j, x_{i_j}^{(j)}\right)$ for each $j = 1, \ldots, u$.
5. For each $i \in [q]$, set $x_i^* := x_{i_j}^{(j)}$ for the maximal $j \in [u]$ for which $i_j = i$, and $x_i^* := x_i^{(0)}$ if no such j exists.
6. $\pi_i \leftarrow \mathsf{VC.Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_u, \mathsf{st}_u, i)$ for each $i \in [q]$.
7. $(\mathcal{I}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}, \mathsf{vcom}_u)$ where $\mathcal{I} \subseteq [q]$.

8. $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_{\mathcal{A}}\right) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i^*)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$ where $\mathcal{J} \subseteq [q]$.
9. $((\widehat{x_j})_{j\in\mathcal{J}}, (\widehat{\pi_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{A}\left(\mathsf{st}_{\mathcal{A}}, (x_i^*)_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$, where $\overline{\mathcal{I}} = [q] \setminus \mathcal{I}$.
10. If $\widehat{\mathsf{vcom}} = \mathsf{vcom}$ or if $\mathsf{VC.Verify}\left(1^\lambda, \mathsf{crs}, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j}\right) = 0$ for some $j \in \mathcal{J}$, then output $\left((x_1^*, \ldots, x_q^*), (\perp)^q, \mathcal{I}\right)$.
Otherwise, output $\left((x_1^*, \ldots, x_q^*), (\widehat{x_1}, \ldots, \widehat{x_q}), \mathcal{I}\right)$, where $\widehat{x_j} = \perp$ for each $j \in [q] \setminus \mathcal{J}$.

The experiment $\mathsf{Ideal}_{\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)$:

1. $\left(\left(x_i^{(0)}\right)_{i\in[q]}, \left(x_{i_j}^{(j)}, i_j\right)_{j\in[u]}\right) \leftarrow \mathcal{D}_\lambda$.
2. For each $i \in [q]$, set $x_i^* := x_{i_j}^{(j)}$ for the maximal $j \in [u]$ for which $i_j = i$, and $x_i^* := x_i^{(0)}$ if no such j exists.
3. $(\mathcal{I}, \mathsf{st}_{\mathcal{S}}) \leftarrow \mathcal{S}(1^\lambda, \mathcal{D})$.
4. $(\mathcal{J}, (\widehat{x_j})_{j\in\mathcal{J}}) \leftarrow \mathcal{S}(\mathsf{st}_{\mathcal{S}}, (x_i^*)_{i\in\mathcal{I}})$.
5. Output $\left((x_1^*, \ldots, x_q^*), (\widehat{x_1}, \ldots, \widehat{x_q}), \mathcal{I}\right)$ where $\widehat{x_i} = \perp$ for every $i \in [q] \setminus \mathcal{J}$.

Valid distributions. Any distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ over the domain $\{(\mathcal{X}_\lambda)^q \times (\mathcal{X}_\lambda \times [q])^u\}_{\lambda\in\mathbb{N}}$ induces a distribution $\mathcal{D}^* = \{\mathcal{D}_\lambda^*\}_{\lambda\in\mathbb{N}}$ over $\{(\mathcal{X}_\lambda)^q\}_{\lambda\in\mathbb{N}}$ in the following natural manner; in order to sample a vector according to $\mathcal{D}_\lambda^*$:

1. Sample $\left(\left(x_i^{(0)}\right)_{i\in[q]}, \left(x_{i_j}^{(j)}, i_j\right)_{j\in[u]}\right) \leftarrow \mathcal{D}_\lambda$.
2. For each $i \in [q]$, set $x_i^* := x_{i_j}^{(j)}$ for the maximal $j \in [u]$ for which $i_j = i$, and $x_i^* := x_i^{(0)}$ if no such j exists.
3. Output $(x_1^*, \ldots, x_q^*)$.

We say that a distribution $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ is valid, if the induced distribution $\mathcal{D}^*$ is valid according to our definition from Section 3. That is, if for every $\lambda \in \mathbb{N}$, for every $\vec{x} = (x_1, \ldots, x_{q(\lambda)})$ in the support of $\mathcal{D}_\lambda^*$, and for every subset $\mathcal{I} = (i_1, \ldots, i_{|\mathcal{I}|}) \subseteq [q(\lambda)]$, it is possible to efficiently sample a vector $\vec{y}$ from the conditional distribution $\mathcal{D}_\lambda^*|(\forall i \in \mathcal{I} : y_i = x_i)$. We denote the process of sampling the entries of $\vec{y}$ in $\overline{\mathcal{I}} = [q] \setminus \mathcal{I}$ by $(y_j)_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}^*|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$.

6.3 Extending Our Construction and its Proof of Security

In order to augment our construction from Section 5 with private updates, we first rely on any standard strongly-unforgeable signature scheme instead of on any one-time strongly-unforgeable signature scheme. Then, we define an additional update algorithm (the remaining four algorithms of $\mathsf{nm}\mathcal{VC}$ remain unchanged) by relying on the update algorithm of the underlying vector commitment scheme $\mathcal{VC}$. On input $(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i, x)$, the algorithm $\mathsf{nmVC.Update}$ is defined as follows:

1. Parse crs as $\mathsf{crs}_{\mathsf{LE}}\|\mathsf{crs}_{\mathsf{VC}}\|h$, vcom as $\mathsf{vcom}_0\|vk\|\sigma$ and st as $\mathsf{st}_0\|c_1\|\cdots\|c_q\|d_1\|\cdots\|d_q$.
2. Compute $(c_i, d_i) \leftarrow \mathsf{LE.Commit}(1^\lambda, \mathsf{crs}_{\mathsf{LE}}, x, i, \tau)$ where $\tau = h(vk)$.
3. Compute $(\mathsf{vcom}_0, \mathsf{st}_0') \leftarrow \mathsf{Update}(1^\lambda, \mathsf{crs}_{\mathsf{VC}}, \mathsf{vcom}_0, \mathsf{st}_0, i, c_i)$ and $\sigma \leftarrow \mathsf{Sig.Sign}_{sk}(\mathsf{vcom}_0)$.
4. Output $(\mathsf{vcom}', \mathsf{st}')$, where $\mathsf{vcom}' = \mathsf{vcom}_0\|vk\|\sigma$ and $\mathsf{st}' = \mathsf{st}_0'\|c_1\|\cdots\|c_q\|d_1\|\cdots\|d_q$.

Extending the proof of Theorem 5.1. In order to extend the proof of Theorem 5.1 to prove that $\mathsf{nm}\mathcal{VC}$ is dynamically non-malleable (i.e., that it satisfies Definition 6.2), there are two technical adjustments that need to be made:

- Whenever we sample from the distribution $\mathcal{D}$, this now a distribution over $(\mathcal{X}_\lambda)^q \times (\mathcal{X}_\lambda \times [q])^u$ rather than over $(\mathcal{X}_\lambda)^q$. In particular, this applies to the definition of the hybrid experiment $\mathsf{Hybrid}_{\mathsf{nm}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ (Step 7); the definition of algorithm $\mathcal{B}_1$ (Step 1) in the proof of Lemma 5.2; the definition of algorithm $\mathcal{B}_2$ (Step 1(b)) in the proof of Claim 5.4; the definition of algorithm $\mathcal{B}_3$ (Step 3(c)) in the proof of Claim 5.5; and in Claim 5.6 – in the definition of the algorithm $\mathcal{B}_4$ (step 8) and in the corresponding step of the algorithm $\mathcal{B}_5$.
- In all places where we sample from the conditional distribution $\mathcal{D}|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$ (recall Section 3), we now need to sample from the conditional distribution $\mathcal{D}^*|(\mathcal{I}, (x_i^*)_{i\in\mathcal{I}})$, where $\mathcal{D}^*$ is the induced distribution over the final values $x_1^*, \ldots, x_q^*$ as defined above. Concretely, this should be done in the definition of the simulator $\mathcal{S}_\mathcal{A}$ (Step 12), and in Claim 5.6 – in the definition of the algorithm $\mathcal{B}_4$ (step 16) and in the corresponding step of the algorithm $\mathcal{B}_5$.

References

[ABC+12] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, abhi shelat, and B. Waters. Computing on authenticated data. In *Proceedings of the 9th Theory of Cryptography Conference*, pages 169–191, 2012.

[Bar01] B. Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 315–334, 2018.

[BBF19] D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Advances in Cryptology – CRYPTO '19*, pages 561–586, 2019.

[BCG+14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[BdM93] J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – EUROCRYPT '93*, pages 274–285, 1993.

[BGV11] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *Advances in Cryptology – CRYPTO '11*, pages 111–131, 2011.

[Bic17] M. Bichler. Market Design: A Linear Programming Approach to Auctions and Matching. Cambridge University Press, 2017.

[BP97] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology – EUROCRYPT '97*, pages 480–494, 1997.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS07] M. Bellare and S. Shoup. Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In *Proceedings of the 10th International Conference on Theory and Practice of Public-Key Cryptography*, pages 201–216, 2007.

[BSW06] D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Proceedings of the 9th International Conference on Theory and Practice of Public-Key Cryptography*, pages 229–240, 2006.

[But14] V. Buterin. Ethereum: A next-generation smart contract and decentralized application platform. Available at `https://ethereum.org/en/whitepaper/`, 2014.

[But17] V. Buterin. The stateless client concept, 2017. Available at `https://ethresear.ch/t/the-stateless-client-concept/172`.

[CDV06] D. Catalano, Y. Dodis, and I. Visconti. Mercurial commitments: Minimal assumptions and efficient constructions. In *Proceedings of the 3rd Theory of Cryptography Conference*, pages 120–144, 2006.

[CF01] R. Canetti and M. Fischlin. Universally composable commitments. In *Advances in Cryptology – CRPYTO '01*, pages 19–40, 2001.

[CF13] D. Catalano and D. Fiore. Vector commitments and their applications. In *Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography*, pages 55–72, 2013.

[CFG+20] M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In *Advances in Cryptology – ASIACRYPT '20*, pages 3–35, 2020.

[CHL+13] M. Chase, A. Healy, A. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. *Journal of Cryptology*, 26(2):251–279, 2013.

[CIO98] G. D. Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitment. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 141–150, 1998.

[CKO+01] G. D. Crescenzo, J. Katz, R. Ostrovsky, and A. D. Smith. Efficient and non-interactive non-malleable commitment. In *Advances in Cryptology – EUROCRYPT '01*, pages 40–59, 2001.

[CL02] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology – CRYPTO '02*, pages 61–76, 2002.

[COS+17] M. Ciampi, R. Ostrovsky, L. Siniscalchi, and I. Visconti. Four-round concurrent non-malleable commitments from one-way functions. In *Advances in Cryptology – CRYPTO '17*, pages 127–157, 2017.

[CRF+11] D. Catalano, M. D. Raimondo, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. *IEEE Transaction on Information Theory*, 57(4):2488–2502, 2011.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DG03] I. Damgard and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 426–437, 2003.

[DKN+04] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In *Advances in Cryptology – EUROCRYPT '04*, pages 609–626, 2004.

[FF00] M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In *Advances in Cryptology – CRYPTO '00*, pages 413–431, 2000.

[Fis01] M. Fischlin. Trapdoor commitment schemes and their applications. PhD Thesis, University of Frankfurt (available at `https://www.math.uni-frankfurt.de/~dmst/research/phdtheses/mfischlin.dissertation.2001.html`), 2001.

[FVY14] C. Fromknecht, D. Velicanu, and S. Yakoubov. A decentralized public key infrastructure with identity retention. Cryptology ePrint Archive, Report 2014/803, 2014.

[GLO+12] V. Goyal, C.-K. Lee, R. Ostrovsky, and I. Visconti. Constructing non-malleable commitments: A black-box approach. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 51–60, 2012.

[GM06] R. Gennaro and S. Micali. Independent zero-knowledge sets. In *Proceedings of the 33th International Colloquium on Automata, Languages and Programming*, pages 34–45, 2006.

[GMY03] J. A. Garay, P. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In *Advances in Cryptology – EUROCRYPT '03*, pages 177–194, 2003.

[GPR16] V. Goyal, O. Pandey, and S. Richelson. Textbook non-malleable commitments. In *Proceedings of the 48th annual ACM Symposium on Theory of Computing*, pages 1128–1141, 2016.

[GR97] R. Gennaro and P. Rohatgi. How to sign digital streams. In *Advances in Cryptology – CRYPTO '97*, pages 180–197, 1997.

[GRW+20] S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. Pointproofs: Aggregating proofs for multiple vector commitments. In *Proceedings of the 27th ACM Conference on Computer and Communications Security*, pages 2007–2023, 2020.

[HIL+99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[Khu17] D. Khurana. Round optimal concurrent non-malleability from polynomial hardness. In *Proceedings of the 15th Theory of Cryptography Conference*, pages 139–171, 2017.

[Kil92] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732, 1992.

[KSS+16] J. Krupp, D. Schröder, M. Simkin, D. Fiore, G. Ateniese, and S. Nürnberger. Nearly optimal verifiable data streaming. In *Proceedings of the 19th International Conference on Practice and Theory in Public-Key Cryptography*, pages 417–445, 2016.

[Lam79] L. Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[LM19] R. W. F. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In *Advances in Cryptology – CRYPTO '19*, pages 530–560, 2019.

[LP09] H. Lin and R. Pass. Non-malleability amplification. In *Proceedings of the 41st annual ACM Symposium on Theory of Computing*, pages 189–198, 2009.

[LP11] H. Lin and R. Pass. Constant-round non-malleable commitments from any one-way function. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 705–714, 2011.

[LPV08] H. Lin, R. Pass, and M. Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *Proceedings of the 5th Theory of Cryptography Conference*, pages 571–588, 2008.

[LY10] B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *Proceedings of the 7th Theory of Cryptography Conference*, pages 499–517, 2010.

[Mer87] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology – CRYPTO '87*, pages 369–378, 1987.

[MGG+13] I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *IEEE Symposium on Security and Privacy*, pages 397–411, 2013.

[Mic94] S. Micali. CS proofs. In *Proceedings of the 35th Annual IEEE Symposium on the Foundations of Computer Science*, pages 436–453, 1994.

[MND+04] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. G. Stubblebine. A general model for authenticated data structures. *Algorithmica*, 39(1):21–24, 2004.

[MRK03] S. Micali, M. O. Rabin, and J. Kilian. Zero-knowledge sets. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 2003.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available at `https://bitcoin.org/bitcoin.pdf`, 2008.

[Nao91] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[Ngu05] L. Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology – CT-RSA '05*, pages 275–292, 2005.

[NN98] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, 1998.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[OWW+20] A. Ozdemir, R. Wahby, B. Whitehat, and D. Boneh. Scaling verifiable computation using efficient set accumulators. In *Proceedings of the 29th USENIX Security Symposium*, pages 2075–2092, 2020.

[PPV08] O. Pandey, R. Pass, and V. Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology – CRYPTO '08*, pages 57–74, 2008.

[PR05] R. Pass and A. Rosen. Concurrent non-malleable commitments. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 563–572, 2005.

[PR08] R. Pass and A. Rosen. New and improved constructions of nonmalleable cryptographic protocols. *SIAM Journal on Computing*, 38(2):702–752, 2008.

[PW10] R. Pass and H. Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *Advances in Cryptology – EUROCRYPT '10*, pages 638–655, 2010.

[Rey01] L. Reyzin. Zero-knowledge with public keys. PhD Thesis, Massachusetts Institute of Technology (available at `https://www.cs.bu.edu/~reyzin/phd-thesis.html`), 2001.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[STS99] T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash. In *Advances in Cryptology – CRYPTO '99*, pages 555–572, 1999.

[SvDJ+12] E. Stefanov, M. van Dijk, A. Jules, and A. Opera. Iris: a scalable cloud file system with efficient integrity checks. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 229–238, 2012.

[TAB+20] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In *Proceedings of the 12th International Conference on Security and Cryptography for Networks*, pages 45–64, 2020.

[Tod16] P. Todd. Making UTXO set growth irrelevant with low-latency delayed TXO commitments, 2016. Available at `https://petertodd.org/2016/delayed-txo-commitments`.

[Wee10] H. Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2010.

A Non-Malleability of Merkle Trees in the Random-Oracle Model

In this section we show that a Merkle tree is a non-malleable static vector commitment scheme in the random-oracle model. Our notion of a non-malleable vector commitment scheme (recall Definition 3.1) naturally extends to the random-oracle model by allowing the scheme itself and the adversary $\mathcal{A}$ to issue random-oracle queries[10], whereas the simulator $\mathcal{S}$, the distribution $\mathcal{D}$ and the distinguisher $\mathcal{R}$ remain standard-model algorithms.

[10]Thus the output of $\mathsf{Real}_{\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$ is a random variable also over the randomness of the oracle.

Notation. For integers $t \geq 1$ and $i \in \{0, \ldots, 2^\ell - 1\}$ we denote by $\langle i \rangle_\ell$ the ℓ-bit binary representation of i. For a string $s \in \{0,1\}^*$ we denote by $\mathsf{sibling}(s)$ the string obtained from s by flipping its least-significant bit, by $\mathsf{parent}(s)$ the string obtained from s by chopping off its least-significant bit (that is, $\mathsf{parent}(s)$ is one bit shorter than s), and by $\mathsf{LSB}(s)$ its least-significant bit.

The construction. Let $\mathsf{H} = \{\mathsf{H}_\lambda\}_{\lambda \in \mathbb{N}}$ be a hash function such that each $\mathsf{H}_\lambda : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ for every $\lambda \in \mathbb{N}$. In what follows, for simplicity of presentation we assume that $q = 2^d$ for some integer d.

A tree-based vector commitment scheme $\mathbf{tree}\mathcal{VC}$

$\underline{\mathsf{Setup}(1^\lambda, q)}$:

1. Output $\perp$.

$\underline{\mathsf{Commit}(1^\lambda, \mathsf{crs}, (x_1, \ldots, x_q))}$:

1. Sample $k \leftarrow \{0,1\}^\lambda$, and for each $i \in [q]$ compute $r_i = \mathsf{H}(k \| \langle i \rangle_\lambda)$.
2. For each $i \in [q]$ let $h_{\langle 2i-2 \rangle_{\log(q)+1}} = x_i$ and $h_{\langle 2i-1 \rangle_{\log(q)+1}} = r_i$.
3. For each $\ell = \log(q), \ldots, 1$ and $i = 0, \ldots 2^\ell - 1$ compute $h_{\langle i \rangle_\ell} = \mathsf{H}(h_{\langle i \rangle_\ell \| 0} \| h_{\langle i \rangle_\ell \| 1})$.
4. Output $(\mathsf{vcom}, \mathsf{st})$ where $\mathsf{vcom} = \mathsf{H}(h_0 \| h_1)$ and $\mathsf{st} = k$.

$\underline{\mathsf{Open}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, \mathsf{st}, i)}$:

1. Let $k = \mathsf{st}$ and for each $j \in [q]$ compute $r_j = \mathsf{H}(k \| \langle j \rangle_\lambda)$.
2. For each $j \in [q]$ let $h_{\langle 2j-2 \rangle_{\log(q)+1}} = x_i$ and $h_{\langle 2j-1 \rangle_{\log(q)+1}} = r_i$.
3. For each $\ell = \log(q), \ldots, 1$ and $i = 0, \ldots 2^\ell - 1$ compute $h_{\langle i \rangle_\ell} = \mathsf{H}(h_{\langle i \rangle_\ell \| 0} \| h_{\langle i \rangle_\ell \| 1})$.
4. $\pi_{\log(q)+1} = r_i$ and $i_{\log(q)+1} = i$.
5. For each $\ell = \log(q), \ldots, 1$:
   (a) Let $i_\ell = \mathsf{parent}(i_{\ell-1})$.
   (b) Compute $\pi_\ell = h_{\mathsf{sibling}(\langle i_\ell \rangle_\ell)}$.
6. Output $\pi = \pi_{\log(q)+1} \| \cdots \| \pi_1$.

$\underline{\mathsf{Verify}(1^\lambda, \mathsf{crs}, \mathsf{vcom}, i, x, \pi)}$:

1. Parse π as $\pi_{\log(q)+1} \| \cdots \| \pi_0$.
2. Let $h^{\mathsf{L}}_{\log(q)+1} = x$, $h^{\mathsf{R}}_{\log(q)+1} = \pi_{\log(q)+1}$ and $i_{\log(q)} = \mathsf{parent}(\langle i \rangle_{\log(q)+1})$.
3. For each $\ell = \log(q), \ldots, 1$:
   (a) Compute $v_\ell = \mathsf{H}(h^{\mathsf{L}}_{\ell+1} \| h^{\mathsf{R}}_{\ell+1})$.
   (b) If $\mathsf{LSB}(i_\ell) = 0$, then let $h^{\mathsf{L}}_\ell = v_\ell$ and $h^{\mathsf{L}}_\ell = \pi_\ell$. Otherwise, let $h^{\mathsf{L}}_\ell = \pi_\ell$ and $h^{\mathsf{L}}_\ell = v_\ell$.
   (c) Let $i_{\ell-1} = \mathsf{parent}(i_\ell)$.
4. If $\mathsf{H}(h^{\mathsf{L}}_1 \| h^{\mathsf{R}}_1) = \mathsf{vcom}$ then output 1, and otherwise output 0.

Theorem A.1. *If H is modeled as a random oracle, then for any algorithm $\mathcal{A}$ issuing at most $p = p(\lambda)$ queries to the oracle, there exists an algorithm $\mathcal{S}$ whose running time is polynomial in that*

of $\mathcal{A}$, such that for every algorithm $\mathcal{R}$ it holds that

$$\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda) < \frac{p^2 + 10 \cdot p \cdot q + 24 \cdot q^2 + 1}{2^\lambda - p - 4 \cdot q}$$

for every $\lambda \in \mathbb{N}$.

Proof. Let $\mathcal{A}$ be an algorithm issuing at most $p = p(\lambda)$ oracle queries and let $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ be a valid distribution over $\left\{\left(\{0,1\}^\lambda\right)^q\right\}_{\lambda\in\mathbb{N}}$. Assume without loss of generality that $\mathcal{A}$ does not issue the same query more than once, and that when receiving a proof π for the ith element x_i (in Step 6 or 7 of the experiment $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$), it executes $\mathsf{Verify}^{\mathsf{H}}(1^\lambda, \perp, \mathsf{vcom}, i, x_i, \pi_i)$ where vcom is the commitment which $\mathcal{A}$ receives as input in Step 5 of the experiment. Note that this adds at most $2q-1$ oracle queries to $\mathcal{A}$, as this is the number of inner nodes in the tree. Further assume without loss of generality, that for each $j \in \mathcal{J}$ (where $\mathcal{J}$ is the set outputted by $\mathcal{A}$ in Step 6 of the experiment), $\mathcal{A}$ executes $\mathsf{Verify}^{\mathsf{H}}(1^\lambda, \perp, \widehat{\mathsf{vcom}}, j, \widehat{x_j}, \widehat{\pi_j})$, where $\widehat{\mathsf{vcom}}$ is the commitment outputted by $\mathcal{A}$ in Step 6 of $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$, and for each $j \in \mathcal{J}$, $\widehat{x_j}$ and $\widehat{\pi_j}$ are the element and the corresponding proof outputted by $\mathcal{A}$ in Step 7. This also adds at most $2q-1$ oracle queries to $\mathcal{A}$.

Consider the following simulator $\mathcal{S}$:

The Simulator $\mathcal{S}$

Input: The security parameter $\lambda \in \mathbb{N}$ and a description of the distribution $\mathcal{D}$.

1. Construct the tree, except for the leaves which are left children (which hold the values $x_1, \ldots, x_q$):
   (a) Sample $k \leftarrow \{0,1\}^\lambda$, and for $i \in [q]$, sample $r_i \leftarrow \{0,1\}^\lambda$ and set $h_{\langle 2i-1\rangle_{\log(q)+1}} := r_i$.
   (b) For each $\ell = \log(q), \ldots, 1$ and for each $i = 0, \ldots, 2^\ell - 1$ sample $h_{\langle i\rangle_\ell} \leftarrow \{0,1\}^\lambda$.
   (c) Sample $\mathsf{vcom} :\leftarrow \{0,1\}^\lambda$.
2. Compute a proof π_i for each location $i \in [q]$:
   (a) $\pi_{i,\log(q)+1} := h_{\langle 2i-1\rangle_{\log(q)+1}}$ and $i_{\log(q)+1} := i$.
   (b) For $\ell = \log(q), \ldots, 1$:
      i. $i_\ell := \mathsf{parent}(i_{\ell-1})$.
      ii. $\pi_{i,\ell} := h_{\mathsf{sibling}(\langle i_\ell\rangle_\ell)}$.
   (c) $\pi_i := \pi_{i,\log(q)+1} \| \cdots \| \pi_{i,1}$.
3. Invoke $(\mathcal{I}, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, \perp, \mathsf{vcom})$, while simulating the oracle H to $\mathcal{A}$, answering each query $z \in \{0,1\}^{2m}$ for some $m \in \mathbb{N}$ as follows (ignore odd-length queries):
   - If $z = k \| \langle i\rangle_\lambda$ for some $i \in [q]$, then reply with $\mathsf{H}(z) = r_i$.
   - If $z = h_{s\|0} \| h_{s\|1}$ for some $t \in [\log(q)]$ and $s \in \{0,1\}^t$, then reply with $\mathsf{H}(z) = h_s$.
   - Otherwise, sample $y \leftarrow \{0,1\}^m$ and reply with $\mathsf{H}(z) = y$.

   Record all query-answer pairs in a set $\mathcal{Q}$.
4. Set $\mathsf{st}_\mathcal{S} := (\mathsf{vcom}, \mathsf{st}_\mathcal{A}, \mathcal{Q})$, output $(\mathcal{I}, \mathsf{st}_\mathcal{S})$ as the output of the simulator in Step 2 of the experiment $\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{S},R,\mathcal{D}}(\lambda)$, and receive $(x_i)_{i\in\mathcal{I}}$ in response from the challenger. If for any $i \in \mathcal{I}$, there exists a value $y \in \{0,1\}^\lambda$ such that $(x_i \| r_i, y) \in \mathcal{Q}$, then output $\perp$ and terminate.
5. Sample $(\widetilde{x_j})_{j\in\overline{\mathcal{I}}} \leftarrow \mathcal{D}|(\mathcal{I}, (x_i)_{i\in\mathcal{I}})$. If for any $j \in \overline{\mathcal{I}}$, there exists a value $y \in \{0,1\}^\lambda$ such that $(\widetilde{x_j} \| r_j, y) \in \mathcal{Q}$, then output $\perp$ and terminate.
6. Invoke $\left(\widehat{\mathsf{vcom}}, \mathcal{J}, \mathsf{st}_\mathcal{A}\right) \leftarrow \mathcal{A}^{\mathsf{H}}\left(\mathsf{st}_\mathcal{A}, (x_i)_{i\in\mathcal{I}}, (\pi_i)_{i\in\mathcal{I}}\right)$, while simulating the oracle H to $\mathcal{A}$, answering each query $z \in \{0,1\}^{2m}$ for some $m \in \mathbb{N}$ as in Step 3 with the following modifications:

- If $z = x_i \| r_i$ for some $i \in \mathcal{I}$, then reply with $\mathsf{H}(z) = h_{\langle \mathsf{parent}(i) \rangle_{\log(q)}}$.
- If $z = \widetilde{x}_i \| h_{r_i}$ for some $i \in \overline{\mathcal{I}}$, then reply with $\mathsf{H}(z) = h_{\langle \mathsf{parent}(i) \rangle_{\log(q)}}$.

7. Invoke $((\widehat{x}_j)_{j\in\mathcal{J}}, (\widehat{\pi}_j)_{j\in\mathcal{J}}) \leftarrow \mathcal{A}^{\mathsf{H}}\left(\mathsf{st}_{\mathcal{A}}, (\widetilde{x}_i)_{i\in\overline{\mathcal{I}}}, (\pi_i)_{i\in\overline{\mathcal{I}}}\right)$, simulating the oracle H to $\mathcal{A}$ as in Step 6.
8. Output $(\mathcal{J}, (\widehat{x}_j)_{j\in\mathcal{J}})$ as the simulator's output in step 3 of the experiment $\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{S},R,\mathcal{D}}(\lambda)$.

Let $\mathcal{R}$ be an algorithm. We now turn to analyze $\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda)$. Let $p' = p + 4q - 2$, and let $\mathsf{Collision}$ denote the event, defined over the experiment $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$, in which $\mathcal{A}$ issues two oracle queries $z, z' \in \{0,1\}^\lambda$ such that $\mathsf{H}(z) = \mathsf{H}(z')$. By a standard birthday bound argument,

$$\Pr[\mathsf{Collision}] \leq \left(p'\right)^2 \cdot 2^{-\lambda}. \tag{A.1}$$

For each $j \in [q]$, let Guess_j denote the event in which $\mathcal{A}$ outputs $\mathcal{J}$ and $\widehat{\mathsf{vcom}}$ without having queried (before outputting $\mathcal{J}$ and $\widehat{\mathsf{vcom}}$) a complete proof in the tree whose root is $\widehat{\mathsf{vcom}}$ for some element $\widehat{x}_j$ in the j-th location. That is, the complementing event $\overline{\mathsf{Guess}_j}$ is the event in which there is a value $\widehat{x}_j \in \{0,1\}^\lambda$ and a proof $\widehat{\pi}_j \in \{0,1\}^{\lambda\cdot(\log(q)+1)}$ such that:

- $\mathsf{VC.Verify}^{\mathsf{H}}(1^\lambda, \bot, \widehat{\mathsf{vcom}}, j, \widehat{x}_j, \widehat{\pi}_j) = 1$; and
- $\mathcal{A}$ issued all queries made by the computation $\mathsf{Verify}^{\mathsf{H}}(1^\lambda, \bot, \widehat{\mathsf{vcom}}, j, \widehat{x}_j, \widehat{\pi}_j)$ before outputting $\mathcal{J}$ and $\widehat{\mathsf{vcom}}$.

Denote by Guess the event in which there exists at least one index $j \in \mathcal{J}$ for which Guess_j holds. Assume without loss of generality that $\mathcal{R}$ always outputs 0 on inputs of the form $((x_1, \ldots, x_q), (\bot)^q, \mathcal{I})$ (observe that any distinguisher can be transformed to a distinguisher for which this holds without affecting its advantage). Note that

$$\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 | \mathsf{Guess}\right] \leq p' \cdot 2^{-\lambda}. \tag{A.2}$$

This is the case, since the event Guess occurring means that there exists a $j \in \mathcal{J}$ for which Guess_j holds, and (by our assumption) the event $\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1$ is contained inside the event in which $\mathsf{Verify}^{\mathsf{H}}(1^\lambda, \bot, \widehat{\mathsf{vcom}}, j, \widehat{x}_j, \widehat{\pi}_j) = 1$. But in order for the verification $\mathsf{Verify}^{\mathsf{H}}(1^\lambda, \bot, j, \widehat{x}_j, \widehat{\pi}_j)$ to pass conditioned on Guess_j, it must be the case that exists some value $y \in \{0,1\}^\lambda$, determined by the view of $\mathcal{A}$ when outputting $\widehat{\mathsf{vcom}}$, and there exists a $z \in \{0,1\}^\lambda$ such that $\mathsf{H}(z)$ was queried by $\mathcal{A}$ *after* outputting $\widehat{\mathsf{vcom}}$ (recall that $\mathcal{A}$ does not issue the same query twice); and $\mathsf{H}(z) = y$. For each query z' made by $\mathcal{A}$ after outputting $\widehat{\mathsf{vcom}}$, it holds that $\Pr[\mathsf{H}(z) = y] = 2^{-\lambda}$. Eq. (A.2) then follows by a union bound over all queries made by $\mathcal{A}$ after outputting $\widehat{\mathsf{vcom}}$.

Let kHit denote the event in which $\mathcal{A}$ queries the oracle for $k \| \langle i \rangle_\lambda$ for some $i \in [q]$. For each $j \in [p']$, let kHit_j denote the event in which a query of the form $k \| \langle i \rangle_\lambda$ (for some $i \in [q]$) is queried within the first j queries of $\mathcal{A}$, and note that

$$\begin{aligned}
\Pr[\mathsf{kHit}] &= \Pr[\mathsf{kHit}_1] + \sum_{j=2}^{p'} \Pr\left[\mathsf{kHit}_j \middle| \overline{\mathsf{kHit}_{j-1}}\right] \\
&\leq \frac{1}{2^\lambda} + \sum_{j=2}^{p'} \frac{1}{2^\lambda - j + 1} \qquad (A.3) \\
&\leq \frac{p'}{2^\lambda - p' + 1}, \qquad (A.4)
\end{aligned}$$

where inequality (A.3) follow from the fact that if $\overline{\mathsf{kHit}_{j-1}}$ occurs, then conditioned on the view of $\mathcal{A}$ before the j-th query, the value of k is uniformly distributed in a set of size at least $2^\lambda - j + 1$.

Let rHit denote the event in which for some $i \in \overline{\mathcal{I}}$, $\mathcal{A}$ queries the oracle with a query $y \| r_i$ *before* receiving the proofs $(\pi_i)_{i \in \overline{\mathcal{I}}}$. By total probability, it holds that:

$$\begin{aligned}
\Pr[\mathsf{rHit}] &\leq \Pr\left[\mathsf{rHit} \middle| \overline{\mathsf{kHit}}\right] + \Pr[\mathsf{kHit}] \\
&\leq \Pr\left[\mathsf{rHit} \middle| \overline{\mathsf{kHit}}\right] + \frac{p'}{2^\lambda - p' + 1} \\
&\leq \frac{q \cdot p'}{2^\lambda - p' + 1} + \frac{p'}{2^\lambda - p' + 1} && \text{(A.5)} \\
&\leq \frac{2 \cdot q \cdot p'}{2^\lambda - p' + 1}. && \text{(A.6)}
\end{aligned}$$

Inequality (A.5) follows by a similar argument to that used to derive inequality (A.4).

Putting everything together, it holds that

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda) \\
&\quad = \left|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1\right] - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)\right) = 1\right]\right| \\
&\quad \leq \Big|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \middle| \overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right] && \text{(A.7)} \\
&\qquad - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)\right) = 1 \middle| \overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right]\Big| \\
&\qquad \cdot \Pr\left[\overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right] \\
&\qquad + \Big|\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge (\mathsf{Collision} \vee \mathsf{Guess} \vee \mathsf{rHit})\right] \\
&\qquad\quad - \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge (\mathsf{Collision} \vee \mathsf{Guess} \vee \mathsf{rHit})\right]\Big| \\
&\quad \leq \max\left\{ \begin{array}{l} \Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge (\mathsf{Collision} \vee \mathsf{Guess} \vee \mathsf{rHit})\right], \\ \Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge (\mathsf{Collision} \vee \mathsf{Guess} \vee \mathsf{rHit})\right] \end{array} \right\} && \text{(A.8)}
\end{aligned}$$

where inequality (A.7) is by total probability, and follows also from the fact that the event $\overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}$ is determined by the view of $\mathcal{A}$, which is identically distributed in both $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},R,\mathcal{D}}(\lambda)$ and in the experiment simulated by $\mathcal{S}$ to $\mathcal{A}$. In particular, $\Pr\left[\overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right]$ is equal in both experiments. In order to prove inequality (A.8), we argue that

$$\begin{aligned}
&\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \middle| \overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right] \\
&\qquad = \Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{S},\mathcal{D}}(\lambda)\right) = 1 \middle| \overline{\mathsf{Collision}} \wedge \overline{\mathsf{Guess}} \wedge \overline{\mathsf{rHit}}\right].
\end{aligned}$$

This is true since conditioned on $\overline{\mathsf{Colliosion}} \wedge \overline{\mathsf{Guess}}$, the view of $\mathcal{A}$ when outputting $\widehat{\mathsf{vcom}}$ uniquely determines the values $(\widehat{x}_j)_{j \in \mathcal{J}}$ which $\mathcal{A}$ can output in Step 7 of $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$. Moreover, conditioned on $\overline{\mathsf{rHit}}$, the view of $\mathcal{A}$ after Step 7 is independent of the values $(x_i)_{i \in \overline{\mathcal{I}}}$. Hence, the joint distribution of $\left((x_i)_{i \in [q]}, (\widehat{x}_j)_{j \in \mathcal{J}}\right)$ in the experiment simulated by $\mathcal{S}$ to $\mathcal{A}$ is equal to the joint distribution of $\left((x_i)_{i \in [q]}, (\widehat{x}_j)_{j \in \mathcal{J}}\right)$ in $\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)$.

By the rule of replacement, the union bound and total probability, it holds that

$$\begin{aligned}
&\Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 \wedge (\mathsf{Collision} \vee \mathsf{Guess} \vee \mathsf{rHit})\right] \\
&\qquad \leq \Pr\left[\mathcal{R}\left(\mathsf{Real}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right) = 1 | \mathsf{Guess}\right] + \Pr[\mathsf{Collision}] + \Pr[\mathsf{rHit}] \\
&\qquad \leq \left(p'\right)^2 \cdot 2^{-\lambda} + 2^{-\lambda} + \frac{2 \cdot q \cdot p'}{2^\lambda - p' + 1} \\
&\qquad \leq \frac{\left(p'\right)^2 + 1 + 2 \cdot q \cdot p'}{2^\lambda - p' + 1}.
\end{aligned}$$

The same analysis can be used to prove that

$$\Pr\left[\mathcal{R}\left(\mathsf{Ideal}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{D}}(\lambda)\right)=1\wedge(\mathsf{Collision}\vee\mathsf{Guess}\vee\mathsf{rHit})\right]\leq\frac{(p')^2+1+2\cdot q\cdot p'}{2^\lambda-p'+1}.$$

Plugging in $p'=p+4\cdot q-2<p+4\cdot q$ to Eq. (A.8), we get that

$$\mathbf{Adv}^{\mathsf{NM}}_{\mathsf{tree}\mathcal{VC},q,\mathcal{A},\mathcal{S},\mathcal{R},\mathcal{D}}(\lambda)<\frac{p^2+10\cdot p\cdot q+24\cdot q^2+1}{2^\lambda-p-4\cdot q}.$$

This concludes the proof of Theorem A.1. ■

B Constructions of Locally-Equivocable Commitments with All-But-One Binding

B.1 A Generic Construction

Our generic construction of a locally-equivocable commitment scheme with all-but-one binding relies on any non-interactive equivocable commitment scheme (see Section 2.1). Such a scheme can be constructed based on the minimal assumption that one-way functions exist [Nao91, CIO98], and therefore this holds for our generic scheme as well. It should be noted, however, that our generic scheme is mainly of theoretical significance (e.g., due to the somewhat impractical yet polynomial length of its common-reference string), and the reader is referred to our number-theoretic constructions for practical alternatives.

Let $\mathcal{EQ}=(\mathsf{EQ.Setup},\mathsf{EQ.Commit},\mathsf{EQ.Decommit},\mathsf{EQ.Equiv}_1,\mathsf{EQ.Equiv}_2)$ be an equivocable commitment scheme over a domain $\mathcal{X}=\{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$. For any polynomial $t=t(\lambda)$ we construct a locally-equivocable commitment scheme with all-but-one binding $\mathcal{LE}=(\mathsf{LE.Setup},\mathsf{LE.Commit},\mathsf{LE.Decommit},\mathsf{LE.AltSetup},\mathsf{LE.Equiv}_1,\mathsf{LE.Equiv}_2)$ over the domain $\mathcal{X}$ and the tag space $\mathcal{T}=\{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{T}_\lambda=\{0,1\}^{t(\lambda)}$ for any $\lambda\in\mathbb{N}$.

A locally-equivocable commitment scheme $\mathcal{LE}$

$\mathsf{LE.Setup}(1^\lambda,q)$:

1. For each $(i,j,b)\in[q]\times[t]\times\{0,1\}$ sample $\mathsf{crs}_{i,j,b}\leftarrow\mathsf{EQ.Setup}(1^\lambda)$.
2. Output $\mathsf{crs}=(\mathsf{crs}_{i,j,b})_{(i,j,b)\in[q]\times[t]\times\{0,1\}}$.

$\mathsf{LE.Commit}(1^\lambda,\mathsf{crs},x,i,\tau)$:

1. Parse crs as $(\mathsf{crs}_{i,j,b})_{(i,j,b)\in[q]\times[t]\times\{0,1\}}$.
2. For each $j\in[t]$ compute $(c_j,d_j)\leftarrow\mathsf{EQ.Commit}(1^\lambda,\mathsf{crs}_{i,j,\tau_j},x)$.
3. Output (c,d) where $c=c_1\|\cdots\|c_t$ and $d=d_1\|\cdots\|d_t$.

$\mathsf{LE.Decommit}(1^\lambda,\mathsf{crs},c,d,i,\tau)$:

1. Parse crs as $(\mathsf{crs}_{i,j,b})_{(i,j,b)\in[q]\times[t]\times\{0,1\}}$, c as $c_1\|\cdots\|c_t$ and d as $d_1\|\cdots\|d_t$.
2. For each $j\in[t]$ compute $x_j=\mathsf{EQ.Decommit}(1^\lambda,\mathsf{crs}_{i,j,\tau_j},c_j,d_j)$.
3. If $x_1=\cdots=x_t\neq\perp$ then output x_1 and otherwise output $\perp$.

$\mathsf{LE.AltSetup}(1^\lambda,q)$:

1. For each $(i,j) \in [q] \times [t]$ sample $\mathsf{crs}_{i,j} \leftarrow \mathsf{EQ.Setup}(1^\lambda)$.
2. Output $\mathsf{st}_0 = (\mathsf{crs}_{i,j})_{(i,j)\in[q]\times[t]}$.

$\mathsf{LE.Equiv}_1(1^\lambda, \mathsf{st}_0, q, \tau)$:

1. Parse st_0 as $(\mathsf{crs}_{i,j})_{(i,j)\in[q]\times[t]}$.
2. For each $(i,j) \in [q] \times [t]$ compute $(\widehat{\mathsf{crs}}_{i,j,\tau_j}, \widehat{c_{i,j}}, \mathsf{st}_{i,j}) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$ and set $\widehat{\mathsf{crs}}_{i,j,1-\tau_j} = \mathsf{crs}_{i,j}$.
3. For each $i \in [q]$ let $\widehat{c_i} = \widehat{c_{i,1}} \| \cdots \| \widehat{c_{i,t}}$ and $\mathsf{st}_i = \mathsf{st}_{i,1} \| \cdots \| \mathsf{st}_{i,t}$.
4. Let $\widehat{\mathsf{crs}} = (\widehat{\mathsf{crs}}_{i,j,b})_{(i,j,b)\in[q]\times[t]\times\{0,1\}}$ and $\mathsf{st} = \mathsf{st}_1 \| \cdots \| \mathsf{st}_q$.
5. Output $(\widehat{\mathsf{crs}}, \widehat{c_1}, \ldots, \widehat{c_q}, \mathsf{st})$.

$\mathsf{LE.Equiv}_2(1^\lambda, x, i, \mathsf{st}, \tau)$:

1. Parse st as $\mathsf{st}_1 \| \cdots \| \mathsf{st}_q$ and st_i as $\mathsf{st}_{i,1} \| \cdots \| \mathsf{st}_{i,t}$.
2. For each $j \in [t]$ compute $\widehat{d_{i,j}} = \mathsf{EQ.Equiv}_2(1^\lambda, x, \mathsf{st}_{i,j})$.
3. Output $\widehat{d_i} = \widehat{d_{i,1}} \| \cdots \| \widehat{d_{i,t}}$.

The following theorem establishes the security of the above generic construction:

Theorem B.1. *Let $\mathcal{EQ}$ be an equivocable commitment scheme over a domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$. Then, for any polynomial $t(\lambda)$, the scheme $\mathcal{LE}$ is a locally-equivocable commitment scheme with all-but-one binding over the domain $\mathcal{X}$ and the tag space $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda\in\mathbb{N}}$, where $\mathcal{T}_\lambda = \{0,1\}^{t(\lambda)}$ for any $\lambda \in \mathbb{N}$.*

Proof. In order to show that the scheme $\mathcal{LE}$ is locally equivocable and all-but-one binding (recall Definitions 4.1 and 4.2), we first observe that the local equivocability of $\mathcal{LE}$ follows immediately from the equivocability of $\mathcal{EQ}$. Specifically, the equivocation correctness of $\mathcal{LE}$ follows directly from that of $\mathcal{EQ}$, and the equivocation indistinguishability of $\mathcal{LE}$ follows via a standard hybrid argument from that of $\mathcal{EQ}$. We therefore focus on proving that $\mathcal{LE}$ is all-but-one binding.

Let $q = q(\lambda)$ be a polynomial, and let $\mathcal{A}$ be a probabilistic polynomial-time algorithm that participates in the experiment $\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{A}}(\lambda)$. We show that there exists a probabilistic polynomial-time algorithm $\mathcal{B}$ such that

$$\mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{A}}(\lambda) \leq q \cdot t \cdot \mathbf{Adv}^{\mathsf{Bind}}_{\mathcal{EQ},\mathcal{B}}(\lambda)$$

for every $\lambda \in \mathbb{N}$. Consider the following probabilistic polynomial-time algorithm $\mathcal{B}$, which on input $(1^\lambda, \mathsf{crs})$ is defined as follows:

1. Invoke $(\tau, \mathsf{st}_\mathcal{A}) \leftarrow \mathcal{A}(1^\lambda)$.
2. Sample $i^* \leftarrow [q]$ and $j^* \leftarrow [t]$.
3. For each $i \in [q] \setminus \{i^*\}$ and for each $j \in [t]$ compute $(\widehat{\mathsf{crs}}_{i,j,\tau_j}, \widehat{c_{i,j}}, \mathsf{st}_{i,j}) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$ and sample $\widehat{\mathsf{crs}}_{i,j,1-\tau_j} \leftarrow \mathsf{EQ.Setup}(1^\lambda)$.
4. For each $j \in [t] \setminus \{j^*\}$ compute $(\widehat{\mathsf{crs}}_{i^*,j,\tau_j}, \widehat{c_{i^*,j}}, \mathsf{st}_{i^*,j}) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$ and sample $\widehat{\mathsf{crs}}_{i^*,j,1-\tau_j} \leftarrow \mathsf{EQ.Setup}(1^\lambda)$.
5. Sample $(\widehat{\mathsf{crs}}_{i^*,j^*,\tau_{j^*}}, \widehat{c_{i^*,j^*}}, \mathsf{st}_{i^*,j^*}) \leftarrow \mathsf{EQ.Equiv}_1(1^\lambda)$ and set $\widehat{\mathsf{crs}}_{i^*,j^*,1-\tau_{j^*}} = \mathsf{crs}$.
6. $(c, d, d', i', \tau') \leftarrow \mathcal{A}(\mathsf{st}_\mathcal{A}, \mathsf{st}_0, \rho)$, where $\mathsf{st}_0 = \left(\mathsf{crs}_{i,j,1-\tau_j}\right)_{(i,j)\in[q]\times[t]}$ and ρ is the concatenation of all random coins used in all invocations of $\mathsf{EQ.Equiv}_1$ in Steps $3-5$.
7. If $i' \neq i^*$ or $\tau_{j^*} = \tau'_{j^*}$, then output $\perp$ and terminate.

8. Compute $x = \mathsf{LE.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, c, d, i', \tau')$ and $x' = \mathsf{LE.Decommit}(1^\lambda, \widehat{\mathsf{crs}}, c, d', i', \tau')$, where $\widehat{\mathsf{crs}} = (\widehat{\mathsf{crs}}_{i,j,b})_{(i,j,b)\in[q]\times[t]\times\{0,1\}}$. If $x = \bot$, $x' = \bot$ or $x = x'$ then output $\bot$ and terminate.
9. Parse c as $c_1 \| \cdots \| c_j$, d as $d_1 \| \cdots \| d_j$ and d' as $d'_1 \| \cdots \| d'_j$.
10. Output $(c_{j^*}, d_{j^*}, d'_{j^*})$.

We turn to bound $\mathbf{Adv}^{\mathsf{Bind}}_{\mathcal{EQ},\mathcal{B}}(\lambda)$. Let Hit denote the event in which $i' = i^*$ or $\tau_{j^*} \neq \tau'_{j^*}$, and let $\mathsf{SuccessA}$ denote the event in which $\tau' \neq \tau$, $x \neq \bot$, $x' \neq \bot$ and $x \neq x'$ (where x and x' are as defined in Step 8). Since $\mathcal{B}$ perfectly simulates the experiment $\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{A}}(\lambda)$ to $\mathcal{A}$, it holds that

$$\Pr[\mathsf{SuccessA}] = \Pr\left[\mathsf{ABOBind}^{\mathcal{LE}}_{q,\mathcal{A}}(\lambda) = 1\right] = \mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{A}}(\lambda).$$

Whenever Hit and $\mathsf{SuccessA}$ occur, it holds that d_{j^*} and d'_{j^*} are decommitments which open c_{j^*} to x and to x', respectively, with respect to the common reference string crs given as input to $\mathcal{B}$. Moreover, in this case it holds that $x \neq \bot$, $x' \neq \bot$ and $x \neq x'$. Hence, for every $\lambda \in \mathbb{N}$ it holds that

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{Bind}}_{\mathcal{EQ},\mathcal{B}}(\lambda) &= \Pr[\mathsf{Hit} \wedge \mathsf{SuccessA}] \\
&= \Pr[\mathsf{Hit}|\mathsf{SuccessA}] \cdot \Pr[\mathsf{SuccessA}] \\
&\geq \frac{1}{q \cdot t} \cdot \mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE},q,\mathcal{A}}(\lambda), \qquad \text{(B.1)}
\end{aligned}$$

where (B.1) follows from the fact that conditioned on $\mathsf{SuccessA}$, it holds in particular that $\tau' \neq \tau$. Hence, there exists at least one index $\widetilde{j} \in [t]$ for which $\tau_{\widetilde{j}} \neq \tau'_{\widetilde{j}}$, and this index is independent of the choice of j^*. ■

B.2 An Efficient Construction Based on the Discrete Logarithm Assumption

Let $\mathsf{GroupGen}$ be a probabilistic polynomial-time group-generation algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and outputs a triplet $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is a cyclic group of order p that is generated by g, and p is a λ-bit prime number. The following construction of a locally-equivocable commitment scheme with all-but-one binding is based on the hardness of the discrete logarithm problem relative to $\mathsf{GroupGen}$. The scheme's domain space and tag space are both $\mathbb{Z}_p$ (they can both be set, for example, to $\{0,1\}^{\lambda-1}$ when injectively embedded into $\mathbb{Z}_p$ in order to depend only on the security parameter λ).

A locally-equivocable commitment scheme $\mathcal{LE}_{\mathsf{DL}}$

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{Setup}(1^\lambda, q)$:

1. Sample $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $x_2, x_3 \leftarrow \mathbb{Z}_p$.
2. Let $g_1 = g$, $g_2 = g^{x_2}$ and $g_3 = g^{x_3}$.
3. Output $\mathsf{crs} = (\mathbb{G}, p, g_1, g_2, g_3)$.

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{Commit}(1^\lambda, \mathsf{crs}, x, i, \tau)$:

1. Parse crs as $(\mathbb{G}, p, g_1, g_2, g_3)$.
2. Sample $r \leftarrow \mathbb{Z}_p$ and compute $c = (g_1^\tau \cdot g_2)^x \cdot g_3^r$.
3. Output (c, d) where $d = (x, r)$.

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{Decommit}(1^\lambda, \mathsf{crs}, c, d, i, \tau)$:

1. Parse crs as $(\mathbb{G}, p, g_1, g_2, g_3)$ and d as (x, r).

2. Compute $c' = (g_1^{\tau} \cdot g_2)^x \cdot g_3^r$.
3. If $c' = c$ then output x and otherwise output $\perp$.

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{AltSetup}(1^\lambda, q)$:

1. Sample $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $x_3 \leftarrow \mathbb{Z}_p$.
2. Let $g_1 = g$ and $g_3 = g^{x_3}$.
3. Output $\mathsf{st}_0 = (\mathbb{G}, p, g_1, g_3)$.

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{Equiv}_1(1^\lambda, \mathsf{st}_0, q, \tau)$:

1. Parse st_0 as $(\mathbb{G}, p, g_1, g_3)$.
2. Sample $u_1, \ldots, u_q, y \leftarrow \mathbb{Z}_p$.
3. Compute $g_2 = g_1^{p-\tau} \cdot g_3^y$ and let $\widehat{\mathsf{crs}} = (\mathbb{G}, p, g_1, g_2, g_3)$.
4. For each $i \in [q]$ compute $\widehat{c}_i = g_3^{u_i}$, and let $\mathsf{st}_1 = (u_1, \ldots, u_q, y)$.
5. Output $(\widehat{\mathsf{crs}}, \widehat{c}_1, \ldots, \widehat{c}_q, \mathsf{st}_1)$.

$\mathsf{LE}_{\mathsf{DL}}.\mathsf{Equiv}_2(1^\lambda, x, i, \mathsf{st}_1, \tau)$:

1. Parse st_1 as $(u_1, \ldots, u_q, y)$.
2. Compute $\widehat{r} = u_i - y \cdot x$.
3. Output $\widehat{d} = (x, \widehat{r})$.

The following theorem establishes the security of the above construction:

Theorem B.2. *Assuming the hardness of the discrete logarithm problem relative to* $\mathsf{GroupGen}$*, the scheme* $\mathcal{LE}_{\mathsf{DL}}$ *is a locally-equivocable commitment scheme with all-but-one binding.*

Proof. In order to show that the scheme $\mathcal{LE}_{\mathsf{DL}}$ is locally equivocable and all-but-one binding (recall Definitions 4.1 and 4.2), we first observe that it satisfies the equivocation correctness requirement since

$$(g_1^{\tau} \cdot g_2)^x \cdot g_3^{\widehat{r}} = (g_1^p \cdot g_3^y)^x \cdot g_3^{u_i - y \cdot x} = g_3^{u_i} = \widehat{c}_i.$$

In addition, one can easily verify that for any algorithm $\mathcal{A}$ and for any integer $q = q(\lambda)$, the views of $\mathcal{A}$ in the experiments $\mathsf{IndParam}_{\mathcal{LE}_{\mathsf{DL}},q,\mathcal{A},0}(\lambda)$ and $\mathsf{IndParam}_{\mathcal{LE}_{\mathsf{DL}},q,\mathcal{A},1}(\lambda)$ are identically distributed, and hence the scheme $\mathcal{LE}_{\mathsf{DL}}$ satisfies the equivocation indistinguishability requirement. We therefore focus on proving that $\mathcal{LE}_{\mathsf{DL}}$ is all-but-one binding based on the hardness of the discrete logarithm problem.

Let $q = q(\lambda)$ be a polynomial, and let $\mathcal{A}$ be a probabilistic polynomial-time algorithm that participates in the experiment $\mathsf{ABOBind}_{q,\mathcal{A}}^{\mathcal{LE}_{\mathsf{DL}}}(\lambda)$. We show that there exists a probabilistic polynomial-time algorithm $\mathcal{B}$ such that

$$\mathbf{Adv}_{\mathcal{LE}_{\mathsf{DL}},q,\mathcal{A}}^{\mathsf{ABOBind}}(\lambda) \leq \Pr\left[\mathcal{B}\left(\mathbb{G}, p, g, g^z\right) = z\right]$$

for every $\lambda \in \mathbb{N}$, where $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $z \leftarrow \mathbb{Z}_p$. Consider the following probabilistic polynomial-time algorithm $\mathcal{B}$, which on input $(1^\lambda, \mathbb{G}, p, g, h)$ where $h = g^z$ is defined as follows:

1. Invoke $(\tau, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^\lambda)$.
2. Sample $u_1, \ldots, u_q, y \leftarrow \mathbb{Z}_p$.
3. Set $g_1 = g$, $g_3 = h$ and $g_2 = g_1^{p-\tau} \cdot g_3^y$.

4. Set $\mathsf{st}_0 = (\mathbb{G}, p, g_1, g_3)$, $\widehat{\mathsf{crs}} = (\mathbb{G}, p, g_1, g_2, g_3)$, and for each $i \in [q]$ set $\widehat{c}_i = g_3^{u_i}$.
5. Invoke $(c, d, d', i, \tau') \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{st}_0, (u_1, \ldots, u_q, y))$. Parse d as (x, r) and d' as (x', r').
6. If any of the following conditions holds, output $\perp$ and terminate:
   - $\tau = \tau'$.
   - $x = x'$.
   - $c \neq (g_1^{\tau'} \cdot g_2)^x \cdot g_3^r$ or $c \neq (g_1^{\tau'} \cdot g_2)^{x'} \cdot g_3^{r'}$.
7. Output $z' = (\tau' - \tau) \cdot (x - x') \cdot (y \cdot x' - y \cdot x + r' - r)^{-1}$.

We now analyze the success probability of the algorithm $\mathcal{B}$ in computing the discrete logarithm z of $h = g^z$. Let $\mathsf{SuccessA}$ denote the event in which $\tau \neq \tau'$, $x \neq x'$, $c = (g_1^{\tau'} \cdot g_2)^x \cdot g_3^r$ and $c = (g_1^{\tau'} \cdot g_2)^{x'} \cdot g_3^{r'}$. Since $\mathcal{B}$ perfectly simulates the experiment $\mathsf{ABOBind}_{\mathcal{LE}_{\mathsf{DL}}, q, \mathcal{A}}(\lambda)$ to $\mathcal{A}$, it holds that

$$\Pr[\mathsf{SuccessA}] = \Pr[\mathsf{ABOBind}_{\mathcal{LE}_{\mathsf{DL}}, q, \mathcal{A}}(\lambda) = 1] = \mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE}_{\mathsf{DL}}, q, \mathcal{A}}(\lambda).$$

Moreover, conditioned on $\mathsf{SuccessA}$, it holds that

$$(g_1^{\tau'} \cdot g_2)^x \cdot g_3^r = c = (g_1^{\tau'} \cdot g_2)^{x'} \cdot g_3^{r'}.$$

Using the identity $g_2 = g_1^{p-\tau} \cdot g_3^y$, we get

$$g_1^{(\tau'-\tau)\cdot x} \cdot g_3^{x \cdot y + r} = g_1^{(\tau'-\tau)\cdot x'} \cdot g_3^{x \cdot y + r'}.$$

Rearranging, it holds that

$$g_1^{(\tau'-\tau)\cdot(x-x')} = g_3^{y \cdot x' - y \cdot x + r' - r}. \tag{B.2}$$

To conclude the proof, note that conditioned on $\mathsf{SuccessA}$, it holds that $(\tau' - \tau) \cdot (x - x') \neq 0$, since $\tau' \neq \tau$ and $x' \neq x$. Therefore, Eq. (B.2) implies that $y \cdot x' - y \cdot x + r' - r$ is non-zero and is therefore invertible in $\mathbb{Z}_p$. Hence, Eq. (B.2) can be rewritten as

$$g_3 = g_1^{(\tau'-\tau)\cdot(x-x')\cdot(y\cdot x' - y\cdot x + r' - r)^{-1}} = g_1^{z'}.$$

Recall that $g_1 = g$ and $g_3 = h$, and therefore $g^{z'} = h$. In other words, conditioned on $\mathsf{SuccessA}$, the algorithm $\mathcal{B}$ computes the discrete logarithm of h with respect to g with probability 1. ■

B.3 An Efficient Construction Based on the RSA Assumption

Let ModGen be a probabilistic polynomial-time modulus-generation algorithm that receives as input the security parameter $\lambda \in \mathbb{N}$ and outputs a pair (N, e), where N is the product of two λ-bit primes and $\gcd(e, \phi(N)) = 1$. The following construction of a locally-equivocable commitment scheme with all-but-one binding is based on the hardness of the RSA problem relative to ModGen. The scheme's domain space and tag space are both $\mathbb{Z}_e$ where $(N, e) \leftarrow \mathsf{ModGen}(1^\lambda)$.

A locally-equivocable commitment scheme $\mathcal{LE}_{\mathsf{RSA}}$

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{Setup}(1^\lambda, q)$:

1. Sample $(N, e) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $g, h \leftarrow \mathbb{Z}_N^*$.
2. Output $\mathsf{crs} = (N, e, g, h)$.

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{Commit}(1^\lambda, \mathsf{crs}, x, i, \tau)$:

1. Parse crs as (N, e, g, h).
2. Sample $u \leftarrow \mathbb{Z}_N^*$ and compute $c = (g^\tau \cdot h)^x \cdot u^e$
3. Output (c, d) where $d = (x, u)$.

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{Decommit}(1^\lambda, \mathsf{crs}, c, d, i, \tau)$:

1. Parse crs as (N, e, g, h) and d as (x, u).
2. Compute $c' = (g^\tau \cdot h)^x \cdot u^e$.
3. If $c' = c$ then output x and otherwise output $\perp$.

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{AltSetup}(1^\lambda, q)$:

1. Sample $(N, e) \leftarrow \mathsf{ModGen}(1^\lambda)$ and $g \leftarrow \mathbb{Z}_N^*$.
2. Output $\mathsf{st}_0 = (N, e, g)$.

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{Equiv}_1(1^\lambda, \mathsf{st}_0, q, \tau)$:

1. Parse st_0 as $(N, e.g)$.
2. Sample $v_1, \ldots, v_q, w \leftarrow \mathbb{Z}_N^*$.
3. Compute $h = w^e / g^\tau$ and let $\widehat{\mathsf{crs}} = (N, e, g, h)$.
4. For each $i \in [q]$ compute $\widehat{c}_i = v_i^e$, and let $\mathsf{st}_1 = (v_1, \ldots, v_q, w)$.
5. Output $(\widehat{\mathsf{crs}}, \widehat{c}_1, \ldots, \widehat{c}_q, \mathsf{st}_1)$.

$\mathsf{LE}_{\mathsf{RSA}}.\mathsf{Equiv}_2(1^\lambda, x, i, \mathsf{st}_1, \tau)$:

1. Parse st_1 as $(v_1, \ldots, v_q, w)$.
2. Compute $\widehat{u} = v_i / w^x$.
3. Output $\widehat{d} = (x, \widehat{u})$.

The following theorem establishes the security of the above construction:

Theorem B.3. *Assuming the hardness of the RSA problem relative to* ModGen*, the scheme* $\mathcal{LE}_{\mathsf{RSA}}$ *is a locally-equivocable commitment scheme with all-but-one binding.*

Proof. In order to show that the scheme $\mathcal{LE}_{\mathsf{RSA}}$ is locally equivocable and all-but-one binding (recall Definitions 4.1 and 4.2), we first observe that it satisfies the equivocation correctness requirement since

$$(g^\tau \cdot h)^x \cdot \widehat{u}^e = w^{e \cdot x} \cdot \left(\frac{v_i}{w^x}\right)^e = v_i^e = \widehat{c}_i.$$

In addition, one can easily verify that for any algorithm $\mathcal{A}$ and for any integer $q = q(\lambda)$, the views of $\mathcal{A}$ in the experiments $\mathsf{IndParam}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}, 0}(\lambda)$ and $\mathsf{IndParam}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}, 1}(\lambda)$ are identically distributed, and hence the scheme $\mathcal{LE}_{\mathsf{RSA}}$ satisfies the equivocation indistinguishability requirement. We therefore focus on proving that $\mathcal{LE}_{\mathsf{RSA}}$ is all-but-one binding based on the hardness of the RSA problem.

Let $q = q(\lambda)$ be a polynomial, and let $\mathcal{A}$ be a probabilistic polynomial-time algorithm that participates in the experiment $\mathsf{ABOBind}_{q, \mathcal{A}}^{\mathcal{LE}_{\mathsf{RSA}}}(\lambda)$. We show that there exists a probabilistic polynomial-time algorithm $\mathcal{B}$ such that

$$\mathbf{Adv}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}}^{\mathsf{ABOBind}}(\lambda) \leq \Pr\left[\mathcal{B}\left(1^\lambda, N, e, g\right) = g^{1/e}\right]$$

for every $\lambda \in \mathbb{N}$, where $(N, e) \leftarrow \mathsf{ModGen}(1^\lambda)$ and $g \leftarrow \mathbb{Z}_N^*$. Consider the following probabilistic polynomial-time algorithm $\mathcal{B}$, which on input $(1^\lambda, N, e, g)$ is defined as follows:

1. Invoke $(\tau, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(1^{\lambda})$.
2. Sample $v_1, \ldots, v_q, w \leftarrow \mathbb{Z}_N^*$.
3. Compute $h = w^e / g^{\tau}$.
4. Set $\mathsf{st}_0 = (N, e, g)$, $\widehat{\mathsf{crs}} = (N, e, g, h)$, and for each $i \in [q]$ set $\widehat{c_i} = v_i^e$.
5. Invoke $(c, d, d', i, \tau') \leftarrow \mathcal{A}(\mathsf{st}_{\mathcal{A}}, \mathsf{st}_0, (v_1, \ldots, v_q, w))$. Parse d as (x, u) and d' as (x', u').
6. If any of the following conditions holds, output $\perp$ and terminate:
   - $\tau = \tau'$.
   - $x = x'$.
   - $c \neq (g^{\tau'} \cdot h)^x \cdot u^e$ or $c \neq (g^{\tau'} \cdot h)^{x'} \cdot (u')^e$.
7. Compute integers a and b such that $a \cdot e + b \cdot (\tau' - \tau) \cdot (x - x') = 1$. Such integers are guaranteed to exist and can be found efficiently using the extended Euclidean algorithm since, as we will later show, if this step is reached then it must be the case that e and $(\tau' - \tau) \cdot (x - x')$ are relatively prime.
8. Output $z = g^a \cdot \left(\frac{w^{x'} \cdot u'}{w^x \cdot u}\right)^b$.

We turn to bound the advantage of the algorithm $\mathcal{B}$ in computing the e-th root of g modulo N. Let $\mathsf{SuccessA}$ denote the event in which $\tau \neq \tau'$, $x \neq x'$, $c \neq (g^{\tau'} \cdot h)^x \cdot u^e$ and $c \neq (g^{\tau'} \cdot h)^{x'} \cdot (u')^e$. Since $\mathcal{B}$ perfectly simulates the experiment $\mathsf{ABOBind}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}}(\lambda)$ to $\mathcal{A}$, it holds that

$$\Pr[\mathsf{SuccessA}] = \Pr[\mathsf{ABOBind}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}}(\lambda) = 1] = \mathbf{Adv}^{\mathsf{ABOBind}}_{\mathcal{LE}_{\mathsf{RSA}}, q, \mathcal{A}}(\lambda).$$

Observe that conditioned on $\mathsf{SuccessA}$, it is the case that

$$(g^{\tau'} \cdot h)^x \cdot u^e = c = (g^{\tau'} \cdot h)^{x'} \cdot (u')^e,$$

and since $h = w^e / g^{\tau}$, it holds that

$$g^{(\tau' - \tau) \cdot x} \cdot (w^x \cdot u)^e = g^{(\tau' - \tau) \cdot x'} \cdot \left(w^{x'} \cdot u'\right)^e$$

This in turn implies that

$$g^{(\tau' - \tau) \cdot (x - x')} = \left(\frac{w^{x'} \cdot u'}{w^x \cdot u}\right)^e. \tag{B.3}$$

Observe that conditioned on $\mathsf{SuccessA}$, it holds $\tau' \neq \tau$ and $x' \neq x$. Since $\tau, \tau', x, x' \in \mathbb{Z}_e$, this means that $(\tau' - \tau) \cdot (x - x')$ is coprime to e. This implies that there exist integers a and b such that $a \cdot e + b \cdot (\tau' - \tau) \cdot (x - x') = 1$ and these are found by $\mathcal{B}$ in Step 7. Hence, it holds that

$$\begin{aligned} z^e &= g^{a \cdot e} \cdot \left(\frac{w^{x'} \cdot u'}{w^x \cdot u}\right)^{b \cdot e} \\ &= g^{a \cdot e + b \cdot (\tau' - \tau) \cdot (x - x')} \\ &= g. \end{aligned}$$

In other words, conditioned on $\mathsf{SuccessA}$, the algorithm $\mathcal{B}$ computes the e-th root of g with probability 1. ∎