Delegation with Updatable Unambiguous Proofs and PPAD-Hardness

by

Lisa L. Yang

B.S. Mathematics
Massachusetts Institute of Technology, 2019

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN COMPUTER SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2021

Signature of Author: _____
Department of Electrical Engineering and Computer Science
August 27, 2021

Certified by: _____
Vinod Vaikuntanathan
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by: _____
Yael Kalai
Adjunct Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: _____
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Delegation with Updatable Unambiguous Proofs and PPAD-Hardness

by

Lisa L. Yang

Submitted to the Department of Electrical Engineering and Computer Science
on August 27, 2021 in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science

ABSTRACT

In this work, we construct an *updatable* and *unambiguous* delegation scheme based on the decisional assumption on bilinear groups introduced by Kalai, Paneth and Yang [STOC 2019]. Using this delegation scheme, we show PPAD-hardness (and hence the hardness of computing Nash equilibria) based on the quasi-polynomial hardness of this bilinear group assumption and any hard language that is decidable in quasi-polynomial time and polynomial space.

The delegation scheme is for super-polynomial time deterministic computations and is publicly verifiable and non-interactive in the common reference string (CRS) model. It is *updatable* meaning that given a proof for the statement that a Turing machine reaches some configuration C in T steps, it is *efficient* to update it into a proof for the statement that the machine reaches the next configuration C' in T+1 steps. It is *unambiguous* meaning that it is hard to produce two different proofs for the same statement.

Thesis Supervisor: Vinod Vaikuntanathan
Title: Professor of Electrical Engineering and Computer Science

Thesis Supervisor: Yael Kalai
Title: Adjunct Associate Professor of Electrical Engineering and Computer Science

Contents

To each of Ourselves

and to Yael who has been with me since Day 0

1 Introduction

The computational complexity of finding a Nash equilibrium in bimatrix games has been the subject of extensive research in recent years. In his seminal work, Papadimitriou [Pap94] defined the complexity class **PPAD** and showed that it contains the problem NASH. Daskalakis, Goldberg and Papadimitriou [DGP09], and Chen, Deng and Teng [CDT09] proved that NASH is **PPAD**-complete.

Currently polynomial (or even subexponential) time algorithms for **PPAD** are not known and NASH is conjectured to be intractable. A promising approach to proving the hardness of **PPAD**, proposed by Papadimitriou, is to base its hardness on assumptions from cryptography. Despite tremendous progress in this direction over the past five years, **PPAD**-hardness is only known under very strong and "non-standard" cryptographic assumptions. Building on [AKV04], Bitanski, Paneth and Rosen [BPR15] show that **PPAD** is hard on average assuming sub-exponentially secure indistinguishability obfuscation. Hubáček and Yogev [HY17] extended this result to **CLS**, a subclass of **PPAD**. The assumption was relaxed in [GPS16, KS17] from indistinguishability obfuscation to strong assumptions related to functional encryption. Very recently, Choudhuri et al. [CHK+19b, CHK+19a] and Ephraim et al. [EFKP19] showed average-case hardness of **PPAD** under an assumption closely related to the soundness of the Fiat-Shamir heuristic when applied to specific protocols. See Section 2.3 for more details on related work.

Basing **PPAD**-hardness on weaker, well-studied cryptographic assumptions remains an important goal.

This work. We prove hardness of **CLS** and **PPAD**, under the following assumptions:

1. A decisional assumption on groups with bilinear maps (Assumption 1.3). This is a quasi-polynomial version of an assumption recently introduced by [KPY19]. It is falsifiable (in quasi-polynomial time) and it holds in the generic group model.
2. The existence of a hard language $\mathcal{L}$ that can be decided in time $n^{(\log n)^\epsilon}$ for some $\epsilon < 1$ and polynomial space. For example, the assumption that SAT over $m = (\log n)^{1+\epsilon}$ variables is hard for 2^{m^c}-size circuits for any $1/(1+\epsilon) < c < 1$ suffices. This is (weaker than) non-uniform Exponential Time Hypothesis (ETH). If $\mathcal{L}$ is hard on average we show average-case hardness of **PPAD**.

Our result follows a similar approach to that of Choudhuri et al. [CHK+19b] exploiting a folklore connection between **PPAD** and the notion of incrementally verifiable computation [Val08]. Specifically, we consider delegation schemes that are both *updatable* and *unambiguous*. Loosely speaking, a delegation scheme for T-time computations is a computationally sound proof system that can be verified in time $<< T$. For the purpose of proving **PPAD**-hardness, in this work we focus on publicly verifiable non-interactive schemes in the CRS model for delegating super-polynomial time computations with polynomial-time verification.[1] A delegation scheme is said to be *updatable* if given a proof of correctness for the first t steps of a computation, we can extend it to a proof of correctness of the first $t+1$ steps without recomputing the proof from scratch (that is, in time independent of t). A delegation scheme is said to be *unambiguous* if it is computationally hard to construct two different accepting proofs for the same statement.

We show that the existence of such a delegation scheme for a hard language $\mathcal{L}$ as above, implies the hardness of a problem known as RELAXED-SINK-OF-VERIFIABLE-LINE (rSVL) that was defined and reduced to a problem in **CLS** in [CHK+19b].

Theorem 1.1 (Informal)**.** *Let $\mathcal{L}$ be a hard (resp. hard on average) language decidable by a deterministic Turing machine running in time $T(n) = n^{\omega(1)}$ and space $S(n) = \mathrm{poly}(n)$. If there exists an updatable and unambiguous delegation scheme for $\mathcal{L}$ then rSVL is hard (resp. hard on average).*

We refer the reader to Theorem 4.4 for the formal statement, and to Section 3 for the definition of delegation.

Our main contribution is the construction of such a delegation scheme. Specifically, we show that for any $0 < \epsilon < 1$ and $T = T(n) \leq n^{(\log n)^\epsilon}$ there exists an updatable and unambiguous delegation scheme for any T-time polynomial-space deterministic computation under Assumption 1.3 below.

[1] More generally, in the literature delegation may also refer to privately verifiable schemes and interactive schemes. The focus is often on delegating polynomial-time computations with near linear-time verification.

Theorem 1.2 (Informal). *Let $0 < \epsilon < 1$ and let $\mathcal{M}$ be a deterministic Turing machine that runs in time $T(n) \leq n^{(\log n)^{\epsilon}}$ and space $S(n) = \mathrm{poly}(n)$. Under Assumption 1.3 for $\Lambda(\kappa) = 2^{(\log \kappa)^{\frac{1+\epsilon}{1-\epsilon}}}$ there exists an updatable and unambiguous delegation scheme for $\mathcal{M}$ with setup and verification time $\mathrm{poly}(S(n))$ where the prover runs in time $T(n) \cdot \mathrm{poly}(S(n))$.*

We refer the reader to Corollary 5.3 for the formal statement, and to Section 5 for our results with more general parameters. We note that in Theorem 1.2 the efficiency of the delegation scheme grows with the space of the computation. This dependency can be removed using standard techniques [KRR14, KPY19]. However, we did not implement this in the current work since it would further complicate the proof and is not needed for showing **PPAD**-hardness.

Assumption 1.3 is a version of the bilinear group assumption from [KPY19] with a hardness parameter $\Lambda = \Lambda(\kappa)$. We mention that [KPY19] rely on this assumption for $\Lambda(\kappa) = \mathrm{poly}(\kappa)$ to construct a delegation scheme for polynomial-time computations. To construct a delegation scheme for super-polynomial time computations, we rely on this assumption for super-polynomial $\Lambda(\kappa)$.

Assumption 1.3. *Let G be a group of prime order $p = 2^{\Theta(\kappa)}$ equipped with a bilinear map. For every $\alpha(\kappa) = O(\log \Lambda(\kappa))$ given the following 3-by-α matrix of group elements:*

$$\left(g^{s^j t^i}\right)_{\substack{i \in [0,2] \\ j \in [0,\alpha]}} = \begin{pmatrix} g^{s^0} & g^{s^1} & \dots & g^{s^\alpha} \\ g^{s^0 t} & g^{s^1 t} & \dots & g^{s^\alpha t} \\ g^{s^0 t^2} & g^{s^1 t^2} & \dots & g^{s^\alpha t^2} \end{pmatrix}$$

for random $g \in G$ and $s \in \mathbb{Z}_p$, it is $\Lambda(\kappa)$-hard to distinguish between the case where $t = s^{2\alpha+1}$ and the case where t is a random independent element in $\mathbb{Z}_p$.

2 Technical Overview

In this section we give an overview of our delegation scheme with updatable and unambiguous proofs. We build on the non-interactive delegation scheme of [KPY19] (KPY). We start by recalling the high-level structure of their scheme.

2.1 The KPY Delegation Scheme

The KPY construction consists of two steps: first, they construct *quasi-arguments* for **NP** which, following [KRR14], are known to imply delegation for **P**. The KPY quasi-arguments have a long CRS which results in a delegation scheme for **P** with a long CRS (of length proportional to the running time of the computation). Then they use quasi-arguments again to "bootstrap" a delegation scheme with a long CRS and get a delegation scheme with a short CRS.

Quasi-arguments. A quasi-argument is a relaxation of an argument-of-knowledge: in a quasi-argument, the standard knowledge extraction requirement is replaced by a weaker requirement called *non-signaling (local) extraction*. To argue about locality, the definition specifically considers the **NP** complete language 3SAT. Roughly speaking, in an argument-of-knowledge for 3SAT, for any prover that convinces the verifier to accept a formula φ there exists an extractor that produces a satisfying assignment for φ. In a quasi-argument, however, the extractor is not required to produce a full assignment. Rather it is given a *small* set of variables $\mathbf{S}$ and it produces an assignment only for the variables in $\mathbf{S}$. This partial assignment is required to be *locally consistent*, satisfying every clause of φ over variables in $\mathbf{S}$. Furthermore, the partial assignments produced by the extractor should satisfy the *non-signaling* property. Loosely speaking, this property requires that for any subsets $\mathbf{S} \subset \mathbf{S}'$ the distribution of the assignments produced by the extractor for the variables in $\mathbf{S}'$, when restricted to the variables in $\mathbf{S}$, is independent of the variables in $\mathbf{S}' \setminus \mathbf{S}$. We refer the reader to Section 8.1 for the formal definition. The notion of a quasi-argument was introduced in [PR17] under the name "core protocol with a local assignment generator". Prior works including [KRR13, KRR14, BHK17] (implicitly) construct *privately verifiable two-message* quasi-arguments for **NP**.

The BMW heuristic. The KPY quasi-argument is inspired by the BMW heuristic [BMW98] for converting a multi-prover interactive proof (MIP) into a two-message privately verifiable delegation scheme. In this delegation scheme, the verifier generates the MIP queries, encrypts each query using a homomorphic encryption scheme (with a fresh key), and sends the encrypted queries to the prover. The prover then homomorphically computes the encrypted answers, and the verifier decrypts and checks the answers. While this heuristic is known to be insecure in general [DNRS03, DHRW16], the work of [KRR13] shows that it is sound for MIPs satisfying a strong soundness condition called non-signaling soundness.

From private to public verification. To obtain a publicly verifiable non-interactive delegation scheme, KPY follow the blueprint of Paneth and Rothblum (PR) [PR17] and place the encrypted queries in the CRS. Now, since the verifier does not encrypt the queries itself, it can no longer decrypt the answers. Instead, the queries are encrypted using a special homomorphic encryption equipped with a *weak zero-test* that allows the verifier to check the validity of the prover's answers without decrypting them. Modularizing the analysis of [KRR13, KRR14], PR show that the resulting protocol is a quasi-argument for **NP**.

The CRS length. Unlike the PR solution that was based on multilinear maps, KPY construct a zero-testable homomorphic encryption scheme based only on bilinear maps. In the KPY scheme, however, the ciphertext length grows exponentially with the length of the encrypted query. This results in a quasi-argument with a long CRS. To shorten the CRS, KPY use an idea known as "bootstrapping" that was previously used to obtain succinct arguments of knowledge for **NP** (SNARKs) with a short CRS [Val08, BCCT13]. In this setting, a SNARK with a long CRS is recursively composed with itself yielding a SNARK with a short CRS. In contrast, KPY compose a delegation scheme for **P** and a quasi-argument for **NP**, both with a long CRS to obtain a delegation scheme for **P** with a short CRS.

2.2 Our Delegation Scheme

We modify the KPY delegation scheme to make its proofs updatable and unambiguous. Obtaining updatability is fairly straightforward. Previous work [Val08, BCCT13] used recursive proof composition to merge proofs and applied this technique both for bootstrapping proofs (with the goal of shortening the CRS) and for creating updatable proofs. In the setting of delegation for **P**, the work of KPY shows how to use quasi-arguments to merge proofs for bootstrapping. Following KPY, our work shows how to use quasi-arguments to merge proofs for updatability.

The main technical challenge and the focus of the following overview is achieving unambiguity. We first construct quasi-arguments for **NP** with a long CRS that satisfy a notion of unambiguity. Then we argue that unambiguity is preserved in the bootstrapping step. We mention that in addition to satisfying the unambiguity property, our quasi-arguments are also more efficient than the quasi-arguments in KPY. As a result, we can delegate $n^{\log n^{\epsilon}}$-time computations with a $\mathrm{poly}(n)$-size CRS, as opposed to KPY that could only delegate $n^{O(\log \log n)}$-time computations.

Unambiguous quasi-arguments. The KPY delegation scheme is constructed from a quasi-argument and, therefore, to get an unambiguous delegation scheme we first construct unambiguous quasi-arguments. In contrast to delegation schemes that argue about deterministic computations, quasi-arguments argue about non-deterministic formulas. We therefore need to take care in defining the required notion of unambiguity. The strongest requirement would be that the prover cannot find two accepting proofs for the same formula, even if the formula has multiple satisfying assignments. This notion, however, is only known under very strong assumptions [SW14, CPW20]. A natural relaxation is to ask for unambiguous proofs only for formulas where the satisfying assignment is unique, or where finding multiple satisfying assignments is intractable. Even this relaxation, however, seems outside the reach of our techniques. The issue is that there exist formulas where the full satisfying assignment is unique, however, there exists an efficient non-signaling local extractor that can produce multiple locally consistent assignments for every small set of variables (without violating the non-signaling property). Therefore, we further relax the unambiguity requirement for quasi-arguments to only require that it is hard to find multiple accepting proofs for formulas where any efficient non-signaling local extractor can only produce a unique assignment to each small set of variables. We refer to such formulas as *locally unambiguous*.

We observe that instantiating the KPY delegation scheme with a quasi-argument satisfying this notion of unambiguity results in an unambiguous delegation scheme. To argue this we need to take a closer look at the bootstrapping step of KPY.

The KPY delegation scheme is obtained by recursive composition of quasi-arguments. In the base of the recursion we use a quasi-argument with a long CRS to construct a delegation scheme for **P** with a long CRS. The delegation scheme simply invokes the quasi-argument for the following 3SAT formula encoding the delegated computation: The formula contains variables describing the state of the computation in each timestep and clauses that enforce that each state is correctly computed from the previous one. Since the computation is deterministic, we can design the formula so it has a unique satisfying assignment. Moreover, we can show that the resulting formula is, in fact, locally unambiguous as follows. Using the fact that the formula has a unique satisfying assignment we can argue that if a non-signaling local extractor produces a unique assignment for the variables describing some state of the computation then, by local-consistency, it must also produce a unique assignment for the variables describing the subsequent state. The assignment for the variables describing the initial state must be unique, and therefore, we can use the non-signaling property of the extractor to argue that the assignments for every set of variables must also be unique. Since the 3SAT formula is locally unambiguous, the unambiguity of the delegation scheme follows from that of the quasi-argument.

In the induction step the delegation scheme for **P** is composed with a quasi-argument to reduce the length of the CRS. The idea is to split the computation into small blocks and use the base delegation scheme to argue about each of the blocks. Simply sending the delegation proof for each block will result in a proof that is too long. Instead the composed delegation scheme invokes the quasi-argument for a 3SAT formula that is satisfiable if and only if the delegation proofs for all blocks are accepting. Using an argument similar to the one used in the base case, we can leverage the unambiguity of the base delegation scheme to show that this 3SAT formula is also locally unambiguous.

Constructing unambiguous quasi-arguments. Next we describe our high-level strategy for making the KPY quasi-argument unambiguous. Recall that in KPY the quasi-argument CRS consists of encrypted MIP queries and the proof contains encrypted answers. Our construction has two steps: first we modify the quasi-argument so the answers encrypted in the proof are unambiguous. That is, for an honestly generated CRS, it is hard to find two accepting proofs for the same locally unambiguous formula that, when decrypted, result in different answers. Then we proceed to argue the unambiguity of the ciphertexts themselves. We show that in the KPY encryption scheme it is hard to find two different ciphertexts that decrypt to the same value without knowing the secret key. Moreover, this task is hard even given the ciphertexts in the CRS. Together, these two steps imply the unambiguity of the quasi-argument proof. We first discuss the unambiguity of the ciphertexts and then explain how to achieve unambiguous answers which is the main challenge.

Unambiguity of ciphertexts. We first show that an adversarial prover cannot find two different ciphertexts that decrypt to the same value, even given the CRS that contains encryptions of random MIP queries. In the KPY quasi-argument, the MIP queries are random in $\mathbb{F}^\ell$ where $\mathbb{F}$ is a large field and ℓ is logarithmic in the number of variables in the formula. In the KPY encryption scheme, the secret key is a random element $\mathsf{sk} \in \mathbb{F}$ and a ciphertext encrypting an element $q \in \mathbb{F}^\ell$ is given by a cryptographic encoding of a random low-degree polynomial $P : \mathbb{F} \to \mathbb{F}^\ell$ such that $P(\mathsf{sk}) = q$. Therefore, the encryption of the random query $q \in \mathbb{F}^\ell$ in the CRS is just an encoding of a random polynomial and hence, it does not reveal any information about sk. Finding two ciphertexts that encrypt the same value requires finding two encoded low-degree polynomials that agree on sk. Since sk is information theoretically hidden, this can only be done with negligible probability.

Unambiguity of answers. Our next step is to modify the KPY quasi-argument so that it has unambiguous answers. Together with the unambiguity of the ciphertexts this implies the unambiguity of the entire quasi-argument proof. In the KPY quasi-argument, the prover's answers are given by low-degree polynomials in the queries. The first polynomial evaluated is denoted by X and it encodes the prover's assignment. Specifically, $X : \mathbb{F}^\ell \to \mathbb{F}$ is the multilinear extension of the assignment. That is, X is multilinear, and for every variable Z of the formula there exists a Boolean input $y \in \{0,1\}^\ell$ such that the assignment to Z is $X(y)$. For each encrypted query in the CRS, the proof contains the evaluation of X on that query as well as evaluations of additional "proof polynomials" that help convince the verifier

that the X evaluations are locally consistent. In this overview we focus on making the evaluations of X unambiguous. We use similar techniques to make the evaluations of the proof polynomials unambiguous.

Unambiguity of X. Our first goal is to ensure unambiguity of the X evaluations. That is, for a locally unambiguous formula and an honestly generated CRS it should be hard to find two accepting proofs that encrypt different evaluations of X. In fact, we show that for any fixed query $q \in \mathbb{F}^\ell$, the evaluation $X(q)$ is unambiguous regardless of the other queries encrypted in the CRS. Before proving the unambiguity of $X(q)$ for any query $q \in \mathbb{F}^\ell$ we focus on proving unambiguity in the special case where $q \in \{0,1\}^\ell$ is a Boolean query. In this case we can prove the unambiguity of $X(q)$ based on the fact that the formula is locally unambiguous using the properties of the KPY local extractor. To see this, recall that for a Boolean q, the evaluation $X(q)$ gives the assignment to some variable Z of the formula. The KPY extractor, given a small set of variables that contains Z, samples a CRS that contains an encryption of q, evaluates the prover on the CRS and obtains an accepting proof. (If the proof is rejecting, the extractor is allowed to fail.) It then decrypts the value $X(q)$ and returns it as the assignment to Z. Since the formula is locally unambiguous, the value that the extractor assigns to Z is unambiguous. Since the CRS sampled by the extractor has the same distribution as an honestly generated CRS that contains an encryption of q, it follows that the evaluation $X(q)$ in the proof is also unambiguous for the special case of Boolean queries.

For the general case of non-Boolean queries we observe that the KPY quasi-argument does not guarantee unambiguity. An adversarial prover can produce a second accepting proof by computing a different polynomial $\widetilde{X} \neq X$ that agrees with X on all inputs in $\{0,1\}^\ell$ and following the honest prover's strategy using $\widetilde{X}$ instead of X. It may be tempting to try and fix this problem by considering a variant of the KPY quasi-argument where the queries in the CRS are Boolean rather than random in $\mathbb{F}^\ell$. For this quasi-argument the unambiguity of the X evaluations follows by the above argument. However, we can no longer argue the unambiguity of ciphertexts. In fact, if we let the CRS contain only Boolean queries we can break the unambiguity of ciphertexts by evaluating two different polynomials that agree on $\{0,1\}^\ell$ over the same query resulting in two different ciphertext that encrypt the same value. We therefore proceed to argue the unambiguity of X for any query in $\mathbb{F}^\ell$.

A proof of multilinearity. Observe that in the above attack on the unambiguity of the KPY quasi-argument, while X is multilinear, $\widetilde{X}$ must have individual degree > 1 since a multilinear polynomial is completely determined by its evaluations on $\{0,1\}^\ell$. Intuitively, to eliminate such attacks our approach is to have the prover convince the verifier that the polynomial X it uses is indeed multilinear. Formally defining the soundness of such a "multilinearity proof", however, requires some care. The main challenge is that the polynomial X used by the cheating prover is not well defined. In fact, any small set of queries and answers is typically consistent with some multilinear polynomial. One way to strengthen the soundness condition is to require that the prover knows the the polynomial X. This, in particular, guarantees that a cheating prover does not choose X in a way that signals information about the queries encrypted in the CRS. Currently, however, such succinct non-interactive arguments of knowledge are only known under non-falsifiable knowledge assumptions [Nao03, GW11].

In order to avoid knowledge assumptions, we introduce a new notion of a multilinearity proof that allows us to argue the unambiguity of X for all queries. We then construct such proofs based on our bilinear assumption. Intuitively, the main idea is to identify some property of multilinear polynomials that can be tested locally, on a small set of queries and answers, and is sufficient to prove the unambiguity of X. We use the fact that the restriction of a multilinear polynomial to a one-dimensional axis-parallel line is a univariate linear function. Therefore, for any set of queries on an axis-parallel line, their answers must be consistent with some linear function. More formally, we consider the setting where the CRS contains m encrypted queries $q_1, ..., q_m \in \mathbb{F}^\ell$. The honest prover is given as input a multilinear polynomial $X : \mathbb{F}^\ell \to \mathbb{F}$ and it homomorphically evaluates $y_i = X(q_i)$ and sends these these m encrypted evaluations together with a multilinearity proof. The soundness requirement is that, except with negligible probability, if the proof is accepted then for every axis-parallel line $L : \mathbb{F} \to \mathbb{F}^\ell$ there exist elements $a, b \in \mathbb{F}$ such that for every $x \in \mathbb{F}$, if $q_i = L(x)$ then $y_i = a \cdot x + b$.

Unambiguity of X via proofs of multilinearity. Before describing our multilinearity proof, we show that by adding a multilinearity proof for X to the KPY quasi-argument we get unambiguity of X. The idea is to reduce any

attack on unambiguity for a query $q \in \mathbb{F}^\ell$ to an attack on unambiguity for a Boolean query. For the KPY quasi-argument we have already argued the unambiguity of X for Boolean queries. Therefore, in our new quasi-argument, the unambiguity of X must hold for any query in $\mathbb{F}^\ell$. Our strategy is to move from the query $q \in \mathbb{F}^\ell$ to a Boolean query one coordinate at a time. That is, we reduce any attack on unambiguity for a query $q \in \{0,1\}^i \times \mathbb{F}^{\ell-i}$ to an attack on unambiguity for a query $q' \in \{0,1\}^{i+1} \times \mathbb{F}^{\ell-i-1}$. In every step the success probability of the attack halves. Therefore, since the dimension ℓ is logarithmic in the size of the formula, the overall loss is polynomial.

Fix a query $q \in \{0,1\}^i \times \mathbb{F}^{\ell-i}$ and assume there is an adversary that given a random CRS that contains an encryption of q outputs two accepting proofs that encrypt two different answers for q. For $b \in \{0,1\}$ let $q_b \in \{0,1\}^{i+1} \times \mathbb{F}^{\ell-i-1}$ be the query q with its $(i+1)$'st coordinate set to b. Now move to an experiment where the CRS contains encryptions of q_0, q_1 and q. Since each query is encrypted under a fresh key, the adversary must output accepting proofs that encrypt two different answers for q with almost the same probability as before. Since the queries q_0, q_1, q are on the same axis-parallel line, by the soundness of the multilinearity proof, if the proofs encrypt two different answers for q they must also encrypt different answers for either q_0 or q_1. It follows that for some $b \in \{0,1\}$, the adversary given a random CRS that contains an encryption of q_b outputs two accepting proofs that encrypt two different answers for q_b with probability at least half as before.

Towards constructing a proof of multilinearity. We construct our multilinearity proof in two steps. First we construct an "equality proof" that provides a weaker soundness than the multilinearity proof. Then we use this equality proof to construct our multilinearity proof. Intuitively, if a multilinearity proof convinces the verifier that all queries were answered by evaluating a single multilinear polynomial, then an equality proof convinces the verifier that all queries were answered by evaluating a single low-degree polynomial that may not be multilinear. More formally, we again consider the setting where the CRS contains m encrypted queries $q_1, ..., q_m \in \mathbb{F}^\ell$. The honest prover is given as input a low-degree polynomial $X : \mathbb{F}^\ell \to \mathbb{F}$ and it homomorphically evaluates $y_i = X(q_i)$ and sends these these m encrypted evaluations together with an equality proof. The soundness requirement is that, except with negligible probability, if the proof is accepted and $q_i = q_j$, then $y_i = y_j$. It is easy to verify that the soundness of the equality proof is indeed weaker than the soundness of the multilinearity proof.

Zero-testable encryption. Our equality proof relies on the weak zero-test of the homomorphic encryption used in KPY.[2] Before describing the construction, we describe the properties of this test. The weak zero-test is a public procedure (not using the secret key) that given a ciphertext, tests if it encrypts zero or not. A perfectly accurate zero-test can clearly be used to break semantic security. Therefore, we consider a weak zero-test that has false negatives: it never passes on encryptions of non-zero values, however, it may fail on some encryptions of zero. The test is only guaranteed to pass on "trivial" encryptions of zero which are ciphertexts that result from homomorphically evaluating a polynomial that is identically zero over $\mathbb{F}$ on some fresh ciphertext.

To exemplify the usefulness of the weak zero-test consider the following dummy protocol: the CRS contains an encryption of some query $q \in \mathbb{F}^\ell$. The honest prover holds three polynomials $A, B, C : \mathbb{F}^\ell \to \mathbb{F}$ such that $A \cdot B \equiv C$. It homomorphically evaluates the polynomials A, B, C on q and sends the verifier the encrypted evaluations a, b, c respectively. Now, the verifier can test that indeed $a \cdot b = c$ by homomorphically computing the value $a \cdot b - c$ and zero-testing the resulting ciphertext. In the honest execution, $A \cdot B - C$ is indeed the zero polynomial over $\mathbb{F}$ and, therefore, the verifier evaluates a trivial encryption of zero and the weak zero-test is guaranteed to pass. If a cheating prover, however, sends encryptions such that $a \cdot b \neq c$ then the verifier's ciphertext encrypts a non-zero value and the weak zero-test is guaranteed to fail.

A proof of equality from zero-testable encryption. We outline our equality proof construction. For simplicity, in this overview we assume that the number of queries in the CRS m is 2. If $m > 2$ we simply give a separate equality proof for every pair of queries.[3] We also assume for simplicity that the polynomial X given as input to the honest

[2] As a homomorphic encryption scheme, the KPY construction has several limitations: it can only encrypt short messages, and it is limited to arity-one one-hop homomorphic computations. For simplicity, in this overview we ignore these limitations.

[3] Our actual construction is slightly more involved due to the limitations of the KPY homomorphic encryption. Our construction is also more efficient with the proof length growing linearly with m and not quadratically.

prover is multilinear. The construction can be extended to support higher degrees in a straightforward way. Note that we do not make any assumptions on the evaluations sent by the cheating prover.

Recall that the CRS contains queries $q_1, q_2 \in \mathbb{F}^\ell$ encrypted under different keys. Our construction crucially relies on the fact that the zero-testable homomorphic encryption from KPY is multi-key homomorphic and therefore we can compute jointly over both queries together.[4] Using multi-key homomorphism, a natural approach to implementing the equality proof is to simply have the verifier homomorphically compute the value $y_1 - y_2$ and zero-test the resulting ciphertext. This approach, however, does not achieve completeness. Even if the prover is honestly evaluating the same polynomial X on both queries, since q_1 and q_2 are encrypted independently, the verifier's ciphertext would be a non-trivial encryption of zero and may fail the zero-test. In more detail, we can think of the tested ciphertext as the result of evaluating the polynomial $D(z_1, z_2) = X(z_1) - X(z_2)$ on a ciphertext encrypting (q_1, q_2). Unless X is constant, D is not identically zero and, therefore, the verifier may reject.

Instead we take a different approach. Suppose that the prover's polynomial X is sparse. In this case, the prover can simply send X's coefficients and the verifier can recompute y_1 and y_2 on its own by evaluating X on q_1 and q_2. For a general polynomial X our idea is inspired by the interactive sum-check proof [LFKN90]. In a nutshell, we restrict X to a sequence of axis-parallel lines transitioning from q_1 to q_2. Each restriction is sparse and its consistency can be checked by the verifier using the weak zero-test.

In more detail, for every $i \in [\ell]$ the prover computes the polynomials $A_i, B_i : \mathbb{F}^{\ell-1} \to \mathbb{F}$ such that $X(z) \equiv A_i(z_{-i}) \cdot z_i + B_i(z_{-i})$ where z_{-i} denotes the vector z with its i'th entry removed. For $i \in [0, \ell]$ we denote by $q^{(i)}$ the "hybrid" input whose first i coordinates are from q_1 and whose last $\ell - i$ coordinates are from q_2 (so $q^{(0)} = q_2$ and $q^{(\ell)} = q_1$). The prover homomorphically computes the evaluations $y_1 = X(q_1)$ and $y_2 = X(q_2)$ and the equality proofs that contain for every $i \in [\ell]$ the encrypted evaluations:

$$\begin{aligned} y_1^{(i)} &= X\left(q^{(i)}\right), \quad & a_1^{(i)} &= A_i\left(q_{-i}^{(i)}\right), \quad & b_1^{(i)} &= B_i\left(q_{-i}^{(i)}\right), \\ y_2^{(i)} &= X\left(q^{(i-1)}\right), \quad & a_2^{(i)} &= A_i\left(q_{-i}^{(i-1)}\right) \quad & b_2^{(i)} &= B_i\left(q_{-i}^{(i-1)}\right). \end{aligned}$$

The verifier checks that $y_2 = y_2^{(1)}$, $y_1 = y_1^{(\ell)}$, $y_1^{(i)} = y_2^{(i+1)}$ for every $i \in [\ell - 1]$ and $(a_1^{(i)}, b_1^{(i)}) = (a_2^{(i)}, b_2^{(i)})$ for every $i \in [\ell]$ by comparing the evaluated ciphertexts. The verifier also uses the weak zero-test to check that $y_j^{(i)} = a_j^{(i)} \cdot (q_j)_i + b_j^{(i)}$ for every $j \in [2]$ and $i \in [\ell]$. The completeness of the proof follows from the properties of the weak zero-test together with the fact that, by construction, $q_{-i}^{(i)}$ and $q_{-i}^{(i-1)}$ are encrypted by the same ciphertext. To show soundness, we argue that if $q_1 = q_2$, then the equalities tested by the verifier imply that $y_1 = y_2$.

From equality to multilinearity. We proceed to construct a multilinearity proof using the zero-testable encryption and the equality proof. Recall that in a multilinearity proof, the CRS contains encrypted queries $q_1, ..., q_m \in \mathbb{F}^\ell$, the prover holds a multilinear polynomial X and it sends the encrypted evaluations $y_i = X(q_i)$ together with a multilinearity proof computed as follows. For every $i \in [\ell]$, the prover computes the polynomials $A_i, B_i : \mathbb{F}^{\ell-1} \to \mathbb{F}$ such that $X(z) \equiv A_i(z_{-i}) \cdot z_i + B_i(z_{-i})$. For every $j \in [m]$, the prover homomorphically evaluates $a_{i,j} = A_i((q_j)_{-i})$ and $b_{i,j} = B_i((q_j)_{-i})$. The proof consists of these encrypted evaluations. To ensure that the prover is evaluating the same polynomial X on every query, the multilinearity proof also includes an equality proof for the m query-answer pairs $\{(q_j, y_j)\}$. The verifier checks the equality proofs, and for every i, j the verifier homomorphically computes the value $a_{i,j} \cdot (q_j)_i + b_{i,j} - y_j$ and checks that all the resulting ciphertexts pass the weak zero-test.

However the proof described above is still not sound. A cheating prover may evaluate a polynomial X of individual degree > 1 and still produce an accepting proof by computing the encrypted evaluations $a_{i,j}, b_{i,j}$ as a function of the entire query q_j and not just of $(q_j)_{-i}$. For example, it may set $a_{i,j} = 0$ and $b_{i,j} = X(q_j)$. Our solution is to add to the proof two more equality proofs for every $i \in [m]$: one for the m query-answer pairs $\{((q_j)_{-i}, a_{i,j})\}$ and one for the m query-answer pairs $\{((q_j)_{-i}, b_{i,j})\}$. Intuitively, these extra equality proofs guarantee that $a_{i,j}, b_{i,j}$ are independent of q_j. To see this, consider a CRS that contains two queries $q_j, q_{j'}$ that agree on all their entries except for the i'th entry. Therefore, by the soundness of the equality proofs, the evaluations $(a_{i,j}, b_{i,j})$ and $(a_{i,j'}, b_{i,j'})$ must also agree.

[4] In KPY, as well as in this work, multi-key homomorphism is also used to evaluate the proof polynomials over multiple queries that are encrypted under different keys.

To argue the soundness of the multilinearity proof, consider a set of CRS queries $\{q_j\}_{j\in S}$ that lie on a line L that is parallel to the i'th axis, that is, the restrictions $\{(q_j)_{-i}\}_{j\in S}$ are all equal to each other. By the soundness of the equality proofs, all the pairs $\{(a_{i,j}, b_{i,j})\}_{j\in S}$ must also be equal to each other. Therefore, since all of the verifier's zero-tests pass, we have that $y_j = a_{i,1} \cdot (q_j)_i + b_{i,1}$ for every $j \in S$.

Unambiguity of the multilinearity and equality proofs. To guarantee the unambiguity of the evaluations of X we add a multilinearity proof to the KPY quasi-argument. Now however, to show the unambiguity of the new quasi-argument we must also guarantee that the multilinearity proofs are themselves unambiguous. Recall that the KPY encryptions already provide unambiguity of cipertexts and therefore, it is sufficient to prove that all the values encrypted in the multilinearity and equality proofs are unambiguous.

We start by arguing the unambiguity of the evaluations in the multilinearity proof. Recall that the multilinearity proof contains encrypted evaluations $a_{i,j}, b_{i,j}$ and equality proofs between these evaluations. We have already argued that the evaluations of X are unambiguous and it remains to show that the evaluations $\{a_{i,j}, b_{i,j}\}$ are also unambiguous. Assume towards contradiction that there exists an adversary that outputs two accepting proofs containing the evaluations $\{a_{i,j}, b_{i,j}\}$ and $\{a'_{i,j}, b'_{i,j}\}$ such that $(a_{i,j}, b_{i,j}) \neq (a'_{i,j}, b'_{i,j})$ for some i, j. That is, if we define the lines $L_{i,j}(z) \equiv a_{i,j} \cdot z + b_{i,j}$ and $L'_{i,j}(z) \equiv a'_{i,j} \cdot z + b'_{i,j}$ then $L_{i,j} \not\equiv L'_{i,j}$. Now consider an experiment where the CRS contains a query $q_{j'}$ that agrees with q_j on all entries except for the i'th entry. Since each query is encrypted under a fresh key, the adversary must continue to output accepting proofs such that $L_{i,j} \not\equiv L'_{i,j}$. Since $(q_j)_{-i} = (q_{j'})_{-i}$, by the soundness of the equality proofs also $L_{i,j} \equiv L_{i,j'}$ and $L'_{i,j} \equiv L'_{i,j'}$. Since $y_j, y_{j'}$ are unambiguous, the verifier's weak zero-test guarantees that:

$$L_{i,j}((q_j)_i) = y_j = L'_{i,j}((q_j)_i) \quad , \quad L_{i,j}((q_{j'})_i) = L_{i,j'}((q_{j'})_i) = y_{j'} = L'_{i,j'}((q_{j'})_i) = L'_{i,j}((q_{j'})_i) \ .$$

However, since $L_{i,j} \not\equiv L'_{i,j}$, the two lines cannot agree on both q_j and $q_{j'}$.

Moving on to the unambiguity of the equality proofs, recall that the equality proof for query-answer pairs $\{(q_j, y_j)\}_{j\in[2]}$ contains the evaluations $a_j^{(i)}, b_j^{(i)}$ and $y_j^{(i)}$ for every $j \in [2]$ and $i \in [\ell]$. First observe that it is sufficient to prove that if $y_1^{(i)}$ is unambiguous then so are $a_1^{(i)}, b_1^{(i)}$. To see this recall that the verifier checks that $(a_1^{(i)}, b_1^{(i)}) = (a_2^{(i)}, b_2^{(i)})$. The verifier's weak zero-test guarantees that $y_2^{(i)} = a_2^{(i)} \cdot (q_2)_i + b_2^{(i)}$. Therefore, if $a_2^{(i)}, b_2^{(i)}$ are unambiguous, then so is $y_2^{(i)}$. Finally, since the verifier checks that $y_1^{(i)} = y_2^{(i+1)}$, the unambiguity of $y_1 = y_1^{(\ell)}$ guarantees the unambiguity of the entire proof.

In an honestly generated equality proof, the evaluations $a_1^{(i)}, b_1^{(i)}$ are computed from $q_{-i}^{(i)}$ (recall that $q_{-i}^{(i)} = q_{-i}^{(i-1)}$). A cheating prover, however, may be able to output two accepting proofs with the same $y_1^{(i)}$ but with different $a_1^{(i)}, b_1^{(i)}$ by computing these evaluations also using $(q_1)_i, (q_2)_i$.[5] We have encountered a similar issue in arguing the soundness of the multilinearity proof. There, the solutions was to evaluate the same polynomial on queries encrypted under two different keys and provide an equality proof for the two evaluations. However, making the equality proof itself unambiguous requires a different solution.

Unambiguous equality proofs from rerandomized ciphertexts. To get unambiguous equality proofs we use a similar solution to the one used in the multilinearity proof, but instead of evaluating the same polynomial on queries encrypted under two different keys, we evaluate it on two encryptions of the same query, generated in a correlated way. In more detail, we add to the KPY zero-testable encryption a rerandomization operation that given a ciphertext c encrypting $q \in \mathbb{F}^\ell$ under a secret key sk, produces a new ciphertext $\widehat{c}$ encrypting q under a correlated key $\widehat{\mathsf{sk}}$. To explain the nature of the correlation between c and $\widehat{c}$ consider evaluating the same polynomial X on both c and $\widehat{c}$ and obtaining the evaluated ciphertexts e and $\widehat{e}$ respectively. The correlation between the two ciphertexts allows us to efficiently verify that e and $\widehat{e}$ indeed encrypt the same value. However, given the ciphertexts c and $\widehat{c}_{-i}$, which is obtained from $\widehat{c}$ by removing the encryption of the i'th entry q_i, it should be hard to recover $\widehat{c}$. In fact, to make the equality proofs unambiguous we need to rely on a stronger requirement: given c and $\widehat{c}_{-i}$ it is hard to generate a pair of ciphertexts encrypting values $\widehat{a}, \widehat{b} \in \mathbb{F}$ under $\widehat{\mathsf{sk}}$ such that $\widehat{a} \cdot q_i + \widehat{b} = 0$ but $\widehat{a}, \widehat{b}$ are not both zero.

[5] This can be done by viewing the verifier's checks as an underdetermined system of linear equations in the proof's ciphertexts.

Next we show how to use rerandomization to ensure the unambiguity of the equality proofs. As argued above, it is sufficient to prove that if $y_1^{(i)}$ is unambiguous then so are $a_1^{(i)}, b_1^{(i)}$. To this end we modify the equality proof as follows. Let $c^{(i)}$ be the encryption of the hybrid query $q^{(i)}$ given in the CRS. We rerandomize $c^{(i)}$, remove the encryption of the i'th element $\tilde{q}_i^{(i)}$, and add the resulting ciphertext $\widehat{c}_{-i}^{(i)}$ to the CRS. As before, the prover homomorphically evaluates the polynomials A_i and B_i on $c_{-i}^{(i)}$ to obtain the encrypted evaluations $a_1^{(i)}, b_1^{(i)}$. Then the verifier uses the weak zero-test to check that $y_1^{(i)} = a_1^{(i)} \cdot (q_1)_i + b_1^{(i)}$. Now, the prover also evaluates the polynomials A_i and B_i on $\widehat{c}_{-i}^{(i)}$ to obtain the encrypted evaluations $\widehat{a}_1^{(i)}, \widehat{b}_1^{(i)}$. Since $c_{-i}^{(i)}$ and $\widehat{c}_{-i}^{(i)}$ are correlated, the verifier can check that $(a_1^{(i)}, b_1^{(i)}) = (\widehat{a}_1^{(i)}, \widehat{b}_1^{(i)})$ and, thus, $y_1^{(i)} = \widehat{a}_1^{(i)} \cdot (q_1)_i + \widehat{b}_1^{(i)}$. Assuming $y_1^{(i)}$ is unambiguous, if a cheating prover given the CRS containing $c^{(i)}$ and $\widehat{c}_{-i}^{(i)}$ could find two accepting equality proof where the values $(a_1^{(i)}, b_1^{(i)}) = (\widehat{a}_1^{(i)}, \widehat{b}_1^{(i)})$ are different, by subtracting the ciphertext from the two proofs, we would obtain a pair of ciphertexts encrypting values $\widehat{a}, \widehat{b} \in \mathbb{F}$ such that $\widehat{a} \cdot q_i + \widehat{b} = 0$ but $\widehat{a}, \widehat{b}$ are not both zero, in contradiction to our requirement on correlated ciphertexts.

2.3 Related Work

Comparison with Choudhuri et al. and followup work. The **PPAD**-hardness proof of Choudhuri et al. [CHK+19b] and followup work [CHK+19a, EFKP19, LV20] can all be seen as constructing an updatable and unambiguous delegation scheme for some particular contrived language. In [CHK+19b] the language is related to the computation of a round-collapsed sum-check proof and [CHK+19a, EFKP19] start from the protocol of Pietrzak [Pie19] instead of sum-check. In contrast, this work constructs an updatable and unambiguous delegation scheme for general (bounded space) deterministic computations.

The delegation schemes in [CHK+19b, CHK+19a, EFKP19, LV20] are based on an interactive protocol that is made non-interactive via the Fiat-Shamir transform. The unambiguity property is inherited from that of the original protocol. Updatability relies on the recursive structure of the interactive protocol and requires augmenting the language to depend on the protocol itself. In comparison, the delegation scheme in our work is based on the scheme from [KPY19] for general computation and relies on a quasi-polynomial version of their assumption on bilinear groups. Updatability follows from the bootstrapping technique developed in [KPY19] and the focus of this work is on achieving unambiguity.

Following the work of Canetti et al. [CCH+19] on instantiating the Fiat-Shamir huristic from simpler assumptions, Choudhuri et al. [CHK+19b] show that the security of their sum-check based scheme follows from a strong assumption on the "optimal security" of Learning with Errors against quasi-polynomial attacks. In a recent work (concurrent to ours) Lombardi and Vaikuntanathan [LV20] start from Pietrzak's protocol and replace the Fiat-Shamir assumption by sub-exponential hardness of Learning with Errors.

In addition to the assumption behind the delegation scheme, previous work as well as ours rely on the hardness of an underlying language. Choudhuri et al. [CHK+19b] assume hardness of #SAT with poly-logarithmic number of variables, while [CHK+19a, EFKP19, LV20] rely on super-polynomial or sub-exponential hardness of the repeated squaring problem that is behind Pietrzak's protocol and the time-lock puzzle of [RSW96]. Since our delegation scheme supports general languages we can rely on any hard language that can be decided in quasi-polynomial time and polynomial space.

Hardness of local search. Recently, Bitansky and Gerichter [BG20] showed the hardness of the class Polynomial Local Search (**PLS**), which is a different subclass of **TFNP** that contains **CLS**, based on the delegation scheme of KPY [KPY19]. They observe that the KPY delegation scheme can be made incremental and use this to show **PLS** hardness. For hardness in **PPAD** and **CLS**, however, we need the unambiguity property achieved in this work.

3 Delegation

In this section we define the notion of a non-interactive delegation scheme for deterministic Turing machines.

Fix any Turing machine $\mathcal{M}$. Let $T(n)$ be an upper bound on the running time of $\mathcal{M}$ on inputs of length n and let $S(n)$ be an upper bound on the size of $\mathcal{M}$'s configuration which includes the machine's state, input tape and all of the

work tapes. We always assume, without loss of generality, that $T(n) \geq S(n) \geq n$. Let $\mathcal{U}^{\mathcal{M}}$ denote the language such that $(\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}^{\mathcal{M}}$ if and only if $\mathcal{M}$ transitions from configuration cf to configuration cf' in exactly t steps. Let $\mathcal{U}^{\mathcal{M}}_n \subseteq \mathcal{U}^{\mathcal{M}}$ be the set of instances $(\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}^{\mathcal{M}}$ such that the input tapes in $\mathsf{cf}, \mathsf{cf}'$ are of length n. Let $n = n(\kappa)$ be a function of the security parameter.

A non-interactive delegation scheme for $\mathcal{U}^{\mathcal{M}}$ consists of algorithms $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ with the following syntax:

Setup: The probabilistic setup algorithm $\mathsf{Del.S}$ takes as input a security parameter $\kappa \in \mathbb{N}$ and outputs a pair of public keys: a prover key pk and a verifier key vk.

Prover: The deterministic prover algorithm $\mathsf{Del.P}$ takes as input a prover key pk and an instance $x \in \mathcal{U}^{\mathcal{M}}$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{Del.V}$ takes as input a verifier key vk, an instance $x \in \mathcal{U}^{\mathcal{M}}$ and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 3.1. *A Λ-sound non-interactive delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = T_{\mathsf{S}}(\kappa)$ and proof length $L_{\Pi} = L_{\Pi}(\kappa)$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$ and $x = (\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}^{\mathcal{M}}_n$:*

$$\Pr\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa) \\ \Pi \leftarrow \mathsf{Del.P}(\mathsf{pk}, x) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time T_{S}.*
- *The prover runs in time $O(t \cdot L_{\Pi})$ and outputs a proof of length L_{Π}.*
- *The verifier runs in time $O(|x| + L_{\Pi})$.*

Λ-Soundness. *For every $\mathrm{poly}(\Lambda(\kappa))$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l} \mathsf{Del.V}(\mathsf{vk}, x, \Pi) = 1 \\ x \notin \mathcal{U}^{\mathcal{M}}_n \end{array} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa) \\ (x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

Next we define the notion of an updatable delegation scheme (incrementally verifiable computation [Val08]).

Definition 3.2 (Updatability)**.** *A non-interactive delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ for $\mathcal{U}^{\mathcal{M}}$ with input length n is updatable if there exists a deterministic polynomial-time algorithm $\mathsf{Del.U}$ such that for every $\kappa \in \mathbb{N}$ and $x_1, x_2 \in \mathcal{U}^{\mathcal{M}}_n$ of the form $x_1 = (\mathsf{cf}, \mathsf{cf}_1, t)$ and $x_2 = (\mathsf{cf}, \mathsf{cf}_2, t+1)$:*

$$\Pr\left[\begin{array}{l} \mathsf{cf}'_2 = \mathsf{cf}_2 \\ \Pi'_2 = \Pi_2 \end{array} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa) \\ \Pi_1 \leftarrow \mathsf{Del.P}(\mathsf{pk}, x_1) \\ \Pi_2 \leftarrow \mathsf{Del.P}(\mathsf{pk}, x_2) \\ (\mathsf{cf}'_2, \Pi'_2) \leftarrow \mathsf{Del.U}(\mathsf{pk}, x_1, \Pi_1) \end{array}\right] = 1 \ .$$

Lastly we define the notion of an unambiguous delegation scheme (adapting the definition in [RRR16]).

Definition 3.3 (Λ-Unambiguity)**.** *A non-interactive delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ for $\mathcal{U}^{\mathcal{M}}$ with input length n is Λ-unambiguous if for every $\mathrm{poly}(\Lambda(\kappa))$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l} \mathsf{Del.V}(\mathsf{vk}, x, \Pi) = 1 \\ \mathsf{Del.V}(\mathsf{vk}, x, \Pi') = 1 \\ \Pi \neq \Pi' \end{array} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa) \\ (x, \Pi, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

4 PPAD-Hardness

The complexity class **PPAD** is a subclass of **TFNP** that consists of all problems that are polynomial-time reducible to the End-of-the-Line problem. We show **PPAD**-hardness by following the blueprint of Choudhuri *et al.* [CHK+19b] and refer the reader to their work for background material. Specifically, we show the hardness of the subclass **CLS** that lies in the intersection of **PPAD** and **PLS**. Towards this end, we consider the Relaxed-Sink-of-Verifiable-Line problem that was defined and proven to be reducible to a problem in **CLS** in [CHK+19b].

Definition 4.1 ([CHK+19b])**.** *A* Relaxed-Sink-of-Verifiable-Line *(rSVL) instance* $(\mathsf{Succ}, \mathsf{Ver}, T, v_0)$ *consists of* $T \in [2^m]$*,* $v_0 \in \{0,1\}^m$ *and circuits* $\mathsf{Succ} : \{0,1\}^m \to \{0,1\}^m$ *and* $\mathsf{Ver} : \{0,1\}^m \times [T] \to \{0,1\}$ *with the guarantee that for every* $(v,i) \in \{0,1\}^m \times [T]$ *such that* $v = \mathsf{Succ}^i(v_0)$*, it holds that* $\mathsf{Ver}(v,i) = 1$*. A solution consists of one of the following:*

1. **The sink:** *A vertex* $v \in \{0,1\}^m$ *such that* $\mathsf{Ver}(v,T) = 1$.
2. **A false positive:** *A pair* $(v,i) \in \{0,1\}^m \times [2^m]$ *such that* $v \neq \mathsf{Succ}^i(v_0)$ *and* $\mathsf{Ver}(v,i) = 1$.

Lemma 4.2 ([CHK+19b])**.** Relaxed-Sink-of-Verifiable-Line *is polynomial-time reducible to a problem in* ***CLS****.*

Average-case hardness. Let $T = T(n)$ be a function. A decision problem given by a language $\mathcal{L}$ is weakly T-hard on average if there exists an efficiently sampleable distribution $\mathcal{D} = \{\mathcal{D}_n\}$ and polynomial p such that for every $\mathrm{poly}(T)$-size circuit $\mathsf{Adv} = \{\mathsf{Adv}_n\}$ and $n \in \mathbb{N}$:

$$\Pr_{x \leftarrow \mathcal{D}_n}\left[\mathsf{Adv}_n(x) = 1 \Leftrightarrow x \in \mathcal{L}\right] \leq 1 - 1/p(n) \ .$$

To prove the main theorem in this section (Theorem 4.4) it will be convenient to rely on a hard search problem rather than a hard decision problem. We start by stating a direct product theorem giving a hard search problem from a hard decision problem. A search problem given by a relation $\mathcal{R}$ is T-hard on average if there exists an efficiently sampleable distribution $\mathcal{D} = \{\mathcal{D}_n\}$ such that for every $\mathrm{poly}(T)$-size circuit $\mathsf{Adv} = \{\mathsf{Adv}_n\}$ there exists a negligible function μ such that for every $n \in \mathbb{N}$:

$$\Pr_{x \leftarrow \mathcal{D}_n}\left[(x, \mathsf{Adv}_n(x)) \in \mathcal{R}\right] \leq \mu(T(n)) \ .$$

Lemma 4.3. *If there exists a language* $\mathcal{L}$ *that is decidable in time* $T(n)$ *and space* $S(n)$ *that is weakly* $\widehat{T}$*-hard on average for some function* $\widehat{T}(n) \geq n$*, then there exists a search problem* $\mathcal{R}$ *that is solvable in time* $T(n) \cdot \mathrm{poly}(n)$ *and space* $S(n) + \mathrm{poly}(n)$ *and is* $\widetilde{T}$*-hard on average for any* $\widetilde{T}(n) = \widehat{T}(n)^{o(1)}$*.*

Proof. Let $\mathcal{L}$ be $\widehat{T}$-hard with respect to $\mathcal{D} = \{\mathcal{D}_n\}$ and polynomial p, let $t(n) = p(n) \cdot \log \widehat{T}$, and let

$$\mathcal{R} = \{(x_1 \ldots x_t, b_1 \ldots b_t) : \forall i \in [t]\ x_i \in \mathcal{L} \Leftrightarrow b_i = 1\} \ .$$

Indeed, $\mathcal{R}$ is solvable in time $T(n) \cdot \mathrm{poly}(n)$ and space $S(n) + \mathrm{poly}(n)$. The fact that $\mathcal{R}$ is $\widehat{T}^{o(1)}$-hard on average follows from the Concatenation Lemma of [GNW11]. The lemma states that if every $s(n)$-size circuit decides $\mathcal{L}$ with probability at most $\delta(n)$ over $\mathcal{D}$, then for any function $\epsilon(n)$, every $(\epsilon/n)^c \cdot s(n)$-size circuit solves $\mathcal{R}$ with probability at most $\delta^t + \epsilon$ over $\mathcal{D}^t = \{\mathcal{D}_n^t\}$ where $c > 0$ is some constant. Setting $s = \widehat{T}^{2c+1}$, $\delta = 1 - 1/p$, and $\epsilon = 1/\widehat{T}$ we have that by the $\widehat{T}$-hardness of $\mathcal{L}$ any circuit of size:

$$\left(\frac{\epsilon}{n}\right)^c s = \left(\frac{1}{\widehat{T} \cdot n}\right)^c \widehat{T}^{2c+1} \geq \widehat{T} \geq \mathrm{poly}(\widetilde{T}) \ ,$$

solves $\mathcal{R}$ with probability at most:

$$\delta^t + \epsilon = \left(1 - \frac{1}{p(n)}\right)^{p(n)\log\widehat{T}} + \frac{1}{\widehat{T}} \leq \frac{2}{\widehat{T}} = \mathrm{negl}(\widetilde{T}) \ .$$

□

Next we show that the existence of a hard search problem and an updatable unambiguous non-interactive delegation scheme implies rSVL is hard.

Theorem 4.4. *Let $\mathcal{R}$ be a search problem that is solvable by a deterministic Turing machine $\mathcal{M}$ that runs in time $T = T(n) = n^{\omega(1)}$ and space $S = S(n) = \mathrm{poly}(n)$ that is $\widehat{T}$-hard on average (in the worst-case respectively) for some function $\widehat{T}(n) \geq n$. Let $n = n(\kappa), \Lambda = \Lambda(\kappa)$ be functions such that $T(n) \leq \Lambda$.*

If there exists a non-interactive delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(n)$ and proof length $L_{\Pi} = \mathrm{poly}(n)$ that is updatable, Λ-sound, and Λ-unambiguous, then rSVL is $\widetilde{T}$-hard on average (in the worst-case respectively) for some $\widetilde{T} = \widetilde{T}(n) = \widehat{T}(n^{O(1)})$.

Proof. We start with the proof for average-case hardness. Then we explain how to modify the proof for worst-case hardness.

Let $\mathcal{R}$ be $\widehat{T}$-hard with respect to a distribution $\mathcal{D} = \{\mathcal{D}_n\}$. Let $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V}, \mathsf{Del.U})$ be a delegation scheme as in the theorem statement. Let A' denote a circuit for solving rSVL. We construct a circuit A that uses A' to solve $\mathcal{R}$.

Given as input an instance $x \in \{0,1\}^n$, the algorithm A proceeds as follows:

1. Set the security parameter κ such that $|x| = n$. Sample $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa)$. Let $m = S(n) + L_{\Pi}(\kappa)$.
2. Let cf_0 be the initial configuration of the Turing machine $\mathcal{M}$ on input x. We assume without loss of generality that at every time step, the configuration of $\mathcal{M}$ contains an index $i \in [T]$ corresponding to the current time step. Let $v_0 = (\mathsf{cf}_0, \Pi_0)$ where $\Pi_0 \leftarrow \mathsf{Del.P}(\mathsf{pk}, (\mathsf{cf}_0, \mathsf{cf}_0, 0))$.
3. Let $\mathsf{Succ} = \mathsf{Succ}_{x,\mathsf{pk}} : \{0,1\}^m \to \{0,1\}^m$ be the circuit that on input (cf_i, Π_i), parses the index $i \in [0,T]$ from cf_i and outputs $(\mathsf{cf}_{i+1}, \Pi_{i+1}) \leftarrow \mathsf{Del.U}(\mathsf{pk}, (\mathsf{cf}_0, \mathsf{cf}_i, i), \Pi_i)$.
4. Let $\mathsf{Ver} = \mathsf{Ver}_{x,\mathsf{vk}} : \{0,1\}^m \times [T] \to \{0,1\}$ be the circuit that on input $(v, i) \in \{0,1\}^m \times [T]$, parses $v = (\mathsf{cf}, \Pi)$ and returns the output of $\mathsf{Del.V}(\mathsf{vk}, (\mathsf{cf}_0, \mathsf{cf}, i), \Pi)$.
5. Run A' on $(\mathsf{Succ}, \mathsf{Ver}, T, v_0)$.
   (a) If A' outputs $v \in \{0,1\}^m$ such that $\mathsf{Ver}(v, T) = 1$ (the sink), then parse $v = (\mathsf{cf}, \Pi)$ and output the solution for x contained in cf.
   (b) Otherwise output $\perp$.

We construct the following $\widetilde{T}$-hard distribution $\mathcal{D}'$ of rSVL instances: sample $x \leftarrow \mathcal{D}_n$ and run Steps 1 to 4 of A to generate $(\mathsf{Succ}, \mathsf{Ver}, T, v_0)$ of length $\ell = \ell(n) \geq n$.

First we show $\mathcal{D}' = \{\mathcal{D}'_\ell\}$ is efficiently sampleable. By the efficiency guarantees of the delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V}, \mathsf{Del.U})$ (given by the theorem statement, Definition 3.1, Definition 3.2) Steps 1 to 4 take $\mathrm{poly}(n) = \mathrm{poly}(\ell)$ steps. Since $\mathcal{D}$ is efficiently sampleable, this shows $\mathcal{D}'$ is efficiently sampleable.

Next we argue that $\mathcal{D}'$ is supported on valid rSVL instances. We show that for any $x \in \{0,1\}^n$, A generates $(\mathsf{Succ}, \mathsf{Ver}, T, v_0)$ such that for every $i \in [T]$ it holds that $\mathsf{Ver}(\mathsf{Succ}^i(v_0), i) = 1$. Consider any $i \in [T]$ and let $v = (\mathsf{cf}, \Pi) = \mathsf{Succ}^i(v_0)$. Let cf_i be the unique configuration such that $(\mathsf{cf}_0, \mathsf{cf}_i, i) \in \mathcal{U}_n^{\mathcal{M}}$ and let $\Pi_i = \mathsf{Del.P}(\mathsf{pk}, (\mathsf{cf}_0, \mathsf{cf}_i, i))$. By the updatability of the delegation scheme (Definition 3.2) $(\mathsf{cf}, \Pi) = (\mathsf{cf}_i, \Pi_i)$ so by the completeness of the delegation scheme (Definition 3.1) $\mathsf{Ver}(v, i) = 1$, as desired.

To show that rSVL is $\widetilde{T}$-hard with respect to $\mathcal{D}'$, assume towards contradiction that there exists a $\mathrm{poly}(\widetilde{T}(\ell))$-size circuit $\mathsf{A}' = \{\mathsf{A}'_\ell\}$ and polynomial function p' such that for infinitely many $\ell \in \mathbb{N}$, given an rSVL instance sampled from $\mathcal{D}'_\ell$, A'_ℓ outputs a solution (the sink or a false positive) with probability at least $1/p'(\widetilde{T}(\ell))$. Since Steps 1 to 4 take $\mathrm{poly}(n)$ steps, $\ell = \mathrm{poly}(n) = n^c$ for some $c > 0$ so $\widetilde{T}(\ell) = \widehat{T}(n)$. Let p be a polynomial such that $p'(\widetilde{T}(\ell)) \leq p(\widehat{T}(n))$. Since $\mathcal{D}'$ is efficiently sampleable and A' is a circuit of size $\mathrm{poly}(\widehat{T}(n))$, A is a circuit of size $\mathrm{poly}(\widehat{T}(n))$. It follows from our assumption that for $x \leftarrow \mathcal{D}$, A' outputs a rSVL solution (the sink or a false positive) in Step 5 with probability at least $1/p(\widehat{T}(n))$. Below we show A' outputs a false positive with probability at most $1/2p(\widehat{T}(n))$ and therefore it outputs the sink with probability at least $1/2p(\widehat{T}(n))$. In this case, we use the sink to recover a solution for x.

Assume towards contradiction that for infinitely many $n \in \mathbb{N}$, A' outputs a false positive (v, i) with probability at least $1/2p(\widehat{T}(n)) \geq 1/2p(\Lambda(\kappa))$ (since $\widehat{T}(n) < T(n) \leq \Lambda(\kappa)$). If $(v = (\mathsf{cf}, \Pi), i)$ is a false positive, then $\mathsf{Del.V}(\mathsf{vk}, (\mathsf{cf}_0, \mathsf{cf}, i), \Pi) = \mathsf{Ver}(v, i) = 1$ and $(\mathsf{cf}, \Pi) \neq (\mathsf{cf}_i, \Pi_i) = \mathsf{Succ}^i(v_0)$, so either $\mathsf{cf} \neq \mathsf{cf}_i$, or $\mathsf{cf} = \mathsf{cf}_i$ and $\Pi \neq \Pi_i$. One of the two cases must occur for infinitely many $\kappa \in \mathbb{N}$ with probability at least $1/4p(\Lambda(\kappa))$. In the first case, $\mathsf{cf} \neq \mathsf{cf}_i$, and A' can be used to break the Λ-soundness of the delegation (Definition 3.1): $(\mathsf{cf}_0, \mathsf{cf}, i) \notin \mathcal{U}_n^{\mathcal{M}}$ but $\mathsf{Del.V}(\mathsf{vk}, (\mathsf{cf}_0, \mathsf{cf}, i), \Pi)$ accepts. In the second case, $\mathsf{cf} = \mathsf{cf}_i$ and $\Pi \neq \Pi_i$, and A' can be used to break the Λ-unambiguity of the delegation (Definition 3.3): by the efficiency of the delegation (cf_i, Π_i) can be computed in time $T(n) \cdot \mathrm{poly}(n) \leq \mathrm{poly}(\Lambda(\kappa))$, and $\mathsf{Del.V}(\mathsf{vk}, (\mathsf{cf}_0, \mathsf{cf}_i, i), \Pi)$ and $\mathsf{Del.V}(\mathsf{vk}, (\mathsf{cf}_0, \mathsf{cf}_i, i), \Pi_i)$ both accept.

This shows A' outputs a false positive with probability at most $1/2p(\widehat{T}(n))$. Thus for infinitely many $n \in \mathbb{N}$, with probability at least $1/2p(\widehat{T}(n))$, A' outputs the sink $v = (\mathsf{cf}, \Pi)$ and $(\mathsf{cf}, \Pi) = \mathsf{Succ}^T(v_0)$. By the updatability of the delegation (Definition 3.2) $(\mathsf{cf}_0, \mathsf{cf}, T) \in \mathcal{U}_n^{\mathcal{M}}$, i.e. cf is the configuration of $\mathcal{M}$ on input x after T steps so it contains a solution for x. In this case, A outputs this solution, contradicting the $\widehat{T}$-hardness of $\mathcal{R}$.

Now we explain how to modify this proof to show worst-case hardness of rSVL. If A' solves rSVL in the worst-case then A is a randomized circuit that solves $\mathcal{R}$ with probability $1 - \mathrm{negl}(\widehat{T}(n))$ for every $x \in \{0,1\}^n$. By standard techniques, such a randomized circuit can be converted into a deterministic circuit that solves $\mathcal{R}$ in the worst-case, contradicting the $\widehat{T}$-hardness of $\mathcal{R}$. This conversion only requires running $\mathrm{poly}(\widehat{T}(n))$ independent executions of A. Since A is of size $\mathrm{poly}(\widehat{T}(n))$, the final deterministic circuit is of size $\mathrm{poly}(\widehat{T}(n))$. □

5 Our Results

In this section we state our results: we show the existence of an updatable unambiguous delegation scheme and use it to show **PPAD**-hardness. Our results are based on the following decisional assumption on groups with bilinear maps. For a function $\Lambda = \Lambda(\kappa)$, the assumption is a Λ-secure version of the assumption in [KPY19].

Assumption 5.1. *There exists an ensemble of groups $G = \{G_\kappa\}$ of prime order $p = p(\kappa) = 2^{\Theta(\kappa)}$ with a non-degenerate bilinear map such that for every $d(\kappa) = O(\log \Lambda(\kappa))$ and $\mathrm{poly}(\Lambda)$-size adversary Adv, there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ g \leftarrow G \\ s \leftarrow \mathbb{Z}_p \\ t_0 \leftarrow \mathbb{Z}_p \\ t_1 \leftarrow s^{2d+1} \\ b' \leftarrow \mathsf{Adv}\left(\left(g^{s^i \cdot t_b^j} \right)_{\substack{i \in [0,d] \\ j \in [0,2]}} \right) \end{array} \right] \leq \frac{1}{2} + \mu(\Lambda(\kappa)) \ .$$

We show the existence of a non-interactive delegation scheme that is updatable and unambiguous under this assumption. Specifically, in Section 6 we construct an encryption scheme based on this assumption and use it to construct a quasi-argument in Section 8. In Section 9 we use a quasi-argument to construct a delegation scheme. By combining Theorem 6.16, Theorem 8.6, and Theorem 9.1, we obtain our main theorem:

Theorem 5.2 (Updatable Unambiguous Delegation)**.** *For any deterministic Turing machine $\mathcal{M}$ that runs in time $T = T(n)$ and space $S = S(n) \geq n$ and functions $n = n(\kappa) \geq \kappa$ and $\Lambda = \Lambda(\kappa)$ such that $T(n) \leq \Lambda \leq 2^{o(\kappa)}$, under Assumption 5.1 there exists a non-interactive delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(S(n), \kappa^{\log_n T(n)})$ and proof length $L_\Pi = \mathrm{poly}(S(n), \kappa^{\log_n T(n)})$ that is updatable, Λ-sound, and Λ-unambiguous.*

Proof. Let $N = 2S(n)$, $\bar{M} = \mathrm{poly}(S(n), \kappa^{\log_n T(n)})$, $\bar{\ell} = \log \bar{M}$, and $\bar{\delta} = 2$. Since $n \geq \kappa$ and $T(n) \leq \Lambda$, $\kappa^{\log_n T(n)} \leq \Lambda$. Thus $\bar{\delta}^{\bar{\ell}} = \bar{M} = \mathrm{poly}(S(n), \kappa^{\log_n T(n)}) \leq \mathrm{poly}(\Lambda)$.

The encryption scheme in Section 6 is defined over $\mathbb{F} = \mathbb{Z}_p$ where $p = 2^{\Theta(\kappa)}$ so $\Lambda \leq 2^{o(\kappa)} \leq |\mathbb{F}|^{o(1)}$. By Theorem 6.16 there exists a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) under Assumption 5.1.

Since $N \leq \bar{M} \leq \mathrm{poly}(\Lambda)$ by Theorem 8.6 there exists a Λ-secure Λ-unambiguous quasi-argument (Definitions 8.3 and 8.4) with formula size bound $\bar{M}$ and input length N. Finally by Theorem 9.1 there exists an updatable Λ-sound Λ-unambiguous delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(\bar{M}) = \mathrm{poly}(S(n), \kappa^{\log_n T(n)})$ and proof length $L_{\Pi} = \mathrm{poly}(\bar{M}) = \mathrm{poly}(S(n), \kappa^{\log_n T(n)})$ as desired. □

Next we state corollaries of Theorem 5.2 for different settings of parameters.

Corollary 5.3 (Quasi-polynomial Secure Delegation)**.** *For any constant $c \geq 1$ and any deterministic Turing machine $\mathcal{M}$ that runs in time $T \leq n^{(\log n)^a}$ where $a = (c-1)/(c+1)$ and space $S = \mathrm{poly}(n)$, let $\Lambda = 2^{(\log \kappa)^c}$ and $n = 2^{\sqrt{\log \Lambda \cdot \log \kappa}}$. Under Assumption 5.1 there exists a non-interactive delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(n)$ and proof length $L_{\Pi} = \mathrm{poly}(n)$ that is updatable, Λ-sound, and Λ-unambiguous.*

Proof. Note that $n = 2^{\sqrt{\log \Lambda \cdot \log \kappa}} = 2^{\sqrt{(\log \kappa)^{c+1}}} \geq \kappa$. Next we show $T(n) \leq \Lambda$ by showing:

$$n^{(\log n)^a} = 2^{(\log n)^{a+1}} \leq 2^{(\log \kappa)^c} \text{ for } a = (c-1)/(c+1) \ .$$

It suffices to prove that:

$$(\log n)^{a+1} \leq (\log \kappa)^c \text{ for } a = (c-1)/(c+1) \ .$$

This follows from the calculation:

$$(\log n)^{a+1} = (\log \Lambda \cdot \log \kappa)^{\frac{a+1}{2}} = ((\log \kappa)^c \cdot \log \kappa)^{\frac{a+1}{2}} = (\log \kappa)^{\frac{(c+1)(a+1)}{2}} = (\log \kappa)^c \ .$$

Finally note that $\Lambda = 2^{(\log \kappa)^c} \leq 2^{o(\kappa)}$.

The above shows the conditions of Theorem 5.2 hold and since:

$$\kappa^{\log_n T(n)} \leq \kappa^{\log_n \Lambda} = 2^{\frac{\log \Lambda \cdot \log \kappa}{\log n}} = 2^{\sqrt{\log \Lambda \cdot \log \kappa}} = n$$

we have $\mathrm{poly}(S(n), \kappa^{\log_n T(n)}) = \mathrm{poly}(n)$. Thus Corollary 5.3 follows. □

Corollary 5.4 (Sub-exponential Secure Delegation)**.** *For any constant $0 < \epsilon < 1$ and any deterministic Turing machine $\mathcal{M}$ that runs in time $T \leq n^{\frac{\epsilon}{2} \cdot \frac{\log n}{\log \log n}}$ and space $S = \mathrm{poly}(n)$, let $\Lambda = 2^{\kappa^\epsilon}$ and $n = 2^{\sqrt{\log \Lambda \cdot \log \kappa}}$. Under Assumption 5.1 there exists a non-interactive delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(n)$ and proof length $L_{\Pi} = \mathrm{poly}(n)$ that is updatable, Λ-sound, and Λ-unambiguous.*

Proof. Note that $\kappa^\epsilon > \log \kappa$ so $n = 2^{\sqrt{\log \Lambda \cdot \log \kappa}} > 2^{\log \kappa} = \kappa$. Next we show $T(n) \leq \Lambda$ by showing:

$$n^{\frac{\epsilon}{2} \cdot \frac{\log n}{\log \log n}} = 2^{\frac{\epsilon \cdot (\log n)^2}{2 \log \log n}} \leq 2^{\kappa^\epsilon} \ .$$

It suffices to prove that:

$$\frac{\epsilon \cdot (\log n)^2}{2 \log \log n} \leq \kappa^\epsilon \ .$$

This follows from the calculation:

$$\log n = (\log \Lambda \cdot \log \kappa)^{1/2} = (\kappa^\epsilon \cdot \log \kappa)^{1/2} \geq \kappa^{\epsilon/2}$$

$$\frac{\epsilon \cdot (\log n)^2}{2 \log \log n} = \frac{\epsilon \cdot \kappa^\epsilon \log \kappa}{2 \log \log n} \leq \frac{\epsilon \cdot \kappa^\epsilon \log \kappa}{2 \cdot (\epsilon/2) \cdot \log \kappa} = \kappa^\epsilon \ .$$

Finally note that $\Lambda = 2^{\kappa^\epsilon} \leq 2^{o(\kappa)}$.

The above shows the conditions of Theorem 5.2 hold and since:

$$\kappa^{\log_n T(n)} \leq \kappa^{\log_n \Lambda} = 2^{\frac{\log \Lambda \cdot \log \kappa}{\log n}} = 2^{\sqrt{\log \Lambda \cdot \log \kappa}} = n$$

we have $\mathrm{poly}(S(n), \kappa^{\log_n T(n)}) = \mathrm{poly}(n)$. Thus Corollary 5.4 follows. □

By Lemma 4.3 and Theorem 4.4, Corollaries 5.3 and 5.4 imply the hardness of rSVL. We remark that a hard language $\mathcal{L}$ can be based on a non-uniform version of the Exponential Time Hypothesis (ETH) for SAT.

Corollary 5.5 (rSVL Hardness based on Quasi-polynomial Hardness)**.** *Assume Assumption 5.1 holds for $\Lambda = 2^{(\log \kappa)^c}$ for some $c \geq 1$. If there exists a language $\mathcal{L}$ that is decidable by a deterministic Turing machine $\mathcal{M}$ that runs in time $T \leq n^{(\log n)^a}$ where $a = (c-1)/(c+1)$ and space $S = \mathrm{poly}(n)$ that is weakly $\widehat{T}$-hard on average (in the worst-case respectively) for some function $\widehat{T}(n) \geq n$, then rSVL is $\widetilde{T}$-hard on average (in the worst-case respectively) for any $\widetilde{T}(n) = \widehat{T}(n^{\epsilon})^{o(1)}$ for some $\epsilon > 0$.*

Corollary 5.6 (rSVL Hardness based on Sub-exponential Hardness)**.** *Assume Assumption 5.1 holds for $\Lambda = 2^{\kappa^{\epsilon}}$ for some $0 < \epsilon < 1$. If there exists a language $\mathcal{L}$ that is decidable by a deterministic Turing machine $\mathcal{M}$ that runs in time $T \leq n^{\frac{\epsilon}{2} \cdot \frac{\log n}{\log \log n}}$ and space $S = \mathrm{poly}(n)$ that is weakly $\widehat{T}$-hard on average (in the worst-case respectively) for some function $\widehat{T}(n) \geq n$, then rSVL is $\widetilde{T}$-hard on average (in the worst-case respectively) for any $\widetilde{T}(n) = \widehat{T}(n^{\epsilon'})^{o(1)}$ for some $\epsilon' > 0$.*

6 Zero-Testable Homomorphic Encryption

Our delegation scheme is based on a variant of the notion of zero-testable homomorphic encryption [PR17, KPY19]. We define this variant of zero-testable homomorphic encryption in Section 6.2. In Section 6.3 we construct such an encryption scheme based on bilinear maps, and in Section 6.4 we analyze the construction.

6.1 Notation

Let $\mathcal{E}$ denote the empty string. For any vector $\mathbf{v} = (v_1, \ldots, v_n)$ and $i \in [n]$ let $\mathbf{v}_i$ denote the element v_i. Let $\mathbf{v}_{-i}$ denote the vector $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$. For any pair of vectors $\mathbf{v} = (v_1, \ldots, v_n)$ and $\mathbf{w} = (w_1, \ldots, w_m)$ let $(\mathbf{v} \mid \mathbf{w})$ denote the concatenated vector $(v_1, \ldots, v_n, w_1, \ldots, w_m)$. We denote by $[\mathbf{v}, \mathbf{w}]_{-i}$ the pair $(\mathbf{v}_{-i}, \mathbf{w})$ if $i \leq n$ and the pair $\left(\mathbf{v}, \mathbf{w}_{-(i-n)}\right)$ if $i > n$.

Polynomials. We represent a polynomial $P : \mathbb{F}^{\ell} \to \mathbb{F}$ over a field $\mathbb{F}$ by a list of its coefficients. For a polynomial $P : \mathbb{F}^{\ell} \to \mathbb{F}$ of individual degree $\delta \in \mathbb{N}$ and $i \in [\ell]$, let $P|_{i,0}, \ldots, P|_{i,\delta} : \mathbb{F}^{\ell-1} \to \mathbb{F}$ be the unique polynomials such that:

$$P(\mathbf{z}) \equiv \sum_{j \in [0,\delta]} P|_{i,j}(\mathbf{z}_{-i}) \cdot \mathbf{z}_i^j \ . \tag{1}$$

Namely, interpreting P as a univariate polynomial in the i'th variable $\mathbf{z}_i$, $P|_{i,j}(\mathbf{z}_{-i})$ is the coefficient of $\mathbf{z}_i^j$.

For $\alpha \in \mathbb{F}$ we denote by $\overline{\alpha} : \mathbb{F}^{\ell} \to \mathbb{F}$ the constant polynomial α when the number of variables ℓ is clear from the context.

6.2 Definition

The notion of zero-testable homomorphic encryption introduced in this section is quite complex. While some of the details are similar to standard formulations of homomorphic encryption in the literature, other details are tailored to the applications presented in the subsequent sections. The definition in this section serves as an abstraction of the properties of our group-based construction that are required for our applications.

We present the definition of our zero-testable homomorphic encryption scheme in several stages. In Section 6.2.1 we present a bare-bones version of the notion featuring the basic interface and properties. In Section 6.2.2 we define the unambiguity of ciphertexts property. In Section 6.2.3 we introduce the zero-test operation. In Section 6.2.4 we define operations for extending and restricting ciphertexts. In Section 6.2.5 we introduce ciphertext rerandomization. The interface and properties defined in Sections 6.2.1 and 6.2.3 are similar to the properties in [KPY19] (with the exception of Definitions 6.2 and 6.3). The rest of the properties are new to this work.

6.2.1 Bare-bones encryption.

In this section we define a bare-bones version of our encryption scheme that includes the basic interface, correctness properties and semantic security. We first describe the syntax and then highlight some important differences between the notion here and the standard formulations of homomorphic encryption in the literature. For security parameter $\kappa \in \mathbb{N}$, the encryption scheme is parameterized by a bound $\bar{\delta} = \bar{\delta}(\kappa)$ on the individual degree of homomorphic computations and by a bound $\bar{\ell} = \bar{\ell}(\kappa)$ on the length of plaintexts. The plaintext space is of the form $\mathbb{F}^{\leq \bar{\ell}}$ where $\mathbb{F} = \mathbb{F}_\kappa$ is a field.

The bare-bones encryption scheme is given by the algorithms:

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec})$$

with the following syntax.

Parameter generation: the probabilistic parameter generation algorithm ParamGen takes as input the security parameter $\kappa \in \mathbb{N}$. It outputs public parameters pp. The running time of ParamGen is $\mathrm{poly}(\kappa, \log \bar{\ell})$.

Key generation: the PPT key generation algorithm KeyGen takes as input the public parameters pp and outputs a secret key sk.

Encryption: the probabilistic encryption algorithm Enc takes an input a secret key sk, a message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$, and randomness $\mathbf{r} \in \{0,1\}^{\kappa \times \ell}$. It outputs a single-key ciphertext c. The running time of Enc is $\mathrm{poly}(\kappa, \bar{\delta}^{\bar{\ell}})$. We explicitly refer to the randomness $\mathbf{r} \in \{0,1\}^{\kappa \times \ell}$, where $r_i \in \{0,1\}^\kappa$ is the randomness used to encrypt the i'th element of the message. This explicit notation will be useful later on.

Multi-key encryption: the probabilistic multi-key encryption algorithm MEnc takes as input a pair (γ_1, γ_2). For every $i \in [2]$, γ_i is a tuple $(\mathsf{sk}_i, \mathbf{m}_i, \mathbf{r}_i)$ including a secret key sk_i, a message $\mathbf{m}_i \in \mathbb{F}^{\ell_i}$ such that $\ell_i \leq \bar{\ell}$, and randomness $\mathbf{r}_i \in \{0,1\}^{\kappa \times \ell_i}$. It outputs a multi-key ciphertext Γ. The running time of MEnc is $\mathrm{poly}(\kappa, \bar{\delta}^{\bar{\ell}})$.

Homomorphic evaluation: the deterministic polynomial-time homomorphic evaluation algorithm Eval takes as input the public parameters pp, a (single-key or multi-key) ciphertext Γ encrypting messages of lengths $\ell_1, \ell_2 \leq \bar{\ell}$ (in the case that Γ is a single-key ciphertext either ℓ_1 or ℓ_2 is zero) and a polynomial $P : \mathbb{F}^{\ell_1 + \ell_2} \to \mathbb{F}$ of individual degree $\delta \leq \bar{\delta}$. It outputs a (single-key or multi-key) evaluated ciphertext e. The length of e is $\mathrm{poly}(\kappa, \ell_1 + \ell_2, \bar{\delta})$.

Encoded elements. Before we introduce the syntax of the decryption algorithm, we introduce the notion of encoded elements. The encryption's public parameters pp define for every element $\alpha \in \mathbb{F}$ an encoded element $\langle \alpha \rangle_{\mathsf{pp}}$. Given α and pp computing $\langle \alpha \rangle_{\mathsf{pp}}$ is efficient. While the encoding is injective, the inversion operation may not be efficient. The encoding is additively homomorphic meaning that given encodings of two elements we can efficiently compute the encoding of their sum or difference. We also define homomorphic multiplication of encodings. This operation however may not be efficient.

Decryption: the deterministic polynomial-time decryption algorithm Dec takes as input either a secret key sk and a single-key evaluated ciphertext e, or a pair of secret keys $(\mathsf{sk}_1, \mathsf{sk}_2)$ and a multi-key evaluated ciphertext e. It outputs the encoding $\langle \alpha \rangle_{\mathsf{pp}}$ of an element $\alpha \in \mathbb{F}$.

Notation. When the public parameters are clear from context, we may omit them from the inputs to the algorithms. We denote single-key ciphertexts by c and use Γ to denote ciphertexts that can be either single-key or multi-key. For a ciphertext Γ, we denote by $\ell(\Gamma)$ the total length of the message it encrypts. That is, Γ is either a single-key ciphertext encrypting a message of length $\ell(\Gamma)$ or a multi-key ciphertext encrypting two messages of lengths ℓ_1, ℓ_2 such that $\ell_1 + \ell_2 = \ell(\Gamma)$.

We highlight some important differences between the notion here and the standard formulations of homomorphic encryption in the literature:

- We only support "arity-one" homomorphic evaluation. That is, the homomorphic evaluation algorithm can only operate on a single ciphertext at a time.

- To homomorphically compute over multiple field elements, these elements must be encrypted together in a single ciphertext. Specifically, we can encrypt a message $\mathbf{m} \in \mathbb{F}^\ell$ into a single ciphertext where the ciphertext size is exponential in ℓ.
- Evaluation is "one-hop" meaning that we cannot continue to compute homomorphically over evaluated ciphertexts. Unlike fresh ciphertexts that can encrypt long messages, an evaluated ciphertext only encrypts a single feild element.
- The homomorphically evaluated polynomial P is represented as a list of monomials with coefficients in $\mathbb{F}$. Note that the size of this representation may be exponentially larger than the size of an arithmetic circuit for P. We emphasize that the running time of the homomorphic evaluation algorithm is polynomial in the number of monomials in P.
- Decryption outputs an additivly homomorphic encoding of the encrypted element rather than the element itself.

Next we define the properties of the encryption scheme. We start by defining the correctness of evaluation for single-key and multi-key ciphertexts. This property requires that evaluated ciphertexts encrypt the correct value.

Definition 6.1 (Correctness of Evaluation)**.** *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and polynomial $P : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[v = \langle P(\mathbf{m}) \rangle_{\mathsf{pp}} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ e \leftarrow \mathsf{Eval}(c, P) \\ v \leftarrow \mathsf{Dec}(\mathsf{sk}, e) \end{array} \right] = 1 \ .$$

Similarly, for every messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$ and polynomial $P : \mathbb{F}^{\ell_1 + \ell_2} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$:

$$\Pr\left[v = \langle P(\mathbf{m}_1, \mathbf{m}_2) \rangle_{\mathsf{pp}} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}, \ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ e \leftarrow \mathsf{Eval}(\Gamma, P) \\ v \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e) \end{array} \right] = 1 \ .$$

In the experiments above, the polynomial P is evaluated over $\mathbb{F}$.

Next we define the stability of evaluation. This property requires that when evaluating a polynomial P that is independent of its i'th input, the evaluated ciphertext is independent of the i'th element of the encrypted message.

Definition 6.2 (Stability of Evaluation)**.** *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$, index $i \in [\ell]$ and polynomials $P : \mathbb{F}^\ell \to \mathbb{F}$ and $P' : \mathbb{F}^{\ell-1} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $P(\mathbf{z}) \equiv P'(\mathbf{z}_{-i})$:*

$$\Pr\left[e' = e \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell}, \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ e \leftarrow \mathsf{Eval}(c, P) \\ c' \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}_{-i}, \mathbf{r}_{-i}) \\ e' \leftarrow \mathsf{Eval}(c', P') \end{array} \right] = 1 \ .$$

Similarly, for every messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ *such that* $\ell_1, \ell_2 \leq \bar{\ell}$*, index* $i \in [\ell_1 + \ell_2]$ *and polynomials* $P : \mathbb{F}^{\ell_1+\ell_2} \to \mathbb{F}$ *and* $P' : \mathbb{F}^{\ell_1+\ell_2-1} \to \mathbb{F}$ *of individual degree at most* $\bar{\delta}$ *such that* $P(\mathbf{z}) \equiv P'(\mathbf{z}_{-i})$*:*

$$\Pr\left[e' = e \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}\,,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ e \leftarrow \mathsf{Eval}(\Gamma, P) \\ (\mathbf{m}'_1, \mathbf{m}'_2) \leftarrow [\mathbf{m}_1, \mathbf{m}_2]_{-i} \\ (\mathbf{r}'_1, \mathbf{r}'_2) \leftarrow [\mathbf{r}_1, \mathbf{r}_2]_{-i} \\ \Gamma' \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}'_1, \mathbf{r}'_1), (\mathsf{sk}_2, \mathbf{m}'_2, \mathbf{r}'_2)) \\ e' \leftarrow \mathsf{Eval}(\Gamma', P') \end{array}\right] = 1 \ .$$

Next we require that the encryption of the empty message is independent of the secret key.

Definition 6.3 (Empty Message Encryption)**.** *For every* $\kappa \in \mathbb{N}$*:*

$$\Pr\left[c_1 = c_2 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}_1, \mathcal{E}, \mathcal{E}) \\ c_2 \leftarrow \mathsf{Enc}(\mathsf{sk}_2, \mathcal{E}, \mathcal{E}) \end{array}\right] = 1 \ .$$

Next we define the correctness of encryption. We require that the ciphertext generated by encrypting a message $\mathbf{m}$ using Enc is identical to the ciphertext generated by encrypting messages $\mathbf{m}$ and $\mathcal{E}$ using MEnc.

Definition 6.4 (Correctness of Encryption)**.** *For every* $\kappa \in \mathbb{N}$ *and message* $\mathbf{m} \in \mathbb{F}^{\ell}$ *such that* $\ell \leq \bar{\ell}$*:*

$$\Pr\left[\begin{array}{l} \Gamma_1 = c \\ \Gamma_2 = c \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}, \mathsf{sk}' \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ \Gamma_1 \leftarrow \mathsf{MEnc}((\mathsf{sk}, \mathbf{m}, \mathbf{r}), (\mathsf{sk}', \mathcal{E}, \mathcal{E})) \\ \Gamma_2 \leftarrow \mathsf{MEnc}((\mathsf{sk}', \mathcal{E}, \mathcal{E}), (\mathsf{sk}, \mathbf{m}, \mathbf{r})) \end{array}\right] = 1 \ .$$

Next we define one-time semantic security.

Definition 6.5 (Λ-Semantic Security)**.** *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$ *and messages* $\mathbf{m}_0, \mathbf{m}_1 \in \mathbb{F}^{\ell}$ *such that* $\ell \leq \bar{\ell}$*:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}_0, \mathbf{r}) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}_1, \mathbf{r}) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array}\right] \leq \frac{1}{2} + \mu(\Lambda(\kappa)) \ .$$

In the above definition we implicitly require that the size of the ciphertext is bounded by the size of Adv. We therefore require that $\bar{\delta}^{\bar{\ell}} \leq \mathrm{poly}(\Lambda)$.

6.2.2 Unambiguity of ciphertexts.

The next property that we require is the unambiguity of ciphertexts. The unambiguity property for evaluated ciphertexts states that without knowing the secret key, an adversary cannot produce two different ciphertexts that encrypt the same value. We require this to hold even if the adversary is given an encryption of a random message in $\mathbb{F}^\ell$.

We note that the adversary can always start with an honestly generated ciphertext and pad it with two different strings. In this case we could treat the padded ciphertexts as encrypting an undefined value. However, we choose to define the function Dec that never fails and always returns some encoded element. Instead we require that the set $\mathsf{Valid} = \mathsf{Valid}_\kappa$ of valid evaluated ciphertext is recognizable in polynomial time.

Definition 6.6 (Unambiguity of Ciphertexts)**.** *For every adversary* Adv*,* $\kappa \in \mathbb{N}$ *and* $\ell \leq \bar{\ell}$*:*

$$\Pr\left[\begin{array}{l|l} e, e' \in \mathsf{Valid} & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ e \neq e' & \mathbf{m} \leftarrow \mathbb{F}^\ell, \\ v = v' & \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ & \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ & c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ & (e, e') \leftarrow \mathsf{Adv}(\mathsf{pp}, c) \\ & v \leftarrow \mathsf{Dec}(\mathsf{sk}, e) \\ & v' \leftarrow \mathsf{Dec}(\mathsf{sk}, e') \end{array}\right] \leq \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \ .$$

Similarly, for every adversary Adv*,* $\kappa \in \mathbb{N}$ *and* $\ell_1, \ell_2 \leq \bar{\ell}$*:*

$$\Pr\left[\begin{array}{l|l} e, e' \in \mathsf{Valid} & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ e \neq e' & \mathbf{m}_1 \leftarrow \mathbb{F}^{\ell_1},\ \mathbf{m}_2 \leftarrow \mathbb{F}^{\ell_2} \\ v = v' & \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ & \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1},\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ & \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ & (e, e') \leftarrow \mathsf{Adv}(\mathsf{pp}, \Gamma) \\ & v \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e) \\ & v' \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e') \end{array}\right] \leq \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \ .$$

6.2.3 Quadratic zero-test.

The encryption scheme supports a zero-test operation. We start with an informal discussion of this notion.

Intuitively, the zero-test indicates whether a given evaluated ciphertext e decrypts to zero or not. If defined naively, however, such zero-test can be used to break the encryption's semantic security. We therefore consider a zero-test satisfying a weak notion of correctness: the test never passes if e does not decrypt to zero, but it may fail even if e does decrypt to zero. The zero-test is only required to pass if there exists an honestly generated ciphertext c such that e is obtained by homomorphically evaluating the zero polynomial $P \equiv 0$ on c.

Two-hop homomorphism. The encryption scheme defined in Section 6.2.1 only supports one-hop homomorphic evaluation. Our application, however, requires a restricted type of two-hop homomorphic evaluation: given two vectors of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{e}' = (e'_1, \ldots, e'_n)$ and a coefficient vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$ it is possible to further evaluate the weighted inner product $\sum_{i \in [n]} \alpha_i \cdot e_i \cdot e'_i$. The ciphertext resulting from the second-hop evaluation can then be zero-tested.

For simplicity, we unify the second-hop evaluation and the zero-test into one operation that we call quadratic zero-test. Given vectors $\mathbf{e}, \mathbf{e}'$ and $\boldsymbol{\alpha}$, the quadratic zero-test is only guaranteed to pass if there exists an honestly generated ciphertext c and vectors of polynomials $(P_1, \ldots, P_n)$ and $(P'_1, \ldots, P'_n)$ such that $\sum_{i \in [n]} \alpha_i \cdot P_i \cdot P'_i \equiv 0$ and the ciphertexts e_i and e'_i are obtained by homomorphically evaluating P_i and P'_i, respectively, on c. If e_i and e'_i encrypt values m_i and m'_i, respectively, such that $\sum_{i \in [n]} \alpha_i \cdot m_i \cdot m'_i \neq 0$ the quadratic zero-test fails.

Formally, quadratic zero-test ZT has the following syntax:

Quadratic zero-test: the deterministic polynomial-time quadratic zero-test algorithm ZT takes as input the public parameters pp, two vectors of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{e}' = (e'_1, \ldots, e'_n)$, and a vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$. It outputs a bit indicating if the test passes or fails. The running time of ZT is linear in n. We may omit the input vector $\boldsymbol{\alpha}$ when all its entries are ones.

The zero-test satisfies the following completeness and soundness requirements.

Definition 6.7 (Weak Completeness of Zero-Test)**.** *For every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$, vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$ and polynomials $\left\{P_i, P'_i : \mathbb{F}^{\ell_1+\ell_2} \to \mathbb{F}\right\}_{i\in[n]}$ of individual degree at most $\bar{\delta}$ such that $\sum_{i\in[n]} \alpha_i \cdot P_i \cdot P'_i \equiv 0$:*

$$\Pr\left[\mathsf{ZT}(\mathsf{pp}, \mathbf{e}, \mathbf{e}', \boldsymbol{\alpha}) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa\times\ell_1}\,,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa\times\ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \mathbf{e} = (e_i \leftarrow \mathsf{Eval}(\Gamma, P_i)\ :\ i \in [n]) \\ \mathbf{e}' = (e'_i \leftarrow \mathsf{Eval}(\Gamma, P'_i)\ :\ i \in [n]) \end{array}\right] = 1\ .$$

Definition 6.8 (Soundness of Zero-Test)**.** *For every $\kappa \in \mathbb{N}$, vectors of single-key evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{e}' = (e'_1, \ldots, e'_n)$ and vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$:*

$$\Pr\left[\begin{array}{l} \mathsf{ZT}(\mathsf{pp}, \mathbf{e}, \mathbf{e}', \boldsymbol{\alpha}) = 1 \\ \sum_{i\in[n]} \alpha_i \cdot v_i \cdot v'_i \neq \langle 0\rangle_{\mathsf{pp}} \end{array}\;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \forall i \in [n] :\ v_i \leftarrow \mathsf{Dec}(\mathsf{sk}, e_i) \\ \forall i \in [n] :\ v'_i \leftarrow \mathsf{Dec}(\mathsf{sk}, e'_i) \end{array}\right] = 0\ .$$

Similarly, for every $\kappa \in \mathbb{N}$, vectors of multi-key evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$ and $\mathbf{e}' = (e'_1, \ldots, e'_n)$ and vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$:

$$\Pr\left[\begin{array}{l} \mathsf{ZT}(\mathsf{pp}, \mathbf{e}, \mathbf{e}', \boldsymbol{\alpha}) = 1 \\ \sum_{i\in[n]} \alpha_i \cdot v_i \cdot v'_i \neq \langle 0\rangle_{\mathsf{pp}} \end{array}\;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \forall i \in [n] :\ v_i \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e_i) \\ \forall i \in [n] :\ v'_i \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e'_i) \end{array}\right] = 0\ .$$

6.2.4 Ciphertext extension and restriction.

The encryption scheme supports operations for adding and removing elements from a given ciphertext (with knowing the secret key). The ciphertext extension operation Extend turns a single-key ciphertext into a multi-key one. The restriction operation $\mathsf{Restrict}$ removes the element in a given location from the encrypted message. The ciphertext random extension operation $\mathsf{RandExtend}$ adds a random unknown element to the encrypted message in a given location. In contrast to the ciphertext extension operation, the random extension operation does not add a new key to the ciphertext and the new element is encrypted under the original key. Formally, the extension, restriction and random extension operations have the following syntax:

Ciphertext extension: the probabilistic ciphertext extension algorithm Extend takes as input γ_1 and γ_2 such that for some $i \in [2]$, γ_i is a single-key ciphertext c and for $j \in [2] \setminus \{i\}$, γ_j is a a tuple $(\mathsf{sk}, \mathbf{m}, \mathbf{r})$ including a secret key sk, a message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$, and randomness $\mathbf{r} \in \{0,1\}^{\kappa\times\ell}$. It outputs a multi-key ciphertext Γ. The running time of Extend is $\mathrm{poly}(\kappa, \bar{\delta}^{\bar{\ell}})$.

Ciphertext restriction: the deterministic polynomial-time ciphertext restriction algorithm $\mathsf{Restrict}$ takes as input the public parameters pp, a ciphertext Γ and an index $i \in [\ell(\Gamma)]$. It outputs a ciphertext Γ_{-i} that omits the i'th element of the message.

Ciphertext random extension: the PPT ciphertext random extension algorithm $\mathsf{RandExtend}$ takes as input the public parameters pp, a multi-key ciphertext Γ, and an index $i \in [\ell(\Gamma)+2]$. It outputs a ciphertext Γ' that includes an encryption of a random element from $\mathbb{F}$ inserted at the specified index (if Γ encrypts messages of lengths ℓ_1, ℓ_2, then $i \in [\ell_1+1]$ index into the first message and $i \in [\ell_1+2, \ell(\Gamma)+2]$) index into the second message).

We may replace the index i with a set of indices I as a shorthand for applying $\mathsf{Restrict}$ or $\mathsf{RandExtend}$ successively for every index $i \in I$. For $\mathsf{Restrict}$ the applications on $i \in I$ are in decreasing order, and for $\mathsf{RandExtend}$ the applications are in increasing order.

Next we define the correctness of the extension algorithm. We require that a ciphertext generated by first encrypting a message using Enc and then extending this ciphertext using Extend, is identical to the multi-key ciphertext generated by MEnc.

Definition 6.9 (Correctness of Extension)**.** *For every $\kappa \in \mathbb{N}$ and messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$:*

$$\Pr\left[\begin{array}{c|l} \Gamma_1 = \Gamma \\ \Gamma_2 = \Gamma \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1},\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1) \\ c_2 \leftarrow \mathsf{Enc}(\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2) \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \Gamma_1 \leftarrow \mathsf{Extend}(c_1, (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \Gamma_2 \leftarrow \mathsf{Extend}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), c_2) \end{array}\right] = 1 \ .$$

Next we define the correctness of the restriction algorithm. We require that the ciphertext obtained by first encrypting a message $\mathbf{m}$ and then applying $\mathsf{Restrict}$ to an index i is identical to the ciphertext generated by encrypting $\mathbf{m}_{-i}$.

Definition 6.10 (Correctness of Restriction)**.** *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^{\ell}$ such that $\ell \leq \bar{\ell}$ and index $i \in [\ell]$:*

$$\Pr\left[c' = c_{-i} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ c_{-i} \leftarrow \mathsf{Restrict}(c, i) \\ \mathbf{m}' \leftarrow \mathbf{m}_{-i} \\ \mathbf{r}' \leftarrow \mathbf{r}_{-i} \\ c' \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}', \mathbf{r}') \end{array}\right] = 1 \ .$$

Similarly, for every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$ and index $i \in [\ell_1 + \ell_2]$:

$$\Pr\left[\Gamma' = \Gamma_{-i} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1},\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \Gamma_{-i} \leftarrow \mathsf{Restrict}(\Gamma, i) \\ (\mathbf{m}'_1, \mathbf{m}'_2) \leftarrow [\mathbf{m}_1, \mathbf{m}_2]_{-i} \\ (\mathbf{r}'_1, \mathbf{r}'_2) \leftarrow [\mathbf{r}_1, \mathbf{r}_2]_{-i} \\ \Gamma' \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}'_1, \mathbf{r}'_1), (\mathsf{sk}_2, \mathbf{m}'_2, \mathbf{r}'_2)) \end{array}\right] = 1 \ .$$

Next we define the correctness of the random extension algorithm. We require that the distribution of ciphertexts obtained by encrypting messages using MEnc and then extending the i'th coordinate using $\mathsf{RandExtend}$ is identical to the distribution of ciphertexts obtained by encrypting the same messages with an additional random i'th coordinate. In the following definition, for a vector $\mathbf{v} = (v_1, \ldots, v_\ell) \in A^\ell$ (where A is either $\mathbb{F}$ or $\{0,1\}^\kappa$) and index $i \in [\ell]$ let

$\mathbf{v}_{+i}$ denote the random variable $(v_1, \ldots, v_{i-1}, v^*, v_i, \ldots, v_\ell)$ where $v^* \leftarrow A$, or $(\mathbf{v} \mid v^*)$ if $i = \ell + 1$. For vectors $\mathbf{v}, \mathbf{w}$ we denote by $[\mathbf{v}, \mathbf{w}]_{+i}$ the random variable $(\mathbf{v}_{+i}, \mathbf{w})$ if $i \leq |\mathbf{v}| + 1$ and the random variable $\left(\mathbf{v}, \mathbf{w}_{+(i-|\mathbf{v}|-1)}\right)$ if $i > |\mathbf{v}| + 1$.

Definition 6.11 (Correctness of Random Extension)**.** *For every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell} - 1$ and index $i \in [\ell_1 + \ell_2 + 2]$:*

$$\left[\begin{array}{c|l} \Gamma_{+i} & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}\ ,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \Gamma_{+i} \leftarrow \mathsf{RandExtend}(\Gamma, i) \end{array} \end{array}\right] \equiv \left[\begin{array}{c|l} \Gamma & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}\ ,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ (\mathbf{m}'_1, \mathbf{m}'_2) \leftarrow [\mathbf{m}_1, \mathbf{m}_2]_{+i} \\ (\mathbf{r}'_1, \mathbf{r}'_2) \leftarrow [\mathbf{r}_1, \mathbf{r}_2]_{+i} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}'_1, \mathbf{r}'_1), (\mathsf{sk}_2, \mathbf{m}'_2, \mathbf{r}'_2)) \end{array} \end{array}\right] .$$

6.2.5 Ciphertext rerandomization.

We add to the encryption scheme the algorithm Rerand with the following syntax:

Ciphertext rerandomization: the PPT ciphertext rerandomization algorithm Rerand takes as input the public parameters pp and a ciphertext Γ. It outputs a rerandomized ciphertext $\widehat{\Gamma}$. We extend the syntax of the algorithms in Sections 6.2.1 and 6.2.3 to rerandomized ciphertexts.

Next we define the weak completeness of the zero-test for rerandomized ciphertexts. This property is similar to the weak completeness of the zero-test (Definition 6.7) except here we require that the zero-test still passes if one vector of evaluated ciphertexts results from evaluating polynomials on a rerandomization of Γ.

Definition 6.12 (Weak Completeness of Zero-Test for Rerandomized Ciphertexts)**.** *For every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$, vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$ and polynomials $\left\{P_i, P'_i : \mathbb{F}^{\ell_1 + \ell_2} \to \mathbb{F}\right\}_{i \in [n]}$ of individual degree at most $\bar{\delta}$ such that $\sum_{i \in [n]} \alpha_i \cdot P_i \cdot P'_i \equiv 0$:*

$$\Pr\left[\mathsf{ZT}(\mathsf{pp}, \mathbf{e}, \mathbf{e}', \boldsymbol{\alpha}) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}\ ,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \widehat{\Gamma} \leftarrow \mathsf{Rerand}(\mathsf{pp}, \Gamma) \\ \mathbf{e} = (e_i \leftarrow \mathsf{Eval}(\Gamma, P_i)\ :\ i \in [n]) \\ \mathbf{e}' = (e'_i \leftarrow \mathsf{Eval}(\widehat{\Gamma}, P'_i)\ :\ i \in [n]) \end{array}\right] = 1 \ .$$

Next we define the unambiguity of decompositions for rerandomized ciphertexts. In the following definition, the adversary is given a ciphertext Γ and a rerandomization $\widehat{\Gamma}_{-i}$ with the i'th coordinate omitted. We require that the adversary cannot compute a ciphertext e along with two different pairs $(\mathbf{Q}^1, \widehat{\mathbf{Q}}^1) \neq (\mathbf{Q}^2, \widehat{\mathbf{Q}}^2)$ that each consist of a decomposition and rerandomized decomposition; namely, for every $k \in \{1, 2\}$, $\mathbf{Q}^k$ is the decomposition for e and $\widehat{\mathbf{Q}}^k$ is the rerandomization of $\mathbf{Q}^k$.

Definition 6.13 (Λ-Unambiguity of Decompositions for Rerandomized Ciphertexts)**.** *For every $\mathrm{poly}(\Lambda)$-size adversary* Adv *there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$*

and $i \in [\ell_1 + \ell_2]$:

$$\Pr\left[\begin{array}{l} \\ (\mathbf{Q}^1, \widehat{\mathbf{Q}}^1) \neq (\mathbf{Q}^2, \widehat{\mathbf{Q}}^2) \\ \forall k \in [2], j \in [0, \bar{\delta}] : \\ \quad \mathbf{Q}^k = \left(Q^k_\delta\right)_{\delta \in [0,\bar{\delta}]} \\ \quad \widehat{\mathbf{Q}}^k = \left(\widehat{Q}^k_\delta\right)_{\delta \in [0,\bar{\delta}]} \\ \quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\ \quad \mathsf{ZT}((Q^k_j \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}^k_j)) = 1 \\ \\ \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}\ ,\ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \widehat{\Gamma}_{-i} \leftarrow \mathsf{Rerand}(\mathsf{Restrict}(\Gamma, i)) \\ \left(e, \mathbf{Q}^1, \widehat{\mathbf{Q}}^1, \mathbf{Q}^2, \widehat{\mathbf{Q}}^2\right) \leftarrow \mathsf{Adv}(\mathsf{pp}, \Gamma, \widehat{\Gamma}_{-i}) \\ e_{-1} \leftarrow \mathsf{Eval}(\Gamma, \overline{-1}) \\ \widehat{e}_1 \leftarrow \mathsf{Eval}(\widehat{\Gamma}_{-i}, \overline{1}) \\ \mathbf{E} = \left(E_j \leftarrow \mathsf{Eval}(\Gamma, Z^j_i)\ :\ j \in [0, \bar{\delta}]\right) \end{array}\right] \leq \mu(\Lambda(\kappa))\ .$$

In the experiment above, $Z^j_i : \mathbb{F}^{\ell_1+\ell_2} \to \mathbb{F}$ denotes the monomial $Z^j_i(z_1, \ldots, z_{\ell_1+\ell_2}) \equiv z^j_i$.

In the above definition we implicitly require that the size of the ciphertext is bounded by the size of Adv. We therefore require that $\bar{\delta}^{\bar{\ell}} \leq \mathrm{poly}(\Lambda)$.

We note that this property holds more generally for any vector of evaluated ciphertexts $\mathbf{E}$ containing at least one ciphertext computed by homomorphically evaluating a polynomial F that depends on z_i.

The final definition. Putting together the definitions from Sections 6.2.1 to 6.2.5, we get the following definition.

Definition 6.14 (Zero-testable Homomorphic Encryption)**.** *A Λ-secure zero-testable homomorphic encryption scheme:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ satisfies the requirements in Definitions 6.1 to 6.13.

6.3 Construction

In this section we construct a zero-testable homomorphic encryption scheme from bilinear maps. We begin with notation.

Notation. For a vector of polynomials $\mathbf{C} = (C_1, \ldots, C_\ell) \in (\mathbb{F}[\mathbf{x}])^\ell$ and a vector $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_\ell) \in [0, \delta]^\ell$, we denote by $\mathbf{C}^{\boldsymbol{\delta}}$ the polynomial $\prod_{i \in [\ell]} C_i^{\delta_i}$.

For every security parameter $\kappa \in \mathbb{N}$, fix a group $G = G_\kappa$ of prime order $p = p(\kappa) = 2^{\Theta(\kappa)}$ with a non-degenerate bilinear map $e : G \times G \to G_T$ and let $\mathbb{F} = \mathbb{Z}_p$.

For any $t \in \mathbb{F}$ and $g \in G$, we refer to the element g^t as the *encoding* of t under g and denote it by $\langle t \rangle_g$. For any n-variate polynomial $P(\mathbf{x}) = \sum_{\boldsymbol{\delta} \in [0,\delta]^n} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{x}^{\boldsymbol{\delta}} \in \mathbb{F}[\mathbf{x}]$ of individual degree $\leq \delta$, the encoding of P under g consists of the encodings of its coefficients $\left(\langle \alpha_{\boldsymbol{\delta}} \rangle_g\right)_{\boldsymbol{\delta} \in [0,\delta]^n}$ and we denote it by $\langle P \rangle_g$. Observe that given the encoded polynomial $\langle P \rangle_g$ and the elements $\mathbf{t} \in \mathbb{F}^n$ we can homomorphically evaluate P on $\mathbf{t}$. That is, we can efficiently compute the encoding:

$$\langle P(\mathbf{t}) \rangle_g = \left\langle \sum_{\boldsymbol{\delta} \in [0,\delta]^n} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{t}^{\boldsymbol{\delta}} \right\rangle_g = \prod_{\boldsymbol{\delta} \in [0,\delta]^n} \left(\langle \alpha_{\boldsymbol{\delta}} \rangle_g\right)^{\mathbf{t}^{\boldsymbol{\delta}}}\ .$$

Similarly, we can efficiently compute $\langle P(\mathbf{t}) \rangle_g$ given the polynomial P and the encoded elements $\left(\left\langle \mathbf{t}^{\boldsymbol{\delta}} \right\rangle_g\right)_{\boldsymbol{\delta} \in [0,\delta]^n}$.

Next we describe the algorithms of our encryption scheme with degree bound $\bar{\delta} = \bar{\delta}(\kappa)$ and message length bound $\bar{\ell} = \bar{\ell}(\kappa)$.

The parameter generation algorithm $\mathsf{ParamGen}$**.** Given as input the security parameter κ the parameter generation algorithm samples a random group generator $g \leftarrow G$ and outputs the public parameters $\mathsf{pp} = (1^\kappa, g)$. In what follows, let $d = \bar{\ell} \cdot \bar{\delta}$ and $d' = 2d + 1$. The encoding $\langle \alpha \rangle_{\mathsf{pp}}$ of an element $\alpha \in \mathbb{F}$ is $\langle \alpha \rangle_g$.

The key generation algorithm KeyGen**.** Given as input the public parameters pp, the key generation algorithm samples a random element $s \leftarrow \mathbb{F}$ and outputs the secret key $\mathsf{sk} = (\mathsf{pp}, s)$.

The encryption algorithm Enc**.** Given as input the secret key $\mathsf{sk} = (\mathsf{pp}, s)$, a message $\mathbf{m} = (m_1, \ldots, m_\ell)$ such that $\ell \leq \bar{\ell}$ and randomness $\mathbf{r} = (r_1, \ldots, r_\ell) \in \{0,1\}^{\kappa \times \ell}$, the encryption algorithm samples a vector of polynomials $\mathbf{C} = (C_1, \ldots, C_\ell)$ as follows: for every $i \in [\ell]$ it uses randomness $r_i \in \{0,1\}^\kappa$ to sample a random polynomial $C_i \in \mathbb{F}[x]$ of degree d' satisfying $C_i(s) = m_i$. It computes and outputs the ciphertext:

$$c = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \in \mathbb{F}[x] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell} \quad .$$

The multi-key encryption algorithm MEnc**.** Given as input a pair (γ_1, γ_2) where γ_i is a tuple containing a secret key $\mathsf{sk}_i = (\mathsf{pp}, s_i)$, a message $\mathbf{m}_i = (m_{i,1}, \ldots, m_{i,\ell_i})$ such that $\ell_i \leq \bar{\ell}$ and randomness $\mathbf{r}_i = (r_{i,1}, \ldots, r_{i,\ell_i}) \in \{0,1\}^{\kappa \times \ell_i}$, the multi-key encryption algorithm proceeds as follows: for every $i \in [2]$ it samples a vector of polynomials $\mathbf{C}_i = (C_{i,1}, \ldots, C_{i,\ell_i})$ by using the randomness $r_{i,j} \in \{0,1\}^\kappa$ to sample a random polynomial $C_{i,j} \in \mathbb{F}[x_i]$ of degree d' satisfying $C_{i,j}(s_i) = m_{i,j}$. It computes and outputs the multi-key ciphertext:

$$\Gamma = \left(\left\langle (\mathbf{C}_1 \mid \mathbf{C}_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_1 + \ell_2}} \quad .$$

If $\ell_1 = 0$ or $\ell_2 = 0$ then the encoded polynomials in Γ are treated as polynomials in a single variable.

The homomorphic evaluation algorithm Eval**.** The homomorphic evaluation algorithm is given as input:

- The public parameters pp.
- A (single-key or multi-key) ciphertext $\Gamma = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell}$ of length $\ell = \ell(\Gamma)$.
- An ℓ-variate polynomial $P(\mathbf{x}) = \sum_{\boldsymbol{\delta} \in [0, \delta]^\ell} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{x}^{\boldsymbol{\delta}}$ of individual degree $\delta \leq \bar{\delta}$.

The homomorphic evaluation algorithm computes and outputs the evaluated ciphertext:

$$e = \langle P(\mathbf{C}) \rangle_g = \left\langle \sum_{\boldsymbol{\delta} \in [0, \delta]^\ell} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g \quad .$$

Note that if Γ is single-key (resp. multi-key), then e is single-key (resp. multi-key). Since P is an ℓ-variate polynomial of individual degree $\leq \bar{\delta}$ and each element of $\mathbf{C}$ is a polynomial of degree d', the individual degree of $P(\mathbf{C})$ is at most:

$$\delta_{\max} = \bar{\ell} \cdot \bar{\delta} \cdot d' = 2\bar{\ell}^2 \bar{\delta}^2 + \bar{\ell}\bar{\delta} \quad .$$

We therefore define the set Valid of valid evaluated ciphertexts to be the set of all encoded polynomials in $\mathbb{F}[x]$ or $\mathbb{F}[x_1, x_2]$ of individual degree $\leq \delta_{\max}$.

The decryption algorithm Dec**.** The decryption algorithm operates on either single-key or multi-key evaluated ciphertexts. In the single-key case, it is given as input a secret key $\mathsf{sk} = (\mathsf{pp}, s)$ and a single-key evaluated ciphertext $e = \langle R \in \mathbb{F}[x] \rangle_g$. In the multi-key case, it is given as input a pair of secret keys $(\mathsf{sk}_1, \mathsf{sk}_2)$ such that $\mathsf{sk}_i = (\mathsf{pp}, s_i)$ and a multi-key evaluated ciphertext $e = \langle R \in \mathbb{F}[x_1, x_2] \rangle_g$. Let $\mathbf{s} = (s)$ in the single-key case and $\mathbf{s} = (s_1, s_2)$ in the multi-key case. The decryption algorithm outputs the encoding $\langle R(\mathbf{s}) \rangle_g$.

The quadratic zero-test algorithm ZT**.** The quadratic zero-test algorithm is given as input:

- The public parameters pp.
- Two vectors of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n), \mathbf{e}' = (e'_1, \ldots, e'_n)$ where $e_i = \langle R_i \in \mathbb{F}[x] \rangle_g, e'_i = \langle R'_i \in \mathbb{F}[x] \rangle_g$.
- A vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}^n$.

If for some $i \in [n]$, $e_i \notin \mathsf{Valid}$ or $e'_i \notin \mathsf{Valid}$ then it outputs 0. Using the bilinear map it computes the encoded polynomial:

$$\left\langle \sum_{i \in [n]} \alpha_i \cdot R_i R'_i \right\rangle_{e(g,g)} .$$

If this encoded polynomial is $\langle \mathbf{0} \rangle_{e(g,g)}$ (that is, if every coefficient is zero) then it outputs 1. Otherwise it outputs 0.

The ciphertext extension algorithm Extend**.** Given as input γ_1 and γ_2 such that for some $i \in [2]$, γ_i is a single-key ciphertext $c = \left(\left\langle \mathbf{C}_i^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_i}}$ and for $j \in [2] \setminus \{i\}$, γ_j is a tuple $(\mathsf{sk}, \mathbf{m}, \mathbf{r})$ including a secret key $\mathsf{sk} = (\mathsf{pp}, s)$, a message $\mathbf{m} = (m_1, \ldots, m_{\ell_j})$ such that $\ell_j \leq \bar{\ell}$, and randomness $\mathbf{r} = (r_1, \ldots, r_{\ell_j}) \in \{0,1\}^{\kappa \times \ell_j}$, the ciphertext extension algorithm samples a vector of polynomials $\mathbf{C}_j = (C_{j,1}, \ldots, C_{j,\ell_j})$ as follows: for every $k \in [\ell_j]$ it uses the randomness $r_k \in \{0,1\}^\kappa$ to sample a random polynomial $C_{j,k} \in \mathbb{F}[x_j]$ of degree d' satisfying $C_{j,k}(s) = m_k$. It computes and outputs the multi-key ciphertext:

$$\Gamma = \left(\left\langle (\mathbf{C}_1 \mid \mathbf{C}_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_1 + \ell_2}} .$$

Note that the encoded polynomials in Γ can be computed efficiently since $\mathbf{C}_j$ is not encoded.

The ciphertext restriction algorithm $\mathsf{Restrict}$**.** Given as input the public parameters pp, a ciphertext $\Gamma = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell}$ and an index $i \in [\ell]$, the ciphertext restriction algorithm outputs the ciphertext:

$$\Gamma_{-i} = \left(\left\langle (\mathbf{C}_{-i})^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell - 1}} = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell \,:\, \boldsymbol{\delta}_i = 0} .$$

The ciphertext random extension algorithm $\mathsf{RandExtend}$**.** Given as input the public parameters pp, a multi-key ciphertext $\Gamma = \left(\left\langle (\mathbf{C}_1 \mid \mathbf{C}_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_1 + \ell_2}}$ and $i \in [\ell_1 + \ell_2 + 2]$, if $i \leq \ell_1 + 1$, the ciphertext random extension algorithm samples a random polynomial $C' \in \mathbb{F}[x_1]$ of degree d' and sets $\mathbf{C}'_1 = (C_{1,1}, \ldots, C_{1,i-1}, C', C_{1,i}, \ldots, C_{1,\ell_1})$ if $i \leq \ell_1$ or $\mathbf{C}'_1 = (\mathbf{C}_1 \mid C')$ if $i = \ell_1 + 1$. It computes and outputs the ciphertext:

$$\Gamma' = \left(\left\langle (\mathbf{C}'_1 \mid \mathbf{C}_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_1 + \ell_2 + 1}} .$$

Note that the encoded polynomials in Γ' can be computed efficiently since C' is not encoded.

Alternatively, if $i > \ell_1 + 1$, it samples a random polynomial $C' \in \mathbb{F}[x_2]$, and for $i' = i - \ell_1 - 1$ it sets $\mathbf{C}'_2 = (C_{2,1}, \ldots, C_{2,i'-1}, C', C_{2,i'}, \ldots, C_{2,\ell_2})$ if $i' \leq \ell_2$ or $\mathbf{C}'_2 = (\mathbf{C}_2 \mid C')$ if $i' = \ell_2 + 1$. It computes and outputs the ciphertext:

$$\Gamma' = \left(\left\langle (\mathbf{C}_1 \mid \mathbf{C}'_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{\ell_1 + \ell_2 + 1}} .$$

The ciphertext rerandomization algorithm Rerand**.** Given as input the public parameters pp and a ciphertext $\Gamma = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell}$, the rerandomization algorithm samples a random element $\lambda \leftarrow \mathbb{F}$. It computes and outputs the ciphertext:

$$\widehat{\Gamma} = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_{g^\lambda} \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell} .$$

6.4 Analysis

In this section we prove the construction in Section 6.3 with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) under the following hardness assumption parameterized by $\bar{\delta}$ and Λ. For $\bar{\delta} = O(1)$ this is identical to Assumption 5.1.

Assumption 6.15. *For every $d(\kappa) = O(\bar{\delta} \cdot \log \Lambda(\kappa))$ and $\mathrm{poly}(\Lambda)$-size adversary Adv, there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ g \leftarrow G \\ s \leftarrow \mathbb{Z}_p \\ t_0 \leftarrow \mathbb{Z}_p \\ t_1 \leftarrow s^{2d+1} \\ b' \leftarrow \mathsf{Adv}\left(\left(\left\langle s^i \cdot t_b^j \right\rangle_g \right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}} \right) \end{array} \right] \leq \frac{1}{2} + \mu(\Lambda(\kappa)) \enspace .$$

Theorem 6.16. *For any $\Lambda = \Lambda(\kappa) \leq |\mathbb{F}|^{o(\kappa)}$, $\bar{\delta} = O(1)$ and $\bar{\ell}(\kappa)$ such that $\bar{\delta}^{\bar{\ell}} \leq \mathrm{poly}(\Lambda)$ the encryption scheme:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

given in Section 6.3 is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) under Assumption 6.15.

To prove Theorem 6.16, we focus on proving that our construction satisfies the following properties: semantic security (Definition 6.5) unambiguity of ciphertexts (Definition 6.6) and unambiguity of zero-test for rerandomized ciphertexts (Definition 6.13). The remaining requirements in Definition 6.14 follow from the construction.

6.4.1 Semantic security.

Assume towards contradiction that there exists a $\mathrm{poly}(\Lambda)$-size adversary Adv and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$, there exists $\ell \leq \bar{\ell}$ and $\mathbf{m}^0, \mathbf{m}^1 \in \mathbb{F}^\ell$ such that:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}^0, \mathbf{r}) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}^1, \mathbf{r}) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array} \right] \geq \frac{1}{2} + \frac{1}{p(\Lambda(\kappa))} \enspace .$$

Fix any such κ, ℓ and $\mathbf{m}^0, \mathbf{m}^1$. For every $i \in [0, \ell]$ define the message $\bar{\mathbf{m}}^i \in \mathbb{F}^\ell$ as follows: let $\bar{\mathbf{m}}^i_j = \mathbf{m}^1_j$ for $j \leq i$ and $\bar{\mathbf{m}}^i_j = \mathbf{m}^0_j$ for $j > i$. Therefore, there exists $i^* \in [\ell]$ such that:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, \bar{\mathbf{m}}^{i^*-1}, \mathbf{r}) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, \bar{\mathbf{m}}^{i^*}, \mathbf{r}) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array} \right] \geq \frac{1}{2} + \frac{1}{\ell \cdot p(\Lambda(\kappa))} \enspace . \tag{2}$$

Fix such $i^* \in [\ell]$. Next we construct an adversary breaking Assumption 6.15 with parameter $\bar{\delta}$ and $d = \bar{\delta} \cdot \bar{\ell}$. Since $\bar{\delta} = O(1)$ and $\bar{\delta}^{\bar{\ell}} \leq \mathrm{poly}(\Lambda)$ we have $\bar{\ell} = O(\log \Lambda)$ and indeed $d(\kappa) = O(\bar{\delta} \cdot \log \Lambda(\kappa))$. First we construct an adversary $\mathsf{Adv}'_{\mathbf{m}}$ for any $\mathbf{m} \in \mathbb{F}^{\ell}$ such that:

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} g \leftarrow G \\ s \leftarrow \mathbb{F} \\ t \leftarrow s^{2d+1} \\ b \leftarrow \mathsf{Adv}'_{\mathbf{m}}\left(\left(\langle s^i \cdot t^j \rangle_g\right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}}\right) \end{array}\right] = \Pr\left[b = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r} \leftarrow \{0,1\}^{\kappa \times \ell} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r}) \\ b \leftarrow \mathsf{Adv}(\mathsf{pp}, c) \end{array}\right] \tag{3}$$

and

$$\Pr_{\substack{g \leftarrow G \\ s,t \leftarrow \mathbb{F}}}\left[\mathsf{Adv}'_{\bar{\mathbf{m}}^{i^*}}\left(\left(\langle s^i \cdot t^j \rangle_g\right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}}\right) = 1\right] = \Pr_{\substack{g \leftarrow G \\ s,t \leftarrow \mathbb{F}}}\left[\mathsf{Adv}'_{\bar{\mathbf{m}}^{i^*-1}}\left(\left(\langle s^i \cdot t^j \rangle_g\right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}}\right) = 1\right] . \tag{4}$$

By Equations (2) and (3):

$$\Pr_{\substack{g \leftarrow G \\ s \leftarrow \mathbb{F} \\ t \leftarrow s^{2d+1}}}\left[\mathsf{Adv}'_{\bar{\mathbf{m}}^{i^*}}\left(\left(\langle s^i \cdot t^j \rangle_g\right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}}\right) = 1\right] - \Pr_{\substack{g \leftarrow G \\ s \leftarrow \mathbb{F} \\ t \leftarrow s^{2d+1}}}\left[\mathsf{Adv}'_{\bar{\mathbf{m}}^{i^*-1}}\left(\left(\langle s^i \cdot t^j \rangle_g\right)_{\substack{i \in [0,d] \\ j \in [0,\bar{\delta}]}}\right) = 1\right] \geq \frac{2}{\ell \cdot p(\Lambda(\kappa))} .$$

Therefore, for some $\mathbf{m} \in \left\{\bar{\mathbf{m}}^{i^*}, \bar{\mathbf{m}}^{i^*-1}\right\}$, $\mathsf{Adv}'_{\mathbf{m}}$ breaks Assumption 6.15.

Now we construct $\mathsf{Adv}'_{\mathbf{m}}$ for $\mathbf{m} = (m_1, \ldots, m_\ell)$. Given as input the encodings $\langle s^i \cdot t^j \rangle_g$ for every $i \in [0, d]$ and $j \in [0, \bar{\delta}]$, the adversary $\mathsf{Adv}'_{\mathbf{m}}$ proceeds as follows:

1. Let $\mathsf{pp} = (1^\kappa, g)$ and $d' = 2d + 1$.
2. For every $i \in [\ell]$ sample a random polynomial $C'_i \in \mathbb{F}[x]$ of degree $d' - 1$.
3. For $i \in [\ell] \setminus \{i^*\}$ let $C_i = x \cdot C'_i - s \cdot C'_i + m_i$. Observe that C_i is distributed like a random degree-d' polynomial subject to $C_i(s) = m_i$. Note that the adversary cannot compute the polynomial C_i since it is only given encodings of s.
4. Let $C_{i^*} = x \cdot C'_{i^*} - s \cdot C'_{i^*} + m_{i^*} + (x^{d'} - t)$. Observe that if $t = s^{d'}$ then C_i is distributed like a random degree-d' polynomial subject to $C_i(s) = m_{i^*}$. If t is random and independent of s, then C_i is distributed like a random degree-d' polynomial.
5. Use the encodings $\langle s^i \cdot t^j \rangle_g$ for $i \in [0, d]$ and $j \in [0, \bar{\delta}]$ to compute the ciphertext:

$$c = \left(\langle (C_1, \ldots, C_\ell)^{\boldsymbol{\delta}} \rangle_g\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^\ell} .$$

   Observe that for every $i \in [\ell]$, every coefficient of C_i depends linearly on s. Moreover, the free coefficient of C_{i^*} depends linearly on t. Therefore, every encoded polynomial in c can indeed be computed from the input.
6. Output the bit returned by $\mathsf{Adv}(\mathsf{pp}, c)$.

By construction, if $t = s^{d'}$ then the ciphertext c generated by $\mathsf{Adv}'_{\mathbf{m}}$ is distributed exactly like an encryption of $\mathbf{m}$ and hence Equation (3) follows. If t is random and independent of s, then the output of $\mathsf{Adv}'_{\mathbf{m}}$ is independent of m_{i^*}. Since $\bar{\mathbf{m}}^{i^*-1}$ and $\bar{\mathbf{m}}^{i^*}$ only differ in location i^*, Equation (4) follows.

6.4.2 Unambiguity of ciphertexts

We prove the property for multi-key ciphertexts. The proof for single-key ciphertexts is similar. We first define the distribution of ciphertexts $\mathsf{RandEnc}(\mathsf{pp}, \ell_1, \ell_2)$ for public parameters pp and $\ell_1, \ell_2 \in [0, \bar{\ell}]$. RandEnc generates a ciphertext as follows:

1. Let $d' = 2 \cdot \bar{\delta} \cdot \bar{\ell} + 1$.
2. For every $i \in [2]$ sample a vector of polynomials $\mathbf{C}_i = (C_{i,1}, \ldots, C_{i,\ell_i})$ such that for every $j \in [\ell_i]$, $C_{i,j} \in \mathbb{F}[x_i]$ is a random polynomial of degree d'.
3. Compute and output the ciphertext:

$$\left(\left\langle (\mathbf{C}_1 \mid \mathbf{C}_2)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^{\ell_1 + \ell_2}} \quad .$$

It follows from the construction of MEnc and RandEnc that for any $\kappa \in \mathbb{N}$ and $\ell_1, \ell_2 \in [0, \bar{\ell}]$, the distribution of $(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2, \Gamma)$ in Definition 6.6 is the same as the distribution of $(\mathsf{pp}, \mathsf{sk}_1, \mathsf{sk}_2, \Gamma')$ sampled as follows:

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \Gamma' \leftarrow \mathsf{RandEnc}(\mathsf{pp}, \ell_1, \ell_2) \quad . \end{array}$$

Therefore it suffices to show that for every adversary Adv, $\kappa \in \mathbb{N}$ and $\ell_1, \ell_2 \in [0, \bar{\ell}]$:

$$\Pr \left[\begin{array}{l|l} & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ e, e' \in \mathsf{Valid} & \Gamma' \leftarrow \mathsf{RandEnc}(\mathsf{pp}, \ell_1, \ell_2) \\ e \neq e' & (e, e') \leftarrow \mathsf{Adv}(\mathsf{pp}, \Gamma') \\ v = v' & \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ & v \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e) \\ & v' \leftarrow \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e') \end{array} \right] \leq \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \quad .$$

Fix any $\kappa \in \mathbb{N}$ and $\ell_1, \ell_2 \in [0, \bar{\ell}]$. In this experiment Adv outputs evaluated ciphertexts $e = \langle R \rangle_g$ and $e' = \langle R' \rangle_g$ such that $R, R' \in \mathbb{F}[x_1, x_2]$. If $e, e' \in \mathsf{Valid}$ then the individual degree of R, R' is at most $\delta_{\max} = 2\bar{\ell}^2\bar{\delta}^2 + \bar{\ell}\bar{\delta}$. If $e \neq e'$ then $(R - R') \in \mathbb{F}[x_1, x_2]$ is a nonzero polynomial of individual degree $\leq \delta_{\max}$.

Since $\mathsf{sk}_1 = (\mathsf{pp}, s_1)$ and $\mathsf{sk}_2 = (\mathsf{pp}, s_2)$ for random elements $s_1, s_2 \leftarrow \mathbb{F}$ that are independent of R, R', if $e \neq e'$ then $(R - R')(s_1, s_2) = 0$ with probability at most $2\delta_{\max}/|\mathbb{F}|$ so $v = \langle R(s_1, s_2) \rangle_g = \langle R'(s_1, s_2) \rangle_g = v'$ with probability at most $2\delta_{\max}/|\mathbb{F}| = \mathrm{poly}(\bar{\delta} \cdot \bar{\ell})/|\mathbb{F}|$.

6.4.3 Unambiguity of zero-test for rerandomized ciphertexts

The proof considers a sequence of experiments. The first experiment, Exp_0 corresponds to Definition 6.13:

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times \ell_1}, \ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times \ell_2} \\ \Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2)) \\ \widehat{\Gamma}_{-i} \leftarrow \mathsf{Rerand}(\mathsf{Restrict}(\Gamma, i)) \\ \left(e, \mathbf{Q}^1 = (Q^1_\delta)_{\delta \in [0,\bar{\delta}]}, \widehat{\mathbf{Q}}^1 = \left(\widehat{Q}^1_\delta\right)_{\delta \in [0,\bar{\delta}]}, \mathbf{Q}^2 = (Q^2_\delta)_{\delta \in [0,\bar{\delta}]}, \widehat{\mathbf{Q}}^2 = \left(\widehat{Q}^2_\delta\right)_{\delta \in [0,\bar{\delta}]} \right) \leftarrow \mathsf{Adv}(\mathsf{pp}, \Gamma, \widehat{\Gamma}_{-i}) \\ e_{-1} \leftarrow \mathsf{Eval}(\Gamma, \overline{-1}) \\ \widehat{e}_1 \leftarrow \mathsf{Eval}(\widehat{\Gamma}_{-i}, \overline{1}) \\ \mathbf{E} = \left(E_j \leftarrow \mathsf{Eval}(\Gamma, Z_i^j) \ : \ j \in [0, \bar{\delta}] \right) \end{array} \quad .$$

Assume towards contradiction that there exists a $\mathrm{poly}(\Lambda)$-size adversary Adv and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$ there exist messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$ and $i \in [\ell_1 + \ell_2]$ such that:

$$\Pr_{\mathsf{Exp}_0}\left[\begin{array}{l} \exists j \in [0,\bar{\delta}] : (Q_j^1, \widehat{Q}_j^1) \neq (Q_j^2, \widehat{Q}_j^2) \\ \forall k \in [2], j \in [0,\bar{\delta}] : \\ \quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\ \quad \mathsf{ZT}((Q_j^k \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}_j^k)) = 1 \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ .$$

Let Exp_1 be the experiment that is defined like Exp_0 except that we sample random independent messages $\mathbf{m}_1 \leftarrow \mathbb{F}^{\ell_1}$ and $\mathbf{m}_2 \leftarrow \mathbb{F}^{\ell_2}$. By semantic security (Definition 6.5) for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}_1}\left[\begin{array}{l} \exists j \in [0,\bar{\delta}] : (Q_j^1, \widehat{Q}_j^1) \neq (Q_j^2, \widehat{Q}_j^2) \\ \forall k \in [2], j \in [0,\bar{\delta}] : \\ \quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\ \quad \mathsf{ZT}((Q_j^k \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}_j^k)) = 1 \end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ .$$

For $i \in [2]$ let $\mathsf{sk}_i = (\mathsf{pp}, s_i)$. For $k \in [2]$ and $j \in [0,\bar{\delta}]$ let $Q_j^k = \left\langle R_j^k \right\rangle_g$ and $\widehat{Q}_j^k = \left\langle \widehat{R}_j^k \right\rangle_g$. Let:

$$\left\langle \alpha_j^k \right\rangle_g = \left\langle R_j^k(s_1, s_2) \right\rangle_g = \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), Q_j^k) \quad , \quad \left\langle \beta_j^k \right\rangle_g = \left\langle \widehat{R}_j^k(s_1, s_2) \right\rangle_g = \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), \widehat{Q}_j^k) \ ,$$

and let $\alpha_j = \alpha_j^1 - \alpha_j^2$ and $\beta_j = \beta_j^1 - \beta_j^2$.

Recall that ZT only accepts if $Q_j^k, \widehat{Q}_j^k \in \mathsf{Valid}$ for all $k \in [2], j \in [0,\bar{\delta}]$. Therefore, by the unambiguity of ciphertexts (Definition 6.6) for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}_1}\left[\begin{array}{l} \exists j \in [0,\bar{\delta}] : \left(\left\langle \alpha_j \right\rangle_g, \left\langle \beta_j \right\rangle_g\right) \neq \left(\left\langle 0 \right\rangle_g, \left\langle 0 \right\rangle_g\right) \\ \forall k \in [2], j \in [0,\bar{\delta}] : \\ \quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\ \quad \mathsf{ZT}((Q_j^k \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}_j^k)) = 1 \end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} - \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} = \frac{\Omega(1)}{p(\Lambda(\kappa))} \ .$$

Since $\bar{\delta} = O(1)$, $\bar{\delta}^{\bar{\ell}} \leq \mathrm{poly}(\Lambda)$ and $\Lambda \leq |\mathbb{F}|^{o(\kappa)}$, indeed $\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})/|\mathbb{F}| = \mathrm{negl}(\Lambda(\kappa))$.

Next we define an algorithm S that samples ciphertexts $(\Gamma, \widehat{\Gamma}_{-i})$ given encoded group elements. Given as input the encodings $\left\langle s^j \right\rangle_g$ for $j \in [0,\bar{\delta}]$ and $\left\langle t \right\rangle_g$ the algorithm S samples Γ that encrypts a random message whose i'th element is s and $\widehat{\Gamma}_{-i}$ that is a rerandomized version of Γ encoded in base g^t with the i'th element removed. The algorithm S proceeds as follows:

1. For every $j \in [\ell_1] \setminus \{i\}$ sample a random polynomial $C_j \in \mathbb{F}[x_1]$ of degree $d' = 2 \cdot \bar{\delta} \cdot \bar{\ell} + 1$. For every $j \in [\ell_1 + 1, \ell_2] \setminus \{i\}$ sample a random polynomial $C_j \in \mathbb{F}[x_2]$ of degree d'.
2. If $i \leq \ell_1$, sample a random polynomial $C_i' \in \mathbb{F}[x_1]$ of degree d' such that then $C_i'(s_1) = 0$. If $i > \ell_1$, sample a random polynomial $C_i' \in \mathbb{F}[x_2]$ of degree d' such that then $C_i'(s_2) = 0$. Let $C_i = C_i' + s$.
3. Let $\mathbf{C} = (C_1, \ldots, C_{\ell_1+\ell_2})$. Use the encodings $\left(\left\langle s^i \right\rangle_g\right)_{i \in [0,\bar{\delta}]}$ to compute the ciphertext:
$$\Gamma = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^{\ell_1+\ell_2}} \ .$$
4. Use the encoding $\left\langle t \right\rangle_g$ to compute the ciphertext:
$$\widehat{\Gamma}_{-i} = \left(\left\langle \mathbf{C}_{-i}^{\boldsymbol{\delta}} \right\rangle_{g^t}\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^{\ell_1+\ell_2-1}} \ .$$

   Observe that in Γ and $\widehat{\Gamma}_{-i}$, only the constant coefficient of C_i is encoded so Γ and $\widehat{\Gamma}_{-i}$ can indeed be computed from the input.

5. Output $(\Gamma, \widehat{\Gamma}_{-i})$.

Let Exp_2 be the experiment that is defined like Exp_1 except that the ciphertexts $(\Gamma, \widehat{\Gamma}_{-i})$ are generated by S on encodings of random elements $s, t \leftarrow \mathbb{F}$:

$$
\begin{array}{l}
s \leftarrow \mathbb{F} \\
t \leftarrow \mathbb{F} \\
\mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\
\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\
\left(\Gamma, \widehat{\Gamma}_{-i}\right) \leftarrow \mathsf{S}\left(\left(\left\langle s^j \right\rangle_g\right)_{j \in [0,\bar{\delta}]}, \langle t \rangle_g\right) \\
\left(e, \mathbf{Q}^1 = \left(Q^1_\delta\right)_{\delta \in [0,\bar{\delta}]}, \widehat{\mathbf{Q}}^1 = \left(\widehat{Q}^1_\delta\right)_{\delta \in [0,\bar{\delta}]}, \mathbf{Q}^2 = \left(Q^2_\delta\right)_{\delta \in [0,\bar{\delta}]}, \widehat{\mathbf{Q}}^2 = \left(\widehat{Q}^2_\delta\right)_{\delta \in [0,\bar{\delta}]}\right) \leftarrow \mathsf{Adv}(\mathsf{pp}, \Gamma, \widehat{\Gamma}_{-i}) \\
e_{-1} \leftarrow \mathsf{Eval}(\Gamma, \overline{-1}) \\
\widehat{e}_1 \leftarrow \mathsf{Eval}(\widehat{\Gamma}_{-i}, \overline{1}) \\
\mathbf{E} = \left(E_j \leftarrow \mathsf{Eval}(\Gamma, Z^j_i) \ : \ j \in [0, \bar{\delta}]\right)
\end{array}
.
$$

We argue that $\Gamma, \widehat{\Gamma}_{-i}$ have the same distribution in Exp_1 and Exp_2. S samples Γ by encrypting messages $\mathbf{m}_1, \mathbf{m}_2$ such that $(\mathbf{m}_1 \mid \mathbf{m}_2)_{-i}$ consists of independent random elements of $\mathbb{F}$ and $(\mathbf{m}_1 \mid \mathbf{m}_2)_i = s$. Since $s \leftarrow \mathbb{F}$ in Exp_2, the distribution of Γ in both experiments is identical. In Exp_1 we compute:

$$
\widehat{\Gamma}_{-i} = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}}_{-i} \right\rangle_{g^\lambda}\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^{\ell_1+\ell_2-1}} \leftarrow \mathsf{Rerand}\left(\mathsf{Restrict}\left(\Gamma = \left(\left\langle \mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^{\ell_1+\ell_2}}, i\right)\right) \ ,
$$

for a random element $\lambda \leftarrow \mathbb{F}$. Since $t \leftarrow \mathbb{F}$ in Exp_2, the distribution of $\widehat{\Gamma}_{-i}$ in both experiments is identical.

Since $\Gamma, \widehat{\Gamma}_{-i}$ have the same distribution in Exp_1 and Exp_2 we have that for infinitely many $\kappa \in \mathbb{N}$:

$$
\Pr_{\mathsf{Exp}_2}\left[\begin{array}{l}
\exists j \in [0,\bar{\delta}] : \left(\langle \alpha_j \rangle_g, \langle \beta_j \rangle_g\right) \neq \left(\langle 0 \rangle_g, \langle 0 \rangle_g\right) \\
\forall k \in [2], j \in [0,\bar{\delta}] : \\
\quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\
\quad \mathsf{ZT}((Q^k_j \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}^k_j)) = 1
\end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ .
$$

Let Exp_3 be the experiment that is defined like Exp_1 except that set $t = s^{2\bar{\delta}+1}$ instead of sampling an independent random element $t \leftarrow \mathbb{F}$. By Assumption 6.15 with parameter $\bar{\delta}$ and $d = \bar{\delta}$ we have that for infinitely many $\kappa \in \mathbb{N}$:

$$
\Pr_{\mathsf{Exp}_3}\left[\begin{array}{l}
\exists j \in [0,\bar{\delta}] : \left(\langle \alpha_j \rangle_g, \langle \beta_j \rangle_g\right) \neq \left(\langle 0 \rangle_g, \langle 0 \rangle_g\right) \\
\forall k \in [2], j \in [0,\bar{\delta}] : \\
\quad \mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1 \\
\quad \mathsf{ZT}((Q^k_j \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}^k_j)) = 1
\end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ . \tag{5}
$$

Using Equation (5) we construct an adversary Adv' breaking Assumption 6.15 with parameter $\bar{\delta}$ and $d = 2\bar{\delta}+1$. Given as input the encodings $\left\langle s^i \cdot t^j \right\rangle_g$ for every $i \in [0,d]$ and $j \in [0,\bar{\delta}]$, Adv' uses only the encodings of s to emulate Exp_3. Let AC be the event tested in Equation (5). If AC does not occur Adv' outputs a random bit. Otherwise, if AC occurs, let j' be the maximal index such that $\left(\langle \alpha_{j'} \rangle_g, \langle \beta_{j'} \rangle_g\right) \neq \left(\langle 0 \rangle_g, \langle 0 \rangle_g\right)$. Adv' test that:

$$
e\left(\langle \alpha_{j'} \rangle_g, \langle t \rangle_g\right) \cdot \prod_{j \in [j']} e\left(\langle \beta_{j'-j} \rangle_g, \left\langle s^{d+1-j} \right\rangle_g\right) = 1 \in G_T \ . \tag{6}
$$

If this test passes, Adv' output 1. Otherwise it output 0. Note that emulating Adv requires time $\mathrm{poly}(\Lambda)$ and the rest of the execution requires time $\mathrm{poly}(\kappa)$.

Claim 6.17.

$$\Pr_{\mathsf{Adv}'}\left[\mathsf{AC} \quad \rightarrow \quad \alpha_{j'} \cdot s^{2d+1} + \sum_{j \in [j']} \beta_{j'-j} \cdot s^{d+1-j} = 0\right] = 1 \ .$$

Proof. Assume AC occurs and, therefore:

1. $\left(\langle \alpha'_j \rangle_g , \langle \beta'_j \rangle_g\right) \neq \left(\langle 0 \rangle_g , \langle 0 \rangle_g\right)$
2. $\forall j > j' : \left(\langle \alpha_j \rangle_g , \langle \beta_j \rangle_g\right) = \left(\langle 0 \rangle_g , \langle 0 \rangle_g\right)$
3. For every $k \in [2]$: $\mathsf{ZT}((\mathbf{E} \mid e_{-1}), (\mathbf{Q}^k \mid e)) = 1$
4. For every $k \in [2], j \in [0, \bar{\delta}]$: $\mathsf{ZT}((Q_j^k \mid e_{-1}), (\widehat{e}_1 \mid \widehat{Q}_j^k)) = 1$

Let $e = \langle R \rangle_g$ and $\langle \alpha \rangle_g = \langle R(s_1, s_2) \rangle_g = \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e)$. Recall that:

$$\begin{aligned}
e_{-1} = \mathsf{Eval}(\Gamma, \overline{-1}) = \langle \overline{-1} \rangle_g \quad &, \quad \langle \overline{-1}(s_1, s_2) \rangle_g = \langle -1 \rangle_g \\
\widehat{e}_1 = \mathsf{Eval}(\widehat{\Gamma}_{-i}, \overline{1}) = \langle \overline{1} \rangle_{g^{s^d}} \quad &, \quad \langle \overline{1}(s_1, s_2) \rangle_{g^{s^d}} = \langle 1 \rangle_{g^{s^d}} = \langle s^d \rangle_g \\
E_j = \mathsf{Eval}(\Gamma, Z_i^j) = \left\langle Z_i^j(\mathbf{C}) \right\rangle_g \quad &, \quad \left\langle Z_i^j(\mathbf{C})(s_1, s_2) \right\rangle_g = \langle s^j \rangle_g
\end{aligned}$$

Therefore, since ZT accepts we have that for all $k \in [2], j \in [0, \bar{\delta}]$:

$$\sum_{j \in [0, \bar{\delta}]} \alpha_j^k \cdot s^j = \alpha \quad \Rightarrow \quad \sum_{j \in [0, \bar{\delta}]} \alpha_j \cdot s^j = 0 \tag{7}$$

$$\alpha_j^k \cdot s^d = \beta_j^k \quad \Rightarrow \quad \alpha_j \cdot s^d = \beta_j \ . \tag{8}$$

Since $\alpha_j = 0$ for $j > j'$, by Equation (7):

$$\sum_{j \in [0, j']} \alpha_j \cdot s^j = 0 \quad \Rightarrow \quad \sum_{j \in [0, j']} \alpha_j \cdot s^{2d+1-j'+j} = 0 \quad \Rightarrow \quad \sum_{j \in [0, j']} \alpha_{j'-j} \cdot s^{2d+1-j} = 0 \ .$$

By Equation (8):

$$\alpha_{j'} \cdot s^{2d+1} + \sum_{j \in [j']} \beta_{j'-j} \cdot s^{d+1-j} = 0 \ .$$

□

By Equation (5) and Claim 6.17 for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}_b}\left[\mathsf{AC} \quad \wedge \quad \alpha_{j'} \cdot s^{2d+1} + \sum_{j \in [j']} \beta_{j'-j} \cdot s^{d+1-j} = 0\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ .$$

Therefore, if $t = s^{2d+1}$ then Equation (6) holds and Adv outputs 1 with probability $\geq \Omega(1)/p(\Lambda(\kappa))$. If $t \leftarrow \mathbb{F}$ is independent of s then Equation (6) holds with probability at most $1/|\mathbb{F}| = \mathrm{negl}(\Lambda(\kappa))$. Therefore, Adv'' contradicts Assumption 6.15.

7 Auxiliary Protocols

In this section we introduce a sequence of sub-protocols used in our unambiguous quasi-argument construction. The protocols in this section are defined with respect to the interface of the encryption scheme introduced in Section 6 and they follow a common framework. Given a (single-key or multi-key) ciphertext Γ as input, we run the protocol's setup algorithm to generate a prover key pk (which includes Γ) and a verifier key vk. The prover is given as input pk and some polynomial (or a sequence of polynomials) to be evaluated homomorphically on Γ. The polynomial must belong to some restricted set of valid polynomials specified by the protocols. The prover sends to the verifier a single message that contains the evaluated ciphertext (or ciphertexts) together with a proof. Intuitively, this proof certifies that the evaluated polynomial is indeed in the set of valid polynomials. Using the verification key vk the verifier can check the proof against the evaluated ciphertext and either accept or reject. While the setup and prover algorithms may run in time that is polynomial in 2^ℓ (the length of Γ), we require that the running time of the verifier is polynomial in ℓ and the security parameter.

For each protocol we define completeness, efficiency, soundness and unambiguity requirements. The completeness requirement says that when the prover honestly evaluates a valid polynomial on Γ the verifier accepts. The soundness requirement says that if the verifier accepts, then the values encrypted in the evaluated ciphertext and in Γ satisfy some conditions. Intuitively, these conditions should only be satisfied if the prover honestly evaluates a valid polynomial. The unambiguity requirement says that it is hard to find two different accepting proofs for the same evaluated ciphertext with respect to the same verification key.

7.1 Notation

Let:

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

be a zero-testable homomorphic encryption scheme with degree bound $\bar{\delta} = \bar{\delta}(\kappa) = O(1)$, message length bound $\bar{\ell} = \bar{\ell}(\kappa)$ and field $\mathbb{F} = \mathbb{F}_\kappa$ (Definition 6.14). The interface of the protocols in this section are defined with respect to this scheme. We use the notation introduced in Section 6.1 as well as the following shorthand:

Encryption: for a message $\mathbf{m}$, we denote by $\mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}}$ the output $(\mathsf{sk}, c, \mathbf{r})$ in the following experiment:

$$\begin{aligned}&\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp})\\ &\mathbf{r} \leftarrow \{0,1\}^{\kappa \times |\mathbf{m}|}\\ &c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{m}, \mathbf{r})\end{aligned}$$

Multi-key encryption: for messages $\mathbf{m}_1, \mathbf{m}_2$, we denote by $\mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}}$ the output:

$$\left(\mathbf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2), \Gamma, \mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2)\right) \ ,$$

in the following experiment:

$$\begin{aligned}&\mathsf{sk}_1, \mathsf{sk}_2 \leftarrow \mathsf{KeyGen}(\mathsf{pp})\\ &\mathbf{r}_1 \leftarrow \{0,1\}^{\kappa \times |\mathbf{m}_1|}, \ \mathbf{r}_2 \leftarrow \{0,1\}^{\kappa \times |\mathbf{m}_2|}\\ &\Gamma \leftarrow \mathsf{MEnc}((\mathsf{sk}_1, \mathbf{m}_1, \mathbf{r}_1), (\mathsf{sk}_2, \mathbf{m}_2, \mathbf{r}_2))\end{aligned}$$

Vector encryption: for a vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ we denote by $\mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \mathbf{R}$ the output:

$$\left(\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_K), \mathbf{c} = (c_1, \ldots, c_K), \mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_K)\right) \ ,$$

in the following experiment:

$$\begin{aligned}&\forall k \in [K] : \ \mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{pp})\\ &\forall k \in [K] : \ \mathbf{r}_k \leftarrow \{0,1\}^{\kappa \times \ell}\\ &\forall k \in [K] : \ c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, \mathbf{m}_k, \mathbf{r}_k)\end{aligned}$$

Homomorphic evaluation: for a ciphertext Γ and a polynomial P, we denote by $\left[P(\Gamma)\right]_{\mathsf{pp}}$ the output of $\mathsf{Eval}(\mathsf{pp}, \Gamma, P)$.

Quadratic zero-test: for two vectors of evaluated ciphertexts $\mathbf{a}, \mathbf{a}'$ of length n_1 and two vectors of evaluated ciphertexts $\mathbf{b}, \mathbf{b}'$ of length n_2, we denote by $\left[\mathbf{a} \cdot \mathbf{a}' = \mathbf{b} \cdot \mathbf{b}'\right]_{\mathsf{pp}}$ the output of $\mathsf{ZT}(\mathsf{pp}, \mathbf{e}, \mathbf{e}', \boldsymbol{\alpha})$ where $\mathbf{e} = (\mathbf{a} \mid \mathbf{b}), \mathbf{e}' = (\mathbf{a}' \mid \mathbf{b}')$ and $\boldsymbol{\alpha} = (1^{n_1} \mid (-1)^{n_2})$.

7.2 Decomposition Protocol

This section describes the decomposition protocol. The protocol's CRS contains a ciphertext Γ encrypting a message $\mathbf{m} = (m_1, \ldots, m_\ell) \in \mathbb{F}^\ell$. The honest prover is given a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. The prover homomorphically evaluates F on Γ and sends the resulting ciphertext e to the verifier. The prover also provides a decomposition $\mathbf{Q} = (Q_0, \ldots, Q_{\bar{\delta}})$ of e along some coordinate $i \in [\ell]$. The decomposition is computed as follows: let $F|_{i,0}, \ldots, F|_{i,\bar{\delta}} : \mathbb{F}^{\ell-1} \to \mathbb{F}$ be the unique polynomials such that:

$$F(\mathbf{z}) \equiv \sum_{j \in [0,\bar{\delta}]} F|_{i,j}(\mathbf{z}_{-i}) \cdot \mathbf{z}_i^j \ .$$

The ciphertext Q_j is obtained by homomorphically evaluating $F|_{i,j}$ on Γ_{-i} which is obtained from Γ by removing the encryption of m_i.

The verifier uses the zero-test to check that the decomposition $\mathbf{Q}$ is consistent with the evaluation e. In more detail, the verifier interprets the decomposition $\mathbf{Q}$ as an encryption of a univariate degree-$\bar{\delta}$ polynomial. Using the zero-test the verifier checks that the evaluation of this polynomial on m_i is indeed equal to the value encrypted in e.

This simple decomposition protocol is ambiguous, meaning that a cheating prover can produce different decompositions of the same evaluated ciphertext e that are both accepted by the verifier. Such an attack can be carried out by homomorphically computing a decomposition that depends on m_i (and the honest decomposition that does not depend on m_i). To prevent this attack and ensure unambiguous decompositions we add to the CRS a rerandomization $\widehat{\Gamma}_{-i}$ of the ciphertext Γ_{-i} which contains no information about m_i. The prover also provides a rerandomized decomposition $\widehat{Q}_0, \ldots, \widehat{Q}_{\bar{\delta}}$ where $\widehat{Q}_j$ is obtained by homomorphically evaluating $F|_{i,j}$ on $\widehat{\Gamma}_{-i}$. The verifier will also check that the rerandomized decomposition and the original one are consistent. The unambiguity of the resulting protocol follows directly from the unambiguity of decompositions for rerandomized ciphertexts property of the encryption (Definition 6.13).

Another useful property of the decomposition protocol is injectivity, meaning that different evaluated ciphertexts cannot have the same decomposition under the same CRS.

7.2.1 Definition.

The decomposition protocol consists of algorithms $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with the following syntax:

Setup: The PPT setup algorithm $\mathsf{DC.S}$ takes as input public parameters pp for the encryption scheme, a ciphertext Γ and an index $i \in [\ell(\Gamma)]$. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic polynomial-time prover algorithm $\mathsf{DC.P}$ takes as input a prover key pk and a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. It outputs an evaluated ciphertext e, a decomposition $\mathbf{Q}$ and a proof Π.

Verifier: The deterministic polynomial-time verifier algorithm $\mathsf{DC.V}$ takes as input a verifier key vk, an evaluated ciphertext e, a decomposition $\mathbf{Q}$ and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 7.1. *A Λ-secure decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$, index $i \in [\ell_1 + \ell_2]$ and polynomial $F : \mathbb{F}^{\ell_1+\ell_2} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[\begin{array}{l} \mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi) = 1 \\ e = [F(\Gamma)]_{\mathsf{pp}} \\ \forall j \in [0,\bar{\delta}] \ : \ Q_j = \left[F|_{i,j}(\Gamma_{-i})\right]_{\mathsf{pp}} \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}[\mathbf{m}_1, \mathbf{m}_2]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ \left(e, \mathbf{Q} = (Q_j)_{j \in [0,\bar{\delta}]}, \Pi\right) \leftarrow \mathsf{DC.P}(\mathsf{pk}, F) \\ \Gamma_{-i} \leftarrow \mathsf{Restrict}(\Gamma, i) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above* $|\mathsf{vk}| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$ *and* $|\Pi| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$.

Soundness. *For every* $\kappa \in \mathbb{N}$*, message* $\mathbf{m} \in \mathbb{F}^{\ell}$ *such that* $\ell \leq \bar{\ell}$*, index* $i \in [\ell]$*, and adversary* Adv*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}\left(\mathsf{vk}, e, \mathbf{Q}, \Pi\right) = 1 \\ \alpha \neq \sum\limits_{j \in [0,\bar{\delta}]} \beta_j \cdot z^j \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ Z = (\mathsf{sk}, c, \mathbf{r}) \leftarrow \mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, c, i) \\ \left(e, \mathbf{Q} = (Q_j)_{j \in [0,\bar{\delta}]}, \Pi\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ \alpha \leftarrow \mathsf{Dec}(\mathsf{sk}, e) \\ \forall j \in [0, \bar{\delta}] \ : \ \beta_j \leftarrow \mathsf{Dec}(\mathsf{sk}, Q_j) \\ z \leftarrow \langle \mathbf{m}_i \rangle_{\mathsf{pp}} \end{array} \right] = 0 \ .$$

Similarly, for every $\kappa \in \mathbb{N}$*, messages* $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ *such that* $\ell_1, \ell_2 \leq \bar{\ell}$*, index* $i \in [\ell_1 + \ell_2]$ *and adversary* Adv*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}\left(\mathsf{vk}, e, \mathbf{Q}, \Pi\right) = 1 \\ \alpha \neq \sum\limits_{j \in [0,\bar{\delta}]} \beta_j \cdot z^j \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ Z = (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ \left(e, \mathbf{Q} = (Q_j)_{j \in [0,\bar{\delta}]}, \Pi\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ \alpha \leftarrow \mathsf{Dec}(\mathbf{sk}, e) \\ \forall j \in [0, \bar{\delta}] \ : \ \beta_j \leftarrow \mathsf{Dec}(\mathbf{sk}, Q_j) \\ z \leftarrow \langle (\mathbf{m}_1 \mid \mathbf{m}_2)_i \rangle_{\mathsf{pp}} \end{array} \right] = 0 \ .$$

Λ**-Injectivity.** *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*, messages* $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ *such that* $\ell_1, \ell_2 \leq \bar{\ell}$ *and index* $i \in [\ell_1 + \ell_2]$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}\left(\mathsf{vk}, e, \mathbf{Q}, \Pi\right) = 1 \\ \mathsf{DC.V}\left(\mathsf{vk}, e', \mathbf{Q}, \Pi'\right) = 1 \\ e \neq e' \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ (e, \mathbf{Q}, \Pi, e', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array} \right] \leq \mu(\Lambda(\kappa)) \ .$$

Λ**-Unambiguity.** *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*, messages* $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}, \mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ *such that* $\ell_1, \ell_2 \leq \bar{\ell}$ *and index* $i \in [\ell_1 + \ell_2]$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}\left(\mathsf{vk}, e, \mathbf{Q}, \Pi\right) = 1 \\ \mathsf{DC.V}\left(\mathsf{vk}, e, \mathbf{Q}', \Pi'\right) = 1 \\ (\mathbf{Q}, \Pi) \neq (\mathbf{Q}', \Pi') \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ (e, \mathbf{Q}, \Pi, \mathbf{Q}', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array} \right] \leq \mu(\Lambda(\kappa)) \ .$$

7.2.2 Construction.

We construct a decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ as follows.

The setup algorithm $\mathsf{DC.S}$**.** The setup algorithm is given as input:

- Public parameters pp for the encryption scheme.
- A ciphertext Γ with $\ell = \ell(\Gamma)$.
- An index $i \in [\ell]$.

It proceeds as follows:

- Set $\widehat{\Gamma}_{-i} \leftarrow \mathsf{Rerand}(\mathsf{Restrict}(\Gamma, i))$.
- Set $\mathbf{E} = \left(E_j \leftarrow \left[Z_i^j(\Gamma)\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]}$ where $Z_i^j : \mathbb{F}^\ell \to \mathbb{F}$ denotes the monomial $Z_i^j(z_1,\ldots,z_\ell) \equiv z_i^j$.
- Set $e_1 \leftarrow \left[\overline{1}(\Gamma)\right]_{\mathsf{pp}}$ and $\widehat{e}_1 \leftarrow \left[\overline{1}(\widehat{\Gamma}_{-i})\right]_{\mathsf{pp}}$.
- Output the prover key $\mathsf{pk} = \left(\mathsf{pp}, \Gamma, i, \widehat{\Gamma}_{-i}\right)$ and verifier key $\mathsf{vk} = (\mathsf{pp}, \mathbf{E}, e_1, \widehat{e}_1)$.

The prover algorithm $\mathsf{DC.P}$**.** The prover algorithm is given as input:

- A prover key $\mathsf{pk} = \left(\mathsf{pp}, \Gamma, i, \widehat{\Gamma}_{-i}\right)$.
- A polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$.

It proceeds as follows:

- Set $e \leftarrow [F(\Gamma)]_{\mathsf{pp}}$.
- Set $\Gamma_{-i} \leftarrow \mathsf{Restrict}(\Gamma, i)$.
- Set $\mathbf{Q} = \left(Q_j \leftarrow \left[F|_{i,j}(\Gamma_{-i})\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]}$.
- Set $\Pi = \left(\widehat{Q}_j \leftarrow \left[F|_{i,j}(\widehat{\Gamma}_{-i})\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]}$.
- Output the evaluated ciphertext e, decomposition $\mathbf{Q}$ and proof Π.

The verifier algorithm $\mathsf{DC.V}$**.** The verifier algorithm is given as input:

- A verifier key $\mathsf{vk} = \left(\mathsf{pp}, \mathbf{E} = (E_j)_{j\in[0,\bar{\delta}]}, e_1, \widehat{e}_1\right)$.
- An evaluated ciphertext e.
- A decomposition $\mathbf{Q} = (Q_j)_{j\in[0,\bar{\delta}]}$.
- A proof $\Pi = (\widehat{Q}_j)_{j\in[0,\bar{\delta}]}$.

It proceeds as follows:

- Test that $e \in \mathsf{Valid}$ (see Definition 6.6).
- Test that $[\mathbf{Q}\cdot\mathbf{E} = e\cdot e_1]_{\mathsf{pp}}$.
- For every $j \in [0,\bar{\delta}]$ test that $\left[Q_j \cdot \widehat{e}_1 = e_1 \cdot \widehat{Q}_j\right]_{\mathsf{pp}}$.
- Output 1 if all tests pass. Otherwise output 0.

7.2.3 Analysis.

In this section we prove the following theorem:

Theorem 7.2. *For any $\Lambda(\kappa)$, $\bar{\delta}(\kappa)$ and $\bar{\ell}(\kappa)$ such that $\Lambda\cdot\bar{\delta}\cdot\bar{\ell} = |\mathbb{F}|^{o(1)}$, assuming that:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$, the decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ given in Section 7.2.2 is a Λ-secure decomposition protocol (Definition 7.1) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

Completeness. Fix any $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}$, $\mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$, index $i \in [\ell_1 + \ell_2]$ and polynomial $F : \mathbb{F}^{\ell_1+\ell_2} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. Consider the completeness experiment where the setup algorithm output the keys:

$$\mathsf{pk} = \left(\mathsf{pp}, \Gamma, i, \widehat{\Gamma}_{-i}\right) \quad , \quad \mathsf{vk} = \left(\mathsf{pp}, \mathbf{E} = \left(E_j = \left[Z_i^j(\Gamma)\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]}, e_1 = \left[\overline{1}(\Gamma)\right]_{\mathsf{pp}}, \widehat{e}_1 = \left[\overline{1}(\widehat{\Gamma}_{-i})\right]_{\mathsf{pp}}\right) \ ,$$

and the prover outputs:

$$\left(e = [F(\Gamma)]_{\mathsf{pp}}, \mathbf{Q} = \left(Q_j = \left[F|_{i,j}(\Gamma_{-i})\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]} \quad , \quad \Pi = \left(\widehat{Q}_j = \left[F|_{i,j}(\widehat{\Gamma}_{-i})\right]_{\mathsf{pp}}\right)_{j\in[0,\bar{\delta}]}\right) \ .$$

Next we show that every test in DC.V passes:

- For every $j \in [0, \bar{\delta}]$ let $F'_{i,j} : \mathbb{F}^{\ell} \to \mathbb{F}$ be the polynomial such that $F'_{i,j}(\mathbf{z}) \equiv F|_{i,j}(\mathbf{z}_{-i})$. By the stability of evaluation (Definition 6.2) and the correctness of restriction (Definition 6.10) for every $j \in [0, \bar{\delta}]$, $Q_j = \left[F'_{i,j}(\Gamma)\right]_{\mathsf{pp}}$. By the weak completeness of the zero-test (Definition 6.7) $[\mathbf{Q} \cdot \mathbf{E} = e \cdot e_1]_{\mathsf{pp}}$ passes since:

$$\sum_{j\in[0,\bar{\delta}]} F'_{i,j} \cdot Z_i^j \equiv F \cdot \overline{1} \ .$$

- By the stability of evaluation (Definition 6.2) and the correctness of restriction (Definition 6.10) $e_1 = \left[\overline{1}(\Gamma_{-i})\right]_{\mathsf{pp}}$. By the weak completeness of zero-test for rerandomized ciphertexts (Definition 6.12) for every $j \in [0, \bar{\delta}]$, $\left[Q_j \cdot \widehat{e}_1 = e_1 \cdot \widehat{Q}_j\right]_{\mathsf{pp}}$ passes.

Soundness. We prove soundness in the multi-key setting. The single-key setting is analogous. Fix $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}$, $\mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$, index $i \in [\ell_1 + \ell_2]$ and adversary Adv. We consider the soundness experiment where the verifier's input includes:

$$\mathsf{vk} = \left(\mathsf{pp}, \mathbf{E} = (E_j)_{j\in[0,\bar{\delta}]}, e_1, \widehat{e}_1\right) \ , \ e \ , \ \mathbf{Q} = (Q_j)_{j\in[0,\bar{\delta}]} \ , \ \Pi = \left(\widehat{Q}_j\right)_{j\in[0,\bar{\delta}]} \ .$$

Assume the verifier accepts in the soundness experiment. By the correctness of evaluation (Definition 6.1) for every $j \in [0, \bar{\delta}]$:

$$\mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), E_j) = \left\langle (\mathbf{m}_1 \mid \mathbf{m}_2)_i^j \right\rangle_{\mathsf{pp}} \quad , \quad \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e_1) = \langle 1 \rangle_{\mathsf{pp}} \ .$$

Since $[\mathbf{Q} \cdot \mathbf{E} = e \cdot e_1]_{\mathsf{pp}}$ passes, the soundness of the zero-test (Definition 6.8) implies that:

$$\sum_{j\in[0,\bar{\delta}]} \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), Q_j) \cdot \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), E_j) = \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e) \cdot \mathsf{Dec}((\mathsf{sk}_1, \mathsf{sk}_2), e_1) = 0 \ .$$

Therefore:

$$\sum_{j\in[0,\bar{\delta}]} \beta_j \cdot z^j = \alpha \ .$$

Λ-Injectivity. Assume towards contradiction that there exists a $\mathrm{poly}(\Lambda)$-size adversary Adv and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$ there exist messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}$, $\mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$ and index $i \in [\ell_1 + \ell_2]$ such that:

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{DC.V}(\mathsf{vk}, e', \mathbf{Q}, \Pi') = 1 & (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ e \neq e' & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ & (e, \mathbf{Q}, \Pi, e', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ .$$

By semantic security (Definition 6.5) for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{DC.V}(\mathsf{vk}, e', \mathbf{Q}, \Pi') = 1 & \mathbf{m}_1 \leftarrow \mathbb{F}^{\ell_1}, \mathbf{m}_2 \leftarrow \mathbb{F}^{\ell_2} \\ e \neq e' & (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ & (e, \mathbf{Q}, \Pi, e', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ .$$

We construct an adversary Adv' that contradicts the unambiguity of ciphertexts (Definition 6.6). Adv' is given as input public parameters pp and a multi-key ciphertext Γ encrypting messages of lengths ℓ_1, ℓ_2 under $\mathbf{sk}$. It sets:

$$(\mathsf{pk}, \mathsf{vk} = (\mathsf{pp}, \mathbf{E}, e_1, \widehat{e}_1)) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \ ,$$
$$(e, \mathbf{Q}, \Pi, e', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \ ,$$

and outputs (e, e'). With probability $\rho \geq \Omega(1)/p(\Lambda(\kappa))$, both $\mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi)$ and $\mathsf{DC.V}(\mathsf{vk}, e', \mathbf{Q}, \Pi')$ accept and $e \neq e'$. In this case, $e, e' \in \mathsf{Valid}$ and both $\left[\mathbf{Q} \cdot \mathbf{E} = e \cdot e_1\right]_{\mathsf{pp}}$ and $\left[\mathbf{Q} \cdot \mathbf{E} = e' \cdot e_1\right]_{\mathsf{pp}}$ pass. By the soundness of the zero-test (Definition 6.8):

$$\mathsf{Dec}(\mathbf{sk}, e) \cdot \mathsf{Dec}(\mathbf{sk}, e_1) = \mathsf{Dec}(\mathbf{sk}, e') \cdot \mathsf{Dec}(\mathbf{sk}, e_1) \ .$$

By the correctness of evaluation (Definition 6.1) $\mathsf{Dec}(\mathbf{sk}, e_1) = \langle 1 \rangle_{\mathsf{pp}}$ and therefore:

$$\mathsf{Dec}(\mathbf{sk}, e) = \mathsf{Dec}(\mathbf{sk}, e') \ .$$

Overall, Adv' breaks the unambiguity of ciphertexts with probability $\rho \geq \Omega(1)/p(\Lambda(\kappa))$. Since $\Lambda \cdot \bar{\delta} \cdot \bar{\ell} = |\mathbb{F}|^{o(1)}$, we have that $\rho \geq \mathrm{poly}(\bar{\delta} \cdot \bar{\ell})/|\mathbb{F}|$ for sufficiently large $\kappa \in \mathbb{N}$, in contradiction to the unambiguity of ciphertexts.

Λ-Unambiguity. Assume towards contradiction that there exists a $\mathrm{poly}(\Lambda)$-size adversary Adv and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$ there exist messages $\mathbf{m}_1 \in \mathbb{F}^{\ell_1}$, $\mathbf{m}_2 \in \mathbb{F}^{\ell_2}$ such that $\ell_1, \ell_2 \leq \bar{\ell}$ and index $i \in [\ell_1 + \ell_2]$ such that:

$$\Pr\left[\begin{array}{l|l} \mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}', \Pi') = 1 & (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathbf{Q}, \Pi) \neq (\mathbf{Q}', \Pi') & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma, i) \\ & (e, \mathbf{Q}, \Pi, \mathbf{Q}', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ . \tag{9}$$

We construct an adversary Adv' that contradicts the unambiguity of the zero-test for rerandomized ciphertexts (Definition 6.13). Given as input public parameters pp and ciphertexts Γ and $\widehat{\Gamma}_{-i}$ with $\ell(\Gamma) = \ell$ and $\ell(\widehat{\Gamma}_{-i}) = \ell - 1$, the adversary Adv' proceeds as follows:

1. Emulate the setup algorithm $\mathsf{DC.S}$ to compute $(\mathsf{pk}, \mathsf{vk})$:
   (a) Set $\mathbf{E} = \left(E_j \leftarrow \left[Z_i^j(\Gamma)\right]_{\mathsf{pp}}\right)_{j \in [0, \bar{\delta}]}$.
   (b) Set $e_1 \leftarrow \left[\overline{1}(\Gamma)\right]_{\mathsf{pp}}$ and $\widehat{e}_1 \leftarrow \left[\overline{1}(\widehat{\Gamma}_{-i})\right]_{\mathsf{pp}}$.
   (c) Set $\mathsf{pk} \leftarrow \left(\mathsf{pp}, \Gamma, i, \widehat{\Gamma}_{-i}\right)$ and $\mathsf{vk} \leftarrow (\mathsf{pp}, \mathbf{E}, e_1, \widehat{e}_1)$.
2. Set $(e, \mathbf{Q}, \Pi, \mathbf{Q}', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma)$ and output $(e, \mathbf{Q}, \Pi, \mathbf{Q}', \Pi')$.

When $(\mathsf{pp}, \Gamma, \widehat{\Gamma}_{-i})$ are distributed according to Definition 6.13, the keys $(\mathsf{pk}, \mathsf{vk})$ computed by Adv' are distributed as in Equation (9). Therefore with probability $\geq 1/p(\Lambda(\kappa))$, both $\mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}, \Pi)$ and $\mathsf{DC.V}(\mathsf{vk}, e, \mathbf{Q}', \Pi')$ accept and $(\mathbf{Q}, \Pi) \neq (\mathbf{Q}', \Pi')$ so the event in Definition 6.13 occurs. Therefore Adv' contradicts Definition 6.13.

7.3 Equality Protocol

This section describes the equality protocol. The protocol's CRS contains a multi-key ciphertext Γ encrypting two messages $\mathbf{m}_1, \mathbf{m}_2$ of equal length ℓ under two secret keys $\mathsf{sk}_1, \mathsf{sk}_2$. The honest prover is given a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. The prover homomorphically evaluates F on each of the messages encrypted in Γ and sends the resulting ciphertexts e_1, e_2 to the verifier. The prover also sends a proof to convince the verifier that indeed it evaluated the same polynomial on both messages. More formally, the soundness requirement is that if $\mathbf{m}_1 = \mathbf{m}_2$ and the verifier accepts the proof, then e_1 and e_2 encrypt the same value.

The equality protocol is constructed as follows. First, for every $i \in [0, \ell]$ we consider the hybrid multi-key ciphertext Γ^i encrypting the first i elements of $\mathbf{m}_1$ under sk_1 and the last $\ell - i$ elements of $\mathbf{m}_2$ under sk_2. In particular Γ^ℓ encrypts $\mathbf{m}_1$ and Γ^0 encrypts $\mathbf{m}_2$. The proof of equality consists of multiple decompositions of F. In more detail, for every $i \in [\ell]$, the prover decomposes F along its i'th coordinate by invoking the decomposition protocol twice: once evaluating F on Γ^i and once on Γ^{i-1}. This results in the evaluations A_1^i and A_2^i and the decompositions $\mathbf{Q}_1^i$ and $\mathbf{Q}_2^i$ respectively.

The verifier first checks that all decompositions are accepting. Next it checks that indeed, $A_1^\ell = e_1$, $A_2^1 = e_2$ and for every $i \in [\ell - 1]$, $A_1^i = A_2^{i+1}$. Finally it checks that for every $i \in [\ell]$, $\mathbf{Q}_1^i = \mathbf{Q}_2^i$. These decompositions should indeed be equal since Γ^i and Γ^{i-1} differ only on the i'th encrypted element. The intuition behind soundness is that if $\mathbf{m}_1 = \mathbf{m}_2$ and $\mathbf{Q}_1^i = \mathbf{Q}_2^i$, it follows from the soundness of the decomposition protocol that A_1^i and A_2^i encrypt the same value. By induction, e_1 and e_2 encrypt the same value as well.

The unambiguity of the decomposition protocol guarantees that this equality protocol is also unambiguous in the following sense: a cheating prover cannot find two different accepting equality proofs for (e_1, e_2) and (e_1', e_2') such that $e_1 = e_1'$ or $e_2 = e_2'$.

7.3.1 Definition.

The equality protocol consists of algorithms $(\mathsf{EQ.S}, \mathsf{EQ.P}, \mathsf{EQ.V})$ with the following syntax:

Setup: The PPT setup algorithm $\mathsf{EQ.S}$ takes as input public parameters pp for the encryption scheme and a multi-key ciphertext Γ encrypting two messages, each of length ℓ. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic polynomial-time prover algorithm $\mathsf{EQ.P}$ takes as input a prover key pk and a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. It outputs two single-key evaluated ciphertexts e_1, e_2 and a proof Π.

Verifier: The deterministic polynomial-time verifier algorithm $\mathsf{EQ.V}$ takes as input a verifier key vk, two single-key evaluated ciphertexts e_1, e_2 and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 7.3. *A Λ-secure equality protocol $(\mathsf{EQ.S}, \mathsf{EQ.P}, \mathsf{EQ.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{EQ.V}(\mathsf{vk}, e_1, e_2, \Pi) = 1 \\ \forall i \in [2] \ : \ e_i = [F(c_i)]_{\mathsf{pp}} \end{array} \middle| \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma) \\ (e_1, e_2, \Pi) \leftarrow \mathsf{EQ.P}(\mathsf{pk}, F) \\ c_1 \leftarrow \mathsf{Restrict}(\Gamma, [\ell + 1, 2\ell]) \\ c_2 \leftarrow \mathsf{Restrict}(\Gamma, [\ell]) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above $|\mathsf{vk}| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$ and $|\Pi| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$.*

Soundness. *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and adversary* Adv*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{EQ.V}(\mathsf{vk}, e_1, e_2, \Pi) = 1 \\ v_1 \neq v_2 \end{array} \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ Z = (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}, \mathbf{m}\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma) \\ (e_1, e_2, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ \forall i \in [2] \ : \ v_i \leftarrow \mathsf{Dec}(\mathbf{sk}_i, e_i) \end{array}\right] = 0 \ .$$

Λ-Unambiguity. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and index $i \in [2]$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{EQ.V}\left(\mathsf{vk}, e_1, e_2, \Pi\right) = 1 \\ \mathsf{EQ.V}\left(\mathsf{vk}, e'_1, e'_2, \Pi'\right) = 1 \\ (e_1, e_2, \Pi) \neq (e'_1, e'_2, \Pi') \\ e_i = e'_i \end{array} \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma) \\ (e_1, e_2, \Pi, e'_1, e'_2, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

7.3.2 Construction.

We construct an equality protocol $(\mathsf{EQ.S}, \mathsf{EQ.P}, \mathsf{EQ.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ as follows. The construction uses a decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

The setup algorithm $\mathsf{EQ.S}$. The setup algorithm $\mathsf{EQ.S}$ is given as input:

- Public parameters pp for the encryption scheme.
- A multi-key ciphertext Γ encrypting two messages, each of length ℓ.

It proceeds as follows:

- For every $i \in [0, \ell]$ set $\Gamma_i \leftarrow \mathsf{Restrict}\left(\Gamma, [i+1, i+\ell]\right)$.
- For every $i \in [\ell]$ set $(\mathsf{DC.pk}_i, \mathsf{DC.vk}_i) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma_i, i)$ and $(\mathsf{DC.}\tilde{\mathsf{pk}}_i, \mathsf{DC.}\tilde{\mathsf{vk}}_i) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma_{i-1}, i)$.
- Output the prover key and verifier key:

$$\mathsf{pk} = \left(\mathsf{pp}, \Gamma, \left(\mathsf{DC.pk}_i, \mathsf{DC.}\tilde{\mathsf{pk}}_i\right)_{i \in [\ell]}\right) \quad , \quad \mathsf{vk} = \left(\mathsf{DC.vk}_i, \mathsf{DC.}\tilde{\mathsf{vk}}_i\right)_{i \in [\ell]} \ .$$

The prover algorithm $\mathsf{EQ.P}$. The prover algorithm $\mathsf{EQ.P}$ is given as input:

- A prover key $\mathsf{pk} = \left(\mathsf{pp}, \Gamma, \left(\mathsf{DC.pk}_i, \mathsf{DC.}\tilde{\mathsf{pk}}_i\right)_{i \in [\ell]}\right)$.
- A polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$.

It proceeds as follows:

- Set $c_1 \leftarrow \mathsf{Restrict}(\Gamma, [\ell+1, 2\ell])$ and $c_2 \leftarrow \mathsf{Restrict}(\Gamma, [\ell])$.
- Set $e_1 \leftarrow [F(c_1)]_{\mathsf{pp}}$ and $e_2 \leftarrow [F(c_2)]_{\mathsf{pp}}$.
- For every $i \in [\ell]$ set $(A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i) \leftarrow \mathsf{DC.P}(\mathsf{DC.pk}_i, F)$ and $(\tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i) \leftarrow \mathsf{DC.P}(\mathsf{DC.}\tilde{\mathsf{pk}}_i, F)$.
- Output the single-key evaluated ciphertexts and proof:

$$e_1 \ , \ e_2 \ , \ \Pi = \left(A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i\right)_{i \in [\ell]} \ .$$

The verifier algorithm $\mathsf{EQ.V}$**.** The verifier algorithm $\mathsf{EQ.V}$ is given as input:

- A verifier key $\mathsf{vk} = \left(\mathsf{DC.vk}_i, \mathsf{DC.}\tilde{\mathsf{vk}}_i\right)_{i\in[\ell]}$.
- Two single-key evaluated ciphertexts e_1 and e_2.
- A proof $\Pi = \left(A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i\right)_{i\in[\ell]}$.

It proceeds as follows:

- Test that $e_1 = A_\ell$ and $e_2 = \tilde{A}_1$.
- For every $i \in [\ell]$ test that $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)$ and $\mathsf{DC.V}(\mathsf{DC.}\tilde{\mathsf{vk}}_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i)$ both accepts.
- For every $i \in [\ell - 1]$ test that $A_i = \tilde{A}_{i+1}$.
- For every $i \in [\ell]$ test that $\mathbf{Q}_i = \tilde{\mathbf{Q}}_i$.
- Output 1 if all tests pass. Otherwise output 0.

7.3.3 Analysis.

In this section we prove the following theorem:

Theorem 7.4. *For any $\Lambda(\kappa)$, $\bar{\delta}(\kappa)$ and $\bar{\ell}(\kappa)$ such that $\bar{\ell} \leq \Lambda^{O(1)}$ and $\Lambda \cdot \bar{\delta} = |\mathbb{F}|^{o(1)}$, assuming that:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$, the equality protocol $(\mathsf{EQ.S}, \mathsf{EQ.P}, \mathsf{EQ.V})$ given in Section 7.3.2 is a Λ-secure equality protocol (Definition 7.3) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

Completeness. Fix any $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. We consider the completeness experiment and show that each of the verifier's tests passes.

- The setup algorithm sets $\Gamma_\ell = c_1$ and $(\mathsf{DC.pk}_\ell, \mathsf{DC.vk}_\ell) \leftarrow \mathsf{DC.S}(\mathsf{pp}, c_1, \ell)$. The prover sets $e_1 \leftarrow [F(c_1)]_{\mathsf{pp}}$ and $(A_\ell, \mathbf{Q}_\ell, \mathsf{DC.}\Pi_\ell) \leftarrow \mathsf{DC.P}(\mathsf{DC.pk}_\ell, F)$. By the correctness of restriction (Definition 6.10) c_1 is in the support of $\mathsf{MEnc}\,[\mathbf{m}_1, \mathcal{E}]_{\mathsf{pp}}$. By the correctness of the decomposition protocol (Definition 7.1) $A_\ell = [F(c_1)]_{\mathsf{pp}} = e_1$.

  Similarly, the setup algorithm sets $\Gamma_0 = c_2$ and $(\mathsf{DC.}\tilde{\mathsf{pk}}_1, \mathsf{DC.}\tilde{\mathsf{vk}}_1) \leftarrow \mathsf{DC.S}(\mathsf{pp}, c_2, 1)$. The prover sets $e_2 \leftarrow [F(c_2)]_{\mathsf{pp}}$ and $(\tilde{A}_1, \tilde{\mathbf{Q}}_1, \mathsf{DC.}\tilde{\Pi}_1) \leftarrow \mathsf{DC.P}(\mathsf{DC.}\tilde{\mathsf{pk}}_1, F)$. By the correctness of restriction (Definition 6.10) c_2 is in the support of $\mathsf{MEnc}\,[\mathcal{E}, \mathbf{m}_2]_{\mathsf{pp}}$. By the correctness of the decomposition protocol (Definition 7.1) $\tilde{A}_1 = [F(c_2)]_{\mathsf{pp}} = e_2$.
- By the correctness of restriction (Definition 6.10) for every $i \in [0, \ell]$ the ciphertexts Γ_i computed by the setup algorithm are in the support of $\mathsf{MEnc}\,[\mathbf{m}'_1, \mathbf{m}'_2]_{\mathsf{pp}}$ for some $\mathbf{m}'_1 \in \mathbb{F}^i, \mathbf{m}'_2 \in \mathbb{F}^{\ell-i}$. Therefore, by the completeness of the decomposition protocol (Definition 7.1) for every $i \in [\ell]$, $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)$ and $\mathsf{DC.V}(\mathsf{DC.}\tilde{\mathsf{vk}}_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i)$ both accept.
- By the completeness of the decomposition protocol (Definition 7.1) for every $i \in [\ell]$, $A_i = [F(\Gamma_i)]_{\mathsf{pp}}$ and $\tilde{A}_i = [F(\Gamma_{i-1})]_{\mathsf{pp}}$. Therefore for every $i \in [\ell - 1]$, $A_i = \tilde{A}_{i+1}$.
- By the correctness of restriction (Definition 6.10) for every $i \in [\ell]$:

  $$\mathsf{Restrict}(\Gamma_i, i) = \mathsf{Restrict}(\Gamma, [i, \ell + i]) = \mathsf{Restrict}(\Gamma_{i-1}, i) \ .$$

  Therefore, by the completeness of the decomposition protocol (Definition 7.1) $\mathbf{Q}_i = \tilde{\mathbf{Q}}_i$.

Soundness. Fix any $\kappa \in \mathbb{N}$, message $\mathbf{m} = (m_1, \ldots, m_\ell) \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and adversary Adv. We consider the soundness experiment. By the correctness of restriction (Definition 6.10) for every $i \in [0, \ell]$ the ciphertext Γ_i computed by the setup algorithm is in the support of:

$$\mathsf{MEnc}\left[(m_1, \ldots, m_i), (m_{i+1}, \ldots, m_\ell)\right]_{\mathsf{pp}} \quad ,$$

encrypted under $\mathbf{sk}$. Let the output of Adv be:

$$\left(e_1, e_2, \Pi = \left(A_i, \mathbf{Q}_i = \left(Q_{i,0}, \ldots, Q_{i,\bar{\delta}}\right), \mathsf{DC.}\Pi_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i = \left(\tilde{Q}_{i,0}, \ldots, \tilde{Q}_{i,\bar{\delta}}\right), \mathsf{DC.}\tilde{\Pi}_i\right)_{i \in [\ell]}\right) \quad .$$

For every $i \in [\ell]$ and $j \in [0, \bar{\delta}]$ let:

$$\alpha_i = \mathsf{Dec}(\mathbf{sk}, A_i) \ , \ \beta_{i,j} = \mathsf{Dec}(\mathbf{sk}, Q_{i,j}) \ , \ \tilde{\alpha}_i = \mathsf{Dec}(\mathbf{sk}, \tilde{A}_i) \ , \ \tilde{\beta}_{i,j} = \mathsf{Dec}(\mathbf{sk}, \tilde{Q}_{i,j}) \ .$$

The tests in $\mathsf{EQ.V}$ imply that:

1. $v_1 = \alpha_\ell$ and $v_2 = \tilde{\alpha}_1$.
2. By the correctness of restriction (Definition 6.10) and the soundness of the decomposition protocol (Definition 7.1) for every $i \in [\ell]$:

$$\alpha_i = \sum_{j \in [0, \bar{\delta}]} \beta_{i,j} \cdot \left\langle m_i^j \right\rangle_{\mathsf{pp}} \quad , \quad \tilde{\alpha}_i = \sum_{j \in [0, \bar{\delta}]} \tilde{\beta}_{i,j} \cdot \left\langle m_i^j \right\rangle_{\mathsf{pp}} \quad .$$

3. For every $i \in [\ell - 1]$: $\alpha_i = \tilde{\alpha}_{i+1}$.
4. For every $i \in [\ell], j \in [0, \bar{\delta}]$: $\beta_{i,j} = \tilde{\beta}_{i,j}$.

The above equalities imply that $v_1 = v_2$.

Λ-Unambiguity. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv, $\kappa \in \mathbb{N}$, messages $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and an index $j \in [2]$ and let Exp denote the unambiguity experiment:

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ \left(\mathsf{pk}, \mathsf{vk} = \left(\mathsf{DC.vk}_i, \mathsf{DC.}\tilde{\mathsf{vk}}_i\right)_{i \in [\ell]}\right) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma) \\ (e_1, e_2, \Pi, e_1', e_2', \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \Gamma) \end{array}$$

where:

$$\Pi = \left(A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i, \tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i\right)_{i \in [\ell]} \quad , \quad \Pi' = \left(A_i', \mathbf{Q}_i', \mathsf{DC.}\Pi_i', \tilde{A}_i', \tilde{\mathbf{Q}}_i', \mathsf{DC.}\tilde{\Pi}_i'\right)_{i \in [\ell]}$$

Let AC be the event that $\mathsf{EQ.V}(\mathsf{vk}, e_1, e_2, \Pi)$ and $\mathsf{EQ.V}(\mathsf{vk}, e_1', e_2', \Pi')$ both accept and $e_j = e_j'$. We need to show that:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ (e_1, e_2, \Pi) \neq (e_1', e_2', \Pi')\right] \leq \mathrm{negl}(\Lambda(\kappa)) \quad . \tag{10}$$

We prove Equation (10) for $j = 1$. The proof for $j = 2$ is analogous. We rely on the following claim.

Claim 7.5. *For every $i \in [\ell]$:*

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ \mathbf{Q}_i = \mathbf{Q}_i' \ \rightarrow \ \mathsf{AC} \ \wedge \ A_i = A_i' \ \rightarrow \ (\mathbf{Q}_i, \mathsf{DC.}\Pi_i) = (\mathbf{Q}_i', \mathsf{DC.}\Pi_i')\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \quad .$$

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ \tilde{\mathbf{Q}}_i = \tilde{\mathbf{Q}}_i' \ \rightarrow \ \mathsf{AC} \ \wedge \ \tilde{A}_i = \tilde{A}_i' \ \rightarrow \ (\tilde{\mathbf{Q}}_i, \mathsf{DC.}\tilde{\Pi}_i) = \left(\tilde{\mathbf{Q}}_i', \mathsf{DC.}\tilde{\Pi}_i'\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \quad .$$

Before proving the claim we use it to prove Equation (10). If AC occurs then:

- By Claim 7.5 for every $i \in [\ell]$:

$$\Pr_{\mathsf{Exp}}[\mathsf{AC} \ \wedge \ A_i = A'_i \ \rightarrow \ (\mathbf{Q}_i, \mathsf{DC}.\Pi_i) = (\mathbf{Q}'_i, \mathsf{DC}.\Pi'_i)] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- Since $\mathbf{Q}_i = \tilde{\mathbf{Q}}_i$ and $\mathbf{Q}'_i = \tilde{\mathbf{Q}}'_i$ we have that $\mathbf{Q}_i = \mathbf{Q}'_i \ \Rightarrow \ \tilde{\mathbf{Q}}_i = \tilde{\mathbf{Q}}'_i$. Therefore, by Claim 7.5:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ A_i = A'_i \quad \rightarrow \quad \begin{array}{c}(A_i, \mathbf{Q}_i, \mathsf{DC}.\Pi_i) = (A'_i, \mathbf{Q}'_i, \mathsf{DC}.\Pi'_i) \\ \left(\tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC}.\tilde{\Pi}_i\right) = \left(\tilde{A}'_i, \tilde{\mathbf{Q}}'_i, \mathsf{DC}.\tilde{\Pi}'_i\right)\end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- $A_\ell = e_1 = e'_1 = A'_\ell$. Also, since $A_i = \tilde{A}_{i+1}$ and $A'_i = \tilde{A}'_{i+1}$, we have that $\tilde{A}_i = \tilde{A}'_i \ \Rightarrow \ A_{i-1} = A'_{i-1}$. Therefore:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \quad \rightarrow \quad \begin{array}{c}(A_i, \mathbf{Q}_i, \mathsf{DC}.\Pi_i)_{i\in[\ell]} = (A'_i, \mathbf{Q}'_i, \mathsf{DC}.\Pi'_i)_{i\in[\ell]} \\ \left(\tilde{A}_i, \tilde{\mathbf{Q}}_i, \mathsf{DC}.\tilde{\Pi}_i\right)_{i\in[\ell]} = \left(\tilde{A}'_i, \tilde{\mathbf{Q}}'_i, \mathsf{DC}.\tilde{\Pi}'_i\right)_{i\in[\ell]}\end{array}\right] \geq 1 - \ell \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

- Since $e_2 = \tilde{A}_1$ and $e'_2 = \tilde{A}'_1$, we have that $\tilde{A}_1 = \tilde{A}'_1 \Rightarrow e_2 = e'_2$.

Since $\bar{\ell} = \Lambda^{O(1)}$, Equation (10) follows. It remains to prove Claim 7.5.

Proof of Claim 7.5. Fix $i^* \in [\ell]$. We prove the first part of the claim:

$$\Pr_{\mathsf{Exp}}[\mathsf{AC} \ \wedge \ \mathbf{Q}_{i^*} = \mathbf{Q}'_{i^*} \ \rightarrow \ \mathsf{AC} \ \wedge \ A_{i^*} = A'_{i^*} \ \rightarrow \ (\mathbf{Q}_{i^*}, \mathsf{DC}.\Pi_{i^*}) = (\mathbf{Q}'_i, \mathsf{DC}.\Pi'_{i^*})] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

The proof of the second part is analogues. Let Exp_1 be the experiment that is defined just like Exp except that we sample random messages $\mathbf{m}_1, \mathbf{m}_2 \leftarrow \mathbb{F}^\ell$. We can emulate Exp_1 given Γ as input. Therefore, by semantic security (Definition 6.5) it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}[\mathsf{AC} \ \wedge \ \mathbf{Q}_{i^*} = \mathbf{Q}'_{i^*} \ \rightarrow \ \mathsf{AC} \ \wedge \ A_{i^*} = A'_{i^*} \ \rightarrow \ (\mathbf{Q}_{i^*}, \mathsf{DC}.\Pi_{i^*}) = (\mathbf{Q}'_i, \mathsf{DC}.\Pi'_{i^*})] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Exp_2 be the experiment that is defined just like Exp_1 except that instead of sampling:

$$\begin{array}{l}(\mathbf{sk}, \Gamma, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1, \mathbf{m}_2\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma) \ ,\end{array}$$

we sample $(\mathsf{pk}^*, \mathsf{vk}^*)$ as follows:

- Sample $\mathbf{m}_1^* \leftarrow \mathbb{F}^{i^*}$, $\mathbf{m}_2^* \in \mathbb{F}^{\ell - i^*}$.
- Set $(\mathbf{sk}, \Gamma^*, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_1^*, \mathbf{m}_2^*\right]_{\mathsf{pp}}$.
- Set $(\mathsf{DC}.\mathsf{pk}^*, \mathsf{DC}.\mathsf{vk}^*) \leftarrow \mathsf{DC.S}(\mathsf{pp}, \Gamma^*, i^*)$
- Set $\Gamma \leftarrow \mathsf{RandExtend}(\Gamma^*, [i^* + 1, \ell] \cup [\ell + 2, 2\ell - i^* + 1])$.
- Set $\left(\mathsf{pk} = \left(\mathsf{pp}, \Gamma, \left(\mathsf{DC}.\mathsf{pk}_i, \mathsf{DC}.\tilde{\mathsf{pk}}_i\right)_{i\in[\ell]}\right), \mathsf{vk} = \left(\mathsf{DC}.\mathsf{vk}_i, \mathsf{DC}.\tilde{\mathsf{vk}}_i\right)_{i\in[\ell]}\right) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma)$.
- Let $(\mathsf{pk}^*, \mathsf{vk}^*)$ be the same as $(\mathsf{pk}, \mathsf{vk})$ except that we replace $(\mathsf{DC}.\mathsf{pk}_{i^*}, \mathsf{DC}.\mathsf{vk}_{i^*})$ by $(\mathsf{DC}.\mathsf{pk}^*, \mathsf{DC}.\mathsf{vk}^*)$.

In the rest of the experiment we use $(\mathsf{pk}^*, \mathsf{vk}^*)$ instead of $(\mathsf{pk}, \mathsf{vk})$. By the correctness of random extension (Definition 6.11) the distribution of Γ in Exp_1 and in Exp_2 is identical and By the correctness of restriction (Definition 6.10), $\Gamma^* = \mathsf{Restrict}\left(\Gamma, [i^* + 1, i^* + \ell]\right)$. Therefore, it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_2}[\mathsf{AC} \ \wedge \ \mathbf{Q}_{i^*} = \mathbf{Q}'_{i^*} \ \rightarrow \ \mathsf{AC} \ \wedge \ A_{i^*} = A'_{i^*} \ \rightarrow \ (\mathbf{Q}_{i^*}, \mathsf{DC}.\Pi_{i^*}) = (\mathbf{Q}'_i, \mathsf{DC}.\Pi'_{i^*})] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \quad (11)$$

If AC occurs then $\mathsf{DC.V}(\mathsf{DC.vk}_{i^*}, A_{i^*}, \mathbf{Q}_{i^*}, \mathsf{DC.}\Pi_{i^*})$ and $\mathsf{DC.V}(\mathsf{DC.vk}_{i^*}, A'_{i^*}, \mathbf{Q}'_{i^*}, \mathsf{DC.}\Pi'_{i^*})$ both accept. Also, we can emulate Exp_2 given $(\mathsf{DC.pk}^*, \mathsf{DC.vk}^*)$ and Γ^* as input. Therefore, by the unambiguity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \ \wedge \ A_{i^*} = A'_{i^*} \ \rightarrow \ (\mathbf{Q}_{i^*}, \mathsf{DC.}\Pi_{i^*}) = (\mathbf{Q}'_{i^*}, \mathsf{DC.}\Pi'_{i^*})\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa))$$

Also, by the injectivity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \ \wedge \ \mathbf{Q}_{i^*} = \mathbf{Q}'_{i^*} \ \rightarrow \ A_{i^*} = A'_{i^*}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Equation (11) follows, concluding the proof of the claim. □

7.4 Multi-equality Protocol

7.4.1 Definition.

The multi-equality protocol consists of algorithms $(\mathsf{MEQ.S}, \mathsf{MEQ.P}, \mathsf{MEQ.V})$ with the following syntax:

Setup: The PPT setup algorithm $\mathsf{MEQ.S}$ takes as input public parameters pp for the encryption scheme and a vector $\mathbf{c}$ of K single-key ciphertexts each encrypting a message of length ℓ. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic polynomial-time prover algorithm $\mathsf{MEQ.P}$ takes as input a prover key pk and a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. It outputs a vector $\mathbf{e}$ of K single-key evaluated ciphertexts and a proof Π.

Verifier: The deterministic polynomial-time verifier algorithm $\mathsf{MEQ.V}$ takes as input a verifier key vk, a vector $\mathbf{e}$ of K single-key evaluated ciphertexts and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 7.6. *A Λ-secure multi-equality protocol $(\mathsf{MEQ.S}, \mathsf{MEQ.P}, \mathsf{EQ.V})$ with degree bound $\bar{\delta}$, message length bound $\bar{\ell}$ and locality bound $\bar{K}$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \forall k \in [K] : \ \mathbf{e}_k = [F(c_k)]_{\mathsf{pp}} & (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\,[\mathbf{V}]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}) \\ & (\mathbf{e}, \Pi) \leftarrow \mathsf{MEQ.P}(\mathsf{pk}, F) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above $|\mathsf{vk}| = K \cdot \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$ and $|\Pi| = K \cdot \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$.*

Λ-Soundness. *For every $\mathrm{poly}(\Lambda)$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and indices $k_1, k_2 \in [K]$ such that $\mathbf{V}_{k_1} = \mathbf{V}_{k_2}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ v_{k_1} \neq v_{k_2} & Z = (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\,[\mathbf{V}]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}) \\ & (\mathbf{e}, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ & \forall k \in [K] \ : \ v_k \leftarrow \mathsf{Dec}(\mathbf{sk}_k, \mathbf{e}_k) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

Λ-Unambiguity. *For every $\mathrm{poly}(\Lambda)$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and index $k^* \in [K]$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{MEQ.V}\,(\mathsf{vk}, \mathbf{e}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{MEQ.V}\,(\mathsf{vk}, \mathbf{e}', \Pi') = 1 & (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\,[\mathbf{V}]_{\mathsf{pp}} \\ (\mathbf{e}, \Pi) \neq (\mathbf{e}', \Pi') & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}) \\ \mathbf{e}_{k^*} = \mathbf{e}'_{k^*} & (\mathbf{e}, \mathbf{e}', \Pi, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathbf{c}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

7.4.2 Construction.

We construct a multi-equality protocol $(\mathsf{MEQ.S}, \mathsf{MEQ.P}, \mathsf{MEQ.V})$ with degree bound $\bar{\delta}$, message length bound $\bar{\ell}$ and locality $\bar{K}$ as follows. The construction uses an equality protocol $(\mathsf{EQ.S}, \mathsf{EQ.P}, \mathsf{EQ.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

The setup algorithm $\mathsf{MEQ.S}$. The setup algorithm $\mathsf{MEQ.S}$ is given as input:

- Public parameters pp for the encryption scheme.
- A vector $\mathbf{c} = (c_k)_{k\in[K]}$ of single-key ciphertexts each encrypting a message of length ℓ.

It proceeds as follows:

- Sample $\tilde{\mathsf{sk}} \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and $\tilde{\mathbf{r}} \leftarrow \{0,1\}^{\kappa\times\ell}$.
- For every $k \in [K]$ set $\Gamma_k \leftarrow \mathsf{Extend}(c_k, (\tilde{\mathsf{sk}}, 0^\ell, \tilde{\mathbf{r}}))$.
- For every $k \in [K]$ set $(\mathsf{EQ.pk}_k, \mathsf{EQ.vk}_k) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma_k)$
- Output the prover key $\mathsf{pk} = (\mathsf{EQ.pk}_k)_{k\in[K]}$ and verifier key $\mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}$:

The prover algorithm $\mathsf{MEQ.P}$. The prover algorithm $\mathsf{MEQ.P}$ is given as input:

- A prover key $\mathsf{pk} = (\mathsf{EQ.pk}_k)_{k\in[K]}$.
- A polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$.

It proceeds as follows:

- For every $k \in [K]$ set $(e_k, \tilde{e}_k, \mathsf{EQ.}\Pi_k) \leftarrow \mathsf{EQ.P}(\mathsf{EQ.pk}_k, F)$.
- Output the vector of single-key evaluated ciphertexts and proof

$$\mathbf{e} = (e_k)_{k\in[K]} \ , \ \Pi = (\tilde{e}_k, \mathsf{EQ.}\Pi_k)_{k\in[K]} \ .$$

The verifier algorithm $\mathsf{MEQ.V}$. The verifier algorithm $\mathsf{MEQ.V}$ is given as input:

- A verifier key $\mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}$.
- A vector $\mathbf{e} = (e_k)_{k\in[K]}$ of evaluated ciphertexts.
- A proof $\Pi = (\tilde{e}_k, \mathsf{EQ.}\Pi_k)_{k\in[K]}$.

It proceeds as follows:

- For every $k \in [K]$ test that $\tilde{e}_1 = \tilde{e}_k$.
- For every $k \in [K]$ test that $\mathsf{EQ.V}(\mathsf{EQ.vk}_k, e_k, \tilde{e}_k, \mathsf{EQ.}\Pi_k)$ accepts.
- Output 1 if all tests pass. Otherwise output 0.

7.4.3 Analysis.

In this section we prove the following theorem:

Theorem 7.7. *For any $\Lambda(\kappa)$, $\bar{\delta}(\kappa)$,$\bar{\ell}(\kappa)$ and $\bar{K}(\kappa)$ such that $\bar{\ell}, \bar{K} \leq \Lambda^{O(1)}$ and $\Lambda \cdot \bar{\delta} = |\mathbb{F}|^{o(1)}$, assuming that:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$, the multi-equality protocol $(\mathsf{MEQ.S}, \mathsf{MEQ.P}, \mathsf{MEQ.V})$ given in Section 7.4.2 is a Λ-secure equality protocol (Definition 7.6) with degree bound $\bar{\delta}$, message length bound $\bar{\ell}$ and locality $\bar{K}$.

Completeness. Fix any $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and polynomial $F : \mathbb{F}^{\ell} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$. We need to show that:

$$\Pr\left[\begin{array}{l|l} \mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ \forall k \in [K] : \ e_k = [F(c_k)]_{\mathsf{pp}} & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \left(\mathbf{sk}, \mathbf{c} = (c_k)_{k\in[K]}, \mathbf{r}\right) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}\right) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}) \\ \left(\mathbf{e} = (e_k)_{k\in[K]}, \Pi = (\tilde{e}_k, \mathsf{EQ}.\Pi_k)_{k\in[K]}\right) \leftarrow \mathsf{MEQ.P}(\mathsf{pk}, F) \end{array} \end{array}\right] = 1 \ .$$

By the correctness of extension (Definition 6.9) for every $k \in [K]$ the ciphertext Γ_k computed by the setup algorithm is in the support of $\mathsf{MEnc}\left[\mathbf{m}_k, 0^{\ell}\right]_{\mathsf{pp}}$. Therefore, by the completeness of the equality protocol (Definition 7.3) for every $k \in [K]$:

- $e_k = [F(\mathsf{Restrict}(\Gamma_k, [\ell+1, 2\ell]))]_{\mathsf{pp}}$.
- $\tilde{e}_k = [F(\mathsf{Restrict}(\Gamma_k, [\ell]))]_{\mathsf{pp}}$.
- $\mathsf{EQ.V}(\mathsf{EQ.vk}_k, e_k, \tilde{e}_k, \mathsf{EQ}.\Pi_k)$ accepts.

By the correctness of restriction (Definition 6.10) for every $k \in [K]$:

$$c_k = \mathsf{Restrict}(\Gamma_k, [\ell+1, 2\ell]) \quad , \quad \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{\ell}, \tilde{\mathbf{r}}) = \mathsf{Restrict}(\Gamma_k, [\ell]) \ .$$

It follows that $e_k = [F(c_k)]_{\mathsf{pp}}$ and that $\tilde{e}_1 = \tilde{e}_k$.

Λ-Soundness. Assume towards contradiction that there exists a $\mathrm{poly}(\Lambda)$-size adversary Adv and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$ there exist a vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and indexes $k_1, k_2 \in [K]$ such that $\mathbf{m}_{k_1} = \mathbf{m}_{k_2}$ and:

$$\Pr\left[\begin{array}{l|l} \mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ v_{k_1} \neq v_{k_2} & \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ Z = \left(\mathbf{sk} = (\mathsf{sk}_k)_{k\in[K]}, \mathbf{c} = (c_k)_{k\in[K]}, \mathbf{r}\right) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}\right) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \mathbf{c}) \\ \left(\mathbf{e} = (e_k)_{k\in[K]}, \Pi = (\tilde{e}_k, \mathsf{EQ}.\Pi_k)_{k\in[K]}\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ \forall k \in [K] \ : \ v_k \leftarrow \mathsf{Dec}(\mathsf{sk}_k, e_k) \end{array} \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ . \quad (12)$$

Let Exp_1 be the experiment that is defined just like in Equation (12) except that we modify the step algorithm $\mathsf{EQ.S}$ as follows: instead of setting $\Gamma_k \leftarrow \mathsf{Extend}(c_k, (\tilde{\mathsf{sk}}, 0^{\ell}, \tilde{\mathbf{r}}))$, it first sets $\tilde{c} = \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{\ell}, \tilde{\mathbf{r}})$ and then sets $\Gamma_k \leftarrow \mathsf{Extend}((\mathsf{sk}_k, \mathbf{m}_k, \mathbf{r}_k), \tilde{c})$. By the correctness of extension (Definition 6.9):

$$\Pr_{\mathsf{Exp}_1}\left[\begin{array}{l} \mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ v_{k_1} \neq v_{k_2} \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ .$$

Let Exp_2 be the experiment that is defined just like Exp_1 except that instead of setting $\tilde{c} = \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{\ell}, \tilde{\mathbf{r}})$ we set $\tilde{c} = \mathsf{Enc}(\tilde{\mathsf{sk}}, \mathbf{m}, \tilde{\mathbf{r}})$ where $\mathbf{m} = \mathbf{m}_{k_1} = \mathbf{m}_{k_2}$. We can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by semantic security (Definition 6.5):

$$\Pr_{\mathsf{Exp}_2}\left[\begin{array}{l} \mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ v_{k_1} \neq v_{k_2} \end{array}\right] \geq \frac{\Omega(1)}{p(\Lambda(\kappa))} \ . \quad (13)$$

For $k^* \in \{k_1, k_2\}$ let $\tilde{v}_{k^*}$ be the encoded element $\mathsf{Dec}(\tilde{\mathsf{sk}}, \tilde{e}^*_k)$. When $\mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ accepts we have that $\mathsf{EQ.V}(\mathsf{EQ.vk}_{k^*}, e_{k^*}, \tilde{e}_{k^*}, \mathsf{EQ}.\Pi_{k^*})$ accepts. By the correctness of extension (Definition 6.9) the ciphertext Γ_k is in the support of $\mathsf{MEnc}\left[\mathbf{m}_{k^*}, \mathbf{m}\right]_{\mathsf{pp}}$. Since $\mathbf{m} = \mathbf{m}_{k^*}$ for $k^* \in \{k_1, k_2\}$, by the soundness of the equality protocol (Definition 7.3), $v_{k^*} = \tilde{v}_{k^*}$. When $\mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ accepts also $\tilde{e}_1 = \tilde{e}_{k_2} = \tilde{e}_{k_1}$ and, therefore, $\tilde{v}_{k_1} = \tilde{v}_{k_2}$. It follows that:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{EQ.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \quad \rightarrow \quad v_{k_1} = v_{k_2}\right] = 1 \ ,$$

contradicting Equation (13).

Λ-Unambiguity. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv, $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and an index $k^* \in [K]$. Let Exp denote the unambiguity experiment:

$$\begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\
(\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\
\left(\mathsf{pk}, \mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}\right) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}) \\
\left(\mathbf{e} = (e_k)_{k\in[K]}, \mathbf{e}' = (e'_k)_{k\in[K]}, \Pi = (\tilde{e}_k, \mathsf{EQ}.\Pi_k)_{k\in[K]}, \Pi' = (\tilde{e}'_k, \mathsf{EQ}.\Pi'_k)_{k\in[K]}\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathbf{c})
\end{array}$$

Let AC be the event that $\mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ and $\mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}', \Pi')$ both accept and $e_{k^*} = e'_{k^*}$. We need to show that:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ (\mathbf{e}, \Pi) \neq (\mathbf{e}', \Pi')\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ . \tag{14}$$

We rely on the following claim.

Claim 7.8. *For every $k \in [K]$:*

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ (e_k, \tilde{e}_k, \mathsf{EQ}.\Pi_k) \neq (e'_k, \tilde{e}'_k, \mathsf{EQ}.\Pi'_k) \ \rightarrow \ e_k \neq e'_k \ \wedge \ \tilde{e}_k \neq \tilde{e}'_k\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving the claim we use it to prove Equation (14). If AC occurs then $e_{k^*} = e'_{k^*}$. Therefore, by Claim 7.8:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \rightarrow \ \tilde{e}_{k^*} = \tilde{e}'_{k^*}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

When AC occurs, $\mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ and $\mathsf{MEQ.V}(\mathsf{vk}, \mathbf{e}', \Pi')$ both accept and, hence, for every $k \in [K]$, $\tilde{e}_1 = \tilde{e}_k$ and $\tilde{e}'_1 = \tilde{e}'_k$. Therefore:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ \tilde{e}_{k^*} = \tilde{e}'_{k^*} \ \rightarrow \ (\tilde{e}_k)_{k\in[K]} = (\tilde{e}'_k)_{k\in[K]}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 7.8:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ (\tilde{e}_k)_{k\in[K]} = (\tilde{e}'_k)_{k\in[K]} \ \rightarrow \ (e_k, \mathsf{EQ}.\Pi_k)_{k\in[K]} = (e'_k, \mathsf{EQ}.\Pi'_k)_{k\in[K]}\right] \geq 1 - K \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\bar{K} \leq \Lambda^{O(1)}$, Equation (14) follows. It remains to prove Claim 7.8.

Proof of Claim 7.8. Fix $k^* \in [K]$. Let Exp_1 be the experiment that is defined just like Exp except that instead of sampling:

$$\begin{array}{l}
(\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\
(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c})
\end{array}$$

we sample $(\mathsf{pk}, \mathsf{vk})$ as follows:

- Sample $(\mathbf{sk}, \Gamma_{k^*}, \mathbf{r}) \leftarrow \mathsf{MEnc}\left[\mathbf{m}_{k^*}, 0^\ell\right]_{\mathsf{pp}}$.
- Set $(\mathsf{EQ.pk}_{k^*}, \mathsf{EQ.vk}_{k^*}) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma_{k^*})$
- Set $\tilde{c} \leftarrow \mathsf{Restrict}(\Gamma_{k^*}, [\ell])$.
- For every $k \in [K]\backslash\{k^*\}$, sample $\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\mathbf{r}_k \leftarrow \{0,1\}^{\kappa \times \ell}$ and set $\Gamma_k \leftarrow \mathsf{Extend}((\mathsf{sk}_k, \mathbf{m}_k, \mathbf{r}_k), \tilde{c})$.
- For every $k \in [K]$ set $(\mathsf{EQ.pk}_k, \mathsf{EQ.vk}_k) \leftarrow \mathsf{EQ.S}(\mathsf{pp}, \Gamma_k)$
- Let $\mathsf{pk} = (\mathsf{EQ.pk}_k)_{k\in[K]}$ and $\mathsf{vk} = (\mathsf{EQ.vk}_k)_{k\in[K]}$.

By the correctness of random extension (Definition 6.11) and the correctness of restriction (Definition 6.10) the distribution of $\Gamma_1 \ldots, \Gamma_K$ in Exp and in Exp_1 is identical. Therefore, it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ (e_k, \tilde{e}_k, \mathsf{EQ}.\Pi_k) \neq (e'_k, \tilde{e}'_k, \mathsf{EQ}.\Pi'_k) \ \rightarrow \ e_k \neq e'_k \ \wedge \ \tilde{e}_k \neq \tilde{e}'_k\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{15}$$

If AC occurs then $\mathsf{EQ.V}(\mathsf{EQ.vk}_{k^*}, e_{k^*}, \tilde{e}_{k^*}, \mathsf{EQ}.\Pi_{k^*})$ and $\mathsf{EQ.V}(\mathsf{EQ.vk}_{k^*}, e'_{k^*}, \tilde{e}'_{k^*}, \mathsf{EQ}.\Pi'_{k^*})$ both accept. Also, we can emulate Exp_1 given $(\mathsf{EQ.pk}_{k^*}, \mathsf{EQ.vk}_{k^*})$ and Γ_{k^*} as input. Therefore, Equation (15) follows from the unambiguity of the equality protocol (Definition 7.3), concluding the proof of the claim. □

7.5 Multilinearity Protocol

This section describes the multilinearity protocol. The protocol's CRS contains a single-key ciphertext c encrypting a message $\mathbf{m}$ of length ℓ under a secret key sk. The honest prover is given a multilinear polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ and it homomorphically evaluates F on c. It sends the resulting ciphertext e to the verifier together with a proof convincing the verifier that F is indeed multilinear. More formally, in the soundness experiment, a cheating prover produces two evaluated ciphertexts e_1 and e_2, each with a proof of multilinearity. We say that the prover splits the ciphertexts e_1 and e_2 if both proofs are accepting and e_1 and e_2 encrypt different values. The soundness requirement is that if for some message $\mathbf{m} \in \mathbb{F}^\ell$ encrypted in c the prover splits the ciphertexts with probability p then there exists a binary message $\mathbf{m}^* \in \{0,1\}^\ell$ such that when c encrypts $\mathbf{m}^*$, the same prover splits the ciphertexts with probability $\geq p \cdot 2^{-\ell}$.

We explain why this soundness requirement intuitively captures the multilinearity of the evaluated polynomial. Consider a cheating prover that computes e_1 by homomorphically evaluating some polynomial F_1 of degree > 1 on c. If the prover is able to produce an accepting proof for e_1 then we can use it to violate the soundness requirement. Let F_2 be the multilinear polynomial that agrees with F_1 on inputs in $\{0,1\}^\ell$. We compute e_2 by homomorphically evaluating F_2 on c and compute the proof of multilinearity honestly. Since F_1 and F_2 must disagree on some $\mathbf{m} \in \mathbb{F}^\ell$, if c encrypts $\mathbf{m}$, then we split e_1 and e_2. If, however, c encrypts a Boolean message $\mathbf{m} \in \{0,1\}^\ell$ then we do not split the ciphertexts.

We note that when switching c from an encryption of a message in $\mathbb{F}^\ell$ to an encryption of a message in $\{0,1\}^\ell$ the probability of splitting may drop by a factor of $2^{-\ell}$ in general. Consider a prover that samples a random message $\mathbf{m}^* \leftarrow \{0,1\}^\ell$ and homomorphically evaluates two random multilinear polynomials that agree on every Boolean input except $\mathbf{m}^*$. If c encrypts any message outside $\{0,1\}^\ell$ then the prover splits the ciphertexts with probability $1 - 1/|\mathbb{F}|$ while if c encrypts any Boolean message, the prover splits the ciphertexts with probability $2^{-\ell}$.

The multilinearity protocol is constructed as follows. The setup algorithm first extends c into a multi-key ciphertext Γ encrypting $\mathbf{m}$ under sk and a random message $\mathbf{m}' \in \mathbb{F}^\ell$ under an independent key sk'. Given a multilinear polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$, the prover uses the decomposition protocol to compute the decomposition of F along every coordinate $i \in [\ell]$ evaluated once on $\mathbf{m}_{-i}$ and once on $\mathbf{m}'_{-i}$. The verifier checks that the decompositions interpreted as encryptions of univariate polynomials are indeed linear . To convince the verifier that the decomposition evaluated on $\mathbf{m}_{-i}$ is indeed independent of $\mathbf{m}_i$, the prover uses the equality protocol to prove that if $\mathbf{m}_{-i}$ and once on $\mathbf{m}'_{-i}$ the two decompositions along the i'th coordinate encrypt the same values.

The unambiguity of this multilinearity protocol follows directly from the unambiguity of the decomposition protocol and the equality protocol.

7.5.1 Definition.

The multilinearity protocol consists of algorithms $(\mathsf{ML.S}, \mathsf{ML.P}, \mathsf{ML.V})$ with the following syntax:

Setup: The PPT setup algorithm $\mathsf{ML.S}$ takes as input public parameters pp for the encryption scheme and a vector $\mathbf{c}$ of K single-key ciphertexts each encrypting a message of length ℓ. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic polynomial-time prover algorithm $\mathsf{ML.P}$ takes as input a prover key pk and a multilinear polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$. It outputs a vector $\mathbf{e}$ of K single-key evaluated ciphertexts and a proof Π.

Verifier: The deterministic polynomial-time verifier algorithm $\mathsf{ML.V}$ takes as input a verifier key vk, a vector $\mathbf{e}$ of K evaluated ciphertexts and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 7.9. *A Λ-secure multilinearity protocol $(\mathsf{ML.S}, \mathsf{ML.P}, \mathsf{ML.V})$ with message length bound $\bar{\ell}$ and locality $\bar{K}$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and multilinear polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \forall k \in [K] : \ \mathbf{e}_k = [F(c_k)]_{\mathsf{pp}} & (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\,[\mathbf{V}]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\ & (\mathbf{e}, \Pi) \leftarrow \mathsf{ML.P}(\mathsf{pk}, F) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above* $|\mathsf{vk}| = K \cdot \mathrm{poly}(\kappa, \bar{\ell})$ *and* $|\Pi| = K \cdot \mathrm{poly}(\kappa, \bar{\ell})$.

Λ-Soundness. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*, vector of messages* $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ *such that* $\ell \leq \bar{\ell}, K \leq \bar{K}$*, index* $i \in [\ell]$ *and set* $\mathbf{S} \subseteq [K]$ *such that* $(\mathbf{m}_k)_{-i} = (\mathbf{m}_{k'})_{-i}$ *for all* $k, k' \in \mathbf{S}$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \forall \beta_0, \beta_1 \in \mathbb{F} : & Z = (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ \quad \exists k^* \in \mathbf{S} : \ \langle \beta_1 \cdot (\mathbf{m}_{k^*})_i + \beta_0 \rangle_{\mathsf{pp}} \neq v_{k^*} & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\ & (\mathbf{e}, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ & \forall k \in [K] \ : \ v_k \leftarrow \mathsf{Dec}(\mathbf{sk}_k, \mathbf{e}_k) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

Λ-Unambiguity. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*, vector of messages* $\mathbf{V} \in \mathbb{F}^{\ell \times K}$ *such that* $\ell \leq \bar{\ell}, K \leq \bar{K}$ *and index* $k^* \in [K]$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{ML.V}\left(\mathsf{vk}, \mathbf{e}, \Pi\right) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{ML.V}\left(\mathsf{vk}, \mathbf{e}', \Pi'\right) = 1 & (\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ (\mathbf{e}, \Pi) \neq (\mathbf{e}', \Pi') & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\ \mathbf{e}_{k^*} = \mathbf{e}'_{k^*} & (\mathbf{e}, \mathbf{e}', \Pi, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathbf{c}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

7.5.2 Construction.

We construct a multilinearity protocol $(\mathsf{ML.S}, \mathsf{ML.P}, \mathsf{ML.V})$ with message length bound $\bar{\ell}$ and locality $\bar{K}$ as follows. The construction uses a decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with degree bound $\bar{\delta} = 1$ and message length bound $\bar{\ell}$ and a multi-equality protocol $(\mathsf{MEQ.S}, \mathsf{MEQ.P}, \mathsf{MEQ.V})$ with degree bound $\bar{\delta} = 1$ message length bound $\bar{\ell}$ and locality $\bar{K}$.

The setup algorithm $\mathsf{ML.S}$. The setup algorithm $\mathsf{ML.S}$ is given as input:

- Public parameters pp for the encryption scheme.
- A vector $\mathbf{c} = (c_k)_{k \in [K]}$ of single-key ciphertexts each encrypting a message of length ℓ.

It proceeds as follows:

- For every $k \in [K]$ and $i \in [\ell]$ set $(\mathsf{DC.pk}_{k,i}, \mathsf{DC.vk}_{k_i}) \leftarrow \mathsf{DC.S}(\mathsf{pp}, c_k, i)$.
- For every $k \in [K]$ and $i \in [\ell]$ set $c_{k,i} \leftarrow \mathsf{Restrict}(c_k, \{i\})$.
- For every $i \in [\ell]$ set $(\mathsf{MEQ.pk}_i, \mathsf{MEQ.vk}_i) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, (c_{k,i})_{k \in [K]})$.
- Output the prover key and verifier key:

$$\mathsf{pk} = \left(\mathsf{pp}, \left(\mathsf{DC.pk}_{k,i}\right)_{k \in [K], i \in [\ell]}, \left(\mathsf{MEQ.pk}_i\right)_{i \in [\ell]}\right) \quad , \quad \mathsf{vk} = \left(\left(\mathsf{DC.vk}_{k,i}\right)_{k \in [K], i \in [\ell]}, \left(\mathsf{MEQ.vk}_i\right)_{i \in [\ell]}\right) \ .$$

The prover algorithm $\mathsf{ML.P}$. The prover algorithm $\mathsf{ML.P}$ is given as input:

- A prover key:
$$\mathsf{pk} = \left(\mathsf{pp}, \left(\mathsf{DC.pk}_{k,i}\right)_{k \in [K], i \in [\ell]}, \left(\mathsf{MEQ.pk}_i\right)_{i \in [\ell]}\right) \ .$$
- A multilinear polynomial $F : \mathbb{F}^{\ell} \to \mathbb{F}$.

It proceeds as follows:

- For every $k \in [K]$ set $e_k \leftarrow [F(c_k)]_{\mathsf{pp}}$.
- For every $k \in [K]$ and $i \in [\ell]$ set $(A_{k,i}, \mathbf{Q}_{k,i}, \mathsf{DC.}\Pi_{k,i}) \leftarrow \mathsf{DC.P}(\mathsf{DC.pk}_{k,i}, F)$.

- For every $i \in [\ell]$ and $j \in [0, \bar{\delta}]$ set $(\mathbf{B}_{i,j}, \mathsf{MEQ.\Pi}_{i,j}) \leftarrow \mathsf{MEQ.P}(\mathsf{MEQ.pk}_i, F|_{i,j})$.
- Output the vector of evaluated ciphertexts and proof:

$$\mathbf{e} = (e_k)_{k\in[K]} \ , \ \Pi = \left((A_{k,i}, \mathbf{Q}_{k,i}, \mathsf{DC.\Pi}_{k,i})_{k\in[K],i\in[\ell]} , (\mathbf{B}_{i,j}, \mathsf{MEQ.\Pi}_{i,j})_{i\in[\ell],j\in[0,\bar{\delta}]} \right) \ .$$

The verifier algorithm $\mathsf{ML.V}$. The verifier algorithm $\mathsf{ML.V}$ is given as input:

- A verifier key:

$$\mathsf{vk} = \left((\mathsf{DC.vk}_{k,i})_{k\in[K],i\in[\ell]} , (\mathsf{MEQ.vk}_i)_{i\in[\ell]} \right) \ .$$

- A vector $\mathbf{e} = (e_k)_{k\in[K]}$ of evaluated ciphertexts.
- A proof:

$$\Pi = \left(\left(A_{k,i}, \mathbf{Q}_{k,i} = (Q_{k,i,j})_{j\in[0,\bar{\delta}]} , \mathsf{DC.\Pi}_{k,i} \right)_{k\in[K],i\in[\ell]} , \left(\mathbf{B}_{i,j} = (B_{k,i,j})_{k\in[K]} , \mathsf{MEQ.\Pi}_{i,j} \right)_{i\in[\ell],j\in[0,\bar{\delta}]} \right) \ .$$

It proceeds as follows. For every $i \in [\ell]$:

- For every $k \in [K]$ test that $e_k = A_{k,i}$.
- For every $k \in [K]$ test that $\mathsf{DC.V}(\mathsf{DC.vk}_{k,i}, A_{k,i}, \mathbf{Q}_{k,i}, \mathsf{DC.\Pi}_{k,i})$ accepts.
- For every $k \in [K]$ and $j \in [0, \bar{\delta}]$ test that $Q_{k,i,j} = B_{k,i,j}$.
- For every $j \in [0, \bar{\delta}]$ test that $\mathsf{MEQ.V}(\mathsf{MEQ.vk}_i, \mathbf{B}_{i,j}, \mathsf{MEQ.\Pi}_{i,j})$ accepts.
- Output 1 if all tests pass. Otherwise output 0.

7.5.3 Analysis.

In this section we prove the following theorem:

Theorem 7.10. *For any $\Lambda(\kappa), \bar{\ell}(\kappa)$ and $\bar{K}(\kappa)$ such that $\bar{\ell} = O(\log \Lambda)$, $\bar{\ell}, \bar{K} \leq \Lambda^{O(1)}$ and $\Lambda = |\mathbb{F}|^{o(1)}$, assuming that:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) with degree bound $\bar{\delta} \geq 1$ and message length bound $\bar{\ell}$, the multilinearity protocol $(\mathsf{ML.S}, \mathsf{ML.P}, \mathsf{ML.V})$ given in Section 7.5.2 is a Λ-secure multilinearity protocol (Definition 7.9) with message length bound $\bar{\ell}$ and locality $\bar{K}$.

Completeness. Fix any $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and multilinear polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$. We need to show that:

$$\Pr \left[\begin{array}{l|l} \mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ \forall k \in [K] : \ e_k = [F(c_k)]_{\mathsf{pp}} \end{array} \right. \left. \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \left(\mathbf{sk}, \mathbf{c} = (c_k)_{k\in[K]} , \mathbf{r} \right) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\ (\mathbf{e} = (e_k)_{k\in[K]} , \Pi) \leftarrow \mathsf{ML.P}(\mathsf{pk}, F) \end{array} \right] = 1 \ ,$$

where:

$$\mathsf{vk} = \left((\mathsf{DC.vk}_{k,i})_{k\in[K],i\in[\ell]} , (\mathsf{MEQ.vk}_i)_{i\in[\ell]} \right) \ ,$$

$$\Pi = \left(\left(A_{k,i}, \mathbf{Q}_{k,i} = (Q_{k,i,j})_{j\in[0,\bar{\delta}]} , \mathsf{DC.\Pi}_{k,i} \right)_{k\in[K],i\in[\ell]} , \left(\mathbf{B}_{i,j} = (B_{k,i,j})_{k\in[K]} , \mathsf{MEQ.\Pi}_{i,j} \right)_{i\in[\ell],j\in[0,\bar{\delta}]} \right) \ .$$

By construction for every $k \in [K]$ the prover sets $e_k \leftarrow [F(c_k)]_{\mathsf{pp}}$. Next we show that for every $i \in [\ell]$ each of the verifier's tests passes. By the completeness of the decomposition protocol (Definition 7.1) for every $k \in [K]$:

- $A_{k,i} = [F(c_k)]_{\mathsf{pp}} = e_k$.
- $\mathsf{DC.V}(\mathsf{DC.vk}_{k,i}, A_{k,i}, \mathbf{Q}_{k,i}, \mathsf{DC.}\Pi_{k,i})$ accepts.
- For every $j \in [0, \bar{\delta}]$, $Q_{k,i,j} = \left[F|_{i,j}(\mathsf{Restrict}(c_k, i)) \right]_{\mathsf{pp}}$.

By the correctness of restriction (Definition 6.10) for every $k \in [K]$ the ciphertext $c_{k,i}$ computed by the setup algorithm is in the support of $\mathsf{Enc}\left[(\mathbf{m}_k)_{-i}\right]_{\mathsf{pp}}$. Therefore, by the completeness of the multi-equality protocol (Definition 7.6) for every $j \in [0, \bar{\delta}]$:

- $\mathsf{MEQ.V}(\mathsf{MEQ.vk}_i, \mathbf{B}_{i,j}, \mathsf{MEQ.}\Pi_{i,j})$ accepts.
- For every $k \in [K]$, $B_{k,i,j} = \left[F|_{i,j}(c_{k,i}) \right]_{\mathsf{pp}}$.

For every $k \in [K]$ and $j \in [0, \bar{\delta}]$, since $c_{k,i} = \mathsf{Restrict}(c_k, i)$ we have that $Q_{k,i,j} = B_{k,i,j}$.

Soundness. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv, $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$, index $i^* \in [\ell]$ and set $\mathbf{S} \subseteq [K]$ such that $(\mathbf{m}_k)_{-i^*} = (\mathbf{m}_{k'})_{-i^*}$ for all $k, k' \in \mathbf{S}$. Let Exp denote the soundness experiment:

$$
\begin{array}{l}
\mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\
Z = \left(\mathbf{sk} = (\mathsf{sk}_k)_{k \in [K]}, \mathbf{c} = (c_k)_{k \in [K]}, \mathbf{r} \right) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\
(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\
(\mathbf{e} = (e_k)_{k \in [K]}, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\
\forall k \in [K] \ : \ v_k \leftarrow \mathsf{Dec}(\mathsf{sk}_k, e_k)
\end{array}
$$

where:

$$
\mathsf{vk} = \left((\mathsf{DC.vk}_{k,i})_{k \in [K], i \in [\ell]}, (\mathsf{MEQ.vk}_i)_{i \in [\ell]} \right) \ ,
$$

$$
\Pi = \left(\left(A_{k,i}, \mathbf{Q}_{k,i} = (Q_{k,i,j})_{j \in [0,\bar{\delta}]}, \mathsf{DC.}\Pi_{k,i} \right)_{k \in [K], i \in [\ell]}, \left(\mathbf{B}_{i,j} = (B_{k,i,j})_{k \in [K]}, \mathsf{MEQ.}\Pi_{i,j} \right)_{i \in [\ell], j \in [0,\bar{\delta}]} \right) \ .
$$

For $k \in [K]$ and $j \in [0, \bar{\delta}]$ let:

$$
\alpha_k = \mathsf{Dec}(\mathsf{sk}_k, A_{k,i^*}) \quad , \quad \beta_{k,j} = \mathsf{Dec}(\mathsf{sk}_k, Q_{k,i^*,j}) \quad , \quad \zeta_{k,j} = \mathsf{Dec}(\mathsf{sk}_k, B_{k,i^*,j}) \ .
$$

We need to show that:

$$
\Pr_{\mathsf{Exp}} \left[\begin{array}{l} \mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \\ \forall \beta_0, \beta_1 \in \mathbb{F} \ \exists k^* \in \mathbf{S} : \ \langle \beta_1 \cdot (\mathbf{m}_{k^*})_{i^*} + \beta_0 \rangle_{\mathsf{pp}} \neq v_{k^*} \end{array} \right] \leq \mathrm{negl}(\Lambda(\kappa)) \ . \tag{16}
$$

We rely on the following claim.

Claim 7.11. *For every $k_1, k_2 \in \mathbf{S}$ and $j \in [0, \bar{\delta}]$:*

$$
\Pr_{\mathsf{Exp}} \left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \ \rightarrow \ \zeta_{k_1,j} = \zeta_{k_2,j} \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .
$$

Before proving the claim we use it to prove Equation (10). If $\mathbf{S}$ is empty, soundness holds trivially. Otherwise, fix some $k' \in \mathbf{S}$ and for $j \in [0, \bar{\delta}]$, let $\beta_j = \beta_{k',j}$. If $\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ accepts then:

- By Claim 7.11 for every $k^* \in \mathbf{S}$ and $j \in [0, \bar{\delta}]$:

$$
\Pr_{\mathsf{Exp}} \left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \ \rightarrow \ \zeta_{k',j} = \zeta_{k^*,j} \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .
$$

Since $\bar{\delta} = 1$ and $|\mathbf{S}| \leq \bar{K} \leq \Lambda^{O(1)}$:

$$
\Pr_{\mathsf{Exp}} \left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \ \rightarrow \ \forall k^* \in \mathbf{S}, j \in [0, \bar{\delta}] : \ \zeta_{k',j} = \zeta_{k^*,j} \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .
$$

- For every $k \in [K]$ and $j \in [0, \bar{\delta}]$, $\beta_{k,j} = \zeta_{k,j}$, and therefore:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \;\rightarrow\; \forall k^* \in \mathbf{S}, j \in [0, \bar{\delta}] :\; \beta_j = \beta_{k',j} = \beta_{k^*,j}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \;\; .$$

- For every $k^* \in \mathbf{S}$, $\mathsf{DC.V}(\mathsf{DC.vk}_{k^*,i^*}, A_{k^*,i^*}, \mathbf{Q}_{k^*,i^*}, \mathsf{DC}.\Pi_{k^*,i^*})$ accepts. Therefore, by the soundness of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}}\left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \;\rightarrow\; \forall k^* \in \mathbf{S} :\; \langle \beta_1 \cdot (\mathbf{m}_{k^*})_{i^*} + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k^*}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \;\; .$$

- For every $k^* \in \mathbf{S}$, $v_{k^*} = \alpha_{k^*}$, and therefore:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \;\rightarrow\; \forall k^* \in \mathbf{S} :\; \langle \beta_1 \cdot (\mathbf{m}_{k^*})_{i^*} + \beta_0 \rangle_{\mathsf{pp}} = v_{k^*}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \;\; .$$

Equation (16) follows. It remains to prove Claim 7.11.

Proof of Claim 7.11. Fix any $k_1, k_2 \in \mathbf{S}$ and $j \in [0, \bar{\delta}]$. Let Exp_1 be the experiment that is defined just like Exp except that instead of sampling:

$$(\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}}$$
$$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c})$$

we sample $(\mathsf{pk}^*, \mathsf{vk}^*)$ as follows:

- Let $\mathbf{V}^* = \left((\mathbf{m}_k)_{-i^*}\right)_{k \in K}$.
- Sample $Z = (\mathbf{sk} = (\mathsf{sk}_k)_{k \in [K]}, \mathbf{c}^*, \mathbf{r}^* = (\mathbf{r}^*_k)_{k \in [K]}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}^*\right]_{\mathsf{pp}}$.
- Set $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}^*)$
- For $k \in [K]$ sample $\mathbf{r}_k \leftarrow \{0,1\}^{\kappa \times \ell}$ such that $(\mathbf{r}_k)_{-i^*} = \mathbf{r}^*_{i^*}$ and set $c_k = \mathsf{Enc}(\mathsf{sk}_k, \mathbf{m}_k, \mathbf{r}_k)$.
- Set $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c} = (c_k)_{k \in [K]})$ where:

$$\mathsf{pk} = \left(\mathsf{pp}, \left(\mathsf{DC.pk}_{k,i}\right)_{k \in [K], i \in [\ell]}, \left(\mathsf{MEQ.pk}_i\right)_{i \in [\ell]}\right) \quad , \quad \mathsf{vk} = \left(\left(\mathsf{DC.vk}_{k,i}\right)_{k \in [K], i \in [\ell]}, \left(\mathsf{MEQ.vk}_i\right)_{i \in [\ell]}\right) \;\; .$$

- Let $(\mathsf{pk}^*, \mathsf{vk}^*)$ be the same as $(\mathsf{pk}, \mathsf{vk})$ except that we replace $(\mathsf{MEQ.pk}_{i^*}, \mathsf{MEQ.vk}_{i^*})$ by $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*)$.

Observe that the distribution of $\mathbf{c}$ in Exp and in Exp_1 is identical. Also, by the correctness of restriction (Definition 6.10), $\mathbf{c}^* = (c_{k,i^*})_{k \in [K]}$. Therefore, it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi) = 1 \;\rightarrow\; \zeta_{k_1,j} = \zeta_{k_2,j}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \;\; . \tag{17}$$

If $\mathsf{ML.V}(\mathsf{vk}, \mathbf{e}, \Pi)$ accept then $\mathsf{MEQ.V}(\mathsf{MEQ.vk}_{i^*}, \mathbf{B}_{i^*,j}, \mathsf{MEQ}.\Pi_{i^*,j})$ accepts as well. Since $k_1, k_2 \in \mathbf{S}$, we have that $(\mathbf{m}_{k_1})_{-i^*} = (\mathbf{m}_{k_2})_{-i^*}$. Also, we can emulate Exp_1 given $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*)$ and Z as input. Therefore, Equation (17) follows from the soundness of the multi-equality protocol (Definition 7.6), concluding the proof of the claim. □

Λ-Unambiguity. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv, $\kappa \in \mathbb{N}$, vector of messages $\mathbf{V} = (\mathbf{m}_1, \ldots, \mathbf{m}_K) \in \mathbb{F}^{\ell \times K}$ such that $\ell \leq \bar{\ell}, K \leq \bar{K}$ and an index $k^* \in [K]$. Let Exp denote the unambiguity experiment:

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \left(\mathbf{sk}, \mathbf{c} = (c_k)_{k\in[K]}, \mathbf{r}\right) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}} \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c}) \\ \left(\mathbf{e} = (e_k)_{k\in[K]}, \mathbf{e}' = (e'_k)_{k\in[K]}, \Pi, \Pi'\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathbf{c}) \end{array}$$

where:

$$\mathsf{pk} = \left(\mathsf{pp}, \left(\mathsf{DC.pk}_{k,i}\right)_{k\in[K],i\in[\ell]}, \left(\mathsf{MEQ.pk}_i\right)_{i\in[\ell]}\right) \ ,$$

$$\mathsf{vk} = \left(\left(\mathsf{DC.vk}_{k,i}\right)_{k\in[K],i\in[\ell]}, \left(\mathsf{MEQ.vk}_i\right)_{i\in[\ell]}\right) \ ,$$

$$\Pi = \left(\left(A_{k,i}, \mathbf{Q}_{k,i} = \left(Q_{k,i,j}\right)_{j\in[0,\bar{\delta}]}, \mathsf{DC.}\Pi_{k,i}\right)_{k\in[K],i\in[\ell]}, \left(\mathbf{B}_{i,j} = \left(B_{k,i,j}\right)_{k\in[K]}, \mathsf{MEQ.}\Pi_{i,j}\right)_{i\in[\ell],j\in[0,\bar{\delta}]}\right) \ .$$

$$\Pi' = \left(\left(A'_{k,i}, \mathbf{Q}'_{k,i} = \left(Q'_{k,i,j}\right)_{j\in[0,\bar{\delta}]}, \mathsf{DC.}\Pi'_{k,i}\right)_{k\in[K],i\in[\ell]}, \left(\mathbf{B}'_{i,j} = \left(B'_{k,i,j}\right)_{k\in[K]}, \mathsf{MEQ.}\Pi'_{i,j}\right)_{i\in[\ell],j\in[0,\bar{\delta}]}\right) \ .$$

Let AC be the event that $\mathsf{ML.V}\left(\mathsf{vk}, \mathbf{e}, \Pi\right)$ and $\mathsf{ML.V}\left(\mathsf{vk}, \mathbf{e}', \Pi'\right)$ both accept and $e_{k^*} = e'_{k^*}$. We need to show that:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ (\mathbf{e}, \Pi) \neq (\mathbf{e}', \Pi')\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ . \tag{18}$$

We rely on the following claim.

Claim 7.12. *For every $i \in [\ell]$ and $j \in [0, \bar{\delta}]$:*

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ B_{k^*,i,j} = B'_{k^*,i,j} \ \rightarrow \ (\mathbf{B}_{i,j}, \mathsf{MEQ.}\Pi_{i,j}) = \left(\mathbf{B}'_{i,j}, \mathsf{MEQ.}\Pi'_{i,j}\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving the claim we use it to prove Equation (18). If AC occurs then for every $i \in [\ell]$:

- $A_{k^*,i} = e_{k^*} = e'_{k^*} = A'_{k^*,i}$.
- $\mathsf{DC.V}(\mathsf{DC.vk}_{k^*,i}, A_{k^*,i}, \mathbf{Q}_{k^*,i}, \mathsf{DC.}\Pi_{k^*,i})$ and $\mathsf{DC.V}(\mathsf{DC.vk}_{k^*,i}, A'_{k^*,i}, \mathbf{Q}'_{k^*,i}, \mathsf{DC.}\Pi'_{k^*,i})$ both accept. Also, we can emulate Exp given $(\mathsf{DC.pk}_{k^*,i}, \mathsf{DC.vk}_{k^*,i})$ and c_{k^*} as input. Therefore, by the unambiguity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ A_{k^*,i} = A'_{k^*,i} \ \rightarrow \ (\mathbf{Q}_{k^*,i}, \mathsf{DC.}\Pi_{k^*,i}) = \left(\mathbf{Q}'_{k^*,i}, \mathsf{DC.}\Pi'_{k^*,i}\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $j \in [0, \bar{\delta}]$, $Q_{k^*,i,j} = B_{k^*,i,j}$ and $Q'_{k^*,i,j} = B'_{k^*,i,j}$. Therefore:

$$\mathbf{Q}_{k^*,i} = \mathbf{Q}'_{k^*,i} \quad \Rightarrow \quad \forall j \in [0, \bar{\delta}] : \ B_{k^*,i,j} = B'_{k^*,i,j} \ .$$

- By Claim 7.12, for every $j \in [0, \bar{\delta}]$:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ B_{k^*,i,j} = B'_{k^*,i,j} \ \rightarrow \ (\mathbf{B}_{i,j}, \mathsf{MEQ.}\Pi_{i,j}) = \left(\mathbf{B}'_{i,j}, \mathsf{MEQ.}\Pi'_{i,j}\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $k \in [K]$ and $j \in [0, \bar{\delta}]$, $Q_{k,i,j} = B_{k,i,j}$ and $Q'_{k,i,j} = B'_{k,i,j}$. Therefore:

$$\forall j \in [0, \bar{\delta}] : \ \mathbf{B}_{i,j} = \mathbf{B}'_{i,j} \quad \rightarrow \quad \forall k \in [K] : \ \mathbf{Q}_{k,i} = \mathbf{Q}'_{k,i} \ .$$

- For every $k \in [K]$, $\mathsf{DC.V}(\mathsf{DC.vk}_{k,i}, A_{k,i}, \mathbf{Q}_{k,i}, \mathsf{DC.}\Pi_{k,i})$ and $\mathsf{DC.V}(\mathsf{DC.vk}_{k,i}, A'_{k,i}, \mathbf{Q}'_{k,i}, \mathsf{DC.}\Pi'_{k,i})$ both accept. Also, we can emulate Exp given $(\mathsf{DC.pk}_{k,i}, \mathsf{DC.vk}_{k,i})$ and c_k as input. Therefore, by the injectivity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ \mathbf{Q}_{k,i} = \mathbf{Q}'_{k,i} \ \rightarrow \ A_{k,i} = A'_{k,i}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

  Also, by the unambiguity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ A_{k,i} = A'_{k,i} \ \rightarrow \ \mathsf{DC.}\Pi_{k,i} = \mathsf{DC.}\Pi'_{k,i}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\bar{\delta} = 1$ and $\ell, K \leq \Lambda^{O(1)}$, Equation (18) follows. It remains to prove Claim 7.12.

Proof of Claim 7.12. Fix any $i \in [\ell]$ and $j \in [0, \bar{\delta}]$. Let Exp_1 be the experiment that is defined just like Exp except that we sample random messages $\mathbf{m}_1, \ldots, \mathbf{m}_K \leftarrow \mathbb{F}^\ell$. Also, we can emulate Exp_1 given $\mathbf{c}$ as input. Therefore, by semantic security (Definition 6.5) and since $K \leq \bar{M} \leq \mathrm{poly}(\Lambda)$ it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ B_{k^*,i,j} = B'_{k^*,i,j} \ \rightarrow \ (\mathbf{B}_{i,j}, \mathsf{MEQ.}\Pi_{i,j}) = \left(\mathbf{B'}_{i,j}, \mathsf{MEQ.}\Pi'_{i,j}\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Exp_2 be the experiment that is defined just like Exp_1 except that instead of sampling:

$$(\mathbf{sk}, \mathbf{c}, \mathbf{r}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}\right]_{\mathsf{pp}}$$
$$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c})$$

we sample $(\mathsf{pk}^*, \mathsf{vk}^*)$ as follows:

- Let $\mathbf{V}^* = \left((\mathbf{m}_k)_{-i}\right)_{k \in K}$.
- Sample $(\mathbf{sk} = (\mathsf{sk}_k)_{k\in[K]}, \mathbf{c}^* = (c^*_k)_{k\in[K]}, \mathbf{r}^* = (\mathbf{r}^*_k)_{k\in[K]}) \leftarrow \mathsf{VEnc}\left[\mathbf{V}^*\right]_{\mathsf{pp}}$.
- Set $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*) \leftarrow \mathsf{MEQ.S}(\mathsf{pp}, \mathbf{c}^*)$
- For $k \in [K]$ set $c_k \leftarrow \mathsf{RandExtend}(c^*_k, [i])$.
- Set $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{ML.S}(\mathsf{pp}, \mathbf{c} = (c_k)_{k\in[K]})$ where:

$$\mathsf{pk} = \left(\mathsf{pp}, \left(\mathsf{DC.pk}_{k,i}\right)_{k\in[K], i\in[\ell]}, \left(\mathsf{MEQ.pk}_i\right)_{i\in[\ell]}\right) \quad , \quad \mathsf{vk} = \left(\left(\mathsf{DC.vk}_{k,i}\right)_{k\in[K], i\in[\ell]}, \left(\mathsf{MEQ.vk}_i\right)_{i\in[\ell]}\right) \ .$$

- Let $(\mathsf{pk}^*, \mathsf{vk}^*)$ be the same as $(\mathsf{pk}, \mathsf{vk})$ except that we replace $(\mathsf{MEQ.pk}_{i^*}, \mathsf{MEQ.vk}_{i^*})$ by $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*)$.

In the rest of the experiment we use $(\mathsf{pk}^*, \mathsf{vk}^*)$ instead of $(\mathsf{pk}, \mathsf{vk})$. By the correctness of random extension (Definition 6.11), the distribution of $\mathbf{c}$ in Exp_1 and in Exp_2 is identical. Also, by the correctness of restriction (Definition 6.10), $\mathbf{c}^* = (c_{k,i})_{k\in[K]}$. Therefore, it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \ \wedge \ B_{k^*,i,j} = B'_{k^*,i,j} \ \rightarrow \ (\mathbf{B}_{i,j}, \mathsf{MEQ.}\Pi_{i,j}) = \left(\mathbf{B'}_{i,j}, \mathsf{MEQ.}\Pi'_{i,j}\right)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{19}$$

We can emulate Exp_2 given $(\mathsf{MEQ.pk}^*, \mathsf{MEQ.vk}^*)$ and $\mathbf{c}^*$ as input. Therefore, Equation (19) follows by the unambiguity of the multi-equality protocol (Definition 7.6), concluding the proof of the claim. □

7.6 Sum-check Protocol

This section describes the sum-check protocol which is implicit in [PR17, KPY19]. The protocol's CRS contains a single-key ciphertext c encrypting a message $\mathbf{m} = (m_1, \ldots, m_\ell)$. The honest prover is given a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $F(\mathbf{m}^*) = 0$ for every Boolean input $\mathbf{m}^* \in \{0,1\}^\ell$. The prover homomorphically evaluates F on c and sends the resulting ciphertext e to the verifier together with a proof convincing the verifier that F vanishes on Boolean inputs. Formally, the soundness requirement states that if $\mathbf{m} \in \{0,1\}^\ell$ and the verifier accepts the proof then e encrypts 0.

Note that the verifier cannot simply apply the zero-test to e. This test will fail since F is not the zero polynomial. Instead we use a sum-check proof, based on the PCP of [BFLS91]. For every $i \in [\ell]$ let F_i be the polynomial that is obtained from F by linearizing the first i variables:

$$F_i(z_1, \ldots, z_\ell) \equiv \sum_{v_1,\ldots,v_i \in \{0,1\}} \mathsf{ID}(z_1, \ldots, z_i, v_1, \ldots, v_i) \cdot F(v_1, \ldots, v_i, z_{i+1}, \ldots, z_\ell) \ ,$$

where $\mathsf{ID}(\mathbf{z}, \mathbf{v})$ is the multilinear extension of the Boolean identity function that outputs 1 if and only if $\mathbf{z} = \mathbf{v}$. Note that since F vanishes on Boolean inputs, so does F_i for every $i \in [\ell]$. In particular, since F_ℓ is multilinear, it is the zero polynomial.

For every $i \in [\ell]$ the sum-check proof includes the ciphertext e_i that results from homomorphically evaluating F_i on c. Let $F_0 = F$ and $e_0 = e$. Using the zero-test, the verifier checks that e_i and e_{i-1} are consistent following the identity:

$$F_i(z_1, \ldots, z_\ell) \equiv \sum_{v_i \in \{0,1\}} \mathsf{ID}(z_i, v_i) \cdot F_{i-1}(z_1, \ldots, z_{i-1}, v_i, z_{i+1}, \ldots, z_\ell) \ .$$

To allow the verifier to perform these tests, the CRS also includes encryptions of the values $\mathsf{ID}(z_i, v_i)$ for $v_i \in \{0,1\}$ and the sum-check proof includes the decomposition of F_{i-1} in the i'th coordinate. Finally, the verifier checks that e_ℓ encrypts zero by using the zero-test. This zero-test passes since F_ℓ is the zero polynomial.

The soundness of the sum-check proof follows from the soundness of the zero-test. Since e_ℓ passes the zero-test, it must encrypt zero. If $\mathbf{m} \in \{0,1\}^\ell$ we have that $\mathsf{ID}(m_i, v_i) = 1$ for $m_i = v_i$ and $\mathsf{ID}(m_i, v_i) = 0$ for $m_i \neq v_i$. Therefore, the consistency test between e_i and e_{i-1} guarantees that the two ciphertexts encrypt the same value. It follows that $e = e_0$ must also encrypt zero.

To show the unambiguity of the sum-check protocol we show that the unambiguity of e_i follows from that of e_{i-1}. We then use the unambiguity of the decomposition protocol to argue that the entire sum-check proof is unambiguous. To argue the the unambiguity of e_i, we consider a CRS that contains an encryption of a random message $\mathbf{m} \in \mathbb{F}^\ell$. We first show that if e_{i-1} is unambiguous, then the value encrypted in e_i must also be unambiguous. This follows from the consistency between e_i and e_{i-1}. (This holds for arbitrary message $\mathbf{m} \in \mathbb{F}^\ell$, not just for a Boolean one.) Then, the unambiguity of e_i follows from the unambiguity of ciphertexts property of the encryption.

7.6.1 Definition.

The sum-check protocol consists of algorithms $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ with the following syntax:

Setup: The PPT setup algorithm $\mathsf{SC.S}$ takes as input public parameters pp for the encryption scheme and a single-key ciphertext c encrypting a message of length ℓ. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic polynomial-time prover algorithm $\mathsf{SC.P}$ takes as input a prover key pk and a polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $F(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^\ell$. It outputs an evaluated ciphertext e and a proof Π.

Verifier: The deterministic polynomial-time verifier algorithm $\mathsf{SC.V}$ takes as input a verifier key vk, an evaluated ciphertext e and a proof Π. It outputs a bit indicating if it accepts or rejects.

Definition 7.13. *A Λ-secure sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $F(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^\ell$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{SC.V}\left(\mathsf{vk}, e, \Pi\right) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ e = \left[F(c)\right]_{\mathsf{pp}} & (\mathsf{sk}, c, \mathbf{r}) \leftarrow \mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{SC.S}(\mathsf{pp}, c) \\ & (e, \Pi) \leftarrow \mathsf{SC.P}(\mathsf{pk}, F) \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above $|\mathsf{vk}| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$ and $|\Pi| = \mathrm{poly}(\kappa, \bar{\ell}, \bar{\delta})$.*

Soundness. *For every $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \{0,1\}^\ell$ such that $\ell \leq \bar{\ell}$ and adversary Adv:*

$$\Pr\left[\begin{array}{l|l} \mathsf{SC.V}\left(\mathsf{vk}, e, \Pi\right) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ v \neq \langle 0 \rangle_{\mathsf{pp}} & Z = (\mathsf{sk}, c, \mathbf{r}) \leftarrow \mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}} \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{SC.S}(\mathsf{pp}, c) \\ & (e, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, Z) \\ & v \leftarrow \mathsf{Dec}(\mathsf{sk}, e) \end{array}\right] = 0 \ .$$

Λ-Unambiguity. *For every $\mathrm{poly}(\Lambda)$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{SC.V}\left(\mathsf{vk}, e, \Pi\right) = 1 & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ \mathsf{SC.V}\left(\mathsf{vk}, e, \Pi'\right) = 1 & (\mathsf{sk}, c, \mathbf{r}) \leftarrow \mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}} \\ \Pi \neq \Pi' & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{SC.S}(\mathsf{pp}, c) \\ & (e, \Pi, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, c) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

7.6.2 Construction.

We construct a sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ as follows. The construction uses a decomposition protocol $(\mathsf{DC.S}, \mathsf{DC.P}, \mathsf{DC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

The setup algorithm $\mathsf{SC.S}$. The setup algorithm is given as input:

- Public parameters pp for the encryption scheme.
- A single-key ciphertext c encrypting a message of length ℓ.

It proceeds as follows:

- For every $i \in [\ell]$ set $(\mathsf{DC.pk}_i, \mathsf{DC.vk}_i) \leftarrow \mathsf{DC.S}(\mathsf{pp}, c, i)$.
- For $i \in [\ell]$ and $b \in \{0,1\}$ let $\mathsf{ID}_{i,b}(z_1, \ldots, z_\ell) \equiv z_i \cdot b + (1 - z_i)(1 - b)$.
- For every $i \in [\ell]$ set $B_{i,0} \leftarrow \left[\mathsf{ID}_{i,0}(c)\right]_{\mathsf{pp}}$ and $B_{i,1} \leftarrow \left[\mathsf{ID}_{i,1}(c)\right]_{\mathsf{pp}}$.
- Set $e_1 \leftarrow \left[\overline{1}(c)\right]_{\mathsf{pp}}$ and $A_{\ell+1} \leftarrow \left[\overline{0}(c)\right]_{\mathsf{pp}}$.
- Output the prover key and verifier key:

$$\mathsf{pk} = \left(\mathsf{pp}, c, \left(\mathsf{DC.pk}_i\right)_{i \in [\ell]}\right) \quad , \quad \mathsf{vk} = \left(\mathsf{pp}, \left(\mathsf{DC.vk}_i, B_{i,0}, B_{i,1}\right)_{i \in [\ell]}, e_1, A_{\ell+1}\right) \ .$$

The prover algorithm $\mathsf{SC.P}$. The prover algorithm is given as input:

- A prover key $\mathsf{pk} = \left(c, (\mathsf{DC.pk}_i)_{i\in[\ell]}\right)$.
- A polynomial $F : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $F(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^\ell$.

It proceeds as follows:

- Let $F_0 : \mathbb{F}^\ell \to \mathbb{F}$ be the polynomial F.
- For every $i \in [\ell]$ let $\mathsf{ID}(z_1, \ldots, z_i, z'_1, \ldots, z'_i)$ be the multilinear polynomial extending the Boolean equality function:
$$\mathsf{ID}(z_1, \ldots, z_i, z'_1, \ldots, z'_i) \equiv \prod_{j\in[i]} \left(z_j \cdot z'_j + (1 - z_j)(1 - z'_j)\right) \ .$$
- For every $i \in [\ell - 1]$ let $F_i : \mathbb{F}^\ell \to \mathbb{F}$ be the polynomial:
$$F_i(z_1, \ldots, z_\ell) \equiv \sum_{v_1,\ldots,v_i\in\{0,1\}} \mathsf{ID}(z_1, \ldots, z_i, v_1, \ldots, v_i) \cdot F_0(v_1, \ldots, v_i, z_{i+1}, \ldots, z_\ell) \ .$$
- Set $e \leftarrow [F(c)]_{\mathsf{pp}}$.
- For every $i \in [\ell]$ set $(A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i) \leftarrow \mathsf{DC.P}(\mathsf{DC.pk}_i, F_{i-1})$.
- Output the evaluated ciphertext e and proof $\Pi = (A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)_{i\in[\ell]}$.

The verifier algorithm $\mathsf{SC.V}$. The verifier algorithm is given as input:

- A verifier key $\mathsf{vk} = \left(\mathsf{pp}, (\mathsf{DC.vk}_i, B_{i,0}, B_{i,1})_{i\in[\ell]}, e_1, A_{\ell+1}\right)$.
- An evaluated ciphertext e.
- A proof $\Pi = \left(A_i, \mathbf{Q}_i = (Q_{i,j})_{j\in[0,\bar{\delta}]}, \mathsf{DC.}\Pi_i\right)_{i\in[\ell]}$.

It proceeds as follows:

- Test that $e = A_1$.
- For every $i \in [\ell]$ test that $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)$ accepts.
- For every $i \in [\ell]$ test that:
$$\left[B_{i,0} \cdot Q_{i,0} + B_{i,1} \cdot \sum_{j\in[0,\bar{\delta}]} Q_{i,j} = A_{i+1} \cdot e_1\right]_{\mathsf{pp}} \ .$$
- Output 1 if all tests pass. Otherwise output 0.

7.6.3 Analysis.

In this section we prove the following theorem:

Theorem 7.14. *For any $\Lambda(\kappa)$, $\bar{\delta}(\kappa)$ and $\bar{\ell}(\kappa)$ such that $\bar{\ell} \leq \Lambda^{O(1)}$ and $\Lambda \cdot \bar{\delta} = |\mathbb{F}|^{o(1)}$, assuming that:*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

is a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$, the sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ given in Section 7.6.2 is a Λ-secure sum-check protocol (Definition 7.13) with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$.

Completeness. Fix any $\kappa \in \mathbb{N}$, message $\mathbf{m} \in \mathbb{F}^{\ell}$ such that $\ell \leq \bar{\ell}$ and polynomial $F : \mathbb{F}^{\ell} \to \mathbb{F}$ of individual degree at most $\bar{\delta}$ such that $F(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^{\ell}$. Let the verifier's input in the completeness experiment include the key vk, ciphertext e and proof Π where:

$$\mathsf{vk} = \left(\mathsf{pp}, (\mathsf{DC.vk}_i, B_{i,0}, B_{i,1})_{i\in[\ell]}, e_1, A_{\ell+1}\right) \quad , \quad \Pi = \left(A_i, \mathbf{Q}_i = (Q_{i,j})_{j\in[0,\bar{\delta}]}, \mathsf{DC}.\Pi_i\right)_{i\in[\ell]} \ .$$

By construction, for every $i \in [\ell]$ and $b \in \{0,1\}$:

$$B_{i,b} = [\mathsf{ID}_{i,b}(c)]_{\mathsf{pp}} \quad , \quad e_1 = \left[\overline{1}(c)\right]_{\mathsf{pp}} \quad , \quad A_{\ell+1} = \left[\overline{0}(c)\right]_{\mathsf{pp}} \quad , \quad e = [F(c)]_{\mathsf{pp}} \ .$$

By the completeness of the decomposition protocol (Definition 7.1) for every $i \in [\ell]$, $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC}.\Pi_i)$ accepts and for every $j \in [0, \bar{\delta}]$:

$$A_i = [F_{i-1}(c)]_{\mathsf{pp}} \quad , \quad Q_{i,j} = \left[F_{i-1}|_{i,j}(c_{-i})\right]_{\mathsf{pp}} \ .$$

Let $\bar{F}_{i,j} : \mathbb{F}^{\ell} \to \mathbb{F}$ be the polynomial such that $\bar{F}_{i,j}(\mathbf{z}) \equiv F_{i-1}|_{i,j}(\mathbf{z}_{-i})$. By the correctness of restriction (Definition 6.10) and the stability of evaluation (Definition 6.2) we have that $Q_{i,j} = \left[\bar{F}_{i,j}(c)\right]_{\mathsf{pp}}$.

Recall that $F_0 \equiv F$ and for every $i \in [\ell - 1]$ we set:

$$F_i(z_1, \ldots, z_\ell) \equiv \sum_{v_1,\ldots,v_i \in \{0,1\}} \mathsf{ID}(z_1, \ldots, z_i, v_1, \ldots, v_i) \cdot F_0(v_1, \ldots, v_i, z_{i+1}, \ldots, z_\ell) \ .$$

Similarly, let F_ℓ be the polynomial:

$$F_\ell(\mathbf{z}) \equiv \sum_{\mathbf{v}\in\{0,1\}^{\ell}} \mathsf{ID}(\mathbf{z}, \mathbf{v}) \cdot F_0(\mathbf{v}) \ .$$

Since $F(\mathbf{v}) = 0$ for every $\mathbf{v} \in \{0,1\}^{\ell}$ we have that $F_\ell \equiv \overline{0}$ and in particular, $A_{\ell+1} = [F_\ell(c)]_{\mathsf{pp}}$.

For every $i \in [\ell]$ we have that:

$$\mathsf{ID}_{i,0}(\mathbf{z}) \cdot \bar{F}_{i,0}(\mathbf{z}) + \mathsf{ID}_{i,1}(\mathbf{z}) \cdot \sum_{j\in[0,\bar{\delta}]} \bar{F}_{i,j}(\mathbf{z}) \equiv$$

$$\sum_{b\in\{0,1\}} \mathsf{ID}_{i,b}(\mathbf{z}) \cdot F_{i-1}(\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, b, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_\ell) \equiv F_i \cdot \overline{1} \ .$$

Therefore, by the weak completeness of the zero-test (Definition 6.7) the verifier's zero-test passes:

$$\left[B_{i,0} \cdot Q_{i,0} + B_{i,1} \cdot \sum_{j\in[0,\bar{\delta}]} Q_{i,j} = A_{i+1} \cdot e_1\right]_{\mathsf{pp}} \ .$$

Soundness. Fix any $\kappa \in \mathbb{N}$, message $\mathbf{m} = (m_1, \ldots, m_\ell) \in \{0,1\}^{\ell}$ such that $\ell \in [\bar{\ell}]$ and adversary Adv. Let the verifier's input in the soundness experiment include the key vk, ciphertext e and proof Π where:

$$\mathsf{vk} = \left(\mathsf{pp}, (\mathsf{DC.vk}_i, B_{i,0}, B_{i,1})_{i\in[\ell]}, e_1, A_{\ell+1}\right) \quad , \quad \Pi = \left(A_i, \mathbf{Q}_i = (Q_{i,j})_{j\in[0,\bar{\delta}]}, \mathsf{DC}.\Pi_i\right)_{i\in[\ell]} \ .$$

By construction, for $i \in [\ell]$:

$$\mathsf{Dec}(\mathsf{sk}, B_{i,0}) = \langle 1 - m_i \rangle_{\mathsf{pp}} \quad , \quad \mathsf{Dec}(\mathsf{sk}, B_{i,1}) = \langle m_i \rangle_{\mathsf{pp}} \quad , \quad \mathsf{Dec}(\mathsf{sk}, e_1) = \langle 1 \rangle_{\mathsf{pp}} \ .$$

Let $v \leftarrow \mathsf{Dec}(\mathsf{sk}, e)$, and for every $i \in [\ell]$ and $j \in [0, \bar{\delta}]$ let:

$$\alpha_i \leftarrow \mathsf{Dec}(\mathsf{sk}, A_i) \quad , \quad \beta_{i,j} \leftarrow \mathsf{Dec}(\mathsf{sk}, Q_{i,j}) \ ,$$

and let $\alpha_{\ell+1} = \mathsf{Dec}(\mathsf{sk}, A_{\ell+1}) = \langle 0 \rangle_{\mathsf{pp}}$.

If the verifier $\mathsf{SC.V}$ accepts then:

- $e = A_1$ and therefore $v = \alpha_1$.
- For every $i \in [\ell]$, $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)$ accepts. Therefore, by the soundness of the decomposition protocol (Definition 7.1):

$$\alpha_i = \sum_{j \in [0,\bar{\delta}]} \beta_{i,j} \cdot \langle m_i \rangle_{\mathsf{pp}}^j \quad .$$

- For every $i \in [\ell]$ the following zero-test passes:

$$\left[B_{i,0} \cdot Q_{i,0} + B_{i,1} \cdot \sum_{j \in [0,\bar{\delta}]} Q_{i,j} = A_{i+1} \cdot e_1 \right]_{\mathsf{pp}} \quad .$$

  Therefore, by the soundness of the zero-test (Definition 6.8) for every $i \in [\ell]$:

$$\alpha_{i+1} = \langle 1 - m_i \rangle_{\mathsf{pp}} \cdot \beta_{i,0} + \langle m_i \rangle_{\mathsf{pp}} \cdot \sum_{j \in [0,\bar{\delta}]} \beta_{i,j} = \beta_{i,0} + \langle m_i \rangle_{\mathsf{pp}} \cdot \sum_{j \in [\bar{\delta}]} \beta_{i,j} \quad .$$

Since $\mathbf{m} \in \{0,1\}^\ell$:

$$\alpha_{i+1} = \beta_{i,0} + \langle m_i \rangle_{\mathsf{pp}} \cdot \sum_{j \in [\bar{\delta}]} \beta_{i,j} = \sum_{j \in [0,\bar{\delta}]} \beta_{i,j} \cdot \langle m_i \rangle_{\mathsf{pp}}^j = \alpha_i \quad .$$

Therefore, $v = \alpha_1 = \alpha_{\ell+1} = \langle 0 \rangle_{\mathsf{pp}}$ as required.

Λ-Unambiguity. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv, $\kappa \in \mathbb{N}$ and message $\mathbf{m} \in \mathbb{F}^\ell$ such that $\ell \leq \bar{\ell}$ and let Exp denote the unambiguity experiment:

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa) \\ (\mathsf{sk}, c) \leftarrow \mathsf{Enc}\left[\mathbf{m}\right]_{\mathsf{pp}} \\ \left(\mathsf{pk} = \left(c, \left(\mathsf{DC.pk}_i\right)_{i \in [\ell]}\right), \mathsf{vk} = \left(\mathsf{pp}, \left(\mathsf{DC.vk}_i, B_{i,0}, B_{i,1}\right)_{i \in [\ell]}, e_1, A_{\ell+1}\right)\right) \leftarrow \mathsf{SC.S}(\mathsf{pp}, c) \\ \left(e, \Pi = \left(A_i, \mathbf{Q}_i = \left(Q_{i,j}\right)_{j \in [0,\bar{\delta}]}, \mathsf{DC.}\Pi_i\right)_{i \in [\ell]}, \Pi' = \left(A'_i, \mathbf{Q}'_i = \left(Q'_{i,j}\right)_{j \in [0,\bar{\delta}]}, \Pi'_i\right)_{i \in [\ell]}\right) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}$$

Let AC be the event that $\mathsf{SC.V}\left(\mathsf{vk}, e, \Pi\right)$ and $\mathsf{SC.V}\left(\mathsf{vk}, e, \Pi'\right)$ both accept. We need to show that:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi'\right] \leq \mathrm{negl}(\Lambda(\kappa)) \quad .$$

Let Exp_1 be the experiment that is defined just like Exp except that we sample $\mathbf{m} \leftarrow \mathbb{F}^\ell$. We can emulate Exp_1 given c as input. Therefore, by semantic security (Definition 6.5) it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi'\right] \leq \mathrm{negl}(\Lambda(\kappa)) \quad . \tag{20}$$

If AC occurs then:

- $e = A_1 = A'_1$
- For every $i \in [\ell]$, $\mathsf{DC.V}(\mathsf{DC.vk}_i, A_i, \mathbf{Q}_i, \mathsf{DC.}\Pi_i)$ and $\mathsf{DC.V}(\mathsf{DC.vk}_i, A'_i, \mathbf{Q}'_i, \Pi'_i)$ both accept. Also, we can emulate Exp_1 given $(\mathsf{DC.pk}_i, \mathsf{DC.vk}_i)$ and c as input. Therefore, by the unambiguity of the decomposition protocol (Definition 7.1):

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ A_i = A'_i \ \rightarrow \ (\mathbf{Q}_i, \Pi_i) = (\mathbf{Q}'_i, \Pi'_i)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \quad .$$

- For every $i \in [\ell - 1]$ the following zero-tests pass:

$$\left[B_{i,0} \cdot Q_{i,0} + B_{i,1} \cdot \sum_{j \in [0,\bar{\delta}]} Q_{i,j} = A_{i+1} \cdot e_1\right]_{\mathsf{pp}} \quad , \quad \left[B_{i,0} \cdot Q'_{i,0} + B_{i,1} \cdot \sum_{j \in [0,\bar{\delta}]} Q'_{i,j} = A'_{i+1} \cdot e_1\right]_{\mathsf{pp}} \quad .$$

  By construction $\mathsf{Dec}(\mathsf{sk}, e_1) \neq \langle 0 \rangle_{\mathsf{pp}}$ and thus, by the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ \mathbf{Q}_i = \mathbf{Q}'_i \ \rightarrow \ \mathsf{Dec}(\mathsf{sk}, A_{i+1}) = \mathsf{Dec}(\mathsf{sk}, A'_{i+1})\right] = 1 \ .$$

  We have that $\mathsf{DC.V}(\mathsf{DC.vk}_{i+1}, A_{i+1}, \mathbf{Q}_{i+1}, \mathsf{DC}.\Pi_{i+1})$ and $\mathsf{DC.V}(\mathsf{DC.vk}_{i+1}, A'_{i+1}, \mathbf{Q}'_{i+1}, \Pi'_{i+1})$ both accept. Therefore, $A_{i+1}, A'_{i+1} \in \mathsf{Valid}$. Also, we can emulate Exp_1 given c as input. Therefore, by the unambiguity of ciphertexts (Definition 6.6):

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ \mathsf{Dec}(\mathsf{sk}, A_{i+1}) = \mathsf{Dec}(\mathsf{sk}, A'_{i+1}) \ \rightarrow \ A_{i+1} = A'_{i+1}\right] \geq 1 - \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \ .$$

  Since $\bar{\delta} = 2$, $\bar{\ell} = 3 \log \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \ \wedge \ \mathsf{Dec}(\mathsf{sk}, A_{i+1}) = \mathsf{Dec}(\mathsf{sk}, A'_{i+1}) \ \rightarrow \ A_{i+1} = A'_{i+1}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\bar{\ell} = \Lambda^{O(1)}$, Equation (20) follows.

8 Quasi-argument

This section describes the unambiguous quasi-argument. We define quasi-arguments in Section 8.1. Our construction is given in Section 8.2 and the analysis is given in Section **??**. Parts of this section are taken verbatim from [KPY19].

8.1 Definition

Given an M-variate 3CNF formula and a locality parameter $K \leq M$, the quasi-argument setup algorithm generates prover and verifier keys. The honest prover is given an assignment σ satisfying the formula that includes an assignment x for the formula's input variables. It produces a proof that can be checked using x. Quasi-arguments satisfy a relaxed notion of soundness called non-signaling extraction. Loosely speaking, we consider an adaptive adversary acting as the prover that given honestly generated keys, produces an input x for φ together with a proof. We require that there exists a non-signaling extractor E that takes as input the formula φ along with a subset $\mathbf{S}$ of φ's variables of size at most K and samples an input x together with a partial assignment for the variables in $\mathbf{S}$.

We make the following requirements of the extractor E. First, we require that for every formula φ and set $\mathbf{S}$, whenever E samples $x \neq \perp$, the partial assignment sampled by E is consistent with x and it satisfies all of φ's clauses that are over the variables in $\mathbf{S}$. Second, the output distribution of E must satisfy the non-signaling requirement. Finally, for every formula φ and set $\mathbf{S}$, E must produce partial assignments for a distribution of inputs x that is indistinguishable from the distribution sampled by the adversary when its corresponding proof is accepting. More formally, we consider an experiment where the adversary produces an input x together with a proof, and if the proof is rejecting then we replace x with $\perp$. We require that this distribution of x is indistinguishable from x sampled by E.

Formally, for security parameter $\kappa \in \mathbb{N}$, the quasi-argument is parameterized by a formula size bound $\bar{M} = \bar{M}(\kappa)$ on the number of variables in the 3CNF formula and by the number of input variables $n = n(\kappa)$. The quasi-argument consists of algorithms $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ with the following syntax.

Setup: The probabilistic setup algorithm $\mathsf{QA.S}$ takes as input a security parameter κ, an M-variate 3CNF formula φ such that $M \leq \bar{M}$ and a locality parameter $K \leq M$. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic prover algorithm $\mathsf{QA.P}$ takes as input a prover key pk and an assignment $\sigma : [M] \to \{0,1\}$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{QA.V}$ takes as input a verifier key vk, an input $x \in \{0,1\}^n$ and a proof Π. It outputs a bit indicating if it accepts or rejects.

We rely on the following notion of locally satisfying assignments. In what follows, we denote the M variables of the formula φ by $z_1, \ldots, z_M$ and assume, without loss of generality, that for $i \in [n]$, the i'th input variable is z_i.

Definition 8.1 (Locally Satisfying Assignment)**.** *For an M-variate 3CNF formula φ and a set $\mathbf{S} \subseteq [M]$ we say that a partial assignment $\sigma : \mathbf{S} \to \{0,1\}$ locally satisfies φ if every clause in φ that only contains variables in $\mathbf{S}$ is satisfied by σ. We denote by $\varphi(\sigma)$ the bit indicating whether or not σ locally satisfies φ.*

Next we define the notion of a non-signaling extractor used in the definition of a quasi-argument.

Definition 8.2 (Λ-Non-signaling Extractor)**.** *A Λ-non-signaling extractor E with formula size bound $\bar{M}$, input length n and locality K satisfies the following requirements:*

Λ-Local consistency: *There exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, M-variate formula φ such that $M \leq \bar{M}$ and set $\mathbf{S} \subseteq [M]$ of size at most K:*

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}(\varphi, \mathbf{S})} \left[x = \bot \quad \vee \quad \begin{array}{l} \forall i \in \mathbf{S} \cap [n] : \ \sigma(i) = x_i \\ \varphi(\sigma) = 1 \end{array} \right] \geq 1 - \mu(\Lambda(\kappa)) \ .$$

Λ-Non-signaling: *For every $\mathrm{poly}(\Lambda)$-size distinguisher D there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$, M-variate formula φ such that $M \leq \bar{M}$ and sets $\mathbf{S}' \subseteq \mathbf{S} \subseteq [M]$ of size at most K:*

$$\left| \Pr_{(x,\sigma) \leftarrow \mathsf{E}(\varphi, \mathbf{S})} [\mathsf{D}(x, \sigma(\mathbf{S}')) = 1] - \Pr_{(x,\sigma') \leftarrow \mathsf{E}(\varphi, \mathbf{S}')} [\mathsf{D}(x, \sigma') = 1] \right| \leq \mu(\Lambda(\kappa)) \ .$$

Definition 8.3 (Quasi-argument)**.** *A Λ-secure quasi-argument $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ with formula size bound $\bar{M}$ and input length n satisfies the following requirements:*

Completeness. *For every $\kappa \in \mathbb{N}$, M-variate 3CNF formula φ such that $M \leq \bar{M}$ locality parameter $K \leq M$ and assignment $\sigma : [M] \to \{0,1\}$ satisfying φ:*

$$\Pr \left[\mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 1 \ \middle| \ \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, K) \\ x \leftarrow \sigma([n]) \\ \Pi \leftarrow \mathsf{QA.P}(\mathsf{pk}, \sigma) \end{array} \right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time $\mathrm{poly}(\kappa, \bar{M})$.*
- *The prover runs in polynomial time and it outputs a proof Π of length $K \cdot \mathrm{poly}(\kappa, \log \bar{M})$.*
- *The verifier runs in time $(K + n) \cdot \mathrm{poly}(\kappa, \log \bar{M})$.*

Λ-Non-signaling extraction. *For every function K there exists a PPT oracle machine E such that for every $\mathrm{poly}(\Lambda)$-size adversary Adv:*

- *E makes a single oracle query to Adv.*
- *$\mathsf{E}^{\mathsf{Adv}}$ is a Λ-non-signaling extractor with formula size bound $\bar{M}$, input length n and locality K.*
- *For every $\mathrm{poly}(\Lambda)$-size distinguisher D there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and M-variate formula φ such that $K \leq M \leq \bar{M}$:*

$$\left| \Pr \left[\mathsf{D}(x) = 1 \ \middle| \ \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, K) \\ (x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \\ \textit{if } \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0 : \textit{set } x = \bot \end{array} \right] - \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \emptyset)} [\mathsf{D}(x) = 1] \right| \leq \mu(\Lambda(\kappa)) \ .$$

Below we define the notion of an unambiguous quasi-argument. Since in general, a formula φ may have multiple satisfying assignments, we cannot hope to construct a quasi-argument that has completeness, for every φ and satisfying assignment σ, and also has unambiguous proofs. The unambiguity property of our quasi-argument states that if a prover outputs two different accepting proofs, then there exist two non-signaling local extractors that output different assignments for some variable of φ.

Definition 8.4 (Λ-Unambiguity)**.** *A quasi-argument* $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ *with formula size bound* $\bar{M}$ *and input length* n *is* Λ*-unambiguous if for every function* K *there exist a pair of PPT oracle machines* $\mathsf{E}_1, \mathsf{E}_2$ *such that for every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv*:*

- E_1 *and* E_2 *make a single oracle query to* Adv*.*
- $\mathsf{E}_1^{\mathsf{Adv}}$ *and* $\mathsf{E}_2^{\mathsf{Adv}}$ *are* Λ*-non-signaling extractors with formula size bound* $\bar{M}$*, input length* n *and locality* K*.*
- *There exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*,* M*-variate formula* φ *such that* $K \leq M \leq \bar{M}$ *and sets* $\mathbf{S}_1, \ldots, \mathbf{S}_B$ *such that* $|\mathbf{S}_i| \leq K$ *and* $\bigcup_{i\in[B]} \mathbf{S}_i = [M]$*:*

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 1 \\ \mathsf{QA.V}(\mathsf{vk}, x, \Pi') = 1 \\ \Pi \neq \Pi' \end{array} \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, K) \\ (x, \Pi, \Pi') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}\right] \leq$$
$$\sum_{i\in[B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \end{array} \forall j \in [2] : \ (x_j, \sigma_j) \leftarrow \mathsf{E}_j^{\mathsf{Adv}}(\varphi, \mathbf{S}_i \ ; \ r)\right] + \mu(\Lambda(\kappa))$$

8.2 Construction

We construct a quasi-argument $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ with formula size bound $\bar{M}$ and input length n. We start by introducing notation.

The construction uses a zero-testable homomorphic encryption scheme:

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

with degree bound $\bar{\delta} = 2$, message length bound $\bar{\ell} = 3 \log \bar{M}$ and field $\mathbb{F} = \mathbb{F}_\kappa$ (Definition 6.14). We also use the multilinearity protocol $(\mathsf{ML.S}, \mathsf{ML.P}, \mathsf{ML.V})$ with message length bound $\bar{\ell}$ and locality $\bar{K} = \bar{M} + \bar{\ell} + 3$ (Definition 7.9) and the sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ with degree bound $\bar{\delta}$ and message length bound $\bar{\ell}$ (Definition 7.13). We use the notation introduced in Sections 6.1 and 7.1.

The input formula. In our quasi-argument, the 3CNF formula φ has n input variables and M variables in total. The honest prover is given an assignment σ that satisfies φ and is consistent with some input $x \in \{0,1\}^n$ given to the verifier. It produces a proof Π. To check the consistency of x and Π we define an a 3CNF formula I_x. For every $i \in [n]$ and $b \in \{0,1\}$ let $I_{i,b}$ be the clause $I_{i,b}(z_i) = (z_i = b \ \vee \ z_i = b \ \vee \ z_i = b)$ and for any $x \in \{0,1\}^n$ let $I_x = \bigwedge_{i\in[n]} I_{i,x_i}$. We assume without loss of generality that φ does not contain any clauses of the form $I_{i,b}$. Note that for every set $\mathbf{S}$ and partial assignment $\sigma : \mathbf{S} \to \{0,1\}$, if $I_x(\sigma) = 1$ then $\sigma(i) = x_i$ for every $i \in \mathbf{S} \cap [n]$.

Formula arithmetization. We represent 3CNF formulas as multilinear polynomials. Let φ be an M-variate 3CNF formula over the variables $z_1, \ldots, z_M$. We identify the variables' indices with strings in $\{0,1\}^\ell$ for $\ell = \log M$.

Definition 8.5. *A multilinear polynomial* $\hat{\varphi} : \mathbb{F}^{3\ell+3} \to \mathbb{F}$ *is an arithmetization of the 3CNF formula* φ *if the following holds: for every triplet of indices* $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \{0,1\}^\ell$ *and bits* $b_1, b_2, b_3 \in \{0,1\}$*,* $\hat{\varphi}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, b_1, b_2, b_3)$ *outputs* 1 *if* φ *contains the clause* $(z_{\mathbf{v}_1} = b_1 \ \vee \ z_{\mathbf{v}_2} = b_2 \ \vee \ z_{\mathbf{v}_3} = b_3)$ *and* 0 *otherwise.*

Let $\hat{\varphi}, \hat{I}_{i,b}, \hat{I}_x$ denote the arithmetizations of the formulas $\varphi, I_{i,b}, I_x$ respectively. Observe that $\hat{I}_x = \sum_{i\in[n]} \hat{I}_{i,x_i}$. Since φ and I_x do not have any clauses in common, $\hat{\varphi} + \hat{I}_x$ is an arithmetization of the formula $\varphi \wedge I_x$.

Next we describe the quasi-argument algorithms $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$.

The setup algorithm $\mathsf{QA.S}$**.** The setup algorithm is given as input:

- A security parameter κ.
- An M-variate 3CNF formula φ such that $M \leq \bar{M}$.
- A locality parameter $K \leq M$.

It proceeds as follows:

- Let $\ell = \log M$ and $K' = K + \ell$.
- Set $\mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa)$.
- For every $k \in [K']$ sample $\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{pp})$, $\mathbf{r}_k \leftarrow \{0,1\}^{\kappa \times \ell}$ and set $c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, 0^\ell, \mathbf{r}_k)$.
- Sample $\tilde{\mathsf{sk}} \leftarrow \mathsf{KeyGen}(\mathsf{pp})$. For every $j \in [3]$ sample $\tilde{\mathbf{r}}_j \leftarrow \{0,1\}^{\kappa \times \ell}$ and set $\tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^\ell, \tilde{\mathbf{r}}_j)$.
- Set $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{3\ell}, (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$
- For $j \in [3]$ set $(\mathsf{ML.pk}_j, \mathsf{ML.vk}_j) \leftarrow \mathsf{ML.S}(\mathsf{pp}, (c_1, \ldots, c_{K'}, \tilde{c}_j))$.
- Set $(\mathsf{SC.pk}, \mathsf{SC.vk}) \leftarrow \mathsf{SC.S}(\mathsf{pp}, \tilde{c})$.
- Set $e_0 \leftarrow \left[\overline{0}(\tilde{c})\right]_{\mathsf{pp}}$ and $e_1 \leftarrow \left[\overline{1}(\tilde{c})\right]_{\mathsf{pp}}$.
- For every $\mathbf{b} \in \{0,1\}^3$ set $A_{\mathbf{b}} \leftarrow [\hat{\varphi}(\tilde{c}, \mathbf{b})]_{\mathsf{pp}}$.
- For every $\mathbf{b} \in \{0,1\}^3$, $i \in [n]$ and $b \in \{0,1\}$ set $B_{\mathbf{b},i,b} \leftarrow \left[\hat{I}_{i,b}(\tilde{c}, \mathbf{b})\right]_{\mathsf{pp}}$.
- Output the prover key and verifier key:

$$\mathsf{pk} = \left(\varphi, \mathsf{pp}, \left(\mathsf{ML.pk}_j\right)_{j\in[3]}, \mathsf{SC.pk}, \tilde{c}\right) \ ,$$

$$\mathsf{vk} = \left(\mathsf{pp}, \left(\mathsf{ML.vk}\right)_{j\in[3]}, \mathsf{SC.vk}, e_0, e_1, \left(A_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(B_{\mathbf{b},i,b}\right)_{\mathbf{b}\in\{0,1\}^3, i\in[n], b\in\{0,1\}}\right) \ .$$

The prover algorithm $\mathsf{QA.P}$**.** The prover algorithm is given as input:

- A prover key $\mathsf{pk} = \left(\varphi, \mathsf{pp}, \left(\mathsf{ML.pk}_j\right)_{j\in[3]}, \mathsf{SC.pk}, \tilde{c}\right)$.
- An assignment $\sigma : [M] \to \{0,1\}$ satisfying φ .

It proceeds as follows:

- Let $\mathsf{ID}(\mathbf{z}, \mathbf{v})$ be the multilinear polynomial extending the Boolean equality function:

$$\mathsf{ID}(z_1, \ldots, z_\ell, v_1, \ldots, v_\ell) := \prod_{j\in[\ell]} (z_j \cdot v_j + (1 - z_j)(1 - v_j)) \ .$$

- Let $\Sigma : \mathbb{F}^\ell \to \mathbb{F}$ be the multilinear extension of the assignment σ:

$$\Sigma(\mathbf{z}) := \sum_{\mathbf{v}\in\{0,1\}^\ell} \mathsf{ID}(\mathbf{z}, \mathbf{v}) \cdot \sigma(\mathbf{v}) \ .$$

- For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and non-empty subset $J \subseteq [3]$ let $\Sigma_{\mathbf{b},J} : \mathbb{F}^{3\ell} \to \mathbb{F}$ be the polynomial:

$$\Sigma_{\mathbf{b},J}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) := \prod_{j\in J} (\Sigma(\mathbf{z}_j) - b_j) \ .$$

- Let $x = \sigma([n])$ and for every $\mathbf{b} \in \{0,1\}^3$ let $P_{\mathbf{b}} : \mathbb{F}^{3\ell} \to \mathbb{F}$ be the polynomial:

$$P_{\mathbf{b}}(\mathbf{z}) := (\hat{\varphi} + \hat{I}_x)(\mathbf{z}, \mathbf{b}) \cdot \Sigma_{\mathbf{b},[3]}(\mathbf{z}) \ .$$

- For $j \in [3]$ set $(\mathbf{C}_j, \mathsf{ML.\Pi}_j) \leftarrow \mathsf{ML.P}(\mathsf{ML.pk}_j, \Sigma)$.
- For every $\mathbf{b} \in \{0,1\}^3$ set $(D_{\mathbf{b}}, \mathsf{SC.\Pi}_{\mathbf{b}}) \leftarrow \mathsf{SC.P}(\mathsf{SC.pk}, P_{\mathbf{b}})$.
- For every $\mathbf{b} \in \{0,1\}^3$ and $J \subseteq [3]$ set $E_{\mathbf{b},J} \leftarrow [\Sigma_{\mathbf{b},J}(\tilde{c})]_{\mathsf{pp}}$.
- Output the proof:

$$\Pi = \left((\mathbf{C}_j, \mathsf{ML.\Pi}_j)_{j\in[3]}, (D_{\mathbf{b}}, \mathsf{SC.\Pi}_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^3}, (E_{\mathbf{b},J})_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]} \right) \ .$$

The verifier algorithm $\mathsf{QA.V}$. The verifier algorithm is given as input:

- A verifier key:

$$\mathsf{vk} = \left(\mathsf{pp}, (\mathsf{ML.vk}_j)_{j\in[3]}, \mathsf{SC.vk}, e_0, e_1, (A_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^3}, (B_{\mathbf{b},i,b})_{\mathbf{b}\in\{0,1\}^3, i\in[n], b\in\{0,1\}} \right) \ .$$

- An input $x \in \{0,1\}^n$.
- A proof:

$$\Pi = \left(\left(\mathbf{C}_j = \left(C_{j,1}, \ldots, C_{j,K'}, \tilde{C}_j \right), \mathsf{ML.\Pi}_j \right)_{j\in[3]}, (D_{\mathbf{b}}, \mathsf{SC.\Pi}_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^3}, (E_{\mathbf{b},J})_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]} \right) \ .$$

It proceeds as follows:

- For every $k \in [K']$ test that $C_{1,k} = C_{2,k} = C_{3,k}$
- For $k \in [K'], \mathbf{b} \in \{0,1\}^3$ and $J \subseteq [3]$ test that $C_{1,k}, D_{\mathbf{b}}, E_{\mathbf{b},J} \in \mathsf{Valid}$ (see Definition 6.6).
- For every $j \in [3]$ test that $\mathsf{ML.V}(\mathsf{ML.vk}_j, \mathbf{C}_j, \mathsf{ML.\Pi}_j)$ accepts.
- For every $\mathbf{b} \in \{0,1\}^3$ test that $\mathsf{SC.V}(\mathsf{SC.vk}, D_{\mathbf{b}}, \mathsf{SC.\Pi}_{\mathbf{b}})$ accepts.
- For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $j \in [3]$ test that:

$$\left[\left(E_{\mathbf{b},\{j\}} + e_{b_j} \right) \cdot e_1 = \tilde{C}_j \cdot e_1 \right]_{\mathsf{pp}} \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ and pair of non-empty disjoint subsets $J, J' \subseteq [3]$ test that:

$$\left[E_{\mathbf{b},J} \cdot E_{\mathbf{b},J'} = E_{\mathbf{b},J\cup J'} \cdot e_1 \right]_{\mathsf{pp}} \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ test that:

$$\left[D_{\mathbf{b}} \cdot e_1 = \left(A_{\mathbf{b}} + \sum_{i\in[n]} B_{\mathbf{b},i,x_i} \right) \cdot E_{\mathbf{b},[3]} \right]_{\mathsf{pp}} \ .$$

- Output 1 if all tests pass. Otherwise output 0.

8.3 Analysis

In this section we prove the following theorem:

Theorem 8.6. *For any $\Lambda(\kappa)$, $\bar{M}(\kappa)$ and $n(\kappa)$ such that $n \leq \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$, if there exists a Λ-secure zero-testable homomorphic encryption scheme (Definition 6.14):*

$$(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{MEnc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{ZT}, \mathsf{Extend}, \mathsf{Restrict}, \mathsf{RandExtend}, \mathsf{Rerand})$$

for degree bound $\bar{\delta} = 2$ and message length bound $\bar{\ell} = 3 \log \bar{M}$, then the quasi-argument $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ given in Section 8.2 is a Λ-secure Λ-unambiguous quasi-argument (Definitions 8.3 and 8.4) with formula size bound $\bar{M}$ and input length n.

8.3.1 Completeness.

Fix any $\kappa \in \mathbb{N}$, M-variate 3CNF formula φ such that $2^\ell = M \leq \bar{M}$, locality parameter $K \leq M$ and assignment $\sigma : [M] \to \{0,1\}$ satisfying φ. We need to prove that:

$$\Pr\left[\mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, K) \\ x \leftarrow \sigma([n]) \\ \Pi \leftarrow \mathsf{QA.P}(\mathsf{pk}, \sigma) \end{array}\right] = 1 \ ,$$

where:

$$\mathsf{vk} = \left(\mathsf{pp}, \mathsf{ML.vk}, (\mathsf{SC.vk})_{j\in[3]}, e_0, e_1, (A_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^3}, (B_{\mathbf{b},i,b})_{\mathbf{b}\in\{0,1\}^3, i\in[n], b\in\{0,1\}}\right) \ ,$$
$$\Pi = \left(\left(\mathbf{C}_j = \left(C_{j,1}, \ldots, C_{j,K'}, \tilde{C}_j\right), \mathsf{ML}.\Pi_j\right)_{j\in[3]}, (D_{\mathbf{b}}, \mathsf{SC}.\Pi_{\mathbf{b}})_{\mathbf{b}\in\{0,1\}^3}, (E_{\mathbf{b},J})_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]}\right) \ .$$

and:

$$\begin{aligned} e_b &= \left[\overline{b}(\tilde{c})\right]_{\mathsf{pp}} \quad : \quad b \in \{0,1\} \ , \\ A_{\mathbf{b}} &= \left[\hat{\varphi}(\tilde{c}, \mathbf{b})\right]_{\mathsf{pp}} \quad : \quad \mathbf{b} \in \{0,1\}^3 \ , \\ B_{\mathbf{b},i,b} &= \left[\hat{I}_{j,b}(\tilde{c}, \mathbf{b})\right]_{\mathsf{pp}} \quad : \quad \mathbf{b} \in \{0,1\}^3, i \in [n], b \in \{0,1\} \ , \\ E_{\mathbf{b},J} &= \left[\Sigma_{\mathbf{b},J}(\tilde{c})\right]_{\mathsf{pp}} \quad : \quad \mathbf{b} \in \{0,1\}^3, J \subseteq [3] \ . \end{aligned}$$

We show that every test in $\mathsf{QA.V}$ passes.

- Since Σ is a multilinear polynomial, by the completeness of the multilinearity protocol (Definition 7.9), for every $j \in [3]$, $\mathsf{ML.V}(\mathsf{ML.vk}_j, \mathbf{C}_j, \mathsf{ML}.\Pi_j)$ accepts and for every $k \in [K']$:

  $$C_{j,k} = \left[\Sigma(c_k)\right]_{\mathsf{pp}} \quad , \quad \tilde{C}_j = \left[\Sigma(\tilde{c}_j)\right]_{\mathsf{pp}} \ .$$

  In particular $C_{1,k} = C_{2,k} = C_{3,k}$ and $C_{1,k} \in \mathsf{Valid}$.

- Since the assignment σ satisfies φ and $x = \sigma([n])$, the assignment σ also satisfies $\varphi \wedge I_x$. Since $\hat{\varphi} + \hat{I}_x$ is an arithmetization of $\varphi \wedge I_x$, we have that for every $\mathbf{b} \in \{0,1\}^3$ and $\mathbf{z} = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) \in \{0,1\}^{3\ell}$:

  $$P_{\mathbf{b}}(\mathbf{z}) = (\hat{\varphi} + \hat{I}_x)(\mathbf{z}, \mathbf{b}) \cdot \Sigma_{\mathbf{b},[3]}(\mathbf{z}) = (\hat{\varphi} + \hat{I}_x)(\mathbf{z}, \mathbf{b}) \cdot \prod_{j\in[3]} \left(\Sigma(\mathbf{z}_j) - b_j\right) = 0 \ .$$

  Therefore, by the correctness of the sum-check protocol (Definition 7.13), $\mathsf{SC.V}(\mathsf{SC.vk}, D_{\mathbf{b}}, \mathsf{SC}.\Pi_{\mathbf{b}})$ accepts and:

  $$D_{\mathbf{b}} = \left[P_{\mathbf{b}}(\tilde{c})\right]_{\mathsf{pp}} \ .$$

- By the stability of evaluation (Definition 6.2), for every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $j \in [3]$:

  $$E_{\mathbf{b},\{j\}} = \left[\Sigma_{\mathbf{b},\{j\}}(\tilde{c})\right]_{\mathsf{pp}} = \left[\Sigma(\tilde{c}_j) - b_j\right]_{\mathsf{pp}} \quad , \quad e_0 = \left[\overline{0}(\tilde{c})\right]_{\mathsf{pp}} = \left[\overline{0}(\tilde{c}_j)\right]_{\mathsf{pp}} \quad , \quad e_1 = \left[\overline{1}(\tilde{c})\right]_{\mathsf{pp}} = \left[\overline{1}(\tilde{c}_j)\right]_{\mathsf{pp}} \ .$$

  Therefore, by the weak completeness of the zero-test (Definition 6.7) the following zero-test passes:

  $$\left[\left(E_{\mathbf{b},\{j\}} + e_{b_j}\right) \cdot e_1 = \tilde{C}_j \cdot e_1\right]_{\mathsf{pp}} \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ and pair of non-empty disjoint subsets $J, J' \subseteq [3]$, $\Sigma_{\mathbf{b},J} \cdot \Sigma_{\mathbf{b},J'} \equiv \Sigma_{\mathbf{b},J \cup J'}$. Therefore, by the weak completeness of the zero-test (Definition 6.7) the following zero-test passes:

  $$\left[E_{\mathbf{b},J} \cdot E_{\mathbf{b},J'} = E_{\mathbf{b},J \cup J'} \cdot e_1\right]_{\mathsf{pp}} \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ we have that:

  $$P_{\mathbf{b}}(\mathbf{z}) \equiv (\hat{\varphi} + \hat{I}_x)(\mathbf{z}, \mathbf{b}) \cdot \Sigma_{\mathbf{b},[3]}(\mathbf{z}) \equiv (\hat{\varphi} + \sum_{j \in [n]} \hat{I}_{j,x_j})(\mathbf{z}, \mathbf{b}) \cdot \Sigma_{\mathbf{b},[3]}(\mathbf{z}) \ .$$

  Therefore, by the weak completeness of the zero-test (Definition 6.7) the following zero-test passes:

  $$\left[D_{\mathbf{b}} \cdot e_1 = \left(A_{\mathbf{b}} + \sum_{j \in [n]} B_{\mathbf{b},j,x_j}\right) \cdot E_{\mathbf{b},[3]}\right]_{\mathsf{pp}} \ .$$

8.3.2 Non-signaling extraction.

For every function $K(\kappa)$ we construct a PPT oracle machine E. E is given as input an M-variate formula φ such that $M \leq \bar{M}$ and a set $\mathbf{S} \subseteq [M] = \{0,1\}^\ell$ of size at most K. It is also given oracle access to an adversary Adv. E proceeds as follows:

- Set $\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{S}|}$ to be the elements of $\mathbf{S}$ in an arbitrary order and set $\mathbf{v}_{|\mathbf{S}|+1}, \ldots, \mathbf{v}_{K'} = 0^\ell$.
- Set $\mathbf{u}_1, \ldots, \mathbf{u}_{K'}$ to be a random reordering of $\mathbf{v}_1, \ldots, \mathbf{v}_{K'}$.
- Emulate the setup algorithm $\mathsf{QA.S}(\kappa, \varphi, K)$ to sample keys $(\mathsf{pk}, \mathsf{vk})$ except that instead of setting $c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, 0^\ell, \mathbf{r}_k)$, set $c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, \mathbf{u}_k, \mathbf{r}_k)$.
- Set $(x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk})$ where:

  $$\Pi = \left(\left(\mathbf{C}_j = \left(C_{j,1}, \ldots, C_{j,K'}, \tilde{C}_j\right), \mathsf{ML}.\Pi_j\right)_{j \in [3]}, (D_{\mathbf{b}}, \mathsf{SC}.\Pi_{\mathbf{b}})_{\mathbf{b} \in \{0,1\}^3}, (E_{\mathbf{b},J})_{\mathbf{b} \in \{0,1\}^3, J \subseteq [3]}\right) \ .$$

- If $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ rejects then set $x = \bot$ and set $\sigma(\mathbf{v}) = 0$ for every $\mathbf{v} \in \mathbf{S}$.
- If $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ accepts then for $k \in [K']$ such that $\mathbf{u}_k \in \mathbf{S}$, if $\mathsf{Dec}(\mathsf{sk}_i, C_{1,k}) = \langle b \rangle_{\mathsf{pp}}$ for some $b \in \{0,1\}$, set $\sigma(\mathbf{u}_k) = b$. Otherwise set $\sigma(\mathbf{u}_k) = 0$.
- Output (x, σ).

Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv. We show that $\mathsf{E}^{\mathsf{Adv}}$ is indeed a Λ-non-signaling extractor (Definition 8.2) and that for every $\mathrm{poly}(\Lambda)$-size distinguisher D there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and M-variate formula φ such that $K \leq M \leq \bar{M}$:

$$\left|\Pr\left[\mathsf{D}(x) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, K) \\ (x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \\ \text{if } \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0 : \text{set } x = \bot \end{array}\right] - \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \emptyset)}[\mathsf{D}(x) = 1]\right| \leq \mu(\Lambda(\kappa)) \ . \tag{21}$$

Since E given $\mathbf{S} = \emptyset$ sets $c_i \leftarrow \mathsf{Enc}(\mathsf{sk}_i, 0^\ell, \mathbf{r}_i)$ for very $i \in [K']$ the distribution of $(\mathsf{pk}, \mathsf{vk})$ in both experiments in Equation (21) is identical. Since in both experiment we set $(x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk})$ and then set $x = \bot$ if $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ rejects, Equation (21) follows. Next we show that $\mathsf{E}^{\mathsf{Adv}}$ satisfies the local consistency and non-signaling requirements (Definition 8.2).

Local consistency. Assume towards contradiction that there exists a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$, there exists an M-variate formula φ such that $M \leq \bar{M}$ and a set $\mathbf{S} \subseteq [M]$ of size at most K such that:

$$\Pr_{(x,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S})}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma) = 0 \end{array}\right] \geq \frac{1}{p(\Lambda(\kappa))} \ .$$

It follows that there exists $\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3 \in \mathbf{S}$ such that:

$$\Pr_{(x,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S})}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] \geq \frac{1}{M^3 \cdot p(\Lambda(\kappa))} \ .$$

Let Exp_1 be the experiment that emulates $\mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S})$ except that instead of setting $\tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^\ell, \tilde{\mathbf{r}}_j)$ we set:

$$\tilde{c}_1 = \mathsf{Restrict}(\tilde{c}, [\ell+1, 3\ell]) \quad , \quad \tilde{c}_2 = \mathsf{Restrict}(\tilde{c}, [\ell] \cup [2\ell+1, 3\ell]) \quad , \quad \tilde{c}_3 = \mathsf{Restrict}(\tilde{c}, [2\ell]) \ .$$

By the correctness of restriction (Definition 6.10):

$$\Pr_{\mathsf{Exp}_1}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] \geq \frac{1}{M^3 \cdot p(\Lambda(\kappa))} \ .$$

Let Exp_2 be the experiment that is defined just like Exp_1 except that instead of setting $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{3\ell}, (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$ we set $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, (\tilde{\mathbf{v}}_1 \mid \tilde{\mathbf{v}}_2 \mid \tilde{\mathbf{v}}_3), (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$. We can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by semantic security (Definition 6.5):

$$\Pr_{\mathsf{Exp}_2}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] \geq \frac{1}{M^3 \cdot p(\Lambda(\kappa))} - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $M \leq \bar{M} \leq \mathrm{poly}(\Lambda)$:

$$\Pr_{\mathsf{Exp}_2}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] \geq \frac{\Omega(1)}{M^3 \cdot p(\Lambda(\kappa))} \ .$$

Let Exp_3 be the experiment that is defined just like Exp_3 except that instead of setting $\tilde{c}_j$ by restricting $\tilde{c}$, we set $\tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, \tilde{\mathbf{v}}_j, \tilde{\mathbf{r}}_j)$. By the correctness of restriction (Definition 6.10):

$$\Pr_{\mathsf{Exp}_3}\left[\begin{array}{l} x \neq \bot \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] \geq \frac{\Omega(1)}{M^3 \cdot p(\Lambda(\kappa))} \ .$$

Let AC be the event that $x \neq \bot$ and $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ accepts. Since E sets $x = \bot$ whenever $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ rejects, it follows that:

$$\Pr_{\mathsf{Exp}_3}\left[\begin{array}{l} \mathsf{AC} \\ (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 0 \end{array}\right] > \frac{\Omega(1)}{M^3 \cdot p(\Lambda(\kappa))} \ . \tag{22}$$

In the experiment Exp_3 let:

$$\begin{aligned}
\mathsf{pk} &= \left(\varphi, \mathsf{pp}, \left(\mathsf{ML.pk}_j\right)_{j\in[3]}, \mathsf{SC.pk}, \tilde{c}\right) \\
\mathsf{vk} &= \left(\mathsf{pp}, \left(\mathsf{ML.vk}\right)_{j\in[3]}, \mathsf{SC.vk}, e_0, e_1, \left(A_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(B_{\mathbf{b},i,b}\right)_{\mathbf{b}\in\{0,1\}^3, i\in[n], b\in\{0,1\}}\right) \ , \\
\Pi &= \left(\left(\mathbf{C}_j = \left(C_{j,1}, \ldots, C_{j,K'}, \tilde{C}_j\right), \mathsf{ML}.\Pi_j\right)_{j\in[3]}, \left(D_{\mathbf{b}}, \mathsf{SC}.\Pi_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(E_{\mathbf{b},J}\right)_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]}\right) \ .
\end{aligned}$$

By construction we have that:

$$\begin{aligned}
\mathsf{Dec}(\tilde{\mathsf{sk}}, e_b) &= \langle b \rangle_{\mathsf{pp}} &&: \quad b \in \{0,1\} \ , \\
\mathsf{Dec}(\tilde{\mathsf{sk}}, A_{\mathbf{b}}) &= \langle \hat{\varphi}(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \rangle_{\mathsf{pp}} &&: \quad \mathbf{b} \in \{0,1\}^3 \ , \\
\mathsf{Dec}(\tilde{\mathsf{sk}}, B_{\mathbf{b},j,b}) &= \left\langle \hat{I}_{j,b}(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \right\rangle_{\mathsf{pp}} &&: \quad \mathbf{b} \in \{0,1\}^3, j \in [n], b \in \{0,1\} \ ,
\end{aligned}$$

Let:

$$\begin{array}{lll} \alpha_{j,k} = \mathsf{Dec}(\mathsf{sk}_k, C_{j,k}) & : & j \in [3], k \in [K'] \ , \\ \tilde{\alpha}_j = \mathsf{Dec}(\tilde{\mathsf{sk}}, \tilde{C}_j) & : & j \in [3] \ , \\ \xi_{\mathbf{b}} = \mathsf{Dec}(\tilde{\mathsf{sk}}, D_{\mathbf{b}}) & : & \mathbf{b} \in \{0,1\}^3 \ , \\ \zeta_{\mathbf{b},J} = \mathsf{Dec}(\tilde{\mathsf{sk}}, E_{\mathbf{b},J}) & : & \mathbf{b} \in \{0,1\}^3, J \subseteq [3] \ . \end{array}$$

For every $j \in [3]$, let $k_j \in [K]$ be the index such that $\tilde{\mathbf{v}}_j = \mathbf{u}_{k_j}$.

The proof of local consistency relies on the following claim.

Claim 8.7. *For every $j \in [3]$:*

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \ \rightarrow \ \alpha_{j,k_j} = \tilde{\alpha}_j\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving the claim we use it to prove local consistency. If AC occurs then $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ accepts and, therefore:

- For every $k \in [K']$, $\alpha_{j,k} = \alpha_{2,k} = \alpha_{3,k}$.
- For every $j \in [3]$, by Claim 8.7:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \alpha_{j,k_j} = \tilde{\alpha}_j\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$, $\mathsf{SC.V}(\mathsf{SC.vk}, D_{\mathbf{b}}, \mathsf{SC.}\Pi_{\mathbf{b}})$ accepts. By the soundness of the sum-check protocol (Definition 7.13):

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \xi_{\mathbf{b}} = \langle 0 \rangle_{\mathsf{pp}}\right] = 1 \ .$$

- For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $j \in [3]$ the following zero-test passes:

$$\left[\left(E_{\mathbf{b},\{j\}} + e_{b_j}\right) \cdot e_1 = \tilde{C}_j \cdot e_1\right]_{\mathsf{pp}} \ .$$

  By the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \zeta_{\mathbf{b},\{j\}} = \tilde{\alpha}_j - \langle b_j \rangle_{\mathsf{pp}}\right] = 1 \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ and pair of non-empty disjoint subsets $J, J' \subseteq [3]$ the following zero-test passes:

$$\left[E_{\mathbf{b},J} \cdot E_{\mathbf{b},J'} = E_{\mathbf{b},J \cup J'} \cdot e_1\right]_{\mathsf{pp}} \ .$$

  By the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \zeta_{\mathbf{b},J} \cdot \zeta_{\mathbf{b},J'} = \zeta_{\mathbf{b},J \cup J'}\right] = 1 \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ the following zero-test passes:

$$\left[D_{\mathbf{b}} \cdot e_1 = \left(A_{\mathbf{b}} + \sum_{j \in [n]} B_{\mathbf{b},j,x_j}\right) \cdot E_{\mathbf{b},[3]}\right]_{\mathsf{pp}}$$

  By the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \xi_{\mathbf{b}} = \left\langle \hat{\varphi}(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) + \sum_{j \in [n]} \hat{I}_{j,x_j}(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \right\rangle_{\mathsf{pp}} \cdot \zeta_{\mathbf{b},[3]}\right] = 1 \ .$$

  Since $\hat{I}_x = \sum_{j \in [n]} \hat{I}_{j,x_j}$:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \xi_{\mathbf{b}} = \langle (\hat{\varphi} + I_x)(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \rangle_{\mathsf{pp}} \cdot \zeta_{\mathbf{b},[3]}\right] = 1 \ .$$

For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ the above equities imply that:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad \langle 0 \rangle_{\mathsf{pp}} = \langle (\hat{\varphi} + I_x)(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \rangle_{\mathsf{pp}} \cdot \prod_{j \in [3]} \left(\alpha_{1,k_j} - \langle b_j \rangle_{\mathsf{pp}} \right)\right] = 1 \ .$$

Recall that if $\alpha_{1,k_j} = \langle b_j \rangle_{\mathsf{pp}}$ then E sets $\sigma(\tilde{\mathbf{v}}_j) = b_j$. Therefore:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad 0 = (\hat{\varphi} + I_x)(\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3, \mathbf{b}) \cdot \prod_{j \in [3]} (\sigma(\tilde{\mathbf{v}}_j) - b_j)\right] = 1 \ .$$

Since $\hat{\varphi} + I_x$ is an arithmetization of $\varphi \wedge I$ (Definition 8.5) it follows that:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \quad \rightarrow \quad (\varphi \wedge I_x)(\sigma(\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \tilde{\mathbf{v}}_3\})) = 1\right] = 1 \ ,$$

contradicting Equation (22). It remains to prove Claim 8.7.

Proof of Claim 8.7. Fix any $j \in [3]$. In the experiment Exp_3 let:

$$Z = \left(\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_{K'}, \tilde{\mathsf{sk}}) \ , \ \mathbf{c} = (c_1, \ldots, c_{K'}, \tilde{c}_j) \ , \ \mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_{K'}, \tilde{\mathbf{r}}_j)\right) \ .$$

If AC occurs then $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ accepts and, therefore, $\mathsf{ML.V}(\mathsf{ML.vk}_j, \mathbf{C}_j, \mathsf{ML.}\Pi_j)$ also accepts. Also, recall that in Exp_3 we set:

$$c_{k_j} \leftarrow \mathsf{Enc}(\mathsf{sk}_{k_j}, \mathbf{u}_{k_j}, \mathbf{r}_{k_j}) \quad , \quad \tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, \tilde{\mathbf{v}}_j, \tilde{\mathbf{r}}_j) \ .$$

Since $\mathbf{u}_{k_j} = \mathbf{v}_j$, we have that for any $i^* \in [\ell]$:

$$\left(\mathbf{u}_{k_j}\right)_{i^*} = (\mathbf{v}_j)_{i^*} \quad , \quad \left(\mathbf{u}_{k_j}\right)_{-i^*} = (\mathbf{v}_j)_{-i^*} \ .$$

We can emulate Exp_3 given $(\mathsf{ML.pk}_j, \mathsf{ML.vk}_j)$ and Z as input. Therefore, by the soundness of the multilinearity protocol (Definition 7.9):

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \ \wedge \ \forall \beta_0, \beta_1 \in \mathbb{F} : \ \exists \alpha^* \in \left\{\alpha_{j,k_j}, \tilde{\alpha}_j\right\} : \ \left\langle \beta_1 \cdot (\mathbf{v}_j)_{-i^*} + \beta_0 \right\rangle_{\mathsf{pp}} \neq \alpha^*\right] \leq K' \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $K' \leq \mathrm{poly}(\Lambda)$:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \ \wedge \ \forall \beta_0, \beta_1 \in \mathbb{F} : \ \exists \alpha^* \in \left\{\alpha_{j,k_j}, \tilde{\alpha}_j\right\} : \ \left\langle \beta_1 \cdot (\mathbf{v}_j)_{-i^*} + \beta_0 \right\rangle_{\mathsf{pp}} \neq \alpha^*\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

The claim follows since:

$$\alpha_{j,k_j} \neq \tilde{\alpha}_j \quad \rightarrow \quad \forall \beta_0, \beta_1 \in \mathbb{F} : \ \exists \alpha^* \in \left\{\alpha_{j,k_j}, \tilde{\alpha}_j\right\} : \ \left\langle \beta_1 \cdot (\mathbf{v}_j)_{-i^*} + \beta_0 \right\rangle_{\mathsf{pp}} \neq \alpha^* \ .$$

□

Non-signaling. Fix any $\mathrm{poly}(\Lambda)$-size distinguisher D. We show that for every $\kappa \in \mathbb{N}$, M-variate formula φ such that $M \leq \bar{M}$ and sets $\mathbf{S}' \subseteq \mathbf{S} \subseteq [M]$ of size at most K:

$$\left| \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S})}\left[\mathsf{D}(x, \sigma(\mathbf{S}')) = 1\right] - \Pr_{(x,\sigma') \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S}')}\left[\mathsf{D}(x, \sigma') = 1\right] \right| \leq \mathrm{negl}(\Lambda(\kappa)) \ . \tag{23}$$

Let Exp_0 be the experiment that is defined just like the execution of $\mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S}')$ except that instead of setting $\sigma(\mathbf{u}_k)$ for every $\mathbf{u}_k \in \mathbf{S}$, set $\sigma(\mathbf{u}_k)$ only for $\mathbf{u}_k \in \mathbf{S}'$. Let E_1 be the oracle machine that is defined just like E_0 except that for every $k \in [K]$ such that $\mathbf{u}_k \in \mathbf{S} \setminus \mathbf{S}'$instead of setting $c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, \mathbf{u}_k, \mathbf{r}_k)$ it sets $c_k \leftarrow \mathsf{Enc}(\mathsf{sk}_k, 0^\ell, \mathbf{r}_k)$.

E_0 and E_1 only differ in the values encrypted in c_k for $k \in [K]$ such that $\mathbf{u}_k \in \mathbf{S} \setminus \mathbf{S}'$. Also, we can emulate E_1 given c_k as input. Therefore, by semantic security (Definition 6.5):

$$\left| \Pr_{(x,\sigma)\leftarrow \mathsf{E}_0} [\mathsf{D}(x,\sigma) = 1] - \Pr_{(x,\sigma)\leftarrow \mathsf{E}_1} [\mathsf{D}(x,\sigma) = 1] \right| \leq |\mathbf{S} \setminus \mathbf{S}'| \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $|\mathbf{S} \setminus \mathbf{S}'| \leq \bar{M} \leq \mathrm{poly}(\Lambda)$:

$$\left| \Pr_{(x,\sigma)\leftarrow \mathsf{E}_0} [\mathsf{D}(x,\sigma) = 1] - \Pr_{(x,\sigma)\leftarrow \mathsf{E}_1} [\mathsf{D}(x,\sigma) = 1] \right| \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

Observe that the distribution of $(\mathsf{pk}, \mathsf{vk})$ generated by E_0 is identical to the distribution of $(\mathsf{pk}, \mathsf{vk})$ generated by $\mathsf{E}(\varphi, \mathbf{S})$ and, therefore:

$$[(x,\sigma) \mid (x,\sigma) \leftarrow \mathsf{E}_0] \equiv [(x, \sigma(\mathbf{S}')) \mid (x,\sigma) \leftarrow \mathsf{E}(\varphi, \mathbf{S})] \ .$$

Recall that $\mathsf{E}(\varphi, \mathbf{S}')$ sets $\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{S}'|}$ to be the elements of $\mathbf{S}'$ in some arbitrary order and sets $\mathbf{v}_{|\mathbf{S}'|+1}, \ldots, \mathbf{v}_{K'} = 0^\ell$. Since the ciphertexts $c_1, \ldots, c_{K'}$ in both $\mathsf{E}(\varphi, \mathbf{S}')$ and E_1 encrypt the same values $\mathbf{v}_1, \ldots, \mathbf{v}_{K'}$ in a random order, it follows that the distribution of $(\mathsf{pk}, \mathsf{vk})$ generated by E_1 is identical to the distribution of $(\mathsf{pk}, \mathsf{vk})$ generated by $\mathsf{E}(\varphi, \mathbf{S}')$ and, therefore:

$$[(x,\sigma) \mid (x,\sigma) \leftarrow \mathsf{E}_1] \equiv [(x, \sigma(\mathbf{S}')) \mid (x,\sigma) \leftarrow \mathsf{E}(\varphi, \mathbf{S})] \ .$$

Equation (23) follows.

8.3.3 Unambiguity.

For every function $K(\kappa)$ we define a pair of PPT oracle machines $\mathsf{E}_1, \mathsf{E}_2$ just like the machine E defined in the non-signaling extraction proof in Section 8.3.2 except that:

- E_1 and E_2 make an oracle call to Adv and obtain x, Π, Π' instead of a single proof.
- E_1 continues just like E using the proof Π and E_2 continues using the proof Π'.
- If $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ accepts then for $k \in [K']$ such that $\mathbf{u}_k \in \mathbf{S} \cap \{0,1\}^\ell$, if $\mathsf{Dec}(\mathsf{sk}_i, C_{1,k}) \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$, instead of setting $\sigma(\mathbf{u}_k) = 0$, E_1 sets $\sigma(\mathbf{u}_k)$ as follows:
  - If $\mathsf{Dec}(\mathsf{sk}_i, C'_{1,k}) = \langle b \rangle_{\mathsf{pp}}$ for some $b \in \{0,1\}$ set $\sigma(\mathbf{u}_k) = 1 - b$.
  - If $\mathsf{Dec}(\mathsf{sk}_i, C'_{1,k}) \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ set $\sigma(\mathbf{u}_k) = 0$.

  If $\mathsf{Dec}(\mathsf{sk}_i, C'_{1,k}) \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$, instead of setting $\sigma(\mathbf{u}_k)$ to a the default value 0, E_2 sets $\sigma(\mathbf{u}_k)$ as follows:
  - If $\mathsf{Dec}(\mathsf{sk}_i, C_{1,k}) = \langle b \rangle_{\mathsf{pp}}$ for some $b \in \{0,1\}$ set $\sigma(\mathbf{u}_k) = 1 - b$.
  - If $\mathsf{Dec}(\mathsf{sk}_i, C_{1,k}) \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ set $\sigma(\mathbf{u}_k) = 1$.

Before proving that E_1 and E_2 satisfy the required properties we start by introducing some notation.

The experiment $\mathsf{Exp}(\mathbf{S})$. For any $\kappa \in \mathbb{N}$ fix an M-variate formula φ such that $K \leq M \leq \bar{M}$. For a set $\mathbf{S} \subseteq \mathbb{F}^\ell$ of size at most K' let $\mathsf{Exp}(\mathbf{S})$ be the experiment where we execute $\mathsf{E}_1^{\mathsf{Adv}}(\varphi, \mathbf{S})$ and $\mathsf{E}_2^{\mathsf{Adv}}(\varphi, \mathbf{S})$ with the same randomness r. We note that:

1. In general, a non-signaling extractor with locality K is only defined on subsets of $\{0,1\}^\ell$ of size at most K. The construction of E_1 and E_1, however, supports subsets of $\mathbb{F}$ of size at most K'.
2. We assume without loss of generality that Adv is deterministic. Therefore, the two execution are identical up until they obtain output (x, Π, Π') from Adv.

In the experiment $\mathsf{Exp}(\mathbf{S})$ let:

$$\begin{aligned}
\mathsf{pk} &= \left(\varphi, \mathsf{pp}, \left(\mathsf{ML.pk}_j\right)_{j\in[3]}, \mathsf{SC.pk}, \tilde{c}\right) \ , \\
\mathsf{vk} &= \left(\mathsf{pp}, \left(\mathsf{ML.vk}_j\right)_{j\in[3]}, \mathsf{SC.vk}, e_0, e_1, \left(A_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(B_{\mathbf{b},i,b}\right)_{\mathbf{b}\in\{0,1\}^3, i\in[n], b\in\{0,1\}}\right) \ , \\
\Pi &= \left(\left(\mathbf{C}_j = \left(C_{j,1}, \ldots, C_{j,K'}, \tilde{C}_j\right), \mathsf{ML}.\Pi_j\right)_{j\in[3]}, \left(D_{\mathbf{b}}, \mathsf{SC}.\Pi_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(E_{\mathbf{b},J}\right)_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]}\right) \ , \\
\Pi' &= \left(\left(\mathbf{C}'_j = \left(C'_{j,1}, \ldots, C'_{j,K'}, \tilde{C}'_j\right), \mathsf{ML}.\Pi'_j\right)_{j\in[3]}, \left(D'_{\mathbf{b}}, \mathsf{SC}.\Pi'_{\mathbf{b}}\right)_{\mathbf{b}\in\{0,1\}^3}, \left(E'_{\mathbf{b},J}\right)_{\mathbf{b}\in\{0,1\}^3, J\subseteq[3]}\right) \ ,
\end{aligned}$$

Let:

$$\begin{array}{lllll}
\alpha_k = \mathsf{Dec}(\mathsf{sk}_k, C_{1,k}) & , & \alpha'_k = \mathsf{Dec}(\mathsf{sk}_k, C'_{1,k}) & : & k \in [K'] \ , \\
\xi_{\mathbf{b}} = \mathsf{Dec}(\tilde{\mathsf{sk}}, D_{\mathbf{b}}) & , & \xi'_{\mathbf{b}} = \mathsf{Dec}(\tilde{\mathsf{sk}}, D'_{\mathbf{b}}) & : & \mathbf{b} \in \{0,1\}^3 \ , \\
\zeta_{\mathbf{b},J} = \mathsf{Dec}(\tilde{\mathsf{sk}}, E_{\mathbf{b},J}) & , & \zeta'_{\mathbf{b},J} = \mathsf{Dec}(\tilde{\mathsf{sk}}, E'_{\mathbf{b},J}) & : & \mathbf{b} \in \{0,1\}^3, J \subseteq [3] \ .
\end{array}$$

For $\mathbf{v} \in \mathbf{S}$ let $k(\mathbf{v}) \in [K']$ denote the index such that $\mathbf{u}_{k(\mathbf{v})} = \mathbf{v}$.

The non-signaling property. The following claim generalizes the non-signaling property (Definition 8.2).

Claim 8.8. *For every* $\mathrm{poly}(\Lambda)$*-size distinguisher* D *there exists a negligible function* μ *such that for every sets* $\mathbf{S}' \subseteq \mathbf{S} \subseteq \mathbb{F}^\ell$ *of size at most* K'*:*

$$\left|\Pr_{\mathsf{Exp}(\mathbf{S})}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right] - \Pr_{\mathsf{Exp}(\mathbf{S}')}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right]\right| \leq \mu(\Lambda(\kappa)) \ .$$

Proof. Fix any $\mathrm{poly}(\Lambda)$-size distinguisher D and sets $\mathbf{S}' \subseteq \mathbf{S} \subseteq \mathbb{F}^\ell$ of size at most K'.

Let $\mathsf{Exp}'(\mathbf{S})$ be the experiment that is defined just like $\mathsf{Exp}(\mathbf{S})$ except that for every $\mathbf{v} \in \mathbf{S} \setminus \mathbf{S}'$, instead of setting $c_{k(\mathbf{v})} \leftarrow \mathsf{Enc}(\mathsf{sk}_{k(\mathbf{v})}, \mathbf{u}_{k(\mathbf{v})}, \mathbf{r}_{k(\mathbf{v})})$ we set $c_{k(\mathbf{v})} \leftarrow \mathsf{Enc}(\mathsf{sk}_{k(\mathbf{v})}, 0^\ell, \mathbf{r}_{k(\mathbf{v})})$. $\mathsf{Exp}(\mathbf{S})$ and $\mathsf{Exp}'(\mathbf{S})$ only differ in the values encrypted in $c_{k(\mathbf{v})}$ for $\mathbf{v} \in \mathbf{S} \setminus \mathbf{S}'$. Also, we can emulate $\mathsf{Exp}'(\mathbf{S})$ and decrypt the values $\left(\alpha_{k(\mathbf{u})}, \alpha'_{k(\mathbf{u})}\right)_{\mathbf{u}\in[\mathbf{S}']}$ given $c_{k(\mathbf{v})}$ for $\mathbf{v} \in \mathbf{S} \setminus \mathbf{S}'$ as input. Therefore, by semantic security (Definition 6.5):

$$\left|\Pr_{\mathsf{Exp}(\mathbf{S})}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right] - \Pr_{\mathsf{Exp}'(\mathbf{S})}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right]\right| \leq |\mathbf{S} \setminus \mathbf{S}'| \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $|\mathbf{S} \setminus \mathbf{S}'| \leq \bar{M} \leq \mathrm{poly}(\Lambda)$:

$$\left|\Pr_{\mathsf{Exp}(\mathbf{S})}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right] - \Pr_{\mathsf{Exp}'(\mathbf{S})}\left[\mathsf{D}\left(\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']}\right) = 1\right]\right| \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

Recall that in $\mathsf{Exp}(\mathbf{S}')$ we set $\mathbf{v}_1, \ldots, \mathbf{v}_{|\mathbf{S}'|}$ to be the elements of $\mathbf{S}'$ in some arbitrary order and sets $\mathbf{v}_{|\mathbf{S}'|+1}, \ldots, \mathbf{v}_{K'} = 0^\ell$. Since the ciphertexts $c_1, \ldots, c_{K'}$ in both $\mathsf{Exp}(\mathbf{S}')$ and $\mathsf{Exp}'(\mathbf{S})$ encrypt the same values $\mathbf{v}_1, \ldots, \mathbf{v}_{K'}$ in a random order, it follows that:

$$\left[\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']} \;\middle|\; \mathsf{Exp}'(\mathbf{S})\right] \equiv \left[\left(\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})}\right)_{\mathbf{v}\in[\mathbf{S}']} \;\middle|\; \mathsf{Exp}(\mathbf{S}')\right] \ .$$

The claim follows. □

The events AC **and** EQ**.** For a set $\mathbf{S} \subseteq \mathbb{F}^\ell$ of size at most K' let AC be the event that in the experiment $\mathsf{Exp}(\mathbf{S})$, Adv outputs (x, Π, Π') such that both $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ and $\mathsf{QA.V}(\mathsf{vk}, x, \Pi')$ accept. Since E_1 outputs $x_1 = \bot$ when $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ rejects and E_2 outputs $x_2 = \bot$ when $\mathsf{QA.V}(\mathsf{vk}, x, \Pi')$ rejects, it follows that:

$$\Pr_{\mathsf{Exp}(\mathbf{S})}\left[\mathsf{AC} \leftrightarrow x_1 = x_2 \neq \bot\right] \ . \tag{24}$$

For $\mathbf{v} \in \mathbf{S}$ let $\mathsf{EQ}_{\mathbf{v}}$ be the event that in the experiment $\mathsf{Exp}(\mathbf{S})$, $\alpha_{k(\mathbf{v})} = \alpha'_{k(\mathbf{v})}$. For $\mathbf{S}' \subseteq \mathbf{S}$ let $\mathsf{EQ}(\mathbf{S}')$ be the event that $\mathsf{EQ}_{\mathbf{u}}$ occurs for every $\mathbf{u} \in \mathbf{S}'$. For $j \in [2]$, let σ_j denote the assignment generated by E_j.

Claim 8.9. *For any set $\mathbf{S} \subseteq \mathbb{F}^\ell$ of size at most K' and $\mathbf{v} \in \mathbf{S} \cap \{0,1\}^\ell$:*

$$\Pr_{\mathsf{Exp}(\mathbf{S})}[\mathsf{AC} \wedge \neg\mathsf{EQ}_{\mathbf{v}} \rightarrow \sigma_1(\mathbf{v}) \neq \sigma_2(\mathbf{v})] = 1 \ .$$

Proof. Assume AC and $\mathsf{EQ}_{\mathbf{v}}$ occur. We consider the following cases:

- If $\alpha_{k(\mathbf{v})} = \langle b \rangle_{\mathsf{pp}}$ and $\alpha'_{k(\mathbf{v})} = \langle 1-b \rangle_{\mathsf{pp}}$ for $b \in \{0,1\}$ then $\sigma_1(\mathbf{v}) = b$ and $\sigma_2(\mathbf{v}) = 1-b$.
- If $\alpha_{k(\mathbf{v})} = \langle b \rangle_{\mathsf{pp}}$ for $b \in \{0,1\}$ and $\alpha'_{k(\mathbf{v})} \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ then $\sigma_1(\mathbf{v}) = b$ and $\sigma_2(\mathbf{v}) = 1-b$.
- If $\alpha_{k(\mathbf{v})} \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ and $\alpha'_{k(\mathbf{v})} = \langle b \rangle_{\mathsf{pp}}$ for $b \in \{0,1\}$ then $\sigma_1(\mathbf{v}) = 1-b$ and $\sigma_2(\mathbf{v}) = b$.
- If $\alpha_{k(\mathbf{v})}, \alpha'_{k(\mathbf{v})} \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ then $\sigma_1(\mathbf{v}) = 0$ and $\sigma_2(\mathbf{v}) = 1$.

In any case $\sigma_1(\mathbf{v}) \neq \sigma_2(\mathbf{v})$. □

The experiment $\mathsf{Exp}'(\mathbf{S})$. For a string $\mathbf{u} \in \{0,1\}^{\leq \ell}$ and a vector $\mathbf{t} = (t_1, \ldots, t_\ell) \in \ell$ let $[\mathbf{u}]_{\mathbf{t}}$ denote the string $(\mathbf{u} \mid (t_{|\mathbf{u}|+1}, \ldots, t_\ell)) \in \mathbb{F}^\ell$. For a set $\mathbf{S} \subseteq \mathbb{F}^{\leq \ell}$ let $[\mathbf{S}]_{\mathbf{t}}$ denote the set $\{[\mathbf{u}]_{\mathbf{t}} : \mathbf{u} \in \mathbf{S}\}$. For a set $\mathbf{S} \subseteq \mathbb{F}^{\leq \ell}$ of size at most K' let $\mathsf{Exp}'(\mathbf{S})$ be the experiment $\mathsf{Exp}([\mathbf{S}]_{\mathbf{t}})$ for a random $\mathbf{t} \leftarrow \mathbb{F}^\ell$. For $\mathbf{u} \in \mathbf{S}$ let $\mathsf{EQ}_{\mathbf{u}}$ be the event that in the experiment $\mathsf{Exp}'(\mathbf{S})$, the event $\mathsf{EQ}_{[\mathbf{u}]_{\mathbf{t}}}$ occurs. For $\mathbf{S}' \subseteq \mathbf{S}$ let $\mathsf{EQ}(\mathbf{S}')$ be the event that in the experiment $\mathsf{Exp}'(\mathbf{S})$, the event $\mathsf{EQ}([\mathbf{S}']_{\mathbf{t}})$ occurs.

The sets $\mathbf{S}_{\leq \mathbf{v}}$ and $\mathbf{Q}_{\leq \mathbf{v}}$. For any $\kappa \in \mathbb{N}$ fix sets $\mathbf{S}_1, \ldots, \mathbf{S}_B$ such that $|\mathbf{S}_i| \leq K$ and $\bigcup_{i \in [B]} \mathbf{S}_i = \{0,1\}^\ell$. We assume without loss of generality that the sets $\mathbf{S}_1, \ldots, \mathbf{S}_B$ are pairwise disjoint. For every $\mathbf{v} \in \{0,1\}^\ell$ let $\mathbf{S}_{\mathbf{v}}$ denote the unique set $\mathbf{S}_i$ that contains $\mathbf{v}$. Let $\mathbf{S}_{<\mathbf{v}}$ denote the set $\{\mathbf{u} : \mathbf{u} \in \mathbf{S}_{\mathbf{v}} \wedge \mathbf{u} < \mathbf{v}\}$ and let $\mathbf{S}_{\leq \mathbf{v}}$ denote the set $\mathbf{S}_{\leq \mathbf{v}} \cup \{\mathbf{v}\}$.

For every $\mathbf{v} \in \{0,1\}^\ell$ we define a set $\mathbf{Q}_{\leq \mathbf{v}} \subseteq \{0,1\}^{\leq \ell}$. Let $\mathbf{Q}_{\leq 0^\ell} = \{0^\ell\}$ and for every $\mathbf{v} > 0^\ell$ we compute $\mathbf{Q}_{\leq \mathbf{v}}$ as follows:

- Set $\mathbf{Q}_{\leq \mathbf{v}} \leftarrow \mathbf{Q}_{\leq \mathbf{v}-1} \cup \{\mathbf{v}\}$.
- While there exists $\mathbf{u} \in \{0,1\}^{<\ell}$ such that $\{\mathbf{u}0, \mathbf{u}1\} \subseteq \mathbf{Q}_{\leq \mathbf{v}}$, remove $\mathbf{u}0$ and $\mathbf{u}1$ from $\mathbf{Q}_{\leq \mathbf{v}}$ and add $\mathbf{u}$.

Let $\mathbf{Q}_{<0^\ell} = \emptyset$ and for every $\mathbf{v} > 0^\ell$ let $\mathbf{Q}_{<\mathbf{v}} = \mathbf{Q}_{\leq \mathbf{v}-1}$. The following claim give some useful properties of $\mathbf{Q}_{\leq \mathbf{v}}$:

Claim 8.10. *For every $\mathbf{v} \in \{0,1\}^\ell$:*

- $|\mathbf{Q}_{\leq \mathbf{v}}| \leq \ell$.
- *For every $\mathbf{u} \leq \mathbf{v}$, $\mathbf{Q}_{\leq \mathbf{v}}$ contains a prefix of $\mathbf{u}$.*
- $\mathbf{Q}_{\leq 1^\ell} = \{\mathcal{E}\}$

The proof of the unambiguity property relies on the following claims.

Claim 8.11.

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell}) \rightarrow \Pi = \Pi'\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Claim 8.12. *For every $\mathbf{v} \in \{0,1\}^\ell$:*

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}} \cup \{\mathbf{v}\})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \wedge \mathsf{EQ}_{\mathbf{v}} \rightarrow \mathsf{EQ}(\mathbf{Q}_{\leq \mathbf{v}})] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Claim 8.13. *For every $\mathbf{v} \in \{0,1\}^\ell$:*

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{\leq \mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{\leq \mathbf{v}}) \rightarrow \mathsf{EQ}(\mathbf{S}_{\leq \mathbf{v}})] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving the claims we use them to prove that E_1 and E_2 satisfy the required properties. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv. It follows from the analysis in Section 8.3.2 that E_1 and E_2 are Λ-non-signaling extractors with formula size bound $\bar{M}$, input length n and locality K (Definition 8.2). Note that in the experiment $\mathsf{Exp}(\mathbf{S})$, for $\mathbf{v} \in \mathbf{S} \cap \{0,1\}^\ell$, if $\alpha_{k(\mathbf{v})} \notin \left\{ \langle 0 \rangle_{\mathsf{pp}}, \langle 1 \rangle_{\mathsf{pp}} \right\}$ then E_1 sets $\sigma(\mathbf{v})$ differently than the extractor E in Section 8.3.2 (and similarly for $\alpha'_{k(\mathbf{v})}$ and E_2). The proof of the non-signaling extraction property in Section 8.3.2, however, only relies in the fact that if $\alpha_{k(\mathbf{v})} = \langle b \rangle_{\mathsf{pp}}$ for $b \in \{0,1\}$ then $\sigma(\mathbf{v}) = b$.

It remains to show that for every $\kappa \in \mathbb{N}$:

$$\Pr\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi' \ \middle| \ \begin{array}{l} (\mathsf{pk},\mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa,\varphi,K) \\ (x,\Pi,\Pi') \leftarrow \mathsf{Adv}(\mathsf{pk},\mathsf{vk}) \end{array}\right] \leq \sum_{i\in[B]} \Pr_{\mathsf{Exp}'(\mathbf{S}_i)}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{25}$$

Observe that in the experiment $\mathsf{Exp}'(\emptyset)$ the distribution of $(\mathsf{pk},\mathsf{vk})$ is identical to the output distribution of the setup algorithm $\mathsf{QA.S}$. Therefore:

$$\Pr\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi' \ \middle| \ \begin{array}{l} (\mathsf{pk},\mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa,\varphi,K) \\ (x,\Pi,\Pi') \leftarrow \mathsf{Adv}(\mathsf{pk},\mathsf{vk}) \end{array}\right] = \Pr_{\mathsf{Exp}'(\emptyset)}\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi'\right] \ . \tag{26}$$

By Claim 8.11:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi'\right] \leq \Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell})\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\emptyset)}\left[\mathsf{AC} \ \wedge \ \Pi \neq \Pi'\right] \leq \Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell})\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{27}$$

By Claim 8.12 for every $\mathbf{v} \in \{0,1\}^\ell, \mathbf{v} > 0^\ell$:

$$\begin{aligned}\Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}} \cup \{\mathbf{v}\})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{\leq \mathbf{v}})\right] \leq & \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}} \cup \{\mathbf{v}\})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] \\ & + \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}} \cup \{\mathbf{v}\})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}})\right] + \mathrm{negl}(\Lambda(\kappa)) \ .\end{aligned}$$

By Claim 8.8:

$$\begin{aligned}\Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq \mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{\leq \mathbf{v}})\right] \leq & \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \{\mathbf{v}\})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] \\ & + \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}})\right] + \mathrm{negl}(\Lambda(\kappa)) \ .\end{aligned}$$

Therefore:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \ \wedge \ \neg\mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell})\right] \leq \sum_{\mathbf{v}\in\{0,1\}^\ell} \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}} \cup \{\mathbf{v}\})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] + 2^\ell \cdot \mathrm{negl}(\Lambda(\kappa)) \ . \tag{28}$$

By Claim 8.13 for every $\mathbf{v} \in \{0,1\}^\ell$:

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{<\mathbf{v}} \cup \mathbf{Q}_{<\mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \rightarrow \ \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}})\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{\leq \mathbf{v}} \cup \mathbf{Q}_{<\mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}} \ \rightarrow \ \mathsf{EQ}(\mathbf{S}_{\leq \mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Therefore:

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{\leq \mathbf{v}} \cup \mathbf{Q}_{<\mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] \leq \Pr_{\mathsf{Exp}'(\mathbf{S}_{\leq \mathbf{v}} \cup \mathbf{Q}_{<\mathbf{v}})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \ \wedge \ \neg\mathsf{EQ}_{\mathbf{v}}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}}\cup\{\mathbf{v}\})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] \leq \Pr_{\mathsf{Exp}'(\mathbf{S}_{\mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{29}$$

By Equation (24) and Claim 8.9:

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{\mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}} \rightarrow x_1 = x_2 \neq \bot \wedge \sigma_1 \neq \sigma_2] = 1 \ .$$

Since for every $i \in [B]$ and $\mathbf{v}, \mathbf{v}' \in \mathbf{S}_i$ the events $(\mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}})$ and $(\mathsf{EQ}(\mathbf{S}_{<\mathbf{v}'}) \wedge \neg\mathsf{EQ}_{\mathbf{v}'})$ are disjoint:

$$\sum_{\mathbf{v}\in\mathbf{S}_i} \Pr_{\mathsf{Exp}'(\mathbf{S}_i)}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] \leq \Pr_{\mathsf{Exp}'(\mathbf{S}_i)}[x_1 = x_2 \neq \bot \wedge \sigma_1 \neq \sigma_2] \ .$$

Since $\bigcup_{i\in[B]} \mathbf{S}_i = \{0,1\}^\ell$:

$$\sum_{\mathbf{v}\in\{0,1\}^\ell} \Pr_{\mathsf{Exp}'(\mathbf{S}_{\mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] \leq \sum_{i\in[B]} \Pr_{\mathsf{Exp}'(\mathbf{S}_i)}[x_1 = x_2 \neq \bot \wedge \sigma_1 \neq \sigma_2] \ . \tag{30}$$

Putting together Equations (26) to (30) we get:

$$\begin{aligned}
&\Pr\left[\mathsf{AC} \wedge \Pi \neq \Pi' \;\middle|\; \begin{array}{l}(\mathsf{pk},\mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa,\varphi,K)\\ (x,\Pi,\Pi') \leftarrow \mathsf{Adv}(\mathsf{pk},\mathsf{vk})\end{array}\right]\\
&\leq \Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}[\mathsf{AC} \wedge \Pi \neq \Pi'] + \mathrm{negl}(\Lambda(\kappa))\\
&\leq \Pr_{\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})}\left[\mathsf{AC} \wedge \neg\mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell})\right] + \mathrm{negl}(\Lambda(\kappa))\\
&\leq \sum_{\mathbf{v}\in\{0,1\}^\ell} \Pr_{\mathsf{Exp}'(\mathbf{Q}_{<\mathbf{v}}\cup\{\mathbf{v}\})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] + 2^\ell \cdot \mathrm{negl}(\Lambda(\kappa))\\
&\leq \sum_{\mathbf{v}\in\{0,1\}^\ell} \Pr_{\mathsf{Exp}'(\mathbf{S}_{\mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{S}_{<\mathbf{v}}) \wedge \neg\mathsf{EQ}_{\mathbf{v}}] + 2^\ell \cdot \mathrm{negl}(\Lambda(\kappa))\\
&\leq \sum_{i\in[B]} \Pr_{\mathsf{Exp}'(\mathbf{S}_i)}[x_1 = x_2 \neq \bot \wedge \sigma_1 \neq \sigma_2] + 2^\ell \cdot \mathrm{negl}(\Lambda(\kappa)) \ .
\end{aligned}$$

Since $2^\ell = M \leq \bar{M} \leq \mathrm{poly}(\Lambda)$, Equation (25) follows. It remains to prove Claims 8.11 to 8.13.

Proof of Claim 8.11. Let Exp_1 be the experiment that is define just like $\mathsf{Exp}'(\mathbf{Q}_{\leq 1^\ell})$ except that instead of setting $\tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^\ell, \tilde{\mathbf{r}}_j)$ we set:

$$\tilde{c}_1 = \mathsf{Restrict}(\tilde{c}, [\ell+1, 3\ell]) \quad , \quad \tilde{c}_2 = \mathsf{Restrict}(\tilde{c}, [\ell] \cup [2\ell+1, 3\ell]) \quad , \quad \tilde{c}_3 = \mathsf{Restrict}(\tilde{c}, [2\ell]) \ .$$

By the correctness of restriction (Definition 6.10) it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell}) \rightarrow \Pi = \Pi'\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Exp_2 be the experiment that is defined just like Exp_1 except that instead of setting $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, 0^{3\ell}, (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$ we sample $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \leftarrow \mathbb{F}^\ell$ and set $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, (\mathbf{z}_1 \mid \mathbf{z}_2 \mid \mathbf{z}_3), (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$. We can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by semantic security (Definition 6.5) and since $M \leq \bar{M} \leq \mathrm{poly}(\Lambda)$ it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_{\leq 1^\ell}) \rightarrow \Pi = \Pi'\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.10, $\mathbf{Q}_{\leq 1^\ell} = \{\mathcal{E}\}$. Therefore, we need to prove that:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{t}} \rightarrow \Pi = \Pi'\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{31}$$

We rely on the following claim.

Claim 8.14. *For every $j \in [3]$:*

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \rightarrow (\mathbf{C}_j, \mathsf{ML}.\Pi_j) = (\mathbf{C}'_j, \mathsf{ML}.\Pi'_j)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving Claim 8.14 we use it to prove Equation (31). If AC occurs:

- $C_{k(\mathbf{t})}, C'_{k(\mathbf{t})} \in \mathsf{Valid}$. Also, we can emulate the experiment Exp_2 given $c_{k(\mathbf{t})}$ as input. Therefore, by the unambiguity of ciphertexts (Definition 6.6):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{t}} \rightarrow C_{1,k(\mathbf{t})} = C'_{1,k(\mathbf{t})}\right] \geq 1 - \frac{K' \cdot \mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \ .$$

  Since $\bar{\delta} = 2$, $K', \bar{\ell} = 3 \log \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{t}} \rightarrow C_{1,k(\mathbf{t})} = C'_{1,k(\mathbf{t})}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- Since $C_{1,k} = C_{2,k} = C_{3,k}$ and $C'_{1,k} = C'_{2,k} = C'_{3,k}$ we have that for every $j \in [3]$:

$$C_{1,k(\mathbf{t})} = C'_{1,k(\mathbf{t})} \Rightarrow C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \ .$$

- By Claim 8.14:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \rightarrow (\mathbf{C}_j, \mathsf{ML}.\Pi_j) = (\mathbf{C}'_j, \mathsf{ML}.\Pi'_j)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $j \in [3]$ the following zero-tests pass:

$$\left[\left(E_{\mathbf{b},\{j\}} + e_{b_j}\right) \cdot e_1 = \tilde{C}_j \cdot e_1\right]_{\mathsf{pp}} \quad , \quad \left[\left(E'_{\mathbf{b},\{j\}} + e_{b_j}\right) \cdot e_1 = \tilde{C}'_j \cdot e_1\right]_{\mathsf{pp}} \ .$$

  By construction $\mathsf{Dec}(\tilde{\mathsf{sk}}, e_1) \neq \langle 0 \rangle_{\mathsf{pp}}$ and thus, by the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \tilde{C}_j = \tilde{C}'_j \rightarrow \zeta_{\mathbf{b},\{j\}} = \zeta'_{\mathbf{b},\{j\}}\right] = 1 \ .$$

  $E_{\mathbf{b},\{j\}}, E'_{\mathbf{b},\{j\}} \in \mathsf{Valid}$. Also, we can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by the unambiguity of ciphertexts (Definition 6.6):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \zeta_{\mathbf{b},\{j\}} = \zeta'_{\mathbf{b},\{j\}} \rightarrow E_{\mathbf{b},\{j\}} = E'_{\mathbf{b},\{j\}}\right] \geq 1 - \frac{\mathrm{poly}(\bar{\delta} \cdot \bar{\ell})}{|\mathbb{F}|} \ .$$

  Since $\bar{\delta} = 2$, $\bar{\ell} = 3 \log \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \zeta_{\mathbf{b},\{j\}} = \zeta'_{\mathbf{b},\{j\}} \rightarrow E_{\mathbf{b},\{j\}} = E'_{\mathbf{b},\{j\}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ and pair of non-empty disjoint subsets $J, J' \subseteq [3]$ the following zero-tests pass:

$$\left[E_{\mathbf{b},J} \cdot E_{\mathbf{b},J'} = E_{\mathbf{b},J \cup J'} \cdot e_1\right]_{\mathsf{pp}} \ .$$

  By construction $\mathsf{Dec}(\tilde{\mathsf{sk}}, e_1) \neq \langle 0 \rangle_{\mathsf{pp}}$ and thus, by the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge E_{\mathbf{b},J} = E'_{\mathbf{b},J} \wedge E_{\mathbf{b},J'} = E'_{\mathbf{b},J'} \rightarrow \zeta_{\mathbf{b},J \cup J'} = \zeta'_{\mathbf{b},J \cup J'}\right] = 1 \ .$$

$E_{\mathbf{b},J\cup J'}, E'_{\mathbf{b},J\cup J'} \in \mathsf{Valid}$. Also, we can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by the unambiguity of ciphertexts (Definition 6.6):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \zeta_{\mathbf{b},J\cup J'} = \zeta'_{\mathbf{b},J\cup J'} \rightarrow E_{\mathbf{b},J\cup J'} = E'_{\mathbf{b},J\cup J'}\right] = 1 - \frac{\mathrm{poly}(\bar{\delta}\cdot\bar{\ell})}{|\mathbb{F}|} \ .$$

Since $\bar{\delta} = 2$, $\bar{\ell} = 3\log \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}_1}\left[\mathsf{AC} \wedge \zeta_{\mathbf{b},J\cup J'} = \zeta'_{\mathbf{b},J\cup J'} \rightarrow E_{\mathbf{b},J\cup J'} = E'_{\mathbf{b},J\cup J'}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$ the following zero-tests pass:

$$\left[D_{\mathbf{b}}\cdot e_1 = \left(A_{\mathbf{b}} + \sum_{i\in[n]} B_{\mathbf{b},i,x_i}\right)\cdot E_{\mathbf{b},[3]}\right]_{\mathsf{pp}} \ .$$

By construction $\mathsf{Dec}(\tilde{\mathsf{sk}}, e_1) \neq \langle 0\rangle_{\mathsf{pp}}$ and thus, by the soundness of the zero-test (Definition 6.8):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge E_{\mathbf{b},[3]} = E'_{\mathbf{b},[3]} \rightarrow \xi_{\mathbf{b}} = \xi'_{\mathbf{b}}\right] = 1 \ .$$

$E_{\mathbf{b},[3]}, E'_{\mathbf{b},[3]} \in \mathsf{Valid}$. Also, we can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by the unambiguity of ciphertexts (Definition 6.6):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \xi_{\mathbf{b},[3]} = \xi'_{\mathbf{b},[3]} \rightarrow D_{\mathbf{b}} = D'_{\mathbf{b}}\right] = 1 - \frac{\mathrm{poly}(\bar{\delta}\cdot\bar{\ell})}{|\mathbb{F}|} \ .$$

Since $\bar{\delta} = 2$, $\bar{\ell} = 3\log \bar{M} \leq \mathrm{poly}(\Lambda)$ and $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge \xi_{\mathbf{b},[3]} = \xi'_{\mathbf{b},[3]} \rightarrow D_{\mathbf{b}} = D'_{\mathbf{b}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- For every $\mathbf{b} \in \{0,1\}^3$, $\mathsf{SC.V}(\mathsf{SC.vk}, D_{\mathbf{b}}, \mathsf{SC.}\Pi_{\mathbf{b}})$ and $\mathsf{SC.V}(\mathsf{SC.vk}, D'_{\mathbf{b}}, \mathsf{SC.}\Pi'_{\mathbf{b}})$ both accepts. We can emulate Exp_2 given $\tilde{c}$ as input. Therefore, by the unambiguity of the sum-check protocol (Definition 7.13):

$$\Pr_{\mathsf{Exp}_2}\left[\mathsf{AC} \wedge D_{\mathbf{b}} = D'_{\mathbf{b}} \rightarrow \mathsf{SC.}\Pi_{\mathbf{b}} = \mathsf{SC.}\Pi'_{\mathbf{b}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Equation (31) follows. It remains to prove Claim 8.14.

Proof of Claim 8.14. Fix any $j \in [3]$. Let Exp_3 be the experiment that is define just like Exp_2 except that instead of setting:

$$\tilde{c}_j = \mathsf{Restrict}(\tilde{c}, [(j-1)\cdot\ell] \cup [j\cdot\ell + 1, 3\ell]) \ ,$$

we set $\tilde{c}_j \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, \mathbf{z}_j, \tilde{\mathbf{r}}_j)$. By the correctness of restriction (Definition 6.10) it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_3}\left[\mathsf{AC} \wedge C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \rightarrow (\mathbf{C}_j, \mathsf{ML.}\Pi_j) = (\mathbf{C'}_j, \mathsf{ML.}\Pi'_j)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Exp_4 be the experiment that is defined just like Exp_3 except that instead of setting $\tilde{c} \leftarrow \mathsf{Enc}(\tilde{\mathsf{sk}}, \mathbf{z}, (\tilde{\mathbf{r}}_1 \mid \tilde{\mathbf{r}}_2 \mid \tilde{\mathbf{r}}_3))$ we set:

$$\tilde{c} = \mathsf{RandExtend}(\tilde{c}_j, [(j-1)\cdot\ell] \cup [j\cdot\ell + 1, 3\ell]) \ .$$

By the correctness of random extension (Definition 6.11) it is sufficient to prove that:

$$\Pr_{\mathsf{Exp}_4}\left[\mathsf{AC} \wedge C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \rightarrow (\mathbf{C}_j, \mathsf{ML.}\Pi_j) = (\mathbf{C'}_j, \mathsf{ML.}\Pi'_j)\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

We can emulate Exp_4 given $(\mathsf{ML.pk}_j, \mathsf{ML.vk}_j)$ and $(c_1, \ldots, c_{K'}, \tilde{c}_j)$ as input. Therefore, by the by the unambiguity of the multilinearity protocol (Definition 7.9):

$$\Pr_{\mathsf{Exp}_4}\left[\mathsf{AC} \wedge C_{j,k(\mathbf{t})} = C'_{j,k(\mathbf{t})} \rightarrow (\mathbf{C}_j, \mathsf{ML.}\Pi_j) = (\mathbf{C'}_j, \mathsf{ML.}\Pi'_j)\right] \geq 1 - K' \cdot \mathrm{negl}(\Lambda(\kappa)) \ . \tag{32}$$

Since $K' \leq \bar{M} \leq \mathrm{poly}(\Lambda)$, Equation (32), concluding the proof of the claim. □

□

Proof of Claim 8.12. We rely on the following claim.

Claim 8.15. *For every* $\mathbf{u} \in \{0,1\}^{<\ell}$*:*

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{u}0,\mathbf{u}1\})}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{u}0} \wedge \mathsf{EQ}_{\mathbf{u}1} \rightarrow \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Before proving Claim 8.15 we use it to prove Claim 8.12.

Fix any $\mathbf{v} \in \{0,1\}^{\ell}$. By the definition of the set $\mathbf{Q}_{\leq \mathbf{v}}$ there exists a sequence of sets $\mathbf{Q}_0, \ldots, \mathbf{Q}_{\ell'}$ such that $\mathbf{Q}_0 = \mathbf{Q}_{<\mathbf{v}} \cup \mathbf{v}$, $\mathbf{Q}_{\ell'} = \mathbf{Q}_{\leq \mathbf{v}}$ and for every $i \in [\ell']$ there exists $\mathbf{u}_i \in \mathbb{F}^{<\ell}$ such that:

$$\mathbf{u}_i 0, \mathbf{u}_i 1 \in \mathbf{Q}_{i-1} \quad , \quad \mathbf{Q}_i = (\mathbf{Q}_{i-1} \setminus \{\mathbf{u}_i 0, \mathbf{u}_i 1\}) \cup \{\mathbf{u}_i\} \ .$$

By Claim 8.10, $|\mathbf{Q}_{<\mathbf{v}}| \leq \ell$ and, hence, $\ell' \leq \ell$. Since $\ell \leq \bar{M} \leq \mathrm{poly}(\Lambda)$, it is sufficient to prove that for every $i \in [0, \ell']$:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_0 \cup \mathbf{Q}_i)}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_0) \rightarrow \mathsf{EQ}(\mathbf{Q}_i)] \geq 1 - i \cdot \mathrm{negl}(\Lambda(\kappa)) \ . \tag{33}$$

We prove Equation (33) by induction on i. For $i = 0$ Equation (33) holds trivially. Assume that Equation (33) holds for $i - 1$:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_0 \cup \mathbf{Q}_{i-1})}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_0) \rightarrow \mathsf{EQ}(\mathbf{Q}_{i-1})] \geq 1 - (i-1) \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_0 \cup \mathbf{Q}_{i-1} \cup \mathbf{Q}_i)}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_0) \rightarrow \mathsf{EQ}(\mathbf{Q}_{i-1})] \geq 1 - (i-1) \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Claim 8.15 and Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_0 \cup \mathbf{Q}_{i-1} \cup \mathbf{Q}_i)}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{u}_i 0} \wedge \mathsf{EQ}_{\mathbf{u}_i 1} \rightarrow \mathsf{EQ}_{\mathbf{u}_i}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\mathbf{u}_i 0, \mathbf{u}_i 1 \in \mathbf{Q}_{i-1}$ and $\mathbf{Q}_i \subseteq \mathbf{Q}_{i-1} \cup \{\mathbf{u}_i\}$:

$$\Pr_{\mathsf{Exp}'(\mathbf{Q}_0 \cup \mathbf{Q}_{i-1} \cup \mathbf{Q}_i)}[\mathsf{AC} \wedge \mathsf{EQ}(\mathbf{Q}_0) \rightarrow \mathsf{EQ}(\mathbf{Q}_{i-1}) \rightarrow \mathsf{EQ}(\mathbf{Q}_i)] \geq 1 - i \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Equation (33) follows by Claim 8.8 concluding the proof of the claim.

Proof of Claim 8.15. Let $i = |\mathbf{u}0|$. We have that:

$$([\mathbf{u}]_{\mathbf{t}})_i = \mathbf{t}_i \quad , \quad ([\mathbf{u}0]_{\mathbf{t}})_i = 0 \quad , \quad ([\mathbf{u}1]_{\mathbf{t}})_i = 1 \quad , \quad ([\mathbf{u}]_{\mathbf{t}})_{-i} = ([\mathbf{u}0]_{\mathbf{t}})_{-i} = ([\mathbf{u}1]_{\mathbf{t}})_{-i} \ .$$

In the experiment $\mathsf{Exp}'(\{\mathbf{u}, \mathbf{u}0, \mathbf{u}1\})$ let:

$$Z = \left(\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_{K'}, \tilde{\mathsf{sk}}) \ , \ \mathbf{c} = (c_1, \ldots, c_{K'}, \tilde{c}_1) \ , \ \mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_{K'}, \tilde{\mathbf{r}}_j)\right) \ .$$

Also, let $\beta_0, \beta_1, \beta'_0, \beta'_1 \in \mathbb{F}$ be the unique values such that:

$$\langle \beta_1 \cdot 0 + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{u}0]_{\mathbf{t}})} \quad , \quad \langle \beta_1 \cdot 1 + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{u}1]_{\mathbf{t}})} \ ,$$
$$\langle \beta'_1 \cdot 0 + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{u}0]_{\mathbf{t}})} \quad , \quad \langle \beta'_1 \cdot 1 + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{u}1]_{\mathbf{t}})} \ .$$

When AC occurs $\mathsf{ML.V}(\mathsf{ML.vk}_1, \mathbf{C}_1, \mathsf{ML.\Pi})$ and $\mathsf{ML.V}(\mathsf{ML.vk}_1, \mathbf{C}'_1, \mathsf{ML.\Pi}')$ both accept. Also, we can emulate $\mathsf{Exp}'(\{\mathbf{u}, \mathbf{u}0, \mathbf{u}1\})$ given $(\mathsf{ML.pk}_1, \mathsf{ML.vk}_1)$ and Z as input. Therefore, by the soundness of the multilinearity protocol (Definition 7.9):

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{u}0,\mathbf{u}1\})}\left[\mathsf{AC} \;\rightarrow\; \begin{matrix}\langle \beta_1 \cdot \mathbf{t}_i + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{u}]_{\mathbf{t}})} \\ \langle \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{u}]_{\mathbf{t}})}\end{matrix}\right] \geq 1 - {K'}^3 \mathrm{negl}(\Lambda(\kappa)) \enspace .$$

Since $K' \leq \mathrm{poly}(\Lambda)$:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{u}0,\mathbf{u}1\})}\left[\mathsf{AC} \;\rightarrow\; \begin{matrix}\langle \beta_1 \cdot \mathbf{t}_i + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{u}]_{\mathbf{t}})} \\ \langle \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{u}]_{\mathbf{t}})}\end{matrix}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \enspace .$$

When $\mathsf{EQ}_{\mathbf{u}0}$ and $\mathsf{EQ}_{\mathbf{u}1}$ both occur, we have that $(\beta_0, \beta_1) = (\beta'_0, \beta'_1)$ and, therefore:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{u}0,\mathbf{u}1\})}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{u}0} \wedge \mathsf{EQ}_{\mathbf{u}1} \rightarrow \beta_1 \cdot \mathbf{t}_i + \beta_0 = \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rightarrow \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \enspace .$$

□

□

Proof of Claim 8.13. We rely on the following claim.

Claim 8.16. *For every* $\mathbf{u} \in \{0,1\}^{\ell}$ *and prefix* $\mathbf{w}$ *of* $\mathbf{u}$*:*

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{w}\})}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{w}} \rightarrow \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \enspace .$$

Before proving Claim 8.16 we use it to prove Claim 8.13. By Claim 8.10 for every $\mathbf{u} \in \mathbf{S}_{\leq \mathbf{v}}$, the set $\mathbf{Q}_{\leq \mathbf{v}}$ contains a prefix $\mathbf{w}$ of $\mathbf{u}$. By Claim 8.16 and Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\mathbf{S}_{\leq \mathbf{v}} \cup \mathbf{Q}_{\leq \mathbf{v}})}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{w}} \rightarrow \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \enspace .$$

Since $|\mathbf{S}_{\leq \mathbf{v}}| \leq \bar{M} \leq \mathrm{poly}(\Lambda)$ the claim follows.

Proof of Claim 8.16. We prove the claim by induction on the length i of the prefix $\mathbf{w}$. If $i = \ell$ we have that $\mathbf{w} = \mathbf{u}$ and the claim holds trivially. Assume that the claim holds for the length i prefix $\mathbf{w}$ of $\mathbf{u}$:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u},\mathbf{w}\})}[\mathsf{AC} \wedge \mathsf{EQ}_{\mathbf{w}} \rightarrow \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \enspace . \tag{34}$$

We show that the claim holds also for the prefix $\mathbf{w}'$ of length $i-1$. Let $\bar{\mathbf{w}}$ be the prefix $\mathbf{w}$ with its last bit flipped. We have that:

$$([\mathbf{w}']_{\mathbf{t}})_i = \mathbf{t}_i \quad , \quad ([\mathbf{w}]_{\mathbf{t}})_i = \mathbf{u}_i \quad , \quad ([\bar{\mathbf{w}}]_{\mathbf{t}})_i = 1 - \mathbf{u}_i \quad , \quad ([\mathbf{w}']_{\mathbf{t}})_{-i} = ([\mathbf{w}]_{\mathbf{t}})_{-i} = ([\bar{\mathbf{w}}]_{\mathbf{t}})_{-i} \enspace .$$

In the experiment $\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})$ let:

$$Z = \left(\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_{K'}, \tilde{\mathsf{sk}}) \;,\; \mathbf{c} = (c_1, \ldots, c_{K'}, \tilde{c}_1) \;,\; \mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_{K'}, \tilde{\mathbf{r}}_j)\right) \enspace .$$

Also, let $\beta_0, \beta_1, \beta'_0, \beta'_1 \in \mathbb{F}$ be the unique values such that:

$$\begin{matrix}\langle \beta_1 \cdot \mathbf{u}_i + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{w}]_{\mathbf{t}})} & , & \langle \beta_1 \cdot (1 - \mathbf{u}_i) + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\bar{\mathbf{w}}]_{\mathbf{t}})} \;, \\ \langle \beta'_1 \cdot \mathbf{u}_i + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{w}]_{\mathbf{t}})} & , & \langle \beta'_1 \cdot (1 - \mathbf{u}_i) + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\bar{\mathbf{w}}]_{\mathbf{t}})} \enspace .\end{matrix}$$

When AC occurs $\mathsf{ML.V}(\mathsf{ML.vk}_1, \mathbf{C}_1, \mathsf{ML.\Pi})$ and $\mathsf{ML.V}(\mathsf{ML.vk}_1, \mathbf{C}'_1, \mathsf{ML.\Pi}')$ both accept. Also, we can emulate $\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})$ given $(\mathsf{ML.pk}_1, \mathsf{ML.vk}_1)$ and Z as input. Therefore, by the soundness of the multilinearity protocol (Definition 7.9):

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}\left[\mathsf{AC} \ \rightarrow \ \begin{array}{l} \langle \beta_1 \cdot \mathbf{t}_i + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{w}']_{\mathbf{t}})} \\ \langle \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{w}']_{\mathbf{t}})} \end{array}\right] \geq 1 - {K'}^3 \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $K' \leq \mathrm{poly}(\Lambda)$:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}\left[\mathsf{AC} \ \rightarrow \ \begin{array}{l} \langle \beta_1 \cdot \mathbf{t}_i + \beta_0 \rangle_{\mathsf{pp}} = \alpha_{k([\mathbf{w}']_{\mathbf{t}})} \\ \langle \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rangle_{\mathsf{pp}} = \alpha'_{k([\mathbf{w}']_{\mathbf{t}})} \end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

When $\mathsf{EQ}_{\mathbf{w}'}$ occurs, $\alpha_{k([\mathbf{w}']_{\mathbf{t}})} = \alpha'_{k([\mathbf{w}']_{\mathbf{t}})}$. Therefore:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}\left[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \rightarrow \ \langle \beta_1 \cdot \mathbf{t}_i + \beta_0 \rangle_{\mathsf{pp}} = \langle \beta'_1 \cdot \mathbf{t}_i + \beta'_0 \rangle_{\mathsf{pp}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

When $\neg\mathsf{EQ}_{\mathbf{w}}$ occurs, $\alpha_{k([\mathbf{w}]_{\mathbf{t}})} \neq \alpha'_{k([\mathbf{w}]_{\mathbf{t}})}$ and, therefore, $(\beta_0, \beta_1) \neq (\beta'_0, \beta'_1)$. In this case, let $t \in \mathbb{F}$ be the unique element such that $\beta_1 \cdot t + \beta_0 = \beta'_1 \cdot t + \beta'_0$. We, therefore, have that:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \wedge \ \neg\mathsf{EQ}_{\mathbf{w}} \ \rightarrow \ \mathbf{t}_i = t] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

It follows that:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \not\rightarrow \ \mathsf{EQ}_{\mathbf{w}}] \leq \Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}, \bar{\mathbf{w}}\})}[\neg\mathsf{EQ}_{\mathbf{w}} \wedge \mathbf{t}_i = t] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Given $\mathbf{t}_i$ and the encoding elements $\alpha_{k([\mathbf{w}]_{\mathbf{t}})}, \alpha_{k([\bar{\mathbf{w}}]_{\mathbf{t}})}, \alpha'_{k([\mathbf{w}]_{\mathbf{t}})}, \alpha'_{k([\bar{\mathbf{w}}]_{\mathbf{t}})}$ we can efficiently test whether or not $\mathbf{t}_i = t$ using the fact that the encoding is additively homomorphic. Therefore, by Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \not\rightarrow \ \neg\mathsf{EQ}_{\mathbf{w}}] \leq \Pr_{\mathsf{Exp}'(\{\mathbf{w}, \bar{\mathbf{w}}\})}[\neg\mathsf{EQ}_{\mathbf{w}} \wedge \mathbf{t}_i = t] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since the experiment $\mathsf{Exp}'(\{\mathbf{w}, \bar{\mathbf{w}}\})$ is completely independent of $\mathbf{t}_i$ we have that:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \not\rightarrow \ \mathsf{EQ}_{\mathbf{w}}] \leq \frac{1}{|\mathbb{F}|} + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\Lambda = |\mathbb{F}|^{o(1)}$:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{w}', \mathbf{w}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \rightarrow \ \mathsf{EQ}_{\mathbf{w}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

By the inductive hypothesis (Equation (34)) and Claim 8.8:

$$\Pr_{\mathsf{Exp}'(\{\mathbf{u}, \mathbf{w}', \mathbf{w}\})}[\mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}'} \ \rightarrow \ \mathsf{AC} \ \wedge \ \mathsf{EQ}_{\mathbf{w}} \ \rightarrow \ \mathsf{EQ}_{\mathbf{u}}] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

The claim follows by Claim 8.8. □

□

9 Updatable Unambiguous Delegation

In this section we construct an updatable unambiguous delegation scheme.

9.1 Construction

Fix an input length $n = n(\kappa)$ and a Turing machine $\mathcal{M}$ that runs in time $T = T(n)$ and space $S = S(n) \geq n$. Recall that $\mathcal{U}^{\mathcal{M}}$ consists of tuples $(\mathsf{cf}, \mathsf{cf}', t)$ such that $\mathcal{M}$ transitions from cf to cf' in exactly t steps, and $\mathcal{U}^{\mathcal{M}}_n \subseteq \mathcal{U}^{\mathcal{M}}$ consists of such tuples where the input tapes in $\mathsf{cf}, \mathsf{cf}'$ are of length n. Let $\bar{M} = \bar{M}(\kappa)$ be a size bound that is specified below, and let $\bar{\ell} = \log \bar{M}$. Let $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ be a quasi-argument with formula size bound $\bar{M}$ and input length $N = 2S$ (Definition 8.3).

High-level structure. The delegation scheme for $\mathcal{U}^{\mathcal{M}}$ is based on recursive proof composition. The recursion has $d = \log_n T(n)$ levels, and the proof in each level is composed of $B = n$ proofs in the level below. For $i \in [0, d]$, in the i'th level of the recursion we invoke the quasi-argument setup algorithm on a formula φ_i with locality parameter K_i and obtain the verification key vk_i. Let M_i denote the number of variables in φ_i and let $\ell_i = \log M_i$. Next we define the formula φ_i and specify K_i. For each $i > 0$, the formula φ_i is defined using the previous level's verification key vk_{i-1}.

The base formula φ_0. Let φ_0 be a 3CNF formula that checks a single computation step of the Turing machine $\mathcal{M}$ on inputs of length n. The formula φ_0 has $M_0 = \mathrm{poly}(S)$ variables describing:

- An input x of length N.
- A witness w of length $\mathrm{poly}(S)$.

We require that φ_0 satisfies the following properties:

- For every input $x = (\mathsf{cf}, \mathsf{cf}') \in \{0,1\}^N$ there exists w such that $\varphi_0(x, w) = 1$ if and only if $(\mathsf{cf}, \mathsf{cf}', 1) \in \mathcal{U}^{\mathcal{M}}$. Given x we can compute w in time $\mathrm{poly}(S)$.
- For every x there exists at most one w such that $\varphi_0(x, w) = 1$.

Since $\mathcal{M}$ is deterministic machine with space S, there exists such a formula φ_0. We set $K_0 = M_0$.

The quasi-argument verification formula. To define the formula φ_i for $i \in [d]$, we use the formula φ_i^{QA} that verifies a single quasi-argument proof form level $i-1$. For security parameter $\kappa \in \mathbb{N}$, let vk_{i-1} and K_{i-1} be the verification key and locality parameter of the quasi-argument in level $i-1$. The formula φ_i^{QA} has $(K_{i-1}+N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$ variables describing:

- An input x of length N.
- A quasi-argument proof Π of length $K_{i-1} \cdot \mathrm{poly}(\kappa, \bar{\ell})$.
- A witness w of length $(K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$.

We require that φ_i^{QA} satisfies the following properties:

- For every $(x = (\mathsf{cf}, \mathsf{cf}'), \Pi)$ there is a witness w such that $\varphi_i^{\mathsf{QA}}(x, \Pi, w) = 1$ if and only if:

  $$\mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}, \mathsf{cf}'), \Pi) = 1 \ .$$

  Given (x, Π) we can compute w in time $(K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$.
- For every (x, Π) there exists at most one w such that $\varphi_i^{\mathsf{QA}}(x, \Pi, w) = 1$.

Since $\mathsf{QA.V}$ is a deterministic algorithm running it time $(K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$, there exists such a formula φ_i^{QA}.

The batch verification formula. For $i \in [d]$ the formula φ_i verifies B successive quasi-argument proofs form level $i-1$. For security parameter $\kappa \in \mathbb{N}$ the formula φ_i has M_i variables:

$$M_i = B \cdot (K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell}) \ ,$$

describing:

- An input $x = (\mathsf{cf}_0, \mathsf{cf}_B)$ of length N.
- For every $j \in [B-1]$, a configuration cf_j of length S.
- For every $j \in [B]$, a quasi-argument proof Π_j of length $K_{i-1} \cdot \mathrm{poly}(\kappa, \bar{\ell})$.
- For every $j \in [B]$, a witness w_j of length $(K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$.

The formula φ_i is given by:

$$\varphi_i\left(\mathsf{cf}_0, (\mathsf{cf}_j, \Pi_j, w_j)_{j\in[B]}\right) = \bigwedge_{j\in[B]} \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j-1}, \mathsf{cf}_j), \Pi_j, w_j) \ .$$

We set $K_i = (K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell})$ to be the number of variables in the formula φ_i^{QA}. Therefore, we have that $M_i < B \cdot K_i$.

Setting the formula size bound $\bar{M}$. We need to set $\bar{M}$ such that $M_i \leq \bar{M}$ for every $i \in [d]$. To this end we assume that $T(n) \leq 2^\kappa$. Recall that we set $K_0 = M_0 \geq N$. Since $K_i \geq K_{i-1}$ for every $i \in [d]$, also $K_i \geq N$ and, therefore:

$$K_i = (K_{i-1} + N) \cdot \mathrm{poly}(\kappa, \bar{\ell}) = K_{i-1} \cdot \mathrm{poly}(\kappa, \bar{\ell}) \ .$$

Since $M_0 = \mathrm{poly}(S)$ and $\bar{\ell} = \log \bar{M}$:

$$K_i = \mathrm{poly}(S) \cdot (\mathrm{poly}(\kappa, \bar{\ell}))^i = \mathrm{poly}(S, \kappa^i, \log^i \bar{M}) \ .$$

Since $M_i < B \cdot K_i$ and $B = n \leq S$:

$$M_i = \mathrm{poly}(S, \kappa^i, \log^i \bar{M}) \ .$$

Since $T(n) \leq 2^\kappa$ it follows that $d = \log_n T(n) \leq \kappa$. Therefore, we can set $\bar{M} = \mathrm{poly}(S, \kappa^d)$ such that $\log \bar{M} \leq \mathrm{poly}(\log S, \kappa)$ and for every $i \in [d]$:

$$M_i = \mathrm{poly}(S, \kappa^i, \log^i \bar{M}) \leq \bar{M} \ .$$

For such $\bar{M}$ we have that:

$$M_i = \mathrm{poly}(S, \kappa^i) \quad , \quad K_i = \mathrm{poly}(S, \kappa^i) \ . \tag{35}$$

We can now describe the scheme's algorithms $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V}, \mathsf{Del.U})$.

The setup algorithm $\mathsf{Del.S}$. The setup algorithm is given as input the security parameter κ and it proceeds as follows:

- For every $i \in [0, d]$ set $(\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i)$.
- Output the prover key $\mathsf{pk} = (\mathsf{pk}_i, \mathsf{vk}_i)_{i\in[0,d]}$ and verifier key $\mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}$.

The prover algorithm $\mathsf{Del.P}$. The prover algorithm is given as input:

- A prover key $\mathsf{pk} = (\mathsf{pk}_i, \mathsf{vk}_i)_{i\in[0,d]}$.
- An input $x = (\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}_n^{\mathcal{M}}$.

It proceeds as follows:

- Set $\mathsf{cf}_0 \leftarrow \mathsf{cf}$ and $\Pi_0 \leftarrow \mathcal{E}$.
- For every $i \in [t]$ set $(\mathsf{cf}_i, \Pi_i) \leftarrow \mathsf{Del.U}(\mathsf{pk}, (\mathsf{cf}_0, \mathsf{cf}_{i-1}, i-1), \Pi_{i-1})$.
- Output the proof Π_t.

The verifier algorithm $\mathsf{Del.V}$**.** The verifier algorithm is given as input:

- A verifier key $\mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}$.
- An input $x = (\mathsf{cf}, \mathsf{cf}', t)$.
- A proof Π.

It proceeds as follows:

- Test that cf and cf' contain the same input tape of length n.
- Let $\beta_0, \ldots, \beta_d \in [0, B-1]$ be such that $t = \sum_{i\in[0,d]} \beta_i \cdot B^i$.
- Parse $\Pi = (x_{i,j}, \Pi_{i,j})_{i\in[0,d], j\in[\beta_i]}$.
- For every $i \in [0,d]$ and $j \in [\beta_i]$ test that $\mathsf{QA.V}(\mathsf{vk}_i, x_{i,j}, \Pi_{i,j})$ accepts.
- Let $(y_k)_{k\in[\bar{k}]}$ be a lexicographic ordering of $(x_{i,j})_{i\in[0,d],j\in[\beta_i]}$ first by decreasing order of i and then by increasing order of j. That is, for $y_k = x_{i,j}$ and $y_{k'} = x_{i',j'}$, if $k < k'$ then either $i > i'$, or $i = i'$ and $j < j'$.
- Parse $y_k = (\mathsf{cf}_k, \mathsf{cf}'_k)$ and test that $\left(\mathsf{cf}_1, \mathsf{cf}'_{\bar{k}}\right) = \left(\mathsf{cf}, \mathsf{cf}'\right)$ and for every $k \in [\bar{k}-1]$, $\mathsf{cf}'_k = \mathsf{cf}_{k+1}$.
- Output 1 if all tests pass. Otherwise output 0.

The update algorithm $\mathsf{Del.U}$**.** The update algorithm is given as input:

- A prover key $\mathsf{pk} = (\mathsf{pk}_i, \mathsf{vk}_i)_{i\in[0,d]}$.
- An input $x = (\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}_n^{\mathcal{M}}$.
- A proof Π.

It proceeds as follows:

- Let cf'' by the configuration such that $(\mathsf{cf}', \mathsf{cf}'', 1) \in \mathcal{U}_n^{\mathcal{M}}$.
- Let w be such that $\varphi_0((\mathsf{cf}', \mathsf{cf}''), w) = 1$.
- Let $\beta_0, \ldots, \beta_d \in [0, B-1]$ be such that $t = \sum_{i\in[0,d]} \beta_i \cdot B^i$.
- Parse $\Pi = (x_{i,j}, \Pi_{i,j})_{i\in[0,d], j\in[\beta_i]}$.
- Set $\beta_0 \leftarrow \beta_0 + 1$, $x_{0,\beta_0} \leftarrow (\mathsf{cf}', \mathsf{cf}'')$ and $\Pi_{0,\beta_0} \leftarrow \mathsf{QA.P}(\mathsf{pk}_0, (x_{0,\beta_0}, w))$.
- While there exists $i \in [0,d]$ such that $\beta_i = B$:
  - For every $j \in [B]$ let w_j be such that $\varphi_i^{\mathsf{QA}}(x_{i,j}, \Pi_{i,j}, w_j) = 1$.
  - For every $j \in [B]$ parse $x_{i,j} = (\mathsf{cf}_j, \mathsf{cf}'_j)$.
  - Set $\beta_i \leftarrow 0$ and $\beta_{i+1} \leftarrow \beta_{i+1} + 1$.
  - Set $x_{i+1,\beta_{i+1}} = (\mathsf{cf}_1, \mathsf{cf}'_B)$ and $\Pi_{i+1,\beta_{i+1}} \leftarrow \mathsf{QA.P}\left(\mathsf{pk}_{i+1}, \left(\mathsf{cf}_1, (\mathsf{cf}'_j, \Pi_{i,j}, w^i_j)_{j\in[B]}\right)\right)$.
- Output the configuration cf'' and proof $\Pi' = (x_{i,j}, \Pi_{i,j})_{i\in[0,d], j\in[\beta_i]}$.

9.2 Analysis

In this section we prove the following theorem:

Theorem 9.1. *For any deterministic Turing machine $\mathcal{M}$ that runs in time $T = T(n)$ and space $S = S(n) \geq n$ and any functions $n = n(\kappa), \Lambda = \Lambda(\kappa)$ such that $n \geq \kappa$ and $T(n) \leq \Lambda \leq 2^{\kappa}$, assuming that $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ is a Λ-secure Λ-unambiguous quasi-argument (Definitions 8.3 and 8.4) with formula size bound $\bar{M} = \mathrm{poly}(S, \kappa^{\log_n T(n)})$ and input length $N = 2S$, the delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V}, \mathsf{Del.U})$ given in Section 9.1 is an updatable Λ-sound Λ-unambiguous delegation scheme for $\mathcal{U}^{\mathcal{M}}$ with input length n, setup time $T_{\mathsf{S}} = \mathrm{poly}(M)$ and proof length $L_{\Pi} = \mathrm{poly}(\bar{M})$.*

The completeness, efficiency, and updatability requirements follow by construction and by the completeness and efficiency of the quasi-argument. We focus on proving the soundness and unambiguity.

9.2.1 Soundness

We rely on the following claim.

Claim 9.2. *For every $\mathrm{poly}(\Lambda)$-size adversary Adv there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and $i^* \in [0, d]$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_{i^*}, x, \Pi) = 1 & \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}\right) \leftarrow \mathsf{Del.S}(\kappa) \\ (\mathsf{cf}, \mathsf{cf}', B^{i^*}) \notin \mathcal{U}_n^{\mathcal{M}} & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}\right] \leq \mu(\Lambda(\kappa)) \ .$$

Before proving Claim 9.2 we use it to prove soundness. Fix a $\mathrm{poly}(\Lambda)$-size adversary Adv and $\kappa \in \mathbb{N}$. Let Exp denote the soundness experiment:

$$\begin{array}{l} \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}\right) \leftarrow \mathsf{Del.S}(\kappa) \\ (x = (\mathsf{cf}, \mathsf{cf}', t), \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}$$

Let $\beta_0, \ldots, \beta_d \in [0, B-1]$ be such that $t = \sum_{i\in[0,d]} \beta_i \cdot B^i$ and let $\Pi = \left(x_{i,j} = (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}), \Pi_{i,j}\right)_{i\in[0,d], j\in[\beta_i]}$. Let $\left(y_k = (\mathsf{cf}_k, \mathsf{cf}'_k)\right)_{k\in[\bar{k}]}$ be a lexicographic ordering of $(x_{i,j})_{i\in[0,d],j\in[\beta_i]}$ first by decreasing order of i and then by increasing order of j. If $\mathsf{Del.V}(\mathsf{vk}, x, \Pi)$ accepts than:

- For every $i \in [0, d]$ and $j \in [\beta_i]$, $\mathsf{QA.V}(\mathsf{vk}_i, x_{i,j}, \Pi_{i,j})$ accepts. Therefore, by Claim 9.2:

  $$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \rightarrow \ \forall i \in [0, d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - d \cdot B \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

  Since $d, B \leq T(n) \leq \Lambda$:

  $$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \rightarrow \ \forall i \in [0, d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- $\left(\mathsf{cf}_1, \mathsf{cf}'_{\bar{k}}\right) = \left(\mathsf{cf}, \mathsf{cf}'\right)$ and for every $k \in [\bar{k} - 1]$, $\mathsf{cf}'_k = \mathsf{cf}_{k+1}$ Therefore:

  $$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \wedge \ \forall i \in [0, d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}} \ \rightarrow \ (\mathsf{cf}, \mathsf{cf}', t) \in \mathcal{U}_n^{\mathcal{M}}\right] = 1 \ .$$

Soundness follows. It remains to prove Claim 9.2.

Proof of Claim 9.2. For an adversary Adv and $\kappa \in \mathbb{N}$, let $\mathsf{Exp}^{\mathsf{Adv}}$ denote the experiment:

$$\begin{array}{l} \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}\right) \leftarrow \mathsf{Del.S}(\kappa) \\ (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \end{array}$$

For $i \in [0, d]$ let AC_i be the event that in the experiment $\mathsf{Exp}^{\mathsf{Adv}}$, $\mathsf{QA.V}(\mathsf{vk}_i, x, \Pi)$ accepts and $(\mathsf{cf}, \mathsf{cf}', B^i) \notin \mathcal{U}_n^{\mathcal{M}}$.

We rely on the following claims.

Claim 9.3. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*:*

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \mu(\Lambda(\kappa)) \ .$$

Claim 9.4. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *and function* ϵ *there exists an adversary* Adv' *of size* $|\mathsf{Adv}|+\mathrm{poly}(\Lambda)$ *and negligible function* μ *such that for every* $\kappa \in \mathbb{N}$ *and* $i \in [d]$ *if:*

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] \leq B \cdot \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] + \mu(\Lambda(\kappa)) \ .$$

Since $B^d = T(n) \leq \Lambda$, Claim 9.2 follows. It remains to prove Claims 9.3 and 9.4.

Proof of Claim 9.3. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv. Let Adv' be the adversary that is given $(\mathsf{pk}_0, \mathsf{vk}_0)$ emulates the experiment $\mathsf{Exp}^{\mathsf{Adv}}$ except that in the emulation of $\mathsf{Del.S}$ it uses $(\mathsf{pk}_0, \mathsf{vk}_0)$ given as input instead of having $\mathsf{Del.S}$ sample these keys. Since $\mathsf{Del.S}$ runs in time $\mathrm{poly}(\bar{M}) = \mathrm{poly}(S, \kappa^d)$ and $\kappa^d \leq n^d = T(n) \leq \Lambda$ we have that Adv' is also of size $\mathrm{poly}(\Lambda)$ and for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] = \Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_0, x, \Pi) = 1 & (\mathsf{pk}_0, \mathsf{vk}_0) \leftarrow \mathsf{QA.S}(\kappa, \varphi_0, K_0) \\ (\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}} & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}'(\mathsf{pk}_0, \mathsf{vk}_0) \end{array}\right] .$$

Or, equivalently:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] = \Pr\left[\begin{array}{l|l} (\mathsf{cf}, \mathsf{cf}') = x \neq \bot & (\mathsf{pk}_0, \mathsf{vk}_0) \leftarrow \mathsf{QA.S}(\kappa, \varphi_0, K_0) \\ (\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}} & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}'(\mathsf{pk}_0, \mathsf{vk}_0) \\ & \text{if } \mathsf{QA.V}(\mathsf{vk}_0, x, \Pi) = 0 : \text{set } x = \bot \end{array}\right] .$$

By the non-signaling extraction of the quasi-argument (Definition 8.3) there exists an oracle machine E such that:

- $\mathsf{E}^{\mathsf{Adv}'}$ is a Λ-non-signaling extractor with formula size bound $\bar{M}$, input length N and locality K_0.
- Since we can test if $(\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}}$ in time $\mathrm{poly}(S) \leq \mathrm{poly}(\Lambda)$ we have that for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'}(\varphi_0, \emptyset)}\left[\begin{array}{l} (\mathsf{cf}, \mathsf{cf}') = x \neq \bot \\ (\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Recall that $K_0 = M_0$ is the number of variables in the formula φ_0. Therefore, by the non-signaling of E (Definition 8.2) for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'}(\varphi_0, [M_0])}\left[\begin{array}{l} (\mathsf{cf}, \mathsf{cf}') = x \neq \bot \\ (\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let $(\tilde{x}, \tilde{w})$ be the values described by the assignment $\sigma : [M_0] \to \{0,1\}$. By the local consistency of E (Definition 8.2) for every $\kappa \in \mathbb{N}$:

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}(\varphi_0, [M_0])}\left[x = \bot \quad \vee \quad \begin{array}{l} x = \tilde{x} \\ \varphi_0(\tilde{x}, \tilde{w}) = 1 \end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Therefore, we have that for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'}(\varphi_0, [M_0])}\left[\begin{array}{l} (\mathsf{cf}, \mathsf{cf}') = x \neq \bot \\ \varphi_0(x, \tilde{w}) = 1 \\ (\mathsf{cf}, \mathsf{cf}', 1) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By the construction of φ_0 there exists $\tilde{w}$ such that $\varphi_0(x, \tilde{w}) = 1$ only if $(\mathsf{cf}, \mathsf{cf}', 1) \in \mathcal{U}^{\mathcal{M}}$. Therefore:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

□

Proof of Claim 9.4. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv and for every $\kappa \in \mathbb{N}$ fix any $i \in [d]$. Let Exp'_r be the experiment $\mathsf{Exp}^{\mathsf{Adv}}$ where we fix the randomness r of $\mathsf{Del.S}$ used to sample the keys $(\mathsf{pk}_j, \mathsf{vk}_j)$ for every $j < i$. Therefore, we have that:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] = \Pr_{r, \mathsf{Exp}'_r}[\mathsf{AC}_i] \ .$$

For every $\kappa \in \mathbb{N}$, fix some r. Note that this also fixes the formula φ_i. Let Adv'_r be the adversary that is given $(\mathsf{pk}_i, \mathsf{vk}_i)$ emulates the experiment Exp'_r except that in the emulation of $\mathsf{Del.S}$ it uses $(\mathsf{pk}_i, \mathsf{vk}_i)$ given as input instead of having $\mathsf{Del.S}$ sample these keys. Since $\mathsf{Del.S}$ runs in time $\mathrm{poly}(\bar{M}) = \mathrm{poly}(S, \kappa^d)$ and $\kappa^d \leq n^d = T(n) \leq \Lambda$, we have that Adv'_r is also of size $\mathrm{poly}(\Lambda)$ and for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] = \Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_i, x, \Pi) = 1 & (\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i) \\ (\mathsf{cf}, \mathsf{cf}', B^i) \notin \mathcal{U}_n^{\mathcal{M}} & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}'_r(\mathsf{pk}_i, \mathsf{vk}_i) \end{array}\right] .$$

Or, equivalently:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] = \Pr\left[\begin{array}{l|l} (\mathsf{cf}, \mathsf{cf}') = x \neq \bot & (\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i) \\ (\mathsf{cf}, \mathsf{cf}', B^i) \notin \mathcal{U}_n^{\mathcal{M}} & (x, \Pi) \leftarrow \mathsf{Adv}'_r(\mathsf{pk}_i, \mathsf{vk}_i) \\ & \text{if } \mathsf{QA.V}(\mathsf{vk}_i, x, \Pi) = 0 : \text{set } x = \bot \end{array}\right] .$$

For $x = (\mathsf{cf}, \mathsf{cf}')$ and $j \in [0, B]$ we denote by $\bar{\mathsf{cf}}_j$ the unique configuration such that $(\mathsf{cf}, \bar{\mathsf{cf}}_j, j \cdot B^{i-1}) \in \mathcal{U}_n^{\mathcal{M}}$. The configuration $\bar{\mathsf{cf}}_j$ can computed in time $\mathrm{poly}(S) \cdot B^i \leq \mathrm{poly}(\Lambda)$ given x. We have that for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] = \Pr\left[\begin{array}{l|l} x \neq \bot & (\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i) \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B & (x, \Pi) \leftarrow \mathsf{Adv}'_r(\mathsf{pk}_i, \mathsf{vk}_i) \\ & \text{if } \mathsf{QA.V}(\mathsf{vk}_i, x, \Pi) = 0 : \text{set } x = \bot \end{array}\right] .$$

By the non-signaling extraction requirement of the quasi-argument (Definition 8.3) there exists an oracle machine E such that:

- E runs in time $\mathrm{poly}(\bar{M})$ and makes a single oracle call to Adv'_r. Therefore, we can emulate $\mathsf{E}^{\mathsf{Adv}'_r}$ in time $\mathrm{poly}(\Lambda)$.
- $\mathsf{E}^{\mathsf{Adv}'_r}$ is a Λ-non-signaling extractor with formula size bound $\bar{M}$, input length N and locality K_i.
- Since the configuration $\bar{\mathsf{cf}}_B$ can computed in time $\mathrm{poly}(\Lambda)$, we have that for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \emptyset)}\left[\begin{array}{l} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Recall that the formula φ_i is given by:

$$\varphi_i\left(\mathsf{cf}_0, (\mathsf{cf}_j, \Pi_j, w_j)_{j \in [B]}\right) = \bigwedge_{j \in [B]} \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j-1}, \mathsf{cf}_j), \Pi_j, w_j) \ ,$$

and that K_i is the number of variables in the formula φ_i^{QA}. For every $j \in [B]$ let $\mathbf{S}_j$ be the set of K_i variables describing the values $(\mathsf{cf}_{j-1}, \mathsf{cf}_j, \Pi_j, w_j)$. By the non-signaling of E (Definition 8.2) for every $j \in [B]$ and $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)}\left[\begin{array}{l} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{36}$$

Let $(\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j, \tilde{\Pi}_j, \tilde{w}_j)$ be the values described by the assignment $\sigma : \mathbf{S}_j \to \{0,1\}$. By the local consistency of E (Definition 8.2) for every $j \in [B]$ and $\kappa \in \mathbb{N}$:

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)}\left[x = \bot \quad \vee \quad \begin{array}{l} \forall i \in \mathbf{S}_j \cap [N] : \ \sigma(i) = x_i \\ \varphi_i^{\mathsf{QA}}((\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j), \tilde{\Pi}_j, \tilde{w}_j) = 1 \end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

By the construction of φ_i^{QA} there exists $\tilde{w}_j$ such that $\varphi_i^{\mathsf{QA}}((\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j), \tilde{\Pi}_j, \tilde{w}_j) = 1$ only if:

$$\mathsf{QA.V}(\mathsf{vk}_{i-1}, (\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j), \tilde{\Pi}_j) = 1 \ .$$

Therefore, we have that for every $j \in [B]$ and $\kappa \in \mathbb{N}$:

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \left[x \neq \bot \quad \rightarrow \quad \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j), \tilde{\Pi}_j) = 1 \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{37}$$

Additionally, for $j = 1$:

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \left[x \neq \bot \quad \rightarrow \quad \bar{\mathsf{cf}}_0 = \mathsf{cf} = \tilde{\mathsf{cf}}_0 \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{38}$$

Similarly, for $j = B$:

$$\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \left[x \neq \bot \quad \rightarrow \quad \bar{\mathsf{cf}}_B \neq \mathsf{cf}' = \tilde{\mathsf{cf}}_B \right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ . \tag{39}$$

By the non-signaling of E (Definition 8.2) for every $j \in [B-1]$ and $\kappa \in \mathbb{N}$:

$$\left| \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \bar{\mathsf{cf}}_j = \tilde{\mathsf{cf}}_j \end{bmatrix} - \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j+1})} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \bar{\mathsf{cf}}_j = \tilde{\mathsf{cf}}_j \end{bmatrix} \right| \leq \mathrm{negl}(\Lambda(\kappa)) \ . \tag{40}$$

For every $j \in [B]$, since $\bar{\mathsf{cf}}_j$ is the unique configuration such that $(\bar{\mathsf{cf}}_{j-1}, \bar{\mathsf{cf}}_j, B^{i-1}) \in \mathcal{U}_n^{\mathcal{M}}$. Therefore:

$$\begin{aligned} &\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \tilde{\mathsf{cf}}_{j-1} = \bar{\mathsf{cf}}_{j-1} \end{bmatrix} \\ &\leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ (\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{bmatrix} + \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \tilde{\mathsf{cf}}_j = \bar{\mathsf{cf}}_j \end{bmatrix} \ . \end{aligned} \tag{41}$$

By Equations (36) and (38):

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_1)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \tilde{\mathsf{cf}}_0 = \bar{\mathsf{cf}}_0 \end{bmatrix} + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Equations (40) and (41):

$$\begin{aligned} \Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \sum_{j \in [B]} &\Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ (\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{bmatrix} \\ &+ \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_B)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ \tilde{\mathsf{cf}}_B = \bar{\mathsf{cf}}_B \end{bmatrix} + \mathrm{negl}(\Lambda(\kappa)) \ . \end{aligned}$$

By Equation (39):

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \sum_{j \in [B]} \Pr_{(x,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_j)} \begin{bmatrix} x \neq \bot \\ \mathsf{cf}' \neq \bar{\mathsf{cf}}_B \\ (\tilde{\mathsf{cf}}_{j-1}, \tilde{\mathsf{cf}}_j, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{bmatrix} + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By Equation (37):

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq \sum_{j\in[B]} \Pr_{(x,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_j)} \left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1},(\tilde{\mathsf{cf}}_{j-1},\tilde{\mathsf{cf}}_j),\tilde{\Pi}_j)=1 \\ (\tilde{\mathsf{cf}}_{j-1},\tilde{\mathsf{cf}}_j,B^{i-1})\notin \mathcal{U}_n^{\mathcal{M}} \end{array} \right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Therefore, there exists $j^* \in [B]$ such that:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] = \Pr_{r,\mathsf{Exp}'_r}[\mathsf{AC}_i] \leq B \cdot \Pr_{r,\mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})} \left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1},(\tilde{\mathsf{cf}}_{j^*-1},\tilde{\mathsf{cf}}_{j^*}),\tilde{\Pi}_{j^*})=1 \\ (\tilde{\mathsf{cf}}_{j^*-1},\tilde{\mathsf{cf}}_{j^*},B^{i-1})\notin \mathcal{U}_n^{\mathcal{M}} \end{array} \right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Adv' be the adversary that given $(\mathsf{pk},\mathsf{vk}=(\mathsf{vk}_i)_{i\in[0,d]})$ sampled from $\mathsf{Del.S}(\kappa)$ emulates $\mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})$ except that in the emulation of $\mathsf{Del.S}$ it uses the input keys $(\mathsf{pk}_j,\mathsf{vk}_j)$ for every $j<i$ instead of having $\mathsf{Del.S}$ sample these keys using the randomness r (note that r is not used anywhere else in the emulation). The adversary Adv' outputs $((\tilde{\mathsf{cf}}_{j^*-1},\tilde{\mathsf{cf}}_{j^*}),\tilde{\Pi}_{j^*})$. We have that:

$$\Pr_{r,\mathsf{E}^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})} \left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1},(\tilde{\mathsf{cf}}_{j^*-1},\tilde{\mathsf{cf}}_{j^*}),\tilde{\Pi}_{j^*})=1 \\ (\tilde{\mathsf{cf}}_{j^*-1},\tilde{\mathsf{cf}}_{j^*},B^{i-1})\notin \mathcal{U}_n^{\mathcal{M}} \end{array} \right] = \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] \ .$$

Therefore:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] \leq B \cdot \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

□

□

9.2.2 Unambiguity.

We rely on the following claim.

Claim 9.5. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa\in\mathbb{N}$ *and* $i^*\in[0,d]$*:*

$$\Pr\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i^*},x,\Pi)=1 \\ \mathsf{QA.V}(\mathsf{vk}_{i^*},x,\bar{\Pi})=1 \\ \Pi\neq\bar{\Pi} \end{array} \middle| \begin{array}{l} (\mathsf{pk},\mathsf{vk}=(\mathsf{vk}_i)_{i\in[0,d]})\leftarrow\mathsf{Del.S}(\kappa) \\ (x=(\mathsf{cf},\mathsf{cf}'),\Pi,\bar{\Pi})\leftarrow\mathsf{Adv}(\mathsf{pk},\mathsf{vk}) \end{array} \right] \leq \mu(\Lambda(\kappa)) \ .$$

Before proving Claim 9.5 we use it to prove unambiguity. Fix a $\mathrm{poly}(\Lambda)$-size adversary Adv and $\kappa\in\mathbb{N}$. Let Exp denote the unambiguity experiment:

$$\begin{array}{l} (\mathsf{pk},\mathsf{vk}=(\mathsf{vk}_i)_{i\in[0,d]})\leftarrow\mathsf{Del.S}(\kappa) \\ (x=(\mathsf{cf},\mathsf{cf}',t),\Pi,\bar{\Pi})\leftarrow\mathsf{Adv}(\mathsf{pk},\mathsf{vk}) \end{array}$$

Let $\beta_0,\ldots,\beta_d\in[0,B-1]$ be such that $t=\sum_{i\in[0,d]}\beta_i\cdot B^i$ and let

$$\Pi=\left(x_{i,j}=(\mathsf{cf}_{i,j},\mathsf{cf}'_{i,j}),\Pi_{i,j}\right)_{i\in[0,d],j\in[\beta_i]} \quad , \quad \bar{\Pi}=\left(\bar{x}_{i,j}=(\bar{\mathsf{cf}}_{i,j},\bar{\mathsf{cf}}'_{i,j}),\bar{\Pi}_{i,j}\right)_{i\in[0,d],j\in[\beta_i]} \ .$$

Let $\left(y_k=(\mathsf{cf}_k,\mathsf{cf}'_k)\right)_{k\in[\bar{k}]}$ be a lexicographic ordering of $(x_{i,j})_{i\in[0,d],j\in[\beta_i]}$ first by decreasing order of i and then by increasing order of j. Similarly, let $\left(\bar{y}_k=(\bar{\mathsf{cf}}_k,\bar{\mathsf{cf}}'_k)\right)_{k\in[\bar{k}]}$ be a lexicographic ordering of $(\bar{x}_{i,j})_{i\in[0,d],j\in[\beta_i]}$. If $\mathsf{Del.V}(\mathsf{vk},x,\Pi)$ and $\mathsf{Del.V}(\mathsf{vk},x,\bar{\Pi})$ both accept then:

- For every $i \in [0,d]$ and $j \in [\beta_i]$, $\mathsf{QA.V}(\mathsf{vk}_i, x_{i,j}, \Pi_{i,j})$ and $\mathsf{QA.V}(\mathsf{vk}_i, \bar{x}_{i,j}, \bar{\Pi}_{i,j})$ both accept. Therefore, by Claim 9.2:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - d \cdot B \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

  Since $d, B \leq T(n) \leq \Lambda$:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

- $\left(\mathsf{cf}_1, \mathsf{cf}'_{\bar{k}}\right) = \left(\mathsf{cf}, \mathsf{cf}'\right)$ and for every $k \in [\bar{k}-1]$, $\mathsf{cf}'_k = \mathsf{cf}_{k+1}$. Let $t_{i,j} = \left(\sum_{i' \in [i+1,d]} \beta_{i'} \cdot B^{i'}\right) + j \cdot B^i$. Since $\mathcal{M}$ is deterministic, the following holds:

$$\Pr_{\mathsf{Exp}}\left[\begin{array}{l}\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \\ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}_{i,j}, \mathsf{cf}'_{i,j}, B^i) \in \mathcal{U}_n^{\mathcal{M}}\end{array} \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}, \mathsf{cf}'_{i,j}, t_{i,j}) \in \mathcal{U}_n^{\mathcal{M}}\right] = 1 \ .$$

Together this implies:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}, \mathsf{cf}'_{i,j}, t_{i,j}) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Similarly:

$$\Pr_{\mathsf{Exp}}\left[\mathsf{Del.V}(\mathsf{vk}, x, \bar{\Pi}) \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : (\mathsf{cf}, \bar{\mathsf{cf}}'_{i,j}, t_{i,j}) \in \mathcal{U}_n^{\mathcal{M}}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\mathcal{M}$ is deterministic, if $(\mathsf{cf}, \mathsf{cf}'_{i,j}, t_{i,j}) \in \mathcal{U}_n^{\mathcal{M}}$ and $(\mathsf{cf}, \bar{\mathsf{cf}}'_{i,j}, t_{i,j}) \in \mathcal{U}_n^{\mathcal{M}}$ then $\mathsf{cf}'_{i,j} = \bar{\mathsf{cf}}'_{i,j}$. Thus:

$$\Pr_{\mathsf{Exp}}\left[\begin{array}{l}\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \\ \mathsf{Del.V}(\mathsf{vk}, x, \bar{\Pi})\end{array} \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : x_{i,j} = \bar{x}_{i,j}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Then by Claim 9.5:

$$\Pr_{\mathsf{Exp}}\left[\begin{array}{l}\mathsf{Del.V}(\mathsf{vk}, x, \Pi) \\ \mathsf{Del.V}(\mathsf{vk}, x, \bar{\Pi}) \\ \forall i \in [0,d], j \in [\beta_i] \ : x_{i,j} = \bar{x}_{i,j}\end{array} \ \rightarrow \ \forall i \in [0,d], j \in [\beta_i] \ : \Pi_{i,j} = \bar{\Pi}_{i,j}\right] \geq 1 - d \cdot B \cdot \mathrm{negl}(\Lambda(\kappa)) \ .$$

Unambiguity follows. It remains to prove Claim 9.5.

Proof of Claim 9.5. For an adversary Adv and $\kappa \in \mathbb{N}$, let $\mathsf{Exp}^{\mathsf{Adv}}$ denote the experiment:

$$\begin{array}{l}(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i \in [0,d]}) \leftarrow \mathsf{Del.S}(\kappa) \\ (x = (\mathsf{cf}, \mathsf{cf}'), \Pi, \bar{\Pi}) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk})\end{array}$$

For $i \in [0,d]$ let AC_i be the event that in the experiment $\mathsf{Exp}^{\mathsf{Adv}}$, $\mathsf{QA.V}(\mathsf{vk}_i, x, \Pi)$ and $\mathsf{QA.V}(\mathsf{vk}_i, x, \bar{\Pi})$ both accept and $\Pi \neq \bar{\Pi}$.

We rely on the following claims.

Claim 9.6. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$*:*

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \mu(\Lambda(\kappa)) \ .$$

Claim 9.7. *For every* $\mathrm{poly}(\Lambda)$*-size adversary* Adv *there exists an adversary* Adv' *of size* $|\mathsf{Adv}| + \mathrm{poly}(\Lambda)$ *and negligible function* μ *such that for every* $\kappa \in \mathbb{N}$ *and* $i \in [d]$*:*

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] \leq B \cdot \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] + \mu(\Lambda(\kappa)) \ .$$

Since $B^d = T(n) \leq \Lambda$, Claim 9.5 follows. It remains to prove Claims 9.6 and 9.7.

Proof of Claim 9.6. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv. Let Adv' be the adversary that is given $(\mathsf{pk}_0, \mathsf{vk}_0)$ emulates the experiment $\mathsf{Exp}^{\mathsf{Adv}}$ except that in the emulation of $\mathsf{Del.S}$ it uses $(\mathsf{pk}_0, \mathsf{vk}_0)$ given as input instead of having $\mathsf{Del.S}$ sample these keys. Since $\mathsf{Del.S}$ runs in time $\mathrm{poly}(\bar{M}) = \mathrm{poly}(S, \kappa^d)$ and $\kappa^d \leq n^d = T(n) \leq \Lambda$ we have that Adv' is also of size $\mathrm{poly}(\Lambda)$ and for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] = \Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_0, x, \Pi) = 1 & (\mathsf{pk}_0, \mathsf{vk}_0) \leftarrow \mathsf{QA.S}(\kappa, \varphi_0, K_0) \\ \mathsf{QA.V}(\mathsf{vk}_0, x, \bar{\Pi}) = 1 & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi, \bar{\Pi}) \leftarrow \mathsf{Adv}'(\mathsf{pk}_0, \mathsf{vk}_0) \\ \Pi \neq \bar{\Pi} & \end{array}\right] .$$

For every $i' \in [B]$ let $\mathbf{S}_{i'} = [M_0]$ and note that $|\mathbf{S}_{i'}| \leq K_0$ and $\bigcup_{i' \in [B]} \mathbf{S}_{i'} = [M_0]$. By the Λ-unambiguity of the quasi-argument (Definition 8.4) there exist oracle machines $\mathsf{E}_1, \mathsf{E}_2$ such that for every $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 1 & (\mathsf{pk}_0, \mathsf{vk}_0) \leftarrow \mathsf{QA.S}(\kappa, \varphi_0, K_0) \\ \mathsf{QA.V}(\mathsf{vk}, x, \bar{\Pi}) = 1 & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi, \bar{\Pi}) \leftarrow \mathsf{Adv}'(\mathsf{pk}_0, \mathsf{vk}_0) \\ \Pi \neq \bar{\Pi} & \end{array}\right] \leq$$
$$\sum_{i' \in [B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_{i'} \ ; \ r) \\ \sigma_1 \neq \sigma_2 & \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) .$$

Furthermore:

$$\sum_{i' \in [B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_{i'} \ ; \ r) \\ \sigma_1 \neq \sigma_2 & \end{array}\right] \leq$$
$$\sum_{i' \in [B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_{i'} \ ; \ r) \\ \exists j \in [2] \ : \ x_j \neq \tilde{x}_j \vee \varphi_0(\sigma_j) \neq 1 & \end{array}\right] .$$

For $i \in [B], j \in [2]$ by the local consistency of E_j (Definition 8.2):

$$\Pr_{(x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_i)}\left[x_j = \bot \quad \vee \quad \begin{array}{l} x_j = \tilde{x}_j \\ \varphi_0(\sigma_j) = 1 \end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) .$$

Thus:

$$\sum_{i' \in [B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_{i'} \ ; \ r) \\ \exists j \in [2] \ : \ x_j \neq \tilde{x}_j \vee \varphi_0(\sigma_j) \neq 1 & \end{array}\right] \leq$$
$$\sum_{i' \in [B], j \in [2]} \Pr_r\left[\begin{array}{l|l} x_j \neq \bot & (x_j, \sigma_j = (\tilde{x}_j, \tilde{w}_j)) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'}(\varphi_0, \mathbf{S}_{i'} \ ; \ r) \\ x_j \neq \tilde{x}_j \vee \varphi_0(\sigma_j) \neq 1 & \end{array}\right] \leq B \cdot \mathrm{negl}(\Lambda(\kappa)) .$$

The above together with $B \leq T(n) \leq \Lambda$ shows that:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_0] \leq \mathrm{negl}(\Lambda(\kappa)) .$$

□

Proof of Claim 9.7. Fix any $\mathrm{poly}(\Lambda)$-size adversary Adv and for every $\kappa \in \mathbb{N}$ fix any $i \in [d]$. Let Exp'_r be the experiment $\mathsf{Exp}^{\mathsf{Adv}}$ where we fix the randomness r of $\mathsf{Del.S}$ used to sample the keys $(\mathsf{pk}_j, \mathsf{vk}_j)$ for every $j < i$. Therefore, we have that:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] = \Pr_{r, \mathsf{Exp}'_r}[\mathsf{AC}_i] .$$

For every $\kappa \in \mathbb{N}$, fix some r. Note that this also fixes the formula φ_i. Let Adv'_r be the adversary that is given $(\mathsf{pk}_i, \mathsf{vk}_i)$ emulates the experiment Exp'_r except that in the emulation of $\mathsf{Del.S}$ it uses $(\mathsf{pk}_i, \mathsf{vk}_i)$ given as input instead

of having Del.S sample these keys. Since Del.S runs in time $\mathrm{poly}(\bar{M}) = \mathrm{poly}(S, \kappa^d)$ and $\kappa^d \leq n^d = T(n) \leq \Lambda$, we have that Adv'_r is also of size $\mathrm{poly}(\Lambda)$ and for every $\kappa \in \mathbb{N}$:

$$\Pr_{\mathsf{Exp}'_r}[\mathsf{AC}_i] = \Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_i, x, \Pi) = 1 & (\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i) \\ \mathsf{QA.V}(\mathsf{vk}_i, x, \bar{\Pi}) = 1 & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi, \bar{\Pi}) \leftarrow \mathsf{Adv}'_r(\mathsf{pk}_i, \mathsf{vk}_i) \\ \Pi \neq \bar{\Pi} & \end{array}\right] .$$

Recall that the formula φ_i is given by:

$$\varphi_i\left(\mathsf{cf}_0, (\mathsf{cf}_j, \Pi_j, w_j)_{j\in[B]}\right) = \bigwedge_{j\in[B]} \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j-1}, \mathsf{cf}_j), \Pi_j, w_j) \ ,$$

and that K_i is the number of variables in the formula φ_i^{QA}. For every $j \in [B]$ let $\mathbf{S}_j$ be the set of K_i variables describing the values $(\mathsf{cf}_{j-1}, \mathsf{cf}_j, \Pi_j, w_j)$. Note that $\bigcup_{j\in[B]} \mathbf{S}_j = [M_i]$. By the Λ-unambiguity of the quasi-argument (Definition 8.4) for every $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_i, x, \Pi) = 1 & (\mathsf{pk}_i, \mathsf{vk}_i) \leftarrow \mathsf{QA.S}(\kappa, \varphi_i, K_i) \\ \mathsf{QA.V}(\mathsf{vk}_i, x, \bar{\Pi}) = 1 & (x, \Pi, \bar{\Pi}) \leftarrow \mathsf{Adv}'_r(\mathsf{pk}_i, \mathsf{vk}_i) \\ \Pi \neq \bar{\Pi} & \end{array}\right] \leq$$
$$\sum_{j^*\in[B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*} \ ; \ r) \\ \sigma_1 \neq \sigma_2 & \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

By the equations above:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] \leq \sum_{j^*\in[B]} \Pr_r\left[\begin{array}{l|l} x_1 = x_2 \neq \bot & \forall j \in [2] : \ (x_j, \sigma_j) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*} \ ; \ r) \\ \sigma_1 \neq \sigma_2 & \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{42}$$

Fix any $j^* \in [B]$. Let $(\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}, \Pi_{j^*}, w_{j^*})$ and $(\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}, \bar{\Pi}_{j^*}, \bar{w}_{j^*})$ be the values described by the assignments $\sigma_1 : \mathbf{S}_{j^*} \to \{0,1\}$ and $\sigma_2 : \mathbf{S}_{j^*} \to \{0,1\}$ respectively. By the local consistency of E_1 (Definition 8.2) for every $\kappa \in \mathbb{N}$:

$$\Pr_{(x_1,\sigma_1)\leftarrow\mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})}\left[x_1 = \bot \quad \vee \quad \begin{array}{l} \forall i \in \mathbf{S}_{j^*} \cap [N] : \ \sigma_1(i) = x_{1,i} \\ \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1 \end{array}\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

This implies:

$$\Pr_{(x_1,\sigma_1)\leftarrow\mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})}\left[x_1 \neq \bot \quad \to \quad \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Similarly:

$$\Pr_{(x_2,\sigma_2)\leftarrow\mathsf{E}_2^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})}\left[x_2 \neq \bot \quad \to \quad \varphi_i^{\mathsf{QA}}((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}, \bar{w}_{j^*}) = 1\right] \geq 1 - \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let $\mathsf{Exp}_r^{j^*}$ denote the experiment:

$$\forall j \in [2] : \ (x_j, \sigma_j) \leftarrow \mathsf{E}_j^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*} \ ; \ r) \ .$$

Then:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \end{array}\right] \leq \Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \\ \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1 \\ \varphi_i^{\mathsf{QA}}((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}, \bar{w}_{j^*}) = 1 \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{43}$$

By the construction of φ_i^{QA} if $(\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}, \Pi_{j^*}) = (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}, \bar{\Pi}_{j^*})$ and both $\varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1$ and $\varphi_i^{\mathsf{QA}}((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}, \bar{w}_{j^*}) = 1$ then $\sigma_1 = \sigma_2$. Additionally there exists $w_{j^*}, \bar{w}_{j^*}$ such that

$$\varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1$$

$$\varphi_i^{\mathsf{QA}}((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}, \bar{w}_{j^*}) = 1$$

only if:

$$\mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1$$

$$\mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1$$

Therefore:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \\ \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, w_{j^*}) = 1 \\ \varphi_i^{\mathsf{QA}}((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}, \bar{w}_{j^*}) = 1 \end{array}\right] \leq \tag{44}$$

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \mathsf{cf}_{j^*-1} \neq \bar{\mathsf{cf}}_{j^*-1} \end{array}\right] + \Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \mathsf{cf}_{j^*-1} = \bar{\mathsf{cf}}_{j^*-1} \\ \mathsf{cf}_{j^*} \neq \bar{\mathsf{cf}}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] \tag{45}$$

$$+ \Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}) = (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}) \\ \Pi_{j^*} \neq \bar{\Pi}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] . \tag{46}$$

In what follows, we upper bound each of the probabilities on the right. First note that:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}) = (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}) \\ \Pi_{j^*} \neq \bar{\Pi}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] \leq \Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \bar{\Pi}_{j^*}) = 1 \\ \Pi_{j^*} \neq \bar{\Pi}_{j^*} \end{array}\right] .$$

Let Adv' be the adversary that given $(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]})$ sampled from $\mathsf{Del.S}(\kappa)$ samples r and for $j \in [2]$ emulates $\mathsf{E}_j^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*})$ except that in the emulation of $\mathsf{Del.S}$ it uses the input keys $(\mathsf{pk}_{j'}, \mathsf{vk}_{j'})$ for every $j' < i$ instead of having $\mathsf{Del.S}$ sample these keys using the randomness r (note that r is not used anywhere else in the emulation). The adversary Adv' outputs $((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}, \bar{\Pi}_{j^*})$. We have that:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \bar{\Pi}_{j^*}) = 1 \\ \Pi_{j^*} \neq \bar{\Pi}_{j^*} \end{array}\right] = \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}]$$

so

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}) = (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}) \\ \Pi_{j^*} \neq \bar{\Pi}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] \leq \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] . \tag{47}$$

Let Adv'' be the adversary that given $(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]})$ sampled from $\mathsf{Del.S}(\kappa)$ samples r and for $j \in [2]$ emulates $\mathsf{E}_j^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*})$ except that in the emulation of $\mathsf{Del.S}$ it uses the input keys $(\mathsf{pk}_{j'}, \mathsf{vk}_{j'})$ for every $j' < i$ instead

of having Del.S sample these keys using the randomness r. Adv'' outputs $((\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*})$ or $\left((\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}\right)$ with equal probability. Since $\mathcal{M}$ is deterministic, if $\mathsf{cf}_{j^*-1} = \bar{\mathsf{cf}}_{j^*-1}$ and $\mathsf{cf}_{j^*} \neq \bar{\mathsf{cf}}_{j^*}$ then either $(\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}}$ or $(\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}}$. Therefore:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \mathsf{cf}_{j^*-1} = \bar{\mathsf{cf}}_{j^*-1} \\ \mathsf{cf}_{j^*} \neq \bar{\mathsf{cf}}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] \leq \Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \\ (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \vee (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right]$$

$$\leq 2 \cdot \Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_{i-1}, x, \Pi) = 1 & \left(\mathsf{pk}, \mathsf{vk} = (\mathsf{vk}_i)_{i\in[0,d]}\right) \leftarrow \mathsf{Del.S}(\kappa) \\ (\mathsf{cf}, \mathsf{cf}', B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} & (x = (\mathsf{cf}, \mathsf{cf}'), \Pi) \leftarrow \mathsf{Adv}''(\mathsf{pk}, \mathsf{vk}) \end{array}\right] .$$

By Claim 9.2:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \mathsf{cf}_{j^*-1} = \bar{\mathsf{cf}}_{j^*-1} \\ \mathsf{cf}_{j^*} \neq \bar{\mathsf{cf}}_{j^*} \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}), \Pi_{j^*}) = 1 \\ \mathsf{QA.V}(\mathsf{vk}_{i-1}, (\bar{\mathsf{cf}}_{j^*-1}, \bar{\mathsf{cf}}_{j^*}), \bar{\Pi}_{j^*}) = 1 \end{array}\right] \leq \mathrm{negl}(\Lambda(\kappa)) . \tag{48}$$

Next we prove the following claim.

Claim 9.8. *For every $j^* \in [B]$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j^*-1} \neq \hat{\mathsf{cf}}_{j^*-1} \end{array}\right] \leq \mu(\Lambda(\kappa))$$

where $x_1 = (\mathsf{cf}, \mathsf{cf}')$, $\sigma_1 = (\mathsf{cf}_{j^*-1}, \mathsf{cf}_{j^*}, \Pi_{j^*}, w_{j^*})$ and $\hat{\mathsf{cf}}_{j^*}$ is the unique configuration of $\mathcal{M}$ after $j^* \cdot B^{i-1}$ steps starting from cf. Similarly:

$$\Pr_{(x_2,\sigma_2)\leftarrow \mathsf{E}_2^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j^*})}\left[\begin{array}{l} x_2 \neq \bot \\ \bar{\mathsf{cf}}_{j^*-1} \neq \hat{\mathsf{cf}}_{j^*-1} \end{array}\right] \leq \mu(\Lambda(\kappa)) .$$

Proof. We focus on proving the claim for E_1. The proof for E_2 is analogous. By the non-signaling of E_1 (Definition 8.4) for every $j' \in [j^* - 1]$ and $\kappa \in \mathbb{N}$, since $\hat{\mathsf{cf}}_{j'}$ can be computed in time $\Lambda(\kappa)$ given cf we have that:

$$\left|\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'+1})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] - \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right]\right| \leq \mathrm{negl}(\Lambda(\kappa)) .$$

Therefore:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'+1})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] \leq \tag{49}$$

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} \neq \hat{\mathsf{cf}}_{j'-1} \end{array}\right] + \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} = \hat{\mathsf{cf}}_{j'-1} \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) . \tag{50}$$

If $\mathsf{cf}_{j'-1} = \hat{\mathsf{cf}}_{j'-1}$ and $\mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'}$ then $(\mathsf{cf}_{j'-1}, \mathsf{cf}_{j'}) \notin \mathcal{U}_n^{\mathcal{M}}$. Thus:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} = \hat{\mathsf{cf}}_{j'-1} \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] \leq \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i, \mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ (\mathsf{cf}_{j'-1}, \mathsf{cf}_{j'}, B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] .$$

By the local consistency of E_1 (Definition 8.2) for every $j' \in [j^* - 1]$ and $\kappa \in \mathbb{N}$:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[x_1 \neq \bot \quad \rightarrow \quad \varphi_i^{\mathsf{QA}}((\mathsf{cf}_{j'-1},\mathsf{cf}_{j'}),\Pi_{j'},w_{j'})=1\right] \geq 1-\mathrm{negl}(\Lambda(\kappa)) \ .$$

Therefore:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} = \hat{\mathsf{cf}}_{j'-1} \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] \leq \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1},(\mathsf{cf}_{j'-1},\mathsf{cf}_{j'}),\Pi_{j'})=1 \\ (\mathsf{cf}_{j'-1},\mathsf{cf}_{j'},B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Let Adv' be the adversary that given $(\mathsf{pk},\mathsf{vk}=(\mathsf{vk}_i)_{i\in[0,d]})$ sampled from $\mathsf{Del.S}(\kappa)$ emulates $\mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})$ except that in the emulation of Del.S it uses the input keys $(\mathsf{pk}_j,\mathsf{vk}_j)$ for every $j<i$ instead of having Del.S sample these keys using the randomness r. The adversary Adv' outputs $((\mathsf{cf}_{j'-1},\mathsf{cf}_{j'}),\Pi_{j'})$. By Claim 9.2:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[\begin{array}{l} \mathsf{QA.V}(\mathsf{vk}_{i-1},(\mathsf{cf}_{j'-1},\mathsf{cf}_{j'}),\Pi_{j'})=1 \\ (\mathsf{cf}_{j'-1},\mathsf{cf}_{j'},B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} \end{array}\right] =$$

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{vk}_{i-1},x,\Pi)=1 & (\mathsf{pk},\mathsf{vk}=(\mathsf{vk}_i)_{i\in[0,d]}) \leftarrow \mathsf{Del.S}(\kappa) \\ (\mathsf{cf}_{j'-1},\mathsf{cf}_{j'},B^{i-1}) \notin \mathcal{U}_n^{\mathcal{M}} & (x=(\mathsf{cf}_{j'-1},\mathsf{cf}_{j'}),\Pi) \leftarrow \mathsf{Adv}'(\mathsf{pk},\mathsf{vk}) \end{array}\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

Therefore:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} = \hat{\mathsf{cf}}_{j'-1} \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

Then by (49):

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'+1})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'} \neq \hat{\mathsf{cf}}_{j'} \end{array}\right] \leq \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j'})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j'-1} \neq \hat{\mathsf{cf}}_{j'-1} \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $j^* \leq B \leq \Lambda$ summing over $j' \in [j^*-1]$ we have that:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j^*-1} \neq \hat{\mathsf{cf}}_{j^*-1} \end{array}\right] \leq \Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_1)}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_0 \neq \hat{\mathsf{cf}}_0 \end{array}\right] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{51}$$

By the local consistency of E_1 (Definition 8.2) for every $\kappa \in \mathbb{N}$:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_1)}[x_1 \neq \bot \quad \rightarrow \quad \forall i \in \mathbf{S}_1 \cap [N] : \ \sigma_1(i) = x_{1,i}] \geq 1-\mathrm{negl}(\Lambda(\kappa)) \ .$$

Since $\mathbf{S}_1 \cap [N]$ consists of the indices corresponding to cf:

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_1)}\left[x_1 \neq \bot \quad \rightarrow \quad \mathsf{cf}_0 = \mathsf{cf} = \hat{\mathsf{cf}}_0\right] \geq 1-\mathrm{negl}(\Lambda(\kappa)) \ .$$

Thus by (51):

$$\Pr_{(x_1,\sigma_1)\leftarrow \mathsf{E}_1^{\mathsf{Adv}'_r}(\varphi_i,\mathbf{S}_{j^*})}\left[\begin{array}{l} x_1 \neq \bot \\ \mathsf{cf}_{j^*-1} \neq \hat{\mathsf{cf}}_{j^*-1} \end{array}\right] \leq \mathrm{negl}(\Lambda(\kappa)) \ .$$

This concludes the proof of Claim 9.8. □

By Claim 9.8:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\ x_1 = x_2 \neq \bot \quad \rightarrow \quad \mathsf{cf}_{j^*-1} = \hat{\mathsf{cf}}_{j^*-1} = \bar{\mathsf{cf}}_{j^*-1} \ \right] \geq 1-\mathrm{negl}(\Lambda(\kappa)) \ . \tag{52}$$

where if $x_1 = x_2$ then $\hat{\mathsf{cf}}_{j^*-1}$ is the unique configuration of $\mathcal{M}$ after $(j^*-1)\cdot B^{i-1}$ steps starting from $\mathsf{cf} = \bar{\mathsf{cf}}$. Combining (43) (44) (47) (48) (52) we have:

$$\Pr_{r,\mathsf{Exp}_r^{j^*}}\left[\begin{array}{l} x_1 = x_2 \neq \bot \\ \sigma_1 \neq \sigma_2 \end{array}\right] \leq \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] + \mathrm{negl}(\Lambda(\kappa)) \ . \tag{53}$$

Finally by (42) and since $B \leq T \leq \Lambda$:

$$\Pr_{\mathsf{Exp}^{\mathsf{Adv}}}[\mathsf{AC}_i] \leq B \cdot \Pr_{\mathsf{Exp}^{\mathsf{Adv}'}}[\mathsf{AC}_{i-1}] + \mathrm{negl}(\Lambda(\kappa)) \ .$$

This concludes the proof of Claim 9.7. □

□

References

[AKV04] Tim Abbot, Daniel Kane, and Paul Valiant. On algorithms for nash equilibria. Unpublished manuscript. http://web.mit.edu/tabbott/Public/final.pdf, 2004.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Boneh et al. [BRF13], pages 111–120.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BG20] Nir Bitansky and Idan Gerichter. On the cryptographic hardness of local search. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 6:1–6:29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017.

[BMW98] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In Kurt Rothermel and Fritz Hohl, editors, *Mobile Agents, Second International Workshop, MA'98, Stuttgart, Germany, September 1998, Proceedings*, volume 1477 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 1998.

[BPR15] Nir Bitansky, Omer Paneth, and Alon Rosen. On the cryptographic hardness of finding a nash equilibrium. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1480–1498, 2015.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090, 2019.

[CDT09] Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009.

[CHK+19a] Arka Rai Choudhuri, Pavel Hubáček, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Ppad-hardness via iterated squaring modulo a composite. *IACR Cryptology ePrint Archive*, 2019:667, 2019.

[CHK+19b] Arka Rai Choudhuri, Pavel Hub'avcek, Chethan Kamath, Krzysztof Pietrzak, Alon Rosen, and Guy N. Rothblum. Finding a nash equilibrium is no easier than breaking fiat-shamir. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1103–1114, 2019.

[CPW20] Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 220–246. Springer, 2020.

[DGP09] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 93–122, 2016.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003.

[EFKP19] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Continuous verifiable delay functions. *IACR Cryptology ePrint Archive*, 2019:619, 2019.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. On yao's xor-lemma. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*, pages 273–301. 2011.

[GPS16] Sanjam Garg, Omkant Pandey, and Akshayaram Srinivasan. Revisiting the cryptographic hardness of finding a nash equilibrium. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 579–604, 2016.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, New York, NY, USA, 2011. ACM.

[HY17] Pavel Hubáček and Eylon Yogev. Hardness of continuous local search: Query complexity and cryptographic lower bounds. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1352–1371, 2017.

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1115–1124, 2019.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In Boneh et al. [BRF13], pages 565–574.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494. ACM, 2014.

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 122–151, 2017.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10. IEEE Computer Society, 1990.

[LV20] A. Lombardi and V. Vaikuntanathan, 2020. Personal communication.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003.

[Pap94] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, 48(3):498–532, 1994.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 60:1–60:15, 2019.

[PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 283–315. Springer, 2017.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, USA, 1996.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, pages 1–18, 2008.